PHYSICALLY BASED MESH-FREE DEFORMATION FRAMEWORK AND TECHNIQUES FOR COMPUTER GRAPHICS

JIAN CHANG

June 2006

National Centre for Computer Animation

Bournemouth University

PHYSICALLY BASED MESH-FREE DEFORMATION FRAMEWORK AND TECHNIQUES FOR COMPUTER GRAPHICS

JIAN CHANG

A thesis submitted in partial fulfilment of the requirements of the Media School of Bournemouth University for the degree of Doctor of Philosophy

June 2006

National Centre for Computer Animation

Bournemouth University

ABSTRACT

In this thesis, we introduce a mesh-free deformation framework. Four different applications are presented based on it. Among them, a technique of mesh-free deformations and a technique of reusable deformations are to model the deformations in two different ways, while the hyper-twist and the force mapping are applied to other graphic purposes related to deformations.

Existing physically-based deformation techniques, such as the finite element method and the mass-spring systems, require the deformed object to be properly meshed. The proposed mesh-free deformations are constructed with unconnected points and no mesh is required in the computation. This process strictly follows the principles of classic mechanics and a deformation is defined as a combination of fundamental solutions. Because no mesh is involved, deforming a complex shape is as straightforward as deforming a simple one and the trade-off between efficiency and accuracy is easy to achieve by redistributing the points concerned. Experiments show that this method is fast and offers similar accuracy to the finite element methods.

Reducing both computational cost and amount of unnecessary human intervention remains a pressing issue in the animation production. To provide a faster and more user-friendly tool, we extend the above mesh-free deformations technique and develop another technique. A key feature is the reusability of deformations. Existing deformations can be simply extracted and reapplied physically using the 'copy' and 'paste' operations. It relieves the modelling efforts. In this way, the visual realism is combined with the modelling efficiency and the user-friendliness for animators.

The mesh-free deformation framework is capable to describe the deformations in an infinite body which is in line with the distortion of a 3D space. The twist of an infinite body, hyper-twist, is investigated to show how a 3D space and the object embedded can be radically deformed. Abstract shapes with aesthetic effects can be created in this process as well as their animations.

Following the idea of mesh-free computation, we apply forces on a surface to create the fine details of the surface. A force map records the applied forces and their distributions. We call this technique force mapping, which can be used for surface modeling, compression, reconstruction and editing. As an alternative to displacement mapping, force mapping benefits from the fact that the physical property, force, is integrated into a geometric surface explicitly.

ACKNOWLEDGMENT

First and foremost, I would like to thank my supervisors, Professor Jian J. Zhang and Dr. Hammadi Nait-Charif. Professor Zhang has guided me to the wonderland of computer animations. He is a wonderful person. It is joyful to discuss problems with him and the sparks of inspirations often come from such discussions. I shall never forget the abundant time and efforts he has put into my thesis. Dr. Nait-Charif has given me a lot of valuable comments on my thesis writing and helped me finalize the draft. I wouldn't have finished the writing so quickly without their help.

I would like to thank Dr. Stephen Bell as well for help with my English. Special thanks go to Dr. Xiaosong Yang for his genius tutorial on using computer software. My sincere appreciation goes to Mrs. Joyce Power and Mrs. Jan Lewis for their daily support and kind helps in non-academic ways. A special thank-you goes to Professor Xianghe Peng, and to all my teachers, for the skills and knowledge that I have learnt.

I can not fully express my gratitude to Mr Ray Reynolds, our cheerful neighbour, for his careful proofreading of the whole thesis. He has done an excellent job to perfect the thesis. My appreciation also goes to Mrs. Marian Mayor and to Mrs. Alison Bell for proofreading a few chapters of this thesis.

This research has been funded in part by the Arts and Humanities Research Board (grant no. B/RG/AN5263/ABPN12727). My study and research here is financially supported by the bursary of Bournemouth University and the Overseas Research Students Award Scheme (ORSAS). I also would like to take this opportunity to acknowledge their support.

Finally, I dedicate this thesis to my family, for their love and encouragement on my journey of growth, and to my wife, Xinyue, for her love.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Deformable object modelling is an open research subject in both engineering and computer graphics. Many successful methods have been developed for computer graphic applications. Among them, the physically-based modelling technique has attracted much attention due to its ability to achieve good accuracy, thus guaranteeing realistic effects in animated scenes.

However, the physically-based model requires human intervention to generate input data which describes its physical states and environment before the problem can be solved by a computer program. In most cases, such preparation is tedious and the user must master the knowledge of both the given software and the underlying physics of the phenomenon. So far, little attention has been paid to this issue and little effort has been made to reduce and simplify the work involved (Hirota 2002).

Most physically-based techniques used in computer graphics rely on an object being properly meshed. The meshes define the underlying physical connections among discretized points. The drawback of using meshes, however, is that the user must define the mesh carefully to best describe the shape and physics of the object in

question, and sometimes this meshing process has to be repeated during computation to prevent distorted elements from ruining the results (Bathe 1996).

To avoid this, in this research we propose a mesh-free deformation framework. Under this framework our research is to develop effective techniques to model deformations for computer graphics and animation applications. This framework is flexible and can also be applied to many applications beyond direct deformation modelling. Several techniques have been developed during the course of the research.

First of all, a technique of mesh-free deformations provides a user-friendly interface and alleviates the labour-intensive part in modelling, which demonstrates how our framework is used to physically generate deformations. A technique of reusable deformations is developed as an extension of the technique of mesh-free deformations, which allows the reuse of deformation data by 'copy' and 'paste' operations.

Related to distortions in the 3D space, what we called the hyper-twist is to create aesthetic shapes procedurally. And the so called force mapping is for geometric contraction and compression which expresses the geometric details into a force map. These two techniques are based on our framework and are applied to two different areas other than deformation modelling.

1.1 Challenges in Modelling Deformations

Nowadays, films and games require high image resolution and quality. This introduces big challenges in modelling and rendering. It is the same for modelling deformations. The following is an outline of the specific difficulties in modelling deformations for computer graphics purposes.

1. To create rich details, we often model the object with a complex geometric shape and hope to deform it correctly. It is challenging in mesh-based techniques as the detailed shape has to be properly meshed.

2. Complex physical properties and behaviours of the material in a real deformation are hard to imitate numerically. Even creation of a visual plausible effect presents a big challenge, not to mention physically correct results.

3. The objects would interact with each other and the environment. Challenges are to predict the interactions and to figure out the results accordingly.

4. The efficiency of the algorithm is important in computer graphics applications as many applications require interactive or even real-time performance.

5. Most users are not experts of physics or mathematics. The provided tools must be easy to handle. We do not want to force our users to learn additional knowledge before they can start.

1.2 Mesh-free Deformation Framework and Applications

1.2.1 Mesh-free Deformation Framework

The mesh-free deformation framework that we propose in this thesis is based on discretized points. It eliminates the problems associated with meshes and relieves the modelling effort in creating the deformation effects.

In the traditional mesh-based way, when a point of an object is moved to a new position, it affects its neighbouring points which are connected to the moving points by mesh. Such disturbance is then spread through the whole object, point by point (or element by element), following the mesh connection, and the object deforms.

In our mesh-free framework, the problem is treated in a different way. We use some discretized points on an object surface to capture the shape changes. There are no connections among the discretized points, and the force and displacement constraints are directly applied to these points.

Our mesh-free framework is constructed on the principle of superposition in elastic mechanics. The principle states that the deformation of an object caused by a sophisticated load case can be equivalently produced with a linear combination of the deformations by two or more simple load cases on the same object, while the linear combination of the simple load cases is equal to the sophisticated load case (Saada 1993). Instead of tackling the deformations directly, we decompose a deformation as a group of fundamental solutions. The fundamental solutions are carefully selected to describe the characteristic of material responses of some known loads, which regulate the location of each point. The distribution of the fundamental solutions is arranged to meet the boundary constraints and the composition of the fundamental solutions approximates the required deformation effect. When a point is moved or constrained, this causes changes of the distribution of the fundamental solutions, and all the other points change their locations accordingly to deform the object. Unlike meshes which define the connection of neighbouring points explicitly, in our framework, all the points are connected to each other implicitly by the definition of fundamental solutions.

1.2.2 Techniques and Applications

This mesh-free framework can be applied to many potential areas. Four different applications are investigated in this thesis.

1. Mesh-free Deformations

Using discretized points, our mesh-free deformations calculate the deformations directly from the boundary constraints. The Kelvin solution (see Chapter 4 for details) is chosen as a fundamental solution, which satisfies the governing partial differential equations (PDEs). Therefore, the sum of fundamental solutions satisfies the PDEs and the solving process is to adjust the relative quantities in the summation to meet the boundary constraints.

2. Reusable Deformations

The reusability of physically-based deformations is a novel concept proposed in this thesis. It would save the modelling effort extensively in many computer graphics

applications. Based on the mesh-free deformation framework, we define 'copy' and 'paste' operations to reuse deformation data in modelling different objects. The 'copy' process involves the extraction of the deformation behaviour from a source object. The 'paste' operation then applies it to a different object (target), resulting in the target object being deformed in a physically plausible manner.

3. Hyper-twist

A space can be distorted and the objects within it are deformed accordingly. Based on our mesh-free deformation framework, a technique is developed to create complicated distortions of a space. Following the distortions, different objects within the space are deformed to procedurally generate shapes with aesthetic effects. This technique is capable of creating a series of artistic works, including animation sequences, from a simple setting. In the thesis, the hyper-twist, which twists a 3D space, is chosen as an example to demonstrate the use of this technique.

4. Force Mapping

To compress geometric data, a displacement map records the difference between a surface and its simplified alternative. The detailed surface is reconstructed from a simple surface by disturbing the vertices along the surface normal according to the displacement map. With our mesh-free deformation frame, we transfer the information previously recorded by displacements into forces. A force map, alternatively, is used to reconstruct a complex surface from its simplified alternative. The force map stores the data compactly and results in a high compression ratio.

1.3 Contributions

The thesis presents a mesh-free deformation framework and its applications. The contributions are listed as follows.

- A mesh-free deformation framework is established in this thesis, which explains the deformation in a way different from most traditional physically-based techniques.

- A technique of mesh-free deformations is developed within the framework. The proposed method has the following advantages:
  - Modelling effort is reduced. There is no need to mesh a complex object.
  - This method can be directly applied to models drawn with polygons. No interior information is required and the vertices can serve as sampling points.
  - It is seamlessly connected to point-based rendering as both are point-oriented.
  - The solving process is fast as only the surface points come into computation.
  - The trade-off between speed and accuracy is controlled by re-sampling the points.
  - It is physically-based, producing force information as well as displacement information. This is useful in virtual reality with haptic rendering.
- Computation efficiency of mesh-free deformations is investigated. Two different approaches are introduced: one is based on sparse matrix computation, and the other is based on a pre-computation technique.
- Reusable deformations are introduced as an extension of the mesh-free deformation, to allow different geometric models to share a pre-defined deformation. 'Copy' and 'paste' operations are defined to accomplish the task. They are physically-based and work under the mesh-free deformation framework. Benefits of this technique are as follows:
  - The modelling cost is reduced due to the use of existing deformations.
  - A database stores the deformations, providing a wide choice for the user who may not have experience in deformation modelling.
  - New effects of deformation can be created by combining existing ones.
  - Deforming a model with rich details can be done by copying the deformation from its alternative with low resolutions, to save modelling effort.

- Anatomic human muscle structure is animated with the proposed mesh-free deformations. The deformations of the major muscles on the limbs and torso have been studied and modelled.
- The deformation of an infinite body is investigated in hyper-twist which is linked to a space distortion. A change in infinite space is encoded within the mesh-free deformation framework. Shapes with special aesthetic effects can be generated by a simple disturbance of an infinite space, providing a tool for artistic creation.
- Following the mesh-free deformation framework, a theory of force mapping is proposed which disturbs a surface with forces recorded in a force map. The force mapping provides an alternative way to displacement mapping to transform a complicated surface into a simple surface and a force map. It provides a higher compression ratio than the displacement mapping.

During the period of the research, we have produced several publications to report the technical developments achieved, which are listed as follows.

- Chang, J., Yang, X. and Zhang, J. J., Hyper-twist, (to be submitted).
- Chang, J., Zhang, J. J. and You L., Physically-Based Deformations: Copy and Paste, Submitted to *The Visual Computer* (accepted to publish)
- Yang, X., Chang, J., Zhang, J. J., 2006, Animating Human Muscle Structure, Submitted to *Computing in Science and Engineering*, (accepted to publish)
- Chang, J., Zhang J. J. and Yang, X., 2005, Fast Mesh-free Deformations, *Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05)*, published by IEEE, p.149 -156.
- Chang, J. and Zhang, J. J., 2004, Mesh-free Deformations. Computer *Animation and Virtual Worlds*, v. 15, n. 3-4, p. 211-217.
- Chang, J. and Zhang, J. J., 2004, Force Mapping, *Theory and Practice of Computer Graphics: Eurographics UK*, published by IEEE, p.204-210, (This paper has been awarded Ken Brodlie Prize and The Best Student Work).

1.4 Thesis Overview

The structure of the thesis is outlined in Figure 1.1.

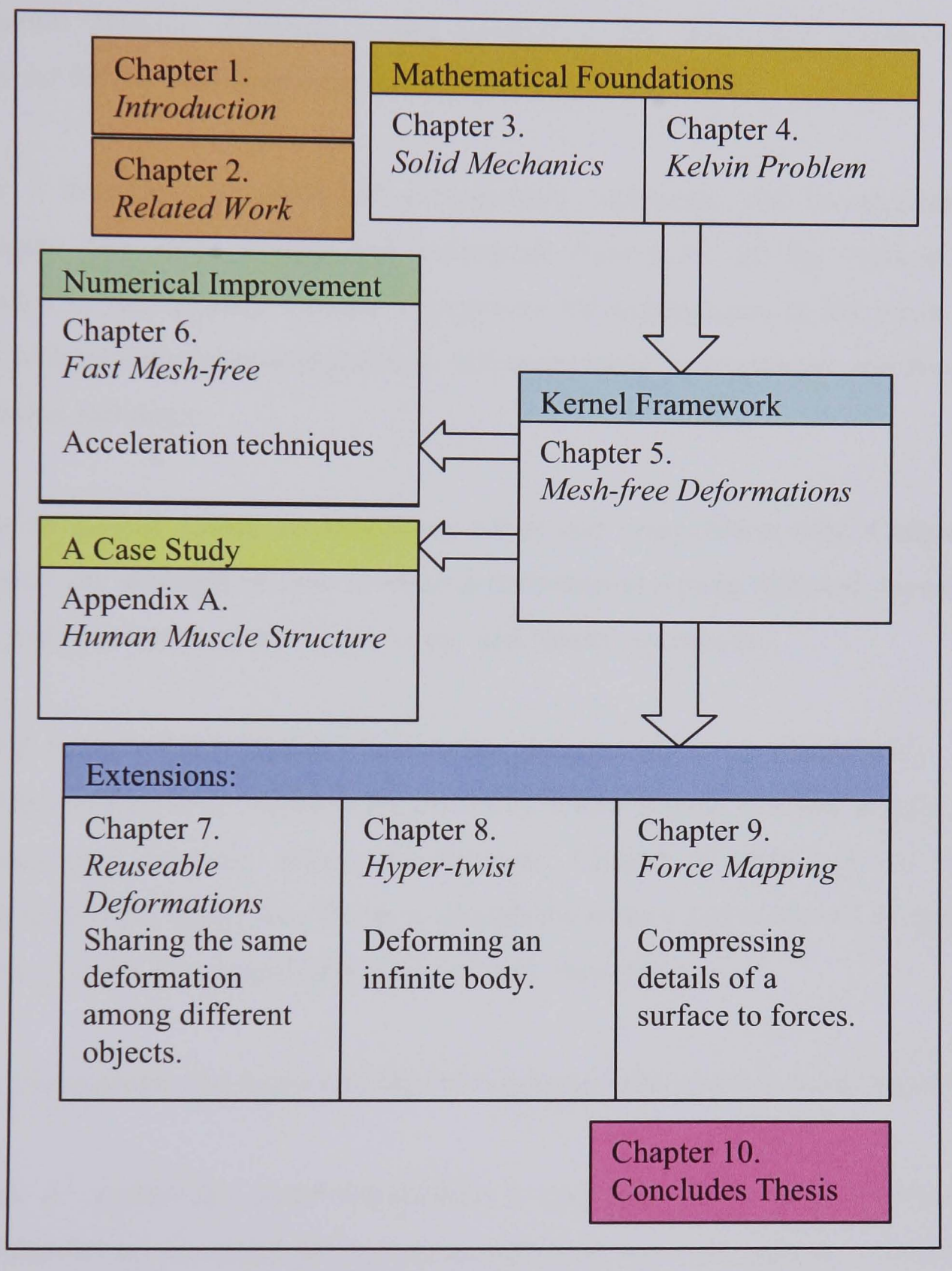


Figure 1.1 Structure of the thesis.

Chapter 1 is a brief introduction of this thesis. Chapter 2 reviews the techniques in deformation modelling, especially for graphic applications.

Chapter 3 reviews the basic definitions and formulisations in solid mechanics and outlines the frame of classic solid mechanics. Chapter 4 mainly introduces the Kelvin problem and the Kelvin solution, which is related to the definition of the fundamental solution. Chapter 3 and Chapter 4 lay down the mathematical foundations for the mesh-free deformation framework.

Chapter 5 develops the mesh-free deformations technique, and investigates its advantages. The main theory and numerical framework of the method are constructed in this chapter. Chapter 6 improves the performance of the mesh-free method with two acceleration algorithms, the sparse matrix computation and the pre-computation technique.

To achieve further saving on modelling effort and computation cost, Chapter 7 investigates the problem of how to share a deformation among different geometric models and develops a solution with 'copy' and 'paste' operations.

Chapter 8 extends the definition of mesh-free deformation to an infinite body. The twist of the infinite body, hyper-twist, driven by forces is used to create images and animations with aesthetic effects. Furthermore, Chapter 9 introduces the force mapping technique, which uses forces to disturb the surface and is viewed as another extension of mesh-free deformation for geometric modelling.

Chapter 10 concludes the thesis and identifies potential directions in future research.

Appendix A[†] applies the mesh-free method to anatomic human muscle structure. Human muscles are animated taking into account their anatomic shapes. It illustrates how human muscles move and deform to drive the movements of bones.

[†] This work has been jointly developed with Dr. Xiaosong Yang. My contribution is mainly on the construction of the mesh-free deformation module.

CHAPTER 2

RELATED WORK

Scientists have been working on the problem of deformation for centuries. As early as the seventeenth and eighteenth centuries, scientists led by Bernoulli and Euler have introduced the basic concept of strain, and extended Newton's Law of Motion to govern the transformation of a deformable body; based on their work, in the eighteenth century, Cauchy assembled the basic framework of three-dimensional continuum mechanics (Antman 1995). A thorough reference on the theory of solid mechanics can be found in (Fung 1965; Fung 1994; Saada 1993). There is no doubt, therefore that the advent of the computer was paramount in the development of numerical methods to solve mechanical problems in both theory research and engineering applications since the sixties.

Animated deformations have been widely applied in TV, movies and computer games. They are different from those in the analyses or simulations for engineering purposes. Animation mainly focuses on the visual effects while the simulation cares more about physical accuracy and correctness. Technically, there are two main streams of modelling deformations, the geometrically-based methods and the physically-based methods.

2.1 Geometrically-based Deformations

A pioneer work by Barr (1984) has discussed how to create deformations with geometric functions. In general, geometric implementation of deformations is directly manipulating the changes of a given object, namely the locations, orientations, velocities and accelerations of mesh vertices.

Normally, deformation is presented as interpolation of changes among the given feature points (elements). For instance, the Free-Form Deformations (FFDs) define and deform a lattice of control points to generate deformation effects of objects embedded (Sederberg and Parry 1986; Greissmair and Purgathofer 1989; Coquillart 1990; Hsu et al. 1992; MacCracken and Joy 1996); WIRES uses feature curves to define and shape the deformable features of objects (Singh and Fiume 1998); the ChainMail technique deforms an object via a chain-like reaction among the discretized points/elements (Gibson 1997; Li and Brodlie 2003). The FFDs is one of the most popular modelling methods. In the FFDs, a few control points define a parametric volume, and the points falling into the volume are expressed as interpolations of these control points. Moving the control points causes changes of the volume and thus the surface embedded in it deforms accordingly (Coquillart 1990). Geometric methods can be used to generate various morphing effects not necessarily restricted to the physics, which give their users extra freedom of creation. However, physically-based methods are advantageous in the creation of realistic scenes.

2.2 Physically-based Deformations

Physically-based methods often build up dynamic models to define the changes of geometric objects. Such dynamic models describe the related physical laws and give out the responses of objects subject to the environment changing, which is a numerical mimic or reproduction of real physical phenomenon. In practice, a dynamic model approximates the real world and characterizes the evolution of geometric models to a certain level of accuracy.

Terzopoulos et al. (1987) pioneered physically-based techniques for graphics applications. In (Terzopoulos et al. 1987), the elastic curve, surface and solids deformed with properly defined deformation energy. Another work (Terzopoulos and Fleischer 1988) extended the application of deformation energy from elasticity to viscolelasticity, plasticity and fracture. Baraff (1992) also shared the similar idea of using the energy in modelling flexible objects. There are various methods based on different theories to model deformation physically nowadays. In the following, we review the main streams of physically-based deformation techniques.

2.2.1 Mass Spring System

The MSS (Mass Spring System) has been widely applied in computer graphics as a branch among physically-based modelling techniques. The basic idea is straightforward. The MSS represents an object with discretized mass points connected by springs. Spring forces caused by movement of mass points balance external loads. Shape change is expressed as movement of mass points.

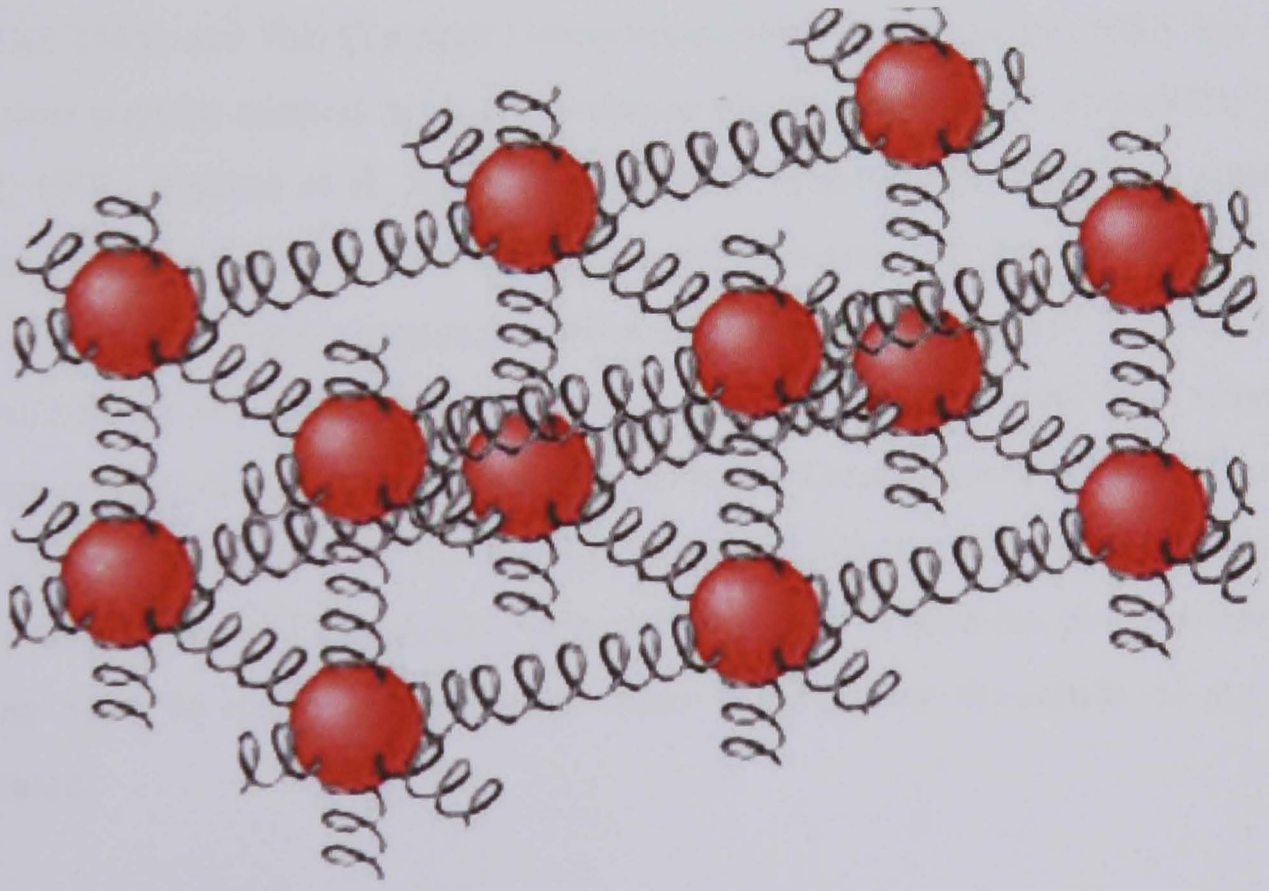


Figure 2.1 Mass Spring System

Generally, the external forces and the internal forces exert on a mass point to regulate its motion. The external forces are normally caused by contact or impact

and the internal forces include the spring forces and the damping force. We can write out the acceleration **a** of a given mass point as follows:

$$\mathbf{a} = \frac{1}{m}\left(\sum_i \mathbf{f}_i^e + \sum_j \mathbf{f}_j^s + \mathbf{f}^d\right) \tag{2.1}$$

where m is the mass, $\mathbf{f}_i^e$ is the ith external force, $\mathbf{f}_j^s$ is the jth spring force, and $\mathbf{f}^d$ is the damping force. The spring force linearly or non-linearly depends on the change of spring length. The damping force is defined as a function of the mass velocity. In the system, damping forces consume the kinetic energy and slow down the movement of mass points. Spring forces and damping forces characterize responses of objects subject to external forces. With the updated acceleration, the speed and the position can be updated. After the states of all mass points are refreshed, the updated information is used to reload the spring forces and the damping forces in the computation of next time-step.

Waters (1987) introduced springs to control the parameterisation for face shape and create facial expression. MSS was also used to create locomotion of worms, snakes (Miller 1988) and fish (Tu and Terzopoulos 1994). Furthermore, MSS has become the most popular method in cloth simulation because of its high adaptability (Volino et al. 1996; Bridson et al. 2003). The implicit integration is introduced (Baraff and Witkin 1998) to achieve fast calculation of dynamic response. With fine meshes, accurate spring and damping parameters, the MSS would provide a good approximation of a soft object to a certain level. However, the mass-spring approximation of continuous material is radically simplified and some of its artefacts might result in unrealistic scenes, e.g. volume of object is not always preserved in MSS deformations. Furthermore, it is a tedious task to tune the physical parameters in the MSS to reach a good result when hundreds or thousands of springs are involved.

2.2.2 Finite Element Method

The finite element method (FEM) is a well-established method, which has a long history of development and is still one of the most effective methods in engineering computation (Bathe 1996; Dhatt et al. 1984). The first introduction of the concept of

FEM dates back to the fifties (Turner et al. 1956; Argyris and Kelsey 1960). A number of FEM software packages have been developed to meet various requirements of industry nowadays. A few can be named here: NASTRAN, MARC, ANSYS, and ABAQUES.

In the FEM, a continuous object is discretized into many small elements, such as tetrahedrons or hexahedrons, which are of simple geometric shape. Each element contains a number of nodes. Each node is to be assigned a value of an unknown variable, which could be the displacement in a deformation simulation. An unknown variable within one element is expressed as the weighted sum of values on its nodes. In this way, a continuous distribution is approximated as piecewise fields element by element. A shape function, which depends on the type of element, defines the weight coefficient in the interpolation. With the above process, a continuous distribution can be represented by discretized values on the nodes. The equilibrium state of the object (the deformed shape) minimizes the potential energy; with the variation method, finding the distribution of the unknown variable is through solving algebra equations with unknowns as node values.

The development cost of a FEM software package for general purposes is relatively high and the standard FEM commercial packages are expensive. The operation interfaces of these packages are complicated and the operator must be familiar with the complicated software and master the theory of the physical phenomena in question. The FEM is computationally expensive and the preparation for a computation is tedious. One of the most time-consuming processes is to mesh the object to generate elements. The elements have to be carefully designed to avoid a degenerated shape that could break down the solving process. The limitation factor on the use of the FEM now is the manpower, rather than the computing power as was the case just a few years ago (Fagan 1992). Though the FEM is expensive for graphical applications, it has been widely applied.

The first adoption of FEM for computer graphics appeared in applications on surgery simulation, which models materials with complex mechanical properties and requires

accurate results (Deng 1988; Chen and Zeltzer 1992; Cotin et al. 1999; Hirota 2002). Based on the finite element theory, Gourret et al. (1989) took into account the active forces of fingers when interacting with a object. The multi-resolution method has been introduced to achieve real-time or interactive performance, where most computation is accomplished on coarse meshes to save time (Debunne et al. 1999; Debunne et al. 2001; Capell et al. 2002; Dewaele and Cani 2004). The concept of non-linear strain has been introduced to overcome the artefacts that occur when using a linear model for large deformations (Zhuang and Canny 1999; Müller et al. 2002). Irving et al. (2004) used invertible elements to take into account the effect of element rotation in large deformations. There is no doubt that the non-linear algorithms are more complicated and far more computationally expensive than the linear alternatives. O'Brien et al. computed out the split between two elements or within one element and the FEM has been used in simulating fractures (O'Brien and Hodgins 1999; O'Brien et al. 2002).

2.2.3 Boundary Element Method

The Boundary Element Method (BEM) is a 'Cinderella' of the numerical method, and is not as well known as the FEM (Beer 2000). The BEM numerically integrates the fundamental solutions along the boundary of an object to get the solution, which uses elements made of nodes to discretize the boundary (Beer 2000; Cartwright 2001). Unlike the FEM which subdivides the whole domain into elements, the BEM only does the subdivision on the boundary, which saves considerable time in creating and modifying the mesh. The BEM usually has advantages when dealing with stress concentration problems or with problems involving infinite or semi-infinite domains.

Not many applications of BEM in computer graphics can be seen. The ArtDefo, a simulator for interactive animation is accomplished with BEM (James and Pai 1999). James and Pai (2003) latter extended this work and introduced the techniques of wavelet to speed up the computation.

2.2.4 Mesh-free Methods

Distinctive from the mesh-based resolutions, the mesh-free methods, which are also called the mesh-less methods in some literatures, are point-based resolutions: adding or removing nodes/points to enhance the performance is much easier as there is no underlying mesh topology related to points. Szeliski and Tonnesen (1992) presented the elastic surfaces based on interacting particle systems, which are easy to split, join, or extend without manual intervention. In general mesh-free methods, many nodal points are distributed within the domain and each one is assigned values of physical variables, e.g. mass, speed, acceleration and displacement. The nodal points interact with each other. The value of a point which may not necessarily be a node is given out as the interpolation of values on its neighbour nodal points, and a shape function defines the weight in the interpolation. Normally, the residual method is used to deduce the governing equations which specify the behaviour of nodes. Different shape functions in the interpolation or different definitions of the residual value are taken into account in various types of mesh-free methods. For a thorough review about mesh-free methods, the reader can refer to (Liu 2003). The main advantage of the mesh-free methods is to get rid of the difficulty of meshing and remeshing. There is no requirement of this human-labour intensive process at the input stage and human resources are saved. Other problems associated with meshes are also alleviated with mesh-free methods, e.g. numerical difficulties caused by mesh distortion in MSS and FEM.

Various types of mesh-free methods have been developed. Among them, Smooth Particle Hydrodynamics (SPH) is popular in fluid simulation (Müller et al. 2003), which was first applied in astronomy to simulate the evolution of galaxies (Monaghan 1992). It was also used in (Desbrun and Gascuel 1996) to compute deformations. Müller et al. (2004) gave out another point-based method to animate deforming and melting, while Pauly et al. (2005) extended this work to animate fracture.

The mesh-free deformation framework proposed in this thesis is different from existing popular mesh-free methods. In the traditional mesh-free methods, the

solution is represented by the interpolation of nodal points. To compute the result of a given point, a search for the neighbouring nodes is carried first. Normally, a list of neighbour nodes is maintained during the computation to perform a quick search. However, our framework gives out the result directly as the function value of the fundamental solutions. No list of neighbours is maintained and the computation does not rely on any sort of connections among nodes. This makes our method distinctive from other mesh-free methods.

In addition, the technique of mesh-free deformations based on our framework only discretizes a boundary surface into points and performs computation on them, while most traditional methods require computation on interior points. This makes our method especially feasible for animation as most geometric models are constructed by surfaces only. There is no extra work to define the distribution of points within the object and some topology defects of the model do not cause problems: this may not be the case in other methods. The analytical fundamental solution is used in our mesh-free method and our solution satisfies the partial differential equations.

Although the idea of mesh-free computation has been used in engineering for some time, to our knowledge, we were among the first to introduce and adopt it to physically-based deformations in computer graphics (Chang and Zhang 2004).

2.3 Conclusions

In this chapter, we have reviewed the typical techniques for modelling deformations, especially those for computer graphics purposes. A greater attention was paid to the physically-based methods, and the traditional mesh-free methods are introduced within this category. It is pointed out that our framework shares the mesh-free idea with traditional methods but we treat the deformation problem in a very different way.

In this thesis, we will also discuss some applications of our presented mesh-free framework. These may include several techniques other than pure deformations

modelling. For better readability, some other techniques will be reviewed in the related chapters, including advances of muscle modelling (Appendix A), and a review of the displacement mapping technique (Chapter 9).

CHAPTER 3

FUNDAMENTALS OF SOLID MECHANICS

In this chapter, we give a brief introduction to the fundamentals of solid mechanics, which lays down the foundation for most physically-based numerical approaches, including the mesh-free resolutions. Most of the formulations in this chapter can be found in (Fung 1965; Fung 1994; Timoshenko and Goodier 1970; Saada 1993; Ogden 1997). The formulations in this chapter are the basic theory of mechanics and the reader can refer to the above books.

In continuum mechanics, it is assumed that material is continually distributed in space, and thus physical properties, such as density, displacement and velocity, can be described as continual functions of time and space. This continuity assumption assures the validity of differential operations of these variables.

3.1 Configuration and Deformation

At time $t = 0$, an object with continually distributed mass occupies known space Ω_0, which could be bounded or unbounded. This implies that there is a material point at any geometric point of Ω_0. Let's suppose that the object is deforming and its material keeps moving from an old position to a new position. At time t, the same object occupies different space Ω.

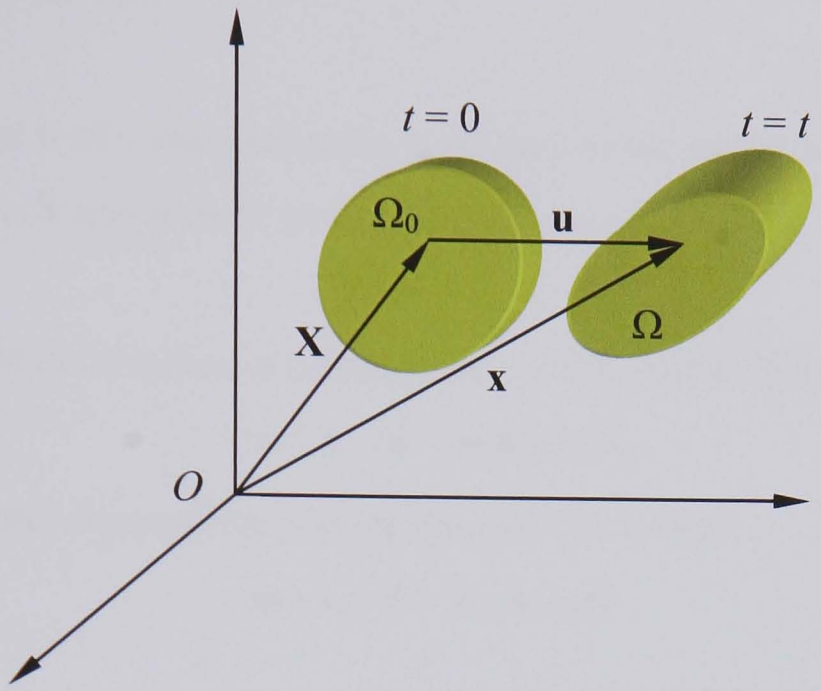


Figure 3.1 Tracking the deformation

Each material point is labelled with its position **X** at time 0. And the material point **X** moves to its new position **x** at time t. Its new position **x** is the function of time t and its initial position **X**.

$$\mathbf{x} = \mathbf{g}(\mathbf{X}, t) \qquad \mathbf{x}, \mathbf{X} \in \mathbf{R}^3 \tag{3.1}$$

By tracking motions of all material points, deformations of an object with continually distributed mass are describable.

A collection of locations about all material points is referred to as a configuration. The configuration at time 0 is referred to as the *material configuration*, or the *initial configuration*. The configuration at time t is referred to as the *spatial configuration* or the *current configuration*.

There are two ways to describe the problem: *the Lagrangian (material) description* in which the material point is labelled by its initial position **X**, *the Lagrangian coordinates*; *the Eulerian (spatial) description* in which the material point is labelled by its current position **x**, *the Eulerian coordinates*. The initial position can be given out as the inverse function of (3.1).

$$\mathbf{X} = \mathbf{h}(\mathbf{x}, t) \qquad \mathbf{x}, \mathbf{X} \in \mathbf{R}^3 \tag{3.2}$$

Both the Lagrangian description and the Eulerian description can convert to each other with (3.1) and (3.2).

The displacement **u** of a material point is defined as the vector which originates from its initial position **X** and ends at its current position **x**.

$$\mathbf{u} = \mathbf{x} - \mathbf{X} \tag{3.3}$$

In the Lagrangian description, **u** is treated as the function of **X** and t,

$$\mathbf{u} = \mathbf{x} - \mathbf{X} = \mathbf{g}(\mathbf{X}, t) - \mathbf{X} \tag{3.4}$$

And in the Eulerian description, it is the function of **x** and t.

$$\mathbf{u} = \mathbf{x} - \mathbf{X} = \mathbf{x} - \mathbf{h}(\mathbf{x}, t) \tag{3.5}$$

In other chapters of this thesis, only the lower case **x** is used to note the coordinates, because all the problems are described in the Lagrange reference coordinates and there is no need to tell the difference between the two reference systems.

3.2 Deformation Gradient and Displacement Gradient

Differentiating (3.1) with respect to **X** leads to:

$$d\mathbf{x} = \mathbf{F} d\mathbf{X} \tag{3.6}$$

or

$$dx_i = \sum_{j=1}^{3} \frac{\partial x_i}{\partial X_j} dX_j \qquad i = 1,2,3 \tag{3.7}$$

F is referred to as *the deformation gradient*, which describes the relative motion of a material point with respect to its neighbours.

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_2}{\partial X_3} \\ \frac{\partial x_3}{\partial X_1} & \frac{\partial x_3}{\partial X_2} & \frac{\partial x_3}{\partial X_3} \end{bmatrix} \tag{3.8}$$

The inverse of **F** is the gradient of the reverse deformation, that is,

$$\mathbf{F}^{-1} = \frac{\partial \mathbf{X}}{\partial \mathbf{x}} \tag{3.9}$$

Displacement gradient **D** is the differential of displacement **u** with respect to **X**.

$$\mathbf{D} = \frac{\partial \mathbf{u}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} - \frac{\partial \mathbf{X}}{\partial \mathbf{X}} = \mathbf{F} - \mathbf{I} \tag{3.10}$$

The displacement gradient is equal to the corresponding deformation gradient subtracting an identity matrix.

If there is no motion, the deformation gradient is equal to an identity matrix and the displacement gradient is a zero matrix.

3.3 Strain Analysis

The movement of a flexible object can be decomposed in two parts, the rigid transformation and the deformation. An intrinsic measurement of deformation would eliminate the influence of rigid transformation. In a pure rotation, **x** = **QX,** where **Q** is the rotation matrix. The deformation gradient **F** is equal to the rotation matrix **Q** in this case. It means **F** would change even if there is no deformation, and **F** is not a good measurement.

Let's define a tensor **C** as follows.

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} \tag{3.11}$$

For an arbitrary rigid transformation, **x** = **QX** + **c**, where **Q** is the rotation matrix and **c** is the translation, $\mathbf{C} = \mathbf{F}^T\mathbf{F} = \mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. Therefore, **C** is rotation-independent, which is a constant identity matrix for all rigid transformations. **C** only takes into account the influence of deformations and it provides a good measurement of deformation. **C** is called *the Right Cauchy-Green deformation tensor.*

The elongation ratio λ of a material line element along direction **n** in the Lagrangian coordinates is given out here.

$$\lambda^2 = \mathbf{nCn} \tag{3.12}$$

And *the left Cauchy-Green deformation tensor* ***B*** is also defined as follows alternatively.

$$\mathbf{B} = \mathbf{F}\mathbf{F}^T \tag{3.13}$$

In practice, the *Lagrangian strain tensor* $\boldsymbol{\gamma}$ and the *Eulerian strain tensor* $\boldsymbol{\eta}$ are defined as follows.

$$\boldsymbol{\gamma} = \frac{1}{2}(\mathbf{C} - \mathbf{I}) \tag{3.14}$$

$$\boldsymbol{\eta} = \frac{1}{2}(\mathbf{I} - \mathbf{B}^{-1}) \tag{3.15}$$

Both strain tensors are equal to zero when the object is only subject to a rigid transformation.

The element γ_{ij} of $\boldsymbol{\gamma}$ is written out here,

$$\gamma_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \frac{\partial u_i}{\partial X_j}\frac{\partial u_j}{\partial X_i}\right) \qquad i, j = 1,2,3 \tag{3.16}$$

where u_i is the projection of displacement **u** on the X_i axis.

The *engineering strain* $\boldsymbol{\varepsilon}$ with elements ε_{ij} is the linear approximation of the Lagrangian strain, which is suitable for problems of small deformations.

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right) \qquad i, j = 1,2,3 \tag{3.17}$$

3.4 Stress Analysis

Given a small surface, which could be inside the object or lying on the boundary, the materials inside it may be pulled by the materials outside. As shown in Figure 3.2, the normal of the small surface is **n**, the area of the small surface is δS, and the pulling force is $\delta\mathbf{p}$.

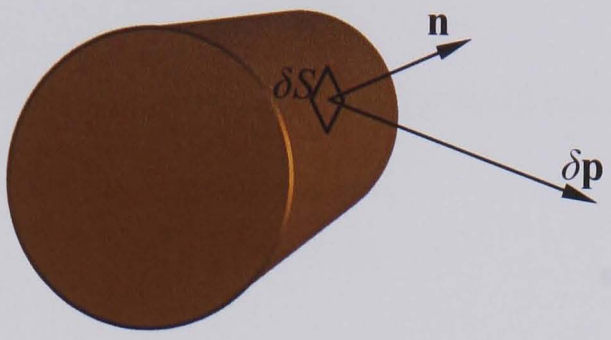


Figure 3.2 The small surface with normal **n** is pulled by force $\delta\mathbf{p}$

When the surface shrinks to a point Q, the surface traction **q** of surface with normal **n** at point Q is defined as follows.

$$\mathbf{q} = \lim_{\delta S \to 0} \frac{\delta \mathbf{p}}{\delta S} \tag{3.18}$$

The surface traction **q** is also given out in detail as:

$$\mathbf{q} = \boldsymbol{\sigma}\mathbf{n} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \begin{Bmatrix} n_1 \\ n_2 \\ n_3 \end{Bmatrix} \tag{3.19}$$

where $\boldsymbol{\sigma}$ is the *Cauchy stress tensor*.

Cauchy stress tensor is a symmetry tensor: $\sigma_{ij} = \sigma_{ji}$. The ith column of $\boldsymbol{\sigma}$, $\{ \sigma_{1i}, \sigma_{2i}, \sigma_{3i} \}^T$, is exactly the traction of surface perpendicular to the coordinate axis X_i at point Q.

3.5 Equilibrium Equations

Let's suppose that an object $\mathcal{B}$ is in equilibrium state, $\mathcal{R}$ is a part of $\mathcal{B}$, and surface $\mathcal{F}$ with normal **n** is surrounding $\mathcal{R}$. In equilibrium state, the sum of all the forces on material of $\mathcal{R}$ must equal to zero.

$$\iint_{\mathcal{F}} \mathbf{q}\, ds + \iiint_{\mathcal{R}} \mathbf{f}\, dv = 0 \tag{3.20}$$

where **q** is the surface traction and **f** is the body force.

Equation (3.21) is obtained by substituting equation (3.19) into equation (3.20).

$$\iint_{\mathcal{F}} \boldsymbol{\sigma}\mathbf{n}ds + \iiint_{\mathcal{R}} \mathbf{f}dv = 0 \tag{3.21}$$

With the divergence theorem (Ogden 1997), the equilibrium equations in integration form are given out as follows.

$$\iiint_{\mathcal{R}} (div\boldsymbol{\sigma} + \mathbf{f})dv = 0 \tag{3.22}$$

where $div\boldsymbol{\sigma}$ is the divergence of stress tensor $\boldsymbol{\sigma}$,

$$\mathrm{div}\boldsymbol{\sigma} = \nabla \cdot \boldsymbol{\sigma} = \begin{Bmatrix} \dfrac{\partial \sigma_{11}}{\partial x_1} + \dfrac{\partial \sigma_{12}}{\partial x_2} + \dfrac{\partial \sigma_{13}}{\partial x_3} \\ \dfrac{\partial \sigma_{21}}{\partial x_1} + \dfrac{\partial \sigma_{22}}{\partial x_2} + \dfrac{\partial \sigma_{23}}{\partial x_3} \\ \dfrac{\partial \sigma_{31}}{\partial x_1} + \dfrac{\partial \sigma_{32}}{\partial x_2} + \dfrac{\partial \sigma_{33}}{\partial x_3} \end{Bmatrix} \tag{3.23}$$

$\mathcal{R}$ is arbitrarily chosen and equations (3.22) are correct at any part of the object $\mathcal{B}$, so at any point in object $\mathcal{B}$,

$$div\boldsymbol{\sigma} + \mathbf{f} = 0 \tag{3.24}$$

which are the equilibrium equations in derivation form.

In case of motion, with the influence of inertia, the equilibrium equations in integration form become,

$$\iiint_{\mathcal{R}} (div\boldsymbol{\sigma} + \mathbf{f} - \rho\mathbf{a})dv = 0 \tag{3.25}$$

where ρ is the density and $\mathbf{a}$ is the acceleration of the material point.

$$\mathbf{a} = \ddot{\mathbf{x}} = \frac{d^2\mathbf{x}}{dt^2} \tag{3.26}$$

t is the time and $\mathbf{x}$ is the current position of the material point.

According to (3.25), the equilibrium equations in derivation form are

$$div\boldsymbol{\sigma} + \mathbf{f} = \rho\mathbf{a} \tag{3.27}$$

3.6 Constitutive Law

The constitutive law describes the mechanical or physical property of material. It defines how the stress state of an object changes according to the kinetic states, such as strain and strain-rate. The stress tensor may relate to other thermal and electromagnetic variables, for instance, temperature. Generally, materials' behaviours are non-linear and complicated. The mathematical models are much complicated for non-linear elasticity, elastoplasticity and plasticity. Here, we will discuss the linear models that can approximate non-linear materials in a linear way. For those non-linear models, readers can refer to the text books (Fung 1965; Fung 1994; Ogden 1997).

For a number of solids, the loads are proportional to the deformations. This means that when the load increases, the strain (deformation) increases in the same ratio, and vice versa. When the object is unloaded, deformations disappear. These experimental facts lead to the constitutive laws of the linear elastic material.

The linear elastic material is also called Hooke's material, whose stress $\boldsymbol{\sigma}$ linearly depends on the engineering strain $\boldsymbol{\varepsilon}$. The constitutive relation is as follows.

$$\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\varepsilon} \tag{3.28}$$

or,

$$\sigma_{ij} = \sum_{k=1}^{3}\sum_{l=1}^{3} C_{ijkl}\varepsilon_{kl} \qquad i,j = 1,2,3 \tag{3.29}$$

where $\mathbf{C}$ is the elastic constant tensor and C_{ijkl} is one of its elements.

Isotropic material possesses elastic properties which are independent of the orientation of the reference coordinate. In other words, it is a material that possesses a rotational symmetry with respect to any two perpendicular axes. The isotropic Hooke's material has only two material constants that appear in the constitutive equations due to its symmetry. The constitutive equations of isotropic Hooke's material are given out by (3.30).

$$\sigma_{ij} = \lambda\delta_{ij}\sum_{k=1}^{3}\varepsilon_{kk} + 2G\varepsilon_{ij} \qquad i,j = 1,2,3 \tag{3.30}$$

where λ and G are called the Lamé's constants, and G is referred to as the shear modulus (also marked μ). δ is the Kronecker delta.

$$\delta_{ij} = \begin{cases} 0, & for \quad i \neq j; \\ 1, & for \quad i = j. \end{cases} \tag{3.31}$$

Isotropic Hooke's material is an ideal material that provides an approximation to a large number of solids. In the real world, the behaviours of material are complicated. Such linear approximation is very rough and not sufficient for many applications. Anisotropy and non-linearity have to be introduced. One example is that the response of most biological material reveals various kinds of non-linearity.

Some material's stress may rely on its deformation history rather than its current strain tensor state only. One example is the linear viscoelastic material, whose constitutive law is as follows (Fung 1994).

$$\boldsymbol{\sigma}(t) = \int_0 \mathbf{C}(t-\tau)\frac{d\boldsymbol{\varepsilon}(\tau)}{d\tau}d\tau \tag{3.32}$$

Here the stress of time t is the integration of effects due to strain change along the whole deformation history. $\mathbf{C}$ of the elastic constant tensor now depends on load history.

3.7 Boundary Value Problem

In general, the deformation of an object is subjected to its external loads/constraints: the surface forces, surface displacements, and body forces. In most engineering applications, people are interested in the stresses and the displacements of a deformed object. In computer graphics, the visible part is the main interest and surface displacements play an important role, as people sense the deformation by noticing the displacements of surfaces. The stresses are needed only in some special applications, e.g. haptic rendering or surgery simulations where the force feedback is taken into account.

For an arbitrary point on the surface of an object, either the surface force or the displacement is known previously. The prescribed physical state of a patch of surface is called the boundary condition: force boundary condition for known force, and displacements boundary condition for known displacement.

The boundary value problem is described by governing equations plus boundary conditions and body forces, which is solvable. Governing equations are equilibrium equations plus constitutive equations, which regulate the responses of the object subject to the external loads. The solving process of a boundary value problem is shown in Figure 3.3.

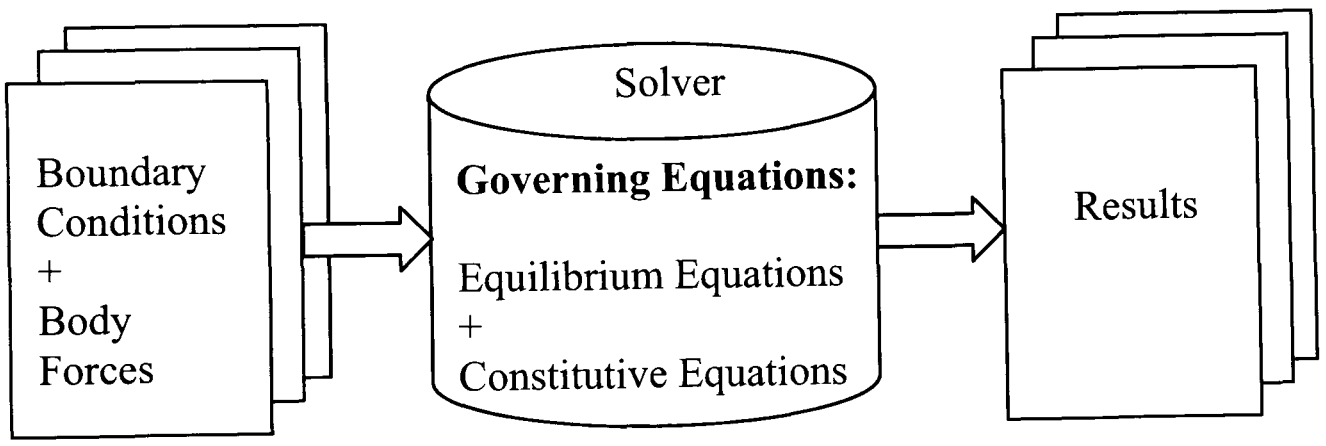


Figure 3.3 The solving process of a boundary value problem

3.8 Conclusions

The solid mechanics theory was introduced briefly in this chapter. It is based on the continuity assumption and serves as the foundation for development of our mesh-free deformation framework. This chapter started from the kinetic description of movement of a group of mass points. The definitions of stress and strain were outlined and their relationship was explained as the constitutive law. The deformation of object under given boundary conditions is governed by the equilibrium equations and the constitutive law.

CHAPTER 4

KELVIN PROBLEM AND SOLUTION

The Kelvin problem and its solution are the fundamental elements in the numerical framework presented in this thesis. The Kelvin problem is one of a few problems where analytic solutions exist. Its solution explicitly gives out continual distributions of displacement and other mechanical variables within an infinite space. This makes the Kelvin solution a good candidate for providing the fundamental solution in our mesh-free deformation framework. In this chapter, the definitions of the Kelvin problem and its solution are outlined.

4.1 Kelvin Problem

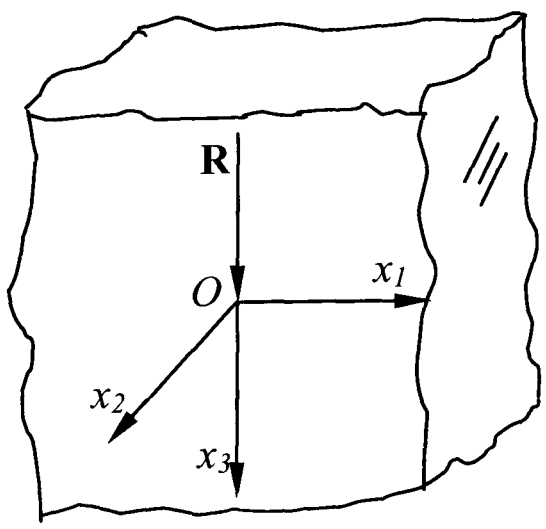


Figure 4.1 A concentrated force **R** exerts on an infinite body

The Kelvin problem is defined as the problem of an infinite isotopic elastic body subject to a concentrated force. As shown in Figure 4.1, the concentrated force with amount **R** is exerted at the origin, and the force is along the direction of axis x_3. The strength of the force is R. It is expected that all the stresses vanish at infinity.

The Kelvin problem is expressed as follows (Saada 1993):

$$\sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} = 0 \qquad for \quad i = 1,2,3 \tag{4.1}$$

$$\sigma_{ij} = \lambda \delta_{ij} \sum_{k=1}^{3} \varepsilon_{kk} + 2G\varepsilon_{ij} \qquad i, j = 1,2,3 \tag{4.2}$$

with force boundary conditions

$$\sigma_{ij} = 0 \qquad for \quad r \to \infty \tag{4.3}$$

$$\lim_{\delta r \to 0} \oiint_{r=\delta r} \sum_{j=1}^{3} \sigma_{1j} \frac{x_j}{\delta r} ds = 0$$

$$\lim_{\delta r \to 0} \oiint_{r=\delta r} \sum_{j=1}^{3} \sigma_{2j} \frac{x_j}{\delta r} ds = 0$$

$$\lim_{\delta r \to 0} \oiint_{r=\delta r} \sum_{j=1}^{3} \sigma_{3j} \frac{x_j}{\delta r} ds = R \tag{4.4}$$

where $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$ and R is the force strength.

Equations (4.1) are the equilibrium equations (3.24) as the body forces **f** vanish. Equations (4.2) are exactly the same as equations (3.30) which describe the constitutive law of isotropic elasticity. Equations (4.3) describe the force boundary condition in the infinite distance where all the stresses vanish. And equations (4.4) describe the force boundary condition at the origin. The external concentrated force is balanced by all the surface traction on a sphere with infinitesimal radius.

4.2 Kelvin Solution

The Kelvin problem described in Section 4.1 has an analytical solution while most mechanical problems only have numerical solutions. The displacement **u** = {u_1, u_2,

$u_3\}^T$ of one nontrivial point Q with coordinates $\mathbf{x}=\{x_1, x_2, x_3\}^T$ in the infinite body is given by (Saada 1993):

$$u_1 = \frac{R}{16\pi G(1-\nu)}\frac{x_1 x_3}{r^3}$$

$$u_2 = \frac{R}{16\pi G(1-\nu)}\frac{x_2 x_3}{r^3}$$

$$u_3 = \frac{R}{16\pi G(1-\nu)}\left(\frac{3-4\nu}{r} + \frac{x_3^2}{r^3}\right) \tag{4.5}$$

where $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$, R is the force strength, G is the shear modulus and ν is the Poisson ratio, which is the ratio between the lateral contraction and the axial elongation under a uniaxial stress condition. Poisson's ratio ν can be expressed by Lame's constants, $\nu = \frac{\lambda}{2(\lambda+G)}$.

The surface traction, $\mathbf{q} = \{q_1, q_2, q_3\}^T$, caused by the concentrated force, at point Q with surface normal $\mathbf{n} = \{n_1, n_2, n_3\}^T$ is (Saada 1993):

$$q_1 = -\frac{R}{8\pi(1-\nu)}\frac{1}{r^2}\left[3\frac{x_1 x_3}{r^2}\cos\theta - (1-2\nu)\left(\frac{n_1 x_3}{r} - \frac{n_3 x_1}{r}\right)\right]$$

$$q_2 = -\frac{R}{8\pi(1-\nu)}\frac{1}{r^2}\left[3\frac{x_2 x_3}{r^2}\cos\theta - (1-2\nu)\left(\frac{n_2 x_3}{r} - \frac{n_3 x_2}{r}\right)\right]$$

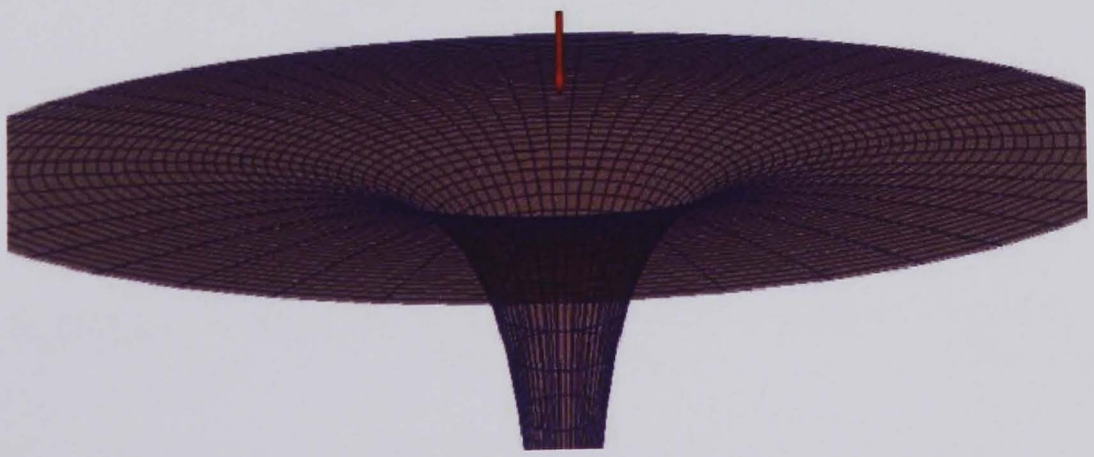
$$q_3 = -\frac{R}{8\pi(1-\nu)}\frac{1}{r^2}\left[3\frac{x_3^2}{r^2} + (1-2\nu)\right]\cos\theta \tag{4.6}$$

where $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$ and $\cos\theta = \frac{x_1}{r}n_1 + \frac{x_2}{r}n_2 + \frac{x_3}{r}n_3$.

Some examples of the Kelvin solution are drawn out in Figure 4.2, where shear modulus G = 100MPa, and Poisson ratio ν = 0.5. In Figure 4.2(a), the surface of a plane parallel to the concentrated force is selected and its deformed shape is drawn out. The distance of the surface to the force is one unit. In the deformed shape, all points are dragged down by the force. There are displacements perpendicular to the force as well. The upper half of the surface is pulled toward the force where the

lower part is pushed away. This causes the wave-like shape in Figure 4.2(a). In Figure 4.2(b), the surface of a plane is deformed. The force right exerts on the centre of the plane. It reveals that the displacements along the direction of the force are dominant. Both figures show that the closer the point is to the force, the larger the displacement is. The displacement at the origin where the force exerts goes to infinity as shown in Figure 4.2(b).

(a) Surface parallel to the force



(b) Surface perpendicular to the force

Figure 4.2 Examples of the Kelvin solution (the surfaces in the infinite body are deformed and the single force is noted by the arrow).

4.3 Kelvin Problem with Arbitrary Force

In general, the concentrated force may not exert on the origin but exert on an arbitrary point P, and the force may have an arbitrary direction instead of along a particular coordinate axis. Let's denote the force as **R** with $\mathbf{R}=\{R_1, R_2, R_3\}^T$, where R_i is the projection of the force on axis x_i.

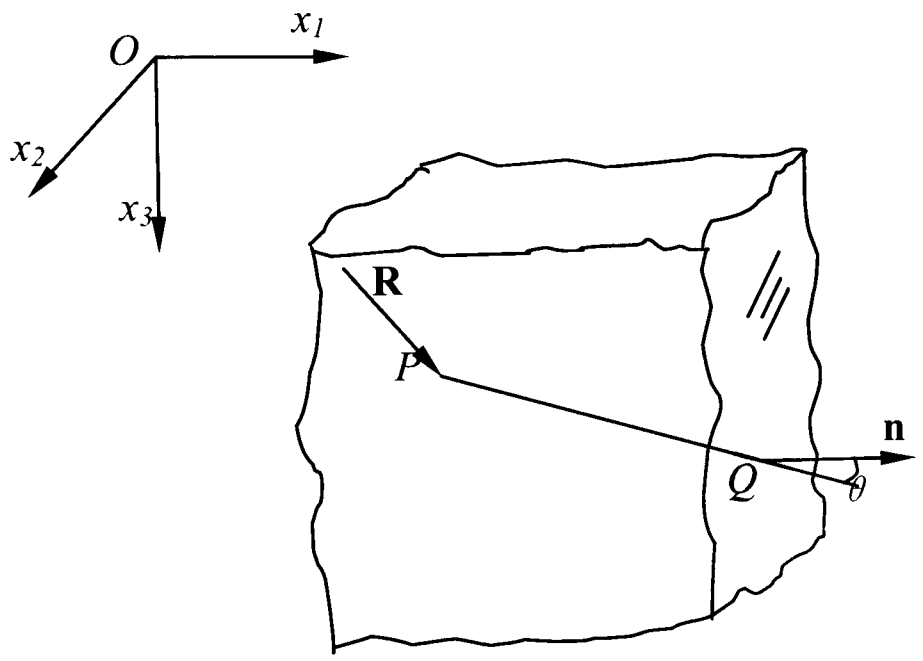


Figure 4.3 An arbitrary concentrated force **R** exerts on an infinite body

After deforming, the displacement $\mathbf{u} = \{u_1, u_2, u_3\}^T$ of one nontrivial point Q in the infinite body is given by:

$$u_j = \sum_{i=1}^{3} R_i U_{ij}(P,Q) \qquad for \quad j = 1,2,3 \tag{4.7}$$

where

$$U_{ij}(P,Q) = \frac{1}{r_{PQ}} W_{ij}(r_{PQ}) = \frac{1}{16\pi G(1-\nu) r_{PQ}} [(3-4\nu)\delta_{ij} + \frac{x_i^{PQ} x_j^{PQ}}{r_{PQ}^2}] \tag{4.8}$$

with

$$\delta_{ij} = \begin{cases} 0 & i \neq j; \\ 1 & i = j; \end{cases}$$

$$x_i^{PQ} = x_i(Q) - x_i(P);$$

$$r_{PQ} = \sqrt{(x_1^{PQ})^2 + (x_2^{PQ})^2 + (x_3^{PQ})^2} .$$

where $x_i(P)$ is the coordinate at point P and $x_i(Q)$ is that at Q. δ is the Kronecker delta, G is the shear modulus and v is the Poisson ratio.

$U_{ij}(P,Q)$ is the Kelvin solution of displacement which calculates the x_j component of the displacement at an arbitrary point Q, when a unit concentrated force exerts on point P in direction x_i. It is deduced from equations (4.5) by coordinate transformation.

Similarly, the surface traction $\mathbf{q} = \{q_1, q_2, q_3\}^T$ at point Q with surface normal $\mathbf{n}$ is given out:

$$q_j = \sum_{i=1}^{3} R_i T_{ij}(P,Q) \qquad for \quad j = 1,2,3 \tag{4.9}$$

where

$$T_{ij}(P,Q,\mathbf{n}) = \frac{1}{r_{PQ}^2} Z_{ij}(r_{PQ})$$

$$= -\frac{1}{8\pi(1-\nu)r_{PQ}^2}\{[(1-2\nu)\delta_{ij} + 3\frac{x_i^{PQ}x_j^{PQ}}{r_{PQ}^2}]\cos\theta - (1-2\nu)(\frac{x_i^{PQ}n_j}{r_{PQ}} - \frac{x_j^{PQ}n_i}{r_{PQ}})\} \tag{4.10}$$

with

$$\delta_{i,j} = \begin{cases} 0 & i \neq j; \\ 1 & i = j; \end{cases}$$

$$x_i^{PQ} = x_i(Q) - x_i(P)$$

$$r_{PQ} = \sqrt{(x_1^{PQ})^2 + (x_2^{PQ})^2 + (x_3^{PQ})^2}$$

$$\cos\theta = \frac{x_1^{PQ}}{r_{PQ}} n_1 + \frac{x_2^{PQ}}{r_{PQ}} n_2 + \frac{x_3^{PQ}}{r_{PQ}} n_3$$

$T_{ij}(P, Q)$ is the Kelvin solution of surface force which calculates the x_j component of the surface traction at an arbitrary point Q with surface normal $\mathbf{n}$, when a unit concentrated force exerts on point P in direction x_i. It is deduced from equations (4.6) by coordinates' transformation.

4.4 Conclusions

In this chapter, we addressed the Kelvin problem and its solution. The Kelvin solution specifies the deformation of an infinite elastic body subject to a concentrated force. The distributions of displacements and forces within the infinite body were formulated. Except singularity which appears at the point where the force is exerted, all the displacements are continually distributed within the space.

CHAPTER 5

MESH-FREE DEFORMATIONS

Existing physically-based deformation techniques, such as the finite element method (FEM) and the mass-spring system (MSS), require the deformed object to be properly meshed. This is arguably the most expensive manual intervention process. This chapter proposes a mesh-free deformation technique where only unconnected points are involved. It demonstrates the idea of the mesh-free deformation framework: to capture the changes on a surface using points; and to develop an approximate analytical solution using the fundamental solutions. Due to the fact that no mesh is involved, deforming a complex shape is as straightforward as deforming a simple one. Furthermore, the trade-off between efficiency and accuracy is easy to achieve by redistributing the points concerned. Experiments show that this method is fast and offers similar accuracy to the FEM.

5.1 Overview

The mesh-free method in this chapter is oriented for computer graphics applications. To deform an object, the user can place displacements (displacement boundary conditions) or tractions (force boundary conditions) on the surface/boundary of the new model and the object deforms accordingly.

Instead of meshing the object, the user only distributes a necessary number of points in the region of interest. Two sets of points are used. They are called *the collocation points* and *the virtual source points*. The collocation points are created on the surface of the object to capture the geometric shape and the boundary conditions. The virtual source points are related to the definition of the fundamental solutions and are distributed around the object, where virtual forces are exerted. The deformation is approximated in such a manner that both the equilibrium state and the boundary conditions are satisfied.

5.2 Mechanical Model of Deformations

As discussed in Chapter 3, a problem of deformation is normally defined as a boundary value problem: the deformation is subjected to its external loads/constraints, boundary conditions, and both the equilibrium equations and the constitutive law govern its behaviours.

In Chapter 3, we introduced the fundamentals of solid mechanics and the equations involved. The equilibrium equations (3.24) enforce the balance of all the forces with the deforming body. The engineering strain (see equation (3.17) for definition) for deformation measurement is introduced to get rid of the unnecessary complicity with non-linearity, which describes the small deformations well and provides a reasonable approximation for large deformations. The constitutive law of isotropic homogeneous elasticity is given by equations (3.30).

Substituting the definition of strain (3.17) and the constitutive equations (3.30) into equations (3.24), provides the equilibrium equations in term of displacements:

$$G\nabla^2 u_i + (\lambda + G)\frac{\partial \theta}{\partial x_i} = -f_i, \qquad i = 1,2,3, \quad \mathbf{x} \in \Omega \tag{5.1}$$

where Ω is a point set for the whole domain of the object, $\mathbf{x} = \{x_1, x_2, x_3\}^T$ denotes the Cartesian coordinates of a material point, $\mathbf{u} = \{u_1, u_2, u_3\}^T$ stands for the displacement vector, $\mathbf{f} = \{f_1, f_2, f_3\}^T$ is the body force vector, θ is the volume

expansion coefficient, and ∇^2 is the Laplacian operator. They take the following forms:

$$\theta = \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i},$$

$$\nabla^2 u_i = \sum_{j=1}^{3} \frac{\partial^2 u_i}{\partial x_j^{\,2}}.$$

G is the shear modulus, and λ is Lame's constant. They both can be rewritten with Young's modulus E and Poisson's ratio ν. Young's modulus E is the ratio of stress to strain along the loading direction when an object is subjected to uniaxial tension:

$$G = \frac{E}{2(1+\nu)},$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}.$$

There are two types of boundary conditions, displacement boundary conditions and force boundary conditions. The boundary conditions are defined on the boundary patches of domain Ω and are given as follows:

$$\begin{aligned} u_i &= u_i^0, \qquad \mathbf{x} \in \Gamma_u, \\ p_i &= p_i^0, \qquad \mathbf{x} \in \Gamma_f \qquad\qquad i = 1,2,3 \end{aligned} \tag{5.2}$$

where Γ_u represents the point set on the boundary where displacement **u** is known, and Γ_f is the point set on the boundary where surface force **p** is known. u_i^0 and p_i^0 are the known values of the displacements and forces at the boundaries, respectively.

If the body forces vanish, equations (5.1) is converted into a set of homogeneous partial differential equations,

$$G\nabla^2 u_i + (\lambda + G)\frac{\partial \theta}{\partial x_i} = 0, \quad i = 1,2,3, \quad x \in \Omega \tag{5.3}$$

5.3 Mesh-free Deformations

According to the principle of superposition in elastic mechanics, the deformation under a sophisticated load case can be equivalently produced with a combination of

simple load cases on the same object (Saada 1993). For instance, if the deformations of an elastic beam under pull, twist and bend are known, separately, the deformation of the beam under a combined load of these three actions is their linear combination. Thus using the known solutions under specific boundary conditions, the solutions for more general cases can be constructed. The known simple cases of deformations are named as the fundamental solutions, which are used to assemble the complex deformation.

However, it is often difficult to obtain solutions for a given model without the help of numerical methods, even under the simplest load case. The Kelvin problem, where an infinite body is subject to an individual concentrated force, described in Chapter 4, is one of a few fellow problems where the analytical solution exists. The core of the new technique is to develop an approximate analytical solution using the Kelvin solution.

Let us assume that an object is embedded within an infinite body. When a concentrated force acts on the infinite body, this embedded object will deform together with it due to the constraint imposed from the infinite body. The location of the concentrated force corresponds to some pattern of boundary constraints (forces and displacements) over the surface of the object, which can be formulated analytically with the solution to a Kelvin problem. The Kelvin solution can define all the physical states (displacements, surface tractions, stresses, strains, etc.) of the object, which satisfy the governing equilibrium equations and constitutive laws in the whole space.

The concentrated force of the infinite body is called a virtual force that is acting on a Virtual Source Point (VSP). With the virtual forces acting on different VSPs, various patterns of boundary conditions are constructed. Once the boundary conditions that the object is subject to are known, they can be approximated by a combination of these boundary condition patterns generated by the virtual forces. As each pattern of boundary conditions generates one deformation, which is given out as the Kelvin solution, the deformation of the object can be approximated by their combination.

Now, the solution is approximated by a combination of a collection of Kelvin solutions that serve as fundamental solutions.

One thing has to be kept in mind is that the Kelvin solution is singular on the VSPs. The VSPs, therefore, should not be located inside the model in order to avoid singularity appearing in the solution.

To meet a set of boundary conditions exactly, generally it needs an infinite number of virtual forces. In practice, however, a satisfactory approximation can be achieved with only a relatively small number of virtual forces. To represent the geometric shape and the boundary conditions, a set of Collocation Points (CPs) are distributed on the surface of the object. Each CP is assigned a boundary constraint, which is either a displacement or force. Where no constraint applies, the CPs are given a zero force value.

Let Q be a CP, P_i be a VSP. Let us also denote $U_{kl}(P_i,Q)$ to be the x_l component of the displacement (solution) at point Q caused by an unit virtual force exerting on point P_i in direction x_k as given in equations (4.8), and denote $T_{kl}(P_i,Q,\mathbf{n})$ the x_l component of the surface force at point Q, with surface normal $\mathbf{n}$, caused by the same unit virtual force as in equations (4.10). Both the displacements and surface forces at any collocation point Q within the domain subject to given constraints are

$$u_l(Q) = \sum_{i=1}^{n}\sum_{k=1}^{3}\alpha_{ik}U_{kl}(P_i,Q), \quad l = 1,2,3, \quad Q \in \Omega, \quad P_i \notin \Omega \tag{5.4}$$

$$p_l(Q,\mathbf{n}) = \sum_{i=1}^{n}\sum_{k=1}^{3}\alpha_{ik}T_{kl}(P_i,Q,\mathbf{n}), \quad l = 1,2,3, \quad Q \in \Omega, \quad P_i \notin \Omega \tag{5.5}$$

where $u_l(Q)$ represents the x_l component of the displacement at point Q and $p_l(Q, \mathbf{n})$ the x_l component of the surface force at point Q, where the surface normal is $\mathbf{n}$. α_{ik} are the weight coefficients to be determined, which stand for the strengths of the corresponding virtual forces.

Suppose there are n VSPs around an object and m CPs on the boundary of the object to be deformed. The residual value R is defined as:

$$R = \sum_{l=1}^{3}\sum_{j=1}^{mu}[u_l(Q_j) - u_l^0(Q_j)]^2 + \beta\sum_{l=1}^{3}\sum_{j=1}^{mf}[p_l(Q_j, \mathbf{n}) - p_l^0(Q_j)]^2 \qquad (5.6)$$

where u_l and p_l are the approximations given by equations (5.4) and (5.5), u_l^0 and p_l^0 are the known boundary conditions defined at the CPs. There are combined mu displacement CPs and mf force CPs, leading to a total of $mu + mf = m$ CPs. $\beta = \xi/E$ normalises the contribution of forces and displacements in equations (5.6), where E is Young's modulus, and ξ is a positive constant which is set to one in this chapter. When β becomes larger, the contribution of the forces appearing in the residual value increases. Minimising the residual value R determines the strength α_{ik} of virtual forces. The combination of the Kelvin solutions of the determined virtual forces provides a good match to the known boundary conditions. The deformation is then computed by equations (5.4), while the surface traction is traced by equations (5.5).

5.4 Implementation

5.4.1 Locating CPs and VSPs

The CPs are used to catch the geometric shape and allow boundary conditions to be represented. The density of CP distribution depends on the level of detail to be represented. From our experience, a sparse and regular distribution is sufficient for an object of a regular shape (Figure 5.1); a random spread of CPs works well for an object of free form nature (Figure 5.5(a)). Many complex models are range-scanned where the object is represented by a large number of points. Without establishing a mesh, these points (or a selection of them) can be directly used as CPs.

The VSPs are located outside the model for virtual forces to be added. The greater the variation of the boundary conditions, the more VSPs should be used. It is easy to locate VSPs for a regular geometric model (Figure 5.1). For a complex model, one can randomly sample the surface points of the model and displace these points some

distance away from the surface (Figure 5.5(b), (c)). It is also acceptable to distribute the VSPs based on a bounding geometry, such as a cube, sphere or cylinder.

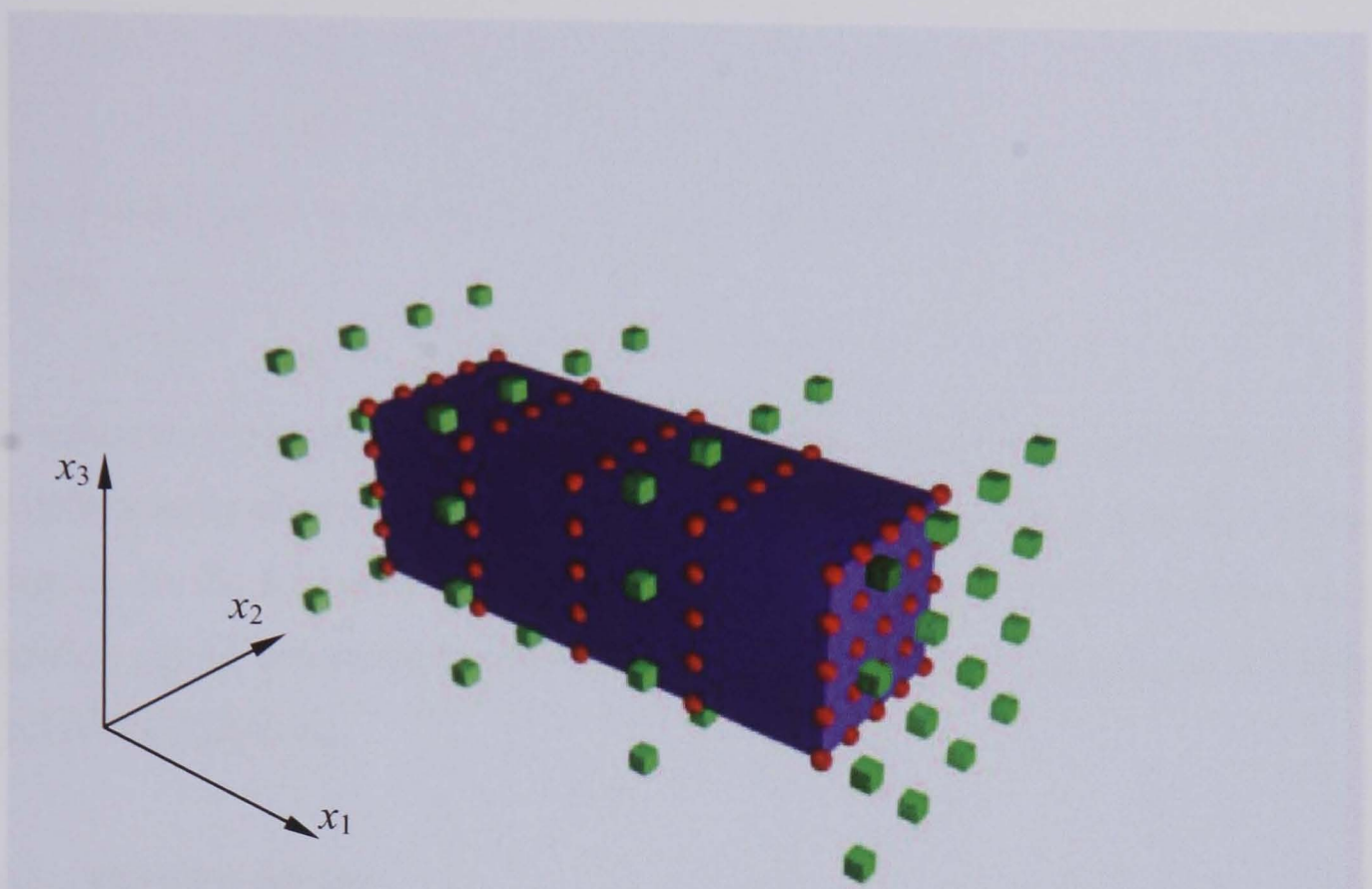


Figure 5.1 CPs and VSPs (the big blue cube represents the deformable domain Ω, the small red spheres represent the CPs, and the small green cubes are the VSPs where the virtual concentrate forces are exerted).

5.4.2 Deformation Computation

Equations (5.4) and (5.5) can be rewritten in a matrix form as follows:

$$\begin{Bmatrix} u \\ p \end{Bmatrix} = \begin{bmatrix} U \\ T \end{bmatrix} \{\alpha\} \tag{5.7}$$

where $\{u\}$ stands for the displacements and $\{p\}$ for the surface forces on the CPs, the elements of matrices U and T are the values of corresponding Kelvin solutions at the CPs, and $\{\alpha\}$ are assembled by the weight coefficients α_{ik}.

Minimising the residual value R of equations (5.6) with respect to the weight coefficients α_{ik} results in the following:

$$\frac{\partial R}{\partial \alpha_{ik}} = 0, \quad i = 1,2,...,n; \quad k = 1,2,3 \tag{5.8}$$

which gives the linear equations:

$$\begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix} \{\alpha\} = \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \begin{Bmatrix} u^0 \\ \beta \cdot p^0 \end{Bmatrix} \tag{5.9}$$

Here, β is the same as that of (5.6). u^0 and p^0 are the known boundary conditions on the CPs.

The solution of equations (5.9) determines the unknown weight coefficients α_{ik}. Thus, the deformation of any point which is not necessary to be a CP, within the deformed object or on the boundary can be computed by (5.4) and the surface forces on the boundary can be computed by (5.5). The standard linear equation solvers are adopted to solve the equations.

5.4.3 Iterative Solver

There are a number of methods to solve linear equations (5.9). An iterative method is uesed due to its efficiency advantages over other standard methods. Let us rewrite (5.9) as:

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{Bmatrix} r \\ \alpha \end{Bmatrix} = \begin{Bmatrix} b \\ 0 \end{Bmatrix} \tag{5.10}$$

where I is the identity matrix, $\{r\}$ is a temporary variable which stands for the residual vector, $\{b\} = \{u^0, \beta \cdot p^0\}^{\mathrm{T}}$, and $[A] = \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}$. The solution of equations (5.10) is the same as that of (5.9). This is because (5.10) lead to $\{r\} + [A]\{\alpha\} = \{b\}$ and $[A]^T[r] = 0$, which implies $[A]^T[A]\{\alpha\} = [A]^T\{b\}$, identical to (5.9).

The conjugate gradient procedure (Golub 1983) is adopted to solve (5.10). The complexity is $O(m \times n)$ for each iteration. It usually converges with a small number of iterations. Compared with the Gaussian elimination method, it is computationally more efficient, especially for a large-scale problem.

5.5 Results and Discussion

In this section, results of the new method are compared with both the analytical results and the results of FEM in terms of accuracy and performance. The material parameters for all examples are given as follows: shear modulus, 120MPa and Poisson's ratio, 0.45.

5.5.1 Computational Accuracy

To investigate the computational accuracy, the results from the mesh-free methods are compared with the analytic results in Figure 5.2 when a beam is under pull, bend or twist as in Figure 5.3. The size of the beam is 2cm×2cm×10cm. And VSPs are located on the surface of a slightly bigger block with the size of 4cm×4cm×14cm. The geometric centers of both the beam and the virtual source point block coincide. Both CPs and VSPs are distributed similarly to the pattern in Figure 5.1. 450 CPs are used for these examples.

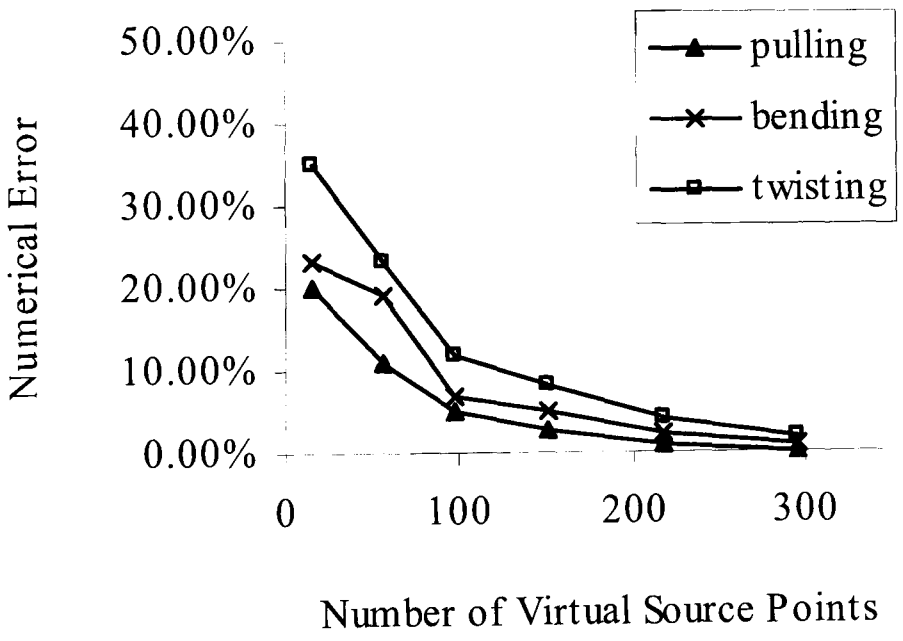
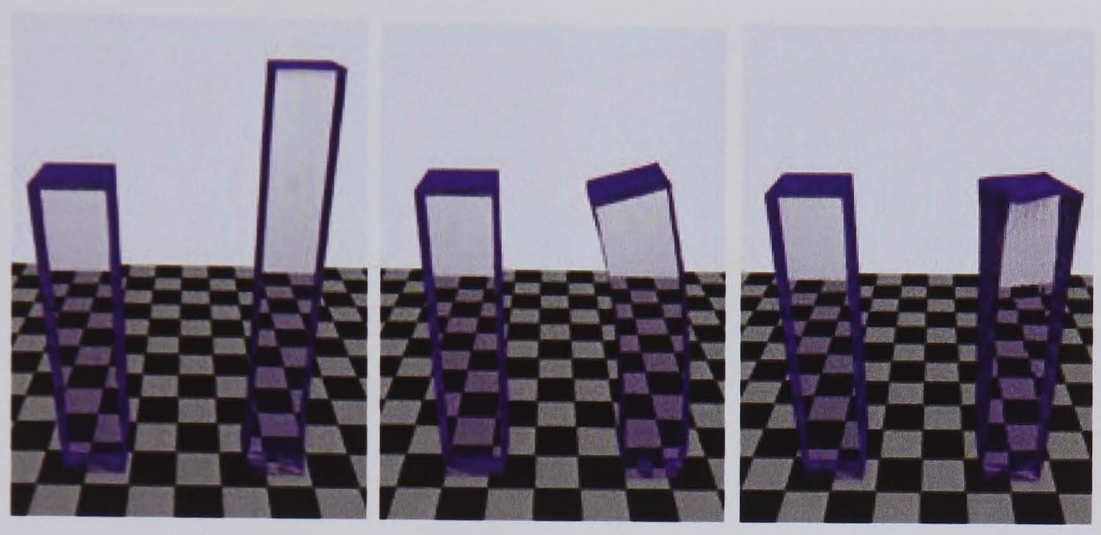


Figure 5.2 Computational accuracy affected by the number of VSPs.

From the diagram (Figure 5.2) it is seen that the accuracy is directly related to the number of VSPs. The greater the number of VSPs, the better accuracy it achieves. Nevertheless, even with a sparse distribution, such as 100 VSPs, a reasonable approximation can be achieved. Increasing the number of VSPs to 200, the error becomes negligible. It is worth pointing out that the number of CPs has a similar effect to resolution, i.e. the higher the number of CPs, the more detail can be

represented. There is no need to distribute more CPs than the required modelling resolution.

450 CPs and 226 VSPs are used to compute the results in Figure 5.3. It shows how the beam deforms.



(a) Beam under pull. (b) Beam under bend. (c) Beam under twist.

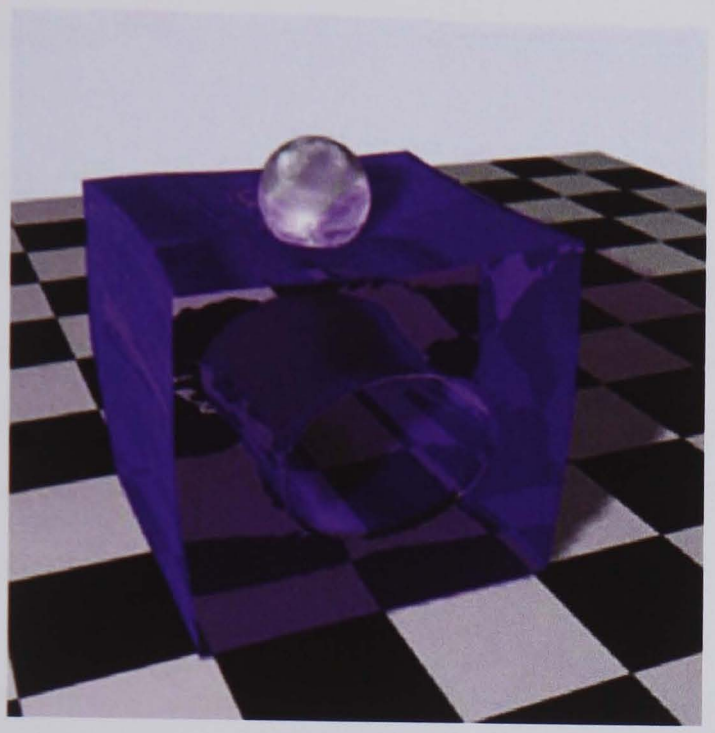
Figure 5.3 Beam deformations: the left represents the original beam and the right represents the deformed in the images.

5.5.2 Comparison with FEM

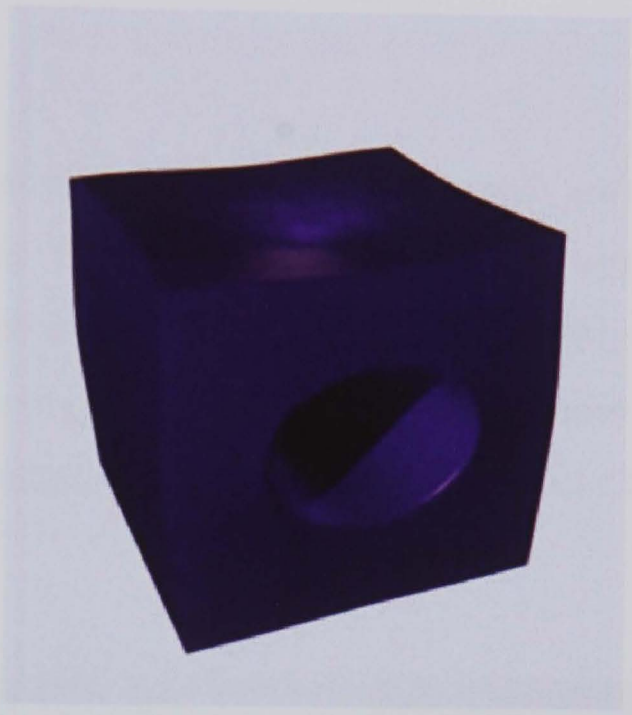
In Figure 5.4, a heavy rigid ball rests on the top of a soft block that has a cylindrical hole in the middle. Figures 5.4(a) and 5.4(b) are produced with the mesh-free method (both represent the same configuration). In Figure 5.4(b) the ball is removed to make it easy to do the comparison. Figures 5.4(c) and 5.4(d) are produced with the commercial FEM software, ANSYS, where Figure 5.4(c) is computed with the linear small strain analysis module and Figure 5.4(d) with the nonlinear large deformation analysis module. 1200 CPs are distributed on the surface of the object, and 432 VSPs are used in mesh-free model. For the FEM computations, 6000 elements are used. The computations were undertaken on a Dell Pentium III 500 MHz PC and the computation time is given out in Table 5.1.

It is interesting to see that not only is the mesh-free method fast, but the result matches well with the nonlinear analysis of the FEM. The reason is that the mesh-free method rephrases the displacement as a combination of solutions that have

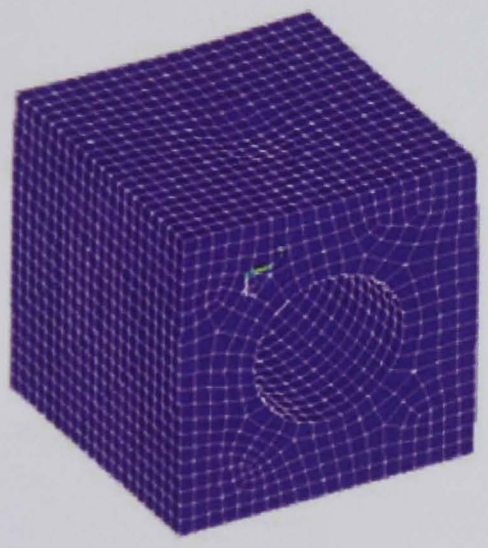
global influence, whereas the FEM decomposes the displacement into solutions of the small elements, which tends to localize the influence.



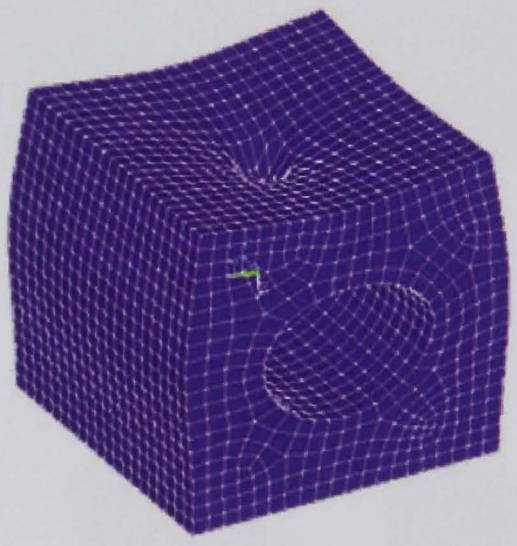
(a) Mesh-free method



(b) Mesh-free method



(c) FEM: linear small strain module



(d) FEM: non-linear large deformation module

Figure 5.4 A soft block with a heavy rigid ball resting on.

Table 5.1 Computation time for the example in Figure 5.4

Mesh-free method (Figure 5.4(a,b))	FEM: linear small strain (Figure 5.4(c))	FEM: non-linear large deformation (Figure 5.4(d))
21s	63s	629s

5.5.3 Bunny Model

In this section, the Stanford bunny model (Stanford 2004) is deformed to demonstrate how the new method can be applied to the deformation of complex objects.

940 CPs are distributed on the surface of the bunny (Figure 5.5(a)), which are the randomly selected vertices. Two settings of VSPs are used. The coarser setting consists of 94 VSPs (Figure 5.5(b)) and the denser setting of 432 VSPs (Figure 5.5(c)). Here all the VSPs are created by moving the randomly selected vertices on Bunny away from the surface a small distance (along the surface normal). The computation time is given out in Table 6.1.

(a) (b) (c)

Figure 5.5 Distribution of CPs and VSPs on the bunny model

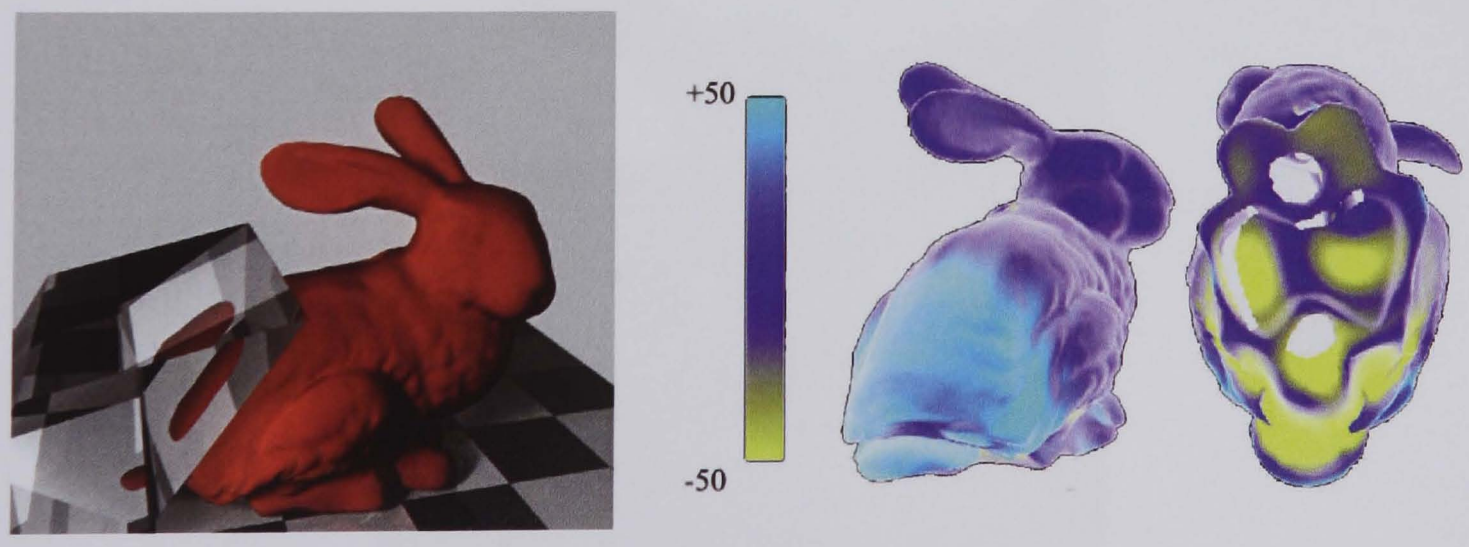
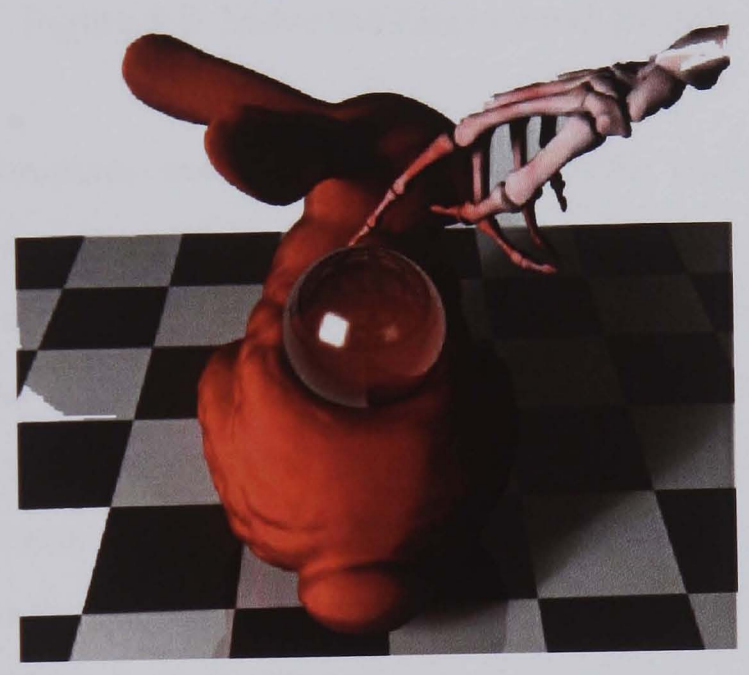
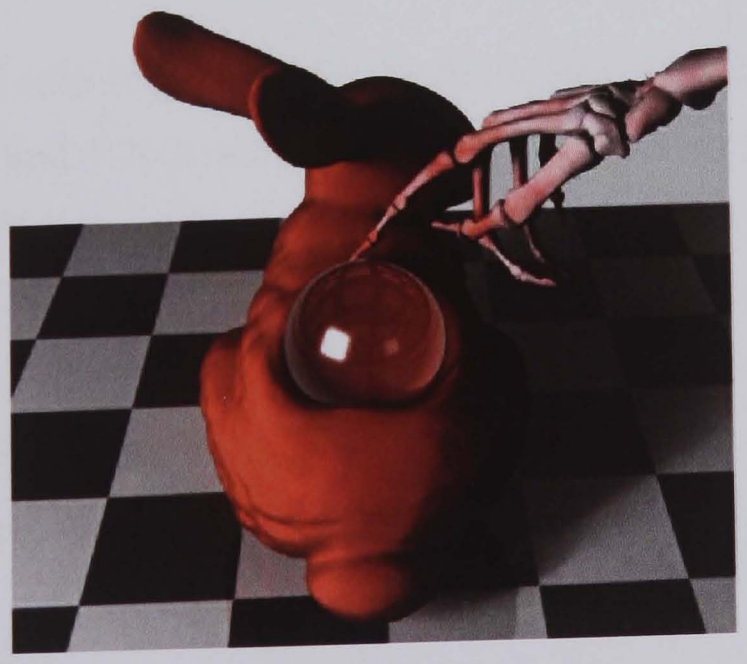


Figure 5.6 Force feedback (one unit of surface force stands for 1Mpa)

(a) Original bunny model;



(b) Deformed with 94 VSPs;



(c) Deformed with 432 VSPs.

Figure 5.7 Bunny with a ball on its back

(a) Bunny's head is dragged up (b)Bunny's head is dragged to the left

Figure 5.8 Move the Bunny head around

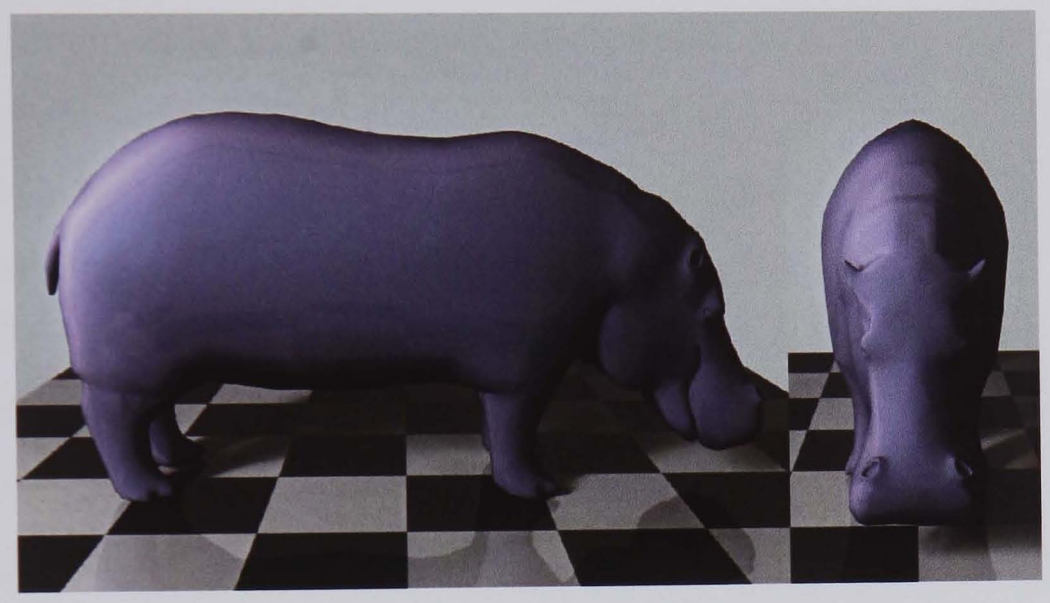
Since the method computes not only deformations from equations (5.4), but also surface forces from equations (5.5), it is useful also for force feedback applications. In Figure 5.6, the bunny model (432 VSPs) is fixed at the bottom and is pushed from the behind by a glass brick. This Figure shows the colour-coded force distribution both on the back surface (left) and the bottom surface (right). Here the forces drawn out are along the x_2 axis which is downward and perpendicular to the ground. The positive value means the direction of the force is downward and the negative value means the direction of the force is upward.

Another deformation case is given in Figure 5.7 where a ball rests on the bunny's back. Figures 5.7(b) and 5.7(c) take the same boundary conditions and collocation points, but with a different number of VSPs. In Figure 5.7(b), 94 VSPs are chosen, and the model looks a little stiffer than that in Figure 5.7(c) where 432 VSPs are used. A bump is noticeable on the bunny near the glass ball in Figure 5.7(c), whereas the deformation in Figure 5.7(b) is smoother, revealing less subtle detail. This is because more VSPs produce more accurate approximation to the real deformation.

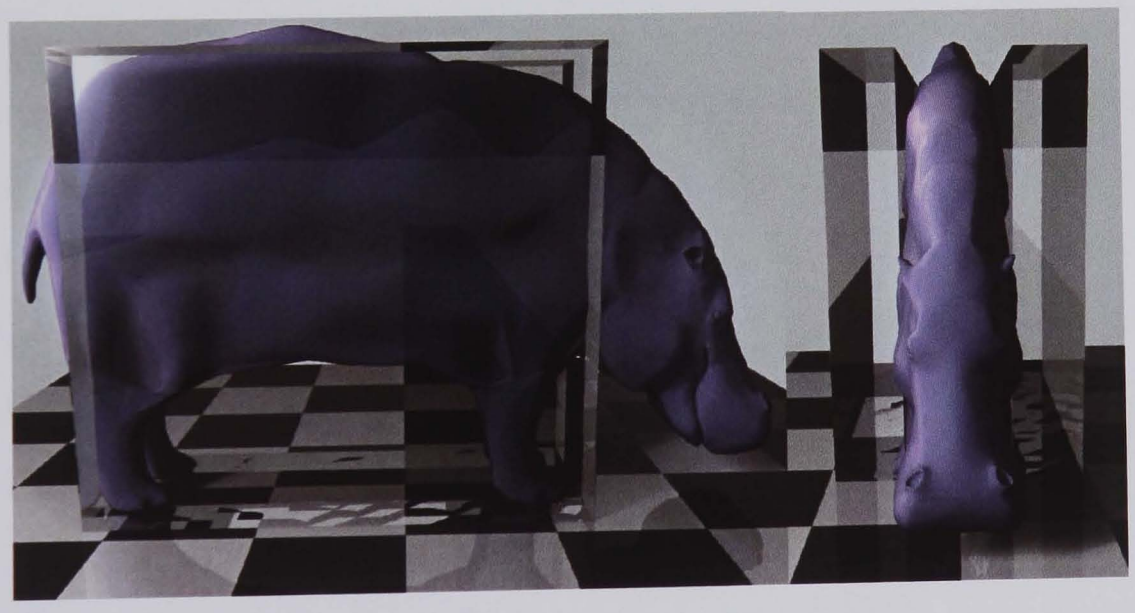
Figure 5.8(a) and (b) also show the images when the bunny's head is dragged around. Due to the volume preservation during deformation, the neck and the body deform naturally when head is pulled.

5.5.4 Hippopotamus Model

In Figure 5.9, a hippopotamus model (Wiedermann 2002) is being squeezed from both sides. It is noted that the hippopotamus deforms naturally and its volume remains nearly unchanged during the process. As the model is assumed to be elastic and to behave like a rubber toy, the shape of the head also deforms due to the squeezing force on the body.



(a) Original model



(b) Deformation model

Figure 5.9 Hippopotamus model being squeezed.

5.6 Advantages of Being Mesh-free

In this chapter, we have presented a deformation method whose distinctive feature is the elimination of the meshing process. Both the deformations and surface forces can

be computed accurately and efficiently. It demonstrated the basic use of the mesh-free framework.

In addition to eliminating the laborious meshing process, being mesh-free, this method has many advantages:

- Deforming a geometrically complex object is just as easy as for simple objects, since the CPs and VSPs are easily distributed.
- It can even be used for point clouds where the internal geometrical relations are unknown. This property is especially valuable in situations where topological and geometrical information is missing, such as for point based graphics applications.
- The resolution and accuracy can be controlled by having a different number of CPs and VSPs. Thus the trade-off between speed and accuracy is controllable.
- The mesh-free method is fast, because the discretization is done only on the surface of the object and all the coefficients are determined directly without requiring integration.
- It has been observed that the result of the new method is more visually plausible than that of the FEM linear module, as it does not suffer from the drawback of deformation localisation caused by the meshing process. Figure 5.4 shows that the mesh-free method is able to generate similar effects to those which are only achievable with the FEM large deformation module, but with a fraction of the computing cost.

Although in the given examples only surface deformations were considered, this method is equally applicable to volumetric models. The CPs can be distributed both on the surface and inside an object, where deformations can be picked up. This property may not seem crucial to animation, but it will be useful in other graphical simulation applications, such as medical visualisation.

5.7 Conclusions

A mesh-free framework for deformation modelling was demonstrated in this chapter, which aimed at computer graphics applications. Discretized points were used to capture both the geometric and mechanical information in the solving process. A deformation technique has been developed within this framework.

The mesh-free deformation method presents a complicated deformation as a sum of fundamental solutions. The Kelvin solution, the case of a single load exerting on an infinite body, is chosen as the fundamental solution. Two sets of points are used in this method. The CPs are used to capture the geometric shape of object and the applied boundary conditions. The VSPs define the distribution of virtual forces, one of which corresponds to a concentrated force.

This method is flexible and robust. It is suitable for both the complicated object and the simple object because the sampling points can be automatically created from the vertices. The solving process is an optimization process which can handle defects in the geometric model. For instance, the Bunny's model has holes on its bottom and neck (Figure 5.6) due to the loss of data in laser scan, and such defects did not cause any problem in our modelling and solving.

Our method does not only relieve the manual effort in modelling, but also provides a user-friendly interface that allows easy control of the trade-off between efficiency and accuracy by re-sampling the VSPs and CPs. The computational efficiency of the presented technique is comparable with other physically-based deformation methods. However, more and more applications require real-time performances. In the next chapter, acceleration algorithms will be investigated.

CHAPTER 6

FAST MESH-FREE DEFORMATIONS

The mesh-free deformation framework and technique developed in Chapter 5 provided an effective method in animating deformable objects and characters. It is flexible, accurate and easy to implement. Similar to other physically-based deformation techniques, however, the computational cost remains a pressing issue, especially for large scale problems. In this chapter we investigate the underlying structure of the numerical approach in order to speed up the computation. Two different schemes are proposed to optimise the computation: one is based on sparse matrix computation which allows the trade-off between efficiency and accuracy; the other is based on pre-computation where it is optimised so that the most work load is accomplished in advance to improve the run-time efficiency.

6.1 Sparse Matrix Scheme

In the sparse matrix scheme, three algorithms are experimented with and their efficiency and convergence are analysed. The results show that a proper algorithm is able to significantly reduce the computation time and memory usage which is paid off by slight degeneration in accuracy. The new technique is capable of achieving interactive frame rates for most applied models.

6.1.1 Shrinking Algorithms

The main computational cost of the mesh-free method comes from solving equations (5.10). When using the conjugate gradient method, the overall complexity of computation relies on the scale of the coefficient matrix. If the coefficient matrix [*A*] in equations (5.10) has *n* rows and *m* columns, it would take **O**(m×*n*) to iterate once in the conjugate gradient method.

It is observed that the influence of the Kelvin solution decreases dramatically with the increase of the distance between a virtual source point (VSP) and a collocation point (CP). This is revealed in Figure 4.2 that the surface deforms little when it is far away from the concentrated force. In displacement solutions (5.4), the actual value of the Kelvin solution U_{kl} drops in the scale of $1/r$, where r is the distance between the related CP and VSP. In surface force solutions (5.5), the value of Kelvin solution T_{kl} drops in the scale of $(1/r)^2$.

This decrease implies that for each CP, there are a small number of sub-items in equations (5.4) and (5.5) dominating the final results. Only the VSPs close to the CP play an important role in computation. Thus it is intuitive to discard the much less important items: those values vanish when assembling coefficient matrix [A], which leads to a sparse matrix for computation. This process is called shrinking.

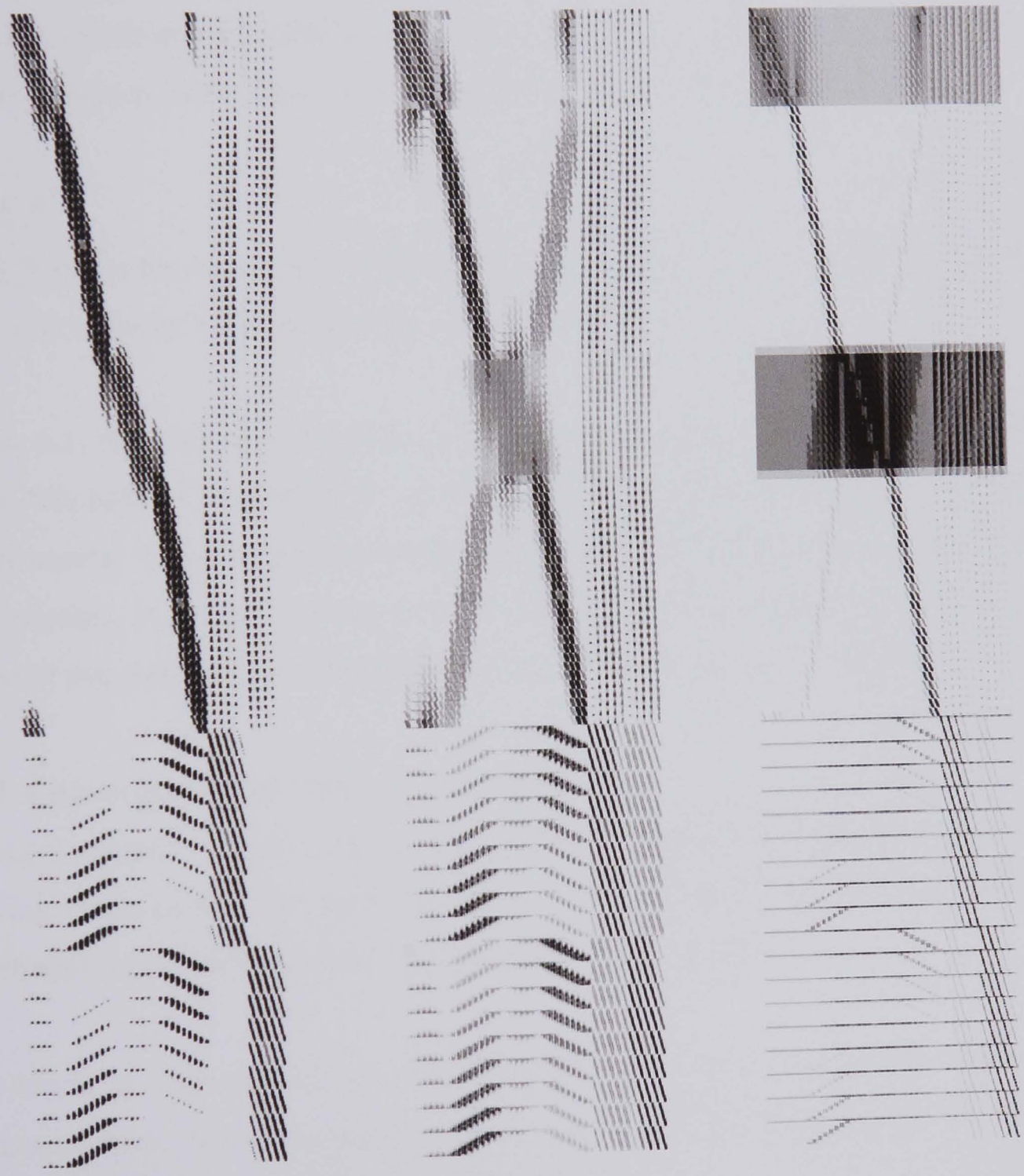
The following outlines the shrinking algorithm:

1. Assembling coefficient matrix [*A*], which is shrunk into a sparse one by vanishing items with minor influence;
2. Solving the equations (5.10) with sparse [*A*] from step 1 and caching all the virtual forces;
3. Computing the displacements on all CPs by equations (5.4) with the solved virtual forces from step 2.

After shrinking, the memory usage is saved in computation as there is no need to store the zero elements and the computation time is saved as there is no need to

calculate the multiplications with zero elements. It is important in step 3 to calculate the final result with all virtual forces. Using only dominant items could result in inconsistency and could violate the equilibrium state.

To shrink the coefficient matrix, three algorithms, MSH_V3, MSH_C, and MSH_P, are presented and examined. If a matrix is shrunk to x%, it means that the non-zero elements take x% of the matrix.



(a) MSH_V3 (b) MSH_C (c) MSH_P

Figure 6.1 The coefficient matrix when it is shrunk to 10%.

MSH_V3

MSH_V3 works point by point. It highlights the VSPs which are viewed as dominant ones related to a given CP. Only those VSPs are used to compute the coefficient elements of the CP in the coefficient matrix, and the other elements are set to zero.

MSH_C

Instead of working out point by point, MSH_C shrinks the coefficient matrix row by row. The elements with small absolute value in each row of coefficient matrix are set to zero and each row is made to have a fixed number of non-zero elements.

MSH_P

MSH_P treats the matrix as a whole, which sets element in the coefficient matrix as zero if its absolute value is less than a given threshold.

Figure 6.1, from left to right, displays the results of the matrix shrinking to 10% by MSH_V3, MSH_C and MSH_P, where the grey areas present the non-zero elements in the matrix. The example is a beam under pulling whose definition is given in the next section. It is very obvious that different shrinking algorithms lead to very different non-zero element distributions shown as the patterns in Figure 6.1.

6.1.2 Convergence and Efficiency Analysis

The convergence is first examined by studying a beam under pulling, bending and twisting respectively. The beam in Figure 5.3 is selected. 588 CPs are uniformly distributed on the surface of the beam and 344 VSPs are used.

Two kinds of error metrics are introduced here to evaluate the results. One is the maximum error, which considers the maximum difference among all the sample points comparing with the method without speed-up.

$$err_{\max} = \max(\left|\mathbf{u}(Q_i) - \mathbf{u}^0(Q_i)\right|) / scale \tag{6.1}$$

where $\mathbf{u}$ is the displacement at point Q_i computed by shrunk matrix, $\mathbf{u}^0$ is the displacement computed by the method without any speed-up and the *scale* is the size of the object, which is defined as the length of the beam in this example.

The other metric is the average error, which considers the average difference of all the sample points comparing with the method without speed-up.

$$err_{ave} = \frac{\sum_{i=1}^{n} \left| \mathbf{u}(Q_i) - \mathbf{u}^0(Q_i) \right|}{n \cdot scale} \tag{6.2}$$

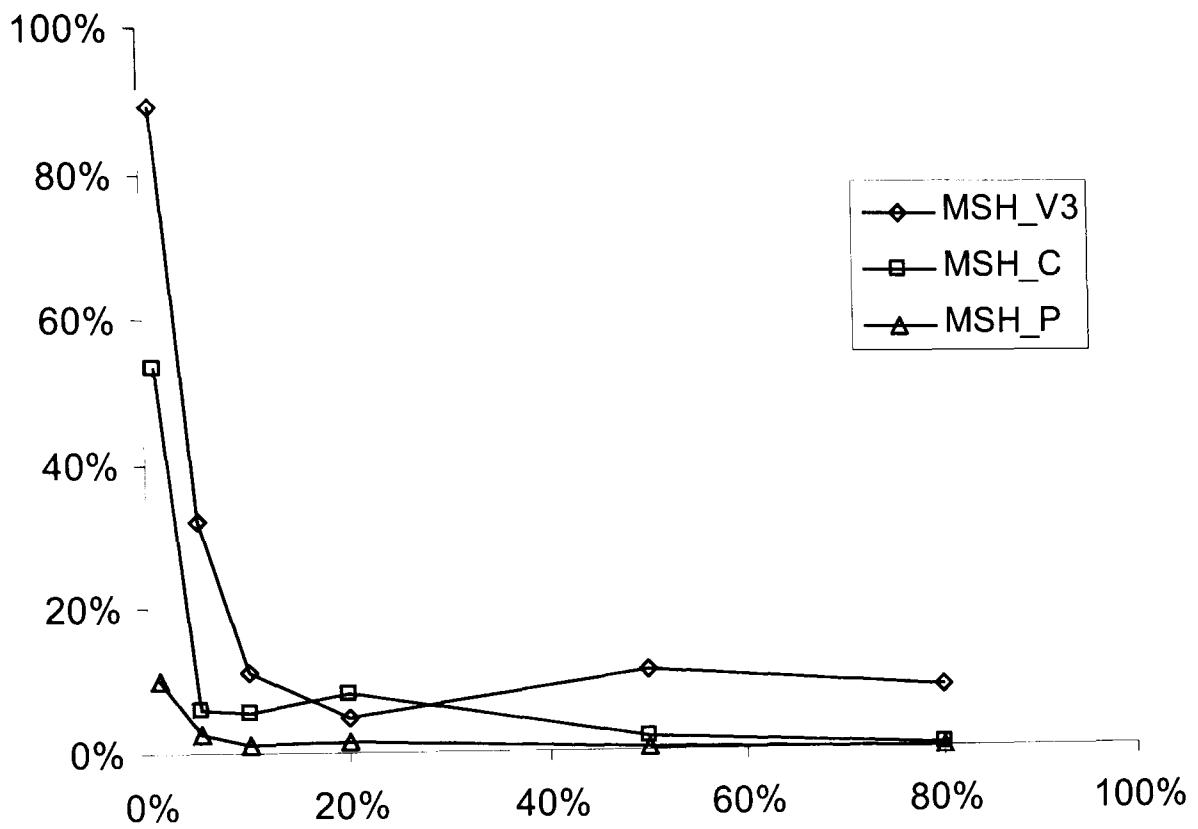
where n is the number of the sample points.

Figure 6.2, 6.3 and 6.4 display the maximum error and the average error by using MSH_V3, MSH_C and MSH_P to compute the beam subject to pulling, bending and twisting load, where the y-axis denotes the error and x-axis denotes the percentage of non-zero elements in the coefficient matrix.

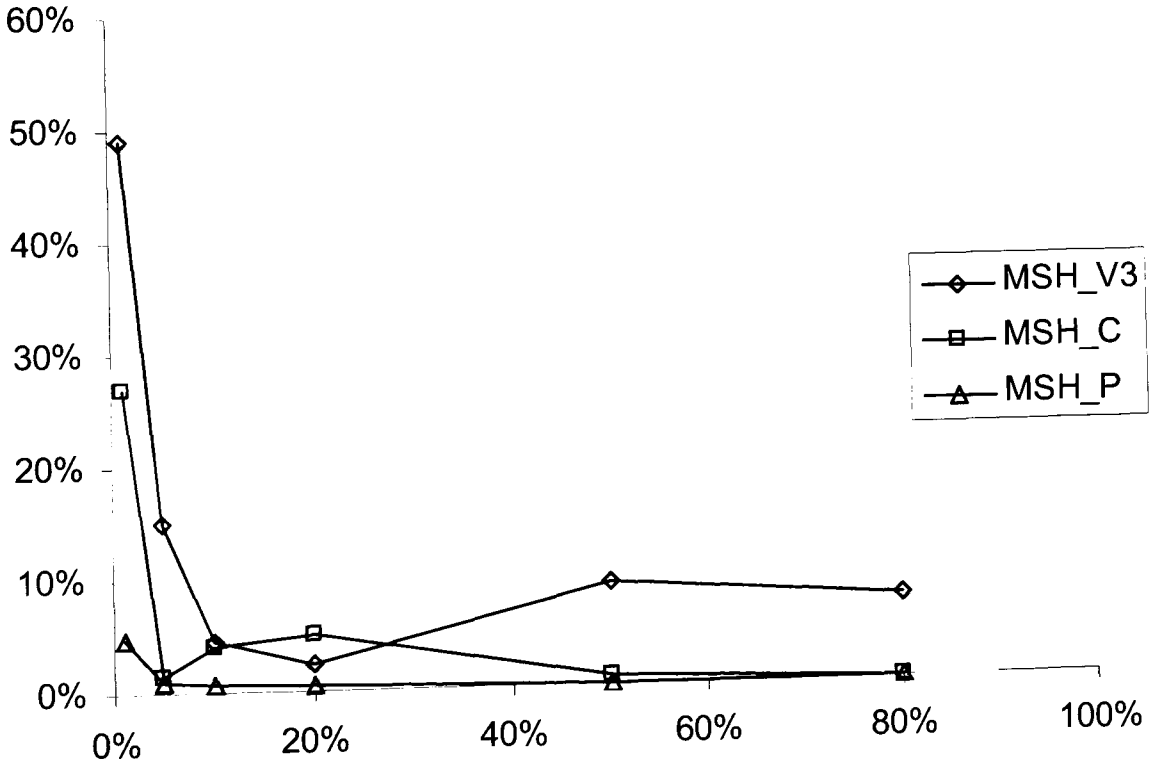
It shows that all the three methods converge to the correct result with the increase of nonzero elements in the coefficient matrix. The overall performances of the three algorithms, however, are different: MSH_P provides much better results than the other two. It is noticed that MSH_P provides a very good approximation even when the matrix is shrunk to 1%.

Figure 6.5 shows the resulting shapes of a beam under twisting, which lists all the results of three different shrinking algorithms with different shrinking percentages. MSH_P provides the best visual result.

In Figure 6.6, the efficiency of the shrinking scheme is investigated, where the x-axis stands for the percentage of the non-zero elements in the coefficient matrix and y-axis is the computational time divided by that without any speed-up. It is clear that the three algorithms cost nearly the same amount of time with the same shrinking percentage. The time saving is proportional to the percentage of the zero elements in the matrix.

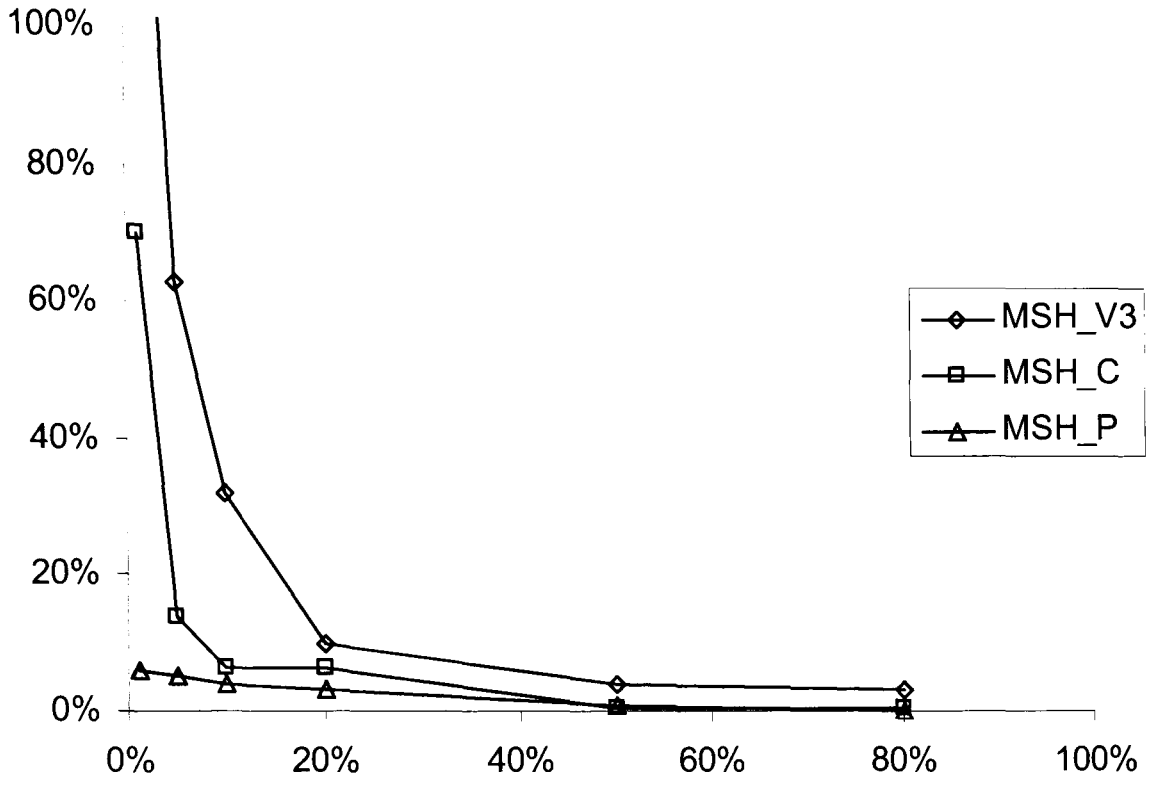


(a) Maximum error

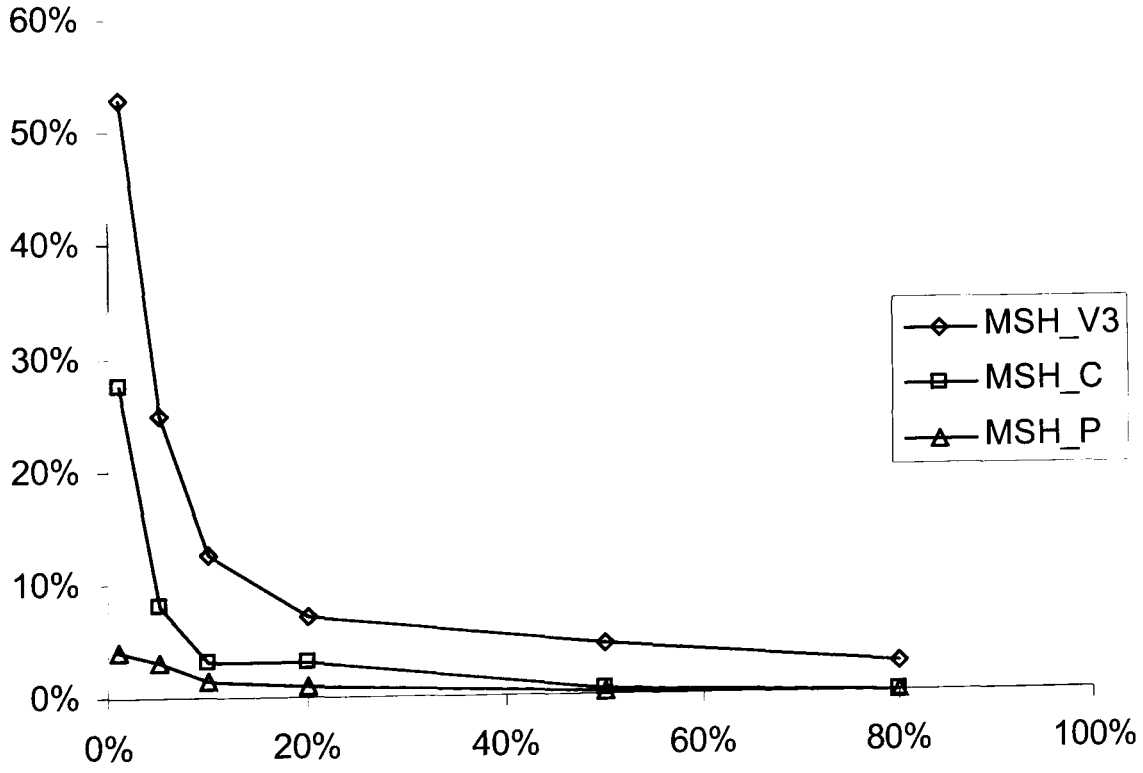


(b) Average error

Figure 6.2 Errors for beam under pulling

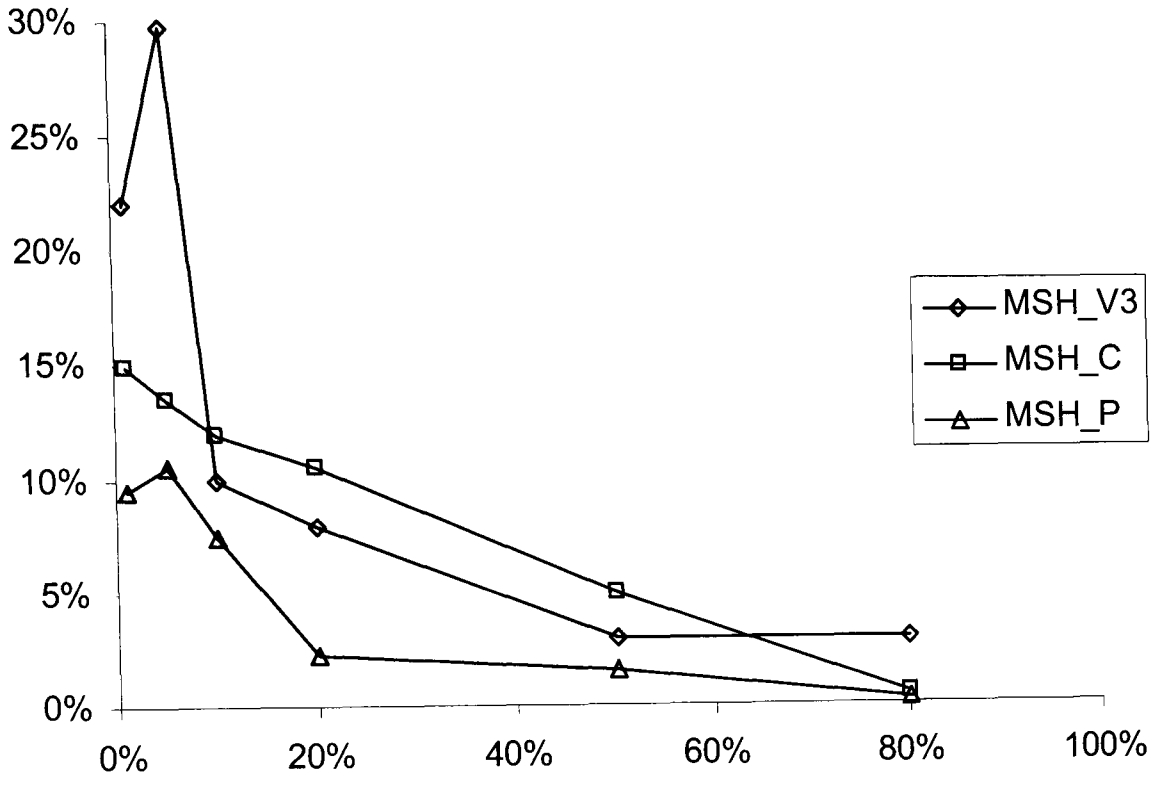


(a) Maximum error

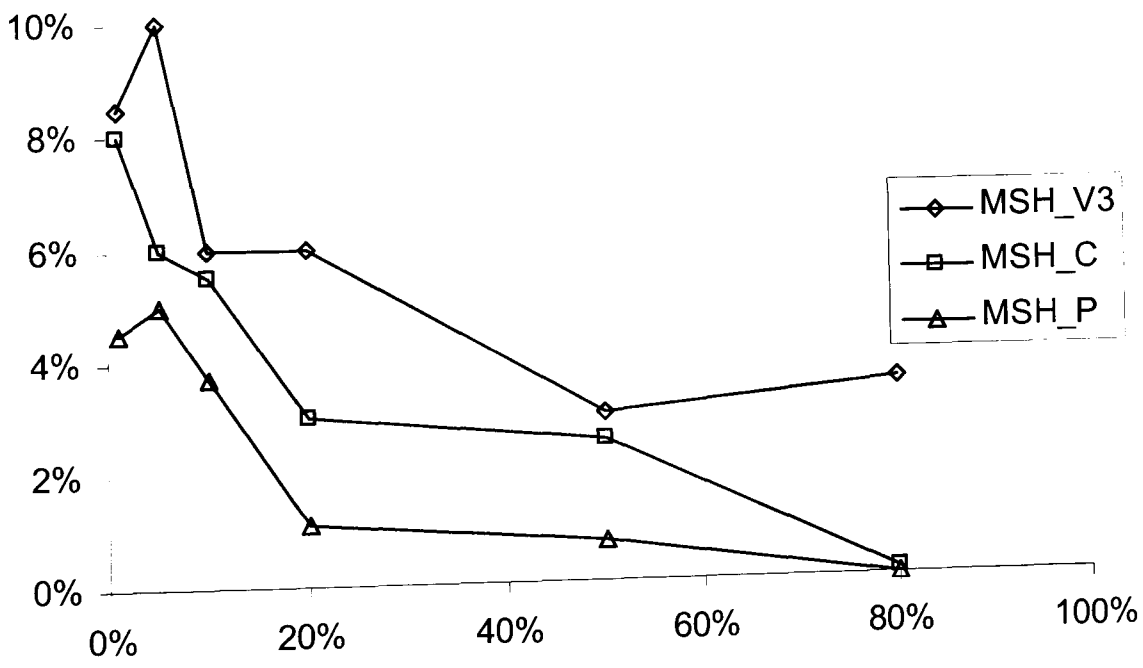


(b) Average error

Figure 6.3 Errors for beam under bending

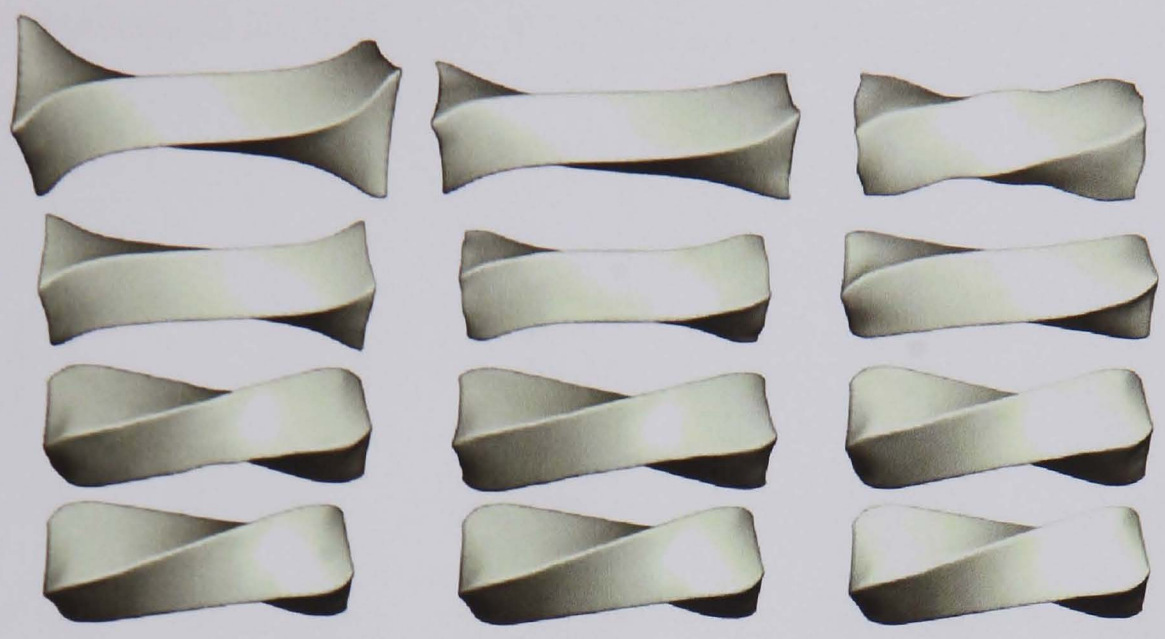


(a) Maximum error



(b) Average error

Figure 6.4 Errors for beam under twisting



(a) MSH_V3 (b) MSH_C (c) MSH_P

Figure 6.5. Results of a beam under twisting (from top to bottom: non-zero elements in coefficient matrix: 1%, 10%, 50% and 100%)

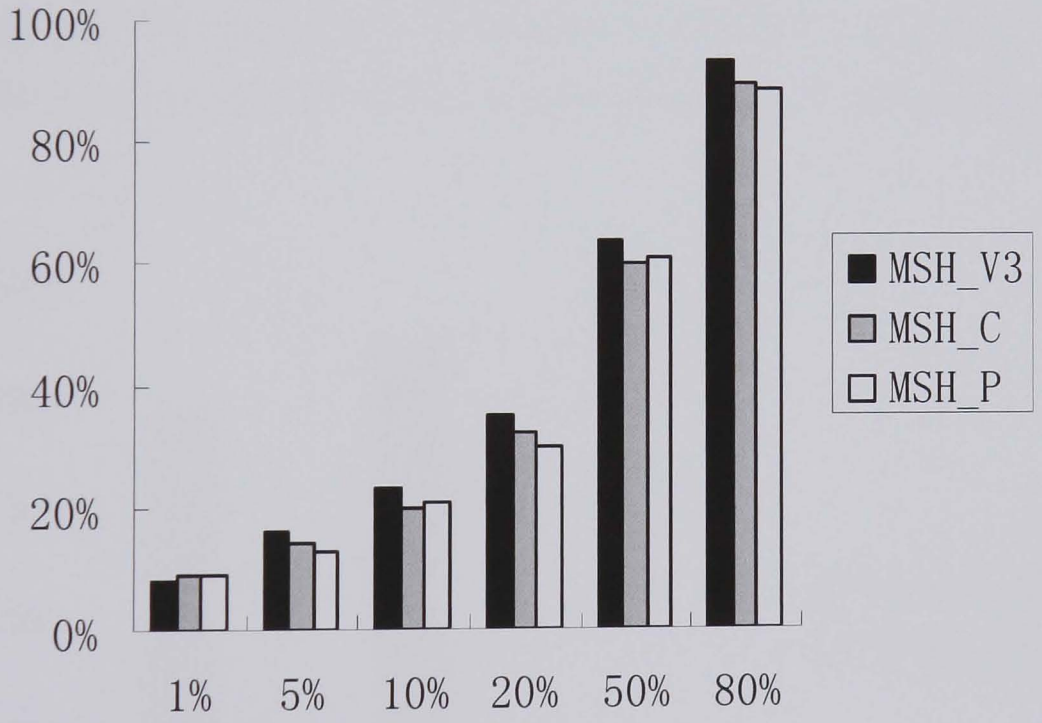


Figure 6.6 Computational time (The computational time of the original method without acceleration is set to *100%*).

In Figure 6.7, three shrinking algorithms are applied to a more complex model, the rabbit (Cyberware 2005), which has 13471 vertices. 579 VSPs are used. The rabbit's back and bottom are fixed. It is dragged as illustrated by the arrows in Figure 6.7(e). Figure 6.7(a), (b) and (c) show the results of applying the different algorithms, respectively. The matrix is shrunk to 10%. Figure 6.8 plots the errors for the rabbit.

It is not surprising to see that MSH_P provides the smallest error here and the best visual result in Figure 6.7.

(a) (b) (c) (d) (e)

Figure 6.7 Results of deformed rabbit: (a) MSH_V3, (b) MSH_C, (c) MSH_P, (d) mesh-free method without speed-up, (e) loads illustration.

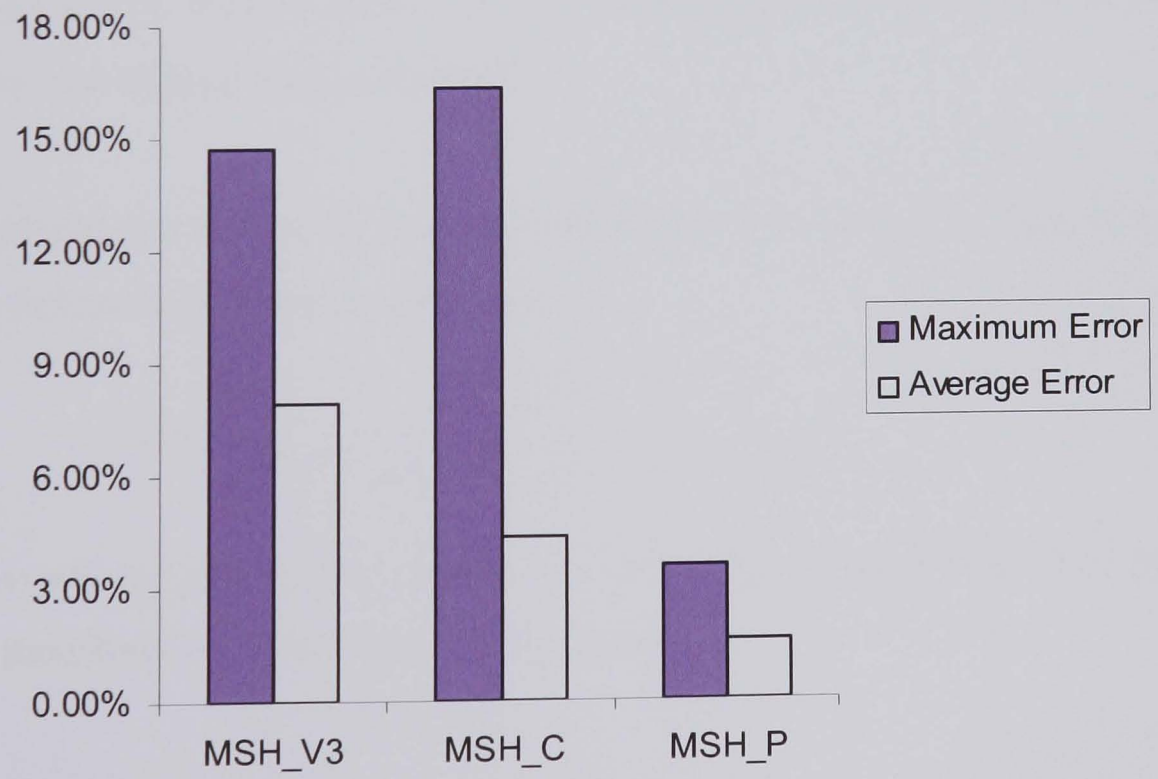


Figure 6.8 Errors of the rabbit model

6.2 Pre-computation Scheme

The sparse coefficient matrix was introduced in Section 6.1 to achieve better performance, which compromised simulation accuracy. The pre-computation scheme is able to overcome this shortcoming by means of separating and pre-computing those quantities which do not change when frame updates. By identifying and caching those quantities that only need to be calculated once, we demonstrate how fast performance is achieved at runtime. This technique is applicable for applications requiring fast response or to bulk render a sequence of frames when a large number of frames of the same object are in queue. There is no sacrifice in accuracy with pre-computation and it would be more suitable than the sparse matrix scheme for applications where the physical correctness is essential, e.g. virtual surgery simulation.

6.2.1 Pre-computation Scheme

It is observed that matrices $[U]$ and $[T]$ in equations (5.9) of Section 5.5.2 only need to be computed once, since the fundamental solutions only depend on the geometric information of the Virtual Source Points (VSPs) and the Collocation Points (CPs) which does not change once initialized.

By multiplying the inverse of the coefficient matrix on the both sides of equations (5.9), the following solution is obtained,

$$\{\alpha\} = \left(\begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix} \right)^{-1} \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \begin{Bmatrix} u^0 \\ \beta \cdot p^0 \end{Bmatrix} \tag{6.3}$$

The deformations and surface forces are computed by equations (5.4) and (5.5). They are rewritten in the following matrix form

$$\{u\} = [V]\{\alpha\} \tag{6.4}$$

$$\{p\} = [B]\{\alpha\} \tag{6.5}$$

where $\{u\}$ is a vector of the unknown displacements and $\{p\}$ the unknown surface forces.

After substituting equations (6.3) into (6.4) and (6.5), they become

$$\{u\} = [S]\begin{Bmatrix} u^0 \\ \beta \cdot p^0 \end{Bmatrix} \tag{6.6}$$

$$\{p\} = [K]\begin{Bmatrix} u^0 \\ \beta \cdot p^0 \end{Bmatrix} \tag{6.7}$$

where

$$[S] = [V]\left(\begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}\right)^{-1} \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \tag{6.8}$$

$$[K] = [B]\left(\begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}\right)^{-1} \begin{bmatrix} U \\ \beta \cdot T \end{bmatrix}^T \tag{6.9}$$

The coefficient matrices [S] and [K] whose elements are all constant can be pre-computed. $\{u\}$ and $\{p\}$ are updated with a small expense on computations with the pre-computed [S] and [K]. In run time, the computation cost is just a single matrix multiplication.

If the object is subject to a set of time-dependent boundary conditions, i.e. the boundary displacements and/or forces change over time, the object will deform continuously. Because the deforming objects are elastic, it is economic to utilize the incremental formulations of both the displacements and the surface forces, i.e. $u^{(k+1)} = u^{(k)} + \Delta u^{(k)}$ and $p^{(k+1)} = p^{(k)} + \Delta p^{(k)}$, where $u^{(k)}$ and $p^{(k)}$ are the values at the previous time step, $u^{(k+1)}$ and $p^{(k+1)}$ at current time step. Then the increments of both displacements and surface forces can be given by:

$$\left\{\Delta u^{(k)}\right\} = [S]\begin{Bmatrix} \Delta u^0_{(k)} \\ \beta \Delta p^0_{(k)} \end{Bmatrix} \tag{6.10}$$

$$\left\{\Delta p^{(k)}\right\} = [K]\begin{Bmatrix} \Delta u^0_{(k)} \\ \beta \Delta p^0_{(k)} \end{Bmatrix} \tag{6.11}$$

where $\Delta u^0_{(k)}$ and $\Delta p^0_{(k)}$ are the increments of the boundary condition quantities.

In most cases, only a small number of entries of the boundary conditions may change in one step. Therefore, only those nonzero entries are taken into account when updating by equations (6.10) and (6.11). This allows a further time saving on top of the already efficient formulation of (6.8) and (6.9).

Compared with the iterative method presented in Chapter 5, which directly solves the problem, this method is able to produce a fast response at the price of pre-computation. It is more suitable to interactive or real-time applications. However, if only a single or a small number of frames need to be updated, the iterative method is usually a better choice, as the pre-computation step is expensive. There exists an important feature that makes the pre-computation advantageous: the pre-computation provides exactly the same result as the one without any speed-up which makes sense in many applications with a high accuracy requirement.

6.2.2 Test of Pre-computation Scheme

Figure 6.9 Screenshots generated by the method of pre-computation

A test on the pre-computation scheme is also undertaken on the Stanford bunny model (Stanford 2004). Figure 6.9 shows a number of frames produced by this

method. The bunny is fixed at the base but its head can be dragged around. The computation performance is compared in Table 6.1.

Table 6.1 Computation cost for the bunny model (Dell Pentium III 500 MHz PC)

No. of CPs	No. of VSPs	Computation process	Time per frame
940	432	Without Pre-computation	6.78 s
940	432	With Pre-computation	0.03 s
940	94	Without Pre-computation	1.07 s
940	94	With Pre-computation	0.03 s

From the above results, it shows the pre-computation is very appealing and the speed-up is very impressive. But it must be kept in mind that the pre-computation step is very expensive as it involves the inverse operation of coefficient matrix. It cost about 20 minutes for the bunny with 432 VSPs and 940 CPs and about 2 minutes for the bunny with 94 VSPs and 940 CPs.

6.3 Conclusions

In this chapter, based on the original mesh-free deformation method in Chapter 5, two different numerical techniques have been proposed to optimise the computation. One is based on sparse matrix computation and the other is based on pre-computation.

In the sparse matrix scheme, the trade-off between efficiency and accuracy is controlled by the shrinking ratio of the matrix. The more elements in the matrix that are shrunk, the faster the computations are but greater errors are introduced into the results. The computational time is nearly proportional to the percentage of non-zero elements in coefficient matrix. The saving of memory usage is similar. In our experiment, a high shrinking ratio can be adopted with acceptable loss in accuracy. In the example of the deformed beam, shrinking to 1% led to about 94% saving in computational time and about 99% saving in memory usage, while the induced average error was only about 5% with MSH_P. Three matrix shrinking algorithms

were presented: MSH_V3, MSH_C and MSH_P. The experiments showed that MSH_P is robust and provides better results than the other two algorithms. This is because it considers the coefficient matrix as a whole when performing the shrinking operation. Shrinking the coefficient matrix to 10% provides a good balance between accuracy and efficiency. In fact, the experiments suggest the coefficient matrix can be shrunk up to 1% without losing significant accuracy.

By identifying and pre-computing those quantities that only need to be calculated once, the pre-computation scheme has been developed. The pre-computation optimises the solving process and it leads to no loss in accuracy, which is advantageous in those applications where the accuracy is as vital as the computing speed, e.g. virtual surgery simulation. The pre-computation is time-consuming but it is a one-off operation. Once it is done the frames can be updated instantaneously. With the extra cost in pre-computation stage, the relatively high speed is achieved in run time.

CHAPTER 7

REUSABLE DEFORMATIONS

In previous chapters, we have discussed a novel physically-based technique under the mesh-free deformation framework, which works on discretized points with no need of underlying meshes. In this chapter, we extend the basic mesh-free deformations to allow versatile deformation effects to be created using simple "copy and paste" operations. In addition to visual realism, the main advantage of the technique over other existing physically-based methods is the reuse of deformation data, which is combined with the computational efficiency and the friendliness to animators.

The technique consists of two parts: extracting and combining known deformations; and applying them to different objects to achieve flexible new deformation effects. Various known deformations are represented as deformation fields. This is consistent with the mesh-free deformation framework. A deformation field is stored separately as a sum of fundamental solutions. These deformation fields are then regarded as deformation primitives and can be recorded in a deformation database. From this database one can choose necessary deformation primitives and combine them to produce a required deformation pattern, which is then applied to the target object.

The technique is envisaged to be useful in two kinds of applications.

(a) Deforming a complex object using a combination of simpler deformations. The animator can choose from the database a number of simple deformation fields and then combine and "paste" them onto the target object to achieve a complex deformation.

(b) Deforming a complex object by creating a simpler alternative. Deforming a complex object with a high polygon count can take a tremendous amount of computing power. With this technique, one can first reduce the level of detail of the object. The simplified object can be deformed using any appropriate techniques. Deforming a simpler object is much cheaper than deforming the original object. The user then extracts (copies) the deformation field from the simplified object and pastes it to the original object. This operation ensures both the original and simplified objects share essential shape and topological properties. Examples are given in this chapter to demonstrate that the technique can significantly speed up the calculation without losing noticeable visual quality.

7.1 Overview

The idea of "copy and paste" is straightforward as illustrated in Figure 7.1. If there are two geometric models: source A and target B, the user can determine the deformation of geometric model B through the known deformation data of model A. The "copy" operation is defined as extracting the deformation information from A and the "paste" operation is defined as transferring such deformation from A to B. The main difficulty is how to maintain the physical plausibility of deformation on B after the "copy" and "paste" operations.

The deformation field is retrieved through the influence of a set of virtual forces that is, to express deformation as a sum of Kelvin solutions following Chapter 5. This enables a compact storage and allows the deformation field to be represented separately from the geometry of the deformed object. The basis of transferring the deformations between two objects has been laid in this way.

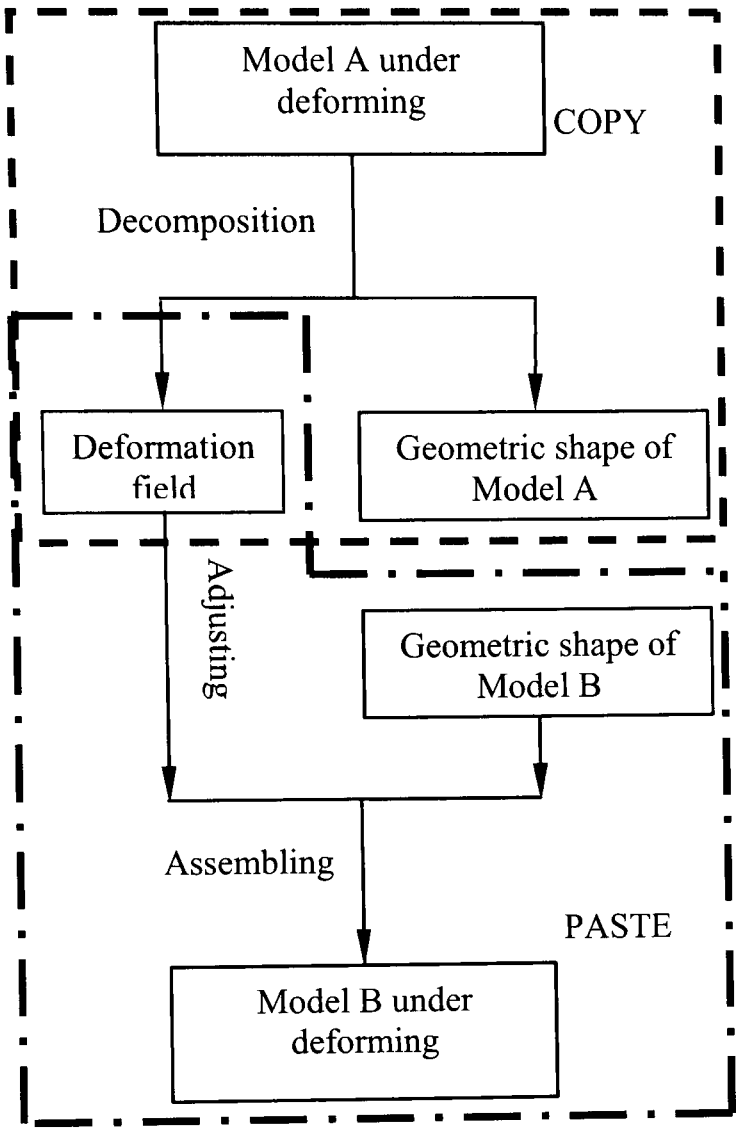


Figure 7.1 "Copy" and "paste" deformations from Model A to Model B

7.2 Deformation Field

As indicated by Figure 7.2, the deformed shape of an object can be regarded as the superimposition of the deformation upon its original geometric model. A displacement can be regarded as the simplest form of deformation. For example, given **u** as the displacement vector at an arbitrary point Q_0 on the original geometric model, the new position of that point Q on the deformed shape is given by

$$Q = Q_0 + \mathbf{u} \tag{7.1}$$

Subject to constraints and forces, the deformation may be very complicated. Equation (7.1) records the deformation as a displacement **u** on any given point Q_0. However, displacement **u** is meaningless without its reference point Q_0. Directly transferring displacement from one mesh to another is normally physically meaningless.

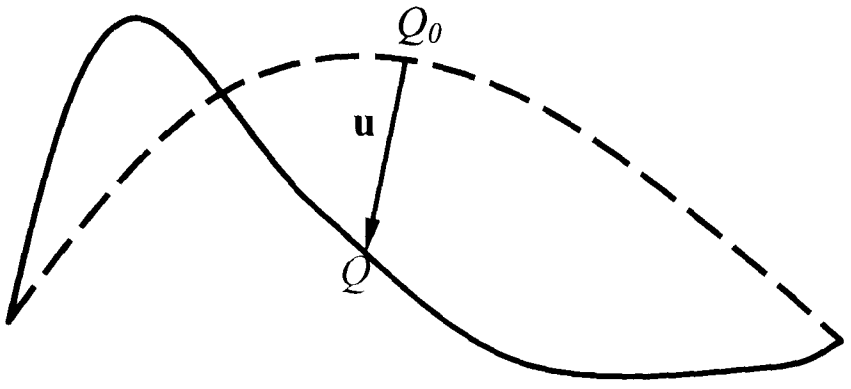


Figure 7.2 Deformation of a surface: new position Q after being deformed is a superposition of a given displacement **u** on the original position Q_0

A key step for the "copy and paste" operation is to separate the deformation from the geometry. Instead of using displacements, the deformation is redefined as a continuous disturbance function in a 3D space. Such disturbance is designed to be a combination of many basic standard modes which can be handled easily. Thus the first step to formulating such a disturbance function is to identify appropriate basic standard modes.

The Kelvin solution of Chapter 4 gives an analytical description of the disturbance of an infinite elastic body under a unit concentrated force. Although an infinite elastic body does not exist in the real world, the Kelvin solution is a good candidate and can serve the purpose well. Thus it is used to represent the basic disturbance modes.

Using the Kelvin solution, given a point and a concentrated force on it, a basic disturbance in 3D space is defined. If an object is wrapped within a sphere or an ellipsoid whose surfaces are represented by discrete vertices as shown in Figure 7.3, these vertices can be regarded as the points where the concentrated forces act on. By carefully adjusting these forces, a specified deformation effect can be approximated to satisfy the equilibrium partial differential equations. At each point, three concentrated forces are applied in the direction of x_i (i =1, 2, 3), respectively, which induces three basic disturbance modes. Here, following the definition in Chapter 5,

the points are called the virtual source points (VSPs) and the concentrated forces are called the virtual forces. The formulation of the displacement is identical to that of mesh-free deformations. If there are n VSPs, the resultant displacement u_l at point Q in direction l can be written as

$$u_l(Q) = \sum_{i=1}^{n}\sum_{k=1}^{3} \alpha_{ik} U_{kl}(P_i, Q), \quad l = 1,2,3, \quad Q \in \Omega, \quad P_i \notin \Omega \tag{7.2}$$

where Ω is the domain of the deforming object, and α_{ik} is the amount of the virtual force along x_k on VSP P_i. The Kelvin solution of displacement $U_{ij}(P, Q)$ is given out in equations (4.8).

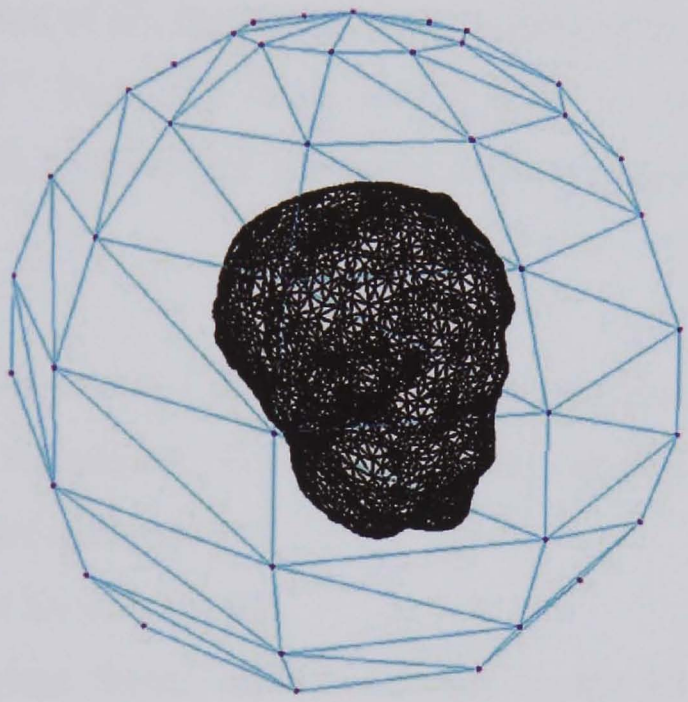


Figure 7.3 Object is wrapped by a sphere of VSPs.

With this treatment, the deformation of the object is cached by the virtual forces α_{ik} on VSPs and can be retrieved physically by the superimposition of the Kelvin Solutions as given in equations (7.2). Thus a deformation field is defined independently from the reference geometry. Not only the displacement but also the other physical properties are able to be calculated in analogous formulation with the help of the related type of Kelvin solutions. For instance, surface forces can be computed by equations (5.5) with known VSPs and virtual forces.

The material properties of an object are described by shear modulus G and Poisson's ratio ν. Stiff materials have a large shear modulus and they are resistant to deformation. Poisson's ratio can vary from -1 to 0.5, but the negative values do not usually appear in reality. Poisson's ratio $\nu = 0.5$ represents complete incompressibility.

7.3 Copy and Paste

Given two different objects Ω_1 (source) and Ω_2 (target), and the deformation of Ω_1, this section describes how to deform Ω_2 using the deformation field of Ω_1.

With known deformation of Ω_1, the virtual forces acting on the VSPs are determined and recorded — "copy". Then, the forces on the VSPs are applied to object Ω_2 — "paste". As a result, Ω_2 is deformed accordingly. Both operations are explained in detail as follows.

Let's suppose the deformation of object Ω_1 is already obtained, which can be done through any existing techniques. To determine the virtual forces in the copying operation, these steps are executed:

1. wrapping object Ω_1 with n VSPs;
2. formulating virtual forces $\alpha_{ik}^{(1)}$ in relation to the known deformation, as below

$$\sum_{i=1}^{n}\sum_{k=1}^{3}\alpha_{ik}^{(1)}U_{kl}(P_i,Q^{(1)}) = u_l(Q^{(1)}),\ \ l=1,2,3,\ \ Q^{(1)}\in\Omega_1,\ \ P_i\notin\Omega_1 \qquad (7.3)$$

   where superscript (1) denotes that the variable is related to object Ω_1.
3. solving equations from (7.3) for the unknown virtual forces $\alpha_{ik}^{(1)}$ with the least square method.

In the experiment, uniformly distributed points on a sphere, which contains the deformable object, are selected as VSPs as shown in Figure 7.3. This works well, although different distribution of VSPs is acceptable. With only 200 VSPs, the deformations can be approximated to certain accuracy as our examples show

Having obtained the virtual forces from object Ω_1, Ω_2 is wrapped with the same set of VSPs. The deformation of Ω_2 is determined directly when it occupies the space of Ω_1 by

$$u_l(Q^{(2)}) = \sum_{i=1}^{n}\sum_{k=1}^{3} \alpha_{ik}{}^{(1)}U_{kl}(P_i, Q^{(2)}), \quad l = 1,2,3, \quad Q^{(2)} \in \Omega_2, \quad P_i \notin \Omega_1 \cup \Omega_2 \qquad (7.4)$$

These equations suggest that object Ω_2 has similar deformation effect to object Ω_1 if they have similar shapes before deformation.

When a deformation is pasted to a target object that has a different location, orientation and size to the source object, additional preparation is necessary in order to ensure accuracy. Examining the Kelvin solution (4.8), the deformation of a surface point Q on an object actually depends on the relative position between the surface point Q and the virtual source points P rather than the absolute position of Q. Therefore, the distribution of the VSPs is able to be adjusted by rotating and scaling to suit the target object.

An interesting property of the virtual forces is that they are invariant from translation, but are susceptible to rotation of the VSPs. When the target object is the source object rotated by some degrees, to achieve similar deformation effect, the displacement on a point of the source object will rotate the same degrees before applying to the target. That is when $Q^{new} = \mathbf{R} \cdot Q^{old}$, we have

$$Q^{new} + \mathbf{u}^{new} = \mathbf{R} \cdot (Q^{old} + \mathbf{u}^{old}) = Q^{new} + \mathbf{R} \cdot \mathbf{u}^{old},$$

where $\mathbf{R}$ is the rotation matrix, Q is a given point on the object and $\mathbf{u}$ is the displacement. With equation (5.4), we have

$$\mathbf{u}^{new} = \mathbf{R} \cdot \mathbf{u}^{old} \Rightarrow \sum_i \sum_k \alpha_{ik}{}^{new}U_{kl}(P_i^{new}, Q^{new}) = \sum_{m=1}^{3}\sum_i \sum_k R_{lm}\alpha_{ik}{}^{old}U_{km}(P_i^{old}, Q^{old}),$$

where R_{lm} is the element of the rotation matrix $\mathbf{R}$ and P_i is the i^{th} VSP. From equation (4.8), we also have

$$U_{kl}(P_i^{new}, Q^{new}) = \sum_{m=1}^{3}\sum_{n=1}^{3} R_{mk}R_{nl}U_{mn}(P_i^{old}, Q^{old})$$

if the VSPs follow the same rotation. The relation of the virtual forces before and after rotation can be deduced by substituting the latter equation into the previous one. This gives grounds for reformulation of the problem. Let **d** be the translation and **R** the rotation matrix with R_{km} as its elements. It comes out that,

$$P_i^{new} = \mathbf{R} \cdot P_i^{old} + \mathbf{d}, \text{ leads to } \alpha_{lk}{}^{new} = \sum_{m=1}^{3} R_{km} \alpha_{lm}{}^{old} \tag{7.5}$$

Scaling the VSPs with factor ζ also leads to a change of virtual forces. When the target object is the source object scaled by a factor ζ, to achieve similar deformation effect, the displacement will be scaled with the same factor before applying to the target. That is

$$Q^{new} + \mathbf{u}^{new} = \zeta \cdot (Q^{old} + \mathbf{u}^{old}) = Q^{new} + \zeta \cdot \mathbf{u}^{old}.$$

From equations (4.8), the displacement on a given point is proportional to the reciprocal of its distance to the VSP. In order to compensate the change, the virtual forces are scaled by ζ^2. The following treatment is undertaken before equations (7.4) are applied

$$P_i^{new} = \zeta \cdot P_i^{old}, \text{ leads to } \alpha_{lk}{}^{new} = \zeta^2 \alpha_{lk}{}^{old} \tag{7.6}$$

Therefore the general operation for the paste operation should substitute the updated VSPs and virtual forces from equations (7.5) and (7.6) into equation (7.4).

The reader may wish to look at the problem in a reversed order. Let us keep the VSPs and virtual sources unchanged, but to translate, rotate and scale object Ω_2 to suit the VSPs. That is

$$Q^{(2)new} = \zeta \cdot \mathbf{R} \cdot Q^{(2)old} + \mathbf{d} \tag{7.7}$$

with **d** as the translation and **R** as the rotation matrix

The new updated position from equations (7.7) is substituted into (7.4), and a displacement $\mathbf{u}(Q^{(2)new})$ is related to the new position $Q^{(2)new}$. The final displacement $\mathbf{u}(Q^{(2)old})$ for the paste operation is derived as

$$\mathbf{u}(Q^{(2)old}) = \zeta^{-1} \cdot \mathbf{R}^{-1} \cdot \mathbf{u}(Q^{(2)new}) \tag{7.8}$$

This gives the same result as equations (7.5) and (7.6).

7.4 Composition of Deformations

Dealing with a complex deformation is not straightforward. To simplify this problem, deformation is decomposed into several basic deformation fields and then they are combined together to arrive at a required effect.

Two methods are proposed to superimpose basic deformations. With the first method, a deformation is decomposed into several different deformation types (phases) which are then combined and pasted onto the geometric model at the same time. This is called the aggregate composition. With the second method, a complex deformation is also divided into a series of different phases. But these phases are pasted one by one in a specific order. This is called the sequential composition. Different order may result in different effects.

7.4.1 Aggregate Composition

In this way, all the loads are applied at the same time. All phases are combined to create an aggregate deformation

$$\alpha_{lk} = \sum_i \beta_i \alpha_{lk}^{(i)} \tag{7.9}$$

where α_{ik} is the virtual force for the final deformation, β_i is the weight factor and $\alpha_{ik}^{(i)}$ is the virtual force for the ith phase.

Equations (7.9) suggest that the animator can adjust the value of the weight factor β_i for the ith phase to control its influence. A larger β_i leads to a bigger influence of phase i. Fine-tuning the weight factors controls the contribution of the input shapes to the final shape.

7.4.2 Sequential Composition

The sequential composition adds the phases in sequence. This can be used to simulate local deformation intuitively. Given a geometric shape S_0, a deformation is pasted to it to obtain a new shape S_1. After that another deformation is pasted onto the already deformed shape S_1 in order to create S_2. The overall effect is the combination of both deformations. These steps can be repeated until all available

deformation phases are added and pasted. This process is mathematically expressed as

$$
\begin{aligned}
& u_l^{(1)}(Q) = \beta_1 \sum_{i=1}^{n} \sum_{k=1}^{3} \alpha_{ik}{}^{(1)} U_{kl}(P_i, Q), \\
& Q_1 = Q_0 + \mathbf{u}^{(1)}; \\
& u_l^{(2)}(Q_1) = \beta_2 \sum_{i=1}^{n} \sum_{k=1}^{3} \alpha_{ik}{}^{(2)} U_{kl}(P_i, Q_1), \\
& Q_2 = Q_1 + \mathbf{u}^{(2)}; \\
& \cdots\cdots \\
& Q_n = Q_{n-1} + \mathbf{u}^{(n)};
\end{aligned}
\tag{7.10}
$$

where Q_i are the points on S_i, β_i is the weight factor and $\alpha_{ik}{}^{(i)}$ is the virtual force for the ith phase. The operation order is important to the final deformation.

Since sequential composition equivalently deforms an object step by step, it is effective in achieving local deformations, such as the poking action shown in Figure 7.8. It is slower than the aggregate composition but reveals different visual effects of different loading sequences.

7.4.3 Deformation Database

Both aggregate and sequential composition methods make use of simple and basic deformation fields, which can be computed easily. Furthermore, they can be stored in an extendable deformation database when every time a new deformation field is created. Objects of simple shapes, such as spheres, ellipsoids, tori and cubes are selected as source geometric primitives, since their deformations are easy to compute. Once an object and its deformed shape are created, the deformation is recorded as virtual forces at VSPs. Usually the VSPs are distributed uniformly on a sphere which is big enough for the object concerned. The same set of VSPs is used and only the virtual forces are needed to be stored. In the experiment, 234 virtual force vectors are sufficient for each case of deformation. This is much more compact compared with storing displacements for thousands of mesh vertices in most applications.

Besides using predefined uniformly distributed VSPs, the user can also define his own VSPs set and store them together with the virtual forces.

7.5 Results and Analysis

This section presents some examples to demonstrate the applications of the technique. All images in this section are rendered by Alias|Maya.

An ellipsoid is used as the source mesh Ω_1; and a more complex shape, the head of the Venus statue (Cyberware 2005) is used as the target mesh Ω_2 (Figure 7.4). The VSPs are distributed on a sphere, inside which both the Venus head and ellipsoid are placed as demonstrated in Figure 7.3. The same set of VSPs is used in all the examples. Since the ellipsoid is a simple geometric object, it is very easy to compute its deformation. This deformation is copied and pasted onto the complex geometries, in this case, the head of the Venus statue.

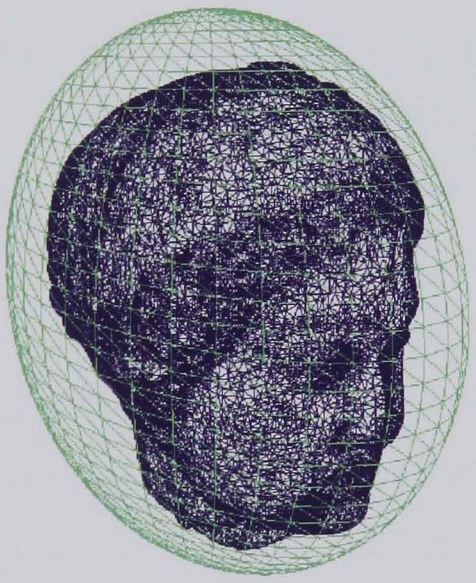


Figure 7.4 Geometric models: Venus head as Ω_2 and ellipsoid as Ω_1.

The material properties of the ellipsoid are set to: shear modulus $G=80$ and Poisson's ratio $\nu=0.5$. Twisting forces are applied on the top and bottom of the ellipsoid, respectively. The deformed shape of ellipsoid is computed with the mesh-free method of Chapter 5, as shown in Figure 7.5.

To study how the presented technique compares with other deformation approaches, the Venus head is deformed with the two different methods. First, it is deformed by the mesh-free method which is physically based, with the same twisting forces and material properties as given in Figure 7.6(a). Secondly, the Maya non-linear deformer is employed, which is purely geometrically based, to twist the Venus head. The generated result is given in Figure 7.6(b). By copying and pasting the deformation of the twisted ellipsoid in Figure 7.5 to the Venus head, the image in Figure 7.6(c) was obtained.

Comparing these images, it is found that Figure 7.6(a) and Figure 7.6(c) produce similar results. Since Figure 7.6(a) considers the physical laws of the deformation of the Venus head, it is the most accurate among the three approaches. Subtle details are revealed and the deformation looks realistic. The deformed shape created by the Maya deformation tool does not take into account the physical behaviour; details of deformation are not clearly shown. By the "copy and paste" method, the image in Figure 7.6(c) offers close approximation to the fine result of Figure 7.6(a). The twisting effect of the shape is evident. While the computation time for Figure 7.6(a) is about five minutes with additional ten minutes for the manual preparation, it took about two seconds to paste the deformation from the ellipsoid to the Venus head. No manual preparation was required. Such computational speed is comparable to the Maya deformer, which however considers no physical properties. To get the result of Figure 7.6(b), it took about three seconds' computation time, and this does not include the manual modelling efforts. To model and deform the ellipsoid is much easier than the Venus head and to paste the deformation to the Venus head achieves a visually satisfying result with much less computational overhead.

Figure 7.7 shows an example of copying the deformation of a heart model and then pasting it to the Venus head model. The shear modulus is kept unchanged. Poisson's ratio is set to zero allowing volume variation while the models are dilating and contracting.

First, the heart was deformed to create a few key phases of the beating action. The mesh-free method was used to calculate the deformations. Second, these key phases were weight-summed to create a continuous deformation sequence, which is done by the virtual forces as shown in equations (7.9). Finally the deformation was copied and pasted to the Venus head separately to generate the animation sequences. Some screenshots from the generated animation are given in Figure 7.7.

Figure 7.5 Deformed simple object: twisted ellipsoid.

(a) directly computed (b) by Maya (c) copied from ellipsoid

Figure 7.6 Twisted Venus head.

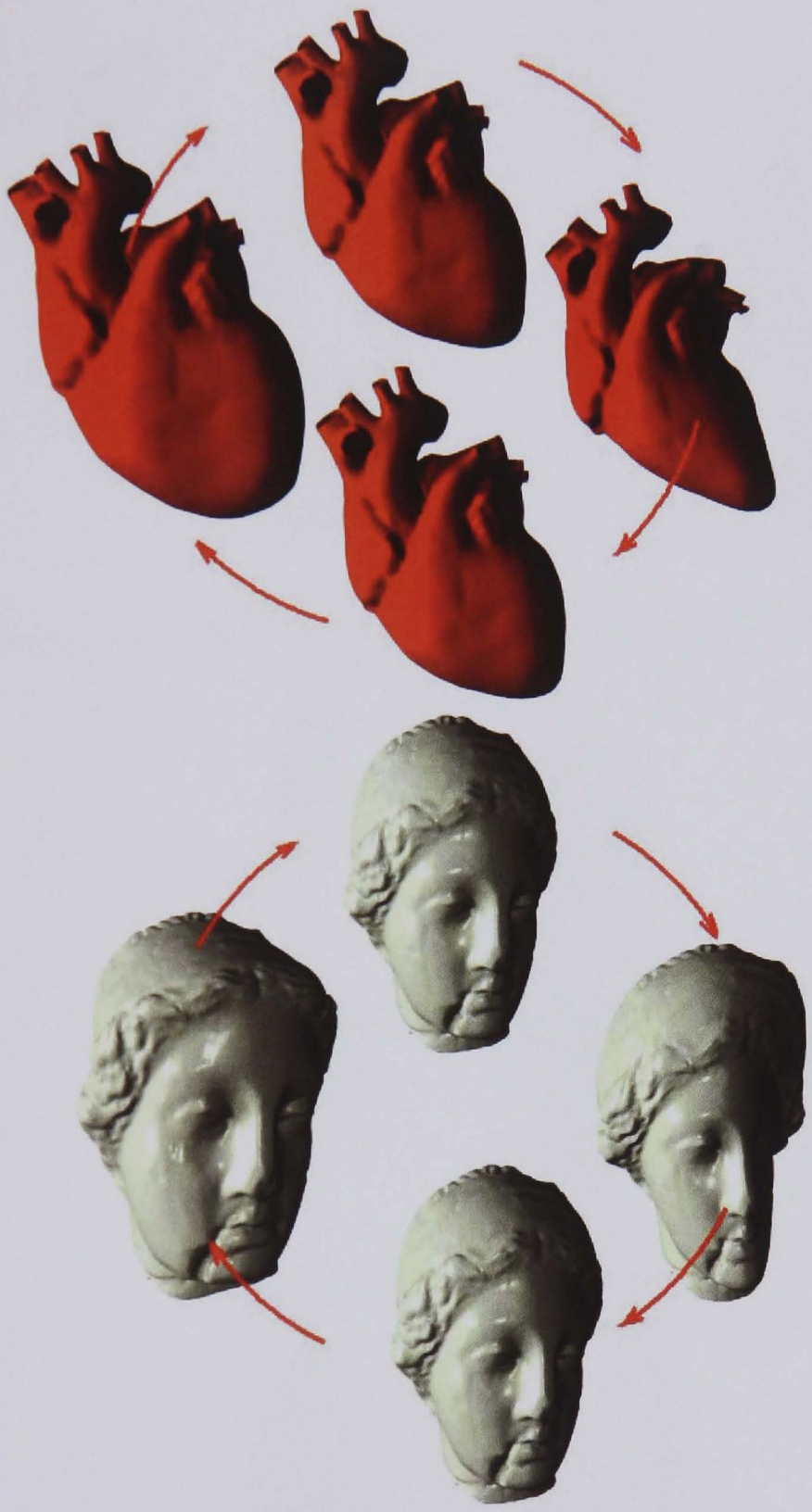
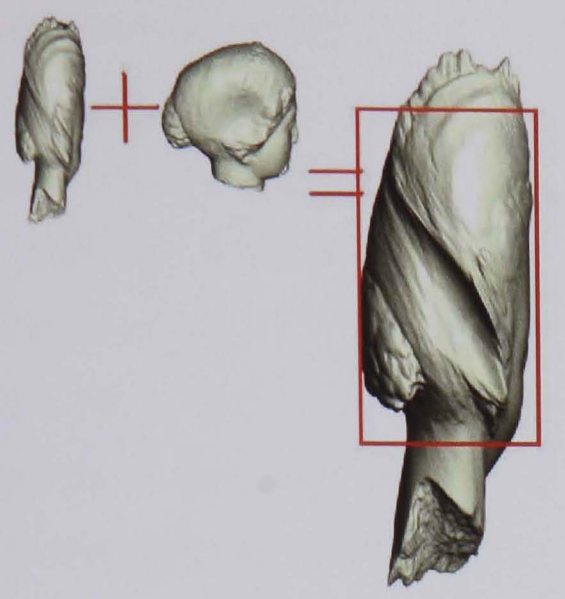
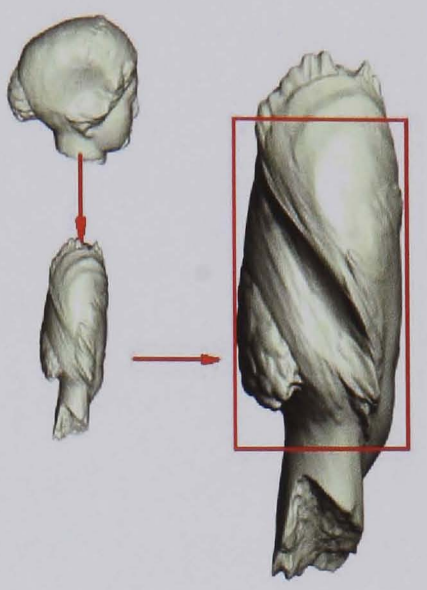


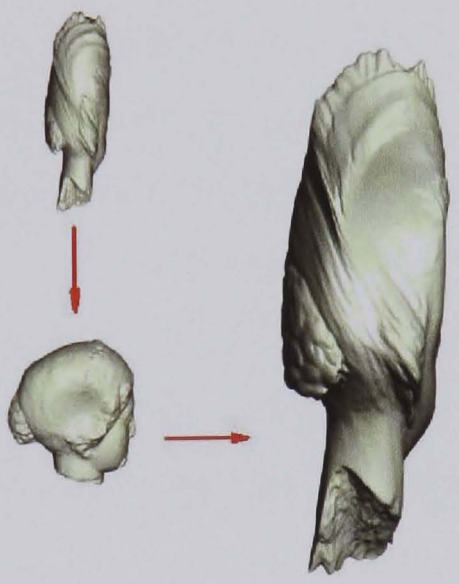
Figure 7.7 Deforming the heart and the Venus head with the same deformation data.



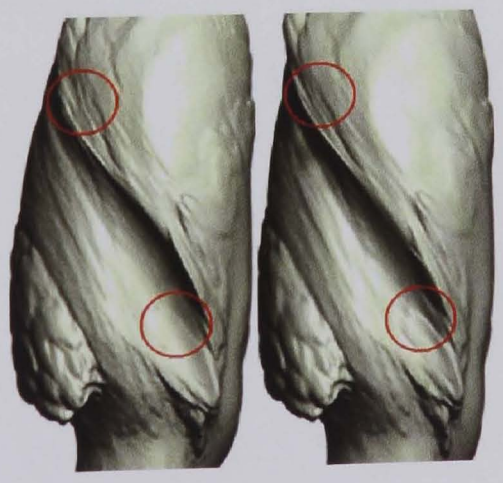
(a) Composition poking and twisting;



(b) Twisting first and then poking;



(c) Poking first and then twisting;



(d) Difference between (a) and (b);

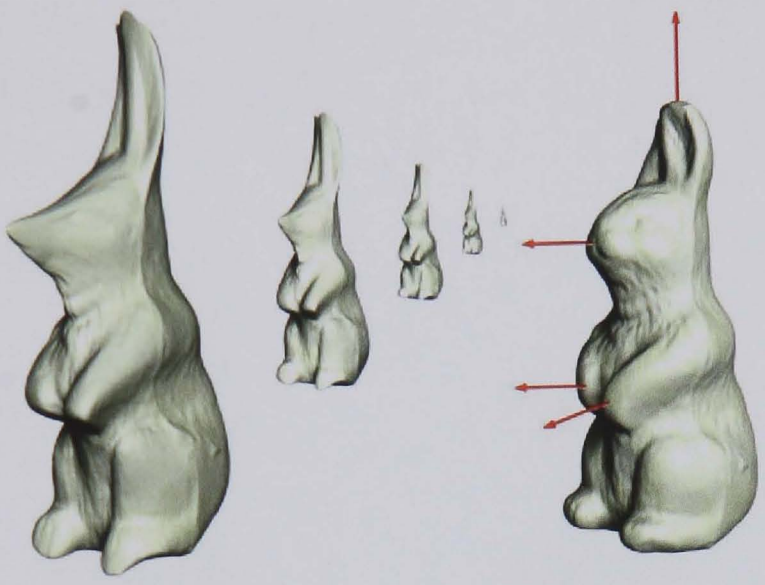
Figure 7.8 Composition of deformations.

Figure 7.8 shows how to produce global and local deformations. In this example, two types of deformation are involved: (1) the Venus head model is being twisted and (2) it is being poked which results in a pit on the head. As mentioned in the above section, there are two methods to combine deformations, aggregate and sequential. With the first method, several deformation fields, which are represented as virtual forces on the VSPs, are composed together by the weighted sum to create an aggregate field. The deformation field is pasted to the object to acquire the composition result. The deformation from this method is depicted in Figure 7.8(a). With the second method, the deformation fields are pasted one after another. The finial results are shown in figures 7.8(b) and 7.8(c).

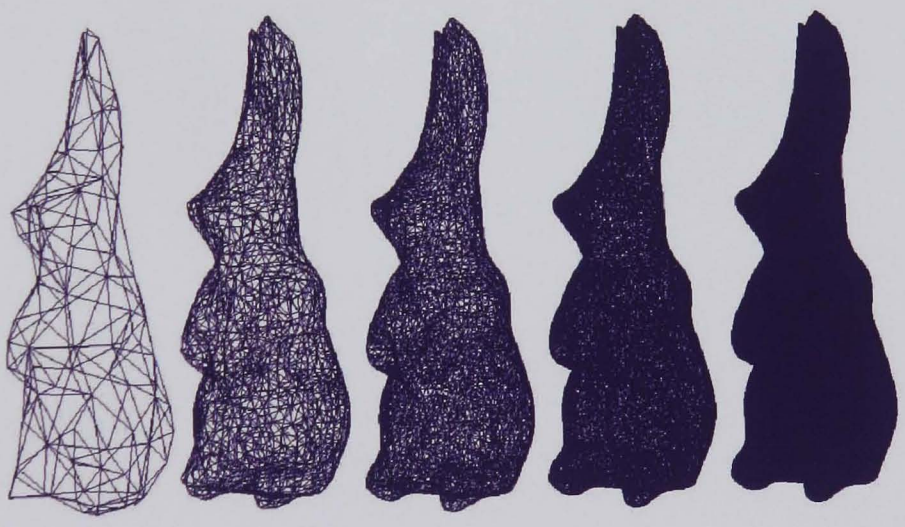
It is obvious that with the second method, the order of the pasting operation has affected the final results. A greater difference can be found between Figure 7.8(b) and 7.8(c). In contrast, using the first method, the deformation can be viewed as a mixture of deformations involved. Therefore, the results from the first method are different from those of the second. Figure 7.8(d) shows the differences between Figure 7.8(a) and 7.8(b), though both look similar. The user can select the appropriate method, according to how the loads are applied.

Pasting deformation fields from simple objects, such as cylinders and ellipsoids, to a more complex object reduces the amount of calculation significantly. On the other hand, however, the accuracy could be compromised, if the object to be deformed is considerably different from the simple object from which the deformation field is copied. To get around this difficulty, one can first create a new object by reducing the complexity using level of detail (LOD) operations. The simplified model is then used to compute a deformation field, which is finally pasted to the original object. As a model simplified using a LOD technique is likely to preserve the features of the object, the resultant deformation field is close to the original object and therefore can achieve very good accuracy. The simplified object has much fewer vertices and smaller polygon count. Comparing with deforming the original complex object directly, it is much cheaper.

As shown in Figure 7.9(a), the rabbit model (Cyberware 2005) is pulled with its base and back fixed. The result on the second coarsest level, level 3, is computed out and then is pasted to all 5 levels to have the results demonstrated in Figure 7.9. Figure 7.9(a) shows the rendered results and 7.9(b) shows the deformed meshes of different levels.



(a) Deformed Rabbits with 5 different levels of details

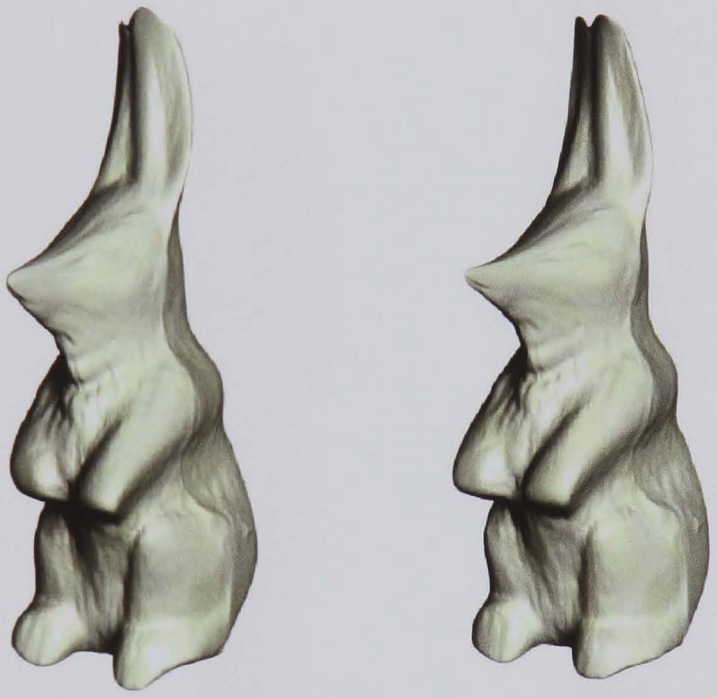


(b)Deformed meshes of the Rabbit models: coarsest mesh at left and finest mesh at right (level 4 to level 0 from left to right).

Figure 7.9 Copy and Paste on LOD models

In Figure 7.10, the deformations are computed on two levels using the mesh-free method: one is the level 0 model with 67038 vertices; the other is the level 3 model with 1730 vertices. The result of pasting the deformation from level 3 to level 0 is shown in Figure 7.10(a). Figure 7.10(b) is computed directly on level 0. Visually,

both results are very similar except a slight difference at the ear tips. The average error between both results is 0.02%. The maximum difference between the corresponding vertices is 0.86%, which is well acceptable for animation purposes. Although the deformations are close, the computation time differs greatly. In practice, the computation time required for level 3 is less than 1% of that required for level 0 in direct computation. And the computation time for pasting is negligible compared to that in the direct computation. The manual preparation for direct computation is about half hour while only a few seconds for the "copy and paste" method.



(a) Deformation from level 3 copied to level 0;

(b) Direct mesh-free computation.

Figure 7.10 Computations on deferent LODs.

Based on (7.3) and (7.4), the error of the deformations between source Ω_1 and target Ω_2 can be quantified with the equation below.

$$erro_u = u_l(Q^{(2)}) - u_l(Q^{(1)}) \approx (\sum_{i=1}^{n}\sum_{k=1}^{3} \alpha_{ik}{}^{(1)} \frac{\partial U_{kl}}{\partial Q})(Q^{(2)} - Q^{(1)}) = K(Q^{(2)} - Q^{(1)}) \qquad (7.11)$$

which suggests that the error depends on the difference in shapes between the target and source. When we reconstruct the deformation from a low level alternative, the similarity of the source and target is justified and our method produces close results

to the direct computation. However, when we use a deformation on a simple geometric primitive, e.g. ellipsoid, such similarity is not always satisfied. It provides a rough approximation but the physical correctness assures the visual quality of the results.

In Figure 7.11, the deformation from the rabbit is copied and pasted to the Isis statue (Cyberware 2005). As expected, the Isis's head is deformed just like the bunny's ears and her hand is dragged forward.

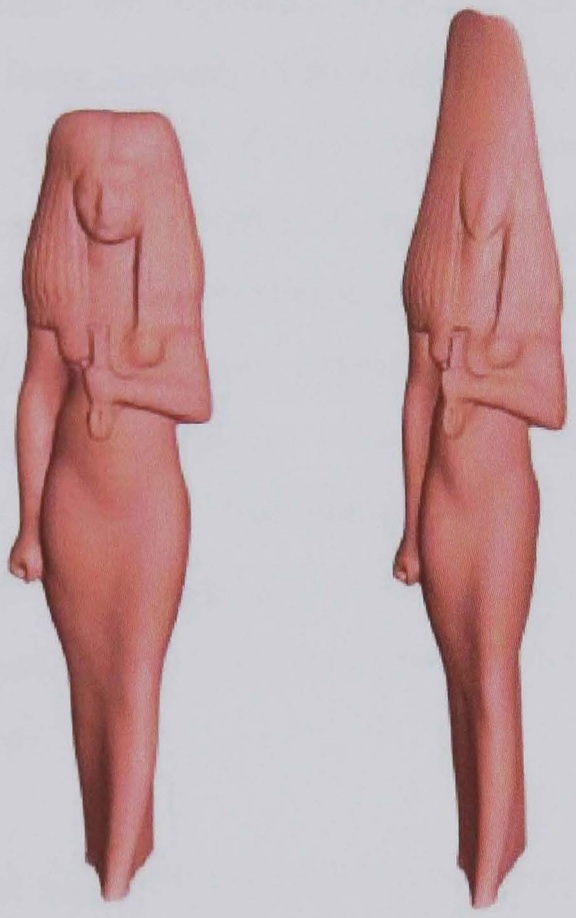


Figure 7.11 Deformation on Rabbit is pasted to the Isis statue.

7.6 Discussion

This technique of reusable deformations is developed based on the simple observation that objects of similar shape under the same force and constraint conditions deform similarly. Although this observation is debatable for engineering applications where physical accuracy is paramount, it has little problem for computer animation where despite the importance of physical accuracy, what is important is its

visual realism. By copying deformation fields from simple objects and pasting to similar but more complex shapes, much less time is required than existing physically-based methods, saving a significant amount of computing time and still achieving visually realistic results. The basic idea of this technique is to record the deformation data in a form of virtual forces acting on the virtual source points. The deformation is then computed as a weighted combination of Kelvin solutions that are mechanically correct. Using the physically correct deformation model ensures good computational accuracy.

In addition to fast computation, the technique supports and encourages reuse of existing deformation data. A large quantity of deformation data have been produced by various physically-based techniques. Current practice makes no use of this resource and every time a new deformation is computed from scratch. This is wasteful. Using the "copy and paste" technique, one can develop an extendable database where deformations from different objects are stored for reuse by other objects. Various techniques and software tools are available for database construction and retrieval. Whenever deformation needs to be calculated, the database is searched first. If no appropriate candidates are found, the object is deformed using a direct method and the resultant deformation data is appended to the database for future applications.

In order to maintain physical plausibility of the deformation during the "copy and paste" process, the source object where the deformation is defined and the target object to which the deformation is "pasted" should both occupy a similar space. The technique does not require vertex correspondence to be established between the source and the target objects. This differs from most morphing methods. Nevertheless, if the source and target objects are different, errors will still arise. Let us consider two possible situations: (1) both source and target objects have similar shape and topology, but are different in orientation and size; and (2) no similar objects can be found in the database.

- Rectifying the first situation is relatively straightforward. Many geometric match techniques are available, which can be used to align and scale both source and target objects. The rotation and scaling parameters are then fed to (7.5) and (7.6) to derive correct deformation.

- To handle the second situation, the complexity of the object is first reduced by undertaking a level of detail operation to produce a simplified object without losing important feature and shape information. Then the user can deform the simplified object using any physical deformation techniques. This usually is significantly quicker than deforming the original object directly. Since there are only minor differences between the original and simplified objects, the deformation data obtained can be easily pasted to the original object. At the same time, the deformation data are stored in the database for future use. Compared with using direct deformation techniques, this technique creates realistic deformation effects with much less modelling effort.

7.7 Conclusions

In this chapter we have developed a technique to reuse known deformations based on the mesh-free technique described in Chapter 5. Several features of the proposed reusable deformations have been demonstrated in this chapter.

The modelling process is substituted by the 'copy' and 'paste' operations which are physically based. A deformation effect can be repeatedly applied to many different objects. The user benefits from a deformation database which stores various deformation effects for reuse. He can choose and combine known deformations before applying them to an object. Our physically based approach ensures visual plausible results. On the other hand, the user can model deformations on a simplified mesh rather than a detailed one to save the work load. Such deformation can be pasted back to the detailed model and the geometric details deforms accordingly. In this way, the physical accuracy of the pasted deformation is assured once the difference of two meshes is relatively small.

Global and local deformations are modelled by compositing existing deformation data. Two methods were presented in the chapter, which are the aggregate deformation and the sequential deformation. The former deforms the whole object in one-go. The latter adds deformations step by step, which permits local deformation to be modelled easily.

Using the simple operations of "copy" and "paste", the animator is freed from cumbersome and tedious manual preparations, as are required by other physically-based deformation techniques. This hopefully, is encouraging for the animators to learn and get used to the seemingly complicated physically-based techniques.

CHAPTER 8

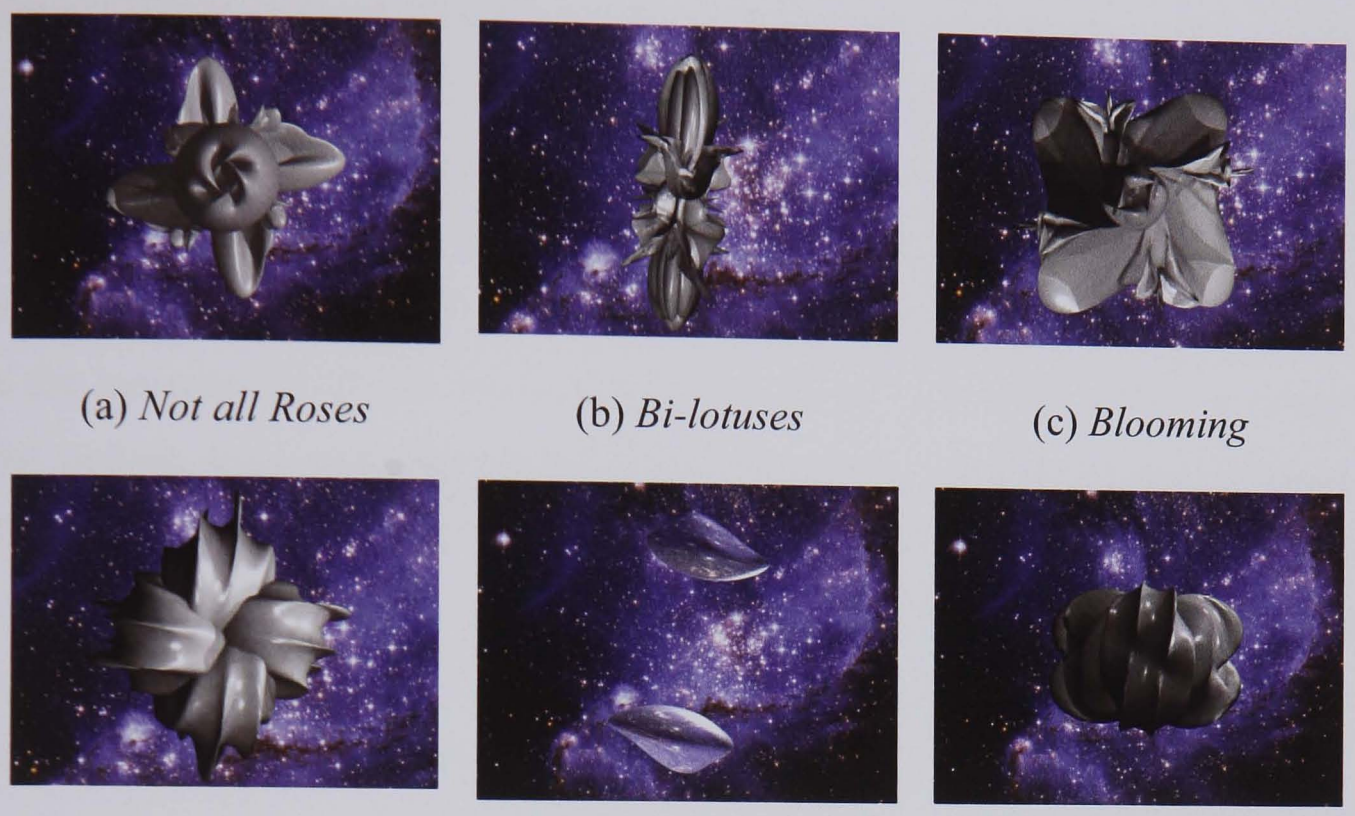
HYPER-TWIST

In this chapter, we link the deformation of an infinite body to the distortion of a space. The objects within the space are transformed accordingly to create various effects. In line with the mesh-free deformation framework, the fundamental solutions describe the deformation of an infinite body, and the discretized points are located on a seed object to capture the initial set up of the deformations. The details are given in this chapter. A hyper-twist is defined as a twist effect in the infinite body. It is used to demonstrate how to create a group of aesthetic shapes from a real twist. It shows that our technique provides a tool that, although easy to use, affords the artist infinite creative possibilities.

We can manipulate distortions of a 3D space to procedurally create aesthetic shapes and animations. It follows the old trail of developing the art from the physical realities (Candy and Edmonds 2002; Mealing 2002) and is different from the popular mathematical art of chaos proposed by Gumowski and Mira (1974).

Some images from Hubble Space Telescope Project (NASA 2005) are used as the background of all the rendered pictures in this chapter. As an art work, each picture is given a name to outline its nature of beauty.

8.1 Distortion of Space



(a) *Not all Roses* (b) *Bi-lotuses* (c) *Blooming*

(d) *Lucky Star* (e) *Tears in Heaven* (f) *Samsara*

Figure 8.1. Some results of the hyper-twist.

Einstein believed that massive objects pull the space around them and the space can be distorted, contracted and dilated. In such a space, an object could be deformed into a new shape, simple or complicated, depending on how the space is distorted. Therefore, in this chapter we create the distortion in space after defining deformations of an infinite elastic body. This idea can be used to develop a modelling and animation technique for the artists to produce various abstract art artefacts. A few examples of this method are listed in Figure 8.1.

All the shapes in Figure 8.1 are the results originated from a same simple twist action. The magic is that the twist has been extended to a hyper-twist of a 3D space. Twist can be associated with a cylinder, a cube, or any other given objects. Under the mesh-free framework, we can separate the deformation of twist from the object and apply it to an infinite volume, the space. The twist is expressed as a sum of the Kelvin's solutions. These solutions distort objects in the space and generate effects listed in Figure 8.1.

Therefore, we can define a hyper-distortion as a distortion of the 3D space. This hyper-distortion is an extension of a known deformation effect of a limited volume to an infinite volume. Different locations in the space have different distortion effects. Embedded different objects into the space, a simple distortion of an infinite space would generate big variations of shapes. The size, the location and the shape are coordinated with the distortion to create various effects. All the distorted shapes are the incarnations of the hyper-distortion at different locations. That is, once an artist defines some distortion of an infinite space, he can harvest his creations of different distorted geometries within the space. In theory, the results that he gains are abundant in one-go because the space is infinite. Furthermore, after numerous shapes are created, it is feasible and very convenient for the artist to store all of them into a database which would provide the raw material in his future creations.

8.2 Deformations of the Infinite Body

Let us address the problem of deforming an infinite elastic body before we move to the example of hyper-twist. Generally, the solution of the infinite elastic body subject to any given distribution of forces can be made by the integration of the solution to the Kelvin problem, the problem of an infinite body subject to a concentrated force.

$$u_k(Q) = \sum\!\!\iiint_{V_m} f_j(P)U_{jk}(P,Q)dv + \sum\!\!\iint_{S_n} p_j(P)U_{jk}(P,Q)ds + \sum_i R_j(P_i)U_{jk}(P_i,Q)$$

$$j,k=1,2,3 \quad (8.1)$$

where $U_{jk}(P,Q)$ is the displacement solution of Kelvin problem given out in (4.8). The left part $u_k(Q)$ in equation (8.1) denotes the displacement of a given point Q in the infinite body. The first integration item on the right is the contribution of all the volume forces, where $f_j(P)$ is the volume force distribution at point P. The second integration item is the contribution of the surface forces where $p_j(P)$ is the surface force distribution at point P. And the third item is the sum-up of the contribution of all the concentrated forces $R_j(P_i)$ at individual points P_i.

In our practice, only the concentrated forces are used to deform the infinite object, so that the difficulty of numerical integration is avoided. This means only the third item on the right of equation (8.1) stands and the first and the second items vanish. Equation (8.1) degenerates to the form of equation (5.4). The concentrated forces are exerted on a group of points which we call virtual source points (VSPs) following the definition in mesh-free deformations. To have various deformations, the user can specify the strength and the direction of each individual force as well as the locations of the forces. The displacement in the infinite body is given out in equation (8.2), which is in line with the formulation (5.4) of mesh-free deformations.

$$u_k(Q) = \sum_{i=1}^{n} \sum_{j=1}^{3} \alpha_{ij} U_{jk}(P_i, Q) \qquad k = 1,2,3 \tag{8.2}$$

where α_{ij} is the value of the force along x_j on the ith VSP P_i and $U_{jk}(P_i,Q)$ is the displacement solution of Kelvin problem.

8.3 Hyper-twist

The artist may find it interesting to explore his creations by manually specifying and editing the forces one by one. Even playing with a small number of forces, e.g. five or ten, allows the freedom to create many variations.

The difficulty emerges when hundreds or thousands of forces exist. It is truly a nightmare for the artist to manually set everything in this case. More forces also mean that it is harder to predict what the result would look like: no human brain is powerful enough to figure that out.

In this section, an example, hyper-twist, is set up, which demonstrates how the deformation of a specific object is extended to that of an infinite body. The user firstly creates a deformation pattern on an object with limited volume, and then the deformation pattern is retargeted to the infinite volume. In this process, the distribution of the forces in the infinite body is adjusted automatically according to the deformation pattern. With a different selected deformation pattern, a different theme of distortion of the infinite body is created. A series of shapes of the same

theme are able to be abstracted from the same distortion. Such predefined deformation pattern is named as seed and the deformation in the infinite body is grown (extended) from the seed. For instance, the twist of a cube is introduced as the seed and then a hyper-twist of the infinite body is generated from it. Furthermore, different seeds lead to different distortion effects. A seed with complicated pattern can lead to very sophisticated distortion of the infinite body, or the space.

In Figure 8.2, the twisted cube is selected as the seed for hyper-twist. The length of the edge of the original cube is four units. The twisted shape is created by the mesh-free tool developed in Chapter 5 by applying coupled forces at its bottom and top.

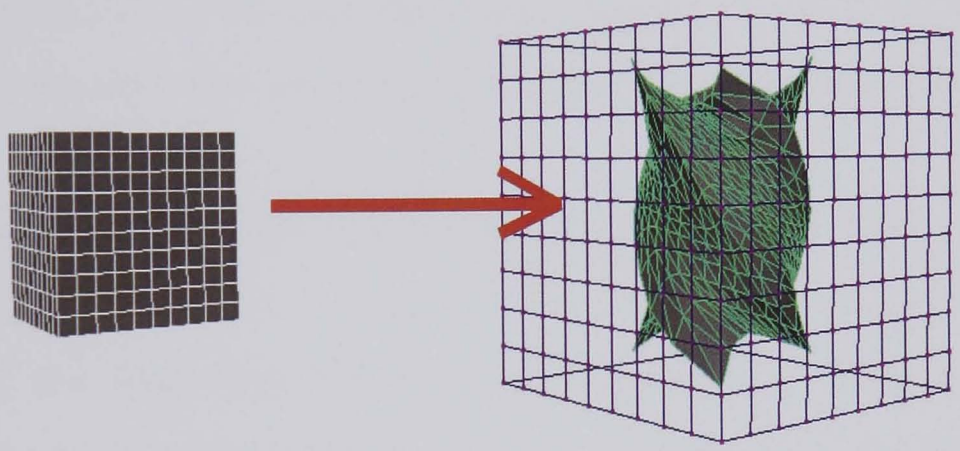


Figure 8.2 The twisted cube serves as the seed (the nodes of surrounding lattice are where the forces are situated).

The seed is the source driving the infinite body to distort as the huge mass distorts the space in Einstein's theory. When the cube is twisting, the whole infinite body is going along with it. Yet no one knows what the whole image of the infinite body is, but it can be assured that the points coincident with the cube vertices would take the same positions of these vertices after twist.

In Figure 8.2, the VSPs where the forces are located are drawn out as the nodes of the lattice surrounding the cube. We set each edge of the lattice to twice the length of the seed cube. This lattice and the seed cube share the same centre, which is set to

the origin in our practices. However, different distributions of VSPs may lead to different results.

Now all VSPs in equation (8.2) are fixed on the lattice. The force strengths, α_{ij}, are unknown but the displacements of the cube vertices are known. If there are m vertices available, there are m equations with unknown α_{ij} according to equation (8.2).

$$u_k(Q_l) = \sum_{i=1}^{n}\sum_{j=1}^{3} \alpha_{ij} U_{jk}(P_i, Q_l) \qquad k = 1,2,3; l = 1,2,..., m \tag{8.3}$$

where n is the number of forces. Equations (8.3) are solved out using the minimal least square method. The movement of any point in the infinite body is computed out using equation (8.2) with the corresponding solution α_{ij} of (8.3). So far the twist of a finite volume, the seed, has been extended to that of an infinite body, the hyper-twist.

In a real twist, the displacement $u_k(Q)$ is meaningful only inside the cube and there is no physical definition of it outside the cube. But after extending the definition domain with the fundamental solutions to the infinite space, the definition of displacement $u_k(Q)$ is mathematically applied to the whole infinite body.

8.4 Results

Deformations of an infinite body have been formulized in the previous sections. When a geometric shape is embedded into the infinite body, every point of the geometry is moved accordingly to form a new shape. This gives a direct impression on how the infinite body is distorted at the specified position of the embedded geometry.

The trajectory of any single point in the space is pinned down via equation (8.2). A continuous surface is discretized into many small triangle facets and each triangle is deformed by displacing its vertices to their new position. The deformed triangle facets are used to render out the distorted surface. The size of the triangle facet is

small enough to keep the object smooth. Incorrect size would introduce unwanted facet aliasing effects after distortion.

Changes of the shape, the size, the orientation of the embedded geometry will influence the results. However, because all the distorted shapes are abstracted out from the hyper-twist, they all inherit the intrinsic property of a twist. This is expressed in their symmetry, stretch, rotation and some kind of inner tension, and enables all the shapes to be labelled with the hyper-twist theme.

In order to show the influence of the size on the distorted shape, a number of cubes with different sizes are chosen to centre at the origin and to share the same orientation of the seed cube. Figure 8.3 lists the results of these cubes.

The orientation or the location of the embedded cube impacts its final distortion, which is proved in Figure 8.4. The size of the cube in Figure 8.4 is 9 units, which is the same as Figure 8.3(e). In Figure 8.4(a) the cube moves up along the twist axis for 1.2 units and in Figure 8.4(b) the cube rotates 22.5 degrees around the twist axis. Some symmetry is missing here due to the arrangement comparing to the Figure 8.3(e).

It is no doubt that the different embedded shape causes a different result. The examples of a sphere and a torus are shown in Figure 8.5, which are much different from the results of cubes in Figure 8.3 and Figure 8.4.

On the other hand, when different seeds are introduced, different themes of distortions are created. For example, Figure 8.6 demonstrates the distortions when the seed is a twist ellipse instead of a cube.

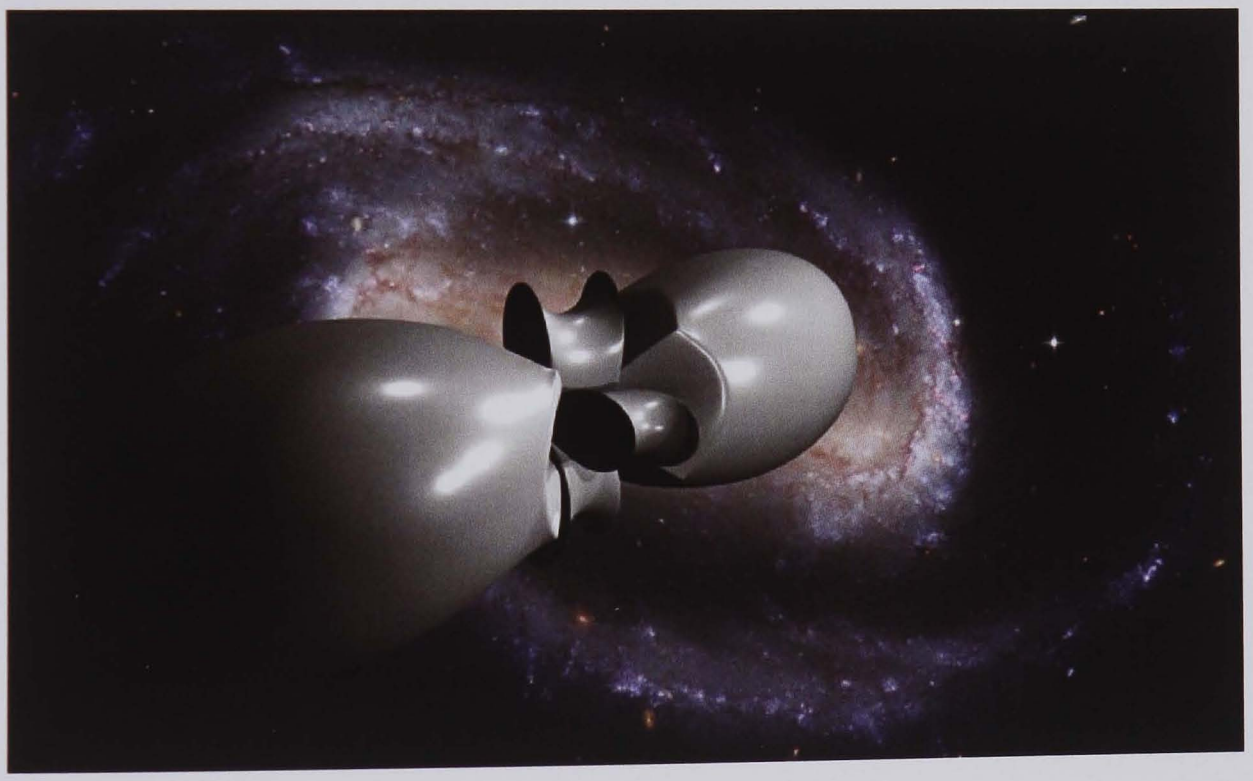
(a) Cube size – 5 units (*Knot 1*)

(b) Cube size – 7 units (*Knot 2*)

(c) Cube size – 7.8 units (*Anger*)

(d) Cube size – 8.4 units (*Fuse*)

(e) Cube size – 9 units (*Bloom in Heaven*)



(f) Cube size – 16 units (*Knot 3*)

Figure 8.3 Distortions of different size cubes.

(a) Cube in Figure 8.3(e) has moved 1.2 units (*Wings of Angel*)

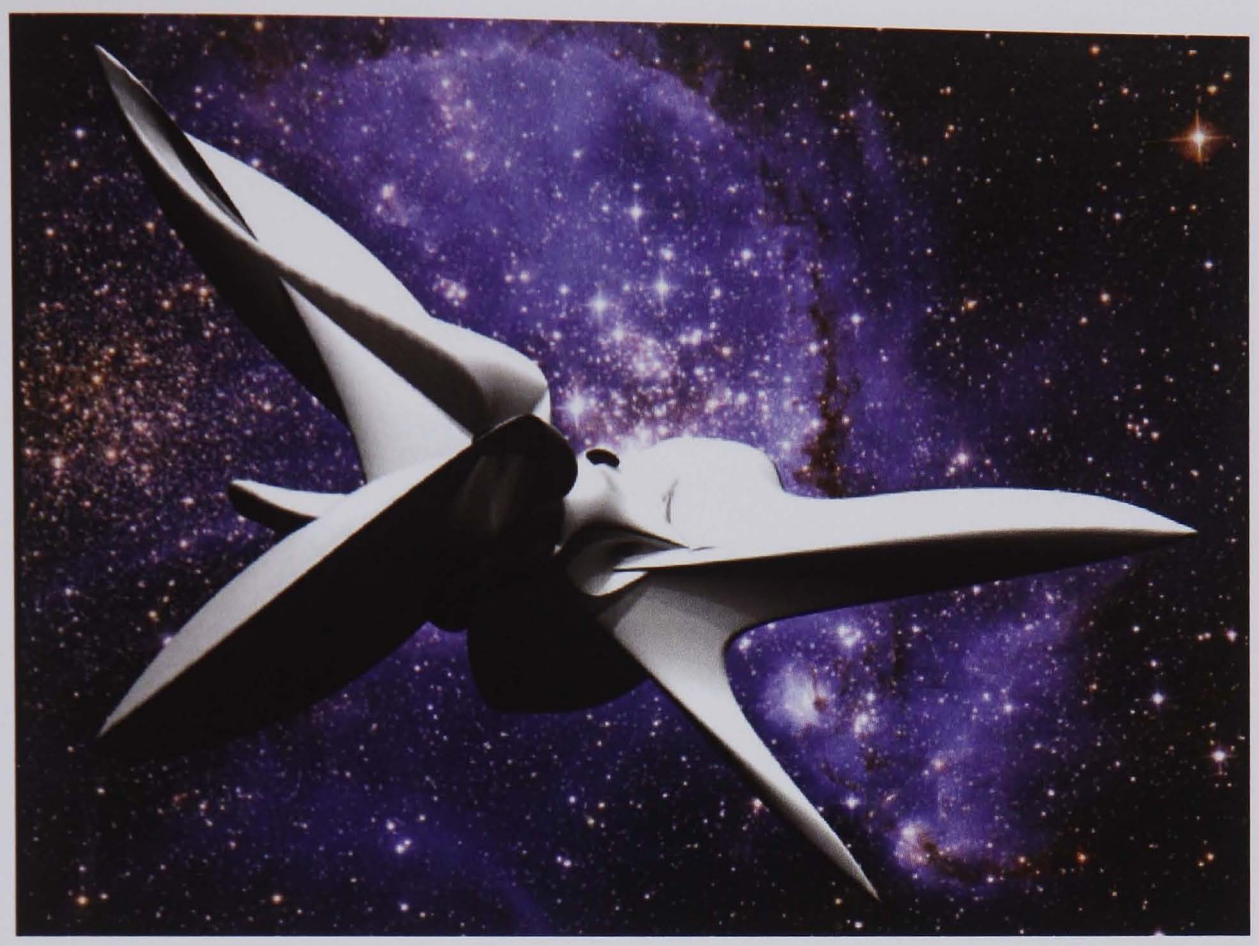
(b)Cube in Figure 8.3(e) has rotated 22.5 degrees (*Hard Dive*)

Figure 8.4 Distortions of cubes with different location or orientation.

(a) Sculpture from a sphere(*Melting*)

(b) Sculpture from a torus(*Grabing*)

Figure 8.5 Distortions of different objects.



(a) *Swan*

(b) *Butterfly*

(c) *Knot 4*

(d) *Knot 5*

Figure 8.6 Distortions of a different seed.

One property of the Kelvin solution is that the displacement of a point caused by a concentrated force increases dramatically when the point gets close to the force

(Figure 4.2). When the size of embedded shape is close to the seed, its distortion is similar to the seed's distortion as revealed in Figure 8.3(a). When the embedded shape's size increases in Figure 8.3(b)-(f), the distortion becomes different from the original twist. Such bias enlarges the twist effect in Figure 8.3(b). In Figure 8.3(c) and (d), the bias is so large that the spokes are dominating the shape rather than twist. This is because the surfaces are very close to the VSPs and the Kelvin solutions have large values. As the box size keeps expending, in Figure 4(e) and (f), the distortion effects become smooth and fade off at the infinite distance.

When the embedded shape is very close to the VSPs, the distortion could be very large and the surface may become faceted. The triangle stretches dramatically with large distortion. Using the small triangle could improve the smoothness but it may fail when the distortion goes extremely large. In Figure 8.4(a), as the cube is moved along the twist axis, it intersects with the force lattice. The bottom of the shape becomes spoky and faceted due to the intersection, while the top is not affected much by the movement which is much similar to the shape of Figure 8.3(e). To avoid the spoke artefact, the mesh should be kept away from the forces.

8.5 Temporal Hyper-twist

Creating a sequence of time-related images using the above developed technique is straightforward. One can think of many possibilities. In this chapter, we demonstrate two ways. The first is to change the strengths of the forces temporarily, but the pattern of force distribution is kept untouched. When gradually increasing the strength of the sources from zero to a certain value, the whole space is continually distorted to its final state. The second is to change the distribution of forces in the space. In practice, we start by spreading the forces over the surface of an object, a cube for example. We then transform the cube with time, which in turn changes the relative distribution of the forces.

A few sequences of animation were produced. Figure 8.7 shows some snapshots of two animation sequences when we increase the force strengths. By changing the

distribution of the forces with time, another sequence of animation is created shown in Figure 8.8.

8.6 Conclusions

The work in this chapter is physically-oriented and also falls into the mesh-free deformation framework. It reconstructs the deformations of an infinite body which are linked to distortions of a 3D space. It serves as a tool for artists as well as animators to create aesthetic effects which could not be generated by any other means.

This tool allows the artist to automatically explore the dramatically shaped objects that may stretch, twist and distort in a space. There are two ways to define the distortions of the infinite body. One is to directly apply the forces in the infinite body and the results are calculated straightforwardly with equation (8.2). The other is to deform a finite volume as a seed which drives the infinite body to distort. In the second way, the forces are automatically assigned to fit the seed distortion. Once the artist seeds a deformed shape, he can create a bunch of distorted shapes in one-go.

Linking necessary parameters, such as the strengths and distribution of the forces, with time, one can easily produce temporal sequences of distorted images. A few animation sequences were presented and they evolved within the time dimension. These sequences revealed the relation of time and space as well as the evolution process.

The singularity of the Kelvin solution should be noticed and it may cause problems. The spoke artefact does appear when the deforming shape is too close to the forces, and will be the subject of further investigation in the future.

Figure 8.7 Animation by varying the force strengths.

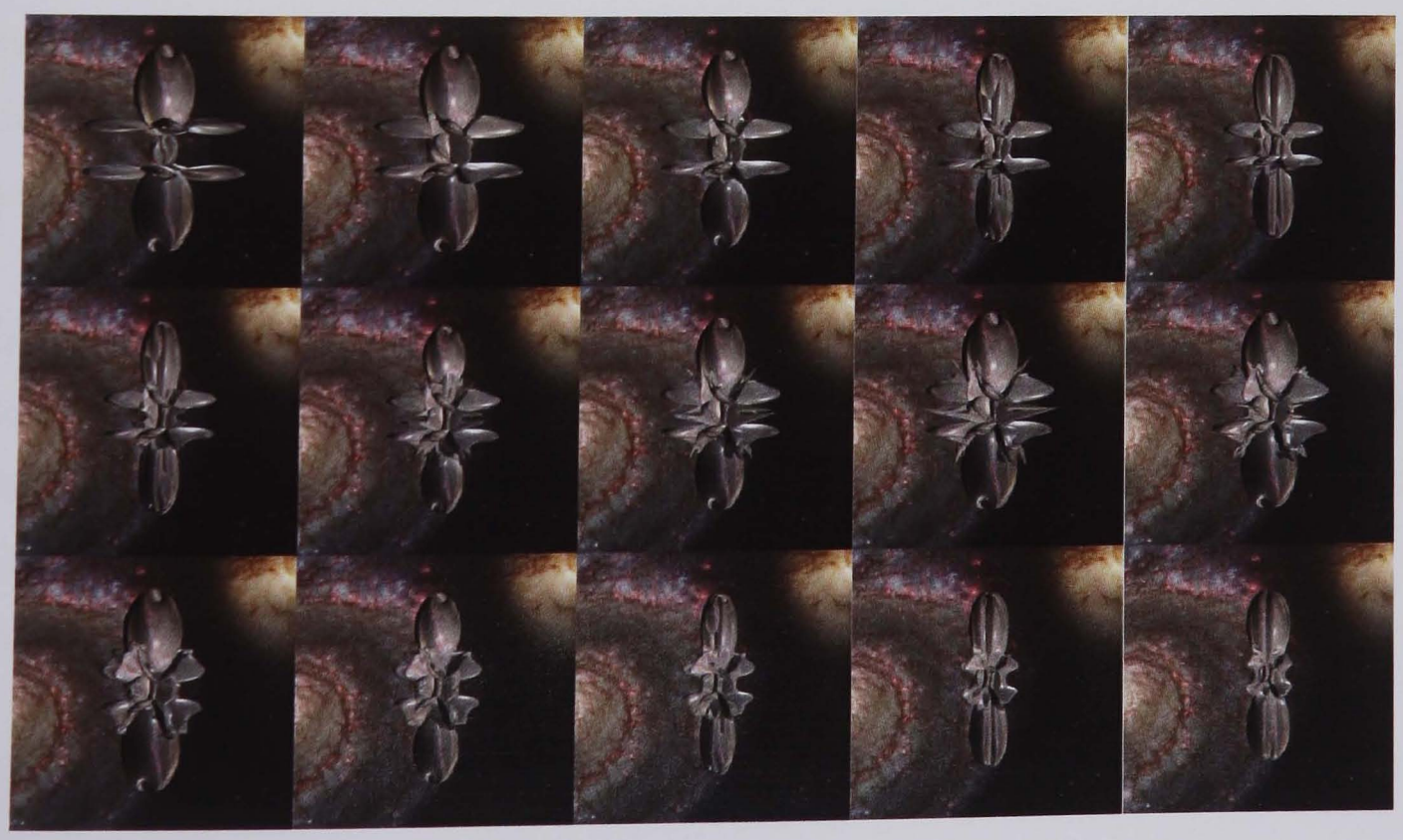


Figure 8.8 Animation by varying the distribution of the forces.

CHAPTER 9

FORCE MAPPING

This chapter extends the mesh-free framework for geometric representation and compression. A technique called force mapping is developed. In the previous chapters, we introduced virtual forces which deform object shapes. The idea in this chapter is similar in that some forces are applied on a surface to 'deform' it and create the fine details. A force map records the applied forces and their distributions. By applying these forces on a relatively simple surface and deforming it, a detailed surface is created. This is just like the displacement mapping technique, which use displacements instead of forces to store the details. Our technique can be used for surface modelling, reconstruction and editing.

In the technique of force mapping, the displacements of a surface are assembled as sums of kernel functions. The kernel function plays the similar role as the fundamental solution does in the mesh-free deformations. It defines the disturbance of a single force on the surface while the latter is the disturbance of a force in a volume.

9.1 Displacement Mapping

Many real and animated objects are represented with surfaces of a sufficient level of detail. Such surfaces can be obtained using various techniques, such as 3D scanning. Since they are typically stored as dense triangle meshes, how to manipulate, animate, render, transmit, and store these meshes is a real challenge in today's computer graphics (Lee et al. 2002).

In order to tackle this problem, the displacement mapping technique is developed as an efficient means for the representation of detailed surfaces. The information of the mesh is decomposed into two parts: a simpler mesh describing the basic shape of the model which contains much fewer vertices than the original dense meshes, and a displacement map to store the difference between the original mesh and the simpler mesh (Cook 1984; Gumhold and Hüttner 1999; Lee et al. 2002; Wang et al. 2003).

Early in 1978, Blinn (1978) introduced the idea of bump mapping which perturbs the surface normal to create and match the geometric subtle details on the surface. Without actually modelling the detailed surface variation, images with realistic looking surface 'bumps' can be produced (Blinn 1978; Kim et al. 2001; Tarini et al. 2000). Tarini et al.(2000) extended the work by introducing the normal-map to achieve excellent performance in rendering time and rendering quality.

Cook (1984; 1987) displaced a surface by a function. The original geometry is presented by a large number of polygons. In Cook's method, such geometry is reconstructed from a coarse model with fewer polygons containing low level of details. The coarse model is subdivided and the vertices after subdivision are displaced in the normal direction to add subtle details. Kugler (1998) developed a method of rendering both bump- and reflection-mapped surfaces. He pre-computed shading- and reflection-maps with coordinate generation tables, which cache pixel-to-pixel normal vectors. Recently, Displacement maps were widely used as procedural displacement shaders in RenderMan and the close relationship between displacement mapping and bump mapping was discussed in (Apodaca and Gritz

1999). Kim et al. (2001) directly calculates the inner product of the perturbed normal vector and the halfway vector. The deflection function is defined by the perturbation angles represented in a spherical polar coordinate system to reduce its computation. The perturbed term is used as an index to the shading look-up tables which enable real-time per-pixel lighting on the bumped surface to be rendered. Kautz and Seidel (2001) also provided a solution with hardware acceleration.

9.2 Force Mapping

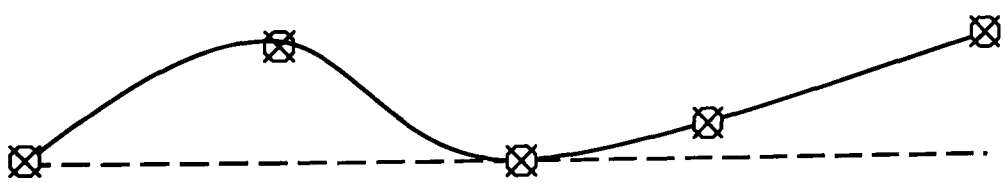
Forces, as an alternative to displacements, again can alter a surface. Replacing the displacements with forces, the force mapping implicitly defines the geometric subtle details of a surface by distributing the virtual forces.

As an alternative of displacement mapping, it inherits the same benefits of displacement mapping. The techniques and algorithms developed for displacement mapping can be almost directly applied, such as the displaced subdivision surface technique proposed by (Lee et al. 2002). Force mapping is effective in modelling and editing complex surfaces. Instead of directly moving the mesh vertices, force mapping simply modifies the number, position and magnitude of the forces to move a group of vertices to the right positions. In doing so, much higher control can be achieved. Similar to displacement mapping, force mapping is able to compress the initial dense mesh. It is easy to achieve different levels of detail (LOD). Models of different LODs can be generated with little extra effort. An obvious advantage of the force mapping technique is that it integrates the physical property, i.e. the force, into geometry by introducing the kernel function and the force map. The simple way to estimate the reaction forces and the displacements under an external force in real time is useful for virtual reality applications. In fact on the same ground, other physical properties, such as temperature and electromagnetic variables, can be similarly integrated into the same geometric meshes and achieve various physically generated effects.

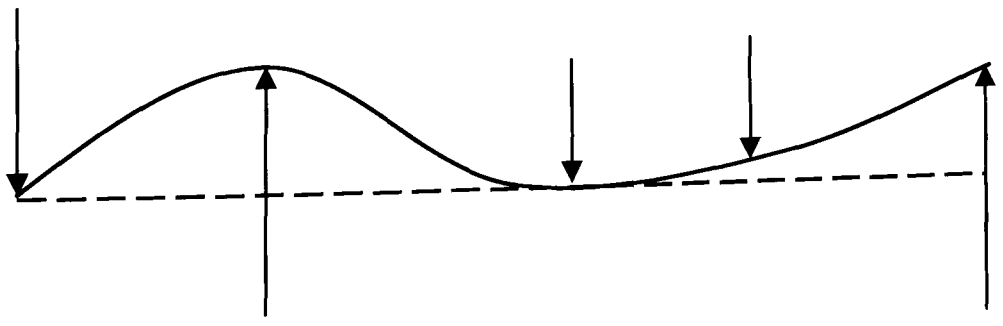
9.2.1 Force Mapping Definition

By applying some external sources of disturbance, a surface containing subtle detail information can be regenerated by deforming a simpler surface that has much fewer vertices. These external disturbances can be displacement constraints in the case of displacement mapping, or forces in the case of force mapping which is described below. In displacement mapping, the subtle information is transformed into displacements. In force mapping, the detail surface information is represented with virtual forces.

In Figure 9.1, a flexible beam is presented as a solid line and its original shape as a dash-line. As shown in Figure 9.1(a), an originally straight beam is changed into a curved one by moving some given points to right positions (displacement constraints). It can also be deformed in the same way by exerting appropriate forces (force constraints) as illustrated in Figure 9.1(b).



(a) Beam is deformed by giving some displacements



(b) Beam is deformed by exerting some forces

Figure 9.1 A straight beam is deformed.

For a two-dimensional problem like plates or shells, the determination of the displacements caused by the forces will require the solution to a 4^{th} order partial differential equation (PDE) that governs the deformation of the object in question (Saada 1993). Except for some simple cases where a closed form solution is obtainable, the resolution of a 4^{th} order PDE is computationally expensive. In order to address this issue, the effects of forces are approximated by directly defining the displacement caused by a single unit force. The resultant displacement, $\Phi^I(x)$, is called the kernel function, whose definition is given in Section 9.2.2.

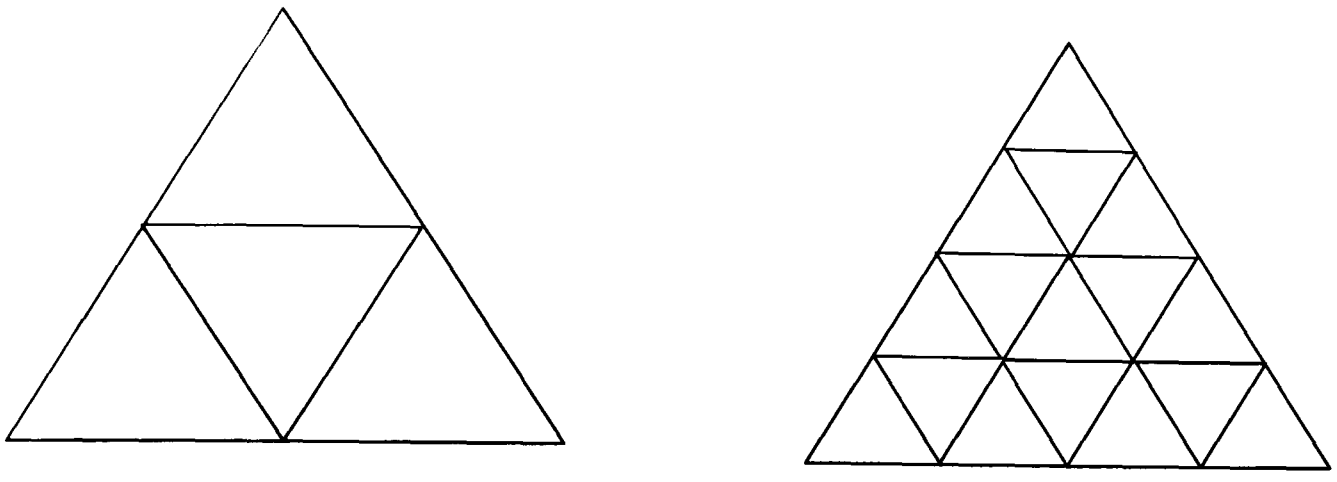
With this kernel function, the overall displacement distribution can be written as:

$$d(x) = \sum_I f^I \Phi^I(x) \tag{9.1}$$

where $d(x)$ is the difference between the original mesh and the simplified mesh at point x, f^I is the strength of the I^{th} force and $\Phi^I(x)$ is the kernel function mentioned above.

For a small deformation, the difference between the simplified mesh and the original mesh is relatively small. Due to this reason, the virtual force is given only in the direction normal to the simplified surface. Therefore, a scalar instead of a vector is needed to store the force at a given point which is similar to displacement mapping.

The first step for force mapping is to subdivide the triangles of the simplified mesh. The 0^{th} level subdivision is the same as the simplest surface. Assuming the k^{th} subdivision generates the triangle meshes illustrated in Figure 9.2(a), the $(k+1)^{th}$ subdivision whose shape is the same as k^{th} level will produce more triangle meshes as demonstrated in Figure 9.2(b). The level of the subdivision is chosen according to the frequency of sampling. A higher level of subdivision is able to reveal more subtle details but requires more memory and computation, while a lower level of subdivision leads to less memory and computation but discards the high frequency information of the subtle details. Then, the forces are exerted on the vertices of the triangles.



(a) the k^{th} level subdivision (b) the $(k+1)^{th}$ level subdivision

Figure 9.2 Subdivision of the surface

For force mapping, the forces in equation (9.1) are unknown constants. In order to determine the value of these forces exerting at the control points, a displacement map containing the information of $d(x)$ is extracted first. Figure 9.3 shows how $d(x)$ is defined. Then according to expression (9.1), a set of linear equations with f as unknowns are deduced:

$$[\Phi]^T \{f\} = \{d\} \tag{9.2}$$

where element Φ^{IJ} in matrix $[\Phi]$ is the displacement at point J caused by a unit force at point I, the I^{th} element f^I in vector $\{f\}$ is the strength of the force at point I, and the J^{th} element d^J in vector $\{d\}$ is the displacement at point J.

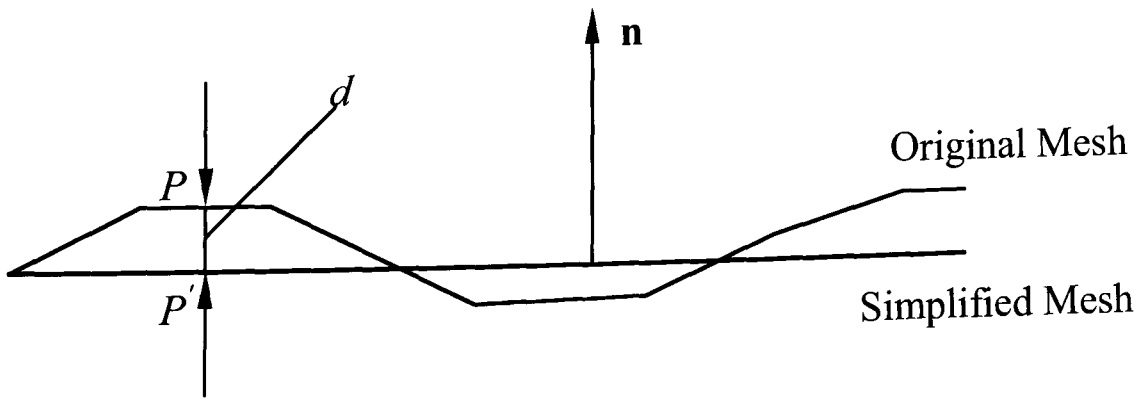


Figure 9.3 Extraction the displacement map.

Using the matrix multiplication, equation (9.2) can be transformed as follows to explicitly give out the forces.

$$\{f\} = ([\Phi][\Phi]^T)^{-1}[\Phi]\{d\} \tag{9.3}$$

Equation (9.2) is solved by the least squares method to obtain the required forces. In the experiment of this chapter, the displacement is sampled at the vertices of the subdivision mesh which is two orders higher than the subdivision used by force mapping. Other choices are available. One natural example is that the sampling points can be chosen as the projection on the simplified mesh of the vertices from the original mesh.

In section 9.2.2, the kernel function is selected with a compact support leading to a sparse matrix [Φ]. The least squares method optimizes the approximation globally.

9.2.2 Definition of the Kernel Function

In the equation (9.1), the kernel function $\Phi^I(x)$ is chosen as the displacement caused by a unit force acting at the I^{th} point. As explained above, only the displacement and the force along the normal direction are taken into consideration due to the fact that the difference between original mesh and simplified mesh is relatively small.

The kernel function $\Phi^I(x)$ can be determined by using the theories of plates and shells. However, it is complicated and not practical here to compute the displacement of the surface using such exact methods because the numerical solution of the equations is difficult and time-consuming. In addition, different surfaces require different kernel functions to be repopulated. Therefore, we seek a function which combines simplicity, generality and the deformation mechanism together. To this end, the Kelvin solution seems to be a good candidate which gives an analytical treatment of a unit concentrated force exerting in an infinite elastic body. Let P be the I^{th} point where the force acts and Q be the J^{th} point where the displacement occurs, and the kernel function Φ^{IJ} is written as

$$\Phi^{IJ} = \{n^I\}^T [U] \{n^J\} \tag{9.4}$$

where $\{n^I\}$ is the normal at point I, $\{n^J\}$ is the normal at point J and (U) is 3×3 matrix with element $U_{ij}(P,Q) = \frac{1}{r_{PQ}} W_{ij}(r_{PQ})$ defined in equation (4.8),.

By embedding the simplified mesh into an infinite elastic body and applying numerous concentrated forces on the infinite elastic body, the elastic body will be deformed. Accordingly, the simplified mesh will be changed into a new shape similar to the original mesh.

Poisson's ratio ν in equation (4.8) is set to zero, which means that the elongation along one direction will not cause the shrinking of the lateral section. The shear modulus G which defines the stiffness of material in the equations is set to one unit.

With the Kelvin solution, the kernel function is defined. Since the displacements given by the Kelvin solution are infinite at the positions where the forces exert, the Kelvin solution is singular at these points. In fact, the displacements at these positions should be bounded. Therefore, the infinite displacements are unacceptable. The numerical calculation will fail at these positions. Another shortcoming is that the Kelvin solution is global and any local perturbation will change the shape on the whole surface, which leads to a full coefficient matrix [Φ] in equation (9.2) and makes the extraction of a force map rather expensive. Due to these reasons, a modified Kelvin solution $U_{ij}^*(P,Q)$ is proposed which eliminates the singularity and has a compact support:

$$U_{ij}^*(P,Q) = \begin{cases} \rho(r_{PQ}) W_{ij}(r_{PQ}) & r_{PQ} \le h \\ 0 & r_{PQ} > h \end{cases} \tag{9.5}$$

where $\rho(r) = \frac{100}{h}(1 - \sqrt{\frac{r}{h}})^2$, the definition of r_{PQ} and $W_{ij}(r_{PQ})$ is the same as that in equation (4.8), h is the value related to the force sampling frequency, which controls the influence range of a single force. If the force is defined on the k^{th} subdivision

mesh and s is the size of the triangle in the simplified mesh where the force falls in, $h=3s/k$.

The Kelvin solution (4.8) and the modified Kelvin solution (9.5) share the same $W_{ij}(r_{PQ})$ and the difference of these two solutions can be revealed by comparing functions $1/r$ with $\rho(r)$ (Figure 9.4).

From this figure, it can be observed that around the position where a force is applied, $1/r$ in the Kelvin solution is infinite and $\rho(r)$ in the modified solution is about 33. It indicates that the infinite displacement at the position has been replaced by a finite one. When r is greater then 2.5, the two solutions become almost the same. The maximum difference occurs at about r=0.5.

With the above treatment, the kernel function becomes:

$$\Phi^{IJ} = \{n^I\}^T [U^*] \{n^J\} \tag{9.6}$$

where the element of $[U^*]$ is $U_{ij}^*(P,Q)$ in (9.5), $\{n^I\}$ is the normal at point I, and $\{n^J\}$ is the normal at point J.

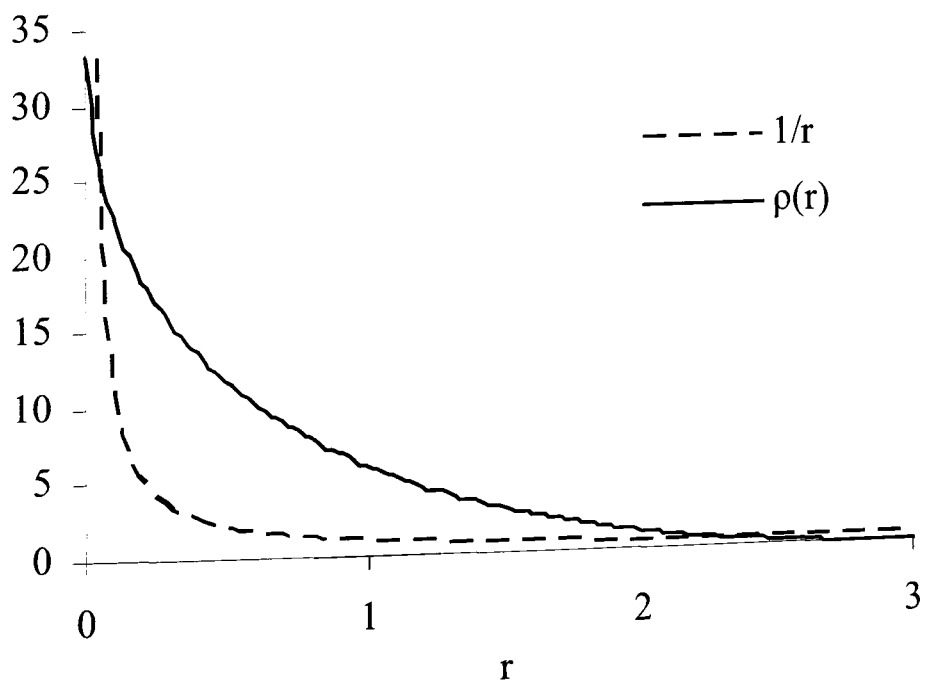


Figure 9.4 Comparing functions $1/r$ and $\rho(r)$ when h=3.

9.2.3 Conversion Algorithm

In order to decompose an arbitrary triangle mesh into a force map and a simplified surface, the following steps should be followed:

1. Simplify the original mesh and obtain an initial simplified surface which preserves the basic shape of the model. The simplification can be achieved with various techniques, such as edge collapsing.
2. Globally optimize the initial simplified mesh vertices so that the simplified surface fits the original mesh well.
3. Sample the difference between the original surface and the simplified surface at the $k+1^{th}$ subdivision level. Construct the linear equations where the unknowns are the forces at the vertices of the k^{th} subdivision mask.
4. Solve the linear systems with the standard least squares method and create the force map.

9.2.4 Rendering Algorithm

Having discussed the determination of the force map for a given surface, the consequent work is to reconstruct a series of surfaces from the simplified surface according to different sampling masks to be used. These surfaces give different levels of detail. Figure 9.2 illustrates how to select sampling points. For sample point J on the newly constructed surface, its location x^J is:

$$x^J = x_0{}^J + d^J n^J = x_0{}^J + n^J \sum_I \Phi^{IJ} f^I \tag{9.7}$$

where $x_0{}^J$ is the location of the sample point J on the simplified surface, n^J is the unit normal vector of the simplified surface at point J, Φ^{IJ} is the kernel function, and f^I is the strength of the I^{th} force at point I.

9.3 Results

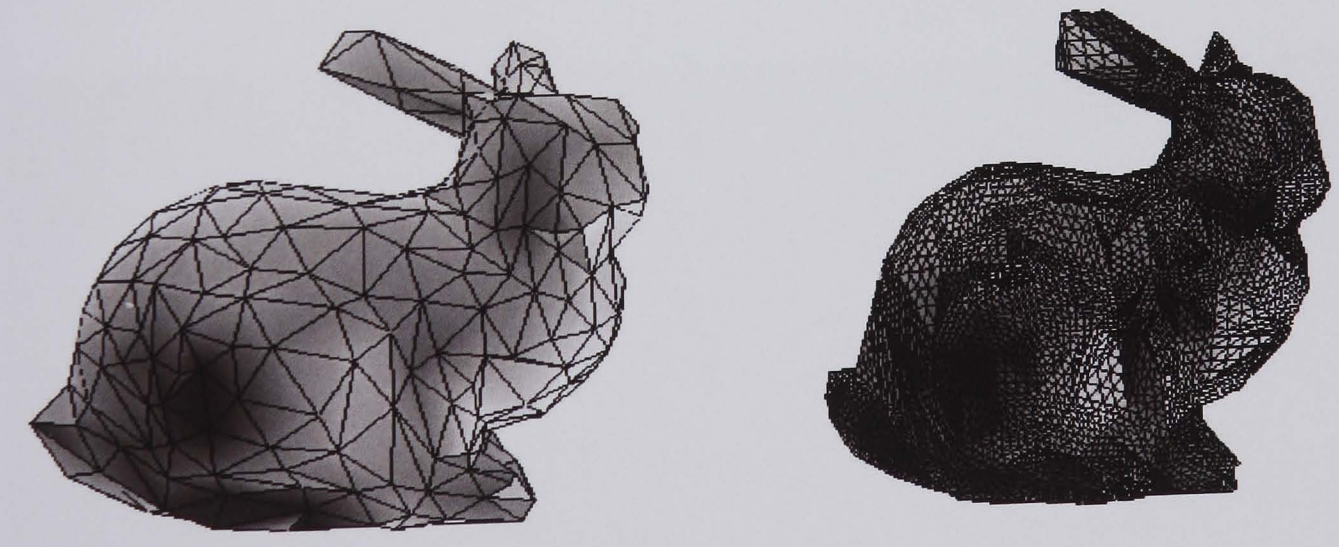
In the following, the well known Stanford Bunny model (Stanford 2004) is used to illustrate the force mapping method. The statistics of the used meshes is illustrated in Table 9.1. The meshes of different LODs are generated by the subdivision method in section 9.2.1. Only the 0^{th} level mesh (Figure 9.5(a)) needs to be stored in practice.

Table 9.1 Statistics of the models

	Vertices No.	Face No.	File Size as PLY Format
Original Model	34834	69451	3374KB
The 0^{th} Level Mesh	241	578	14KB
The 1^{st} Level Mesh	940	1850	75KB
The 2^{nd} Level Mesh	3731	7400	232KB
The 3^{rd} Level Mesh	14863	29600	970KB

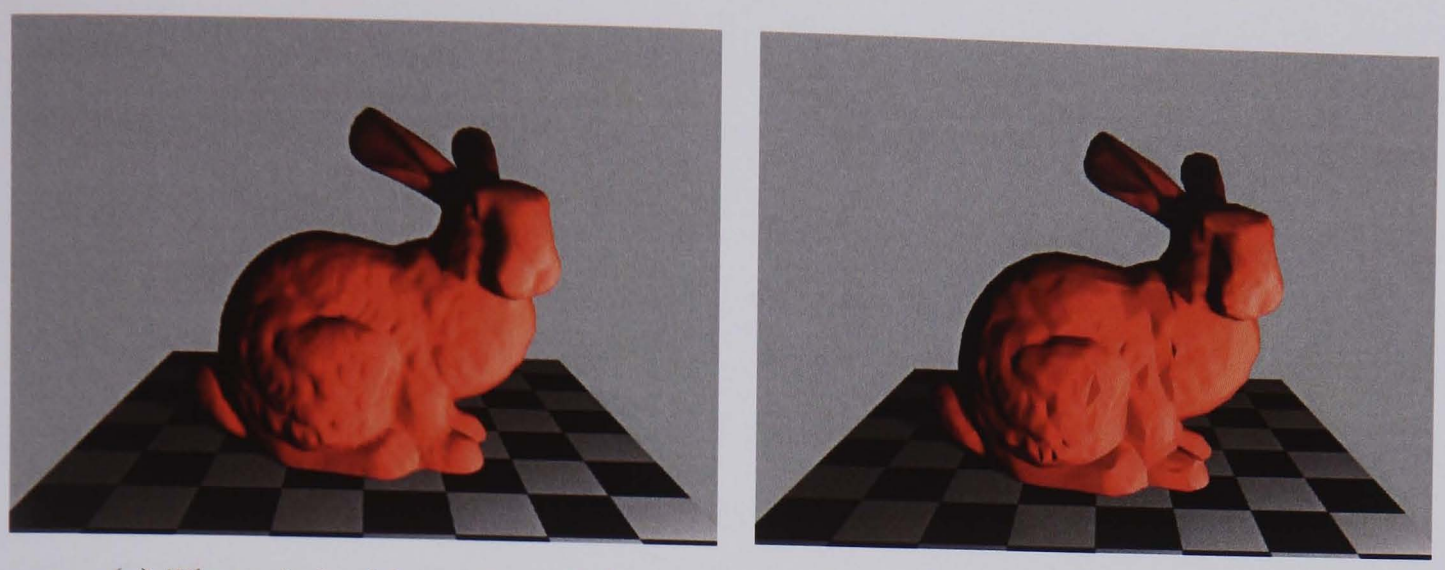
The displacement map is extracted from the mesh of the 3rd level. The distance from the vertex of the 3rd level mesh in its normal direction to the original mesh is stored. The original mesh and the 3rd level mesh reconstructed with the displacement map techniques are shown in Figure 9.6. The displacement map is stored in ASCII format. The size is 153KB.

The force map is defined on the mesh of the 1st level, where the displacement map extracted above is used as the input textured height field. With the method described in section 9.2, where the storage of the force map required is only 9KB in ASCII format, the meshes in different levels with the detailed information are reconstructed.



(a) The mesh of the 0^{th} level (b) The mesh of the 3rd level.

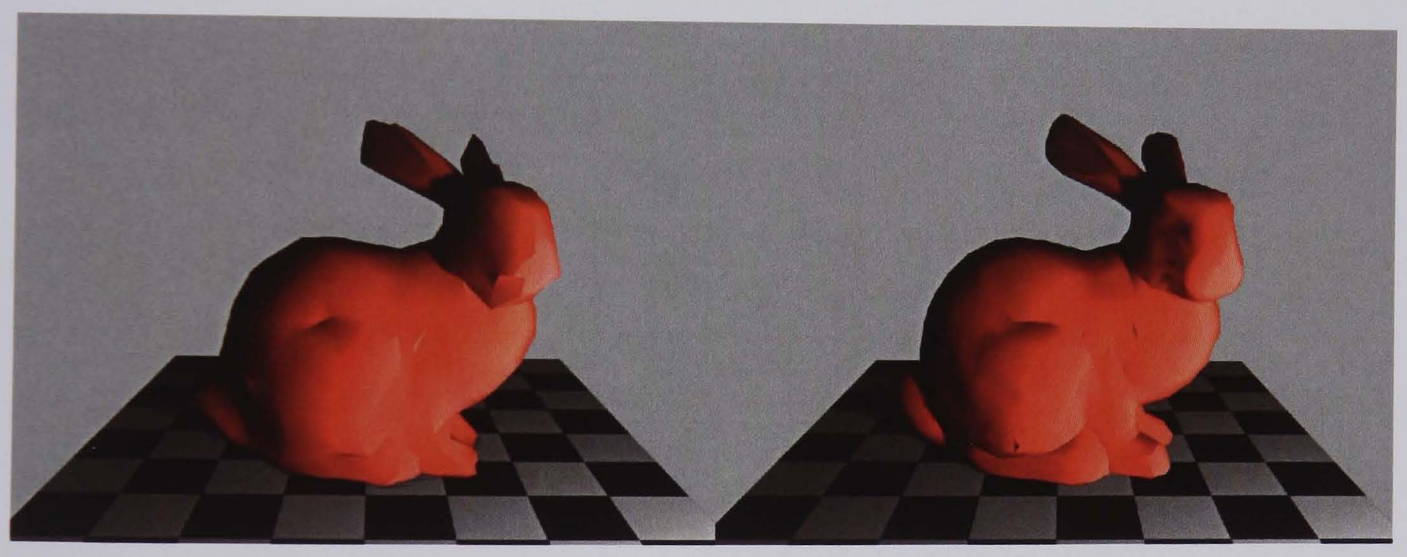
Figure 9.5 Different levels of meshes of Bunny without details.



(a) The original bunny model

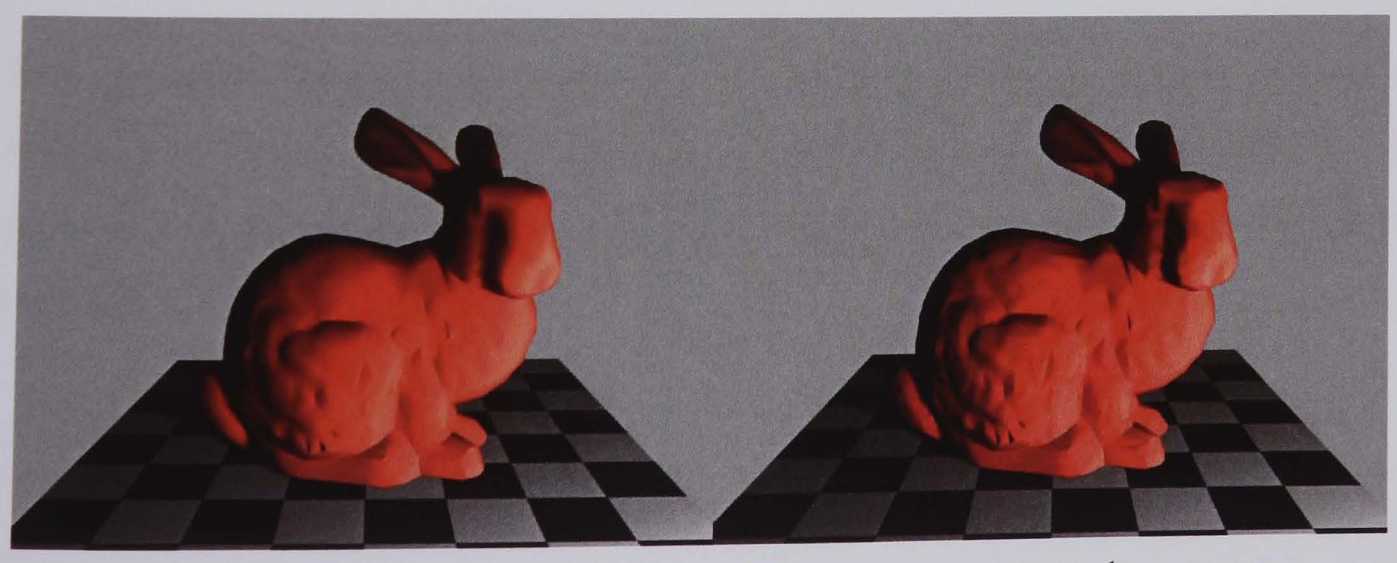
(b) The 3rd level mesh reconstructed with displacement mapping

Figure 9.6 Example from displacement mapping.



(a) The 0th level

(b) The 1st level



(c) The 2nd level

(d) The 3rd level

Figure 9.7 The meshes of different LODs are created with the force mapping.

Although the scene of the 3rd level mesh created by the force map (Figure 9.7(d)) is similar to that by the displacement mapping (Figure 9.6(b)), the storage of the force map here is only one sixteenth of the displacement map. The difference of these two images is difficult to tell.

The reconstruction process is efficient, because the modified Kelvin solution has a tight support. It took no more than 0.2 second to reconstruct the 3rd level detailed mesh in our Bunny example with a Dell Pentium III 500 MHz PC.

9.4 Conclusions

As an application of the mesh-free deformation framework, this chapter has presented a technique, called force mapping, for the representation, editing and compression of complex surface models. The force mapping technique uses the recorded forces to shape the fine details of the surface.

By modifying the Kelvin solution, a kernel function was defined and the relationship between the applied forces and the resultant displacements was obtained. A simplified surface was subdivided into a number of triangles and the forces were applied on them to generate a surface of higher complexity. The process was repeated to create a series of surfaces with different levels of detail.

The geometric compression demonstrated here is only one application of the force mapping technique. It can also be used in complex surface editing by definition of a proper force map in a way similar to the bump mapping and displacement mapping techniques.

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

10.1 Conclusions

Physically-based deformation techniques have attracted a great deal of attention from the computer graphics research community. Existing techniques require the deformed object to be properly meshed, which often represents a time consuming process.

In this research, a mesh-free deformation framework has been introduced. The basic concept of the framework is to use discretized points to capture the physical information on a surface and to formulate the deformations with the help of fundamental solutions. The principle of superposition is the foundation of the framework. It implicates that the deformation of a certain load can be decomposed as a sum of other deformations that are caused by the sub-phases of the load. Under this framework, we have investigated the problems of deformation modelling and have developed a number of techniques that can alleviate the labour intensive work in animating deformations.

Directly applying the framework to deformation modelling, we have proposed a technique of mesh-free deformations that is distinguished from other known mesh-

free techniques. It only requires discretization on the surface of a 3D object while others discretize the whole volume. This has made the method particularly suitable for applications in computer graphics because most models in computer graphics are expressed as surfaces. Trade-off between efficiency and accuracy can be controlled by adjusting the density of sampling points. To speed up the computation, two different ways have been adopted: one is the sparse matrix scheme which trades accuracy for speed, and the other is the pre-computation which accelerates at run time by computing some inter-quantities in advance.

As a straightforward application, the animation of the anatomic human muscle structure has been constructed in Appendix A. The mesh-free method deforms the muscles. The deformations are driven by movements of bones taking into account the anatomic shape of the muscles. Unlike the static images, our model illustrated the active muscle movement and deformations in moving sequences, which demonstrated well the validation of our mesh-free deformations.

Since the physically-based deformations are computationally intensive, to further relieve the modeling effort, the technique of reusable deformations has been developed within the mesh-free framework. It simplified the modelling process considerably to save the production cost.

A deformation is identified as a group of virtual forces in the mesh-free framework. Based on it, "copy" and "paste" operations have been developed. The "copy" abstracts the deformation of a specified object out as virtual forces which regulate a deformation field. The "paste" applies the deformation field to another object by exerting the virtual forces on it. With this technique, deformations can be reused as many times as possible, and furthermore a few simple deforming primitives can be combined to create a complex deformation. In processing, the technique does not require vertex correspondence to be established between the source and the target

objects. This differs from most morphing methods. Importantly, using the physically correct deformation model ensures good computational accuracy.

Other applications beyond deformations can be constructed under our mesh-free framework. Two techniques have been addressed in this thesis. One is hyper-twist which uses the definition of distortions in 3D space to create aesthetic shapes. The other is force mapping, which transforms a complicated surface into a simplified one and a force map.

In hyper-twist, the mesh-free deformation was extended to create a deformation of an infinite body. We linked it to the distortion of a 3D space, a hyper-distortion, and any object within the space deforms accordingly. It provided a useful tool for the production of abstract artistic forms.

In the implementation, a hyper-distortion was extended from a pre-defined deformation on a specified object, a seed. It shows that with a very simple seed shape, one is able to produce geometry of significantly different shapes. The number of shapes one can get in one-go is extremely large. Linking control parameters with time, one can easily produce temporal sequences of distorted images. A few animation sequences have been presented and they evolve within the time dimension. These sequences revealed the relation of time and space and the evolution process of the distortion in space.

Force mapping provided a surface construction technique with a similar idea of displacement mapping. It uses forces rather than displacements recorded in the force map to shape the fine details on a surface. It shares the same framework with the other applications. The continual disturbance on a simple surface can be assembled with a sum of kernel functions. The method has potential uses in geometric compression, surface modelling, reconstruction and editing.

10.2 Future Work

10.2.1 Modelling Nonlinearity

In the proposed methods, the Kelvin solutions were used to assemble the deformations. It is an analytic solution of linear deformations. Thus our solutions only provided approximations to the linear problem even though it has better results than the FEM linear module when simulating large deformations as shown in Figure 5.4.

The lack of general analytic solutions in non-linear problems raises a great challenge. However, this could be resolved by distributing points within the 3D volume rather than on the surface only. The traditional mesh-free techniques (Liu 2003) have done so to formulate the equations by minimizing the overall potential energy.

Nonlinearity is caused by geometric changes, such as large deformations, and the nonlinear property of a material. So far, there exist many numerical difficulties to closely approximate the real phenomenon. A good algorithm for simulating the nonlinear deformation which is easy to use is a great challenge.

10.2.2 Haptic Rendering

The method of our mesh-free deformations can provide output of forces as well as displacements. We will envisage that the technique can be extended to the applications of haptic rendering where it can be used to generate the force feedback.

In our implementation, the deformation was controlled by manually painting constraint on a surface. It is not efficient in a virtual environment where objects may interact with each other by collision or contact. To define the precise contact between soft objects, an effective collision detection algorithm is needed. The robustness and the efficiency of the collision detection contribute to the overall performance. Once the intersection area is identified, it is possible to compute the deformation by fixing the applied boundary conditions of this area automatically and

the contact forces on the boundary can be dynamically adjusted. However, the most challenging issue is to accomplish all the computations efficiently as most applications require real-time responses.

10.2.3 Anatomically-based Modelling

Graphical simulation of anatomic structure is useful in art production, medical education, illustration and research. Most bio-material has complex response to the environment. It is challenging to identify the interaction among tissues and draw clear images on the actual anatomic relations. For example, the reaction force of an active muscle is sophisticated to model. A human muscle structure was demonstrated in Appendix A. The muscles were treated as simple elastic material and the interaction among them was not considered. It is hard to claim that our animation is physically correct but it still provided a good illustration on how the muscle moves with bones. Successfully modelling the tissues with correct anatomic structure remains a challenge and it certainly has a great impact on the theory and the research methods related to the study of bio-structures.

APPENDIX A

ANIMATING HUMAN MUSCLE STRUCTURE[†]

The human muscle motion and deformation is of great interest to character animation, medical education, illustration and research. A physically-based method is helpful to create realistic-looking results.

We have developed the mesh-free deformation technique in Chapter 5. In this appendix, we apply this technique to deform human muscles. Taking into account the anatomic shape, deformations of muscles are driven by movement of bones. The mesh-free computation has been encoded allowing fast simulation to be achieved with visually convincing results. Such method can serve as the deformation engine for a muscle illustration system. The static anatomic shapes of muscles are from the data of Ultimate Human (Snoswell 2005). This case study also assures the validity of our technique of mesh-free deformations when it is applied to a complicated system.

[†] This work has been developed jointly with Dr. Xiaosong Yang. My main contribution is on the development of deformation module.

A.1 Illustration of Moving Muscles

Graphical reproduction of the human anatomical structures and their motions are of significant interest to character animation, medical education, illustration and simulation. With the advancement of computer graphics technology, medical illustration can be made not only on paper, but also digitally, allowing multimedia interaction between the user and the virtual human body. A great deal of research efforts have been made by the computer graphics research community to offer increasingly realistic visual fidelity.

To date however, most computer graphical illustrations are primarily passive, i.e. they are basically digital reproduction of their paper counterparts. Primary Pictures has extended the 2D images to the 3D static digital models (Primary 2005). Although one is able to view the anatomical elements, possibly with sound and textural descriptions, interactively through HTML-like links, most available systems cannot depict the motion of the human anatomical structure.

The human muscle structure is a sophisticated system and most muscles deform to a great extent when the human body moves (Creager 1992; Palastanga et al. 1994). The animation of its movement and deformations represents a tremendous technological challenge due to the complex shape of the muscles and the mechanics properties. This explains why most medical illustration techniques where muscles are involved are primitive. Although some simple movements can be visualised for simplified shapes, the movements are displayed as primarily pre-recorded animations (Snoswell 2005).

To illustrate the deformations of muscles, we first rig the movement of the human skeleton structure. The animation of the human skeleton can be reconstructed from the motion capture data, which record the movement of a human character in detail. For some simple movement with a few joints only, we can manually set the parameters of the joint rotations temporarily to create the animation. Linking the muscles to the bone transformations, we generate animation sequences of muscle

deformations with the technique of mesh-free deformations. The mesh-free method deforms the muscle in a physically plausible way and produces realistic-looking illustrations. The kinetic relations between bones and muscles are drawn out and it shows how the muscles drive the skeleton moving.

Although the primary applications are identified as medical illustration, the technological developments presented in this appendix can be extended to other tasks. For example, by considering accurate physical properties, detailed anatomical modelling and complex interaction among the soft tissues can be produced. This is useful for proper medical simulation and is potentially valuable for medical diagnosis of muscle and joint related conditions, such as sports injuries. It may also be useful for the evaluation of rehabilitation treatments.

A.2 Review of Muscle Modelling Techniques

Over the past two decades, computerised muscle deformation modelling has become one of the primary areas of research for human character animation. Existing methods can be broadly classified into two groups: geometrically based and physically based.

In the first category, Scheepers et al. (1997) and Wilhems (1997) used ellipsoids to approximate the appearance of fusiform muscles. The multi-belly muscles are represented by a set of ellipsoids positioned along two spline curves (Scheepers et al. 1997), Sheepers also presented a general model that consists of tubularly-shaped bi-cubic patches. Although these methods try to mimic the physical reality by considering some physical properties, such as volume preservation, the models they employed are too simple to represent the complexity of the human muscle structure.

Chadwick et al. presented the first physically based technique. A muscle is deformed by embedding it into a free-form deformation (FFD) lattice (Chadwick et al. 1989). Hookean springs were used to simulate the dynamics of physical muscle tissues. The springs deform the FFD lattice that in turn deforms the embedded muscle. Although

this represents a progress in animation, the complex deformations of muscles cannot be well simulated by a simple spring system. Nedel and Thalmann (1998) introduced the idea of action lines for the deformation of the muscles. The idea is very simple to implement, but the limitations are the same as the simple spring systems. Chen and Zeltzer (1992) presented a bio-mechanical muscle model using the finite element method. They discussed a single muscle working in isolation. Delp and Loan's work also addressed single muscles (Delp and Loan 1995). In (Middleton et al. 1999; Yucesoy et al. 2002), highly detailed muscles and their stress and deformation relationship were modelled using nonlinear solid mechanics. These methods, despite being theoretically more accurate than the earlier techniques, suffer from very high computing costs, which render it impractical for interactive illustration applications.

The cgCharacter company also developed an illustration system for the human muscle structure (Snoswell 2005). But they only supply still images and models. Their motion demos are pre-recorded animations.

A.3 Mesh-free Muscle Deformations

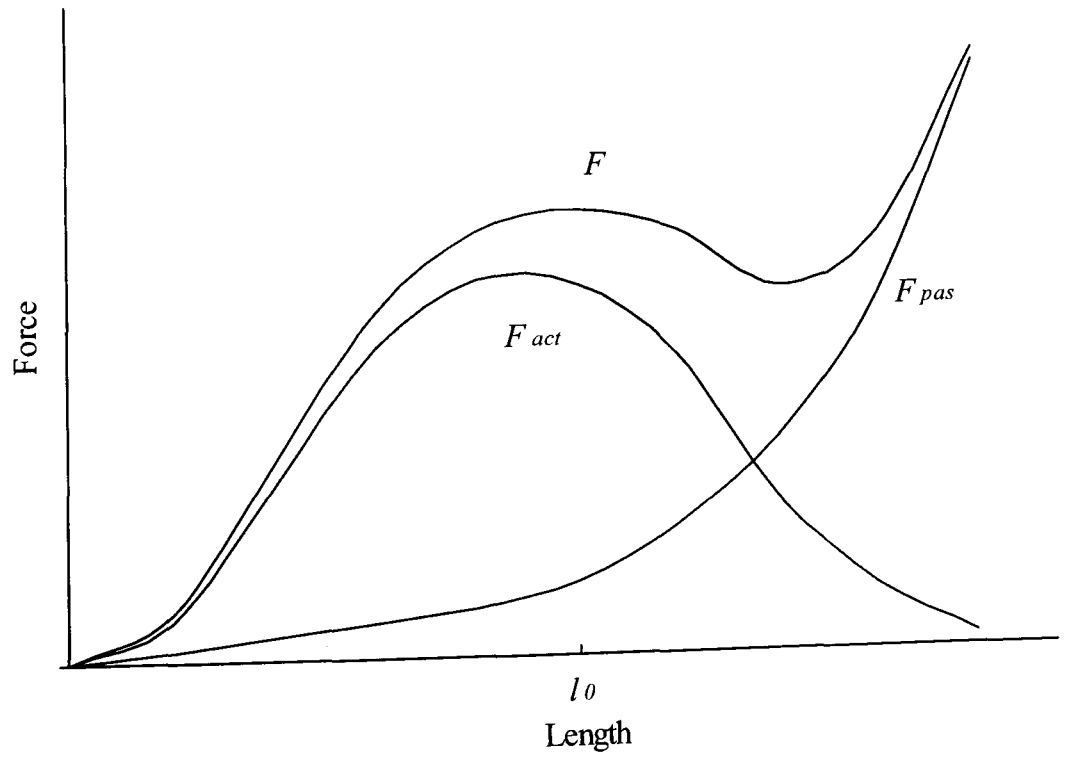


Figure A.1 The force-length relationship of muscles (Winter 1990)

A muscle is composed of fascicles which contain bundles of fibers, and the fibers

themselves are composed of parallel bundles of myofibrils (Wawick and Williams 1973). The myofibril serves as the basic fiber of muscle, which consists of many contractile units, sacromeres, composed of arrays of actin and myosin myofilaments, which form the contractile mechanism of muscle.

The force-length relationship of muscles is shown in Figure A.1. Muscle forces F contain both passive and active components, where the passive part F_{pas} acts in the non-linear fashion similar to other soft tissues and the active part F_{act} is due to the change in the micro contractile element, myofilament (Winter 1990).

The theory of mesh-free deformations is addressed in Chapter 5. In this method, the surface is discretized into a number of collocation points (CPs), which capture the boundary constraints, and virtual source points (VSPs) are used to define the fundamental solutions. The trade-off between efficiency and accuracy is achieved simply by distributing more or less of those points. Governed by the physical laws, the behaviours of muscles are numerically represented.

Instead of using a non-linear active constitutive model, the mesh-free method proposed uses the linear elastic media to describe the deformation of muscle. It is obvious that some important physical characteristics of muscle are omitted and the computations are simplified. Therefore, as the method serves only for illustration purposes, visual plausible rather than accurate images are produced.

The anatomic shapes of muscles from (Snoswell 2005) are used to produce realistic-looking results. The muscles are attached to the bones according to the anatomic relationship. In our implementation, for a given muscle, we simply identify those CPs that are connected to a given bone and force them to move with the bone temporarily. A dynamic displacement constraint is added to them.

$$\mathbf{x}_t = \mathbf{R}_t \mathbf{x}_0 \tag{A.1}$$

Here the $\mathbf{x}_t$ is the position of a CP attached to the bone at time t, $\mathbf{x}_0$ is its original

position, $\mathbf{R}_i$ is the transformation matrix which describes the bone movement. The rest part of the muscle deforms accordingly.

In our implementation, the contact actions have not been modelled and their effects do not appear in the computation, which somehow causes defects in the final images. In the future, it will be worthwhile to implement the contact engine to fix the intersections amongst muscles and the intersection between muscles and bones.

A.4 Results

Based on the mesh-free deformations, we can visualize the movement and deformations of the human muscle structure. In general, a muscle was represented with about 400 collocation points and about 100 virtual source points were used to create the deformations. It would cost about one second to update one frame of a single muscle with a Dell Pentium III 500 MHz PC.

So far some deformations of the major muscles on the limbs and torso have been accomplished. We show some of the deformation samples in Figure A.2, A.3 and A.4. The pictures here are the scene-shots of animation sequences computed out by our program. Among them, Figure A.2 shows three major muscles, Pectoralis major, Deltoid and Latissimus, work together to lift the arm, where other muscles are hidden to highlight the three. It clearly illustrates the shapes and the orientations of the muscles as they change with different skeleton configuration. Figure A.3 shows the muscles of the arm deforming when the arm bends and Figure A.4 illustrates how the muscles control the twist of the forearm.

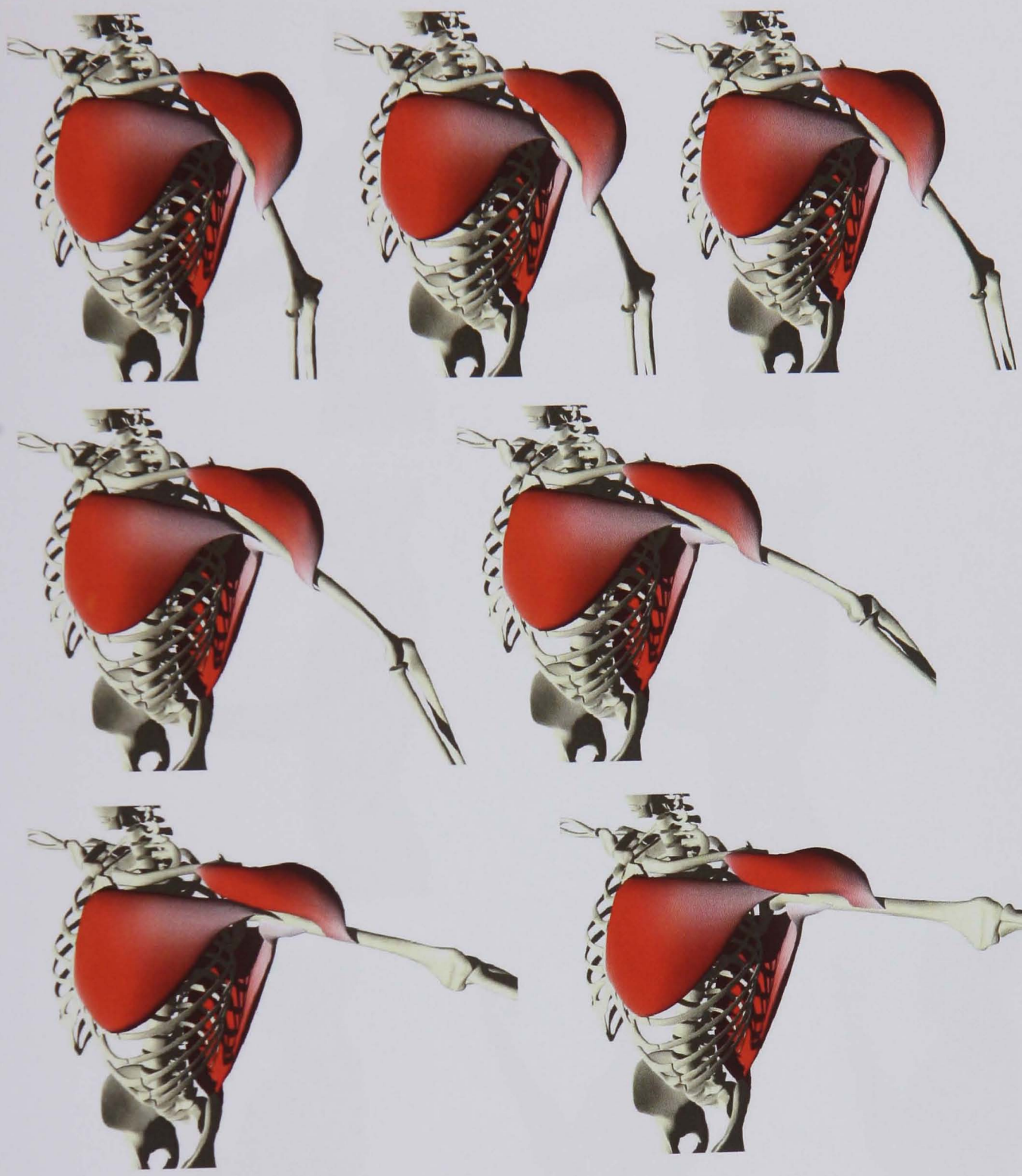


Figure A.2 Illustration of the deformations of Pectoralis major, Deltoid and Latissimus dorsi following the shoulder movement.

Figure A.3 Illustration of the muscle deformations for the left arm bending.

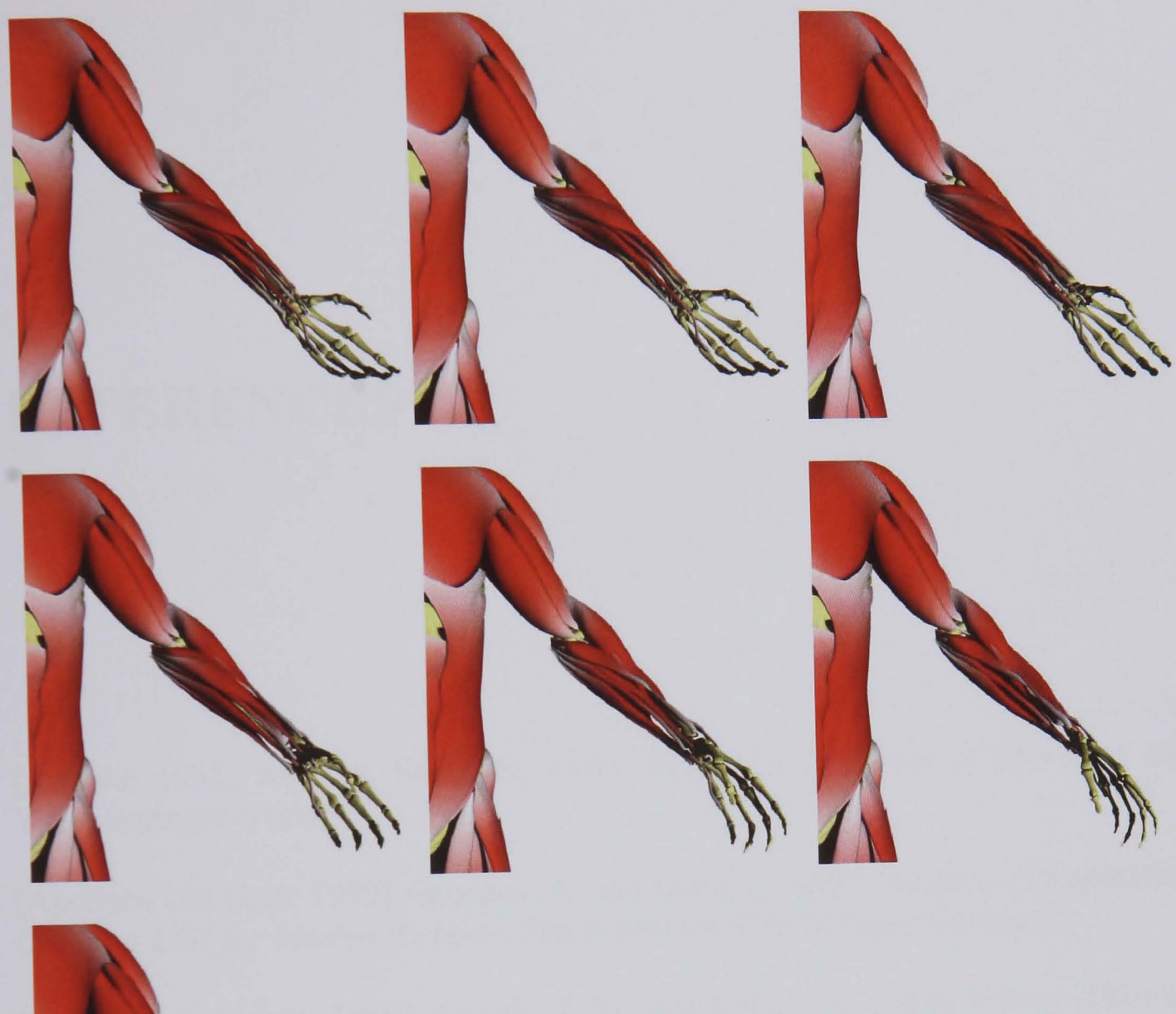


Figure A.4 Illustration of the muscle deformations following the forearm twist movement.

REFERENCES

[Antman 1995] Antman, Stuart S., 1995. *Nonlinear Problems of Elasticity*. New York: Springer-Verlag.

[Apodaca and Gritz 1999] Apodaca, A. and Gritz, L., 1999. *Advanced RenderMan – Creating CGI for Motion Pictures*, San Francisco, CA: Morgan Kaufmann.

[Argyris and Kelsey 1960] Argyris, J. H. and Kelsey, S., 1960. *Energy Theorems and Structural Analysis*, London: Butterworth.

[Baraff 1992] Baraff, D., 1992. Dynamic Simulation of Non-Penetrating Flexible Bodies, *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, p.303-308.

[Baraff and Witkin 1998] Baraff, D. and Witkin, A., 1998. Large Steps in Cloth Simulation, *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, p. 43-54.

[Barr 1984] Barr, Alan H., 1984. Global and Local Deformations of Solid Primitives, *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, p.21-30.

[Bathe 1996] Bathe, K. J., 1996. *Finite Element Procedures*, 2nd edition, London: Prentice-Hall.

[Beer 2000] Beer, G., 2000. *Programming the Boundary Element Method: an Introduction for Engineers*, New York: Wiley.

[Blinn 1978] Blinn, J. F., 1978. Simulation of Wrinkled Surfaces, *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*. p. 286-292.

[Bridson et al. 2003] Bridson, R., Marino, S. and Fedkiw, R., 2003. Simulation of Clothing with Folds and Wrinkles, *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p. 28-36.

[Candy and Edmonds 2002] Candy, L. and Edmonds, E., 2002. *Explorations in Art and Technology*, London: Springer.

[Capell et al. 2002] Capell, S., Green, S., Curless, B., Duchamp, T. and Popović, Z., 2002. A Multiresolution Framework for Dynamic Deformations, *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p.41-47

[Cartwright 2001] Cartwright, D. J., 2001. *Underlying Principles of the Boundary Element Method*, Southampton: WIT.

[Chadwick et al. 1989] Chadwick, J., Haumann, D., and Parent, R., 1989. Layered Construction for Deformable Animated Characters, *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, p. 243-252

[Chang and Zhang 2004] Chang, J. and Zhang, J. J., 2004. Mesh-free Deformations. *Computer Animation and Virtual Worlds*, v. 15, n. 3-4, p. 211-217.

[Chen and Zeltzer 1992] Chen, David T. and Zeltzer, D., 1992. Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle Using the Finite Element Method, *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, p.89-98,

[Cook 1984] Cook, R. L., 1984, Shade Trees. *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, p. 223-231.

[Cook et al. 1987] Cook, R. L., Carpenter, L., and Catmull, E., 1987. The Reyes Image Rendering Architecture, *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, p. 95-102.

[Coquillart 1990] Sabine Coquillart, 1990. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling, *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, p.187-196.

[Cotin et al. 1999] Cotin, S., Delingette, H. and Ayache, N., 1999. Real-Time Elastic Deformations of Soft Tissues for Surgery Simulation, *IEEE Transactions on Visualization and Computer Graphics*, v.5 n.1, p.62-73.

[Creager 1992] Creager, Joan G., 1992. *Human Anatomy and Physiology*. 2nd edition, Dubuque, Iowa: Wm. C. Brown

[Cyberware 2005] Cyberware, 2005. *Cyberware.*
Available from: http://www.cyberware.com/products/scanners/index.html
[Accessed Mar 2005]

[Debunne et al. 1999] Debunne, G., Desbrun, M., Barr, A. and Cani M.P., 1999. Interactive Multiresolution Animation of Deformable Models. *Eurographics Workshop on Computer Animation and Simulation '99*, Springer-Verlag Editor

[Debunne et al. 2001] Debunne, G., Desbrun, M., Cani M.P., and Barr, A., 2001. Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling, *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, p.31-36.

[Delp and Loan 1995] Delp, Scott L. and Loan, J. Peter, 1995. A Graphics-based Software System to Develop and Analyze Models of Musculoskeletal Structures, *Computers in Biology and Medicine*, v. 25, n. 1, p. 21-34.

[Deng 1988] Deng, X. Q., 1988. *A Finite Element Analysis of Surgery of the Human Facial Tissue*, Ph. D. thesis, Columbia University.

[Desbrun and Gascuel 1996] Desbrun, M. and Gascuel, M.-P., 1996. Smoothed Particles: a New Paradigm for Animating Highly Deformable Bodies, *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96*, p.61-76.

[Dewaele and Cani 2004] Dewaele, G. and Cani, M. P., 2004. Interactive Global and Local Deformations for Virtual Clay, *Graphical Models*, v.66 n.6, p.352-369.

[Dhatt et al. 1984] Dhatt, G., Touzot, G. and Cantin, G. 1984. *The Finite Element Method Displayed.* A Wiley-Interscience Publication.

[Fagan 1992] M. J. Fagan, 1992. *Finite Element Analysis: Theory and Practice*, Harlow: Longman Scientific & Technical

[Fung 1965] Fung, Y. C., 1965. *Foundations of Solid Mechanics*. Englewood Cliffs, N. J.: Prentice Hall.

[Fung 1994] Fung, Y. C., 1994. *A First Course in Continuum Mechanics: For Physical and Biological Engineers and Scientists*. 3rd Edition, Englewood Cliffs, N. J.: Prentice Hall.

[Gibson 1997] Gibson, S. F., 1997. 3D ChainMail: a Fast Algorithm for Deforming Volumetric Objects, *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, p.149-154

[Greissmair and Purgathofer 1989] Greissmair,J.and Purgathofer, W. 1989. Deformation of Solids with Trivariate B-Splines. In *Eurographics '89*, Eurographics, North Holland, Amsterdam, The Neterlands, p. 137-148.

[Golub 1983] Golub, G.H., Charles, F. and Loan, V., 1983. *Matrix Computations*. The Johns Hopkins University Press.

[Gourret et al. 1989] Gourret, J. P., Thalmann, N. M. and Thalmann, D., 1989. Simulation of Object and Human Skin Formations in a Grasping Task, *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, p.21-30.

[Gumhold and Hüttner 1999] Gumhold, S. and Hüttner, T., 1999. Multiresolution Rendering with Displacement Mapping, *SIGGRAPH workshop on Graphics hardware*, p.8-9

[Gumowski and Mira 1974] Gumowski, J. and Mira, C., 1974. Point Sequences Generated by Two-dimensional Occurrences, *Information Processing 1974*, Amsterdam, p. 851-855

[Hirota 2002] Hirota, G., 2002. *An Improved Finite Element Contact Model for Anatomical Simulations*. Ph. D. thesis, University of North Carolina.

[Hsu et al. 1992] Hsu, William M., Hughes, John F. and Kaufman, Henry, 1992. Direct Manipulation of Free-Form Deformations, *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, p.177-184.

[Irving et al. 2004] Irving, G., Teran, J. and Fedkiw, R., 2004. Invertible Finite Elements for Robust Simulation of Large Deformation, *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p.131-140.

[James and Pai 1999] James, D. L. and Pai, D. K., 1999. ArtDefo: Accurate Real Time Deformable Objects. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, p.65-72.

[James and Pai 2003] James, D. L. and Pai, D. K. 2003. Multiresolution Green's Function for Interactive Simulation of Large-Scale Elastostatic Objects. *ACM ACM Transactions on Graphics (TOG)*, v.22 n.1, p.47-82.

[Kautz and Seidel 2001] Kautz, J., and Seidel, H. P., 2001. Hardware Accelerated Displacement Mapping for Image Based Rendering, *Graphics Interface*, p. 61-70.

[Kim et al. 2001] Kim, J. S., Lee, J. H. and Park, K. H., 2001. A Fast and Efficient Bump Mapping Algorithm by Angular Perturbation, *Computers & Graphics,* v. 25, p.401-407.

[Kugler 1998] Kugler, A., 1998. IMEM: An intelligent memory for bump- and reflection-mapping, *Proceedings of the SIGGRAPH/Eurographics Workshop on Graphics Hardware,* New York, p. 113-122.

[Lee et al. 2002] Lee, A., Moreton, H., and Hoppe, H., 2002. Displaced Subdivision Surfaces, *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, p. 85-94.

[Li and Brodlie 2003] Li, Y. and Brodlie, K., 2003. Soft Object Modelling with Generalised ChianMail – Extending the Boundaries of Web-based Graphics, *Computer Graphics Forum*, v. 22, n. 4, p. 717-727.

[Liu 2003] Liu, G. L., 2003. *Mesh Free Methods, Moving Beyond the Finite Element Method*, CRC Press LLC.

[MacCracken and Joy 1996] MacCracken, Ron and Joy, Kenneth I., 1996. Free-Form Deformations with Lattices of Arbitrary Topology, *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, p.181-188.

[Mealing 2002] Mealing, S., 2002. *Computers and Art*. 2nd Edition, Bristol: Intellect Ltd.

[Middleton et al. 1999] Middleton, J., Pandel, G. N., and Jones M. L. 1999. *Computer Methods in Biomechanics and Biomedical Engineering*, Taylor & Francis.

[Miller 1988] Miller, Gavin S.P., 1988. The Motion Dynamics Of Snakes and Worms, *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, p. 169-178.

[Monaghan 1992] Monaghan, J., 1992. Smoothed Particle Hydrodynamics, *Annual Reviews of Astronomy & Astrophysics*, v.30, p.543-574.

[Müller et al. 2002] Müller, M., Dorsey, J., McMillan, L., Jagnow, R. and Cutler, B., 2002. Stable Real-time Deformations, *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p. 49-54

[Müller et al. 2003] Müller M., Charypar, D. and Gross, M., 2003. Particle-based Fluid Dimulation for Interactive Applications, *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p.154-159.

[Müller et al. 2004] Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M. and Alexa, M., 2004. Point Based Animation of Elastic, Plastic and Melting Objects, *Proceedings of 2004 ACM SIGGRAPH/Eurographics on Computer Animation*, p.141-151.

[NASA 2005] NASA, 2005. *The Hubble Space Telescope*.
Available from: http://hubble.nasa.gov/index.php [Accessed 10 Feb, 2005]

[Nedel and Thalmann 1998] Nedel, L. P. and Thalmann, D., 1998. Real Time Muscle Deformation using Mass Spring Systems, In *Proceedings of Computer Graphics International 1998*, IEEE Computer Society Press, p156-165.

[O'Brien and Hodgins 1999] O'Brien, J. F. and Hodgins, J. K., 1999. Graphical Modeling and Animation of Brittle Fracture, *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, p.137-146.

[O'Brien et al. 2002] O'Brien, J. F., Bargteil, A. W. and Hodgins, J. K., 2002. Graphical Modeling and Animation of Ductile Fracture, *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, p. 291- 294.
[Ogden 1997] Ogden, R. W. 1997, *Non-linear Elastic Deformations*, Dover Publications.

[Palastanga et al. 1994] Palastanga, N., Field, D. and Soames R., 1994. *Anatomy and Human Movement: Structure and Function*. 2nd Edition, Oxford: Butterworth-Heinemann.

[Pauly et al. 2005] Pauly, M., Keiser, R., Adams, B., Dutré, P., Gross M. and Guibas, L.J., 2005. Meshless Animation of Fracturing Solids, *ACM Transactions on Graphics (TOG)*: *Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques*, v.24, n.3, p.957-964.

[Primary 2005] Primary Pictures, 2005. *The 3D Human Anatomy Resource*, Primary Pictures Ltd.
Available from: http://www.primalpictures.com/index.aspx [Accessed Mar 2005]

[Saada 1993] Saada, Adel S., 1993. *Elasticity Theory and Applications*. 2nd Edition, Malabar, Florida: Krieger Publishing Company.

[Scheepers et al. 1997] Scheepers, F., Parent, R. E., Carlson, W. E., and May, S. F., 1997. Anatomy-based modeling of the human musculature. In *Proceedings of the 24th Annual Conference on Computer Graphics and interactive Techniques*, p. 163-172.

[Sederberg and Parry 1986] Sederberg, Thomas W. and Parry, Scott R., 1986. Free-Form Deformation of Solid Geometric Models, *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, p.151-160.

[Singh and Fiume 1998] Singh, Karan and Fiume, Eugene, 1998. Wires: A Geometric Deformation Technique, *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, p.405-414.

[Snoswell 2005] Snoswell, Mark, 2005. *The Ultimate Human*, cgCharacter.
Available from: http://www.cgcharacter.com/ultimatehuman.html
[Accessed Mar 2005]

[Stanford 2004] Stanford, 2004. *The Stanford 3D Scanning Repository*, Stanford University.
Available from: http://www-graphics.stanford.edu/data/3Dscanrep/
[Accessed Mar 2004]

[Szeliski and Tonnesen 1992] Szeliski, R. and Tonnesen, D., 1992. Surface Modeling with Oriented Particle Systems, *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, p.185-194

[Tarini et al. 2000] Tarini, M., Cignoni, P., Rocchini, C. and Scopigno, R., 2000. Real Time, Accurate, Multi-Featured Rendering Of Bump Mapped Surfaces, *Computer Graphics Forum,* v. 19 (No. 3), p. 119-130.

[Terzopoulos et al. 1987] Terzopoulos, D., Platt, J., Barr A., and Fleischer, K., 1987. Elastically Deformable Models, *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, p. 205-214.

[Terzopoulos and Fleischer 1988] Terzopoulos, D. and Fleischer, K., 1988. Modeling Inelastic Deformation: Viscolelasticity, Plasticity, Fracture, *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques,* p.269-278.

[Timoshenko and Goodier 1970] Timoshenko, S. and Goodier, J. N., 1970. *Theory of Elasticity*, McGraw-Hill, New York.

[Tu and Terzopoulos 1994] Tu, X. and Terzopoulos, D., 1994. Artificial Fishes: Physics, Locomotion, Perception, Behavior, *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, p.43-50.

[Turner et al. 1956] Turner, M. J., Clough, R. W., Martin, H. C. and Topp, L. J., 1956. Stiffness and Deflection Analysis of Complex Structures, *Journal of Aeronautical Science*, v. 23, p. 805-823.

[Volino et al. 1996] Volino, P., Thalmann, N. M., Shen J. and Thalmann D., 1996. An Evolving System for Simulating Clothes on Virtual Actors, *IEEE Computer Graphics and Applications*, v.16 n.5, p.42-51.

[Wang et al. 2003] Wang, L., Wang, X., Tong, X., Lin, S., Hu, S., Guo B. and Shum, H.-Y., 2003. View-Dependent Displacement Mapping, *ACM Transactions on Graphics (TOG),* v. 22, n. 3, p. 334-339.

[Waters 1987] Waters, Keith., 1987. A Muscle Model for Animation Three-Dimensional Facial Expression, *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, p. 17-24.

[Wawick and Williams 1973] Wawick, R, and Williams, P. L., 1973. *Gray's Anatomy*, 35th British edition. Philadelphia: W.B. Saunders

[Wiedermann 2002] Wiedermann, J., 2002. *500 3D-objects*, Taschen.

[Wilhelms 1997] Wilhelms, J., 1997. Animals with Anatomy, *IEEE Computer Graphics and Applications,* v.17, May 1997, p.22-30

[Winter 1990] D. Winter, 1990. Muscle Mechanics, In *Biomechanics and Motor Control of the Human Movements*, 2nd edition, New York: Wiley-Interscience

[Yucesoy et al. 2002] Yucesoy, C. A., Koopman, B. H., Huijing, P. A. and Grootenboer, H. J., 2002. Three-dimensional Finite Element Modelling of Skeletal Muscle Using a Two-domain Approach: Linked Fiber-matrix Mesh Model, *Journal of Biomechanics*, v. 35 n. 9, p. 1253-1262.

[Zhuang and Canny 1999] Zhuang, Y. and Canny, J., 1999. Real-time and Physically Realistic Global Deformation, *SIGGRAPH'99 Sketches and Apllication Proceedings.*