

SPATIAL PARTITIONING AND FUNCTIONAL SHAPE MATCHED DEFORMATION
ALGORITHM FOR INTERACTIVE HAPTIC MODELING

A dissertation presented to
the faculty of
the Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Doctor of Philosophy

Wei Ji

November 2008

© 2008 Wei Ji. All Rights Reserved.

This dissertation titled

SPATIAL PARTITIONING AND FUNCTIONAL SHAPE MATCHED DEFORMATION
ALGORITHM FOR INTERACTIVE HAPTIC MODELING

by

WEI JI

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Robert L. Williams II

Professor of Mechanical Engineering

Dennis Irwin

Dean, Russ College of Engineering and Technology

Abstract

Ji, WEI, Ph.D., November 2008, Computer Science

SPATIAL PARTITIONING AND FUNCTIONAL SHAPE MATCHED DEFORMATION
ALGORITHM FOR INTERACTIVE HAPTIC MODELING (161 pp.)

Director of Dissertation: Robert L. Williams II

This dissertation focuses on the fast rendering algorithms on real-time 3D modeling. To improve the computational efficiency, binary space partitioning (BSP) and octree are implemented on the haptic model. Their features are addressed in detail. Three methods of triangle assignment with the splitting plane are discussed. The solutions for optimizing collision detection (CD) are presented and compared. The complexities of these partition methods are discussed, and recommendation is made. Then the deformable haptic model is presented. In this research, for the haptic modeling, the deformation calculation is only related to the graphics process because the haptic rendering is invisible and non-deformable rendering saves much computation cost. To achieve the fast deformation computation in the interactive simulation, the functional shape matched deformation (FSMD) algorithm is proposed for the tangential and normal components of the surface deformation. The benefit of the octree model is taken for searching the deformed area. To simplify the boundary of the deformed region, an ellipse deformation map is created based on the experimental data. The Gaussian radial

interpolation (GRI) is developed for building this map which is pre-computed. The FSMD algorithm is tested and considered feasible for the virtual haptic back (VHB) project. Moreover, some recommendations for the future work are suggested.

Approved: _____

Robert L. Williams II

Professor of Mechanical Engineering

Acknowledgments

I would initially like to thank my advisor Dr. Robert Williams for his constant support and insightful guidance for my master and doctoral studies throughout the past five years. He has provided me excellent opportunities to work with the virtual haptic back (VHB) research team. His theoretical knowledge helped me to solve many problems in my research work. I am deeply impressed by his patience and kindness. I would also like to express my thanks to Dr. John Howell, my college representative, for his support, many valuable suggestions on my research and his daily organization on the VHB project. I love working with him because of his enthusiasm for research and humor.

I would additionally like to thank the members of my committee, Dr. Jundong Liu, Dr. Chang Liu and Dr. Frank Drews, for their instructions and time on my research. The knowledge gained from them will be my lifetime fortune. I, of course, am grateful to Dr. Sergio Lopez-Permouth for accepting to be my college representative and reviewing my dissertation. Many thanks go to the members of the VHB team: Bob Conatser, Ernur Karadogan, Meng-Yun Chen, Bajaj Kapil, David Noyes and Dr. Janet Burns.

Finally, I express my deep thanks to my wife, Yan, and my two-year-old daughter, Eulina, for their love and support. Also I especially thank my parents and my sister for their encouragements.

Table of Contexts

Abstract.....	3
List of Tables.....	9
List of Figures.....	10
List of Abbreviations.....	13
1. Introduction.....	14
1.1 Background of Deformation in Virtual Reality (VR).....	14
1.2 The Need for Deformation in the Virtual Haptic Back (VHB) Project.....	16
1.3 Haptic and Graphic Renderings	18
1.3.1 Haptic Rendering	18
1.3.2 Graphic Rendering.....	21
1.4 Literature Review	23
1.4.1 Deformation Algorithms	23
1.4.1.1 Shape Interpolation (SI)	23
1.4.1.2 Free-Form Deformation (FFD)	24
1.4.1.3 Skeleton-Subspace Deformation (SSD)	26
1.4.1.4 Mass-Spring Model (MSM)	27
1.4.1.5 Finite Element Method (FEM)	29
1.4.1.6 Collision Detection (CD).....	30
1.4.2 Space Partition Methods	31
1.4.2.1 Binary Space Partitioning (BSP).....	31
1.4.2.2 Octree Partition.....	32
1.4.3 Major Area Discussions of Deformation Modeling.....	32
1.5 Research Objectives	37
1.6 Dissertation Organization.....	38
2. Related Research Work	41

	7
2.1 Haptic Model, Stiffness Map and Stereo Viewing System.....	41
2.2 Displacement Driven Force and Proxy Sphere.....	45
2.3 Non Deformable Modeling in Haptic Rendering.....	47
2.4 Friction Effect in Tangential Deformation	49
2.5 Utility Libraries and Platform	51
3. Spatial Partitioning for Fast Rendering.....	52
3.1 Data Structure of the Haptic Model.....	54
3.2 Rendering Performance Measurement	56
3.3 Binary Space Partitioning (BSP) for Haptic Rendering.....	57
3.3.1 Construction of BSP Tree	57
3.3.2 Triangles Distribution	62
3.3.3 Haptic Detecting Box (HDB).....	65
3.3.4 Performance and Complexity Analysis.....	70
3.4 Octree for Haptic Rendering	75
3.4.1 Haptic Triangles Detection	78
3.4.2 Complexity Analysis.....	79
4. Functional Shape Matched Deformation	84
4.1 Deformation Shape on Elastic Material	85
4.2 Normal Deformation	90
4.2.1 Normalized Gaussian Curve Mapping Array.....	94
4.3 Tangential Deformation.....	99
4.4 Point Mapping to Ellipse Plane.....	101
4.5 The Neighbor Nodes Ring Algorithm	106
4.6 Maximum Deformation.....	111
4.7 Visualization of Surface Strain.....	112
5. Defined Deformation Boundary Shape.....	115
5.1 Deformation Shape Study of Human Back and Simplification Method	116

	8
5.2 Interpolation of the Deformation Ellipses	122
5.2.1 Problem Statement	122
5.2.2 Radial Interpolation Based on Gaussian Curve	124
5.3 Ellipse Conversion from Deformation Shape Map to 3D Model.....	138
6. Conclusions and Future Work.....	140
6.1 Summary and Conclusion	140
6.2 Recommendations for Future Work	142
References.....	143
Appendix A. Models for the Research.....	157
Appendix B. Haptic Rendering Performance in Octree Model	159
Appendix C. Back Deformation Ellipses Measurement Data	160

List of Tables

Table 3.1	Haptic and Graphic Rendering Rates on Different-sized Models	53
Table 3.2	Haptic Rendering Performance of the Binary Tree Model	71
Table 3.3	The Worst w Related to the Binary Trees in Different Heights	74
Table 3.4	The Worst w Related to the Octrees in Different Heights	80
Table 4.1	Features of the Existing Deformation Algorithms	85
Table 4.2	Comparison of GC, Cos, and Log Functions	91
Table 5.1	Major Axes of the Ellipses at Key Points	135

List of Figures

Figure 1.1	Virtual Haptic Back (VHB) Project.....	17
Figure 1.2	Haptic and Graphic Renderings.....	19
Figure 1.3	Back Skin Deformation in Cross Section.....	21
Figure 1.4	Shape Interpolation (SI)	24
Figure 1.5	Free-Form Deformation (FFD), [Sederberg, 2004].....	25
Figure 1.6	Collapsing Joint in SSD [Lewis et al., 2000]	27
Figure 1.7	A Portion of a MSM	28
Figure 1.8	FEM Applied on a Deformed Metal Part	29
Figure 2.1	VHB Models with Skeleton (left) and Skin-Textured (right).....	42
Figure 2.2	The Original High Resolution Back Surface Built by 3D Camera	43
Figure 2.3	Statistic Stiffness Map of Human Back.....	44
Figure 2.4	3D Stereo Viewing System for the VHB Project.....	45
Figure 2.5	Force Generation of the Haptic Interface Depends on the Displacement	46
Figure 2.6	Deformation in Haptic Rendering Changes the Stiffness.....	47
Figure 2.7	Friction Anchor.....	50
Figure 3.1	Plots for Table 3.1.....	53
Figure 3.2	The Back Mesh.....	55
Figure 3.3	Performance Measurement of the Haptic and Graphic Renderings	57
Figure 3.4	Binary Partition Aligned x, y and z Axes on the Model Mesh.....	59
Figure 3.5	BSP Tree of the Back Model	60
Figure 3.6	Splitting Planes and Leaf Boxes of the BSP Tree	62
Figure 3.7	Three Cases of a Splitting Plane Crossing the Triangles.....	63
Figure 3.8	Leaf Borders of the Case C	65
Figure 3.9	Haptic Triangles When the Proxy is Closing to the BSP Bunny.....	69

	11
Figure 3.10	Plots for Table 3.2..... 71
Figure 3.11	Traversed Nodes in the Worst Case of w 74
Figure 3.12	Octree Partition on the AABB 76
Figure 3.13	Leaf Boxes of the Octree on the Haptic Models 76
Figure 3.14	Triangles Intersection with Bounding Box of the Node 78
Figure 3.15	HDB is Closing to the Octree Back Mesh, Few Leaves Rendered 79
Figure 3.16	Haptic Triangles Drawing without Haptic APIs Involved..... 82
Figure 3.17	Haptic Rendering Rate in Untouch Condition..... 82
Figure 3.18	Haptic Rendering Rate in Touch Condition 82
Figure 4.1	Deformation Cross Section on Soft Object 86
Figure 4.2	Deformation Due to a Finger is Similar to a Round Rigid Object 86
Figure 4.3	Flipped Gaussian Curve in Horizon 88
Figure 4.4	Modified Gaussian Curve for Sharp Object 88
Figure 4.5	Normal and Tangential Deformation Combination 89
Figure 4.6	Deformation Region and Contact Point 92
Figure 4.7	Normal Deformation of Back Mesh Mapping with a Gaussian Curve 94
Figure 4.8	Half Gaussian Curve and Its Storage Array $M[i]$ 95
Figure 4.9	Smoothness in Different Array Sizes 97
Figure 4.10	Gaussian Curve Array Mapping 98
Figure 4.11	Tangential Deformation in the Circle 99
Figure 4.12	Tangential Deformation of Back Mesh Mapping on Gaussian Curve 101
Figure 4.13	Point Projection on Deformation Shape Plane 102
Figure 4.14	Point Location in Deformation Boundary Ellipse 104
Figure 4.15	Neighbor Nodes Deformation Rings 107
Figure 4.16	Deformation Combined with Normal and Tangential Components 110
Figure 4.17	Average of the Neighboring Points Color Mapping For Visualization 113
Figure 4.18	Strain Visualization of the Surface Deformation 114

Figure 5.1	Human Back Muscles and Seven Key Points.....	116
Figure 5.2	Two Deformation Contours at P_2	118
Figure 5.3	The Deformation Ellipses at the Key Points	121
Figure 5.4	Key Deformation Ellipses in the Uniform Deformation Area.....	123
Figure 5.5	Effect Region Circle and Gaussian Decline Trend Curve	126
Figure 5.6	Effect Depending on the Distance.....	127
Figure 5.7	Gaussian Curve Mapping for the Effect Level.....	128
Figure 5.8	Effect Ellipses in a Square Area	135
Figure 5.9	Results do not Fit the Key Points	136
Figure 5.10	Interpolated Results Satisfies All the Key Points.....	136
Figure 5.11	Map of Major Axis on Real Back Based on 7 Key Points	137
Figure 5.12	Projection from Model to Deformation Map Plan	139

List of Abbreviations

AABB	Axis-Aligned Bounding Box
CD	Collision Detection
EA	Effect Radius
FEM	Finite Element Method
FFD	Free-Form Deformation
FPS	Frames Per Second
FSMD	Functional Shape Matched Deformation
GC	Gaussian Curve
GRI	Gaussian Radial Interpolation
HDB	Haptic Detecting Box
HIP	Haptic Interface Point
HT	Haptic Triangles
IDSD	Interpolation of Defined Shape Deformation
MSM	Mass-Spring Model
RBF	Radial Basis Function
SI	Shape Interpolation
SSD	Skeleton-Subspace Deformation
SCP	Surface Contact Point
TPS	Thin Plate Spline
VHB	Virtual Haptic Back

1. Introduction

1.1 Background of Deformation in Virtual Reality (VR)

This research focuses on interactive deformable object modeling in computer graphics and its implementation in haptics-augmented virtual reality (VR). Deformable modeling has been widely used in engineering design, medical simulation, animation, movie special effects, and game design, among other applications. In the real world, deformation, break, and union are the most common types of change seen daily. Deformation algorithms in geometry modeling must be developed for improved realism in VR, although this requires much more computation than animation of rigid objects. Many deformation algorithms have been developed for different purposes, such as the key frame interpolation that is used in 3D animation and movie special effects. The finite element method (FEM) is used in engineering bridge design and dynamic analysis. The spring mass method is used in surgery simulations. Low resolution modeling is used in 3D games. But so far, there is not a universal method that is suitable for most applications. Developers have to study the principles and properties of deformable objects in the real world. Due to the limitations of the computer and interactive devices, deformation simulation needs to be optimized to get a good balance between computational efficiency and graphical performance. For example, in engineering design, the user usually pursues the most accurate results (which requires more computation) more than simple structured

modeling (which would save computation time). So FEM is widely used in engineering and science. In the movie industry, animators prefer to see the desired postures of the actors. So they will spend several hours to render only one minute of video by using the key frame edition method. But for the real-time or interactive applications, the efficiency of computation is the main factor affecting the final result. The doctor really needs instant response from the virtual organ he/she interacts with on a simulator. The current main deformation algorithms will be discussed later.

Haptics is the science of incorporating the sense of touch and control into computer applications through force (kinesthetic) or tactile (touch) feedback. Haptics is one of the most important human sensory modes. Special input/output devices called haptic interfaces are used in haptically-enabled applications. Users can manipulate and feel virtual 3D objects through haptic interfaces. Including haptic feedback significantly improves the realism of VR applications, in medical training and in flight simulation for example. Although the latest haptic hardware in the market has limitations (such as small work space, small forces, and problems with rigidity) haptics enhances the realism of VR simulations with graphics. Surgery and injection simulations are example applications of haptics. The models of these applications are usually based on soft organs which deform under the touch via the haptic interface. The haptic interface sends its position to computer and is able to generate the vector forces bound by the capabilities of the device.

The graphical model on the screen presents the deformation, showing how far the haptic effector pressed or dragged from the model surface.

1.2 The Need for Deformation in the Virtual Haptic Back (VHB) Project

The Virtual Haptic Back (VHB) project at Ohio University [Howell et al., 2007] is trying to achieve smooth surface deformation graphics with a high resolution haptic model. The purpose of the VHB project is to develop a realistic haptic/graphical model of the human back that can be used for palpation (diagnosis through touch) in medical training at the Ohio University College of Osteopathic Medicine (OUCOM) for doctors and students. In this haptic application two commercial PHANTOM 3.0 haptic interfaces from SensAble Technologies Inc. are used to send the force feedback to the user, permitting palpation by force feedback with two fingers of a life-sized virtual human back (see Figure 1.1). The movement of the back skin, by exertion of palpatory force from the user should be reflected graphically as a surface deformation. Physical properties of the back, e.g. spring constants of the surface, were chosen based on feedback from physicians experienced in palpatory diagnosis and more recently in physical measurements [Williams et al., 2007]. To get the most realistic simulation, we are interested in force feedback performance (which depends on 1) the knowledge of the physical properties of the human soft tissue, which we can obtain, 2) the simulation methods and 3) the performance of the haptic hardware) on the user's fingertip and the

3D graphical behavior of the skin model (because the surface is the only visible part).



Figure 1.1 Virtual Haptic Back (VHB) Project

For the haptic model, 3D human models can be built of bones, soft tissue (muscle layers, connective tissue, and adipose layers), and skin. On the haptic model, model surface skin deformation should happen via force interaction through the haptic interface. The action of this deformation also depends on the pose of the skeleton and on the model's soft tissue stiffness properties. Several fundamental theories have been developed for 3D object deformation. Mesh deformation is a popular method for computer modeling [Watt 2000, Shi et al. 2006]. Many techniques have been developed to help designers to deform body shapes, such as shape interpolation [Wolberg, 1998], free-form deformation [Sederberg, 1986], and a skeleton-based method [Magnenat-Thalmann et al., 1988]. The last one uses a 'skeleton', in which two or more bones meet at each joint, to control shape deformation. This allows for intuitive control,

naturally describing deformation in many objects. However, traditional skeleton-based methods are often criticized for requiring a more tedious process of weight selection to obtain satisfactory results. And so far there is no criterion for weight selection suitable for most cases.

The practical VHB interactive system has been developed using two PHANTOM[®] 3.0 haptic interfaces. The author mostly used a portable haptic device called the Omni (also from SensAble Technologies Inc.). The results from the Omni can be transferred to the PHANTOM[®] 3.0, which is bigger and has higher force capacity than the Omni. This haptic system from SensAble works on a multi-thread platform where haptic rendering and graphic rendering are executed as individual processes. In each process, the mesh of the model's surface is drawn once. The mesh is used for collision detection and force generation handled by the haptic APIs in haptic rendering. But this rendering is invisible on the screen. Immediately after that, the whole model is drawn again on the screen in the graphical rendering process. So, the model is rendered twice in each system loop but only the graphical process is visually displayed.

1.3 Haptic and Graphic Renderings

1.3.1 Haptic Rendering

A point-based haptic rendering technique is used in which the PHANTOM's stylus endpoint (haptic interface point, HIP) is modeled as the probing object. During the course

of haptic simulation, a collision detection algorithm checks continuously if the HIP collides with any virtual objects. For an object modeled with a triangular mesh, the three vertices of the collided polygon are detected after collision and assigned as the stimulated nodes for the deformable simulation. To constrain the HIP on the object surface, the collided polygon is displaced according to the indentation made by the HIP. The associated reaction force is evaluated by the vector sum of spring forces at the three stimulated nodes. See Figure 1.2 for an overview of the servo loop of the haptic (force feedback) and graphic (surface deformation) renderings.

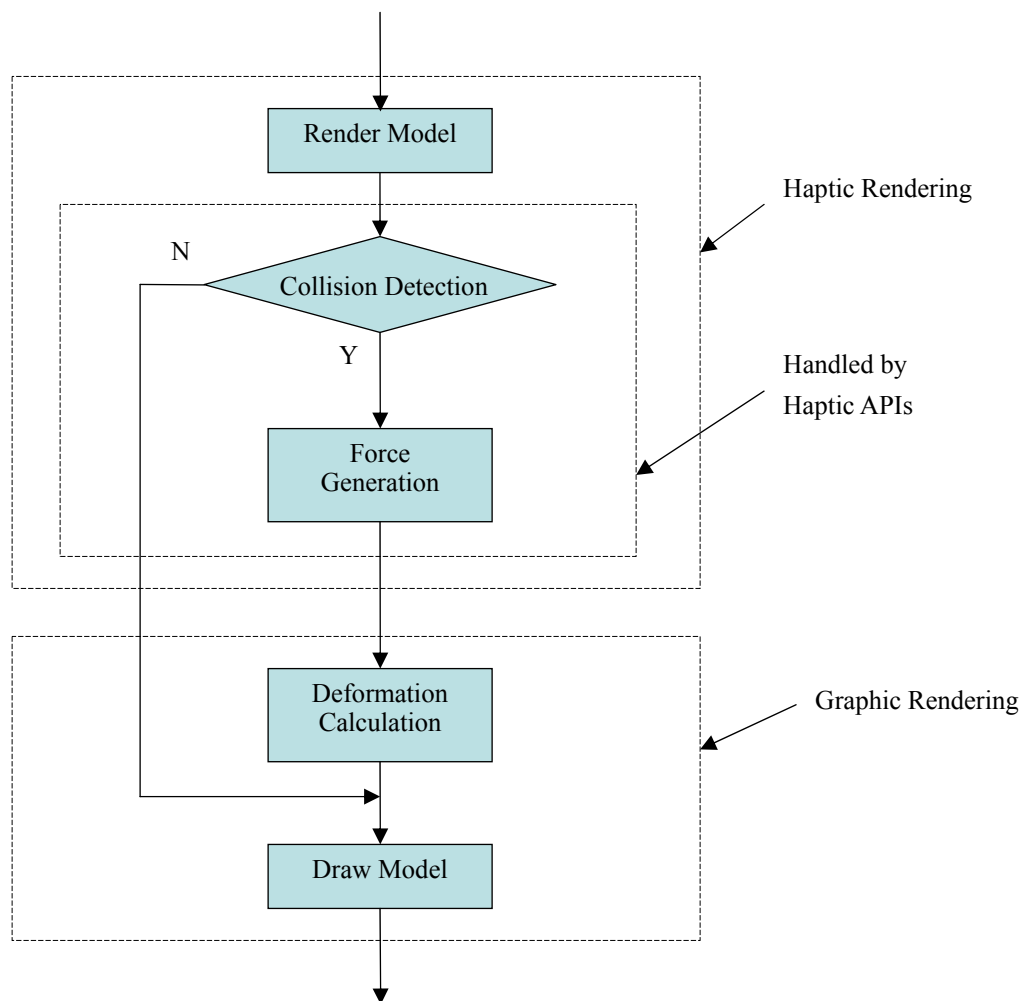


Figure 1.2 Haptic and Graphic Renderings

Since the VHB is an interactive simulation, the haptic and graphic responses become slow with an increase in complexity of the model, such as the increase of number of the triangles (or polygons). This performance can be shown by how many frames per second (FPS) the computer can render. Although we can use a better computer to improve the performance of the system, fast algorithms are always a more practical and reasonable solution than updating computer hardware. In haptic rendering, the result is invisible. To shorten the rendering time, we considered that only a partial mesh around the HIP is drawn instead of the whole model because only the neighboring triangles of the touch point are involved in collision detection and force generation. The rendered area becomes a small region which contains much fewer triangles than the whole model mesh. Also this area can follow the moving of the touch point dynamically. To achieve this goal, the model is divided into several small parts by the space partition method. Only the nearest parts of the touch point are rendered for the haptic process. Furthermore the nearest node to the touch point needs to be identified in the deformation calculation by a distance comparison to the touch point position. So the local computation of finding the nearest node is much more efficient in the nearest divided space than using global comparison. The binary space partition (BSP) and octree is implemented on VHB. These algorithms are described later.

1.3.2 Graphic Rendering

The whole model is drawn on the screen in this process for visualization. In the palpation simulation, the posture of the model does not change. So when the user presses the model at the touch point, only the local mesh deforms under the external force. On the VHB model, the deformation is not implemented in haptic rendering that is invisible in the simulation. The reason will be detailed later in the related work chapter. Figure 1.3 shows the deformation under the finger force in the cross section of back skin. This research has focused on simulating such deformation.

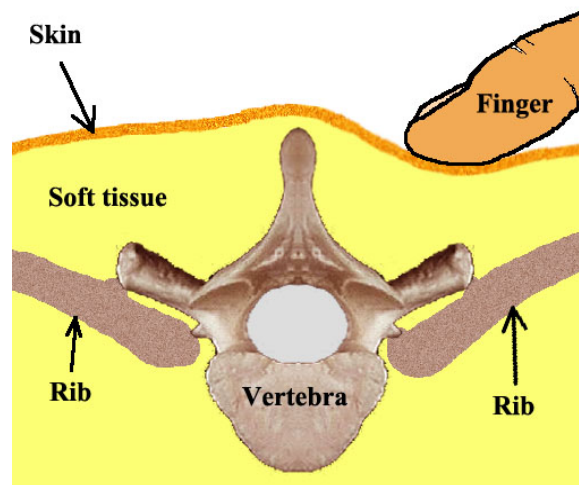


Figure 1.3 Back Skin Deformation in Cross Section

Many deformation algorithms have been assumed for geometric modeling to date. The advantages and drawbacks of them will be discussed in the literature review. In the VHB, we do not use a finite element method (FEM), free form deformation (FFD), or

mass-spring method (MSM) because they are computationally expensive for real-time deformable modeling. The functional shaped matched deformation (FSMD) algorithm has been established for our haptic model. In this method, we do not need to define the complex boundary conditions like the deformation in FFM or MSM. We studied the characteristics of deformation on real human backs and use a simplified elliptical shape to define the deformation region. The deformation is simulated by combining with two components in the normal and tangential directions on the touching surface. A functional shape Gaussian curve is used in both of normal and tangential deformations simulation.

In the graphics rendering, the same intent of saving computation time was considered for detecting the deformed vertices. We use a neighbor-expanding algorithm to avoid useless computation on the whole model. The deformation calculation is executed from the touch point toward the model edges until no position change occurs. This algorithm will be described in detail in Chapter 4.

Living tissues are composed of materials with different physical properties. For small deformations, they can be considered as Hookean materials for which linear elastic approximation is applicable [Chen et al., 2006]. The MSM and FEM are two common physics-based modeling techniques. The discrete MSM requires less computation and has easier implementation than FEM. In FEM, rigorous mathematical analysis based on continuum mechanics is applied to model the mechanical behaviors, which offers better simulation realism than other existing deformation methods. However, the computational

complexity of FEM might cause problems to interactive (real-time) VR applications in which high refresh rates of 30 Hz and 1000 Hz are demanded for visual and haptic renderings, respectively [Choi et al., 2003]. It is therefore critical to maintain a balance between the accuracy and computational complexity in improving the level of realism. On the other hand, to popularize the usage of VR-based deformable simulation, it is advantageous to enable economical and simple implementation in a generic computing environment with standardized haptic interfaces.

1.4 Literature Review

Our deformable haptic model is related to geometry deformation methodologies including efficient searching and interpolation algorithms. The following subsections will briefly describe the fundamental algorithms of geometric deformation, space partition methods, and interpolation algorithms.

1.4.1 Deformation Algorithms

1.4.1.1 Shape Interpolation (SI)

Shape Interpolation (SI) is also named shape blending and multi-target morphing. It probably is the most widely used method to shape deformation by 3D animator [Maestri, 1999]. In this procedure, the user needs to manipulate key poses or shapes (also

called key frames), and then an interpolation algorithm (such as scattered data interpolation) generates the interval shapes (or frames) automatically (see Figure 1.4). SI does not depend on the physical properties of the deformed object. To get smooth and detailed deformations, the adjunct key shapes cannot be much different. That means more key shapes are need to be edited manually to ensure the interpolation results are closer to the realism we desire. In interactive deformation, the motions of the user cannot be predicted. So it is a lot of work to make a large number of key shapes to cover the possible key frames.

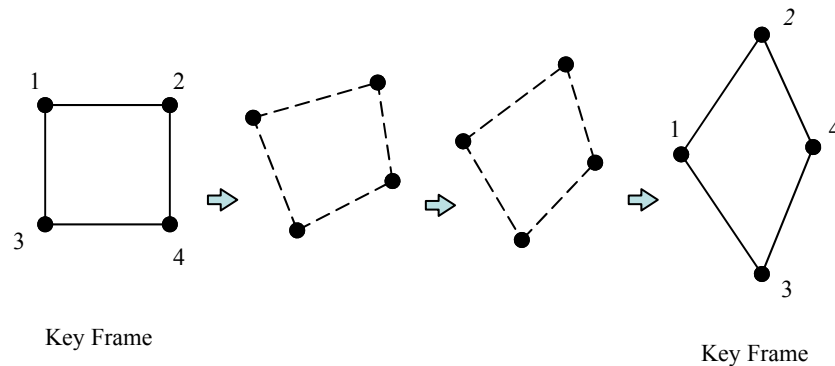


Figure 1.4 Shape Interpolation (SI)

1.4.1.2 Free-Form Deformation (FFD)

Free-Form Deformation (FFD) is a well known technique for carrying out deformation. It has been also widely used in commercial software such as 3D Studio Max and Maya. In the classic FFD, a shape is placed in an elastic control lattice space, such as a Bezier volume, or a more general lattice, then deforms the volume by moving the

control points. As a result, the shape within the lattice space is deformed. This method can be used to many different graphical models which contain nodes, polygons, splines, and implicit surfaces. Since the 3D FFD is similar to the 2D case, only the 2D case of FFD is presented here.

2D FFD is a map from $R^2 \rightarrow R^2$ in mathematics. It defines the new location for each point in a predefined region, usually rectangle. Any line or curve in that region is then changed. Evaluating the moved point is simply way of solving the Bezier equation for the deformed set of control points. In Figure 1.5, the FFD defines nine control points in a rectangle. The original shape (the circle) is presented in the left, and the right picture shows the moved positions of the control points and the shape change of the circle. The grid in the rectangle is drawn for the visualization of the space change. Anything inside of the predefined, undeformed region will be distorted with the shape change of the region.

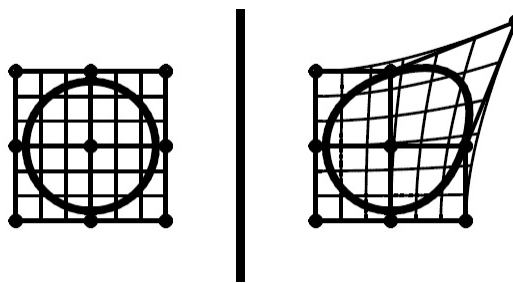


Figure 1.5 Free-Form Deformation (FFD), [Sederberg, 2004]

If a part of a shape lies inside the FFD region, only this part will be deformed corresponding to the deformation of the FFD region. The part outside the grid region will keep its original shape.

1.4.1.3 Skeleton-Subspace Deformation (SSD)

Skeleton-Subspace Deformation (SSD) is an important deformation algorithm and applied to many interactive applications. It is also standard in video games and virtual environments because it produces reasonably good results and is easy to understand and computationally efficient to implement.

The SSD algorithm combines the geometry surfaces or internal vertices with the movable skeleton by assigning a set of bones (and the associated transformation matrices) and a weight for each influence to each vertex. The position of a moved vertex is obtained by transforming this vertex rigidly related to each of its effect weights and then using the weights as coefficients to compute a linear combination of these transformed positions as the final position. So all control jobs of the deformation of the linear blend surface are the adjustment on the influences and weights of each vertex [Mohr et al., 2003].

The SSD has two main limitations [Lewis et al., 2000]. The major one is that the deformation is restricted to the defined subspace. If some bones are out of the subspace, they cannot affect the deformation although they should do. Figure 1.6 shows the

‘collapsing joint’ problem that is an extreme case of simulating the twist of a human forearm after a rotation of 180 degrees. The second difficulty with SSD is that, unlike SI, it does not allow the user to manipulate points directly. Users have to directly or indirectly edit the meshes instead of the weights.

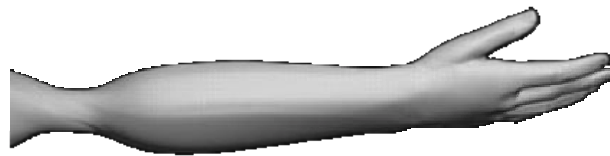


Figure 1.6 Collapsing Joint in SSD [Lewis et al., 2000]

Skinning techniques have been introduced to solve these problems. Many of them can be considered as corrections to SSD [Allen, 2002]. One method is combining the rigid transformation or linear blend with radial basis example interpolation [Hsu, 1992]. The Eigen-Skin method [Kry et al., 2002] presented a different example-driven linear blend skin correction technique.

1.4.1.4 Mass-Spring Model (MSM)

The Mass-Spring Model (MSM) is a physics-based technique for modeling deformable objects [Gibson et al., 1997]. In this method, an object is built in a lattice structure. Each point is assigned a mass and connected with other points by springs (see

Figure 1.7). The spring is massless. k is the spring constant and m is the mass at that vertex. The spring forces are usually linear via Hooke's Law, but nonlinear springs or springs-dampers can be used to simulate soft tissues. In a dynamic system, the movement of a single mass of this MSM is driven by the connected springs under Newton's Second Law.

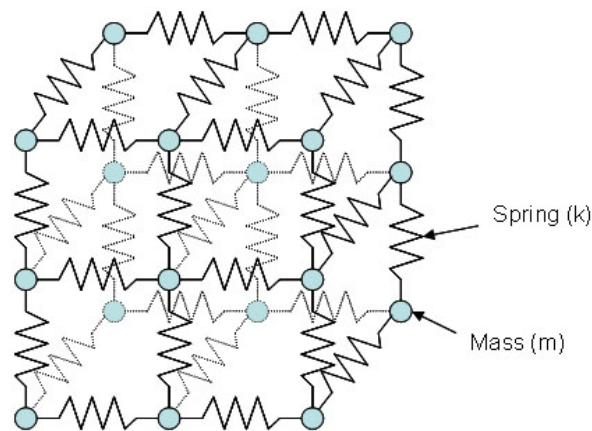


Figure 1.7 A Portion of a MSM

This method has some drawbacks. This is a discrete structure model with simplified approximation of the true physical properties which occur in a continuous object. The springs directly affect the vertex movement. But the proper values for the spring constants are always not easy to be obtained from the desired materials. Moreover, certain constraints of the springs and points cannot be expressed naturally in the model. For instance, undeformable objects or thin membranes are difficult to be considered as

MSM. Therefore, additional springs might be added to for adjustment. That will result in more computational cost.

1.4.1.5 Finite Element Method (FEM)

The Finite Element Method (FEM) is a standard numerical method to solve partial differential equation problems in many fields. In FEM, the model is divided into small simple elements (see Figure 1.8). Each element the some equilibrium equations related to a solution function. The sum of the solutions (approximation) of each element is evaluated and modified to match the boundary conditions in a defined tolerance. The result of the point within the element can be obtained by interpolation. The precision of the result depends on the number of the divided elements and the tolerance.

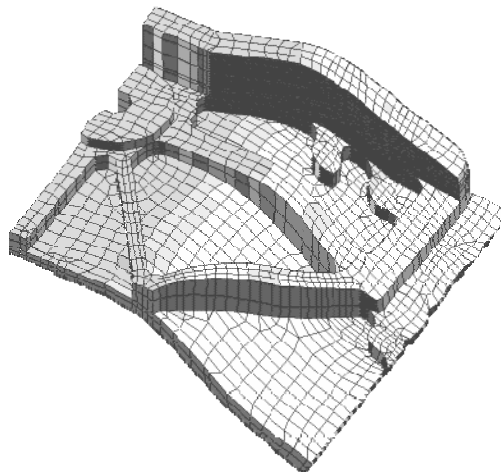


Figure 1.8 FEM Applied on a Deformed Metal Part

(Picture is from www.e-sac.org)

FEM is widely used in CAD and engineering analysis. In computer graphics, FEM can be used to find an approximation for a continuous function that satisfies some equilibrium expressions such as the deformation equations [Gibson et al., 1997]. Normally, FEM is applied to small deformation materials such as metals, concrete, etc. The max deformation is usually limited to less than 1% of the object dimensions. But in soft materials such as human tissue, the deformation might be more than 100% of the object size. In that case, additional deformation assumption needs to be supposed [Bathe, 1996].

In the real-time simulations, the use of FEM has been proved to be limited because of its expensive computational cost [Blemker et al., 2003].

1.4.1.6 Collision Detection (CD)

Collision detection (CD) has become a fundamental problem in computer animation, physics-based modeling, geometric modeling and robotics [Lin et al., 1998]. In reality, an object often contacts or detaches from many other types of objects such as rigid objects, deformable objects, fluids, etc. The intersection problems at the boundaries where two objects meet affect their shape and movement. Efficient collision detection algorithms are necessary for deformable objects. Some techniques have taken initial steps toward solving such problems. However, faster and more feasible algorithms are still expected for different applications.

In the VHB simulation, the collision thread (haptic API from the manufacturer) of the haptic device handles collision detection during interactions. This thread is executed at 100 Hz, less than the force rendering rate of 1000 Hz [SensAble Technologies Inc., 2005]. We implemented few methods to optimize the computation of CD.

1.4.2 Space Partition Methods

Our haptic model is based on spatial data structures. The data traversal efficiency significantly affects the performance (FPS) of the renderings. The linear storage method, such as array or linear link, is not adaptive to the different models and might consume a lot of memory. Moreover, array is not efficient in spatial data searching. The binary space partitioning (BSP) tree and octree are two data structures with subdividing the scene to increase the searching efficiency.

1.4.2.1 Binary Space Partitioning (BSP)

BSP works recursively to split a space into two by the hyperplane until the partitioning satisfies the user-defined requirements. This method not only can be used in haptic and graphics renderings but also is a fast algorithm to access a small part of a big volume mesh in collision detection [Chrysanthou, 1992]. BSP can improve the complexity from $O(n)$ to $O(\log n)$ for touched point detection.

Normally, there two types of BSP, called axis-aligned and polygon-aligned. The axis-aligned BSP requires less computation than polygon-aligned BSP.

1.4.2.2 Octree Partition

Gervautz and Purgathofer [1988] first proposed the octree for encoding color data. Then octree has been used widely in geometry modeling and ray tracing. Its structure is similar to the axis-aligned BSP. Octree subdivides the space in to $2 \times 2 \times 2$ boxes equivalently along the coordinate x , y , and z axes and continues to partition the child node with the same manner as its parent recursively. So each parent node has 8 child subspaces with same volume. Yau and Tsou [2006] applied octree to avoid a large number of voxels to their virtual dental training system with a haptic device. We have built the octree in haptic modeling and for efficient contact-point detection. More details of this procedure will be described and discussed in the next chapter.

1.4.3 Major Area Discussions of Deformation Modeling

Many improved deformation algorithms have been presented for various different purposes related to the human body, organs, muscles, skin or hair. Some articles discussed local deformations and some focused on full body (global) deformation behaviors. With the haptic simulation involved in VR, the deformation during the haptic

interaction becomes more complex. The deformation can be driven by internal factors such as skeleton changes or by the external factors such as the force of touching.

Allen and Curles [2002] introduced a sample-based method to calculate a skeleton body deformation. Their example data consists of a range scan of the human body in different poses. They built a mutually-consistent parameterization of all scans using a possible subdivision surface template and combined the range scans using the k -nearest neighbors interpolation in pose space. But their method does not encompass dynamical behaviors or deformation due to collisions.

Magnenat-Thalmann et al. [1988] and Komatsu [1988] presented a human body deformation method driven by underlying skeletal movement. They defined the size and shape of deformation for each of these approaches. Magnenat-Thalmann's work focused on developing the algorithms for the different joints of the human hand. Komatsu presented his work on the elbow and explained how the skin wrinkle on the critical side can be obtained by a proper manipulation of the surface control points. This algorithm did not cause the phenomenon "collapsing elbow" that happens with the SSD.

Yan et al. [2006] described a mesh deformation method that combines the skeleton-based method and the simplex transformation method, with two main differences from traditional skeleton-based methods. First, they used the skeleton to drive the transformation of simplexes rather than vertices as in previous methods. Second, they avoid using any weights, yet the approach still gives high-quality results. Their approach

can be applied to 2D or 3D meshes. Their inputs are the initial mesh, the initial skeleton (a set of straight line segments connected together at joints), and the changed skeleton. The output is the deformed mesh.

Lewis et al. [2000] developed a pose space deformation algorithm to generalize and improve both SI and skeleton-driven deformation. In this technique, several deformation types can be represented as mappings from a pose space, which is specified by an underlying skeleton or a simple parametrical space system, to new locations in the model local coordinate space. This algorithm improves the expressive power and allows user to manipulate directly the model to desired shapes. It can be applied to body deformation and facial animation for entertainment, 3D games, and other applications which need direct sculpting deformations or requires interactive synthesis of a deforming model.

Tagawa [2006] proposed a method to solve the problem of degrees of freedom of interaction and to apply the record reproduction approach to deforming interaction. In their approach, the characteristic of deformation was described by a set of data which they called impulse response deformation model. They assumed that the resulting deformation has linearity regarding input forces. Hence the deformation is obtained by computing convolution of the sequence of impulse forces and the impulse response deformation model. As they described in their paper, that approach enables haptic

interaction with dynamic deformable object model that is too complex to be solved by a common FEM.

Koruda et al. [2003] demonstrated a simulation of organ-to-organ interaction which is indispensable for practical and advanced medical VR simulators such as surgery and indirect palpation. They gave a method to represent interaction between elastic objects, i.e. organs, in a medical VR simulation. They showed a model defined displacements of colliding elements based on temporary surface forces caused by the temporary displacements so that the model produces accurate deformation and force feedback considering collisions of objects as well as preventing unrealistic overlap of objects. Their experimental results showed organ-organ interaction in real-time and produced sensate force feedback.

Chen, Sun, and Jin [2006] developed an interactive haptic deformable modeling in physical Bezier volume lattice space. Their haptic deformable approach involves the physical realism of MSM and the flexible control of FFD. Through distributing physical properties including mass, spring and damping coefficients of the object to bounded Bezier volume lattice, the deformations of the object in response to the haptic input follow physical laws and acquires a high deformation working rate. It also was suggested to be coupled in game design to augment the force feedback of the avatar when dragging and clashing.

Chen, Barner, and Steiner [2006] showed a displacement-driven spring-net deformation method for interactive surgery simulation. They built a high resolution triangular mesh via a 3D spring network consisting of mesh nodes. They assumed that the velocity of the deformation is low enough and the mesh reaches its equilibrium at each instant in surgical procedures. So mass was not considered in their method. A deformed index table was created for each node of the model. Its topology is like concentric circles. The center is like the node and the circle is treated as the neighbor nodes link. When the node (the center) is moved by the haptic device, its neighbor nodes (the circles from in to out) are drawn under Hooke's law. Its child nodes are always fixed. The movement of the node on the circle is only affected by its parent. In this way, the deformation boundary is not specified because they set a displacement threshold to identify the minimum moved nodes. This deformation algorithm is fast but the shape of the haptic contact area of the mesh is not adaptable.

J. Noh et al. [2000] proposed a deformation approach with radial basis functions (RBFs) that is similar to our deformation algorithm. This method was applied to creating facial expressions in animation design. They specified the feature point (containing control and anchor points) of their geometry deformation element. The distinction of each feature point was computed with its BSF system. Hardy multi-quadrics radial basic function was used in their system. That is a fast method, but if the control point is moved too far from its original position, large discontinuity occurs around the anchor and no

influence of the control point will propagate through the anchor points. So they assumed the movement of the control point has to keep within the specified region.

de Boer et al. [2007] developed an interpolation method based on radial basis functions. That method was applied to their unstructured grid domain such as fluid translation or rotation. Radial basis functions (RBFs) were used to interpolate the displacements of the boundary vertices of the mesh to the inner domain. That method requires solving a small equation system, only involving the nodes on the boundary of the flow domain. Several RBFs were tested in a variety of cases. However, the performance depends on the RBFs used. They found a best accuracy RBF with compact support for their model closely followed by the thin plate spline (TPS).

1.5 Research Objectives

To improve the realism and computation efficiency in haptic modeling with deformation, we develop some unique deformation methods in graphics and implement the octree and BSP trees in haptic rendering and touch point detection. Since the haptic VR application is interactive and real-time, the deformation algorithms need to be optimized to get efficient computation when it works on a complex (high resolution surface or complex structure) models. The objectives of this research are:

1. Measure and study the stiffness characteristics of human subject back to determine experimental models for the deformation.
2. Build BSP and octree haptic models. Compare the computational efficiency of normal, BSP, and octree methods. Also implement octree in detecting the touch point. BSP and octree will be used in graphic rendering.
3. Develop the functional shape matched deformation (FSMD) method. This algorithm is used in normal and tangential deformations. Build an array-map for the supposed curve. Solve the curve mapping from 2D to 3D surface.
4. Measure the Study the deformation shape of the human back. Develop the defined-shape tangential deformation algorithm. Create the interpolation algorithm for the deformation shape map.

1.6 Dissertation Organization

This dissertation is structured as follows:

Chapter 1 introduces the deformable modeling in virtual reality applications, investigates the main existing deformation algorithms, and discusses the advantages and limitations of each. The development and application of haptic VR is introduced and then its characteristics are described. The FSMD algorithms are presented briefly on the VHB modeling and the spatial partitioning algorithms to improve computational efficiency.

Chapter 2 analyzes the main principles and the system rigidity problem of the VHB and explains the solution of keeping the undeformed model in haptic rendering. The friction anchor problem is solved and its calculation optimized. The haptic SDK is explained and the research platform is described. The back stiffness map has been established and applied to the VHB model. This map can be modified to simulate the different dysfunctions. Chapter 2 also presents other related works of this dissertation.

Chapter 3 focuses on the BSP and octree implementations on the VHB model. It describes the method about dividing the model into subspaces and how to traverse BSP and octree. It gives the solution about the overlap problem during cutting the boundary of the subspaces. It discusses the computational efficiency of octree related to different sizes of the minimum defined subspace requirement. In programming, the data structure of VHB deformable model is described.

Chapter 4 investigates linear and nonlinear elastic deformation in the real world. It discusses why and how the Gaussian curve is chosen to match the shape of the elastic deformation. This matching shape is flexible in different conditions based on the variable materials or constructs. The deformation algorithms in the normal and tangential directions are described. The functional curve storage array is suggested and its resolution (size) is compared. The mapping algorithm for deformation curves is then presented. A visualization method of the deformation strain is presented at the end of Chapter 4.

Chapter 5 studies the deformation shapes of the real human back. Most of them deform in irregular area in the tangential plane at the touch point. Then a simplified method using ellipses is proposed and implemented on the VHB. To get a deformation shape map, a unique accurate interpolation algorithm is developed in this chapter and its features are discussed. It is also compared with the stacking interpolation algorithm. The mapping algorithm from 2D plane to 3D model is presented.

Chapter 6 summarizes the major contributions of this dissertation. It gives the conclusion of this research and proposes suggestions for future work.

2. Related Research Work

Some previous work has been accomplished to improve the VHB modeling and the realism. The VHB model is a pre-defined stiffness application. That means we need the physical property data of the human back for the haptic force generation. Since such physical information of the human back and soft tissue is reported very little so far, we measured live subjects to obtain the data ourselves (one model from each individual subject). Some solutions are suggested to overcome the limitations of the current haptic APIs.

2.1 Haptic Model, Stiffness Map and Stereo Viewing System

Haptic Model: A 3D human virtual haptic back (VHB) model had been implemented for the VHB project. The surface of this 3D digital upper body model was obtained from a typical adult male subject by using a 3D camera (the 3D Mega Capturor from Inspeck Inc., www.inspeck.com) in vivo. Bones of the upper body skeleton are from an open source of the Visible Female dataset from NIH. The 3D skeleton model in this research was from the open source on Internet. Because the VHB research is mainly interested in back diagnoses, the thoracic vertebrae need higher resolution than the other bones. In Figure 2.1, the left one is the previous VHB model covered with a non-textured semitransparent skin. In this model, all the ribs, both scapulae and the thoracic vertebrae

T1-T12 are involved. T# denotes the number of the thoracic vertebra and L# denotes the number of the lumbar vertebra. The right picture shows a textured model for the general experiments in VHB project development. The texture image is from a real subject's back.

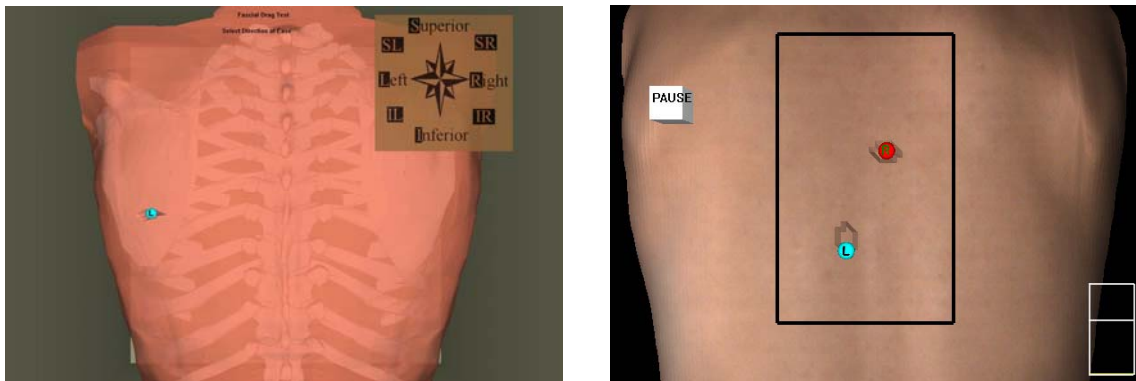


Figure 2.1 VHB Models with Skeleton (left) and Skin-Textured (right)

In the haptic interaction of the VHB, two small spheres (blue and red in Figure 2.1, right) are used to present the positions of the user's two fingers. Blue is left (L) and red is right (R). When a sphere contacts the back surface, the collision detection APIs will be called to handle this event to generate the force feedback on the user's fingertip through the PHANToM haptic interface, see Figure 1.1. In the VHB project, the haptic interface works in displacement-based mode. That means the change of force corresponds to positional movement of the haptic interface tip.

The back model in this dissertation research was implemented in high resolution polygonal mesh by the 3D camera mentioned earlier (see Figure 2.2). After resolution conversion, it becomes a triangular mesh and the grid size is about 5×5mm. With modification on the model such as trimming edges for smoother results, the number of vertices is slightly decreased. The bounding box of this model is 350×400×120mm containing 1588 vertices and 2962 triangles.

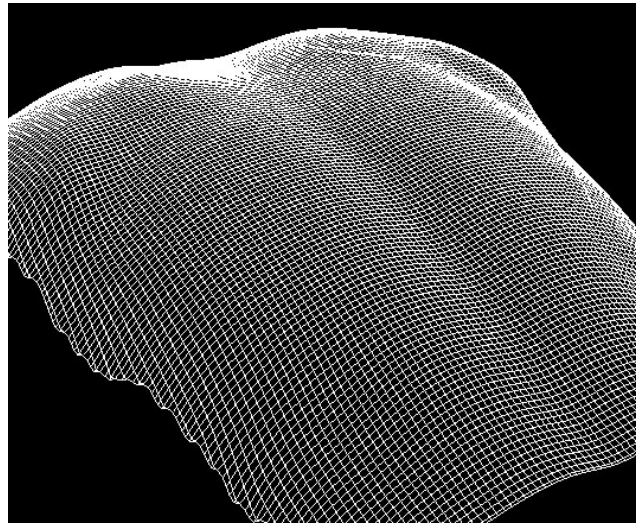


Figure 2.2 The Original High Resolution Back Surface Built by 3D Camera

Stiffness Map: Our haptic model is based on a predefined surface stiffness. To get the stiffness of the skin for the back model, we measured the compliance (reciprocal of stiffness) characteristics on several human subjects in vivo [Williams et al., 2007]. To build a stiffness map, we defined some key points and measured them on human subject backs. Several types of continuous stiffness fitting functions were used to match those

measured points (see Figure 2.3 for the resulting stiffness map for one individual).

Furthermore, the stiffness map can be adjusted manually to simulate different somatic dysfunctions.

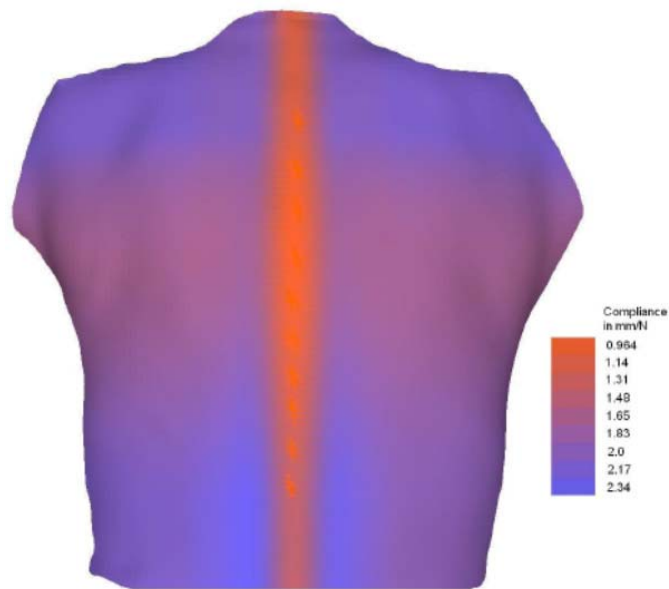


Figure 2.3 Statistic Stiffness Map of Human Back

Stereo Viewing System: In the VHB application, the haptic back model has been modified for stereo viewing on our stereo viewing system (SVS) [Ji et al., 2006]. This system allows the user to touch the virtual back via the haptic interface, where the haptic and graphical models are aligned (see Figure 2.4), as opposed to touching the virtual model some distance (up to a foot) in front of the graphical screen (as in Figure 1.1). Model data structure was optimized in this application.

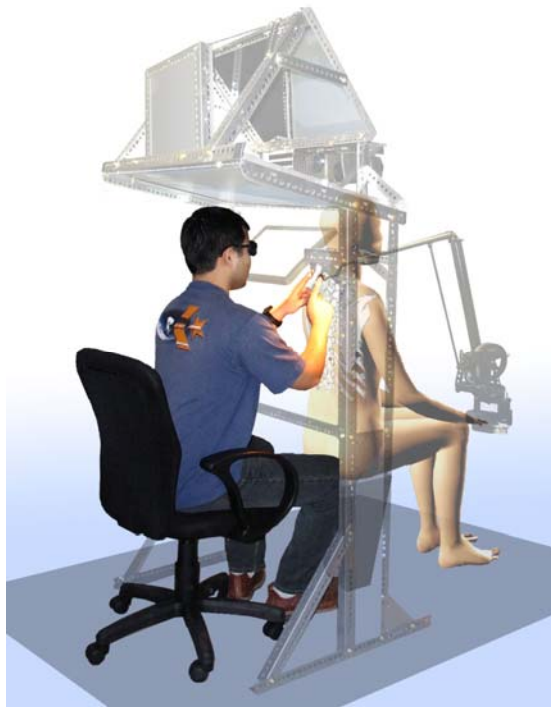


Figure 2.4 3D Stereo Viewing System for the VHB Project

2.2 Displacement Driven Force and Proxy Sphere

The PHANToM is used in our haptic applications; it is a displacement-driven haptic interface. The tip of its arm is the force effector. The user put his/her finger in the effector and moves it. The force generated by motors is sent to the user through the arm when the effector is touching the virtual model. In Figure 2.5, the PHANToM touches the model at the contact point, also called haptic interface point (HIP). Then the PHANToM penetrates the model surface and moves to a new position. The feedback force is calculated corresponding to the displacement d relative to the contact point. The force is

$$F = k * d \quad (\text{Hooke's law}) \quad (2.1)$$

where k is the spring constant at the contact point of the model.

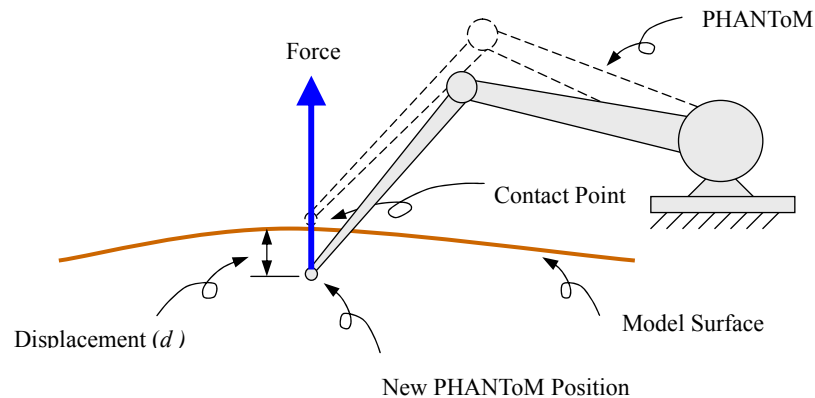


Figure 2.5 Force Generation of the Haptic Interface Depends on the Displacement

If the PHANToM tip stays at the contact point, no force is generated because zero displacement means zero force displayed to the user. The force loop is executed at a consistent 1000Hz rate to keep the feedback stable, continuous feeling, for the user on the haptic interface. In the haptic loop, the surface is not drawn on the screen, but the model surface is rendered only in the haptic process for force feedback generation.

In graphics, the position of the effector is presented with a small sphere (see Figure 2.1). When the PHANToM tip touches the model, the sphere always stays on the surface of the model even when penetration occurs. The sphere has to remain outside of the graphical model to avoid confusing the user. The PHANToM penetration is only used for force feedback generation. That sphere is called the proxy.

2.3 Non Deformable Modeling in Haptic Rendering

As the previous chapter discussed, the FEM and MSM have relatively expensive computational cost and boundary definition problems. They are more suitable for large deformations. A stiffness map has been measured and used on the VHB because the principal deformations are in the normal direction of the back and usually in small range of 0-10mm. The stiffness at a point on the back can be thought of as linear in such simulation. The force depends on the PHANToM displacement at the contact point and the local stiffness. In Figure 2.6, a deformable model is assumed and its haptic rendering is analyzed in the following text.

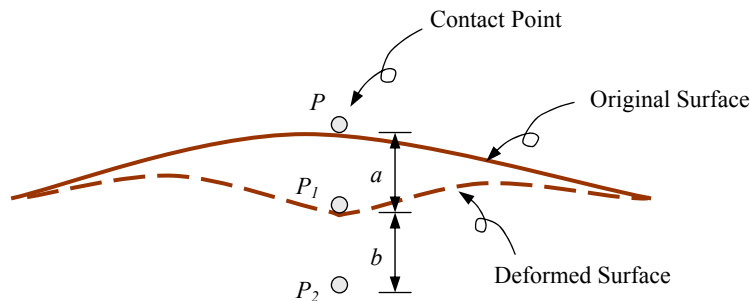


Figure 2.6 Deformation in Haptic Rendering Changes the Stiffness

Suppose the PHANToM presses the model at contact point P ; the force is f and the stiffness at the contact point is k . We get the deformation a at the contact point by Hooke's law.

$$a = \frac{f}{k} \quad (2.2)$$

The user can feel the displacement a through the PHANToM. With the deformable assumption, the model surface needs to deform to P_1 as the dashed curve shows in Figure 2.6. The contact point is changed to P_1 . The PHANToM has to move to P_2 to get enough penetration b (displacement to the contact point P) to keep the force f with the same stiffness value. To get the same force, b is equal to a . Finally the PHANToM actually moved $2a = a + b$. The force does not change but the displacement is doubled. So the final stiffness is changed to half of its original value.

$$2a = \frac{f}{\frac{k}{2}} \quad (2.3)$$

where the denominator in (2.3) is the new stiffness.

For this reason, the user will feel more compliance on such a model than the non-deformable haptic rendering model. This change is unreasonable and needs to be corrected. One solution is to double the stiffness, which is based on the original stiffness map, for the force generation in (2.2). Then the model is deformable in a dynamic haptic rendering. Because the dynamic (deformable) haptic rendering requires extra processing in haptic APIs, this method greatly increases the computational cost [pp.6-28, SensAble 2005]. That is why we consider the non-deformable haptic rendering so that the contact point does not change and (2.2) does not need to be modified. The user will feel the same stiffness as it is at the contact point.

On the other hand, the graphics rendering model is deformable to present a visible surface change under touch. This is important to the user to improve the visual realism.

2.4 Friction Effect in Tangential Deformation

In our method, the final deformation is defined as the combination of normal and tangential deformations. The penetration depth of the haptic interface at the touch point can be obtained with calls to the haptic APIs. This depth is the maximum the normal deformation can reach. The normal deformation calculation is based on the penetration depth (displacement). After studying the skin deformation shapes on real subject's backs, we found that the tangential deformation on the human back is more obvious than the normal deformation. Also, tangential deformation only happens under the friction effect between the finger and skin. The behavior of the friction is that the effector of the PHANToM is dragged back to a static point if the tangential force the user applies is less than $F_n * \mu$ (where F_n is the normal force and the μ is the coefficient of friction). That static point of friction is called the anchor. The tangential deformation calculation is based on the PHANToM displacement relative to the friction anchor.

SensAble Technical Inc. (the PHANToM manufacturer) does not currently provide an API for the anchor position. But their APIs can read the proxy sphere position,

PHANToM tip position, and the force vector. We derived the anchor with existing variables provided by the haptic APIs.

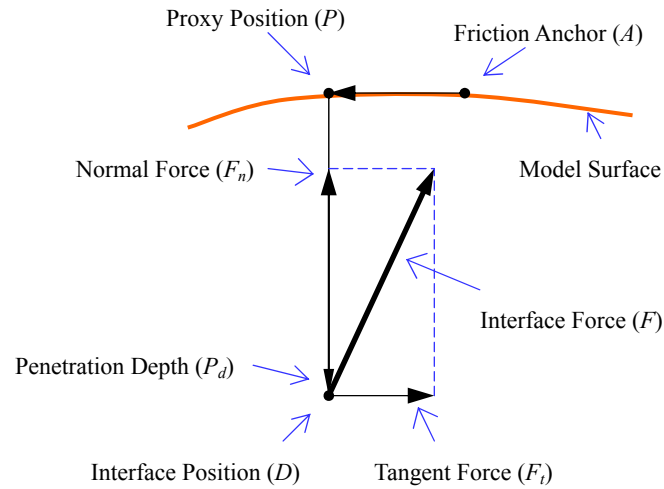


Figure 2.7 Friction Anchor

Figure 2.7 is a non-deformable model in haptic rendering; the PHANToM moves from A to D following the applied force from the user. The proxy sphere moves from A to P to keep P on the normal of the proxy position. Vector F is the force generated by the haptic interface, counteracting the force that the user applies. But F is not perpendicular to the surface because it is affected by friction. Its tangential component F_t keeps dragging the PHANToM back to the friction anchor A , and F is always toward A . In a critical case, when the friction decreases to zero, A will coincide with P .

Positions P , D and F can be determined in PHANToM API calls. Then A can be calculated as follows, where all the symbols are vectors.

$$P_d = D - P \quad (2.4)$$

Since $P_d \perp (P-A)$

$$\text{We get } \frac{P_d}{A-P} = -\frac{F_n}{F_t} \quad (2.5)$$

$$\Rightarrow A = -\frac{F_t}{F_n} P_d + P \quad (2.6)$$

So the maximum deformation in tangent direction is corresponding to the displacement $(P-A)$. This anchor A works well in our applications.

For the VHB, the static coefficient of friction is set to 0.46 and the dynamic coefficient of friction is set to 0.44 as proposed in [Zhang and Mak, 1999].

2.5 Utility Libraries and Platform

In our applications, the haptic interfaces are from SensAble Technologies Inc. This company has provided two haptic toolkits, GHOST and OpenHaptics. GHOST (which supports the PHANTOM 3.0) is the older version and easy to implement. OpenHaptics (which supports the PHANTOM and the Omni) is efficient, flexible, and compatible with most haptic products of SensAble Technologies Inc. This dissertation work is based on the OpenHaptics SDK.

C++ is the major development platform for our programming. We use OpenHaptics APIs for the haptics component and OpenGL V2.1 for graphics rendering.

3. Spatial Partitioning for Fast Rendering

In this chapter, two spatial partitioning algorithms based on BSP tree and octree will be discussed and implemented to the haptic model. The faster method is always of benefit to realism with an increase in model complexity. In the VHB, the surface of the model needs to be rendered twice separately for haptics and graphics. In graphics rendering, all triangles need to be drawn. OpenGL can optimize the shape rendering with a culling algorithm which only renders the triangles facing the viewer of the visible region of the model within the viewing frustum. In the haptic process, the OpenHaptics APIs capture the shape geometries from the OpenGL buffer [SensAble, 2005, pp. 6-5]. Therefore, all shape primitives drawn in the graphics process are rendered in haptics as well. Touching only happens at a point on the model surface but the whole visible model has to be rendered for force calculation by haptic APIs without optimization. So we can decrease the number of the rendered triangles to achieve faster haptic response.

OpenHaptics does not provide the standard library functions to optimize the haptic rendering area because they considered that the various optimization algorithms are very dependent on the specific application. So in normal and simple haptic applications with OpenHaptics, Table 3.1 shows the haptic rendering rates when the proxy touches the models containing a different numbers of triangles. In these tests, all the triangles of the model were drawn in the haptic process without any optimization

algorithm. The table also lists the graphic rendering performance (all triangles are drawn) for reference. The tested models in the first column of Table 3.1 are listed in Appendix A.

Table 3.1 Haptic and Graphic Rendering Rates on Different-sized Models

Model	Number of Triangles	Haptic Rendering (Touch Model) (FPS)	Graphic Rendering (FPS)
1	2915	104	198
2	6261	51	92
3	8163	34	65
4	20389	12	25
5	41582	6	11

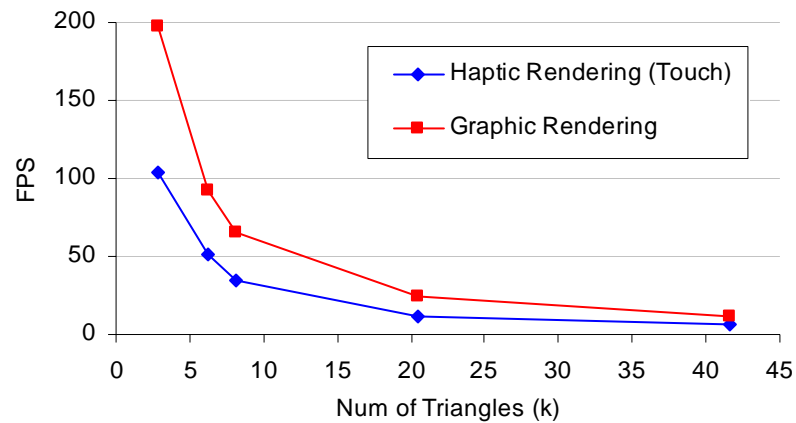


Figure 3.1 Plots for Table 3.1

In Figure 3.2, the performance of haptic rendering drops significantly with an increasing number of triangles. The haptic rendering rate is almost half of the graphic

rendering because the haptic rendering does not only draw the model same as graphics does but also includes the haptic APIs processes, such as collision detection (CD) and force generation.

Since the haptic rendering is invisible, it is not necessary to draw the whole model when the haptic device touches the model at a point. The OpenHaptics APIs handle collision detection involving the rendered triangles in the haptic process. So if only a small number of triangles around the proxy are involved in the collision detection, the computation should be faster with rendering fewer triangles. Such region of the haptic triangles should follow the motion of the proxy dynamically. However, this region cannot be too small to avoid the proxy going through the model surface. The size of the rendered region for haptics is important and its solution will be described later.

BSP and octree have been implemented on the VHB model to divide the model into small subsets so that a certain size 3D space (an axis-aligned cube) can be specified following the proxy. Any triangles coming into this space should be drawn for haptics.

3.1 Data Structure of the Haptic Model

A few different back models with different resolutions have been created for the VHB project. The main haptic mesh of this research is shown in Figure 3.2. This back mesh was originally taken from a real human back by a 3D camera and then was trimmed

and edited in 3D MAX for this research. The final model contains 1588 vertices and 2962 triangles.

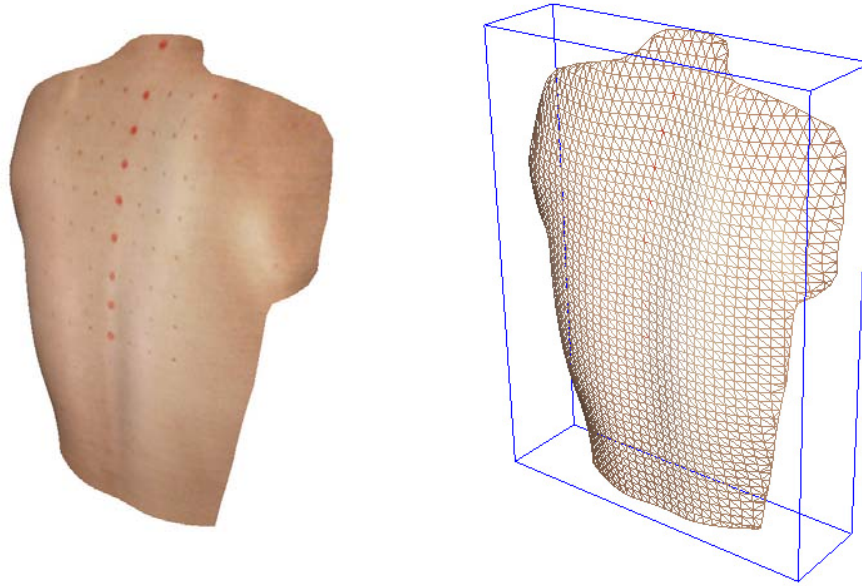


Figure 3.2 The Back Mesh

Two vertex structures were defined for haptic and graphic rendering. In the graphic rendering, since all primitives of the model need to be drawn in the scene, the triangles and the vertices are stored in a linear link for the most efficient time $O(n)$, where n is the number of the triangles of the model. Since only the graphic surface deforms, these vertices are involved the calculation of the deformation algorithm. In haptic rendering, the triangle structure is stored in the hierarchical binary tree or octree. The vertices are stored in another linear link. These vertices keep in their original positions because the haptic surface is static in this deformation simulation. The following are the descriptions of the vertex and the triangle structures

```
struct Vertex
{
    double position_xyz[3];
    double normal[3];
    double texture_xy[2];

    list <int> jointVerticesList;
    list <int> jointTrianglesList;
}

struct Triangle
{
    int vertexIndex[3];
    double normal[3];
}
```

Each vertex contains the lists of its adjacent vertices and triangle. These lists are pre-computed and static since no topology is changed. The position and normal of each vertex are dynamically updated for the graphic rendering. Each triangle has a list of its vertices. Also, the normal of the triangle is computed in real-time corresponding to the change of its vertices to get the Gouraud shading for smoothing the surface.

3.2 Rendering Performance Measurement

The haptic and graphic renderings are in serial order as shown in Figure 3.3. To evaluate their performance, the computation time of these two processes are measured independently with two timers in each system loop (Figure 3.3).

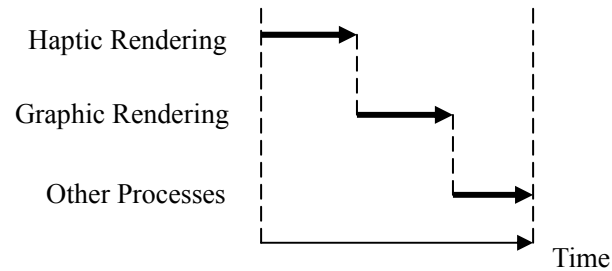


Figure 3.3 Performance Measurement of the Haptic and Graphic Renderings

We mainly measured the haptic and graphic renderings (shown in solid arrows) but the whole computation time of each system loop was also measured for reference.

3.3 Binary Space Partitioning (BSP) for Haptic Rendering

A BSP tree is an irregular data structure. In theory, the space of the model can be subdivided by arbitrary planes. In this research, we use axis-aligned planes to build the BSP tree. The construction time of BSP tree is expensive so it is done as a preprocess. The topology of the BSP does not change after it is created since the mesh for haptics is static as discussed before.

3.3.1 Construction of BSP Tree

The partitioning is based on the bounding box of the model. In this BSP algorithm,

the model is divided into hierarchical sub-trees using three types of the planes that are orthogonal to the x , y and z axes. The boundary box of the whole model is the root space (root node) of the tree, considered to be level-0. Each node is divided into two child boxes by a plane. The following is the description of the partitioning steps (see Figure 3.4).

(1) First, split the root space with a plane (green in Figure 3.4) which is orthogonal to the x -axis and crosses the center of the root space. Two new subspaces (nodes) are created in level-1.

(2) Then, partition each space (node) in level-1 with a plane (red in Figure 3.4) which is orthogonal to the y -axis and bisecting the center of this node. Four new subspaces in level-2 are created.

(3) Use a plane (yellow) that is orthogonal to the z -axis and cross the center of each box in level-2 to divide each level-2 node continuously.

(4) Repeat step (1), (2) and (3) to partition the nodes from level-2 until the partitioning stops.

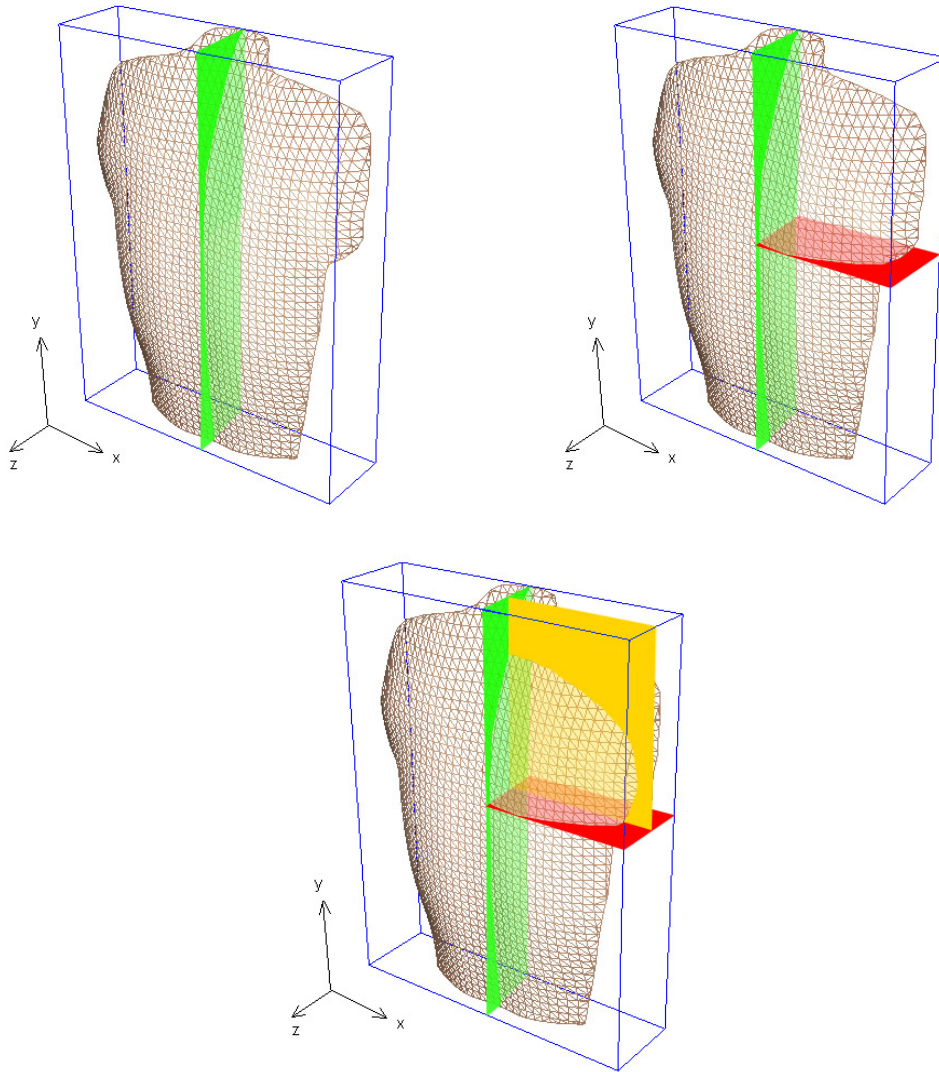


Figure 3.4 Binary Partition Aligned x, y and z Axes on the Model Mesh

The partitioning is executed in a recursive call. A requirement is needed to stop the recursive subdivision. In this algorithm, the number of triangles, n_i , located in the bounding box of each node is counted during the partitioning process. We set a threshold t to monitor the number of triangles in each node. If n_i of the subspace node is less than t , the partitioning stops automatically and this node becomes a leaf (end node) that contains

the triangles. Otherwise, the algorithm continues to divide this node into two child nodes. If the node does not contain any triangle, it is marked as empty (solid circles shown in Figure 3.5); the partitioning also stops in this case. The nodes in this BSP tree do not contain any triangles; only leaves do. Further, leaves do not have any children.

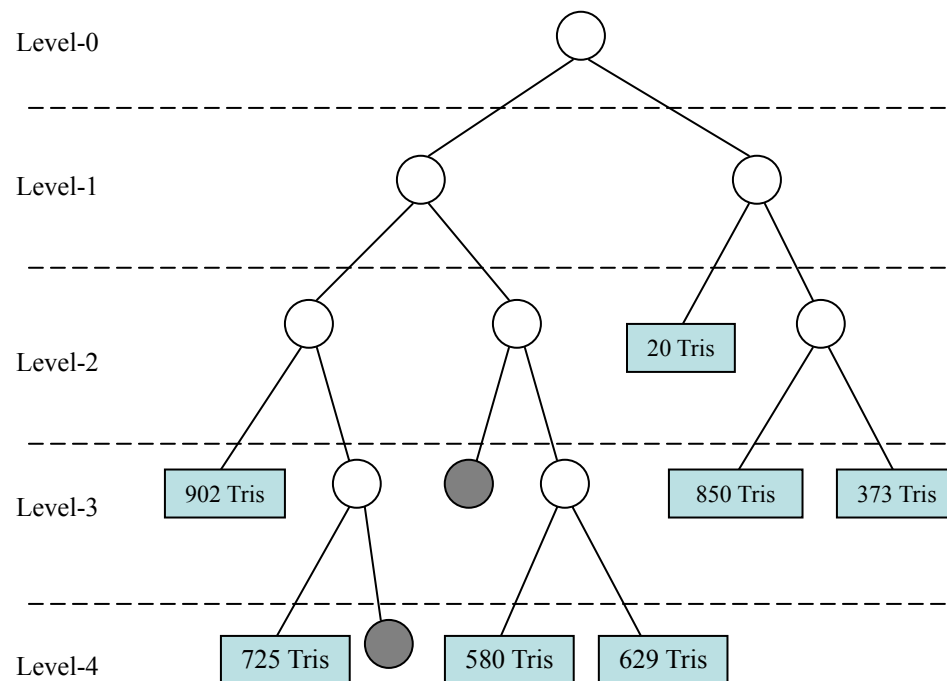


Figure 3.5 BSP Tree of the Back Model

In Figure 3.5, for example, the threshold t is set as 1024. Each leaf of the tree contains fewer than 1024 triangles. t is an important factor that affects the complexity of the BSP tree. Smaller t leads to larger tree height and traversing the tree becomes more complex. Larger t causes less partition benefits. Finally, 1024 was proved as the best value in our experiments. The pseudocode for the BSP is as follows:

```

1:  HapticBSP(node, level, threshold)
2:      IF node = "empty" THEN
3:          RETURN
4:      ENDIF
5:
6:      IF node->triangleNum > threshold THEN
7:          CASE mod(level/3) OF
8:              0: CALL PartitionNode(node, x_axisPlane)
9:              1: CALL PartitionNode(node, y_axisPlane)
10:             2: CALL PartitionNode(node, z_axisPlane)
11:          ENDCASE
12:
13:          IF node->child1->triangleNum = 0 THEN
14:              node->child1 = "empty"
15:          ENDIF
16:
17:          IF node->child2->triangleNum = 0 THEN
18:              node->child2 = "empty"
19:          ENDIF
20:
21:          CALL HapticBSP(node->child1, node->child1->level, threshold)
22:          CALL HapticBSP(node->child2, node->child2->level, threshold)
23:      ELSE
24:          CALL BuildLeafNode(node)
25:      ENDIF

```

Partitioning from the x -axis to the z -axis is a cycle. After z -axis partitioning, start x -axis partitioning again. During each subdivision, check the number of the triangles inside the subspace (triangles on the space border should be counted). The BSP result is shown in Figure 3.6.

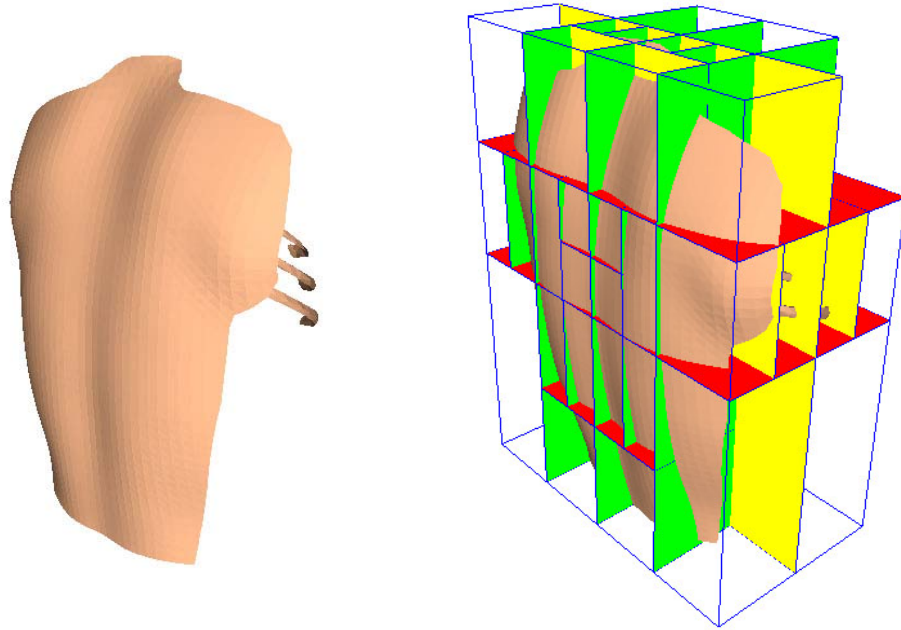


Figure 3.6 Splitting Planes and Leaf Boxes of the BSP Tree

The leaf spaces bounding of the binary tree are presented in blue wireframe boxes in the right portion of Figure 3.6. The green, red, and yellow rectangles indicate the x , y and z axis-orthogonal splitting planes. At the right top corner of the partition wireframe space, there is no blue wireframe box which means that there is no triangle in that space. That node is empty and no leaf is there.

3.3.2 Triangles Distribution

When the partition plane cuts the space of the node, in most cases, the plane has to pass through some of the triangles that are close to it. Since the triangles need to be

assigned in two groups logically but not physically (the triangle position does not change), the intersected triangles need to be allocated by some means as to which group (child) they will belong to. Figure 3.7 shows three cases about the partition on crossed triangles.

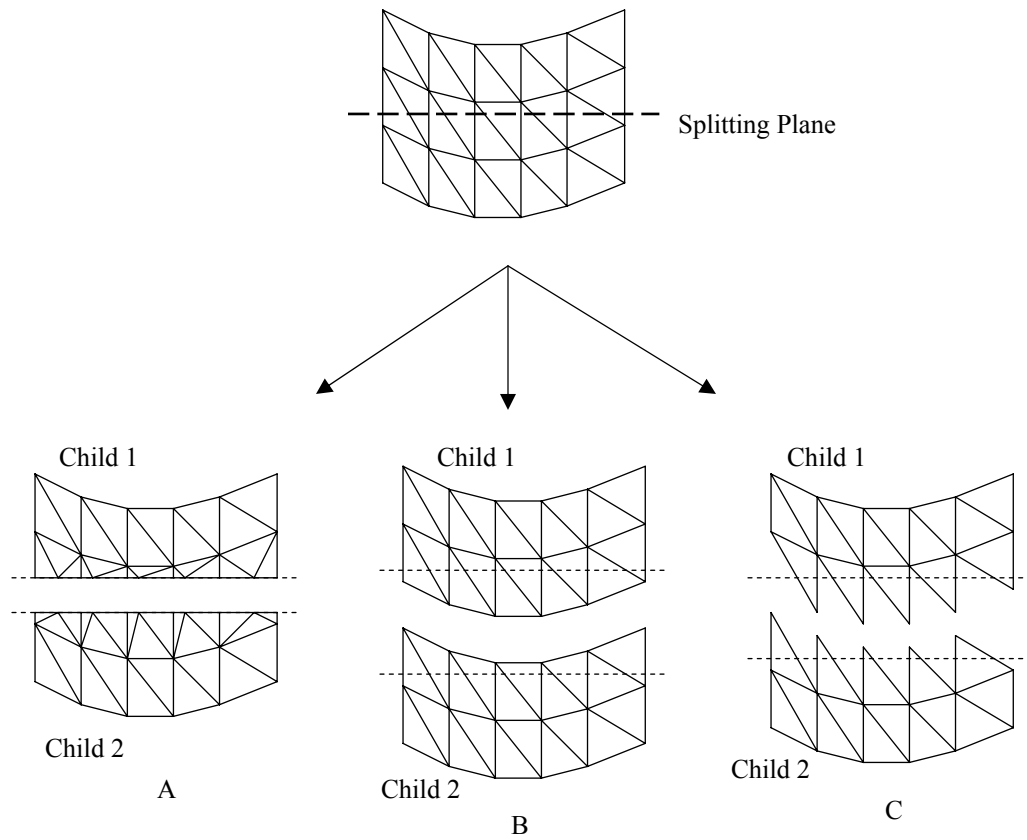


Figure 3.7 Three Cases of a Splitting Plane Crossing the Triangles

Case A: The plane cuts the intersected triangle and subdivides it into three small triangles located in child1 (upper) and child2 (lower). The edge of the cut triangles is

straight. This method causes the number of the triangles to be increased (topology changed) and needs additional computation to handle the subdivision.

Case B: If any vertex of the triangle lies on one side of the plane, the triangle is assigned to the child of this side. In this case, any triangle crossing the plane will be owed by the both child spaces. The leaf is the minimum unit of haptic rendering and all the triangles in the leaf should be drawn. Therefore, the intersected triangles will be drawn twice if both children need to be rendered. This leads to overlapping and a waste of rendering steps and computation time. The edge of the triangles of the child is still smooth (maybe not straight).

Case C: The triangle can only belong to one side of the splitting plane. If there are two or three vertices of the triangle on the same side of the plane, the triangle is assigned to the child of this side. Otherwise this triangle belongs to another child space. Such a rule can avoid the overlap in rendering problem. In Figure 3.7, the edges of the triangles are aliasing. Since the haptic rendering is invisible and the BSP tree is for haptic rendering only, the alias-edge does not appear in graphics. This method is feasible and the most efficient for rendering.

Although the partitioning is performed as a pre-computation (non-real-time), cases A and B affect the complexity of the model significantly when the threshold t is set to a small value. Case A increases the number of the triangles and case B increases overlapping rendering and requires more computation time. We use the method of case C

in the BSP tree construction. The haptic triangles are visualized for convenience in green in Figure 3.8. The boundary of the haptic region looks like saw teeth.

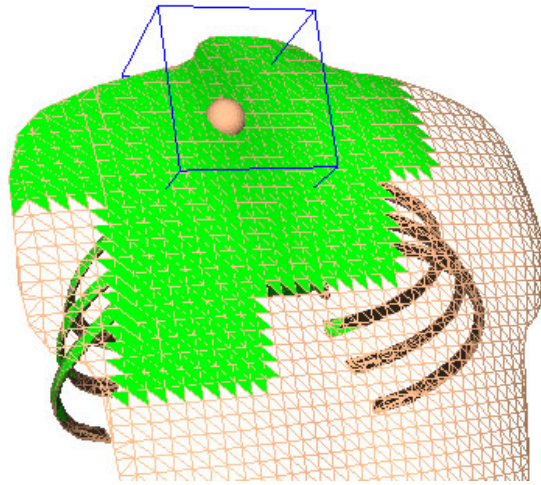


Figure 3.8 Leaf Borders of the Case C

The green region includes few leaf nodes and no overlapping rendering occurs. Each leaf contains a set of triangles. In that case, the proxy touches on the model and only a few leaves are rendered for haptic APIs. The method will be presented in the next section.

3.3.3 Haptic Detecting Box (HDB)

The model is subdivided into small parts by BSP. When the proxy is close to or touching on the model, we take the advantage of the BSP to determine which parts should

be rendered for collision detection and force generation for haptics. To achieve this goal, an axis-aligned cube is defined to follow the proxy. This cube center is always coincident to the proxy. When the cube is intersected with any leaf, this leaf should be drawn in haptic rendering because it is close to the proxy so that it has a high possibility to be touched by the proxy. All the leaves nearby the proxy are picked up and rendered during traversing of the tree. This procedure is recursive also. The algorithm is presented in the following pseudocode.

```

1: traverseWithHDB(node, HDB)
2:   IF node = "empty" THEN
3:     RETURN
4:   ENDIF
5:
6:   IF node is leaf THEN
7:     Draw this leaf for haptics
8:   ELSE
9:     Determine which child the HDB intersects with
10:    IF Child 1 intersects with HDB THEN
11:      traverseWithHDB(child1, HDB)
12:    ELSEIF Child2 intersects with HDB THEN
13:      traverseWithHDB(child2, HDB)
14:    ELSE
15:      traverseWithHDB(child1, HDB)
16:      traverseWithHDB(child2, HDB)
17:    ENDIF
18:  ENDIF

```

When a node is not empty, its two child spaces need to be checked if any of them is intersected with HDB (line 9 in above pseudocode). To do that, if any of the eight

vertices of the HDB is on one side of the node-splitting plane, this side child has an overlap volume with the HDB. If each side contains at least one vertex of the HDB, the both children intersect with the HDB. That means the plane cuts through the HDB. The Hessian Normal Form [Gellert et al., 1998] is employed to determine the distance between a vertex of the HDB and the plane. The sign of the result S indicates the side of the vertex location since we do not care about the exact distance between the vertex and the partitioning plane. (3.1) is simplified from Hessian Normal Form.

$$S = V \bullet N + d \quad (3.1)$$

where V is the vector of the vertex, N is the normal (normalized) of the partition plane, and d is the last parameter of the plane equation $ax + by + cz + d = 0$. If $S \geq 0$, it means the vertex is in the same side of the positive normal (for left child rendering) of the plane. And $S < 0$ means the vertex is in another half space of negative normal of the plane.

When the HDB is far from the model and in the empty node space, the HDB sometimes does not have any intersection with the model. Therefore, in the haptic rendering, no triangle is rendered. The resulting efficiency is very fast. In some tested cases, the rate of haptic rendering reached over 955 FPS.

The size of the HDB is critical to computation efficiency. Small cubes mean few leaves will be included. But the size cannot be too small (say the edge length of the HDB is 1/10 of the max length edge of the axis-aligned bounding box (AABB) of the whole model). Otherwise the proxy might get through the model surface frequently when the

proxy moves quickly on the model. That is because the collision detection finishes a little later after the proxy position update. If the proxy moves quickly, the rendered leaves close to the proxy may be not able to follow it in time. Then a delay results in collision detection involving the current proxy and the previous rendered leaves. So the proxy might be already out of the previous haptic region and puncture through the surface of the model abnormally. On the other hand, if the HDB is very big, say $1/2$ (or more) of the max edge of the AABB of the haptic model, the collision detection delay would not result in puncture. But the more leaves intersected with the HDB, the more rendering time is required. We finally define the HDB in $1/4$ size of the AABB of the model after several experiments (see Figure 3.8). This HDB works well on the VHB models.

The triangles rendering area (green in Figure 3.9) in haptic rendering when the proxy is moving close to a bunny mesh is visualized. The green area is getting big when the proxy is closing to the surface mesh. In picture 1, there are only 35 triangles are rendered (in green) in the haptic process, a very small percentage of the total triangles. So the haptic rendering efficiency is relatively fast, 955 FPS. In picture 4, 650 triangles are rendered in 49 FPS. This rate is better than the 28 FPS recorded on the model without BSP applied (in that situation, the whole model is green).

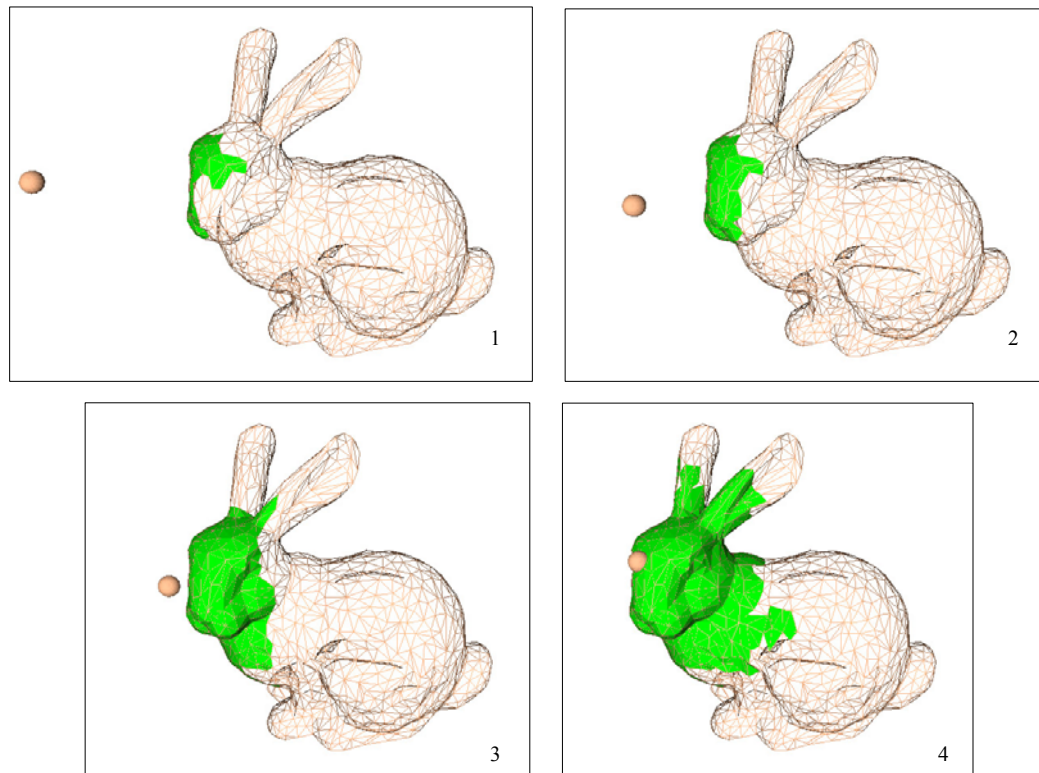


Figure 3.9 Haptic Triangles When the Proxy is Closing to the BSP Bunny

It can be noticed in picture 1 that the proxy is far from the model but a few triangles are still rendered. The reason is as follows. These triangles are usually lying outer which are opposite to the inner triangles, and the model structure is like that combined with few leaf layers. Each layer contains some leaves and is not overlapped with other layers, like an onion. In the intersection detection between the HDB and the leaf, the leaf has the half infinite space divided by the plane of its parent node. So if the proxy is in this half space and even is far from the triangles of the leaf, the proxy still has intersection with this half space of the leaf. So this leaf, say leaf 1, is still rendered. If there is another leaf, say leaf 2, between leaf 1 and the HDB, the intersection detection

should find that leaf 2 intersects with HDB instead of leaf 1. The case of picture 1 happens only on the outer leaves. Only when the dynamic HDB locates completely in an outer empty node, there is not any rendered triangle.

The rendering efficiency in picture 1 is still quick enough although a few leaves might be rendered needlessly. The intersection detection method is quick for the binary tree. We used another intersection detection algorithm in the octree with the HDB and a finite box space of the leaf.

3.3.4 Performance and Complexity Analysis

Optimization of the haptic rendering with the spatial partitioning algorithms has two benefits on saving the computational cost.

Performance: Just a small region of the model is drawn in the haptic process. In the worst case, the proxy is touching the model, 1/3 triangles (tested in experiments) in the leaves intersected with the HDB are rendered. So 2/3 of the time is saved in the triangle drawing. Use t_r to denote the time ratio of the BSP triangle rendering over the traditional triangles rendering (draw the whole model). So $t_r=1/3$. This benefit is not only in drawing triangles but also in the related haptic APIs which handle the collision detection and force generation. Table 3.2 shows the rendering rates of the triangle drawing, touch and untouch haptic renderings in the haptic process.

Table 3.2 Haptic Rendering Performance of the Binary Tree Model

No Haptic API (Triangle Drawing)			Untouch (Triangle Drawing, Collision Detection)			Touch (Triangle Drawing, Collision Detection, Force Generation)		
# of Rendered Triangles	Time (ms)	FPS	# of Rendered Triangles	Time (ms)	FPS	# of Rendered Triangles	Time (ms)	FPS
1387	2.6	378	1357	7.6	132	1671	11.0	91
3503	4.9	204	3620	18.5	54	3284	21.7	46
6541	8.1	124	6688	33.3	30	6688	43.5	23
8944	11.6	86	8679	47.6	21	9138	66.7	15
10134	13.2	76	10488	58.8	17	10347	71.4	14
13171	16.4	61	13354	76.9	13	13530	111.1	9
14722	17.2	58	14478	90.9	11	14842	142.9	7

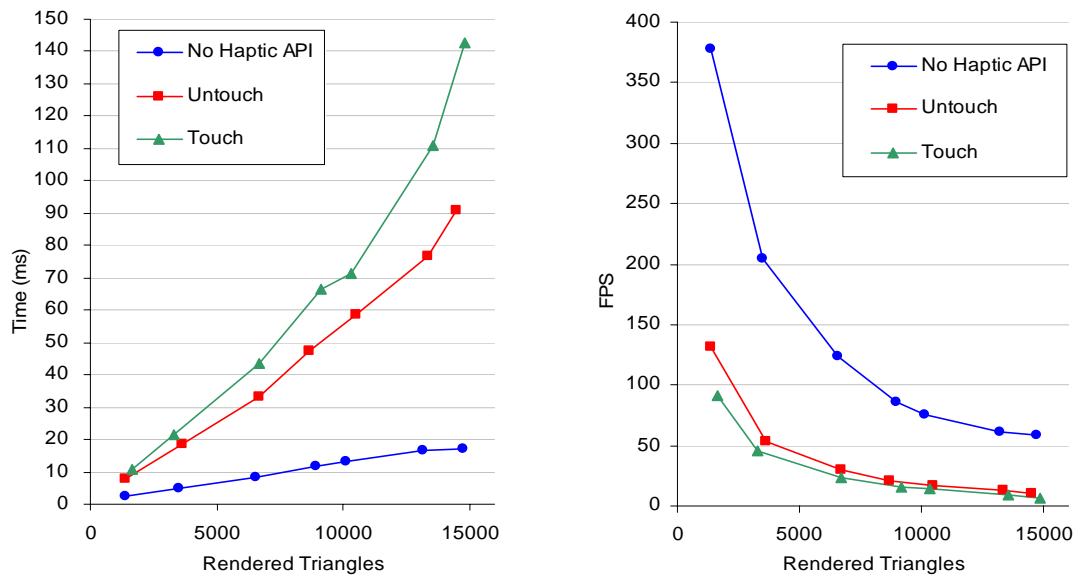


Figure 3.10 Plots for Table 3.2

The data are obtained from experiments on a 41.6k triangles BSP model. All the FPS were recorded for the haptic rendering process. The first row of Table 3.2 indicates what types of operations are included in each test. For example, the “untouch” test only involves the triangles drawing and collision detection. The right chart of Figure 3.10 presents the reciprocal results (FPS) from the data (time) of the left chart. The triangles drawing time (blue polyline) without haptic APIs (mainly contain collision detection and force generation) involved is significantly faster than the other two cases (red and green polylines) which involve haptic APIs. Even when 14000 triangles are drawn, it still achieves a rate of 58 FPS and does not delay the system loop significantly. But haptic APIs have to handle a number of triangles in rate 7 FPS such that the decrease in system speed is obvious. Although current graphics hardware can draw millions of triangles per second and hardly affect the system efficiency on such haptic models, the decreased number of the triangles for the haptic APIs noticeably improves the haptic response time.

Complexity: To determine the leaves intersection with the HDB, first, assuming that the model is divided into subsets stored in a linear link, the searching time on this structure is $O(n)$, where n is the number of the leaves. Let’s see the situation on the BSP model in the worst case (complete binary tree). The height of the complete binary tree is

$$h_{Bin} = \log_2(n) \quad (3.1)$$

Then the number of the total nodes n_{node} of the binary tree is:

$$n_{Bin} = \frac{2^{h_{Bin}+1} - 1}{2 - 1} = 2^{\log_2(n)+1} - 1 = 2n - 1 \quad (3.2)$$

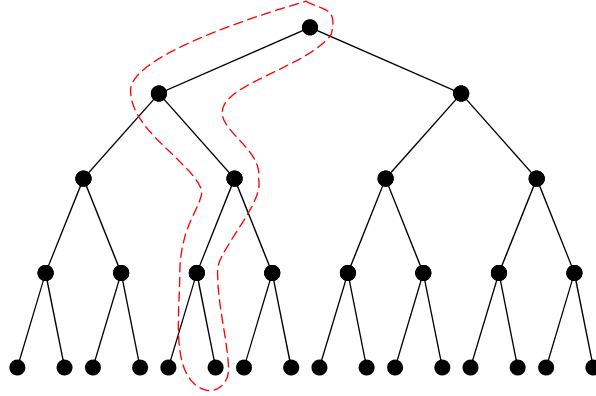
As we set the edge of the HDB to 1/4 of the maximum edge of the model bounding box, the volume of the HDB including the intersected leaves is in the range [0.1, 0.02] of the model bounding space. 0.1 is the worst case because the larger value means more leaves need to be detected and rendered.

Denote m as the number of the rendered leaves and r as the ratio of the rendered leaves over the total leaves ($r = \frac{m}{n}$). We can consider the maximum r is 0.1. Then, at most $r*n$ leaves will be found intersected with HDB. In this search process, we wonder how many nodes of the tree, denote as c_{Bin} , are traversed. Then c_{Bin} is the time cost of the intersection detection. Let's denote the ratio r_{Bin} .

$$r_{Bin} = \frac{c_{Bin}}{n_{Bin}} \quad (3.3)$$

$$\Rightarrow c_{Bin} = r_{Bin} n_{Bin} \quad (3.4)$$

What is the worst case of r_{Bin} relative to r ? Denote $w = \frac{r_{Bin}}{r}$. The worst case happens in the complete tree because all leaves are located at the bottom of the tree and the search for the leaves has to get through the whole height of the tree. It leads the maximum c_{Bin} . The worst case of w (maximum value) is in the minimum leaves of the rn , where rn should be at least one leaf. The reason is that the search for only one leaf passes the whole height of the tree (see Figure 3.11). Table 3.3 presents the worst w on the BSP trees in the different heights.

Figure 3.11 Traversed Nodes in the Worst Case of w Table 3.3 The Worst w Related to the Trees in Different Heights

Tree Height	3	4	5	6	7	8	9	10
r	1/8	1/16	1/32	1/64	1/128	1/256	1/512	1/1024
r_{Bin}	4/15	5/31	6/63	7/127	8/255	9/511	10/1023	11/2047
w	2.13	2.58	3.04	3.53	4.02	4.51	5.01	5.50

Usually the number of the triangles of the haptic model does not exceed 100k so that the height of the BSP tree in this research is no more than 7. The worst w can be chosen as 4.02 from Table 3.3. We get the complexity of the intersection detection in the worst case as follows:

$$\begin{aligned}
 c_{Bin} &= r_{Bin} n_{Bin} = r * w * (2n-1) \\
 &= 0.1 * 4.02 * (2n-1) \\
 &= 0.8n
 \end{aligned} \tag{3.5}$$

The result of (3.5) is better than n of linear searching. And in the practical applications with the BSP models, the intersection detections are mostly executed much more efficiently than the worst case.

Also compare Tables 3.1 and 3.2. The model used in Table 3.2 is the same as the model 5 of Table 3.1. In Table 3.1, the rate of the haptic rendering in touching is 6 FPS. In Table 3.2, most cases of column “Touch” are much better than 6 FPS (from 7 to 91). It tells that on the same model, the BSP structure improves the haptic rendering rate significantly.

3.4 Octree for Haptic Rendering

Octree is another method we have implemented to partition the model. It is similar to the axis-aligned BSP tree. But each node of octree has 8 child nodes (octants). The AABB of the model is subdivided simultaneously along the x , y , and z axes, and the split point must be the center of the box. This creates eight new child boxes (see Figure 3.12). The subdivision is recursive. Such partition makes the structure regular, and some queries may become more efficient because of this. Octree can be used in the same manner as the axis-aligned BSP tree.

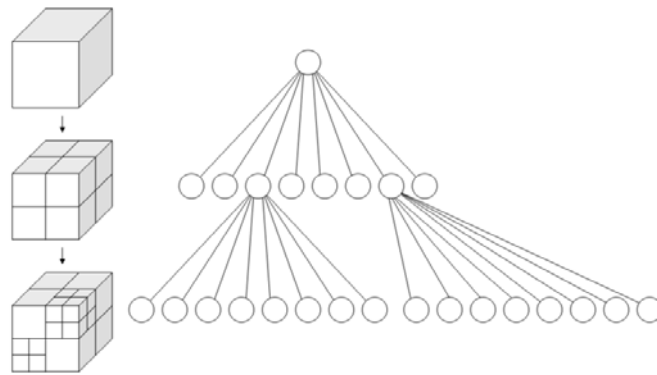


Figure 3.12 Octree Partition on the AABB

(Picture is from en.wikipedia.org)

We set the same threshold as BSP for the octree subdivision. The partition will be stopped if the number of the triangles in the node is less than the threshold. Only the leaf nodes store triangles. The octree algorithm splits the node into eight finite smaller boxes. The bounding box of each node is recorded in the struct of itself. Figure 3.13 shows the leaf boxes (wire framed) of the octrees on the models of the bunny and back.

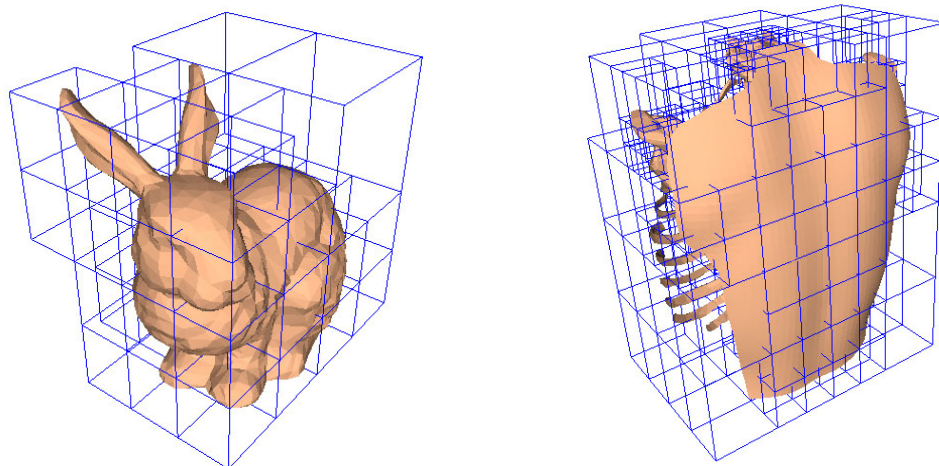


Figure 3.13 Leaf Boxes of the Octree on the Haptic Models

Some nodes of the tree are empty (no triangles), so there are no wire frame boxes there. The bunny has 2915 triangles without separated parts. The back model has 20389 triangles which contain the back skin, the ribs, scapulae and most vertebrae. The construction of the octree is described in the following pseudocode:

```

1:  HapticOctree(node, threshold)
2:    IF node = "empty" THEN
3:      RETURN
4:    ENDIF
5:
6:    IF node->triangleNum > threshold THEN
7:      BuildChildNodes(node)
8:
9:      AssignTrianglesToChild(node)
10:
11:     FOR child1 TO child8
12:       IF node->child#->triangleNum = 0 THEN
13:         node->child# = "empty"
14:       ENDIF
15:
16:       HapticOctree(node->child#, threshold)
17:     ENDFOR
18:   ELSE
19:     BuildLeafNode(node)
20:   ENDIF

```

Line 7 is to build eight child subspaces by three axis-aligned planes. In the triangles distribution, we use a similar method of case C in the BSP to allocate the triangles to certain child nodes. The difference is that the node of octree has a finite bounding box. If at least two vertices of the triangle lie in this child box, this triangle

should belong to this child node (see Figure 3.14). Line 9 handles this job. And also, the partition is pre-computed and the topology of the octree is static after creation.

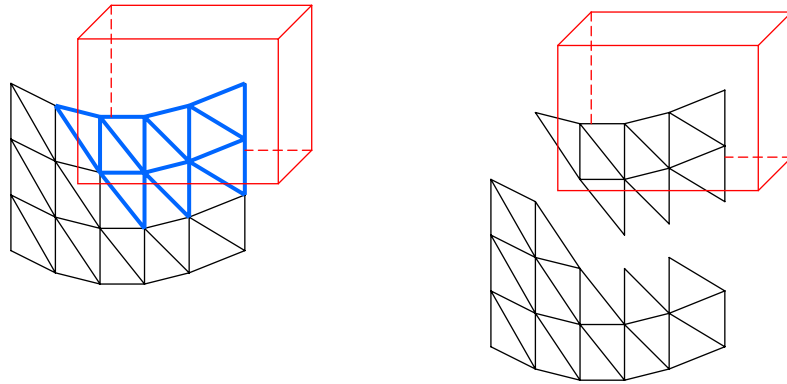


Figure 3.14 Triangles Intersection with Bounding Box of the Node

3.4.1 Haptic Triangles Detection

A sphere can be used to define a space to detect the intersected leaves around the proxy instead of a box. The proxy is the center of the sphere. If any of the eight vertices of the node (leaf) bounding box is in the sphere, the node is intersected with the proxy sphere. But computing the distances between the proxy and the eight vertices of the node box is inefficient. However, for the cube (HDB), the AABB/AABB intersection [Akenine, 2002, pp.600] is employed to detect the intersection. In this way, there are only six Boolean operations needed.

Figure 3.15 shows the situations of the haptic triangles rendering when the proxy is in different positions. There are no triangles drawn when the HDB of the proxy does

not touch the model, meaning no haptic API is involved. Only if the proxy is very close to or touching the model, few leaves are rendered for haptics.

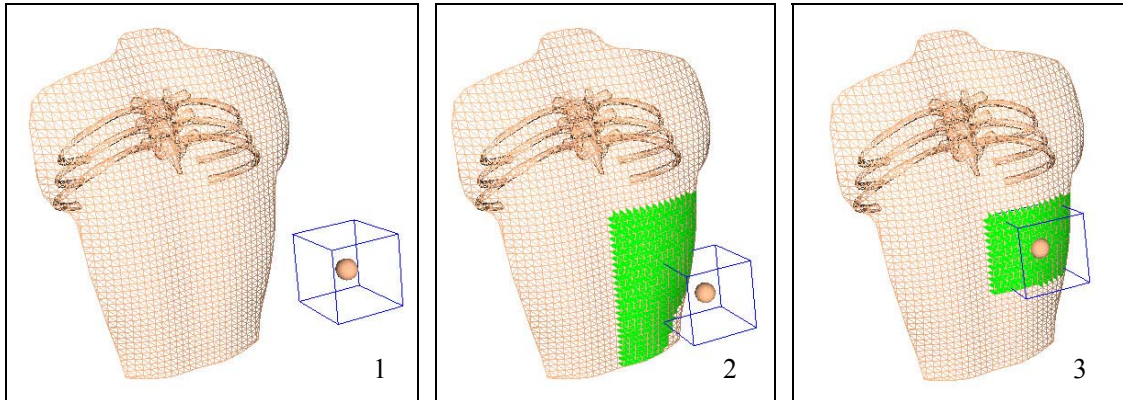


Figure 3.15 HDB is Closing to the Octree Back Mesh, Few Leaves Rendered

The haptic performance on the octree model is much better than on the model without partition as well as BSP.

3.4.2 Complexity Analysis

Let's set h_{Oct} as the height of the octree. The triangles lie in the space without an obvious density change so that the leaf boxes are in similar size in the octree. Normally, the height is not more than 3 because if the leaves are too many, as we can see in the last two columns in Table 3.2, the complexity in the worst case might be over $O(n)$. The threshold needs to be adjusted according to the number of the triangles of the model to keep the leaves in a reasonable range.

$$h_{Oct} = \log_8(n) \quad (3.6)$$

where n is the number of the leaves. The number of the total nodes of the complete octree is denoted as n_{Oct} . Then,

$$n_{Oct} = \frac{8^{h_{Oct}+1} - 1}{8 - 1} = \frac{8^{\log_8(n)+1} - 1}{7} = \frac{8n - 1}{7} \quad (3.7)$$

The discussion about the HDB of the octree model is similar to the BSP. Denote c_{Oct} as the number of the nodes during the searching for the intersected leaves. So c_{Oct} is also the time cost of the intersection detection in the octree. Denote the ratio r_{Oct} .

$$r_{Oct} = \frac{c_{Oct}}{n_{Oct}} \quad (3.8)$$

Tables 3.4 The Worst w Related to the Octrees in Different Heights

Octree Height	2	3	4
r	1/64	1/512	1/4096
r_{Oct}	3/73	4/585	5/4681
w	2.63	3.50	4.38

Since in the octree of the haptic models in this research, its height is no more than 3, the worst w is 3.50 from Table 3.4. The time cost of the intersection detection in the worst case can be written as follow:

$$c_{Oct} = r_{Oct} * n_{Oct} = r * w * \frac{8n - 1}{7}$$

$$\begin{aligned} &= 0.1 * 3.5 * \frac{8n - 1}{7} \\ &= 0.4n \end{aligned} \tag{3.9}$$

$0.4n$ is better than $0.8n$ of the binary tree in the worst case. Tests concerning the rendering rates of 3 cases (non-haptic API, untouch and touch) on the octree model were done. The same model as Table 3.2, 41.6k triangles, was used in this test. The original measured data is attached in Appendix B. The comparisons to the BSP are shown in the following charts.

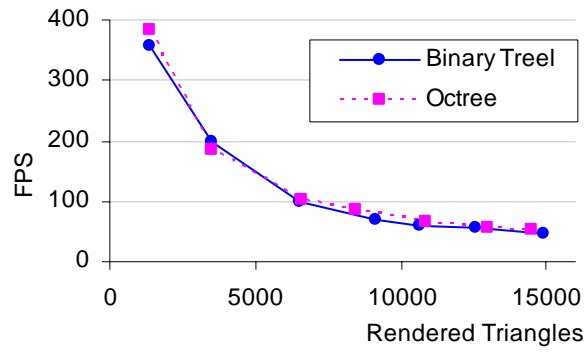


Figure 3.16 Haptic Triangles Drawing without Haptic APIs Involved

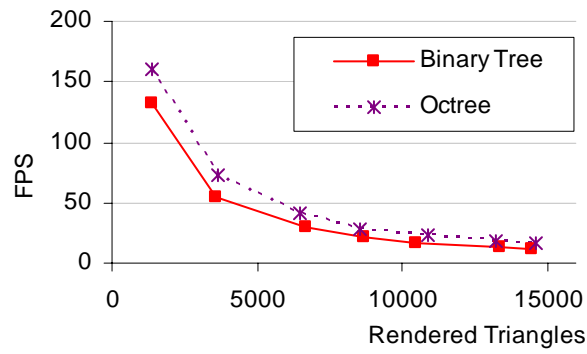


Figure 3.17 Haptic Rendering Rate in Untouch Condition

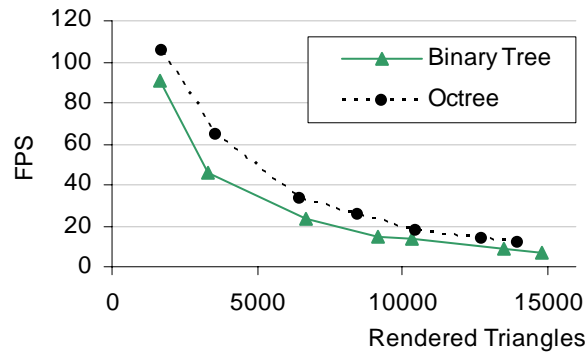


Figure 3.18 Haptic Rendering Rate in Touch Condition

The haptic triangles drawing of the octree model is almost same as the binary tree model in the Figure 3.16. It indicates that the intersection detection between the HDB and the leaves of the octree is as fast as a binary tree. But in Figure 3.17 and 3.18, we can see that the rendering performances of the API in the octree are slightly better (10%-20%) than the binary tree. The haptic rendering performance is much improved with octree spatial partition on the model. So we use the octree partition in the deformation algorithm of this dissertation.

4. Functional Shape Matched Deformation

In this chapter, the focus is on the surface deformation algorithm for haptic models with human-like tissue. When haptic simulation is applied to soft objects, the deformable model is desired to enhance the graphical and haptic realism. The VHB is such a project, based on human back modeling. In general, the purpose of the VHB is to diagnose back somatic dysfunctions concerning the vertebrae, muscles, and other soft tissue. The deformation of soft tissue under external forces is presented on the back. The goal of this chapter is to simulate the deformation of the back skin.

In the literature review, several deformation algorithms were discussed. Table 4.1 summarizes the features of the current popular deformation methods.

Table 4.1 Features of the Existing Deformation Algorithms

Method	Advantages	Disadvantages
Finite Element Method	Most accurate, Physical based	Very expensive computation cost, need the accurate physical parameters, Complex boundary condition definition.
Mass-Spring Model	Fast, Physical model	Linear simulation, boundary condition definition.
Free Form Deformation	Fast	Not accurate, need manipulation
Shape Interpolation	Simple algorithm	Expensive computation cost, much manipulation work
Skeleton Driven Deformation	For deformation without external force, good for global deformation.	Much work of manipulation, not flexible

The most accurate approach is Finite Element Method (FEM). Both FEM and Mass-Spring Model are based on accurate, well studied properties of the material. But the soft tissue and the muscles are nonlinear in most cases. Their characteristics vary on different subjects. We use a predefined curve to simulate the shape of deformation on the VHB model. The curve can be from math functions or be defined by user for the desired shape.

4.1 Deformation Shape on Elastic Material

The deformation of elastic object surfaces such as skin, rubber, or sponge looks

like the shape in Figure 4.1. That is a sharp rigid object pressed on the surface of a soft object. This deformed surface can restore completely when the external force is removed. On the human skin, the mass of the deformed part is small so that the damping can be ignored. The deformation is activated immediately corresponding to the pressing force without delay.

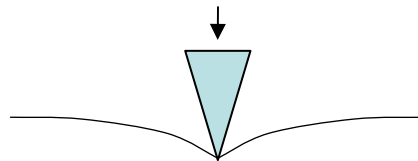


Figure 4.1 Deformation Cross Section on Soft Object

In the haptic simulation of palpation, the user usually touches the virtual patient by fingers. In Figure 4.2, the rigid touching object is a ball. The right picture shows the finger pushing on the soft surface. Because of the round shape of the finger, the skin deformation shape under the finger is similar to the left one. The shape of finger-induced deformation is considered to be symmetric about the y -axis (vertical axis) under the force perpendicular to the model surface.



Figure 4.2 Deformation Due to a Finger is Similar to a Round Rigid Object

The properties of the homogenous isotropic elastic objects were discussed by [Luo and Xiao, 2006]. They derived certain geometric properties from physical properties of the contact between a rigid object and an elastic object that affect the shape change of the elastic object under deformation. Some typical deformation cross sections of the elastic object were presented. For homogenous elastic objects, the deformation is maximum at the contact point. The displacement of the surface is decreasing with increasing distance to the contact point. The contour of the deformation is a circle. The boundary of the deformation (zero displacement) is a circle too.

The human back is composed of some different types of elastic tissues and rigid bones in layers. The back surface is an even area of the human body. There are few bumps on it. The skin and the muscle are the main factors affecting the elastic properties of the back. The elasticity of the back is considered as a normal elastic material in this haptic application. Since the VHB group already measured the stiffness of human backs [Williams, 2007], the force generation in haptic simulation depends on an experimentally-derived linear stiffness map. To simplify the calculation, the maximum deformation at the contact point is assumed to be linear according to the relative displacement of the proxy and using Hooke's Law.

In the deformation area, all the deformed points follow the movement of the contact point. The cross section of the deformation can be fitted to some predefined shapes such as a Gaussian Curve, etc. Such curves are expressed in algebraic functions

and can be modified for different materials based on the study of the specific materials.

Figure 4.3 shows a standard Gaussian Curve (flipped in horizon) which is suitable for the deformation under the round rigid object.

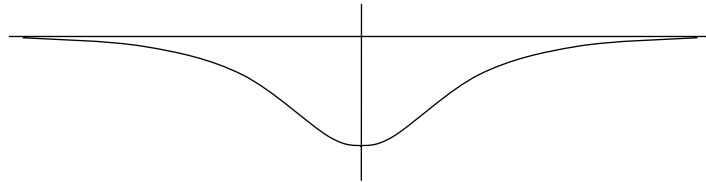


Figure 4.3 Flipped Gaussian Curve in Horizon

Figure 4.4 presents a method for the deformation under a sharp rigid object pressure by a modified Gaussian Curve. For the sharp applied object, the first order of the surface of the deformed object from the contact point should be monotonic decreasing. In the instance of the Gaussian Curve, the part between its two inflection points can be removed. Then join the rest of the two curves to get the picture to the right of Figure 4.4.



Figure 4.4 Modified Gaussian Curve for Sharp Object

Since the back muscles pennate in different directions in the tangent of the back surface, this affects the deformation shape such that it is not a regular circle. The

deformation boundary will be discussed in Chapter 5.

In real palpatory diagnosis, the operator usually pushes his/her fingers on the patient and moves on the skin to feel the shape and stiffness changes of the bone or soft tissues. The deformation caused by this action of the operator can be considered combining with two components in the normal and tangential directions to the body surface.



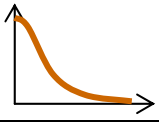
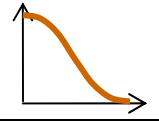
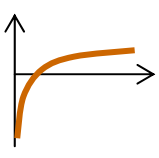
Figure 4.5 Normal and Tangential Deformation Combination

The tangential deformation of the skin is driven by the friction under the finger. The left picture in Figure 4.5 is the deformation only in the normal direction. The right one is the deformation when the finger moves on the skin. The tangential deformation is perpendicular to the normal direction. In the experimental measurement of back deformation, we found that the maximum normal deformation (at the contact point) is around 10 mm and the maximum of the tangential deformation is around 20 mm. So the tangential deformation is more obvious than normal deformation during palpation. We apply predefined shape matching to both the normal and tangential deformations.

4.2 Normal Deformation

Most elastic material and soft tissues are continuous in physical properties. Thus physical changes, like deformation on the surface, are also gradually continuous. The physical properties of human soft tissue are mostly anisotropic and nonlinear. When the surface is deformed by an external force applied at a contact point, the surface deformation is significant near the point and attenuates gradually with the distance increasing to the contact point. To achieve faster deformation computation, we choose some algebraic functions to simulate the change. After investigation in the most familiar algebraic functions, we found that the Gaussian Curve (GC) has some similar features like the gradual change properties of soft tissue. Table 4.2 shows the reasons why the GC is better than others. They are not full comparisons of every algebraic curve, but some typical curves.

Table 4.2 Comparison of GC, Cos, and Log Functions

Shape	Function	First-order derivative at the ends	Non-Symmetry
	Gaussian Curve (half)	0, 0	Yes
	Cos (half period)	0, 0	No
	Log	$+\infty, 0$	Yes

In this table, only three curves are listed since they are more similar to the deformation cross-section than others. The first-order derivative at the end of the curve indicates if the deformation is smooth at the contact point and at the deformation boundary. Zero derivative is reasonable at the both ends of the deformation curve for the round applied rigid object in the palpation simulation. The non-symmetry indicates if the curve has any symmetrical or similar part itself. Since the deformation from the contact point should be a decreasing curve, the shape should not have similar properties. So the non-symmetry is good for the deformation. From Table 4.2, we can see that the half Gaussian Curve has the best features in those three shapes. Also, it is suitable for the sharp object in the previous discussion. Equation (4.1) is the Gaussian Curve function with zero offset.

$$d_n = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{r^2}{2\sigma^2}} \quad (4.1)$$

where d_n is the displacement of a point on the surface in the normal direction. In Figure 4.6, $r=OD$ is the distance from the original position of the deformed point D to the contact point O . At the contact point, the user pushes the haptic device and penetrates the haptic surface to get the force feedback according to the stiffness at that point. The depth of the penetration can be computed by subtracting the current position of haptic device from the proxy position. The maximum normal deformation is at the contact point, see d_{max} in Figure 4.6.

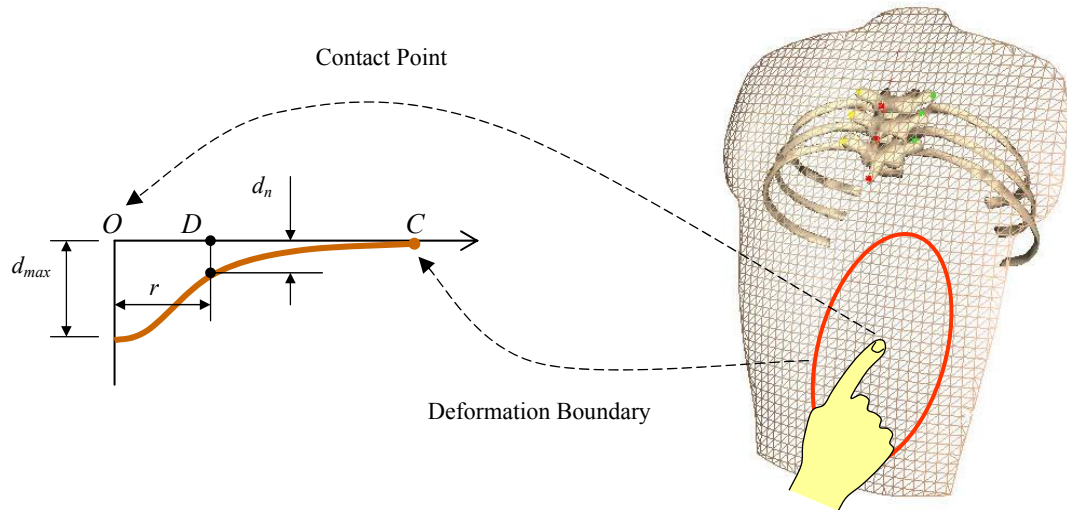


Figure 4.6 Deformation Region and Contact Point

In this method, the deformation boundary is supposed as a regular shape such as a circle or ellipse. The center of this shape is the contact point. The size of the deformation area is variable depending dynamically on changes in the applied force. The force change

is related to the penetration depth of the proxy. In the programming, the size of the normal deformation boundary is a linear function of the penetration depth P_n which is in normal direction. P_n is also the maximum displacement in the normal direction. To get the normal displacement d_n of a point D (in Figure 4.6), which is within the boundary, in efficient computation, we do not calculate the Gaussian function for each point but use an array to store the Gaussian Curve and map the D to the array. The ratio $q = \frac{OD}{OC}$ is used for this mapping. Following are the steps to compute the normal deformation at any point in the deformation boundary.

```

1: NormalDeformation(point,  $P_n$ )
2:   IF point is out of the boundary THEN
3:     RETURN 0
4:   ENDIF
5:
6:   compute locationRatio  $q$ 
7:   map the point on Gaussian Curve

```

In the case of a circular deformation boundary, $q = \frac{OD}{CircleRadius}$. But in other cases, r is derived in a different way. The above derivation is based on a planar surface. However, the surface of the haptic back model is not planar, but 3D. The deformation shape is defined in a plane that is perpendicular to the normal of surface. So the point on the model needs to be projected on the deformation shape plane of the contact point first. D is the projected point on the deformation shape plane. Then transform the Gaussian Curve displacement mapping to the normal plane. Figure 4.7 shows the deformation cross

section in the normal direction.

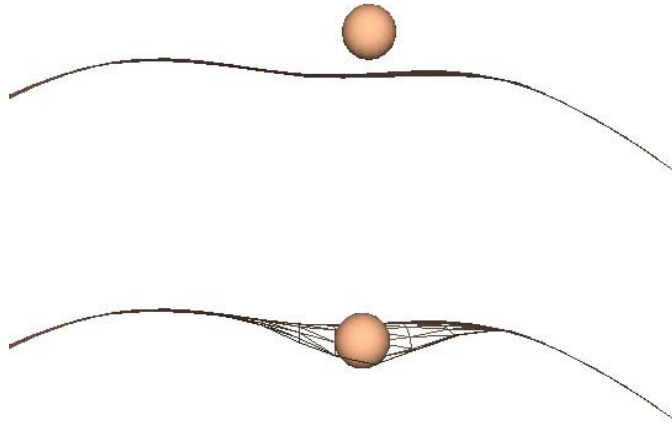


Figure 4.7 Normal Deformation of Back Mesh Mapping with a Gaussian Curve

The top picture is the cross section of the original back mesh without touching. The bottom one is the cross section of the deformed mesh in the normal direction matched on Gaussian Curve.

4.2.1 Normalized Gaussian Curve Mapping Array

Since the Gaussian Curve is predefined for the deformation shape, to save computation time, the GC is stored in an array. For each point within the deformation boundary, its distance ratio is mapped to the index of the array to get the deformation value.

The standard Gaussian Curve is expressed in (4.2).

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.2)$$

In the normal deformation, the position change of each point on the surface should not exceed the change range of the contact point. The level of deformation is corresponding to the GC. So GC needs to be normalized to present the change from the maximum (100%) to minimum (0%). Since the minimum of GC is zero, the maximum value is only to be normalized.

The GC is symmetrical about the y -axis. Therefore the half GC in the positive x -axis only needs to be computed when the offset is $\mu = 0$.

Set $y = 1$,

when $x = 0$ and $\mu = 0$ in (4.2), we get $\sigma^2 = 0.1592$ and $\sigma = 0.4$.

And when $x=1.4$, $y=0.0021$. y is around 1/500 of the maximum. We can consider that it is close to zero to be the minimum of the GC, see the left curve in Figure 4.8.

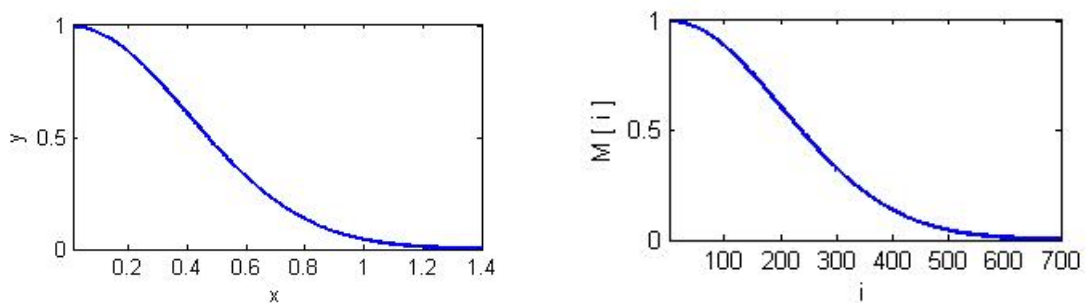
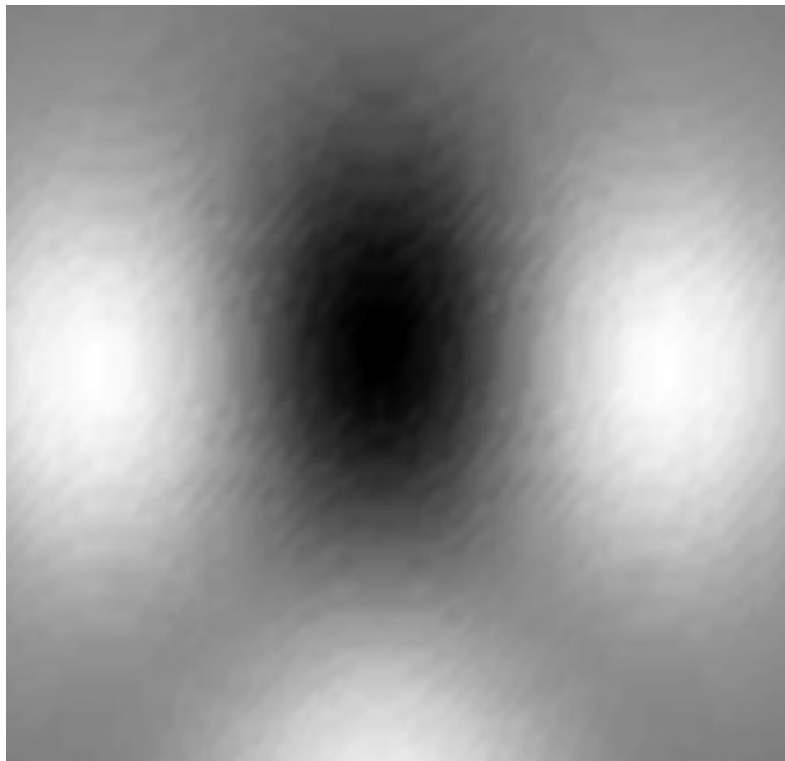


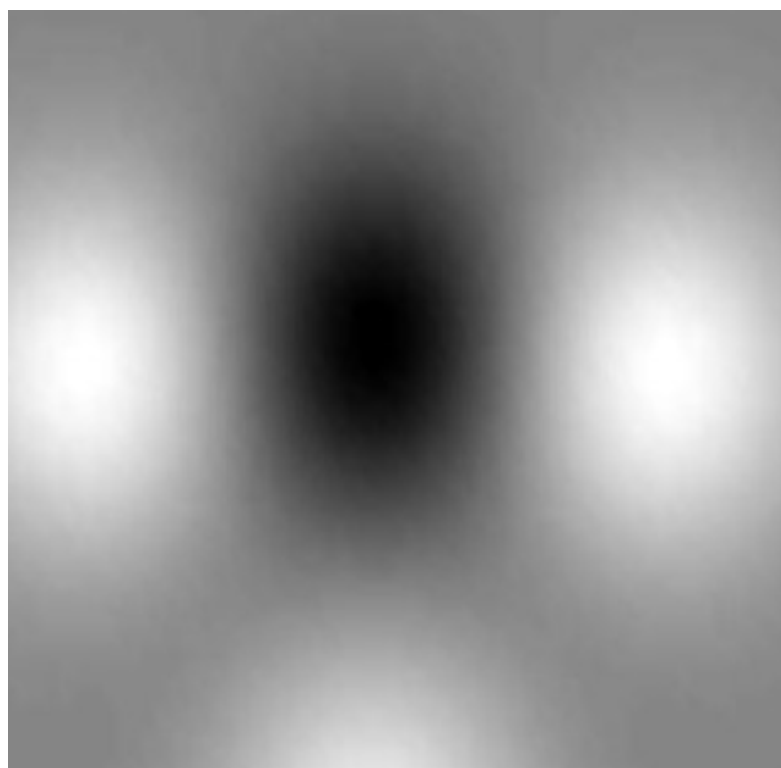
Figure 4.8 Half Gaussian Curve and Its Storage Array $M[i]$

The right picture in Figure 4.8 presents the mapping of the storage array $M[i]$.

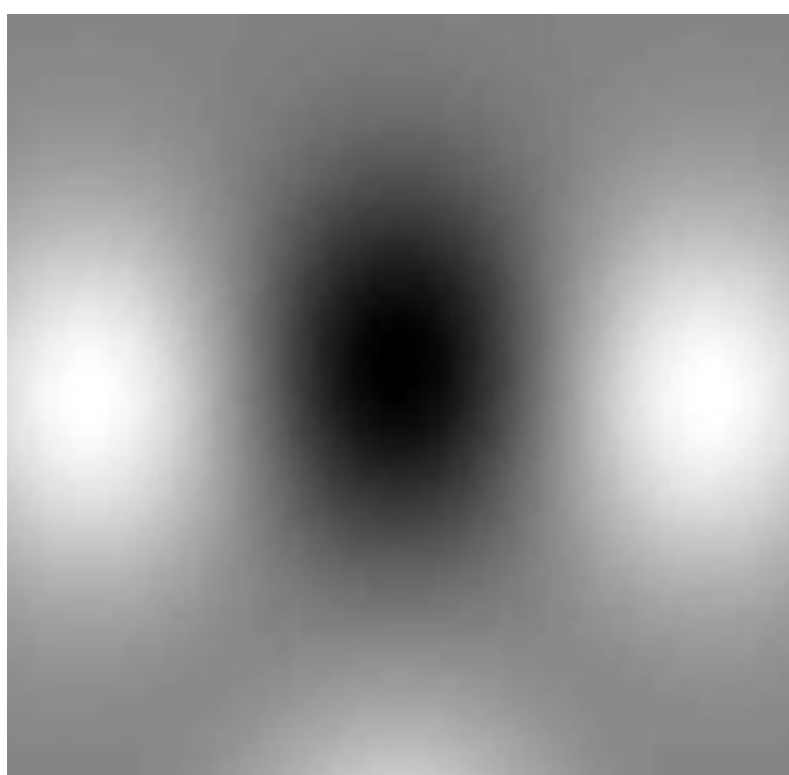
The horizontal axis indicates the index of the array. The vertical axis is the normalized GC. The size of the array affects the smoothness of the resulting shape. Figure 4.9 shows the final smoothness of an array with a size of 20, 130, and 700. Case (a) is significantly different from (b). But (b) is not obviously different from (c). (c) is slightly better than (b). A higher resolution than 700 is not necessary. We choose 700 as the resolution of the Gaussian Curve map.



(a) Array Size:20



(b) Array Size:130



(c) Array Size:700

Figure 4.9 Smoothness in Different Array Sizes

The range of the array is $[0, 1.0]$ according to its index from 0 to 699. The index i is mapped linearly on the x -axis within $[0, 1.4]$. Using a nearest interpolation to get i then locate the $M[i]$.

$$d_n = M \left[\text{round} \left(699 \frac{OD}{OC} \right) \right] * P_n \quad (4.3)$$

Figure 4.10 shows the mapping which is from the deformation plane to the Gaussian Curve array. Point C is the intersection of the deformation boundary and the extended line of OD . It is the minimum deformation of the Gaussian Curve since it is on the boundary. The GC presents the trend of the displacement along OC from minimum to maximum. The Gaussian Curve and array mapping is also used in the tangential deformation.

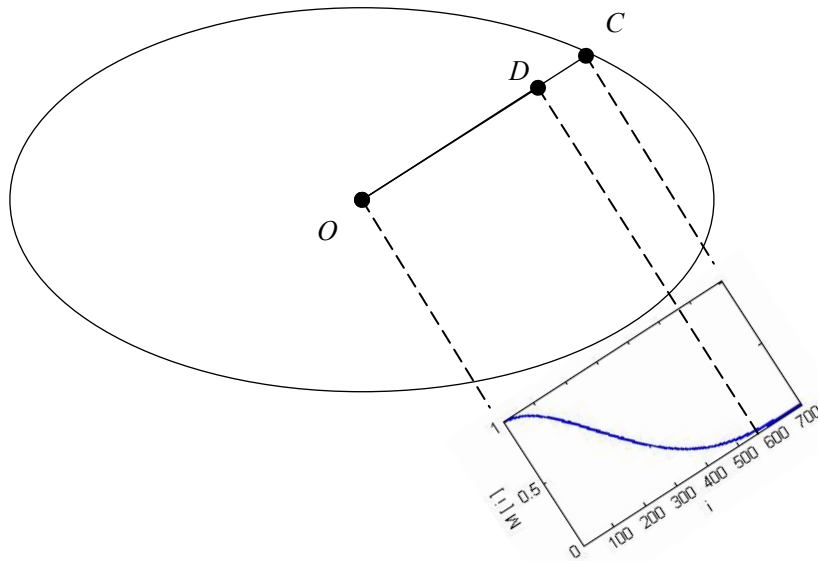


Figure 4.10 Gaussian Curve Array Mapping

4.3 Tangential Deformation

This is another deformation component, perpendicular to the normal deformation. The tangential deformation of the surface is caused by the friction and the movement of the palpating object. We again use the Gaussian Curve to calculate the tangential displacement of the moved point according to the distance from its original location to the contact point in the tangential plane.

OpenHaptics does not provide the API for obtaining the original contact point position, also called the friction anchor, in the condition of the friction defined. But it has been derived in Chapter 2 (see point A in Figure 4.11).

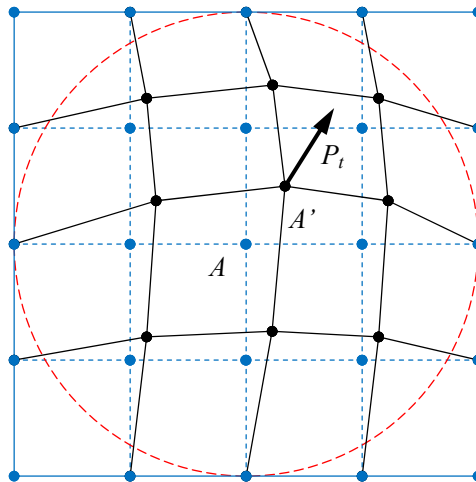


Figure 4.11 Tangential Deformation in the Circle

The blue-dashed-line mesh is the original model. Black-solid-line is the deformed mesh. P_t is the tangent vector of the external force. In this case, the boundary of the

deformation is a circle (dashed red). A is the friction anchor and the center of the deformation area. A' is the new position of the moved point A . The displacement vector AA' is the maximum tangential displacement of the surface under the friction effect since A is the contact point in this case. The friction P_t at contact point is given as follow,

$$P_t = P_n * \mu \quad (4.4)$$

where μ is the coefficient of friction mentioned in Section 2.5, and P_n is the force in the normal direction at the contact point. The tangential displacement vector at any point on the surface is denoted as d_t . It is always in the same direction of P_t . Dividing the distance, which is from this point to A , by the radius of the circle leads to the location ratio for the Gaussian Curve array. We can get the following,

$$d_t = M[\textit{point location ratio}] * P_t \quad (4.5)$$

The tangential deformation is computed individually without the effect from the normal deformation because all the calculations are based on a static model. The results of the displacements in the x , y , and z axes of the normal and tangential directions can be vectorially added to the original position.

The above tangential deformation method is discussed based on the same stiffness character as the normal. For variable type subjects or models, the stiffness in the tangential direction might be different than the normal direction. In such cases, an adjustment factor f_{adj} (depending on the measurement results) can be included in (4.4) by multiplication to change the tangential stiffness.

$$P_t = P_n * u * f_{adj} \quad (4.6)$$

For example, loose skin generally moves in a larger range than tight skin. f_{adj} can be set as 1.3 for loose skin and 0.7 for tight skin. Figure 4.12 shows the tangential deformation on the back mesh.

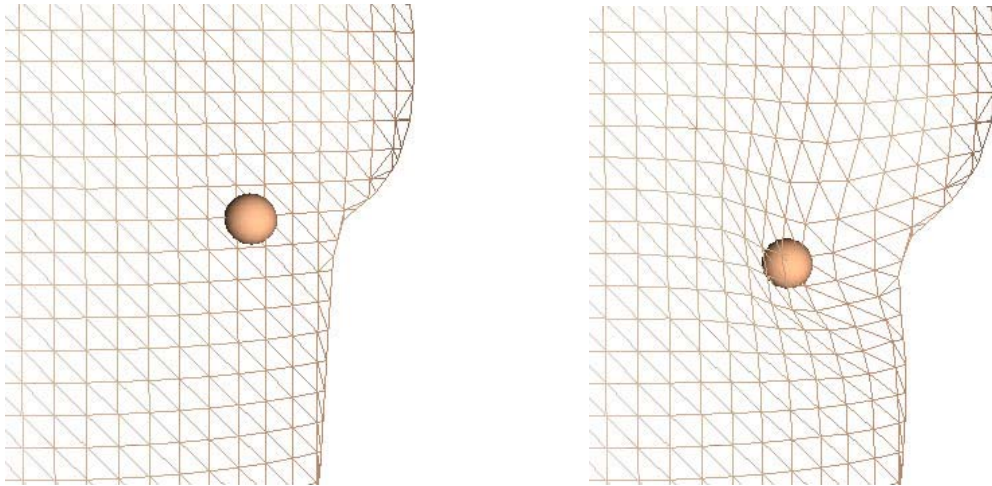


Figure 4.12 Tangential Deformation of Back Mesh Mapping on Gaussian Curve

The left picture of Figure 4.12 is a part of the back mesh without deformation. The right one is the tangential deformation of the back mesh from the viewing direction perpendicular to the model surface.

4.4 Point Mapping to Ellipse Plane

When the deformation shape is assumed as a circle, the location ratio of the point,

which is on the surface, for the Gaussian Curve array mapping can be obtained by computing the distance to the contact point. But for other deformation boundary shapes, we need to predefine the shape plane and project the point from the model surface on that plane. Then the location ratio will be computed on this shape plane. The plane is perpendicular to the normal of any point on the surface and passes the point. Because we will use an ellipse to define the deformation boundary, the conversion from the original mesh node onto the deformation shape defined plane is based on an ellipse. It is the same manner of the conversion for other shapes on the deformation shape plane.

Each vertex (point) on the surface and in the ellipse is deformed under an external force. In Figure 4.13, J is a vertex on the model surface. The contact point is O . Its normal is ON and it is normalized already.

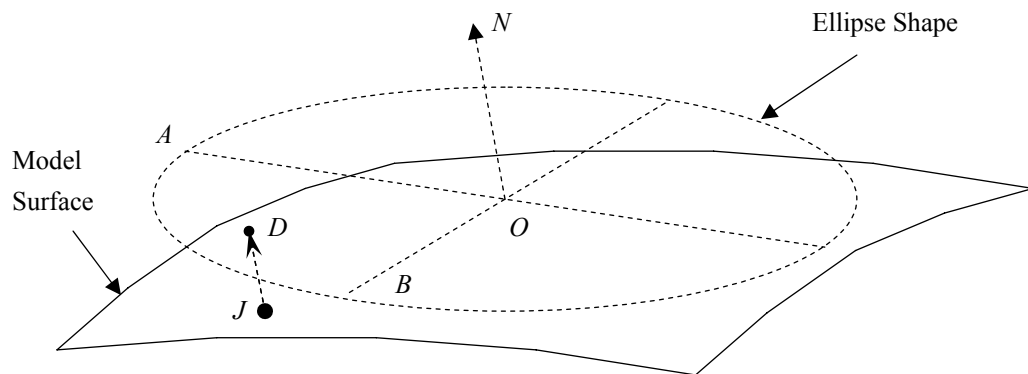


Figure 4.13 Point Projection on Deformation Shape Plane

The ellipse is dashed and the plane is perpendicular to the normal ON . O is the

origin of the ellipse. Each vertex on the deformable surface has a predefined ellipse. If the contact point is not exact at a vertex, its deformation shape ellipse should be obtained by the interpolation of its surrounding ellipses. The normal at this point is also computed by the interpolation of its neighboring normals. D is the projection of point J on the ellipse plane. We need to get D for the Gaussian Curve mapping. This is an intersection problem of a line (containing J) and a plane (containing O). Refer to the method by [Bourke, 1991].

Denote m as the factor for the line JD . Since $|ON| = 1$,

$$D = J + m * ON \quad (4.7)$$

where m is a real number. D , J and ON are all vectors.

Solving for m gives

$$m = \frac{ON \bullet (O - J)}{ON \bullet ON} = \frac{ON \bullet (O - J)}{1} = ON \bullet (O - J) \quad (4.8)$$

Substituting (4.8) in (4.7) gives

$$D = J + [ON \bullet (O - J)] * ON \quad (4.9)$$

At this time, we do not know if D is in the ellipse or not by just calculating $|OD|$.

If D is out of the ellipse, it does not deform. Mapping it to Gaussian Curve can tell us how much displacement is associated with it.

The ellipse is defined with its major axis a and the ratio t to the minor axis. Figure 4.14 shows the projected point D in the deformation ellipse.

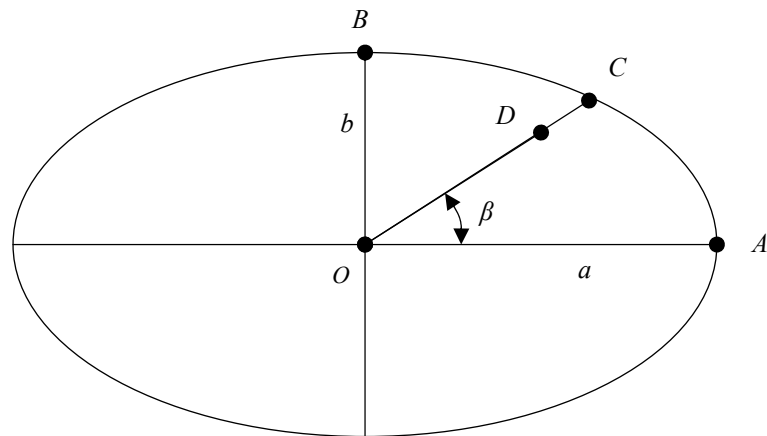


Figure 4.14 Point Location in Deformation Boundary Ellipse

Major axis: $a = |OA|$ and Minor axis: $b = |OB|$

The ratio of the x and y axes is t .

$$t = \frac{b}{a}$$

The ellipse equation is below:

$$x = a * \cos \beta$$

$$y = b * \sin \beta = a * t * \sin \beta$$

To calculate the ratio of the deformation related to the Gaussian Curve, denote it as following,

$$q_e = \frac{|OD|}{|OC|}$$

This ratio is for an ellipse. Deriving the right expression with the ellipse equations gives the following.

$$\begin{aligned}
\frac{|OD|}{|OC|} &= \frac{|OD|}{\sqrt{(a \cos \beta)^2 + (b \sin \beta)^2}} \\
&= \frac{|OD|}{\sqrt{a^2 \cos^2 \beta + a^2 t^2 \sin^2 \beta}} \\
&= \frac{|OD|}{\sqrt{a^2 \cos^2 \beta + a^2 t^2 (1 - \cos^2 \beta)}} \\
&= \frac{|OD|}{\sqrt{a^2 (1 - t^2) \cos^2 \beta + a^2 t^2}} \\
&= \frac{|OD|^2}{\sqrt{|OD|^2 a^2 (1 - t^2) \cos^2 \beta + |OD|^2 a^2 t^2}} \tag{4.10}
\end{aligned}$$

Since

$$\begin{aligned}
OD \bullet OA &= |OD| * |OA| * \cos \beta \\
&= |OD| * a * \cos \beta \tag{4.11}
\end{aligned}$$

We get

$$(OD \bullet OA)^2 = |OD|^2 * a^2 * \cos^2 \beta \tag{4.12}$$

Substituting (4.12) in (4.10) gives

$$q_e = \frac{|OD|}{|OC|} = \frac{|OD|^2}{\sqrt{(OD \bullet OA)^2 (1 - t^2) + |OD|^2 a^2 t^2}} \tag{4.13}$$

The location ratio q_e indicates where the point is in the deformation area. If q_e is equal to or greater than 1.0, the point is on the boundary of the deformation area or out of the area. That means no deformation happens at this point. If q_e is less than 1.0, this point

is in the deformation area and needs to be involved in a deformation calculation.

$$\left. \begin{array}{l} q_e \geq 1: \text{ on boundary or out of deformation area} \\ q_e < 1: \text{ in deformation area} \end{array} \right\} \quad (4.14)$$

Since the user can drive the haptic device to slide on the model surface freely, the contact point O (see Figure 4.13) may be at an arbitrary position on the surface. But J must be one of the vertices of the surface.

4.5 The Neighbor Nodes Ring Algorithm

In the defined deformation shape method, only the vertices that are in the deformation boundary need to be involved in the deformation calculations. The deformation region at any position covers a small part of the whole model surface. It is a benefit to computational efficiency to just check the vertices around the contact point in a small range (not all the vertices of the model). The searching algorithm is based on the expanding nodes ring enclosing the contact point (see Figure 4.15).

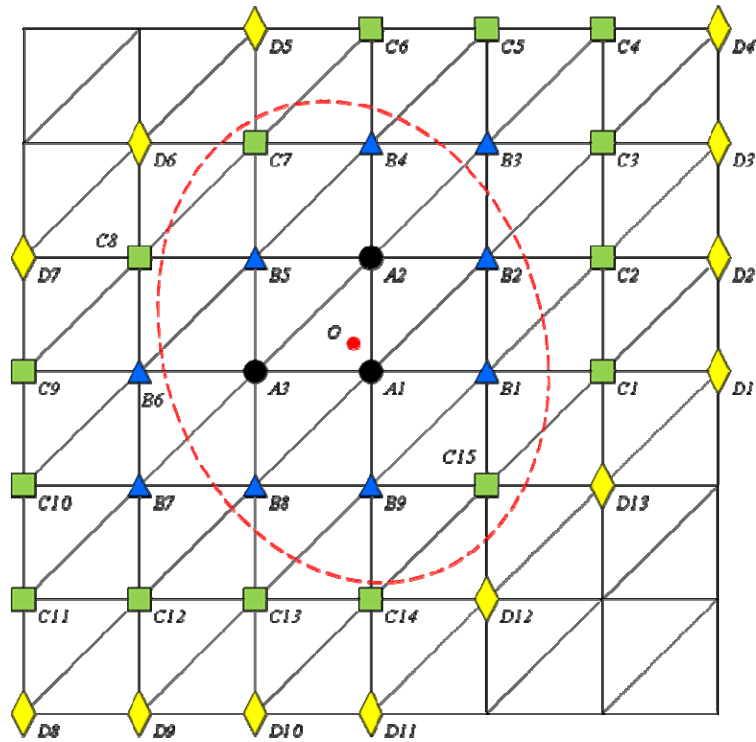





Figure 4.15 Neighbor Nodes Deformation Rings

The red dot O is the center of the ellipse (dashed red) which is the deformation boundary. To determine which vertices are in the ellipse, we build node rings with the same “center” O . As we know that O is the friction anchor, the method of intersection of line and triangle [Sunday, 2003] is employed to find which triangle O lies on. In Figure 4.13, O is on the triangle with vertices A_1 , A_2 , and A_3 . So the three black round nodes A_1 , A_2 and A_3 (●) are treated as the first ring surrounding O . The nodes on this ring should be the most possible enclosed in the ellipse than other nodes. The location ratio q_e is computed for each node of the ring to determine if these nodes are in the deformation area. Then create the second node ring surrounding the first ring. Each node of the second

ring is adjacent to the first ring. B_1 - B_9 , the blue triangles (), are treated as the second ring. We can see that some nodes of this ring, B_3 , B_6 and B_7 , are not in the ellipse. Continue to construct the expanded rings until no nodes of the ring is in the ellipse. Same thing happens on the third ring C_1 - C_{15} () that few nodes are enclosed in the ellipse. But D_1 - D_{13} () is such ring that does not has any node in the ellipse. So the deformation calculation stops. This algorithm is summarized in pseudocode as follows.

```

1:  SearchNodesInEllipse(frictionAnchor, ellipse)
2:      Determine the triangle  $ring[1]$  which the frictionAnchor lies on
3:      WHILE (true)
4:          WHILE traverse  $ring[i]$ 
5:               $q_e = \text{calculateLocationRatio}(\text{node}, \text{ellipse})$ 
6:              IF  $q_e < 1$  THEN          // in the ellipse
7:                  CALL  $\text{calculateNormalDeformation}(\text{node}, q_e)$ 
8:                  CALL  $\text{calculateTangentDeformation}(\text{node}, q_e)$ 
9:              ENDIF
10:         ENDWHILE
11:
12:         IF all  $q_e \geq 1$  THEN // none node of  $ring[i]$  is in the ellipse
13:             RETURN // stop searching the nodes enclosed by the ellipse
14:         ELSE
15:             Construct the next expended  $ring[i=i+1]$ 
16:         ENDIF
17:     ENDWHILE

```

This procedure is traversing the rings from the inner ring to outside. Line 7 and 8 are the main deformation calculation processes. The normal and tangential deformations at the vertex (node) are calculated separately and without affecting each other. The construction of the node ring, line 15, is computationally costly. In this algorithm, a new

vertices list containing all the vertices is created, any vertex has been assigned to the ring should be marked and will not be checked for the new ring. Once the traverse of a ring finishes, the next ring is created. The rings are stored in a link and the vertex indices are stored in a link of the ring.

The mesh in Figure 4.15 is simple with not many vertices since it is for explaining the concept only. The real surface of the model contains a large number of vertices so that most of the nodes of the model will be detected not in the deformation area and thus save much computational time.

To get a smooth looking surface of the haptic model, the normal of each vertex is pre-computed by averaging the normals of its adjacent triangles. If any triangle is changed, the normals of its vertices should be recalculated. A changed triangle list is created dynamically to handle the normal recalculation.

Figure 4.16 presents the deformation on the textured back model. An undeformable hand model is used instead of the proxy sphere. This hand model was created by MakeHuman and edited by 3DS Max. It has 1418 vertices and 2814 faces (triangles).

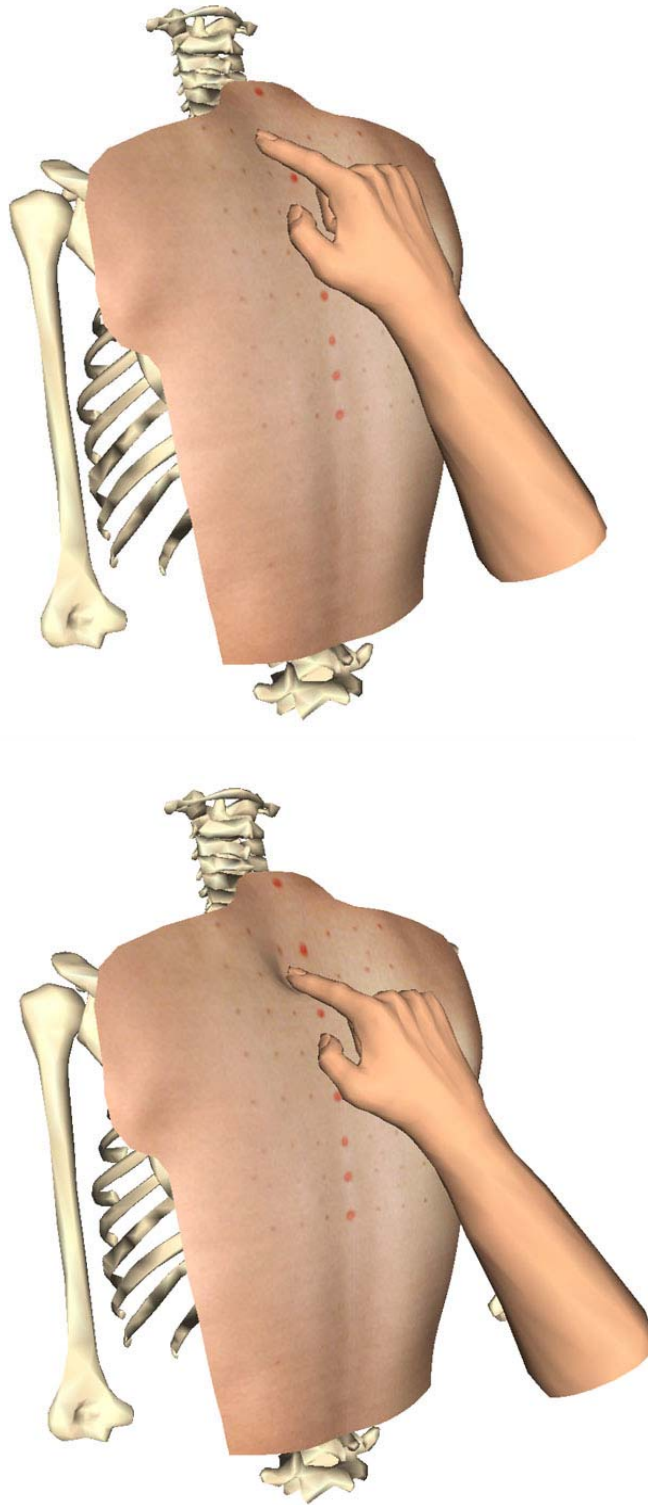


Figure 4.16 Deformation Combined with Normal and Tangential Components

The normal and tangential deformations are calculated separately and combined together vectorially. The tip of the index finger (the proxy position) is kept on the deformed surface

4.6 Maximum Deformation

Because of the displacement-driven force generation of the haptic interface, the haptic tip can be moved a significant displacement (say 20 mm) under a significant force (say 5 N) from the user when the proxy is touching the model. It is not realistic if the contact point always follows the position of the haptic tip. So the maximum displacement of the deformation is defined as follows.

Denote $m_{n-device}$ to be the displacement of the haptic interface relative to the friction anchor in the normal direction (penetration depth). As mentioned in Section 4.1, the maximum deformation in the normal direction d_{n-max} is defined as:

$$d_{n-max} = 10 \text{ mm} \quad (4.15)$$

When $m_{n-device}$ is less than d_{n-max} , the displacement of the contact point in the normal direction is changed linearly corresponding to the penetration depth of the haptic tip. If $m_{n-device}$ is greater than or equal to d_{n-max} , the displacement of the contact point in the normal direction is equal to 10 mm.

$$\left. \begin{array}{l} d_n = m_{n-device} \\ \text{if } m_{n-device} < d_{n-max} \end{array} \right\} \quad (4.16)$$

$$d_n = d_{n-max} \quad \text{if } m_{n-device} \geq d_{n-max}$$

The maximum tangential deformation d_{t-max} , is similar to the maximum normal deformation. We set it as follows.

$$d_{t-max} = 20 \text{ mm} \quad (4.17)$$

and

$$\left. \begin{array}{ll} d_t = m_{t-device} & \text{if } m_{t-device} < d_{t-max} \\ d_t = d_{t-max} & \text{if } m_{t-device} \geq d_{t-max} \end{array} \right\} \quad (4.18)$$

4.7 Visualization of Surface Strain

On the haptic model, color mapping is used to visualize the strain of the deformed surface. In Figure 4.17, the original mesh is presented in black solid lines. The mesh deforms as driven by the haptic interface. The blue vectors (arrows) present the displacements of each vertex. So the strain of point E can be calculated by averaging the strains of its neighboring points.

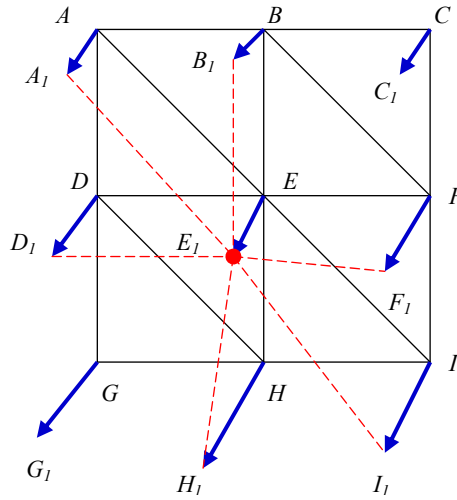


Figure 4.17 Average of the Neighboring Points Color Mapping For Visualization

E has 6 neighboring points $A, B, D, F, H,$ and I . Red lines show all the joint points of E . S_E denotes the final strain at E . S_{AE} denotes the strain at E in the AE direction.

$$S_{AE} = \frac{|A_1E_1| - |AE|}{|AE|} \quad (4.19)$$

Then $S_{BE}, S_{DE}, S_{FE}, S_{HE}, S_{IE}$ can be derived in the same manner as (4.19).

$$S_E = \text{Average}(S_{AE}, S_{BE}, S_{DE}, S_{FE}, S_{HE}, S_{IE}) \quad (4.20)$$

The color at vertex E is updated dynamically. S_E is mapped to a red-blue color diagram. The screen shot of the strain visualization is shown in Figure 4.18. The red area presents the part of the surface under tension and the intensity of the color indicates the intensity of the strain of this area. The blue area indicates the part of the surface under compression and the intensity of the strain.

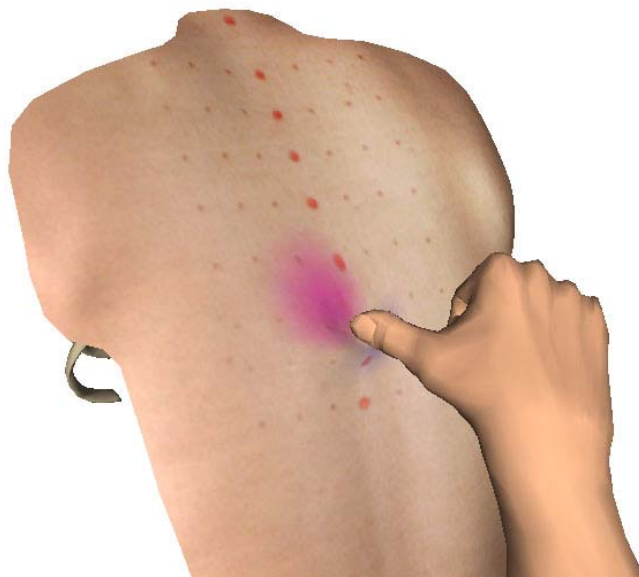


Figure 4.18 Strain Visualization of the Surface Deformation

5. Defined Deformation Boundary Shape

In the previous chapter, the curve matched deformation is defined continuously between the maximum and minimum deformed points. For standard elastic materials (linear or nonlinear), it is reasonable to set the maximum deformed point always at the contact point and the minimum deformed point is always on the boundary of the deformation (zero displacement). For materials with uniform properties and only normal external forces, the displacement contours on the surface should be concentric circles because the deformation is the same mapping on the deformation curve when the distance to the center (the contact point) is the same. So the deformation boundary is a circular contour with zero displacement.

On the human body, the skin is relatively thin and can be treated as an isotropic elastic membrane. Under the skin, the muscles and the bones are the main structures that affect the deformation. Muscle is a very elastic soft tissue [McKinley, 2005]. Because the lay of muscle (including tendon) is between the skin and the bone, and it is much thicker than skin, it is much more important part involved in the deformation than bone. In the anatomy picture of the human back (Figure 5.1) the muscles on the back are not isotropic and are located in different directions. The deformation contour on the muscle is not circle for this reason.

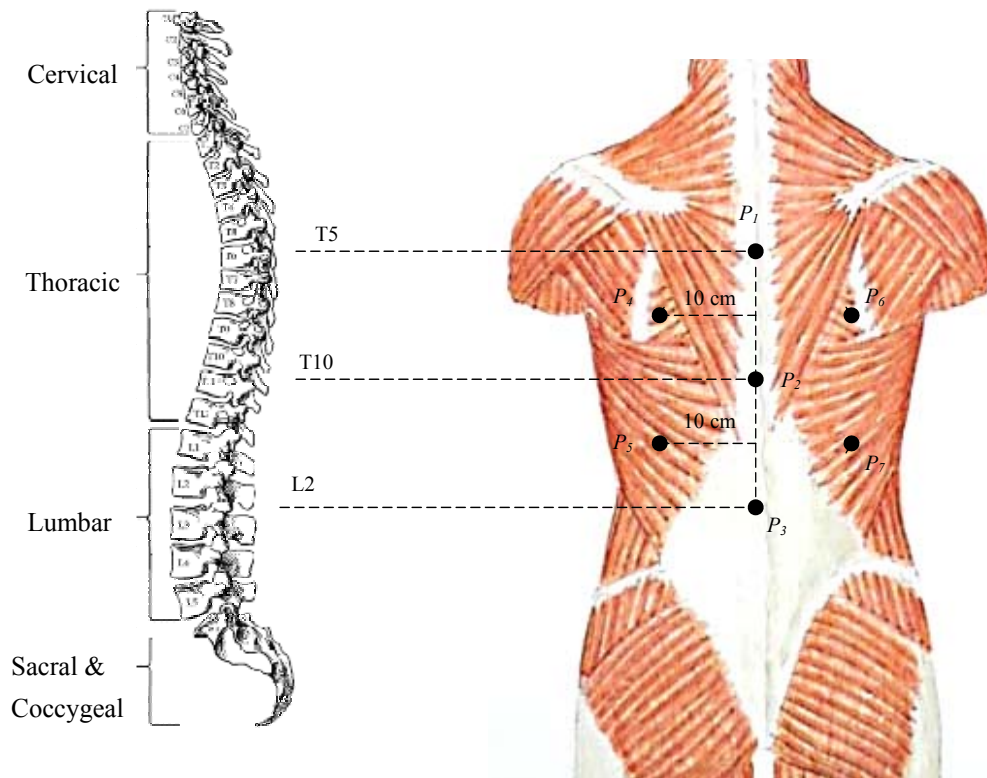


Figure 5.1 Human Back Muscles and Seven Key Points

(Back Muscle Picture is from www.dkimages.com)

In order to calculate the deformation with the shape matched algorithm, the deformation boundary needs to be predefined. The deformation boundaries vary at different points on the back depending on the muscle directions. So a boundary shape map has been created for each vertex of the haptic back model.

5.1 Deformation Shape Study of Human Back and Simplification Method

In deformation experiments with human subjects, we found that the shape of

tangential deformation on the skin under friction is irregular, not a circle or other regular shape. Few reports about the deformation characters of the soft tissue can be found in the literature. The physical properties of soft tissue and muscle vary and are dependent on gender, age, and body type, etc. When we studied the deformation on the human back, an obvious phenomenon of the deformation shape was observed: the deformation region along the direction of the muscle is usually greater than in other directions. So this feature can be considered as an ellipse instead of a circle because an ellipse has a major axis which is longer than the minor axis. The average data from several subjects are used for this research. The data can be adjusted or re-measured depending on the user's needs and the specified part of the human body. We measured the back deformations on five subjects.

Figure 5.1 shows the locations and the directions of the back muscles. It also shows the key points that are chosen for the measurement of the deformation shape.

The external back muscles mainly includes the trapezius (on both sides of back points P_1P_2 , see Figure 5.1), teres minor, teres major (around P_4 and P_6), and latissimus dorsi (around P_5 and P_7) [McKinley, 2005]. The trapezius and latissimus dorsi cover about 3/4 of the back. So, in Figure 5.1, P_1 , P_2 , P_3 , P_5 , and P_7 are marked as five typical deformation points. P_1 , P_2 and P_3 are on the mid line. They are located at thoracic vertebrae T5 and T10 and lumbar vertebra L2. P_5 and P_7 are on the perpendicular bisector of P_2P_3 and 10 cm from P_2P_3 . However, teres minor and teres major are small and

complex muscles. So P_4 and P_6 are marked in this area on both sides of the back. They are on the perpendicular bisector of P_1P_2 and 10 cm from P_1P_2 . Since P_4, P_6 and P_5, P_7 are two pairs of symmetrical points, they have similar deformation shape properties. Therefore, only five points P_1, P_2, P_3, P_4 and P_5 were measured.

Figure 5.2 presents the measurement method and the back deformation contours on a subject.

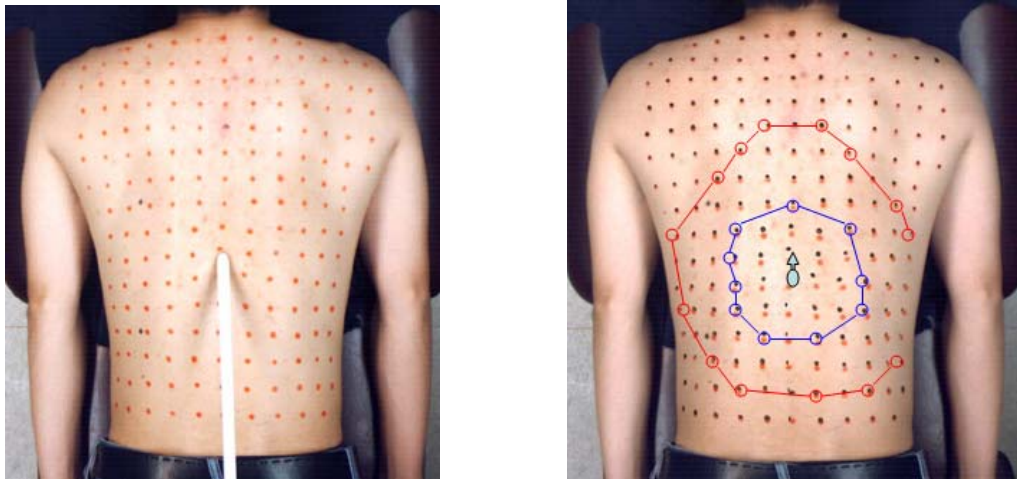


Figure 5.2 Two Deformation Contours at P_2

In this experiment, the subject was prone on a firm table. A fixed camera took pictures of the back from the top. The back of the subject was marked in red dots with a grid size of 3×3 cm. A stick (left picture) with a round tip, which is about the size of a finger, was used to apply the force to the back to avoid blocking the red dots from the camera. Two static pictures were taken separately, before and after the force was applied.

During this time, the subject was asked to hold his breath for about 5-10 seconds to keep the whole body steady under the force. After taking the first picture, the force was applied at the key point on the subject's back using the stick in 45 degrees to horizontal and towards the shoulder. The force direction is shown as a small light blue arrow in the right picture of Figure 5.2. The force execution was stopped until the stick could not move the skin at which time the second picture was taken. The experiment was repeated on the same subject, with the applied force in the opposite direction (toward the waist).

To view the displacement of each dot, these two pictures were compared by overlapping the dots of the second picture (only pick the red dots from the second picture) on the first picture. The black dots in the right of the Figure 5.2 are from the second picture. These dots were red originally and were changed to black to distinguish from the dots in the first picture. We identified the dots with the same displacements and connected them to be a displacement contour. All dots of the red contour represent 2 mm displacement. The blue contour is for reference, presenting the deformation shape around the contact point.

It is difficult to find the zero displacement contour because, in some cases, almost all points on the back skin were moved, even points far from the contact point. Therefore, when the zero-displacement points were connected by lines, the shape is not an ellipse but some irregular shape whose border may cross the neck, shoulder and waist. So we set a threshold, 2 mm, of the displacement to be considered as a zero deformation boundary.

The red polyline in Figure 5.2 is such a zero-deformation contour. The long axis of this shape was measured to be the major axis of the deformation ellipse, and the major axis direction was also recorded. The short axis perpendicular to the long axis was measured as the minor axis of the ellipse. Each ellipse is described such that the major axis is upward, within $\pm 90^\circ$ to the positive y -axis. The minor axis is within $\pm 90^\circ$ to the positive x -axis. The definition of the deformation ellipse at the key point, named P_i , is presented in the following format.

$$\text{KeyEllipse}[P_i] \{ x, y, z, r_i, t_i, d_i \} \quad (5.1)$$

where

x, y, z : Location of P_i , center of the ellipse

r_i : Half length of major axis

t_i : Ratio of major axis over minor axis

d_i : Direction (in degrees) of the major axis to the positive x -axis

The experimental human back deformation data is attached in Appendix C. The results of the deformation ellipses at $P_1, P_2, P_3, P_4,$ and P_5 are shown in Figure 5.3. They were computed by averaging the results of the measured data at each key point.

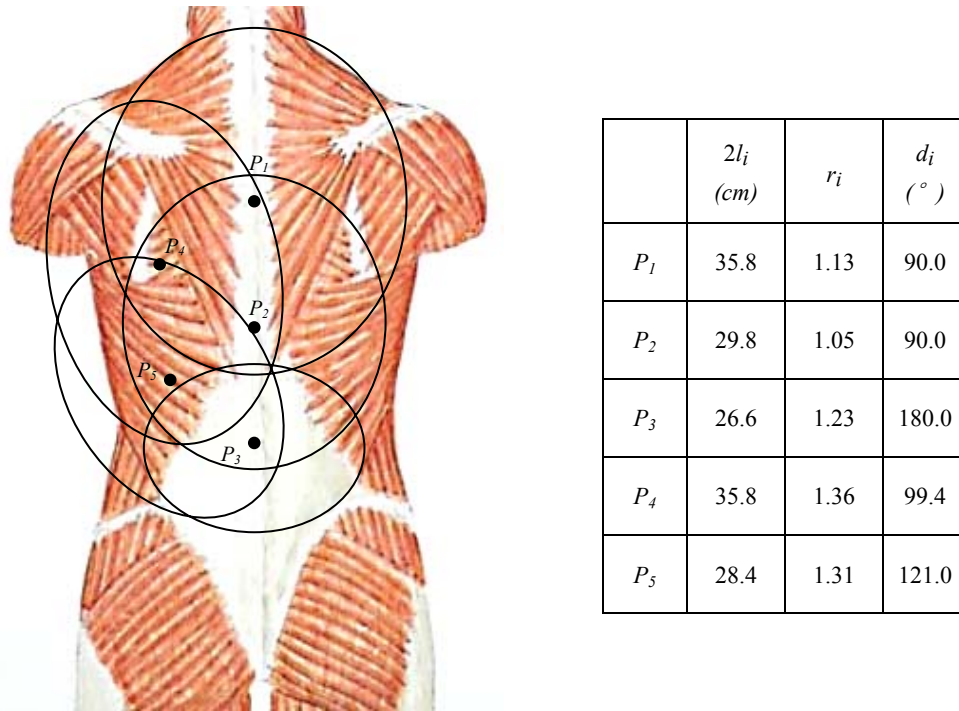


Figure 5.3 The Deformation Ellipses at the Key Points

(Back Muscle Picture is from www.dkimages.com)

The ellipses of P_6 and P_7 are assumed to be symmetrical to P_4 and P_5 . They were not tested so they are not presented in Figure 5.3. The deformation ellipses at the seven key points lie far from each other. When the deformation is at any point except key points, the ellipse has to be obtained by interpolation. Because the contact point must locate on a triangle combined with three vertices and the triangle size is very small (the edge is around 5 mm), the deformation ellipse at the contact point is considered equal to its nearest vertex. An ellipse map has been built for each vertex of the back mesh based on

the seven key points. Therefore each vertex of the back model has a deformation ellipse definition stored in the vertex struct. This map is created in pre-processing.

5.2 Interpolation of the Deformation Ellipses

The contact point is computed dynamically on the octree model. The deformation ellipse at the contact point should change during the movement of the haptic device. Since the neighboring vertices are already stored, the ellipse can be determined quickly by obtaining directly from its nearest vertex. The ellipse of each vertex is determined by the seven key points. The Gaussian Radial Interpolation (GRI) algorithm is developed to calculate the parameters of the deformation ellipse at each vertex of the back model. If no key points are defined in advance, the deformation boundary is always assumed as a circle.

5.2.1 Problem Statement

We have assumed that the physical changes are continuous on the haptic back model. So the changes of the deformation shapes are continuous and gradual. In the definition of the deformation ellipse (5.1), the length of major axis (r), ratio (t) and direction (d) are the three variables of the ellipse according to its location (x, y, z). All the ellipses are defined and computed in a 2D plane. They will be mapped on the 3D haptic

model at each vertex. Since few key points have been measured, the deformation ellipses in other locations need to be generated by GRI.

Before the interpolation, a uniform back with a constant deformation circular shape was assumed because we assume that the ellipses at the key points are the effect factors to a normal material with homogenous physical properties. Put the key ellipses on the uniform back (basic deformation map); the final ellipse at any point on the new map is then generated by the interpolation algorithm with the constant circle and the key (affecting) ellipses.

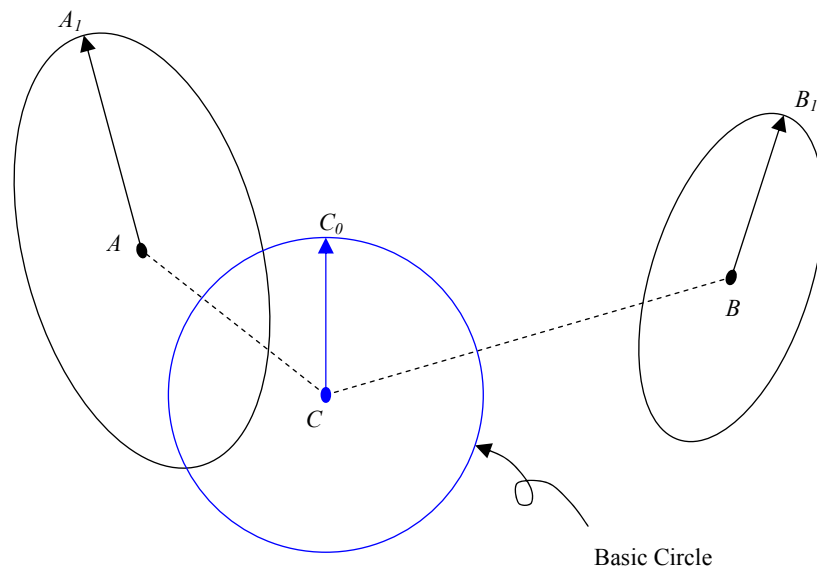


Figure 5.4 Key Deformation Ellipses in the Uniform Deformation Area

For example, in Figure 5.4, the blue circle with center C is the constant basic circle of the map. This pre-defined deformation circle (deformation shape base) is same and everywhere in the basic map. To get the new deformation shape at point C , we need

to calculate the shape change of the blue circle under the effects by ellipses A and B .

AA_1 and BB_1 are the vectors indicating the half major axis of ellipses A and B .

Assume that the measured ellipses at the key points are the true deformation boundaries. The new (final) deformation shape map should satisfy the following conditions.

- 1) The shapes at the key points are the same as the measured ellipses.
- 2) A key point affects other points only when they are in a certain range around the key point. The effects of the different key points can be stacked.
- 3) The effect level of the key point is from maximum to zero in a gradual decline trend in its range.
- 4) The shape at a point without any effect of the key point is the basic circle.

5.2.2 Radial Interpolation Based on Gaussian Curve

The definition of the circle C is similar to the deformation ellipse (5.1). The differences of circle C are that its ratio is always 1.0 and major axis direction is always 90° (upright, see vector CC_0 in Figure 5.4). The basic circle is expressed as follows.

$$\text{BasicCircle} \{x, y, z, r_c, t_c, d_c\} \quad (5.2)$$

where

x, y, z : Position of the point (vertex of the model)

r_c : Radius of the basic circle

$$t_c: \quad 1.0$$

$$d_c: \quad 90^\circ$$

The deformation shape definition at any desired point is similar to the key ellipse and presented in the following.

$$\textit{DeformationShape} \{ x, y, z, R, T, D \} \quad (5.3)$$

In the interpolation method, the effect of the key point is from maximum to zero, restricted in a limit range when the effect declines with increasing distance. In Figure 5.5, suppose that A and B are two key points that have been measured for the deformation boundary ellipses. C is a vertex of the back model. The interpolation algorithm will generate a new ellipse at C . Ellipse A is close to point C so that it affects C . But ellipse B is further from C , say somewhere outside of a circular range (the dashed green), so that the deformation shape at C should not be affected by the ellipse B . Let's define a circle with radius e to be the boundary of the effect region. e is constant on the haptic back model. It is an estimated value from the experiments.

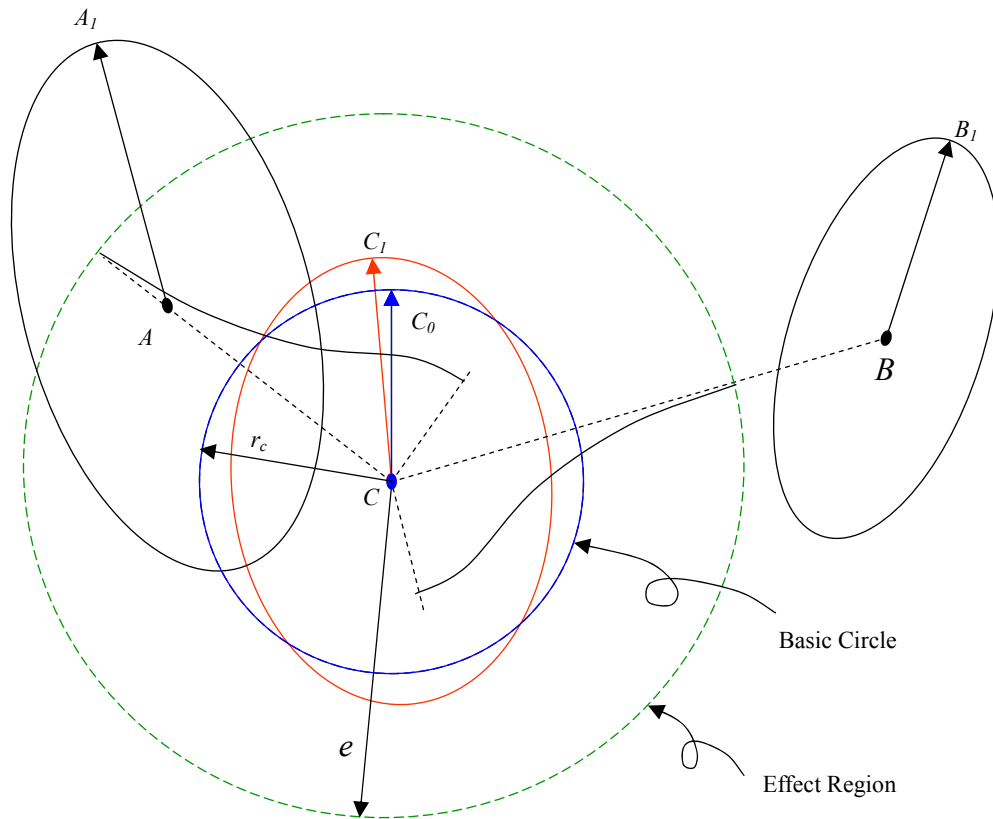


Figure 5.5 Effect Region Circle and Gaussian Decline Trend Curve

The blue (solid) circle at center C is the basic circle. The shape at this point becomes the red ellipse, the new deformation shape with major axis (half) CC_1 , under the effects of ellipses A and B . Because we can notice that ellipse B is out of the effect region, it should not affect the shape at C . But ellipse A is in the effect region and affects C . So the red final ellipse at center C is similar to ellipse A in size and direction. Since r , t and d are the individual parameters in the deformation shape definition, they are computed separately by the same interpolation algorithm.

To get smooth interpolation, a Gaussian Curve is used again to be the attenuation trend for the effect level of the key ellipse (see Figure 5.6).

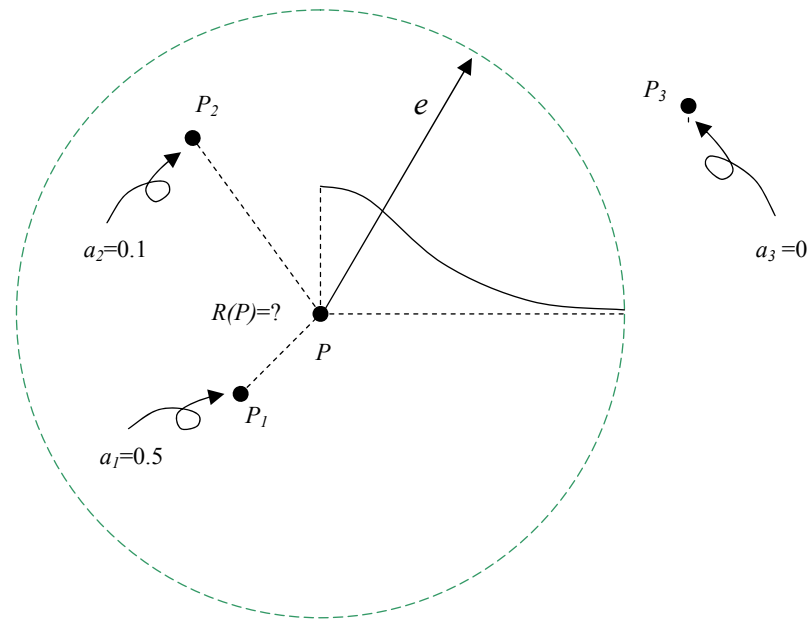


Figure 5.6 Effect Depending on the Distance

A Gaussian curve is drawn in solid from the center to the border of the effect region circle. P_1 and P_2 are in the effect region of center P so that they affect P . And P is closer to P_1 than P_2 . So P_1 affects P more than P_2 does; $a_1=0.5$ and $a_2=0.1$ indicate the effect levels. Since P_3 is out of the range, it does not affect P and a_3 is zero.

Figure 5.7 shows the Gaussian mapping within the effect range in Figure 5.6. Gaussian curve array is used again. The ratio of the distance over the effect radius is mapped to the index of the array to get the effect level.

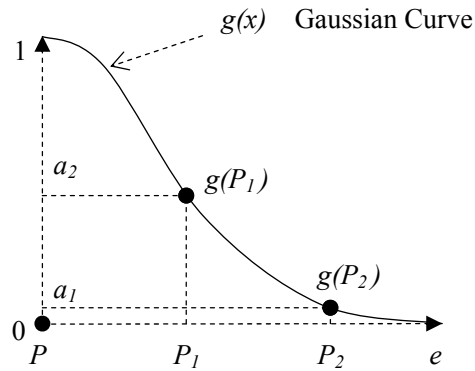


Figure 5.7 Gaussian Curve Mapping for the Effect Level

Denotations:

$a_1 = g(P_1)$ Effect level from P_1 at P according to the distance of PP_1 mapped on the Gaussian curve

$a_2 = g(P_2)$ Effect level from P_2 at P according to the distance of PP_2 mapped on the Gaussian curve

r_c Radius of the basic circle (constant)

$r_i = r(P_i)$ Half length of the major axis of the ellipse at the key point P_i

V_i Half length of the major axis at P only affected by key point P_i

$R(P)$: Final result of the half length of the major axis at P under the effects of the all key points ($P_1, P_2, P_3 \dots\dots$)

The major axis interpolation is derived as follow.

To get the result V_i at point P with the effect from P_i , between two points, the linear interpolation is suitable for this situation. The distance between P and P_i is the

effect factor and be mapped to the normalized Gaussian curve.

If $|PP_I| > e$ (effect radius), P_I does not affect P .

If $|PP_I| \leq e$, P_I affects P corresponding to the Gaussian curve show in Figure 5.7.

The linear interpolation is written as follow.

$$V_I = r_c(1 - a_1) + r_1 a_1 \quad (5.4)$$

In this case, there is only one effect point locating in the deformation effect circle.

When there are more than one effect points, the final value V is a combination of all the effect points depending on the effect weight of each point. Apply the linear interpolation again to compute the final major axis:

$$R(P) = V_1 \frac{a_1}{a_1 + a_2 + \dots + a_n} + V_2 \frac{a_2}{a_1 + a_2 + \dots + a_n} + \dots + V_n \frac{a_n}{a_1 + a_2 + \dots + a_n} \quad (5.5)$$

For example, in Figure 5.6, there are two deformation ellipses P_1 and P_2 in the deformation effect circle P and one deformation ellipse P_3 is out of this circle. The half length of the major axis of final deformation shape (ellipse) is obtained as follows.

$$\begin{aligned} R(P) &= [r_c(1 - a_1) + r_1 a_1] \frac{a_1}{a_1 + a_2} + [r_c(1 - a_2) + r_2 a_2] \frac{a_2}{a_1 + a_2} \\ &= \frac{1}{a_1 + a_2} [r_c(a_1 - a_1^2) + r_1 a_1^2 + r_c(a_2 - a_2^2) + r_2 a_2^2] \\ &= \frac{1}{a_1 + a_2} \{ r_c [a_1 + a_2 - (a_1^2 + a_2^2)] + r_1 a_1^2 + r_2 a_2^2 \} \end{aligned} \quad (5.6)$$

The above derivation just involves two ellipses. For an arbitrary case with any number (n) of points, R is:

$$R(P) = \sum_{i=1}^n \left\{ [r_c(1 - a_i) + r_i a_i] \frac{a_i}{\sum_{j=1}^n a_j} \right\} \quad (5.7)$$

$$\text{Or } = \frac{1}{\sum_{i=1}^n a_i} \left[r_c \left(\sum_{j=1}^n a_j - \sum_{j=1}^n a_j^2 \right) + \sum_{j=1}^n (r_j a_j^2) \right] \quad (5.8)$$

Substituting $g(P_i)$ for a_i in (5.7) gives:

$$R(P) = \sum_{i=1}^n \left\{ [r_c(1 - g(P_i)) + r_i g(P_i)] \frac{g(P_i)}{\sum_{j=1}^n g(P_j)} \right\} \quad (5.9)$$

Substituting $r(P_i)$ for r_i in (5.9), we get:

$$R(P) = \sum_{i=1}^n \left\{ [r_c(1 - g(P_i)) + r(P_i)g(P_i)] \frac{g(P_i)}{\sum_{j=1}^n g(P_j)} \right\} \quad (5.10)$$

This is the result about the half major axis of the effected ellipse at the desired point P . But for efficient computation, the equation is chosen by the previous one (5.9).

A problem was found in this algorithm. The result at the key point is not equal to its original values (key ellipse). For instance, at P_1 , its original length of the half major axis is 120. The effect level from P_1 (itself) is 1.0 because effect distance is 0. And then, at this point, we expect the result 120. But the equation (5.10) gives 112. That is not correct because in the satisfied conditions of the interpolation, the key point should keep its original value. In this case, the effect from P_1 is full effect to itself. The effects from P_2

(say, effect level = 0.26) and P_3 (say, effect level = 0.18) should affect P_I . The previous method needs to be adjusted.

When the effect factor of P_I is 1, the effects from other points should be 0. But the sum of the effects should be still 1. To achieve this goal, some correction factors are added to each element of the numerator and denominator in (5.5); see the following derivation.

$V_I \frac{a_1}{a_1 + a_2 + \dots + a_n}$ is modified to

$$V_I \frac{a_1(1-a_2)(1-a_3)\dots(1-a_n)}{a_1(1-a_2)(1-a_3)\dots(1-a_n) + a_2(1-a_1)(1-a_3)\dots(1-a_n) + \dots + a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})} \quad (5.11)$$

and $V_2 \frac{a_2}{a_1 + a_2 + \dots + a_n}$ is modified to

$$V_2 \frac{a_2(1-a_1)(1-a_2)\dots(1-a_n)}{a_1(1-a_2)(1-a_3)\dots(1-a_n) + a_2(1-a_1)(1-a_3)\dots(1-a_n) + \dots + a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})} \quad (5.12)$$

.....

and $V_n \frac{a_n}{a_1 + a_2 + \dots + a_n}$ is modified to

$$V_n \frac{a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})}{a_1(1-a_2)(1-a_3)\dots(1-a_n) + a_2(1-a_1)(1-a_3)\dots(1-a_n) + \dots + a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})} \quad (5.13)$$

All the denominators in (5.11), (5.12), and (5.13) are same. Only their numerators are different. The sum of the fractions is equal to 1.

In (5.11), at point P_n , the numerator is 0 because the last element $(1-a_n)$ is 0. So the result of (5.11) is zero. In the same manner, (5.12) gives zero too. But in (5.13), the

numerator is non-zero and the denominator just has the same thing left as the numerator (the other elements are 0 in the manner described). So the fraction in (5.13) is 1. The result about (5.11) and (5.12) all give 0. Only in (5.13) is $V_n * 1 = V_n$. This way guarantees the interpolation result at the specified point keeping to its original value and reducing the effects from other point to zero.

Use a denotation to simplify the expression of the element in the former equations:

$$\prod_{j=1; j \neq i}^n (1 - a_j) = (1 - a_1)(1 - a_2) \dots (1 - a_{j-1})(1 - a_{j+1}) \dots (1 - a_n) \quad (5.14)$$

where $i \in [1, n]$,

This denotes a continuous multiplication operation without including the *ith* element. For example,

$$a_1 \prod_{j=1; j \neq 1}^n (1 - a_j) = a_1 (1 - a_2)(1 - a_3) \dots (1 - a_n)$$

$$a_2 \prod_{j=1; j \neq 2}^n (1 - a_j) = a_2 (1 - a_1)(1 - a_3) \dots (1 - a_n)$$

$$a_n \prod_{j=1; j \neq n}^n (1 - a_j) = a_n (1 - a_1)(1 - a_2) \dots (1 - a_{n-1})$$

So (5.11) can be modified to:

$R(P) =$

$$\begin{aligned}
& V_1 \frac{a_1(1-a_2)(1-a_3)\dots(1-a_n)}{a_1(1-a_2)(1-a_3)\dots(1-a_n) + a_2(1-a_1)(1-a_3)\dots(1-a_n) + \dots + a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})} + \\
& V_2 \frac{a_2(1-a_1)(1-a_2)\dots(1-a_n)}{a_1(1-a_2)(1-a_3)\dots(1-a_n) + a_2(1-a_1)(1-a_3)\dots(1-a_n) + \dots + a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})} + \dots + \\
& V_n \frac{a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})}{a_1(1-a_2)(1-a_3)\dots(1-a_n) + a_2(1-a_1)(1-a_3)\dots(1-a_n) + \dots + a_n(1-a_1)(1-a_2)\dots(1-a_{n-1})} \\
& = V_1 \frac{a_1 \prod_{j=1; j \neq 1}^n (1-a_j)}{a_1 \prod_{j=1; j \neq 1}^n (1-a_j) + \dots + a_n \prod_{j=1; j \neq n}^n (1-a_j)} + \dots + V_n \frac{a_n \prod_{j=1; j \neq 1}^n (1-a_j)}{a_1 \prod_{j=1; j \neq 1}^n (1-a_j) + \dots + a_n \prod_{j=1; j \neq n}^n (1-a_j)} \\
& = \sum_{i=1}^n \left(V_i \frac{a_i \prod_{j=1; j \neq i}^n (1-a_j)}{a_1 \prod_{j=1; j \neq 1}^n (1-a_j) + \dots + a_n \prod_{j=1; j \neq n}^n (1-a_j)} \right) \tag{5.15}
\end{aligned}$$

Substituting $[r_c(1-a_i) + r_i a_i]$ for V_i in (5.15) gives

$$R(P) = \sum_{i=1}^n \left([r_c(1-a_i) + r_i a_i] \frac{a_i \prod_{j=1; j \neq i}^n (1-a_j)}{a_1 \prod_{j=1; j \neq 1}^n (1-a_j) + \dots + a_n \prod_{j=1; j \neq n}^n (1-a_j)} \right) \tag{5.16}$$

Substitute $g(P_i)$ for a_i in (5.16), we get

$$R(P) = \sum_{i=1}^n \left(\frac{[r_c(1-g(P_i)) + r_i g(P_i)] \prod_{j=1, j \neq i}^n (1-g(P_j))}{\sum_{k=1}^n \left(g(P_k) \prod_{j=1, j \neq k}^n (1-g(P_j)) \right)} \right) \quad (5.17)$$

The following example uses (5.7) and (5.16) for the interpolations. The results are compared. In Figures 5.8, in a square area with size 500×500 mm, there are three deformation boundary shape ellipses E_1 , E_2 and E_3 at key points O_1 , O_2 and O_3 on the y -axis. And the z -axis (outward from the paper, not presented in the figure) indicates the length of half the major axis of the deformation ellipse. As we assumed before, in this square, it is a uniform base initialized with a constant deformation circle. In this example, the value of the basic circle radius is presented in z -axis. In the same manner, the lengths of major-axis of the ellipses are presented in z -axis also

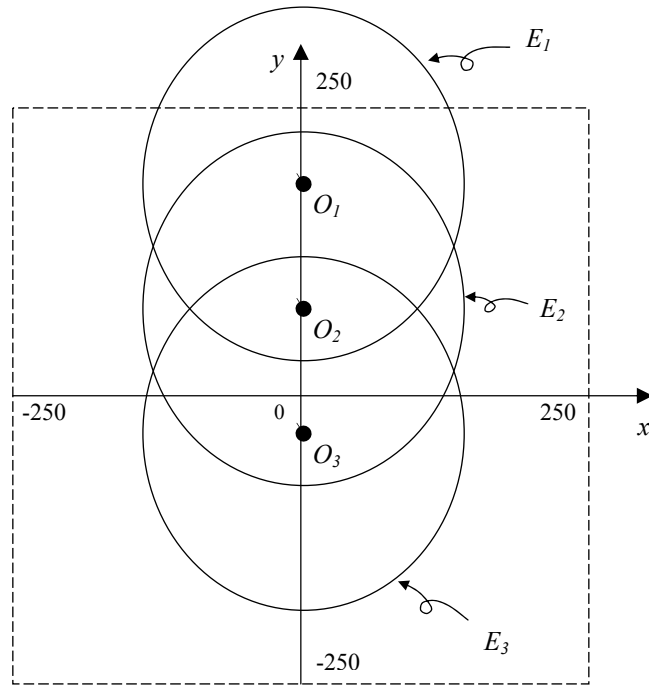


Figure 5.8 Effect Ellipses in a Square Area

Table 5.1 list the locations and the lengths of the major-axis of the ellipses in Figure 5.8.

Table 5.1 Major Axes of the Ellipses at Key Points

Ellipse	Center (x, y)	Half Major Axis (z)
E_1	$O_1(0, 150)$	120
E_2	$O_2(0, 60)$	120
E_3	$O_3(0, -30)$	120

Suppose that the radius of the basic circle is 90 and the effect region radius is 400.

$$r_c=90, \quad e=400$$

Use the method of (5.7) to interpolate with these three key ellipses in the uniform 500×500 mm area; we got the result shown in Figure 5.9. The value in the z-axis indicates the result of the half major axis value in this square area. The right picture is the view seen along x-axis (projected on y-z plane). We can see that $R(E_1)=115$, but it should be 120 as its original value. And $R(E_2)=112$, but it should be 120 also.

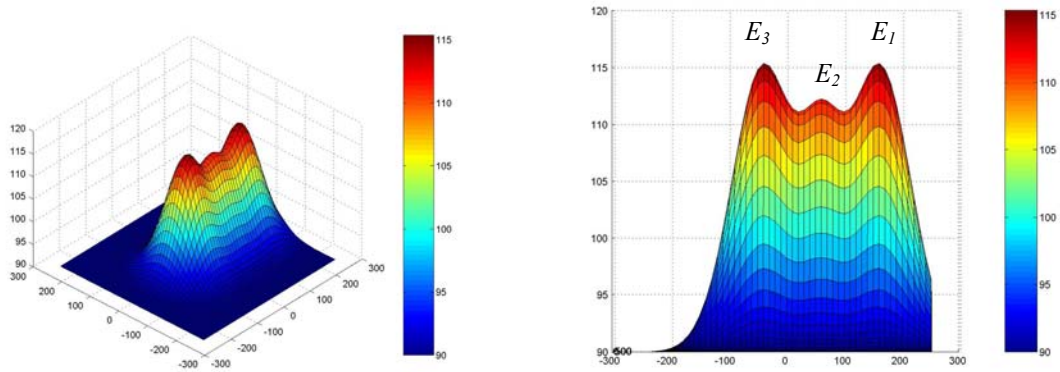


Figure 5.9 Results do not Fit the Key Points

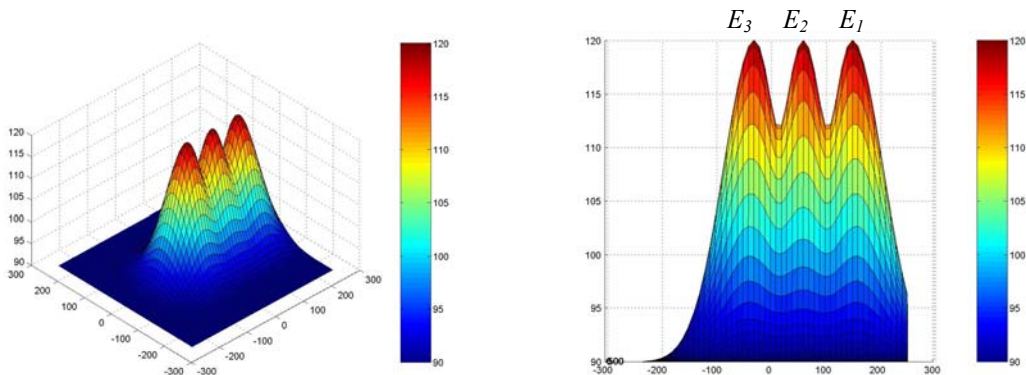


Figure 5.10 Interpolated Results Satisfies All the Key Points

Figure 5.10 shows the results after using the interpolation of (5.16). The results at E_1 , E_2 , and E_3 are 120 and all keep their original values at the key points. This interpolation can generate the map with accurate match at the key points and continuous gradient of z .

The algorithm of (5.16) is applied to the deformation simulation in this dissertation research. Figure 5.11 presents a map of the length of the half major axis of the deformation shape on a real back. It is based on seven key measured points. Red area indicates that the half major axis is greater than the normal value (base value), and the blue area around the waist means that the major axis is less than the base value.

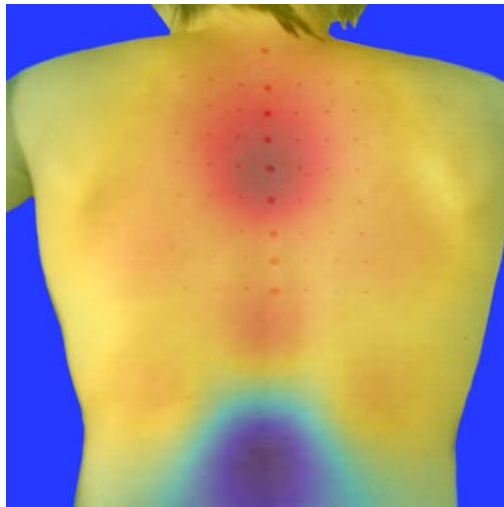


Figure 5.11 Map of Major Axis on Real Back Based on 7 Key Points

Since the last three components (r , t , and d) of the deformation shape are defined separately, the maps of ratio (t) and direction (d) are created separately too. These maps

are pre-computed and static in the haptic simulation. Each vertex of the model contains the deformation ellipse information converted from these maps.

5.3 Ellipse Conversion from Deformation Shape Map to 3D Model

The deformation ellipse at each vertex of the model is converted from the deformation ellipse map and stored. At the vertex, the ellipse plane is perpendicular to the normal as presented in Figure 4.13. The ellipse is stored in the following format.

$$EllipseAtVertex\{Position, Normal, HalfMajorAxis, Ratio\} \quad (5.17)$$

where

Position: Vertex location (x, y, z)

Normal: Normalized normal vector (3D) at the vertex

HalfMajorAxis: Half major axis vector (3D) of the ellipse

Ratio: Major axis over minor axis of the ellipse, real number

Since the deformation ellipse map is created in 2D plane, the ellipse at 3D vertices needs to be converted from the map. In the ellipse definition (5.17), *Ratio* is a real number so that it can be used directly, only *HalfMajorAxis* (*OA* in Figure 4.13 of the ellipse needs to be converted from the map. Figure 5.12 shows the projection relationship between the model and the deformation shape map.

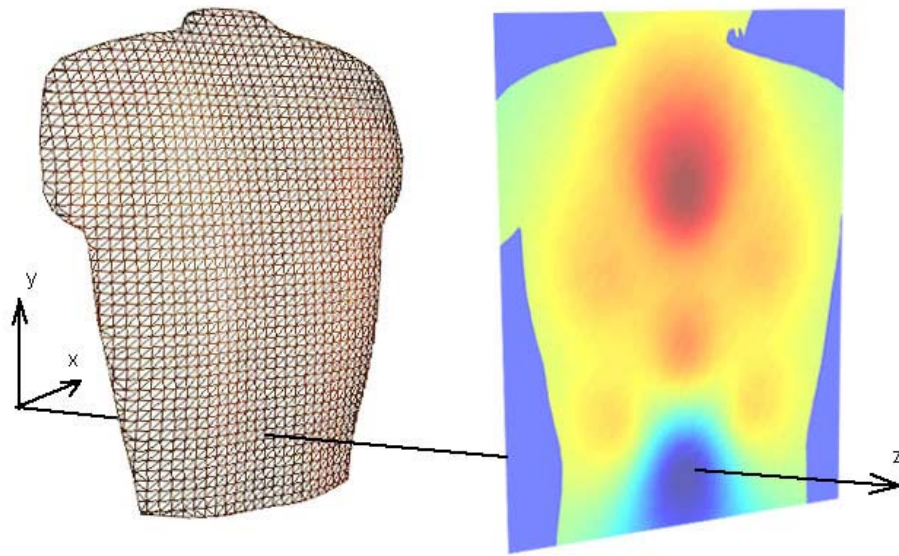


Figure 5.12 Projection from Model to Deformation Map Plan

The right image of Figure 5.12 is the deformation shape map which is perpendicular to the z -axis. The ellipse conversion procedure is model→map→model.

First, a vertex V_m of the model is projected, along the positive z -axis, on the map named P_e . And then the ellipse, defined in (5.1), at P_e is calculated on the map by using the GRI algorithm discussed before.

Second, calculate the vector *HalfMajorAxis* of (5.17) by using the half length major axis and direction of (5.1). This is a conversion from the 2D map to a 3D model. The half major axis of the ellipse on the map can also be expressed as a vector. Project this vector, along the negative z -axis, on the tangent plane at V_m of the model. The new vector on the tangent plane is considered as *HalfMajorAxis*.

6. Conclusions and Future Work

6.1 Summary and Conclusion

In this dissertation, the current deformation algorithms in graphics modeling are summarized. We then present a new algorithm FSMD about the deformation modeling for haptic applications. The computation of deformation is always based on the movement of the contact point when the haptic interface touches the model. Since the haptic modeling requires real-time response, to achieve faster interaction, BSP and octree have been implemented to divide the model into subspaces. The performances of both partitioning methods were analyzed and compared. In the haptic rendering and contact point detection, octree performs better in time complexity than BSP so that it is considered best to be applied to the VHB project. Then a fast search algorithm is described with HDB to find the contact point based on the octree model. So The octree optimizes the data structure of the haptic model and increases the haptic rendering performance significantly.

To develop a new deformation method on the haptic model, the deformation characteristics of the human back were measured and analyzed. Since the Gaussian curve has some features that are similar to the deformation of a continuous elastic material, it is employed for the FSMD and the deformation shape map generation. FSMD is discussed in details of normal and tangential deformations. The displacement of each deformed

vertex is along the direction of the haptic interface penetration. The mapping method on the predefined deformation curve is simplified. We explained the anisotropic characteristics of the back model because of the different directions of back muscles. So an elliptical deformation boundary is proposed. The distance ratio to the contact point is then used to map on a Gaussian curve. The vertex of the 3D model needs to be projected on its deformation ellipse plane. The projection result of the derivation from the vertex location to the Gaussian Curve is not complicated. Finally, the computation time is satisfied on VHB. The haptic rendering rate on the model with 20389 triangles was recorded as 26 FPS on average (the program was executed on a desktop with a P4 3.4GHz CPU, 2GB Memory, and GeForce6500 256MB video card). For the pre-computed deformation map, an accurate interpolation algorithm was suggested. We defined and measured seven key points on human backs to generate a big map with the deformation ellipse of each vertex of the model. This interpolation method can guarantee that the map passes every key point and changes in a smooth gradient. This deformation algorithm is based on some simplified algorithms and the measured data from real subjects. The deformation boundary shape and the deformation curve can be specified on different part of the body depending on the measurement results. The advantages of FSMD are fast, measurement data based, and adaptable. Disadvantages are linear elasticity simulation, no affections from bones.

6.2 Recommendations for Future Work

A few possible goals are suggested for future work.

1) The deformation curve can be mapped combined with skin and contour lines of the bones because the bones might affect the deformation cross-section when the surface is close to bone. We found that the normal deformation boundary is smaller than the tangential deformation boundary. For the normal deformation computation, the ellipse size can be reduced. That also decreases the number of the vertices involved in the deformation ellipse and saves computation time.

2) Since human tissue displacement is nonlinear, nonlinear stiffness should be used. A Gaussian Curve might be used to present the trend of the stiffness change from zero to maximum displacement at the contact point.

3) The deformation boundary shape is defined as an ellipse in this dissertation. It is a regular shape. But the real boundary is dependant on the shoulders, arms, neck, and waist because the skin is easily moved under the palpation forces even when the touch point is far from the deformation area of interest. The deformation shape map should include these effects. At each vertex the deformation boundary may be irregular. A new projection method is needed to convert the deformation shape from the map to the vertex.

4) The efficiency of haptic rendering might be improved if a linear link to save the triangles intersected with the HDB is added. A hybrid octree can be considered for this.

References

- T. Akenine-Möller and E. Haines, *Real-Time Rendering, Second Edition*, A K Peters, Ltd., pp. 662-664, 2002.
- A. Al-Ihalifah and D. Roberts, "Survey of Modeling Approaches for Medical Simulators", 5th Intl. Conf. Disability, Virtual Reality & Assoc. Tech., Oxford, 2004.
- B. Allen, B. Curless, and Z. Popovic, "Articulated body deformation from range scan data", *Transactions on Graphics*, SIGGRAPH 2002, pp. 612-619.
- E. Angel, *Interactive Computer Graphics, Fourth Edition*, Addison-Wesley, 2005, pp. 541-545.
- O.R. Astley and V. Hayward, "Design Constraints for Haptic Surgery Simulation", Proc. IEEE Int'l Conf. Robotics & Automation, April 2000.
- Y. Bando, T.i Kuratate, and T. Nishita, "A Simple Method for Modeling Wrinkles on Human Skin", Pacific Conference on Computer Graphics and Applications 2002, pp.166-175.
- K-J. Bathe, *Finite Element Procedures*, Prentice Hall, Englewood Cliffs, 1996.
- J. Barbic and D. James, "Real-Time Subspace Integration for St.Venant-Kirchhoff Deformable Models", ACM SIGGRAPH 2005, Volume 24, Issue 3, July 2005, pp.982-990.

- M. d. Berg, M. v. Krefeld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications, Second Edition*, Springer, pp. 256-262, 2000.
- A. de Boer, M. S. van der Schoot and H. Bijl, "Mesh deformation based on radial basis function interpolation", *Computers and Structures*, Volume 85, Issue 11-14, pp.784-795, 2007.
- J. Bonet and R. D. Wood, *Nonlinear Continuum Mechanics for Finite Element Analysis*, Cambridge University, 1997.
- P. Borrel, A. Rappoport, "Simple Constrained Deformations for Geometric Modeling and Interactive Design", *ACM Transactions on Graphics*, 13(2), pp. 137-155, April 1994.
- R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," Proc. of SIGGRAPH'02, San Antonio, pp. 594-603.
- M. Bro-Nielsen, "Finite Element Modeling in Surgery Simulation", *Proceedings of the IEEE*, Vol.86, No.3, pp. 490-503, 1998.
- P. Bourke, "Intersection of a plane and a line",
<http://ozviz.wasp.uwa.edu.au/~pbourke/geometry/planeline/>, 1999.
- S. R. Buss, *3-D Computer Graphics-A Mathematical Introduction with OpenGL*, Cambridge University Press, 2003.
- S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovi, "Interactive Skeleton-Driven

Dynamic Deformations”, SIGGRAPH 2002, July.

H. Chen, H. Sun, and X. Jin, "Interactive Haptic Deformation of Dynamic Soft Objects", ACM International Conference on VRCIA, pp. 255-261, 2006.

H. Chen, W. Wu, H. Sun, and P. Heng, "Dynamic touch-enabled virtual palpation", *Computer Animation and Virtual Worlds*, Volume 18, Numbers 4-5, pp. 339-348, September 2007.

M.-Y. Chen, R.L. Williams II, R.R. Conatser Jr and J.N. Howell, “The Virtual Movable Human Upper Body for Palpatory Diagnostic Training”, SAE Digital Human Modeling Conference, July 2006.

N. Chin, and S. Feiner, "Near Real-Time Shadow Generation Using BSP Trees", *Computer Graphics* (SIGGRAPH '89 Proceedings), 23(3), pp. 99-106, July 1989.

K. S. Choi, H. Sun, and P. A. Heng, "Interactive Deformation of Soft Tissues with Haptic Feedback for Medical Learning", *IEEE Transactions on Information Technology in Biomedicine*, Vol.7, No.4, pp. 358-363, December 2003.

Y. Chrysanthou and M. Slater, "Computing Dynamic Changes to BSP trees", *Computer Graphics Forum* (EUROGRAPHICS '92 Proceedings), 11(3), pp.321-332, Sep 1992.

T. H. Cormen, C. E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms, Second Edition*, MIT Press, 2001.

S. Cotin, H. Delingette, and N. Ayache, "Real-Time Elastic Deformations of Soft Tissues

- for Surgery Simulation", *IEEE Transactions on Visualization and Computer Graphics*, 1999.
- A. de Boer, M. S. van der Schoot, and H. Bijl, "Mesh deformation based on radial basis function interpolation", *Computers and Structures*, Volume 85, Issue 11-14, pp.84-795, June 2007.
- F. Dachille, H. Qin, and A. Kaufman, "A Novel Haptics-Based Interface and Sculpting System for Physics-Based Geometric Design", *Computer-Aided Design*, pp. 403-420, 2001.
- F. Dong, G. J. Clapworthy, M. A. Krokos, and J. Yao, "An Anatomy-Based Approach to Human Muscle Modeling and Deformation", *Visualization and Computer Graphics*, Volume 8, Issue 2, pp. 154-170, 2002.
- C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments", *IEEE Transactions on Visualization and Computer Graphics*, 12(1), pp. 36-47, 2006.
- D. H. Eberly, *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics, Second Edition*, Morgan Kaufmann, 2006, pp. 479-481.
- S. R. Eskola, "Binary Space Partitioning Trees and Polygon Removal in Real Time 3D Rendering", Uppsala University, Thesis, 2001.
- L. V. Fausett, *Numerical Methods: Algorithms and Applications*, Prentice Hall, pp.68-310, 2002.

- S. F. Frisken and R. N. Perry, "Simple and Efficient Traversal Methods for Quadtrees and Octrees". *Journal of Graphics Tools*, 7(3), 2002.
- N. Galoppo, S. Tekin, M.A. Otaduy, M. Gross, and M. C. Lin, "Interactive Haptic Rendering of High-Resolution Deformable Objects", *Virtual Reality*, pp. 215-223, August 2007.
- W. Gellert, S. Gottwald, M. Hellwich, H. Kastner and H. Kunstner, *VNR Concise Encyclopedia of Mathematics, 2nd ed*, New York: Van Nostrand Reinhold, pp.539-543, 1989.
Or refer <http://mathworld.wolfram.com/HessianNormalForm.html>.
- M. Gervautz and W. Purgathofer, "A Simple Method for Color Quantization: Octree Quantization," in *Magenat-Thalmann and Thalmann*, pp. 219-231, 1988.
- S. F. Gibson, B. Mirtich, "A survey of deformable models in computer graphics", *Technical Report TR-97-19, MERL*, Cambridge, MA, 1997.
- X. Guo, H. Qin, "Real-time Meshless Deformation", *Computer Animation and Virtual Worlds*, Vol. 16, No. 3-4, pp. 189-200, 2005.
- A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "A Framework for Fast and Accurate Collision Detection for Haptic Interaction", *IEEE Virtual Reality Conference*, pp.38-45, 1999.
- J. Griessmair and W. Purgathofer, "Deformation of Solids With Trivariate B-splines",

Eurographics, pp. 137-148, 1989.

M. Hauth, J. Gross, W. Straber, and G. F. Buess, "Soft Tissue Simulation Based on Measured Data", *Medical Image Computing and Computer Aided Intervention*, pp.262-270, 2003.

G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs, "An Implicit Finite Element Method for Elastic Solids in Contact", *Computer Animation*, 2001.

J. N. Howell, R. R. Conatser Jr., R. L. Williams II, J. M. Burns, and D. C. Eland, "The Virtual Haptic Back (VHB): A Teaching Tool for Palpatory Diagnosis", *FASEB Journal*, 21 (5 – part 1): A594, 2007.

J. N. Howell, R. L. Williams, R. R. Conatser, J. M. Burns, and D. C. Eland, "Training for Palpatory Diagnosis on the Virtual Haptic Back: Performance Improvement and User Evaluations", *The Journal of the American Osteopathic Association*, (AOA) 2007.

W. M. Hsu, J. F. Hughes, and H. Kaufman, "Direct manipulation of free-form deformations", SIGGRAPH 1992, v26, pp. 177-184.

G. Irving, J. Teran and R. Fedkiw, "Invertible Finite Elements for Robust Simulation of Large Deformation", ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA), pp. 131-140, 2004.

C. L. Jackins and S. L. Tanimoto, "Oct-trees and Their Use in Representing Three Dimensional Objects", *Computer Graphics and Image Processing*, 14(3), pp.249-270, 1980.

- D. L. James and K. Fatahalian, "Precomputing interactive dynamic deformable scenes", *Proc. ACM SIGGRAPH 2003*, pp. 879-887.
- D. L. James and D. K. Pai, "ArtDefo, Accurate Real Time Deformable Objects", *Computer Graphics (ACM SIGGRAPH 99 Conference Proceedings)*, pp. 65-72, August 1999.
- W. Ji, R. L. Williams II, J. N. Howell and R. R. Conatser, "3D Stereo Viewing Evaluation for the Virtual Haptic Back Project", *Proceedings of the Symposium on Haptic Interfaces for Virtual Environment*, 2006.
- P. Jimenez, F. Thomas, and C. Torras, "3D Collision Detection - A Survey", *Computers and Graphics*, 25(2), 2001.
- P. G. Kry, D. L. James, and D. K. Pai, "Eigenskin: Real time large deformation character skinning in hardware", *ACM SIGGRAPH 2002 Symposium on Computer Animation*, pp. 153-160.
- L. Kavan, J. Zara, "A Real-Time Deformation of Articulated Models", *Symposium on Interactive 3D Graphics and Games*, 9-16, 2005.
- J. P. Kim and J. Ryu, "Hardware Based 2.5D Haptic Rendering Algorithm using Localized Occupancy Map Instance", *ICAT*, pp. 132-137, 2004.
- A. Knoll, "A Survey of Octree Volume Rendering Methods", *Scientific Computing and Imaging Institute, University of Utah*, 2006.

- K. Komatsu, "Human Skin Model Capable of Natural Shape Variation," *The Visual Computer*, vol. 4, no. 3, pp. 265-271, 1988.
- Y. Kuroda, M. Nakao, T. Kuroda, H. Oyama, M. Komori, and T. Matsuda, "Interaction Model between Elastic Objects for Accurate Haptic Display", Proceedings of 13th International Conference on Artificial Reality and Telexistence, pp. 148-153, December 2003.
- J. P. Lewis, M. Cordner, and N. Fong. "Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation", ACM SIGGRAPH 2000, July.
- D. Libs, "Modeling Dynamic Surfaces with Octrees", *Computers & Graphics*, Pergamon Press, Vol.15, No. 3, 1991.
- M. Lin and S. Gottschalk, "Collision detection between geometric models: A survey", Proc. of IMA Conference on Mathematics of Surfaces, pp. 37-56, 1998.
- Q. Luo and J. Xiao, "Geometric Properties of Contacts Involving a Deformable Object", Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'06), pp. 533-538, 2006.
- G. Maestri, "Digital Character Animation 2", Vol 1. New Rider, Indianapolis, 1999.
- N. Magnenat-Thalmann, R. Laperriere, and D. Thalmann. "Joint-dependent local deformations for hand animation and object grasping", *In Proceedings of Graphics*

Interface, pp. 26-33, June 1988.

- N. Magnenat-Thalmann, H. Seo, and F. Cordier, "Automatic Modeling of Virtual Humans and Body Clothing", *Special issue on computer graphics and computer-aided design*, Volume 19, Issue 5, September 2004.
- D. Marchal, F. Aubert and C. Chaillou, "Collision between Deformable Objects Using Fast Matching on Tetrahedral Models", SIGGRAPH Symposium on Computer Animation, 2004.
- M. Matthias, L. McMillan, J. Dorsey, and R. Jagnow, "Real-Time Simulation of Deformation and Fracture of Stiff Materials," Eurographics CAS, Computer Animation and Simulation 2001, pp. 113-124.
- M. McKinley and V. Loughlin, *Human Anatomy*, McGraw-Hill, 2005.
- A. Mohr, L. Tokheim, and M. Gleicher, "Direct Manipulation of Interactive Character Skins", Symposium on Interactive 3D Graphics, 2003.
- M. Muller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler, "Stable Real-Time Deformations", ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 49-54, 2002.
- M. Muller, B. Heidelberger, M. Teschner, and M. Gross, "Meshless Deformation Based on Shape Matching", *ACM Trans. Graph.*, 24-3, 2005.
- M. Muller, L. McMillan, J. Dorsey, and R. Jagnow, "Real-Time Simulation of

- Deformation and Fracture of Stiff Materials", Eurographic Workshop on Computer Animation and Simulation, pp. 113-124, 2001.
- J. Noh, D. Fidaleo and U. Neumann, "Animated Deformations with Radial Basis Functions", Proceedings of the ACM symposium on Virtual reality software and technology, pp. 166-174, 2000.
- M.A. Otaduy, and M.C. Lin, "Sensation Preserving Simplification for Haptic Rendering", ACM SIGGRAPH 2003, pp. 543-553.
- S. Park and J. K. Hodgins, "Capturing and Animating Skin Deformation in Human Motion", SIGGRAPH 2006, 25(3), pp. 881-889.
- S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P. Sloan, "Interactive Ray Tracing. In Proceedings of Interactive 3D Graphics, pp. 119-126, 1999.
- J-P. Pernot, B. Falcidieno, F. Giannini, S. Guillet, and J-C. Léon, "Modelling Free-Form Surfaces using a Feature-Based Approach", Proceedings of the eighth ACM symposium on Solid modeling and applications, pp. 270-273, 2003.
- G. Picinbono, H. Delingette, and N. Ayache, "Non-linear and Anisotropic Elastic Soft Tissue Models for Medical Simulation", *Robotics and Automation*, ICRA, pp.1370-1375, 2001.
- G. Picinbono, J-C. Lombardo, H. Delingette, and N. Ayache, "Improving Realism of a Surgery Simulator: Linear Anisotropic Elasticity, Complex Interactions and Force Extrapolation", *Visualization and Computer Animation*, 13(3), pp. 147-167, 2002.

- D. Rose and T. Ertl, "Haptic Modeling of Finite Element Surfaces", *Vision Modeling and Visualization (VMV)*, Germany, pp. 123-130, Nov 2005.
- H. Samet, "Implementing Ray Tracing with Octrees and Neighbor Finding", *Computers and Graphics*, 13(4), 1989.
- H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1990.
- W. Schroeder, K. Martin, B. Lorensen., *Visualization Toolkit, Third edition*, Kitware Inc., 2004.
- T. W. Sederberg, "Class Notes of Computer Aided Geometric Design", 2004, <http://cagd.cs.byu.edu/~557/>.
- T. W. Sederberg and S. R. Parry, "Free-Form Deformation of Solid Geometric Models", SIGGRAPH 1986, pp. 151-159.
- SensAble Technologies Inc., *Openhaptic Toolkit Programmer's Guide*, SensAble Technologies Inc., 2005.
- L. Shi, Y. Yu, N. Bell, and W.-W. Feng, "A Fast Multigrid Algorithm for Mesh Deformation", SIGGRAPH 2006.
- D. Sunday, "Intersections of Rays, Segments, Planes and Triangles in 3D", http://www.softsurfer.com/Archive/algorithm_0105/, 2003

- R. Szeliski, S. Lavalley, "Matching 3-D anatomical surfaces with non-rigid deformations using octree-splines", Biomedical Image Analysis, Proceedings of the IEEE Workshop, June 1994.
- K. Tagawa, K. Hirota, and M. Hirose, "Impulse Response Deformation Model: an Approach to Haptic Interaction with Dynamically Deformable Object", IEEE Virtual Reality Conference, 2006.
- J. Teran, S. Blemker, V. Ng, Thow Hign, and R. Fedkiw, "Finite volume method for the simulation of skeletal muscle". SIGGRAPH/Eurographics Symp. on Computer Animation 2003, pp. 68-74.
- J. Teran, E. Sifakis, G. Irving, and R. Fedkiw, "Robust Quasistatic Finite Elements and Flesh Simulation", Symposium on Computer Animation, July 2005.
- D. Terzopoulos and A. Witkin, "Physically Based Model with Rigid and Deformable Components", *IEEE Computer Graphics & Applications*, pp. 41-51, 1988.
- T. Vassilev, "Simulation of a Deformable Human Body for Virtual Try-on", Proceedings of the 2007 international conference on Computer systems and technologies.
- F. Velasco and J. C. Torres, "Cell Octree: A New Data Structure for Volume Modeling and Visualization", VI Fall Workshop on Vision, Modeling and Visualization, 2001.
- A. Watt, *3D computer Graphics, Third Edition*, Addison-Wesley, 2000.
- J. Weber, *Run-Time Skin Deformation*, Intel Architecture Labs, 2000.

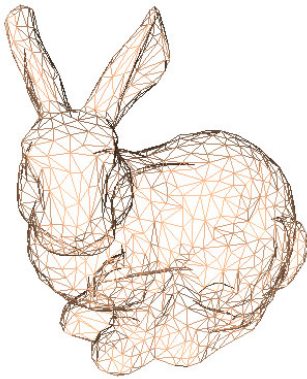
- G. Wolberg, "Image Morphing Survey", *The Visual Computer*, 14, 8/9, 1998.
- J. Wilhelms and A. V. Gelder, "Octrees for faster isosurface generation", *ACM Transactions of Graphics*, 11(3), pp. 201-227, July 1992.
- R. L. Williams II, W. Ji, J. N. Howell, and R. R. Conatser Jr, "Device for Measurement of Human Tissue Properties In Vivo", *ASME Journal of Medical Devices*, 1(3), pp.97-205, 2007.
- R. S. Wright and B.n Lipchak, *OpenGL SuperBible, Third Edition*, SAMS, 2004.
- H. B. Yan, S. M. Hu, and R. Martin, "Skeleton-Based Shape Deformation Using Simplex Transformations", *Computer Graphics International 2006*, pp. 66-77.
- H. T. Yau, L. S. Tsou, and M.J. Tsai, "Octree-based Virtual Dental Training System with a Haptic Device", *Computer-Aided Design & Applications*, Vol. 3, Nos. 1-4, pp 15-424, 2006.
- L. Zhang, Y. Kim, and D. Manocha, "A Fast and Practical Algorithm for Generalized Penetration Depth Computation", *Robotics: Science and Systems Conference (RSS07)*, 2007.
- M. Zhang and A.F. Mak, "In Vivo Friction Properties of Human Skin", *Prosthet Orthot Int*, 1999 Aug; 23(2): pp. 135-41.
- Y. Zhong, B. Shirinzadeh, G. Alici, and, J. Smith, "Haptic Deformation Modelling

Through Cellular Neural Network”,

Q. Zhu, Y. Chen and A. Kaufman, “Real-time Biomechanically-Based Muscle Volume Deformation Using FEM”, Computer Graphics Forum, 190-3, pp. 285-284, 1998.

Appendix A. Models for the Research

These models are used in this research to test the BSP and octree algorithms. The numbers of the vertices (ver) and triangles (tri) of each are listed.



ver:1494 tri:2915



ver:1588 tri:2962



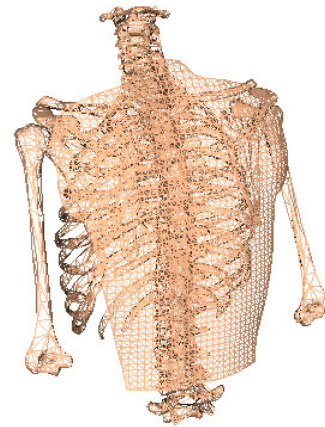
ver:3245 tri:6261



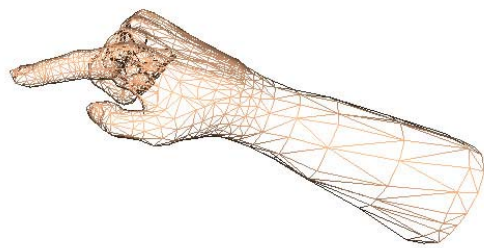
ver:4164 tri:8163



ver:10303 tri:20389



ver:21082 tri:41582



ver:1418 tri:2818

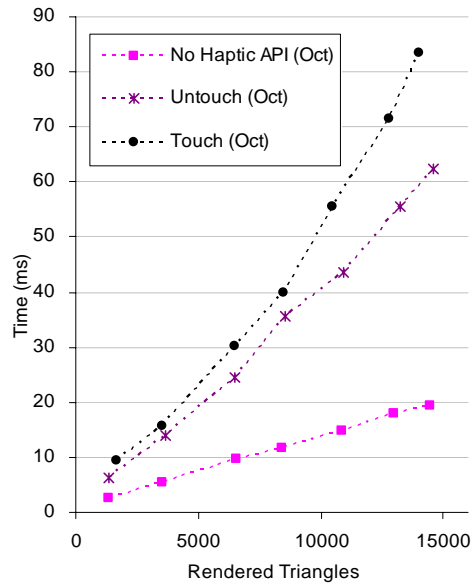
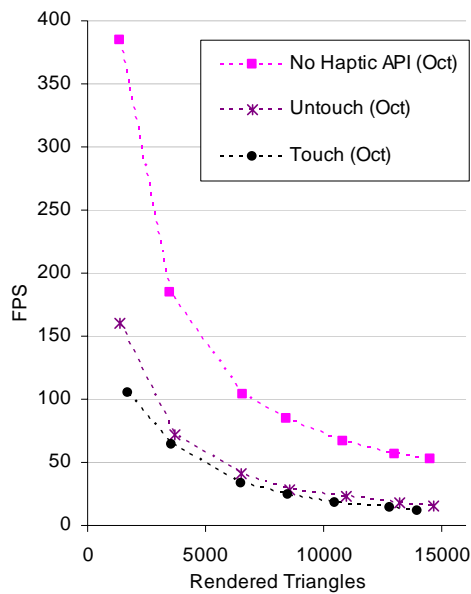


ver:1401 tri:2786

Appendix B. Haptic Rendering Performance in Octree Model

In the haptic process of the octree partitioned model, the triangles drawing, untouched and touched haptic renderings were measured. The original data are attached here.

No Haptic API (Triangle Drawing)			Untouched (Triangle Drawing, Collision Detection)			Touched (Triangle Drawing, Collision Detection, Force Generation)		
# of Rendered Triangles	Time (ms)	FPS	# of Rendered Triangles	Time (ms)	FPS	# of Rendered Triangles	Time (ms)	FPS
1373	2.6	385	1373	6.3	160	1696	9.5	105
3507	5.4	184	3675	13.9	72	3552	15.6	64
6580	9.6	104	6466	24.4	41	6488	30.3	33
8415	11.8	85	8550	35.7	28	8482	40.0	25
10830	14.9	67	10928	43.5	23	10464	55.6	18
12987	17.9	56	13219	55.6	18	12788	71.4	14
14481	19.2	52	14615	62.5	16	13981	83.3	12



Appendix C. Back Deformation Ellipses Measurement Data

The data were obtained from the measurement about the zero-deformation contour at five of the seven key points. The experiments were taken on five subjects.

Force Direction	Point	Subject	Major Axis (cm)	Minor Axis (cm)	Ratio (Major/Minor)	Degree (Major Axis to x-axis)
Up	1	1	42	32	1.313	90
		2	35	33	1.061	90
		3	34	32	1.063	90
		4	36	33	1.091	90
		5	32	28	1.143	90
		Ave	35.8	31.6	1.134	90.0
	2	1	33	27	1.222	90
		2	28	28	1.000	90
		3	23	28	0.821	90
		4	35	31	1.129	90
		5	30	28	1.071	90
		Ave	29.8	28.4	1.049	90.0
	3	1	26	18	1.444	180
		2	28	28	1.000	180
		3	29	21	1.381	180
		4	26	21	1.238	180
		5	24	22	1.091	180
		Ave	26.6	22.0	1.231	180.0
	4	1	36	25	1.440	95
		2	35	29	1.207	87
		3	41	29	1.414	135
		4	38	27	1.407	90
		5	29	22	1.318	90
		Ave	35.8	26.4	1.357	99.4
	5	1	26	20	1.300	145
2		30	24	1.250	90	
3		34	24	1.417	140	
4		27	20	1.350	155	
5		25	20	1.250	75	
Ave		28.4	21.6	1.313	121.0	

Force Direction	Point	Subject	Major Axis (cm)	Minor Axis (cm)	Ratio (Major/Minor)	Degree (Major Axis to x-axis)
Down	1	3	27	28	0.964	90
		4	32	28	1.143	90
		5	23	25	0.920	90
		Ave	27.3	27.0	1.009	90.0
	2	3	39	31	1.258	90
		4	36	31	1.161	90
		5	33	26	1.269	90
		Ave	36.0	29.3	1.230	90.0
	3	3	28	21	1.333	180
		4	31	32	0.969	180
		5	25	25	1.000	180
		Ave	28.0	26.0	1.101	180.0
	4	3	35	27	1.296	60
		4	38	30	1.267	45
		5	32	25	1.280	100
		Ave	35.0	27.3	1.281	68.3
	5	3	33	33	1.000	90
		4	46	34	1.353	85
		5	37	24	1.542	90
		Ave	38.7	30.3	1.298	88.3

The final results are obtained from the averages in the table of the up force direction because it is typical and similar to down.

	P_1	P_2	P_3	P_4	P_5
Major Axis (mm)	358	298	266	358	284
Ratio	1.134	1.049	1.231	1.357	1.313
Degree	90.0	90.0	180.0	99.4	121.0