PHYSICALLY-BASED ANIMATION OF ELASTICALLY DEFORMABLE MODELS

A DISSERTATION SUBMITTED TO
THE DEPARTMENT OF COMPUTER ENGINEERING AND
INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

by
Uğur Güdükbay
February 1994

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Bülent Özgüç (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Yılmaz Tokad

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Mehmet Baray

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Assoc. Prof. Varol Akman

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Assist. Prof. Cevdet Aykanat

Approved by the Institute of Engineering and Science:

Prof. Mehmet Baray,
Director of the Institute of Engineering and Science

ABSTRACT

PHYSICALLY-BASED ANIMATION OF ELASTICALLY DEFORMABLE MODELS

Uğur Güdükbay
Ph.D. in Computer Engineering and Information Science
Supervisor: Prof. Bülent Özgüç
February 1994

Although kinematic modeling methods are adequate for describing the shapes of static objects, they are insufficient when it comes to producing realistic animation. Physically-based modeling remedies this problem by including forces, masses, strain energies, and other physical quantities. The behavior of physically-based models is governed by the laws of rigid and nonrigid dynamics expressed through a set of equations of motion. In this thesis, we describe a system for the animation of deformable models. A spring force formulation for animating deformable models is also presented. The animation system uses the physically-based modeling methods and the approaches from elasticity theory for animating the models. Three different formulations, namely the *primal*, *hybrid*, and the *spring force* formulations, are implemented so that the user could select the suitable one for an animation, considering the advantages and disadvantages of each formulation. Collision of the models with impenetrable obstacles and constraining model points to fixed positions in space are implemented.

Keywords: Physically-based modeling, deformable models, animation, simulation, constraints, collision detection, collision response, partial differential equations, linear system solver.

ÖZET

ELASTİK OLARAK DEFORME EDİLEBİLEN MODELLERİN FİZİĞE DAYALI ANİMASYONU

Uğur Güdükbay
Bilgisayar Mühendisliği ve Enformatik Bilimleri Bölümü
Doktora
Tez Yöneticisi: Prof. Dr. Bülent Özgüç
Şubat 1994

Kinematik modelleme yöntemleri nesnelerin şekillerini tanımlamakta yeterli olmakla beraber gerçeğe uygun animasyon üretmek sözkonusu olduğunda yetersiz kalmaktadır. Fiziğe dayalı modelleme yöntemleri bu sorunu kuvvet, kütle, enerji, v.b. büyüklükleri kullanarak çözmektedir. Fiziğe dayalı modellerin hareketi rijit ve rijit olmayan dinamik yasaları ile belirlenmiştir. Hareket denklemleri bu modellerin dinamik hareketini tanımlar. Bu çalışmada rijit olmayan (deforme edilebilen) modellerin animasyonu için geliştirilmiş bir sistem anlatılmaktadır. Bu sistem, modellerin animasyonu için fiziğe ve elastisite kuramına dayanan yaklaşımları kullanmaktadır. Aynı zamanda, deforme edilebilen nesnelerin animasyonu için yeni bir yöntem ("yay kuvvet yöntemi") geliştirilmiştir. Animasyon sisteminde "primal", "hibrid", ve "yay kuvvet" yöntemleri kullanılarak modeller hareket ettirilmektedir. Bu yolla kullanıcı yöntemlerin avantaj ve dezavantajlarına göre modele uygun olan yöntemi seçebilmektedir. Modellerin sabit engellerle çarpışması ve modeller üzerindeki bazı noktaların hareketinin kısıtlanması gibi seçenekler animasyonlarda kullanılabilmektedir.

Anahtar sözcükler: Fiziğe dayalı modelleme, deforme olabilen modeller, animasyon, benzetim, kısıtlamalar, çarpışma tespiti, çarpışma sonrası hareket, kısmi türevsel denklemler, doğrusal denklem sistemi çözücüsü.

Acknowledgements

I would like to express my deepest gratitude to Prof. Bülent Özgüç for his supervision, encouragement, and invaluable advice in all steps of the development of this work. It is my extraordinary chance to work with him throughout my graduate study, to be instructed and trained in all aspects of the research.

I appreciate Prof. Yılmaz Tokad of the Eastern Mediterranean University for his valuable discussions on the implementation of deformable models, and for his motivating advice on my research. I would like to thank to Assoc. Prof. Varol Akman for carefully reading my papers, proposal, and thesis and offering various suggestions.

I also appreciate Murat Aktıhanoğlu for the shading program, and Aydın Ramazanoğlu for his help in producing the color photographs.

I am grateful to Faruk Polat, İsmail Hakkı Toroslu, Veysi İşler, Özlem Özgü, and Erkan Tın, for their continuous encouragement in all stages of this study, and to Bilge Aydın, secretary of the CEIS Department, for her logistical support, and to all members of the CEIS Department for their contributions in one way or another through formal and informal discussions.

Finally, my sincere thanks are due to my family for their continuous moral support throughout my graduate study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The use of computer graphics and numerical methods for three-dimensional design and modeling provides an interactive environment in which designers can formulate and represent shapes of objects. Modeling the shapes as a composition of geometrically and algebraically defined primitives, simulating scenes with shading and texture, and producing usable design images are the most important requirements for application areas such as Computer-Aided Design and Computer-Aided Manufacturing.

Currently, most of the methods used for modeling are *kinematic*. This becomes a major drawback when we want to create realistic animation since these methods are passive; they do not interact with each other or with external forces. The behavior and form of many objects are determined by the objects' gross physical properties. For example, how a cloth drapes over objects is determined by the surface friction, the weave, and the internal stresses and strains generated by forces from the objects. As another example, a chain suspended between two poles hangs in an arc determined by the force of gravity and the forces between adjacent links that keep the links from separating.

To achieve realism in animation a model should be able to follow pre-defined paths while still moving in an interesting manner and interacting with other models as real physical objects would do. To build and animate active models, physically-based techniques should be used. These techniques facilitate the

creation of models capable of automatically synthesizing complex shapes and realistic motions that are attainable only by skilled animators. *Physically-based modeling* achieves this by adding physical properties to the models [23]. Such properties may be forces, torques, velocities, accelerations, kinetic and potential energies, and heat. Physical simulation is then used to produce animation based on these properties. To this end, solution of the *initial value problems* is required so that the course of a simulation is determined by objects' initial positions and velocities, and by the forces and torques applied to the object along the way.

Physical simulation alone is not enough since the animator also wants to "control" the motion of objects so that he can specify the goals of motion, the way motion should be performed, and so on. Physical simulation produces impressive results, but is difficult to control since an animator cannot easily establish an intuitive link between the parameters of a simulation and the resulting motion. Besides, physical simulation is computationally expensive. Due to these reasons, it is not in wide use. However, researchers continue to present faster and simpler formulations to build and control the motion of models [3, 12, 45, 47]. Some of the proposed methods are related to the animation* of models composed of parts which are connected by joints, namely *articulated bodies* (such as humans and robots) [2, 4, 15, 18, 29, 31, 43].

Constraints provide a unified method to build objects and animate them [26]. The models assemble themselves as the elements move to satisfy the constraints. Geometric constraints, namely attachment constraints, are used to create complex models from primitive bodies. In other words, they model the joints between the links of a complex model. There are other constraints, such as "point-to-path" constraints, which can be used to control the motion of the models. Such constraints are also called "kinematic constraints."

*A useful bibliography of computer animation can be found in [39].

1.1 Application Areas for Animation

Animation covers all changes that have a visual effect, which are time-varying position (*motion dynamics*), shape, color, transparency, structure, and texture of an object (*update dynamics*), and changes in lighting, camera position, orientation, and focus, and even changes of rendering technique.

The most important application areas of animation are the entertainment industry, education, industrial applications such as control systems and flight simulators for aircraft, and scientific research. The animations in scientific visualization (scientific applications of computer graphics) are generated from simulations of scientific phenomena. The results of the simulations may be large data sets representing 2D or 3D data (e.g., in the case of fluid-flow simulations); these data are converted into images that then constitute the animation. At the other extreme, the simulation may generate positions and locations of physical objects, which must then be rendered in some form to generate animation. This happens, for example, in chemical simulations where the position and orientation of the various atoms in a reaction may be generated by simulation, but the animation may show a ball-and-stick view of each molecule, or may show overlapping smoothly shaded spheres representing each atom. In some cases, the simulation program will contain an embedded animation language so that the simulation and the animation proceed simultaneously.

1.2 Deformable Models

An important aspect in realistic animation is modeling the behavior of *deformable objects*. To simulate the behavior of deformable objects, we should approximate a continuous model by using discretization methods, such as finite difference and finite element methods. For finite difference discretization, a deformable object could be approximated by using a grid of control points where the points are allowed to move in relation to one another. The manner in which the points are allowed to move determines the properties of the deformable object. Simulating

the physical properties (such as tension and rigidity), static shapes exhibited by a wide range of deformable objects (including string, rubber, cloth, paper, and flexible metals) can be modeled. For example, to obtain the effect of an elastic surface, the grid points are connected by springs. The physical quantities cited earlier should be used to simulate the dynamics of these objects. Various researchers [11, 16, 32, 30, 37, 36, 44] presented discrete models which are based on *elasticity and plasticity theory* and use *energy fields* to define and enforce constraints for modeling and animating deformable objects.

This thesis presents a new formulation for the animation of deformable models, called the *spring force formulation.* In other formulations based on elasticity theory (*primal* and *hybrid* formulations), the elastic properties of the materials are stored in the stiffness matrix. However, the formation of the stiffness matrix automatically is very difficult and sometimes it becomes impossible to solve the differential equations for animating the models because of the numerical ill-conditioning problems. In this formulation, instead of forming the stiffness matrix automatically, elastic forces are represented as external spring forces. Although handling the elasticities using the stiffness matrix approach is elegant and the most suitable way, our approach is more effective and very fast.

An animation system which is implemented for the animation of nonrigid (deformable) models is also discussed. The system is built on top of a modeling system for representing 3D free-form objects, that uses superquadrics [6] and Bézier surfaces [9] as modeling techniques, and regular deformations [7] and Free-Form Deformations [34] for deforming these models to obtain irregular, free-form objects [25, 21]. The static models obtained by these methods can be animated using the techniques discussed in this thesis. The system is implemented using the C language [19] on a Unix[†] workstation (Sun-3 or Sparc). The implementation uses the facilities provided by Sun View[‡] system such as windows, panels, and menus [24, 22].

[†]Unix is a registered trademark of AT&T Bell Laboratories.

[‡]Sun View [17] is a registered trademark of Sun Microsystems, Inc.

1.3 The Organization of the Thesis

In Chapter 2, constraint-based methods for animation are discussed. Different uses of constraints are also given.

In Chapter 3, the collision detection and response problem is explained. Different algorithms to solve this problem are discussed together with their applicability to rigid, flexible, and articulated bodies.

In Chapter 4, a short description of the methods proposed by Terzopoulos et al. [37, 36, 38] for elastically deformable models is presented. Then, the implementation details of these methods in the context of our system, and algorithmic solution of the problems, such as collision of flexible models with impenetrable obstacles, are explained. A spring force formulation for animating deformable models is also presented together with its implementation details. Different formulations are compared to each other in terms of their processing times and their ability to model elastic properties.

In Chapter 5, some simulation results representing the features of our system are presented.

Chapter 6 gives conclusions and suggestions for further research.

Chapter 2

Constraint-Based Methods for Animation

A good deal of research has been done towards the use of constraint methods to create realistic animation [45, 46, 47, 32]. Many constraint-based modeling systems have been developed, including constraint-based models for human skeleton [4] (in which the connectivity of segments and limits of angular motion on joints are specified), the *dynamic constraints* [8], and the *energy constraints* [44].

Constraints provide a way to specify the behavior of physical objects in advance without specifying their exact positions, velocities, etc. In other words, constraints are partial descriptions of the objects' desired behavior. So given a constraint, we must determine the forces to meet the constraint and then find forces to maintain the constraint. For example, consider a bead sliding freely on a rigid wire (Fig. 2.1). The behavior of the bead can be described by the fact that it will stay on the wire during its motion no matter what forces act on it. To keep the bead on wire during its motion, a constraint force $\vec{f_c}$ must be applied. If $\vec{f_a}$ is the force applied to the bead at any time and $\vec{f_c}$ is the constraint force then the total force $\vec{f} = \vec{f_a} + \vec{f_c} = k\vec{t}$ where $\vec{t}$ is the tangent vector to the wire. In other words, $\vec{f_c}$ is the force to be added to the applied force to make the bead accelerate in a manner consistent with the constraint.

Constraint-based modeling systems allow the user to specify a collection of

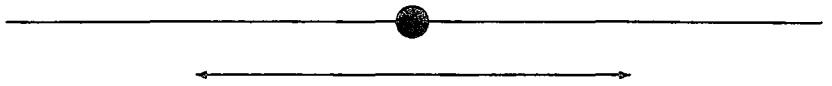


Figure 2.1: A bead sliding freely on a rigid wire.

constraints that the parts of a model are supposed to satisfy. A model may be *underconstrained*, in which case there are additional degrees of freedom that the modeler can adjust (e.g., the location of the point of contact of a sphere and a cube), or *overconstrained*, in which case some of the constraints may not be satisfied (which could happen if both the top and bottom of the sphere were constrained to lie on the top face of the cube). In constraint-based modeling, the constraints must be assigned a priority, to satisfy the most important constraints first.

In energy-constraint systems, constraints are represented by functions that are nonnegative everywhere, and are zero exactly when the constraints are satisfied. (These are functions on the set of all possible states of the objects being modeled.) These are summed to give a single function E. A solution to the constraint problem occurs at a state for which E is zero. Since zero is minimum for E (its component terms are all nonnegative), such states can be located by starting at any configuration and altering it so as to reduce the value of E. The minimum is found by way of numerical methods.

The specification of constraints is complex. Certain constraints can be given by sets of mathematical equalities (e.g., two objects that are constrained to touch at specific points), or by sets of inequalities (e.g., when one object is constrained to lie inside another). Other constraints are more difficult to specify. For example, constraining the motion of an object to be governed by the laws of physics requires the specification of a collection of differential equations. Such constrained systems, however, lie at the heart of physically-based modeling.

2.1 The Uses of Constraints

Constraints can be used for different purposes in animation (cf. Fig. 2.2) [8]:

- *Point-to-nail constraint*: This is used to fix a point on a model to a user-specified location in space. The body can make a pendulum motion about the constrained point, but the constrained point may not move.
- *Point-to-point (attachment) constraint*: This is used to attach two points on different bodies to create complex models from simpler ones. In other words, attachment constraints model the joints between bodies. The bodies may move around freely, as long as the two constrained points stay in contact.
- *Point-to-path constraint*: This type of constraint requires some points on a model to follow an arbitrary user-specified path (a trajectory which is specified as a function of time). Here, the rest of the body is allowed to move according to the forces and torques acting on the body.
- *Orientation constraint*: This type of constraint is used to align objects by rotating them.
- Other constraints, such as *point-on-line* which restricts a point to move on a given line, *sphere-to-sphere* which requires two spheres to touch while sliding along each other, etc.

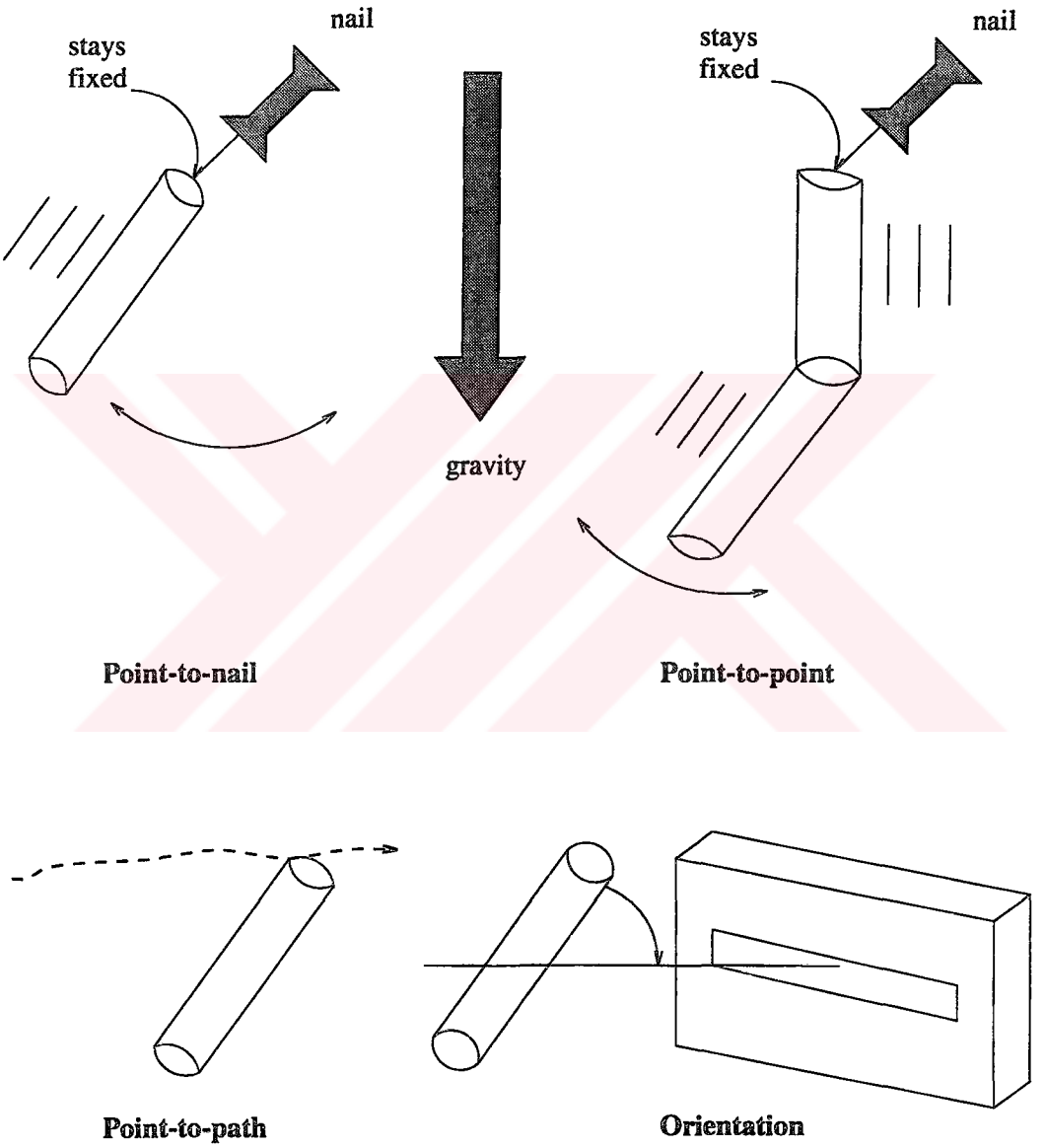


Figure 2.2: Examples of constraints.

Chapter 3

Collision Detection and Response

When several objects are involved in a computer animation at once, the problem of detecting and controlling object interactions is encountered. In such an animation, we may have more than one object moving around, or we may have impenetrable obstacles (such as walls) that do not move. When no special attention is paid to object interactions, the objects will sail through each other; this is usually not physically reasonable and produces a disconcerting visual effect. Whenever two objects attempt to interpenetrate each other (i.e., the surface of one object comes into contact with the surface of a second object), a *collision* is said to occur [5, 28].

The general requirement that arises then is an ability to *detect* collisions. Most animation systems at present do not provide even minimal collision detection, but require the animator to visually inspect the scene for object interactions and respond accordingly. This is time consuming and difficult even for keyframe or parameter systems where the user explicitly defines the motion; it is even worse for procedural and dynamical animation systems where the motion is generated by functions and laws defining their behavior. Although automatic collision detection is expensive to code and to run, it is a considerable convenience for animators, particularly when more automated methods of motion control, such as dynamics or behavioral control, are used.

The other related issue is the *response* to a collision once it is detected. Even keyframe systems could benefit from automatic suggestions about the motion

of objects immediately following a collision; animation systems using dynamic simulation inherently must respond to collisions automatically and realistically. Linear and angular momentum must be preserved, and surface friction and elasticity must be reasonable. Collision response algorithms can be classified into two groups:

- *Analytical methods*, which are limited to rigid and articulated objects, are typically faster. Analytical algorithms could be used within a kinematic animation system.

- *Penalty, or spring methods*, which introduce restoring spring forces when objects inter-penetrate each other. These methods are more general, working equally well for flexible, rigid, and articulated bodies. They could not be used within a kinematic animation system since they assume the ability to use the dynamics equations of motion to predict the motion immediately after impact.

In the following, collision detection and response algorithms are explained in detail.

3.1 Collision Detection Algorithms

Collision detection involves determining when objects penetrate each other. It is clearly an expensive operation, particularly when large numbers of objects with complex shapes are involved. Another issue is the ability to detect simultaneous collisions (multiple contacts at the same time). Furthermore, two objects may collide in such a way that a region, rather than a set of isolated points, may contact. Collision detection has been extensively pursued in the fields of CAD/CAM and robotics [1]. Some of the proposed algorithms solve the problem in more generality (and at higher cost), and some others do not easily produce the collision points and the normal directions necessary if collision response is to be calculated. Finally, many collision detection algorithms are quite intricate and must deal with assorted special cases.

3.1.1 Collision Detection for Flexible Surfaces

Flexible surfaces are generally modeled as a grid of points which are connected to form quadrilaterals or triangles. Collisions between surfaces are detected by testing for penetration of each vertex point through the planes of any triangle, or quadrilateral, not including that vertex (thus, self-intersection of surfaces is detected). Initially, the surfaces should be assumed disjoint. For each time step of animation, the positions of points at the beginning and at the end of the time step must be compared to see if any point went through a triangle, or quadrilateral, during that time step. If so, a collision has occurred. Consequently, the time complexity of such an algorithm is $O(nm)$ for n triangles, or quadrilaterals, and m points. A correct test must consider edges and triangles, as polyhedral objects can collide edge-on without any vertices being directly involved. However, in many cases merely testing points versus triangles produces acceptable results.

3.1.2 Collision Detection for Convex Polyhedra

The detection of collisions between solids (or closed surfaces) that have a distinguishable *inside* and *outside* could be treated somewhat differently. The objects' implicit (inside-outside) functions can be used for detecting collisions. We have used this approach in our animation system to detect the collisions of deformable models with impenetrable obstacles.

A very important problem with collision detection algorithms is that they may fail to detect a collision if one object moved entirely through another during a single time step. To minimize the frequency of occurrence of such a failure, time steps should be reduced; this is the case in dynamic animations.

3.2 Collision Response Algorithms

After detecting collisions between two objects, the objects should move in a physically correct manner. Collision response algorithms are developed to achieve this goal.

3.2.1 Analytical Collision Response Algorithms

An analytical solution for the collision of two arbitrary objects depends on the conservation of momentum during a collision, and results in a new angular and linear velocity for each body [5]. Thus, such a solution bypasses the question of collision forces and can be used independent of dynamic simulation, assuming information concerning the body's mass and mass distribution is provided. Analytical solutions are typically faster for strong collisions, because the solution need only be found once. However, for gentle collisions, such as a body resting quietly on top of another, spring solutions may be desirable. An important deficiency of analytical methods is that they are not suitable for flexible bodies.

3.2.2 Penalty (Spring) Methods for Collision Response

Penalty methods introduce restoring forces when objects inter-penetrate each other [37, 44]. These methods do not generate any contact surface between interacting objects but instead use the amount of local interpenetration to find a force that pushes the objects apart. They are computationally expensive for rigid bodies, give only approximate results, and may require different simulation conditions. These undesirable behaviors arise from the attempt to model infinite quantities (such as rigidity) with finite values. In particular, the differential equations that arise using penalty methods may be "stiff" and require an excessive number of time-steps during simulation to obtain accurate results. The advantages of penalty methods are that they are easy to implement, and are easily extensible to non-rigid bodies. We have used this approach in our animation system.

Chapter 4

Nonrigid Models

To animate nonrigid objects in a simulated physical environment, we should use the methods of elasticity theory. Elasticity theory provides methods to construct the differential equations that model the behavior of nonrigid curves, surfaces, and solids as a function of time. Real materials exhibit both elastic and inelastic behavior. Some materials undergo perfectly elastic deformations so that when the forces acting on the materials are removed, objects restore themselves to their natural shapes completely. However, there are other materials, such as cloth and paper, that restore themselves to their initial shapes slowly (or partially) upon removal of the forces that cause deformations.

To model elastic materials, physical properties such as tension and rigidity should be simulated. In this way, static shapes of a wide range of deformable objects, including string, rubber, cloth, paper, and flexible metals, can be modeled. Dynamics of these materials can be simulated by including physical properties, such as mass and damping. The simulation involves numerical solution of the partial differential equations that govern the evolving shape of the deformable object and its motion through space.

4.1 Formulation of Deformable Models

To simulate the dynamics of elastically deformable models, we use two existing formulations: the *primal formulation* [37] and the *hybrid formulation* [38].

4.1.1 Primal Formulation

Here, a deformable model is formulated by using the material coordinates of points in the body (denoted by Ω). For a solid body $\mathbf{u} = (u_1, u_2, u_3)$ (for a surface $\mathbf{u} = (u_1, u_2)$ and for a curve $\mathbf{u} = (u_1)$) denotes the material coordinates. The Euclidean 3-space positions of points in the body are given by time-varying vector-valued function $\mathbf{x}(\mathbf{u}, t) = [x_1(\mathbf{u}, t), x_2(\mathbf{u}, t), x_3(\mathbf{u}, t)]$. The body in its natural rest state is given by $\mathbf{x}^0(\mathbf{u}) = [x_1^0(\mathbf{u}), x_2^0(\mathbf{u}), x_3^0(\mathbf{u})]$ (Fig. 4.1). The equations of motion for a deformable model can be written in Lagrange's form as follows (this should hold for all $\mathbf{u}$ in the material domain Ω):

$$\frac{\partial}{\partial t}(\mu \frac{\partial \mathbf{x}}{\partial t}) + \gamma \frac{\partial \mathbf{x}}{\partial t} + \frac{\delta \varepsilon(\mathbf{x})}{\delta \mathbf{x}} = \mathbf{f}(\mathbf{x}, t), \tag{4.1}$$

where $\mu(\mathbf{u})$ is the mass density of the body at $\mathbf{u}$, $\gamma(\mathbf{u})$ is the damping density of the body at $\mathbf{u}$, $\mathbf{f}(\mathbf{x}, t)$ is the net externally applied force, and $\varepsilon(\mathbf{x})$ is the energy functional which measures the net instantaneous potential energy of the elastic deformation of the body.

The shape of a body is determined by the Euclidean distances between nearby points. As the body deforms, these distances change. Let $\mathbf{u}$ and $\mathbf{u} + d\mathbf{u}$ denote the material coordinates of two nearby points in the body. The distance between these points in the deformed body is given by

$$dl = \sum_{i,j} G_{ij} du_i du_j, \tag{4.2}$$

where the symmetric matrix

$$G_{ij}(\mathbf{x}(\mathbf{u})) = \frac{\partial \mathbf{x}}{\partial u_i} \cdot \frac{\partial \mathbf{x}}{\partial u_j} \tag{4.3}$$

is the metric tensor, which is a measure of deformations. (The dot indicates the scalar product of two vectors.)

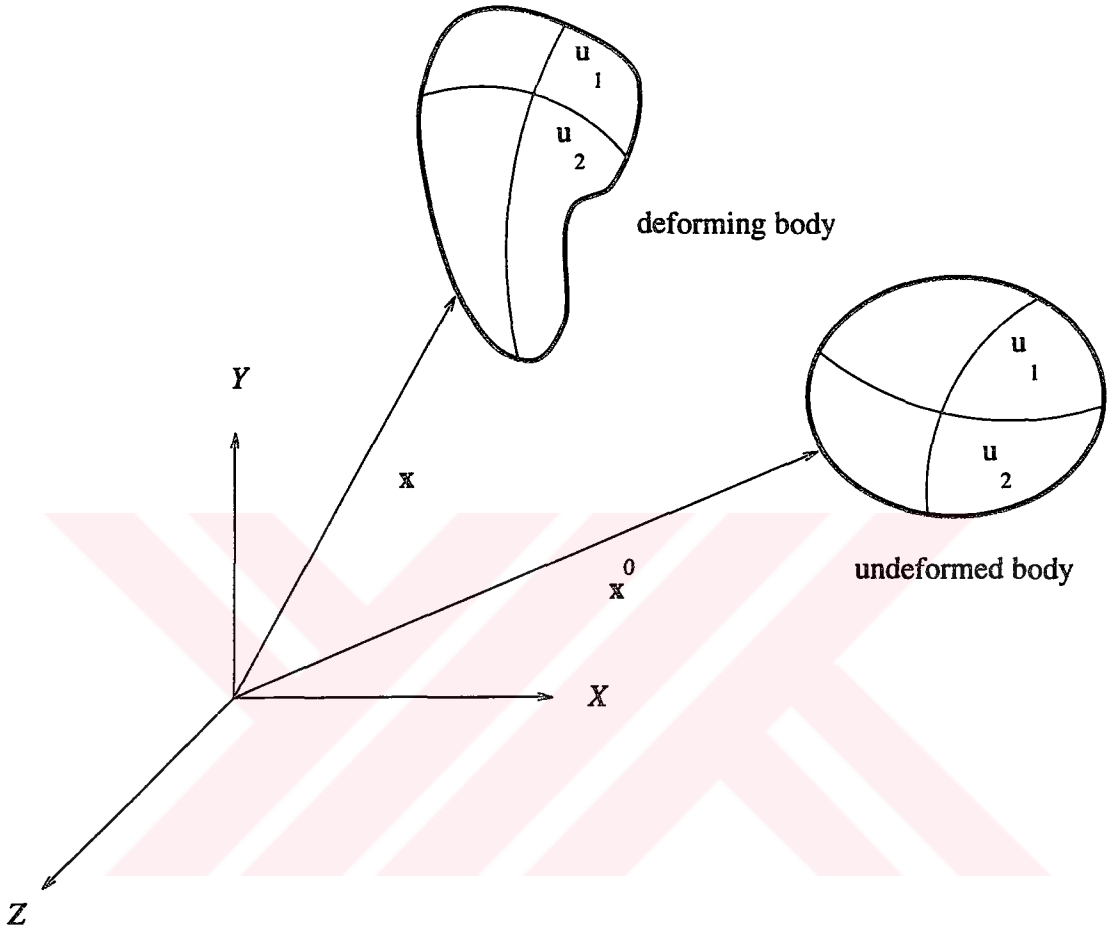


Figure 4.1: Geometric representation of a deformable body for primal formulation.

Two 3D solids have the same shape (differ only by a rigid body motion) if their 3×3 metric tensors are identical forms of $\mathbf{u} = [u_1, u_2, u_3]$, for all $\mathbf{u}$ in the material domain Ω. Two surfaces have the same shape if their metric tensors $\mathbf{G}$ as well as their curvature tensors $\mathbf{B}$ are identical forms of $\mathbf{u} = [u_1, u_2]$, for all $\mathbf{u}$ in the material domain Ω. The components of the curvature tensor are

$$B_{ij}(\mathbf{x}(\mathbf{u})) = \mathbf{n} \cdot \frac{\partial^2 \mathbf{x}}{\partial u_i \partial u_j}, \tag{4.4}$$

where $\mathbf{n} = [n_1, n_2, n_3]$ is the unit surface normal. Two space curves have the same shape if their arc length $s(\mathbf{x}(u))$, curvature $\kappa(\mathbf{x}(u))$, and torsion $\tau(\mathbf{x}(u))$

are identical forms of $\mathbf{u} = [u_1]$. (See [13] for a detailed discussion of these formulations.)

Using the above differential quantities, potential energies of deformation for use in Lagrange equations can be defined as the norm of the difference between the fundamental forms of the deformed body and those of the undeformed body. This norm measures the amount of deformation away from the natural shape so that the potential energy is zero when the body is in its natural shape and increases as the model gets increasingly deformed away from its natural shape.

If the fundamental forms associated with the natural shape are denoted by the superscript 0, then the strain energy for a curve can be defined as

$$\varepsilon(\mathbf{x}) = \int_\Omega w^1(s - s^0)^2 + w^2(\kappa - \kappa^0)^2 + w^3(\tau - \tau^0)^2 du \ , \tag{4.5}$$

where w^1, w^2, and w^3 are the coefficients for the curve, showing the amount of resistance to stretching, bending, and twisting, respectively. The strain energy for a surface can be defined in a similar way:

$$\varepsilon(\mathbf{x}) = \int_\Omega \|\mathbf{G} - \mathbf{G}^0\|^2_{\mathbf{w}^1} + \|\mathbf{B} - \mathbf{B}^0\|^2_{\mathbf{w}^2} du_1 du_2, \tag{4.6}$$

where the weighted matrix norms $\|\cdot\|_{\mathbf{w}^1}$ and $\|\cdot\|_{\mathbf{w}^2}$ involve the weighting functions $w^1_{ij}(u_1, u_2)$ and $w^2_{ij}(u_1, u_2)$. Analogously, a strain energy for a deformable solid is

$$\varepsilon(\mathbf{x}) = \int_\Omega \|\mathbf{G} - \mathbf{G}^0\|^2_{\mathbf{w}^1} du_1 du_2 du_3 \tag{4.7}$$

where the weighted matrix norm $\|\cdot\|_{\mathbf{w}^1}$ involves the weighting function $w^1_{ij}(u_1, u_2, u_3)$.

These energies denote the amount of energy to restore the deformed objects to their natural shapes. The net external force in Lagrange's equations is the sum of various types of external forces, such as gravitational force, spring forces, viscous forces, etc.

The weighting functions in the above energies ($w^1_{ij}(u_1, u_2)$ and $w^2_{ij}(u_1, u_2)$ for an elastic surface) determine the properties of the simulated deformable material. The weighting function $w^1_{ij}(u_1, u_2)$ determines surface tensions and sheer strengths which minimize the deviation of the surface's actual metric coefficients

G_{ij} from its natural coefficients G^0_{ij}. As w^1_{ij} is increased, the material becomes more resistant to length deformation, with w^1_{11} and w^1_{22} determining this resistance along u_1 and u_2, and $w^1_{12} = w^1_{21}$ determining the resistance to shear deformation. The functions $w^2_{ij}(u_1, u_2)$ control surface rigidities which act to minimize the deviation of the surface's actual curvature coefficients B_{ij} from its natural coefficients B^0_{ij}. As w^2_{ij} is increased, the material becomes more resistant to bending deformation, with w^2_{11} and w^2_{22} determining this resistance along u_1 and u_2, and $w^1_{12} = w^1_{21}$ determining the resistance to twist deformation. To simulate a strechy rubber sheet, for example, we make w^1_{ij} relatively small and set $w^2_{ij} = 0$. To simulate relatively stretch-resistant cloth, we increase the value of w^1_{ij}. To simulate paper, we make w^1_{ij} relatively large and we introduce a modest value for w^2_{ij}. Springy metal can be simulated by increasing the value of w^2_{ij} [37].

To create animation with deformable models, the differential equations of motion should be discretized and the system of linked ordinary differential equations obtained from the discretization process should be solved. For the discretization process, there are two basic ways. One way is to choose a finite number of points for a continuous model, and to replace derivatives by differences; this is called the *finite difference method.* The other way is to choose a finite number of functions, and to approximate the exact solution by a combination of those trial functions. If the functions are piecewise polynomials, then the pieces can be chosen to fit the geometry of the problem and a program can generate the polynomials. This method is called the *finite element method.* It allows a program to assemble the discrete problem and solve it. In our implementation, the finite difference method is chosen for discretization process since difference equations are easier to program and faster to run than a full finite element code.

4.1.2 Hybrid Formulation

In this formulation, a deformable body is represented as the sum of a reference component $\mathbf{r}(\mathbf{u}, t)$ and a deformation component $\mathbf{e}(\mathbf{u}, t)$ (Fig. 4.2). The positions of mass elements in the body relative to a body frame ϕ (whose origin coincides

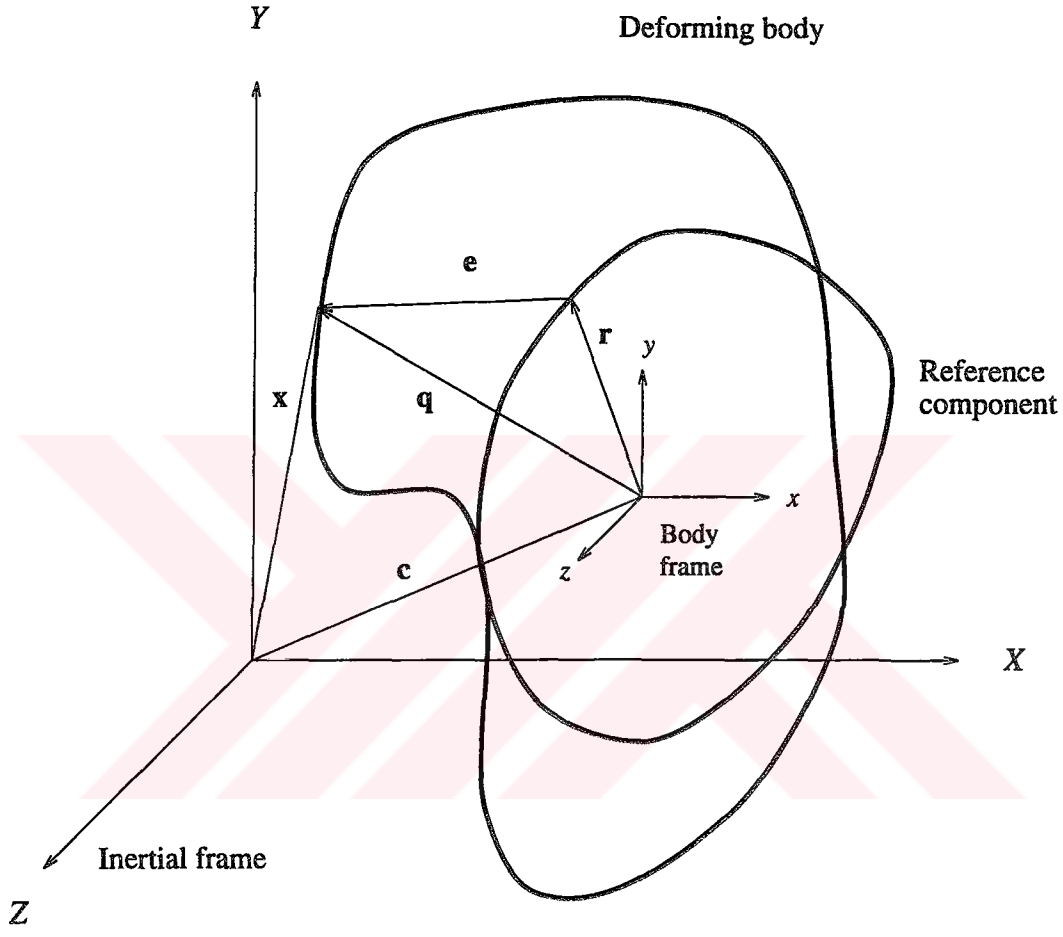


Figure 4.2: Geometric representation of deformable models for hybrid formulation.

with the body's center of mass and which should be evolved over time according to the rigid body dynamics to have a rigid body motion besides its elastic motion) are given by

$$\mathbf{q}(\mathbf{u},t) = \mathbf{r}(\mathbf{u},t) + \mathbf{e}(\mathbf{u},t). \tag{4.8}$$

In this formulation, deformations are measured with respect to the reference shape $\mathbf{r}$. Elastic deformations are represented by an energy $\varepsilon(\mathbf{e})$ that depends on the position of the reference (body) frame ϕ. The potential energy functional

for the hybrid formulation can be written by using simple, linear restoring forces resulting from controlled continuity spline energies [35] as follows:

$$\varepsilon(\mathbf{e}) = \frac{1}{2}\int_{\Omega}\sum_{m=0}^{p}\sum_{|j|=m}\frac{m!}{j_1!\ldots j_d!}w_j \mid \partial_j^m \mathbf{e} \mid^2, \tag{4.9}$$

where $j = (j_1, \ldots, j_d)$ is a multi-index with $\mid j \mid = j_1 + \cdots + j_d$, and

$$\partial_j^m = \frac{\partial m}{\partial u_1^{j_1}\ldots\partial u_d^{j_d}} \tag{4.10}$$

($d = 1$ for curves, $d = 2$ for surfaces, and $d = 3$ for solids). The energy density under the integral is a weighted sum of the magnitude of the deformation e and its partial derivatives with respect to material coordinates. The order p of the highest partial derivative included in the sum determines the order of smoothness of the deformation.

The weighting functions $w_j(\mathbf{u})$ in 4.9 control the properties of the deformable model over the body coordinates as in the primal formulation. In the case of surfaces ($d = 2$), the function w_{00} penalizes the total magnitude of the deformation; w_{10} and w_{01} penalize the magnitude of its first partial derivatives; w_{20}, w_{11} and w_{02} penalize the magnitude of its second partial derivatives [38].

4.2 Implementation of the Primal Formulation

To simulate the dynamics of a deformable surface, we should discretize the following expression for the elastic force, which is the approximation of the variational derivative of the expression in 4.6.

$$\epsilon(x) = \sum_{i,j=1}^{2} -\frac{\partial}{\partial u_i}\left(\alpha_{ij}\frac{\partial \mathbf{x}}{\partial u_j}\right) + \frac{\partial^2}{\partial u_i u_j}\left(\beta_{ij}\frac{\partial^2 \mathbf{x}}{\partial u_i u_j}\right), \tag{4.11}$$

where the functions $\alpha_{ij}(u, \mathbf{x})$ and $\beta_{ij}(u, \mathbf{x})$ determine the elastic properties of the material. The expressions for $\alpha_{ij}(u, \mathbf{x})$ and $\beta_{ij}(u, \mathbf{x})$ are as follows:

$$\alpha_{ij}(u, \mathbf{x}) = w_{ij}^1(u)(G_{ij} - G_{ij}^0) \tag{4.12}$$

$$\beta_{ij}(u, \mathbf{x}) = w_{ij}^2(u)(B_{ij} - B_{ij}^0) \tag{4.13}$$

The discretization is achieved by applying finite difference approximation method.

Since the body coordinates of the models are in the unit square domain, $\Omega = 0 \leq u_1, u_2 \leq 1$, we discretize this domain as a regular $(M+1) \times (N+1)$ discrete grid of nodes. Here, the inter-node spacings are $h_1 = 1/M$ and $h_2 = 1/N$ in the u_1 and u_2 directions, respectively. The nodes on the discrete model are indexed by integers $[m, n]$ where $0 \leq m \leq M$ and $0 \leq n \leq N$. Thus, if $\mathbf{x}$ (which is a continuous vector function $\mathbf{x}(u, t)$) is the 3D coordinates of the positions of points, then we discretize it by arrays of continuous time vector-valued nodal variables $\mathbf{x}_t[m, n] = \mathbf{x}(mh_1, nh_2, t)$.

Since the elastic force requires the approximations to the first and second derivatives of the nodal variables, we should first define them for the vector-valued position function $\mathbf{x}$.

The forward difference operators

$$D_1^+\mathbf{x}[m, n] = (\mathbf{x}[m+1, n] - \mathbf{x}[m, n])/h_1 \tag{4.14}$$

$$D_2^+\mathbf{x}[m, n] = (\mathbf{x}[m, n+1] - \mathbf{x}[m, n])/h_2 \tag{4.15}$$

and the backward difference operators

$$D_1^-\mathbf{x}[m, n] = (\mathbf{x}[m, n] - \mathbf{x}[m-1, n])/h_1 \tag{4.16}$$

$$D_2^-\mathbf{x}[m, n] = (\mathbf{x}[m, n] - \mathbf{x}[m, n-1])/h_2 \tag{4.17}$$

can be used to define the forward and backward cross difference operators

$$\begin{aligned} D_{12}^+\mathbf{x}[m, n] &= D_{21}^+\mathbf{x}[m, n] = D_1^+ D_2^+\mathbf{x}[m, n] = \\ &(\mathbf{x}[m+1, n+1] - \mathbf{x}[m+1, n] - \mathbf{x}[m, n+1] + \mathbf{x}[m, n])/h_1h_2 \end{aligned} \tag{4.18}$$

$$\begin{aligned} D_{12}^-\mathbf{x}[m, n] &= D_{21}^-\mathbf{x}[m, n] = D_1^- D_2^-\mathbf{x}[m, n] = \\ &(\mathbf{x}[m, n] - \mathbf{x}[m, n-1] - \mathbf{x}[m-1, n] + \mathbf{x}[m-1, n-1])/h_1h_2 \end{aligned} \tag{4.19}$$

and the central difference operators

$$D_{11}\mathbf{x}[m,n] = D_1^- D_1^+ \mathbf{x}[m,n] = (\mathbf{x}[m+1,n] - 2\mathbf{x}[m,n] + \mathbf{x}[m-1,n])/h_1^2 \quad (4.20)$$

$$D_{22}\mathbf{x}[m,n] = D_2^- D_2^+ \mathbf{x}[m,n] = (\mathbf{x}[m,n+1] - 2\mathbf{x}[m,n] + \mathbf{x}[m,n-1])/h_2^2 \quad (4.21)$$

Now, using the grid functions $\mathbf{x}[m,n], w_{ij}^1[m,n], w_{ij}^2[m,n]$ to represent their continuous counterparts, we can discretize 4.12 and 4.13 as follows:

$$a_{ij}[m,n] = w_{ij}^1[m,n](D_i^+\mathbf{x}[m,n] \cdot D_j^+\mathbf{x}[m,n] - G_{ij}^0[m,n]) \quad (4.22)$$

$$b_{ij}[m,n] = w_{ij}^2[m,n](\mathbf{n}[m,n] \cdot D_{ij}^+\mathbf{x}[m,n] - B_{ij}^0[m,n]), \quad (4.23)$$

where the superscript $(+)$ indicates that the forward cross difference operator is used when $i \neq j$, and

$$\mathbf{n}[m,n] = \frac{D_1^+\mathbf{x}[m,n] \times D_2^+\mathbf{x}[m,n]}{\mid D_1^+\mathbf{x}[m,n] \times D_2^+\mathbf{x}[m,n] \mid} \quad (4.24)$$

is the surface normal grid function. The elastic force in 4.11 can be approximated as

$$\epsilon[m,n] = \sum_{i,j=1}^{2} -D_i^-(a_{ij}D_j^+\mathbf{x}[m,n]) + D_{ij}^-(b_{ij}D_{ij}^+\mathbf{x}[m,n]). \quad (4.25)$$

To introduce free boundary conditions on the free edges of a surface, where the inner difference operators in 4.25 attempt to access nodal variables outside the discrete domain, we set the value of the inner difference operators to zero.

Expressing the grid functions $\mathbf{x}[m,n]$ and $\epsilon[m,n]$ as $\underline{\mathbf{x}}$ and $\underline{\epsilon}$ in grid vector notation, which denote the 3D positions of model points and elastic force for each model point stored in $(M+1) \times (N+1)$ vector for an $(M+1) \times (N+1)$ discrete grid of a deformable model, elastic force can be written in vector form as

$$\underline{\epsilon} = \mathbf{K}(\underline{\mathbf{x}}) \cdot \underline{\mathbf{x}}, \quad (4.26)$$

where $\mathbf{K}$ is an $(M+1)(N+1) \times (M+1)(N+1)$ matrix. $\mathbf{K}$ is a sparse and banded matrix. This becomes a major advantage when we solve the simultaneous system

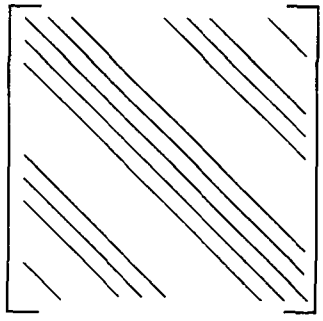


Figure 4.3: The band structure of the stiffness matrix **K**.

of second-order ordinary differential equations. The band structure of **K** is shown in Fig. 4.3.

Then, we should calculate the total external force for each point of the model. In order to achieve this, we should add the forces effecting a point, which are gravitational, viscous, collision, and constraint forces. The constraint forces are taken into account in the following way. When a constrained point tends to move, an opposite force for bringing it back to its original position is calculated and added to the total external force for that point. Each constrained point has an effect on the total external force for all points in the model depending on the difference between the body coordinates of the points. This coupling effect is taken into account automatically according to the elastic properties of the models. This method for calculating constraint forces gives good results for small time steps. For larger time steps, the model points make small oscillations since this approach corresponds to a corrective action.

The constrained points are specified by the user interactively. The system displays a grid specifying the body coordinates of each point existing in the model to be animated and the user selects the points to be constrained during the animation (that is, the points that will not move during the animation) using mouse buttons (Fig. 4.3). In other words, any point on a model could be constrained to a fixed location in space so that when the model is animated, the constrained points remain in their initial positions. The constraint force that connects a material point u_0 on a deformable model to a point $\mathbf{p}_0$ in space by a

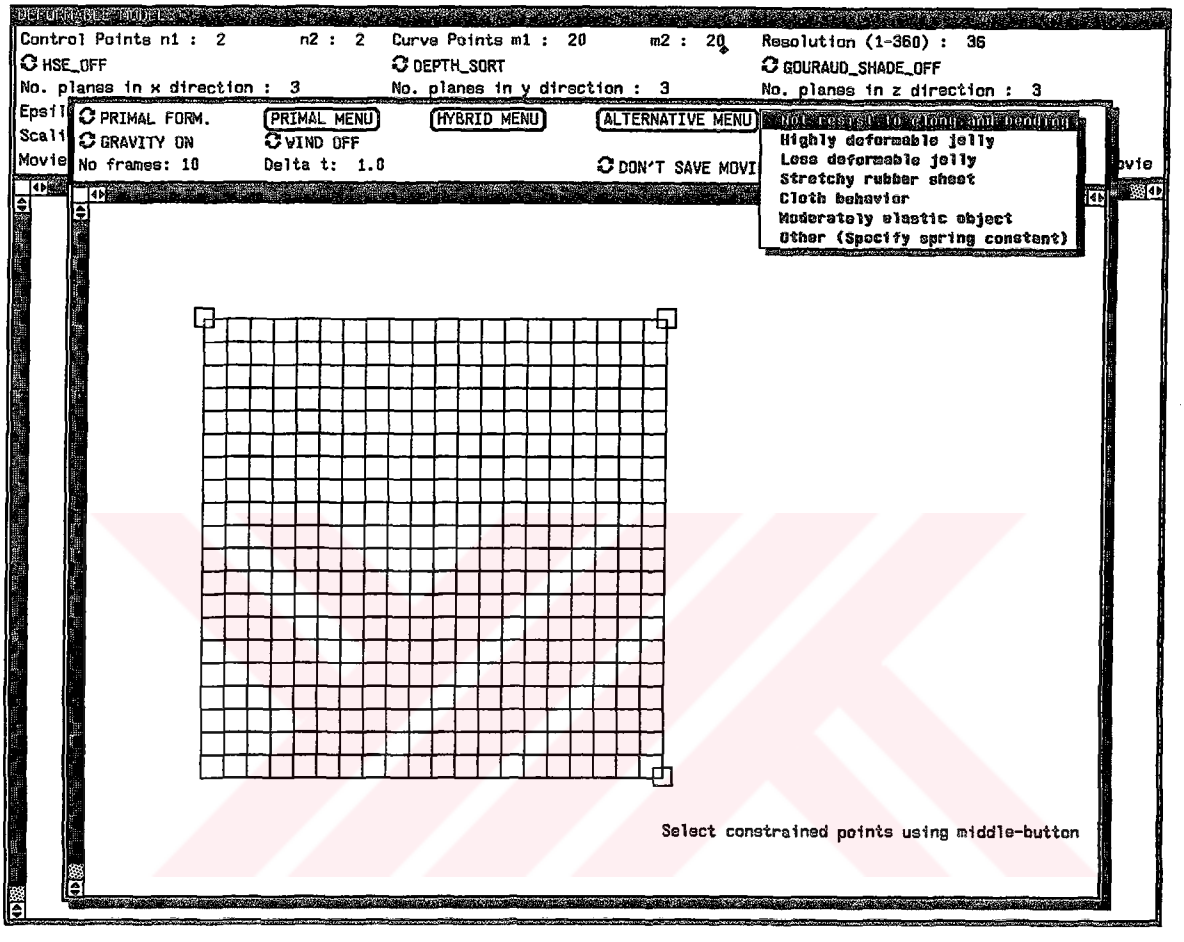


Figure 4.4: Screen dump during the specification of the parameters for an animation.

spring is

$$\mathbf{f}_s(u,t) = k(\mathbf{p}_0 - \mathbf{x}(u_0,t))\delta(u-u_0), \tag{4.27}$$

where k is the spring constant and δ is the unit delta function.

The forces due to the collision of deformable models with impenetrable obstacles are calculated using the obstacle's implicit (inside-outside) function. The obstacle exerts a repulsive force on the deformable model which can be calculated as a function of the obstacle's implicit function such that the force grows quickly if the model attempts to penetrate the obstacle. This is achieved by creating a potential energy function $c\ exp(f(\mathbf{x})/\xi)$ around each obstacle, where f is the

obstacle's implicit function, and c and ξ are constants determining the properties of the obstacle. In our system, the user can select different obstacles to exist in an animation sequence by the help of a menu. Ellipsoids, toroids, hyperboloids are possible choices for an obstacle. The repulsive force due to an impenetrable obstacle (expressed using the gradient ∇ of the potential energy function) is

$$\mathbf{f}_c(u,t) = -c\,((\nabla f(\mathbf{x})/\xi)\; exp(-f(\mathbf{x})/\xi) \cdot \mathbf{n})\;\mathbf{n}, \tag{4.28}$$

where $\mathbf{n}(u,t)$ is the unit surface normal vector of the deformable body's surface. The implicit functions for different obstacles are given in Appendix A.

The mass density $\mu(u_1,u_2)$ and the damping density $\gamma(u_1,u_2)$ are discretized as grid functions $\mu[m,n]$ and $\gamma[m,n]$. Let $\mathbf{M}$ be the *mass matrix*, an $(M+1)(N+1)\times(M+1)(N+1)$ diagonal matrix with the $\mu[m,n]$ variables as diagonal elements, and $\mathbf{C}$ be the *damping matrix* constructed similarly from $\gamma[m,n]$.

Then the Lagrange equations can be expressed in grid vector form by the simultaneous system of second-order ordinary differential equations

$$\mathbf{M}\frac{d^2\underline{\mathbf{x}}}{dt^2} + \mathbf{C}\frac{d\underline{\mathbf{x}}}{dt} + \mathbf{K}(\underline{\mathbf{x}})\underline{\mathbf{x}} = \underline{\mathbf{f}}(\underline{\mathbf{x}}), \tag{4.29}$$

where the net external force on the surface $\mathbf{f}(u_1,u_2)$ has been discretized into the grid vector $\underline{\mathbf{f}}$ which represents the grid function $\underline{\mathbf{f}}[m,n]$.

We integrate this system through time using a step-by-step procedure. Evaluating $\mathbf{K}(\underline{\mathbf{x}})$ at time $t+\Delta t$ and $\underline{\mathbf{f}}$ at t, and substituting the discrete time approximations

$$\frac{d^2\underline{\mathbf{x}}}{dt^2} \approx (\underline{\mathbf{x}}_{t+\Delta t} - 2\underline{\mathbf{x}}_t + \underline{\mathbf{x}}_{t-\Delta t})/\Delta t^2, \tag{4.30}$$

$$\frac{d\underline{\mathbf{x}}}{dt} \approx (\underline{\mathbf{x}}_{t+\Delta t} - \underline{\mathbf{x}}_{t-\Delta t})/2\Delta t \tag{4.31}$$

into 4.29, we obtain the semi-implicit integration procedure

$$\mathbf{A}_t\underline{\mathbf{x}}_{t+\Delta t} = \underline{\mathbf{g}}_t, \tag{4.32}$$

where the $(M+1)(N+1)\times(M+1)(N+1)$ matrix

$$\mathbf{A}_t(\underline{\mathbf{x}}_t) = \mathbf{K}(\underline{\mathbf{x}}_t) + \left(\frac{1}{\Delta t^2}\mathbf{M} + \frac{1}{2\Delta t}\mathbf{C}\right) \tag{4.33}$$

and the effective force vector

$$\underline{\mathbf{g}}_t = \underline{\mathbf{f}}_t + \left(\frac{1}{\Delta t^2}\mathbf{M} + \frac{1}{2\Delta t}\mathbf{C}\right)\underline{\mathbf{x}}_t + \left(\frac{1}{\Delta t}\mathbf{M} - \frac{1}{2}\mathbf{C}\right)\dot{\underline{\mathbf{x}}}_t, \tag{4.34}$$

with

$$\dot{\underline{\mathbf{x}}}_t = (\underline{\mathbf{x}}_t - \underline{\mathbf{x}}_{t-\Delta t})/\Delta t. \tag{4.35}$$

Applying the above semi-implicit procedure, we can obtain the dynamic solution from given initial conditions $\underline{\mathbf{x}}_0$ and $\dot{\underline{\mathbf{x}}}_0$ at $t = 0$. During each time step, we solve the sparse linear algebraic system in 4.32 for the instantaneous configuration $\underline{\mathbf{x}}_{t+\Delta t}$ using the preceding solution $\underline{\mathbf{x}}_t$ and $\dot{\underline{\mathbf{x}}}_t$ [37].

Implementation of the hybrid formulation follows the same steps described for the primal formulation. The elastic force for the hybrid formulation can be written as the variational derivative of the expression in 4.9 as follows:

$$\begin{aligned}\epsilon(\mathbf{e}) = w_{00}\mathbf{e} - \tfrac{\partial}{\partial u_1}\left(w_{10}\tfrac{\partial \mathbf{e}}{\partial u_1}\right) - \tfrac{\partial}{\partial u_2}\left(w_{01}\tfrac{\partial \mathbf{e}}{\partial u_2}\right) + \tfrac{\partial^2}{\partial u_1^2}\left(w_{20}\tfrac{\partial^2 \mathbf{e}}{\partial u_1^2}\right) + \\ 2\tfrac{\partial^2}{\partial u_1 \partial_2}\left(w_{11}\tfrac{\partial^2 \mathbf{e}}{\partial u_1 \partial u_2}\right) + \tfrac{\partial^2}{\partial u_2^2}\left(w_{02}\tfrac{\partial^2 \mathbf{e}}{\partial u_2^2}\right).\end{aligned} \tag{4.36}$$

Here, $\mathbf{u} = (u_1, u_2)$ are the surface's material coordinates.

The only difference between the primal formulation and the hybrid formulation is that the sparse, banded stiffness matrix $\mathbf{K}$ is constant in hybrid formulation. The equations of motion can be expressed in semidiscrete form by a system of coupled ordinary differential equations. The system contains two ordinary differential equations for the translational and rotational motion of the model as if all of its mass is concentrated at its center of mass, and a system of ordinary differential equations whose size is proportional to the size of the discrete model. These equations are solved in tandem for each time step with respect to the initial conditions given [38].

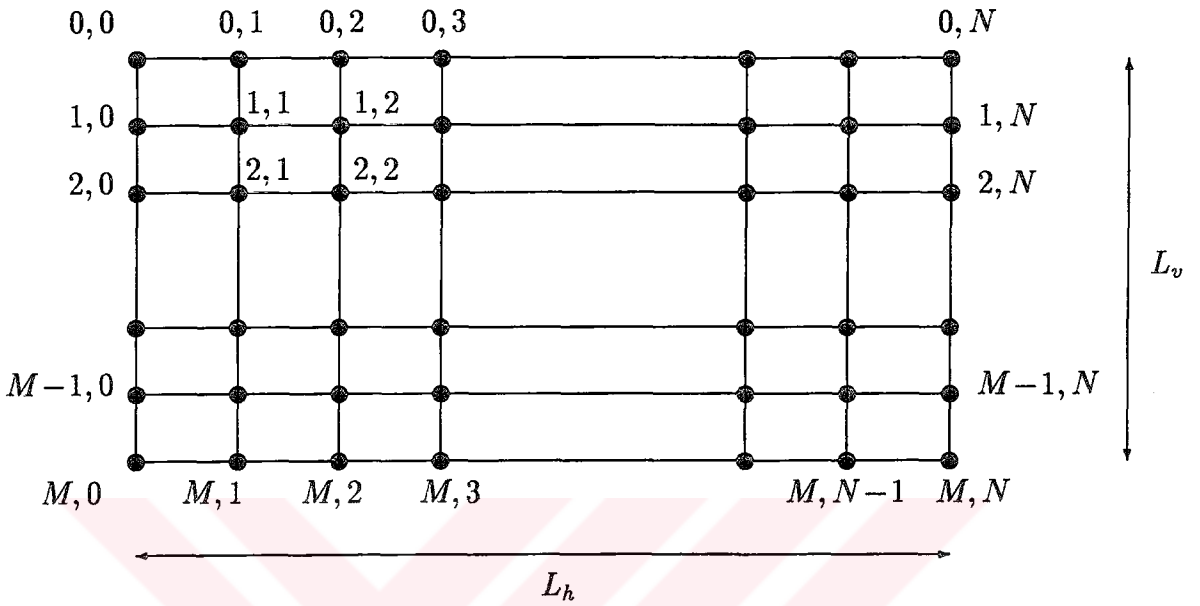


Figure 4.5: Numbering of the grid.

4.3 Spring Force Formulation for Deformable Models

In the previous sections, the *primal* and the *hybrid* formulations were presented and their implementation details were given. However, the formation of the stiffness matrix automatically is very difficult and sometimes it becomes impossible to solve the differential equations for animating the models because of the numerical ill-conditioning problems. In this section, a new formulation is presented. In this formulation, instead of forming the stiffness matrix automatically, elastic properties are represented as external spring forces. Although handling the elasticities using the stiffness matrix approach is elegant and the most suitable way, our approach is more effective and very fast.

The inter-node spacings on the grid are $h_1 = L_h/N$, $h_2 = L_v/M$ in the horizontal and vertical directions, respectively. Initially, we take $h_1 = h_2 = h$, for simplicity.

We can apply external forces to many of the grid points at the same time. One type of such external force can be the gravitational force. These external forces are known. Besides, if some of the grid points are constrained to the fixed positions in space, then there will be some unknown spring (constraint) forces at these points.

The line segments in the grid (Fig. 4.5) will correspond to the spring elements. According to the initial positions of the grid points, there will be some spring forces on the model.

The equations of motion for a deformable model (this should hold for all grid points) are

$$\mathbf{M}\frac{d^2}{dt^2}\mathbf{x} + \mathbf{C}\frac{d}{dt}\mathbf{x} + \mathbf{K}(\mathbf{x})\mathbf{x} = \mathbf{f}(\mathbf{x}) \tag{4.37}$$

We can take the elastic force expression as an external force $\mathbf{f}_K = \mathbf{K}(\mathbf{x})\mathbf{x}$, and take $\mathbf{f}_K$ to the right hand side of the equation (4.37). This new form of the equation will simplify the formulation procedure.

The position vector $\mathbf{x}$ for the model points is as follows (T denotes the transpose of a matrix):

$$\mathbf{x}^T = [\,\mathbf{x}_0^T\ \mathbf{x}_1^T\ \cdots\ \mathbf{x}_M^T\,] \tag{4.38}$$

where $\mathbf{x}_i$ represents all the position vectors of the grid points on the i-th row, and

$$\mathbf{x}_i^T = [\,\mathbf{x}_{i,0}^T\,\mathbf{x}_{i,1}^T\,\cdots\,\mathbf{x}_{i,N}^T\,] \tag{4.39}$$

where $\mathbf{x}_{i,j}$ is the position vector of the grid point (i,j) $(i = 0, 1, \cdots, M\,;\, j = 0, 1, \cdots, N)$.

In 4.37, $\mathbf{M}$ is the *mass matrix*, an $(M+1)(N+1) \times (M+1)(N+1)$ diagonal matrix which contains masses of the grid points as diagonal elements, and $\mathbf{C}$ is the *damping matrix*, an $(M+1)(N+1) \times (M+1)(N+1)$ diagonal matrix which contains dampers of the grid points as diagonal elements.

Note that 4.37 can be rewritten as

$$\mathbf{M}\frac{d^2}{dt^2}\mathbf{x} + \mathbf{C}\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) - \mathbf{f}_K(\mathbf{x}) \tag{4.40}$$

In this way, there will be no need for calculating the entries of the stiffness matrix. Instead of this, it is necessary to find the expressions for the column matrix $\mathbf{f}_K^T$ (external spring forces representing elasticities). The spring force vector can also be partitioned as

$$\mathbf{f}_K^T = [\mathbf{f}_0^T \; \mathbf{f}_1^T \; \cdots \; \mathbf{f}_M^T] \tag{4.41}$$

where the entries in the vector $\mathbf{f}_i^T = [\mathbf{f}_{i,0}^T \; \mathbf{f}_{i,1}^T \; \cdots \; \mathbf{f}_{i,N}^T]$ correspond to the spring forces acting at the grid points.

Using the discussion in [41] (pp. 359–362), the terminal equation of a two-terminal spring component of free length ℓ in three-dimensional space is given as

$$\mathbf{f}_K = k \left[(\mathbf{x}_1 - \mathbf{x}_2) - \ell \frac{\mathbf{x}_1 - \mathbf{x}_2}{||\mathbf{x}_1 - \mathbf{x}_2||} \right] \tag{4.42}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the position vectors of its terminal points. Note that calculation of the vector $(\mathbf{x}_1 - \mathbf{x}_2)$ is essential; it also appears in the second term of this expression. Equation 4.42 can be used to obtain expressions for the entries of $\mathbf{f}_K$ in 4.41.

For the grid points not on the boundaries, the elastic force is calculated by adding the spring forces applied to the grid point by its four neighbors.
If $i = 1, 2, \cdots, M-1$ and $j = 1, 2, \cdots, N-1$ then (see Fig. 4.6)

$$\left.\begin{aligned} \mathbf{f}_{ij} \;=\; & k \left[(\mathbf{x}_{i,j} - \mathbf{x}_{i,j-1}) - \ell \tfrac{\mathbf{x}_{i,j} - \mathbf{x}_{i,j-1}}{||\mathbf{x}_{i,j} - \mathbf{x}_{i,j-1}||} \right] \\ + \; & k \left[(\mathbf{x}_{i,j} - \mathbf{x}_{i-1,j}) - \ell \tfrac{\mathbf{x}_{i,j} - \mathbf{x}_{i-1,j}}{||\mathbf{x}_{i,j} - \mathbf{x}_{i-1,j}||} \right] \\ + \; & k \left[(\mathbf{x}_{i,j} - \mathbf{x}_{i,j+1}) - \ell \tfrac{\mathbf{x}_{i,j} - \mathbf{x}_{i,j+1}}{||\mathbf{x}_{i,j} - \mathbf{x}_{i,j+1}||} \right] \\ + \; & k \left[(\mathbf{x}_{i,j} - \mathbf{x}_{i+1,j}) - \ell \tfrac{\mathbf{x}_{i,j} - \mathbf{x}_{i+1,j}}{||\mathbf{x}_{i,j} - \mathbf{x}_{i+1,j}||} \right] \end{aligned}\right\} \tag{4.43}$$

For the grid points on the boundaries, three neighbors have an effect on the elastic force (Fig. 4.7). For the grid points on the corners, only two neighbors have an effect on the elastic force (Fig. 4.8). The elastic force expressions for the grid points on the boundaries and corners are given in Appendix B.

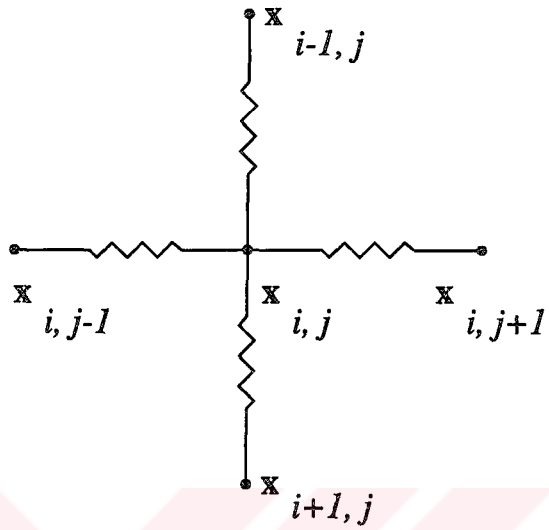


Figure 4.6: Interactions (couplings) between grid points (general case).

4.3.1 Implementation of the Spring Force Formulation

- Since the initial position vectors of the grid points are known, the vector $\mathbf{f}_K$ can be calculated from the external spring force equations.
- Then by solving the differential equation in 4.40 at the first step, next values of the position vectors of the grid points are determined.
- The next value of the vector $\mathbf{f}_K$ is calculated and the process is repeated.

As initial positions, we have $h \neq \ell$ in general. Therefore $\mathbf{f}_K \neq \mathbf{0}$. In other words, there will be some internal stresses in the system. If $h = \ell$, then $\mathbf{f}_K \equiv \mathbf{0}$. On the other hand, if $h_1 \neq h_2$, then $\mathbf{f}_K \neq \mathbf{0}$ initially (assuming that all the springs have the same lengths). We may select the lengths of the horizontal springs as $\ell_1 = h_1$ and the lengths of the vertical springs as $\ell_2 = h_2$. In this case, $\mathbf{f}_K = \mathbf{0}$ initially, and some of the ℓ factors will change to ℓ_1 and the remaining ones to ℓ_2 in the external spring force equations. Other modifications are also possible; e.g., on the spring coefficients (k).

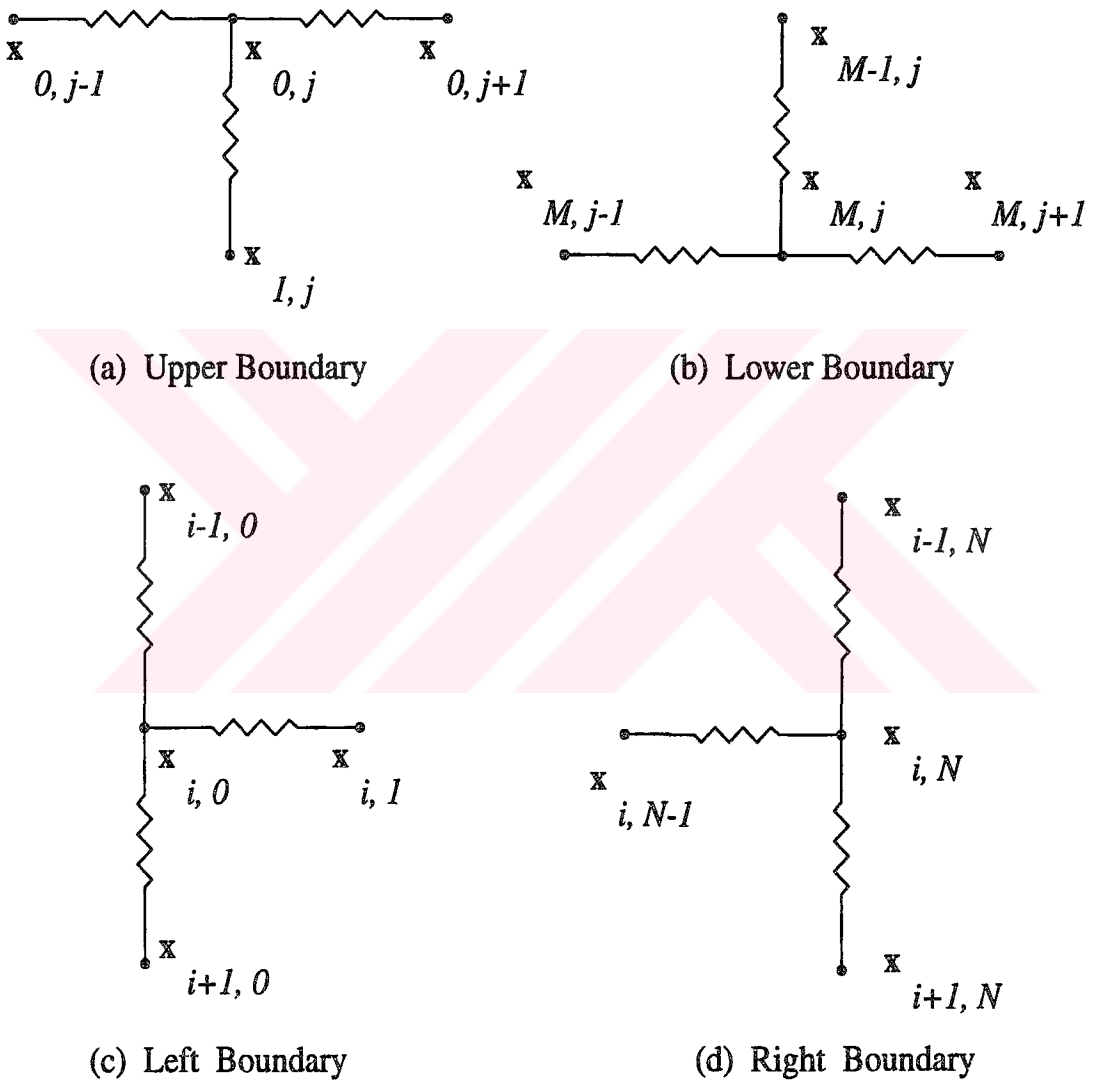


Figure 4.7: Interactions (couplings) between grid points (boundaries).

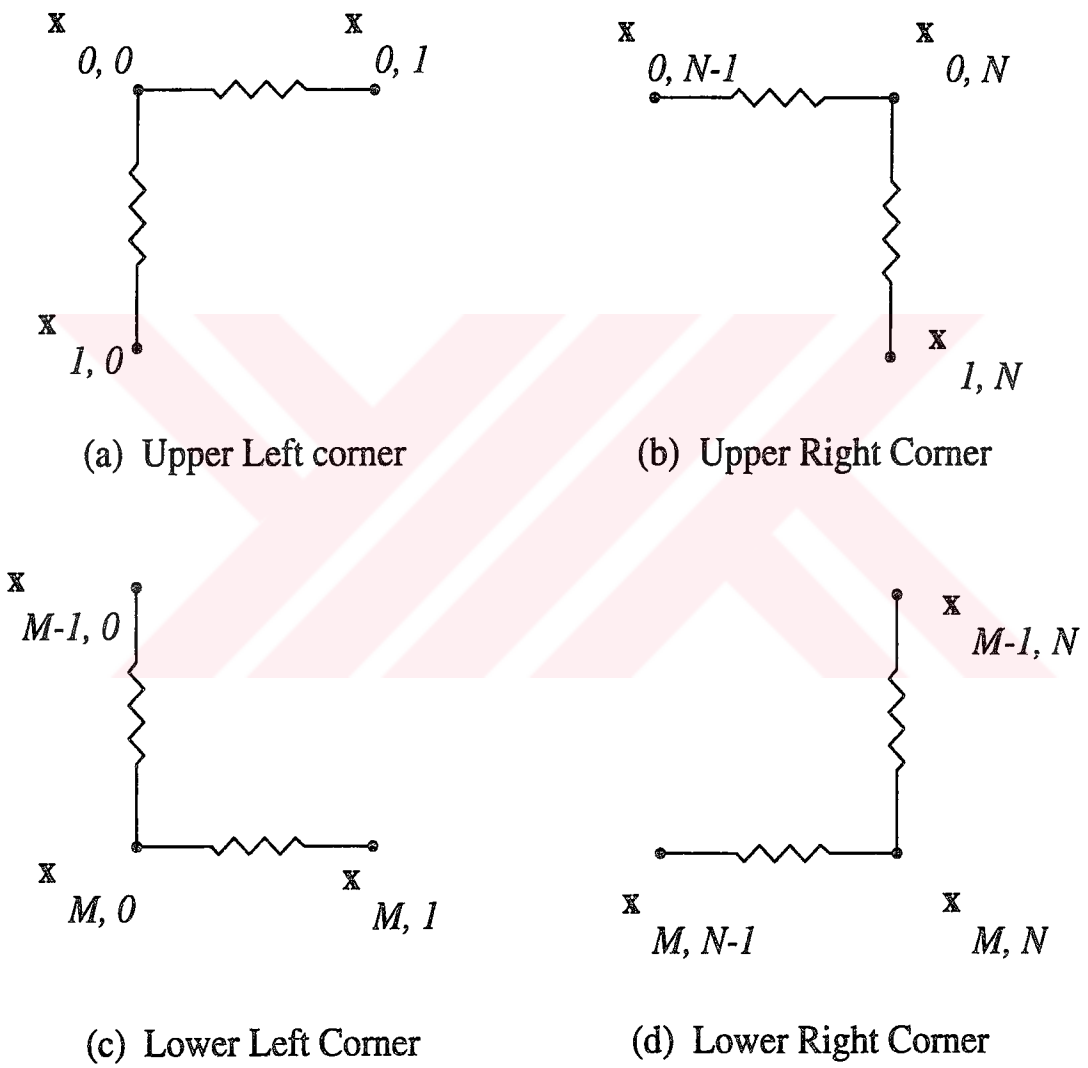


Figure 4.8: Interactions (couplings) between grid points (corner points).

Primal Formulation		
Number of Model Points	Preprocessing Time (millisec.)	Processing Time of One Frame (millisec.)
3 × 3	0	67
5 × 5	1017	83
10 × 10	2400	283
15 × 15	3950	600
20 × 20	6250	1183
25 × 25	9500	2000
30 × 30	14216	3883
35 × 35	21749	26182

Table 4.1: Preprocessing and processing times using the primal formulation.

4.4 Comparison of the Formulations

We have compared the processing times for generating an animation frame using different formulations. To compute processing times, simple Bézier surfaces, having similar elastic properties, are animated using different formulations. Tables 4.1, 4.2, and 4.3 give the processing times of the animations of the Bézier surfaces of different sizes, for the primal, hybrid, and spring force formulations, respectively. The processing times for each frame given in the tables include

- the time for calculating the external forces for each model point,
- the time for calculating the entries of the stiffness matrix*,
- the time for calculating the 3D positions of model points, and
- wireframe rendering time of the calculated frame.

The same information in the tables is also plotted as two graphs (Figs. 4.9, and 4.10) to compare the formulations in terms of the preprocessing times and

*This is for the primal formulation; for the hybrid formulation it is included in the preprocessing time. For the spring force formulation stiffness matrix is not formed; external spring forces between model points are calculated for each frame.

Hybrid Formulation		
Number of Model Points	Preprocessing Time (millisec.)	Processing Time of One Frame (millisec.)
3 × 3	1483	67
5 × 5	2483	83
10 × 10	3700	233
15 × 15	5500	400
20 × 20	8200	650
25 × 25	12233	1050
30 × 30	18749	2017
35 × 35	31932	20166

Table 4.2: Preprocessing and processing times using the hybrid formulation.

Spring Force Formulation		
Number of Model Points	Preprocessing Time (millisec.)	Processing Time of One Frame (millisec.)
3 × 3	0	67
5 × 5	950	83
10 × 10	2067	200
15 × 15	3500	433
20 × 20	5533	817
25 × 25	8383	1417
30 × 30	12350	2317
35 × 35	17766	3867

Table 4.3: Preprocessing and processing times using the spring force formulation.

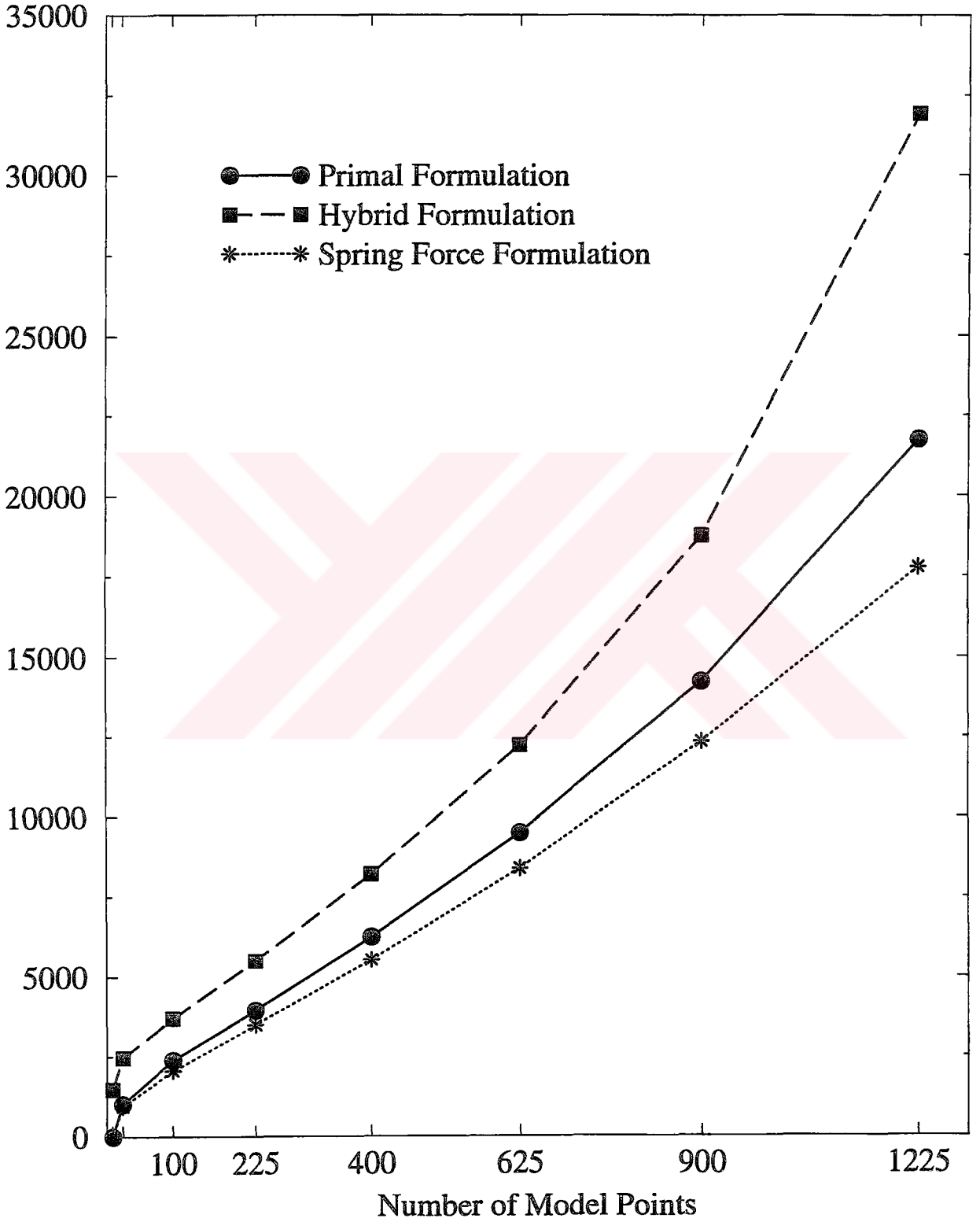


Figure 4.9: Preprocessing times using different formulations.

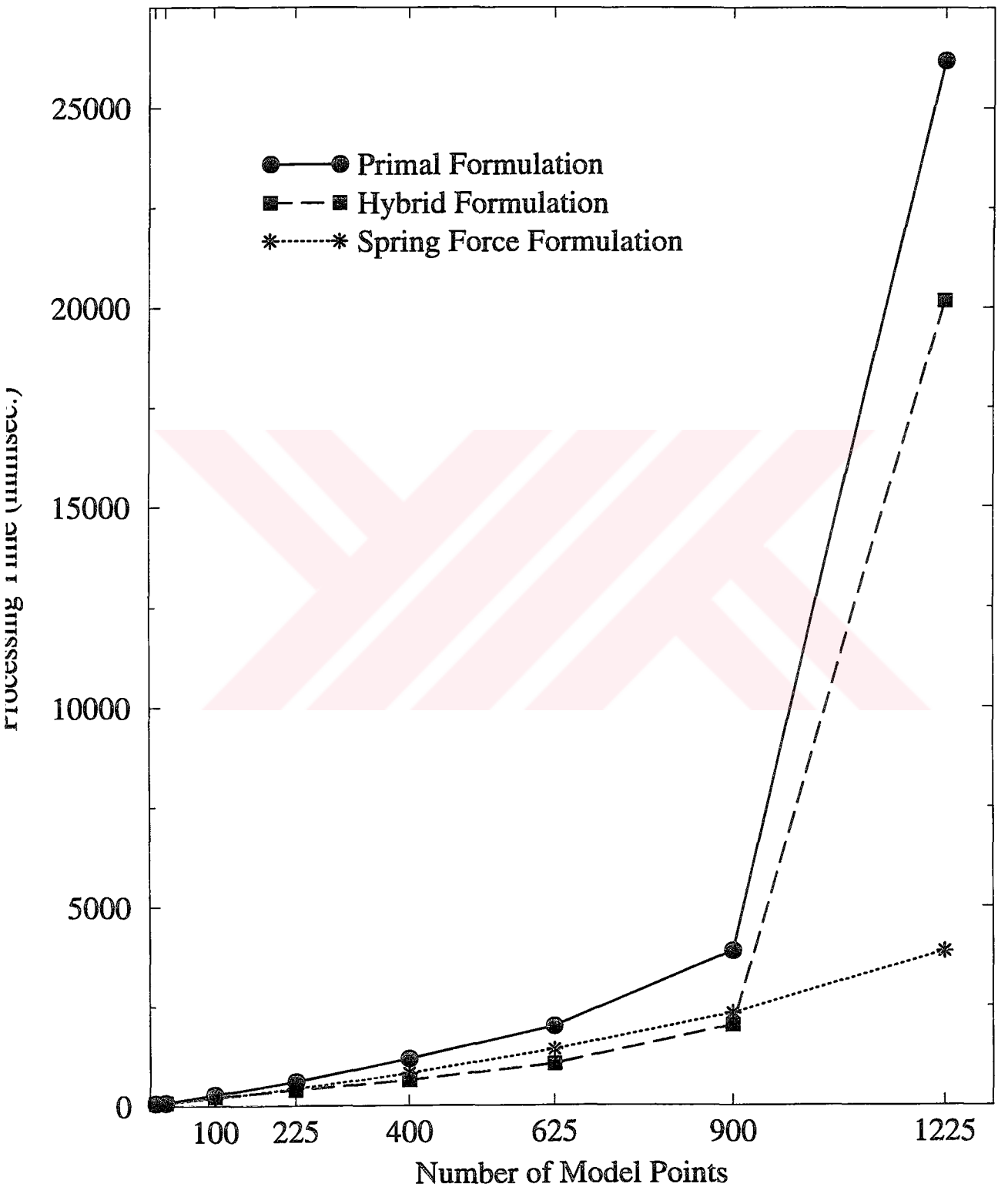


Figure 4.10: Processing times of one frame using different formulations.

processing times. Although it seems from the graphs that the hybrid formulation is superior to the primal formulation, they are complementing each other for different elasticity properties. The nonquadric energy functional in primal formulation causes a nonlinear elastic force associated with the deformable body to appear in the partial differential equations of motion. Nonlinearity results because the elastic force attempts to restore the shape of the deformed body to a rest shape. The advantage of nonlinear elasticity is that it is in principle the most accurate way to characterize the behavior of certain elastic phenomena. However, it can lead to serious practical difficulties in the numerical implementation of deformable models for animation. The hybrid formulation offers a practical advantage for fairly rigid models, whereas primal formulation becomes unpractical due to the nonquadric energy functional with increasing rigidity and complexity of the models.

The spring force formulation generates animation frames faster than the primal formulation. The hybrid formulation is superior to the spring force formulation for models having small size (less than 1000 model points). For larger models, spring force formulation is faster. The primal formulation is the most suitable formulation for highly rigid models. However, it is very difficult to form the stiffness matrix automatically. Spring force formulation bypasses this problem by modeling elasticities using external spring forces between model points.

An important advantage of the primal formulation over other formulations is that it is easier to establish an intuitive link between the weighting functions of the deformable models and the resulting elastic behavior. This is due to the nature of the weighting functions as explained in section 4.1.

4.5 Other Methods for the Animation of Nonrigid Models

The formulations that we have used employ continuous elasticity theory to model the shapes and motions of deformable models. There are other approaches to

model and animate deformable models. In this section, some of these approaches are explained.

Witkin et al. formulate a model for nonrigid dynamics based on global deformations with relatively few degrees of freedom [45]. This model is restricted to simple linear deformations that can be formulated by affine transformations. The use of deformations that are linear in the state of the system causes the constraint matrices in equations of motion to be constant. So, pre-inverting these matrices yields an enormous benefit in performance. In [30], Pentland and Williams describe the use of *modal analysis* to create simplified dynamic models of nonrigid objects. This approach breaks nonrigid dynamics down into the sum of independent vibration modes. This allows Pentland and Williams to achieve a level of control not possible with the massed equations normally used in dynamic simulation. This approach reduces the dimensionality and stiffness of the models by discarding high-frequency modes. High-frequency modes have no effect on linear deformations and rigid body dynamics. Both of these methods achieve large computational savings at the expense of limited deformations.

Another method, based on physics and optimization theory, uses mathematical constraint methods to create realistic animation of flexible models [32]. This method of Platt and Barr uses reaction constraints for fast computation of collisions of flexible models with polygonal models, and augmented Lagrangian constraints for creating animation effects, such as volume preserving squashing, and the molding of taffy-like substances. To model the flexible objects, the finite element method is used in Platt and Barr's work.

Thingvold and Cohen [40] define a model of elastic and plastic B-spline surfaces which supports both animation and design operations. They develope "refinement" operations for spring and hinge B-spline models which are compatible with the physics and the mathematics of B-spline models. Their model can be viewed as a continuous physical representation of a physical model rather than the more standard discretized geometry point mass models. The motion of their models is controlled by assigning different physical properties and kinematic constraints on various portions of the surface.

In [44], an approach to imposing and solving geometric constraints on parameterized models is given. This approach is applicable to animation as well as model construction. Constraints are expressed as energy functions, and constraint satisfaction is achieved by solving energy minimization problems. Although this approach is not as realistic as the above three approaches because of the lack of physics, it is simple and general.

Metaxas and Terzopoulos [27] propose an approach for creating dynamic solid models capable of realistic physical behaviors starting from common solid primitives such as spheres, cylinders, cones, and superquadrics. Such primitives can "deform" kinematically in simple ways. For example, a cylinder deforms as its radius (or height) is changed. To gain additional modeling power they allow the primitives to undergo parameterized global deformations (bends, tapers, twists, shears, etc.). Even though their models' kinematic behavior is stylized by the particular solid primitives used, the models behave in a physically correct way with prescribed mass distributions and elasticities. Metaxas and Terzopoulos also proposed efficient constraint methods for connecting the dynamic primitives together to make articulated models.

Breen et al. [10] propose a physically-based model and a simulation methodology, which when used together are able to reproduce many of the attributes of the characteristic behavior of cloth. Their model utilizes a microscopic particle representation that directly treats the mechanical constraints between the threads in a woven material rather than a macroscopic continuum approximation. Their simulation technique is hybrid, employing force methods for gross movement of the cloth and energy methods to enforce constraints within the material. Although limited only to cloth object behavior in scope, their approach is very realistic since a microscopic particle representation is utilized.

There are other physically-based models of flexible objects which are concerned only with the static shape. Weil [42] propose a geometric approach for interpolating surfaces to produce draped "cloth" effects. The cloths synthesized with his model contain folds and appear very realistic. The cloth is assumed to be rectangular, and is represented as a grid of three-dimensional coordinates.

He uses the catenary curves to define the positioning of the points along a given thread.

Feynman [14] described a technique for modeling the appearance of cloth. His computational framework minimizes energy functions defined over a grid of points. Feynman derives his functions from the theory of elasticity and from the assumption that cloth is a flexible shell.

Chapter 5

Simulation Examples

This chapter gives some simulation examples produced by our animation system. The examples show the salient features that are present in the animation system.

5.1 Simulation Examples Using Primal and Hybrid Formulations

We have implemented both primal and hybrid formulation in our system so that the user can interactively select between them. In this way, the primal formulation can be employed for highly nonrigid models, and the hybrid formulation can be employed for highly rigid models.

In Fig. 5.1, we have used the primal formulation. The material properties are adjusted to simulate a membrane not resistant to elongation or contraction, and not resistant to bending. In this example, a discrete model of size 30×30 is constrained from its four corners and falls by the effect of the gravitational force.

In Fig. 5.2, a flat surface which has the same material properties as the surface in Fig. 5.1 and constrained from its center of mass falls by the effect of the gravitational force.

In Fig. 5.3, we have used the hybrid formulation and set the material properties to simulate a paper. The model is constrained from three corners and a

downward force is exerted on it.

In Fig. 5.4, an elastic model not resistant to elongation or contraction and not resistant to bending falls on an impenetrable obstacle which is an ellipsoid. The deformable model takes the shape of the obstacle when it collides with it. To get better results in collision simulations, we should either take a very small time step or use adaptive time stepping. Otherwise, we may detect collisions very late, namely after the model points penetrate the obstacle.

In Fig. 5.5, a flat surface not resistant to elongation or contraction and not resistant to bending falls on an impenetrable obstacle which is a toroid. The surface takes the shape of the toroid when it collides with it.

Shaded versions of these simulation are given in Figs. 5.6, ... , 5.10.

5.2 Simulation Examples Using Spring Force Formulation

In the spring force formulation, by setting the stiffness constants to different values it is possible to obtain different elastic properties. In Figs. 5.11, 5.12, and 5.13, a surface is assigned different elastic properties and constrained from different points. Each part of the figures shows the form of the surface after a specific number of animation frames. Initially, all the surfaces are flat.

In Fig. 5.14, a strechy sheet constrained from its three corners falls with the effect of gravity. In Fig. 5.15, a strechy sheet constrained from its four corners falls. In Fig. 5.16, a strechy sheet constrained from its center of mass falls. In Fig. 5.17, a piece of cloth collides with an impenetrable obstacle, which is an ellipsoid. In Fig. 5.18, a strechy sheet drops over a toroid. In Fig. 5.19, an elastic surface drops over a toroid with a very small hole. In Fig. 5.20, an elastic surface passes through a toroid.

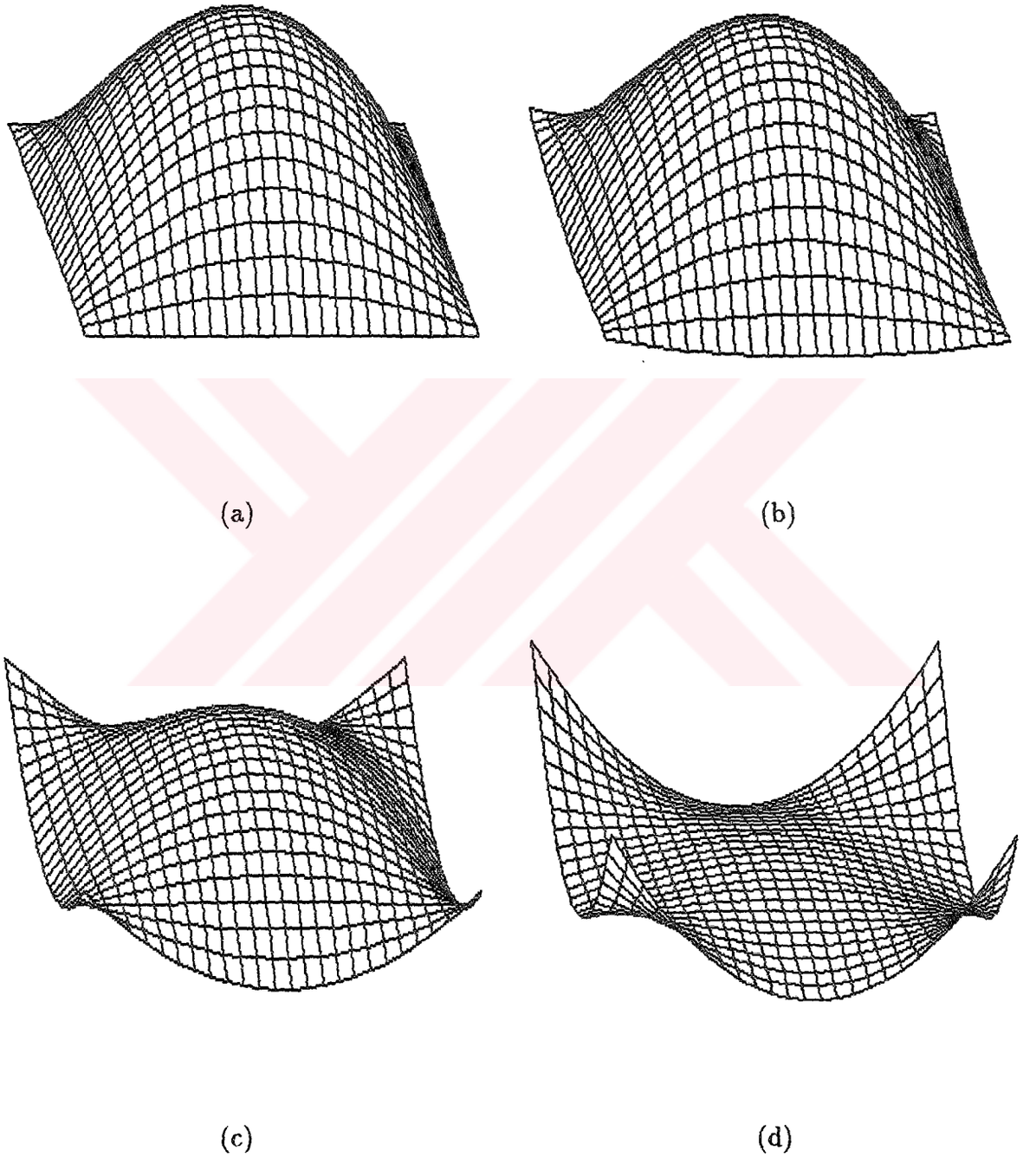


Figure 5.1: A highly nonrigid surface, constrained from its four corners, falls.

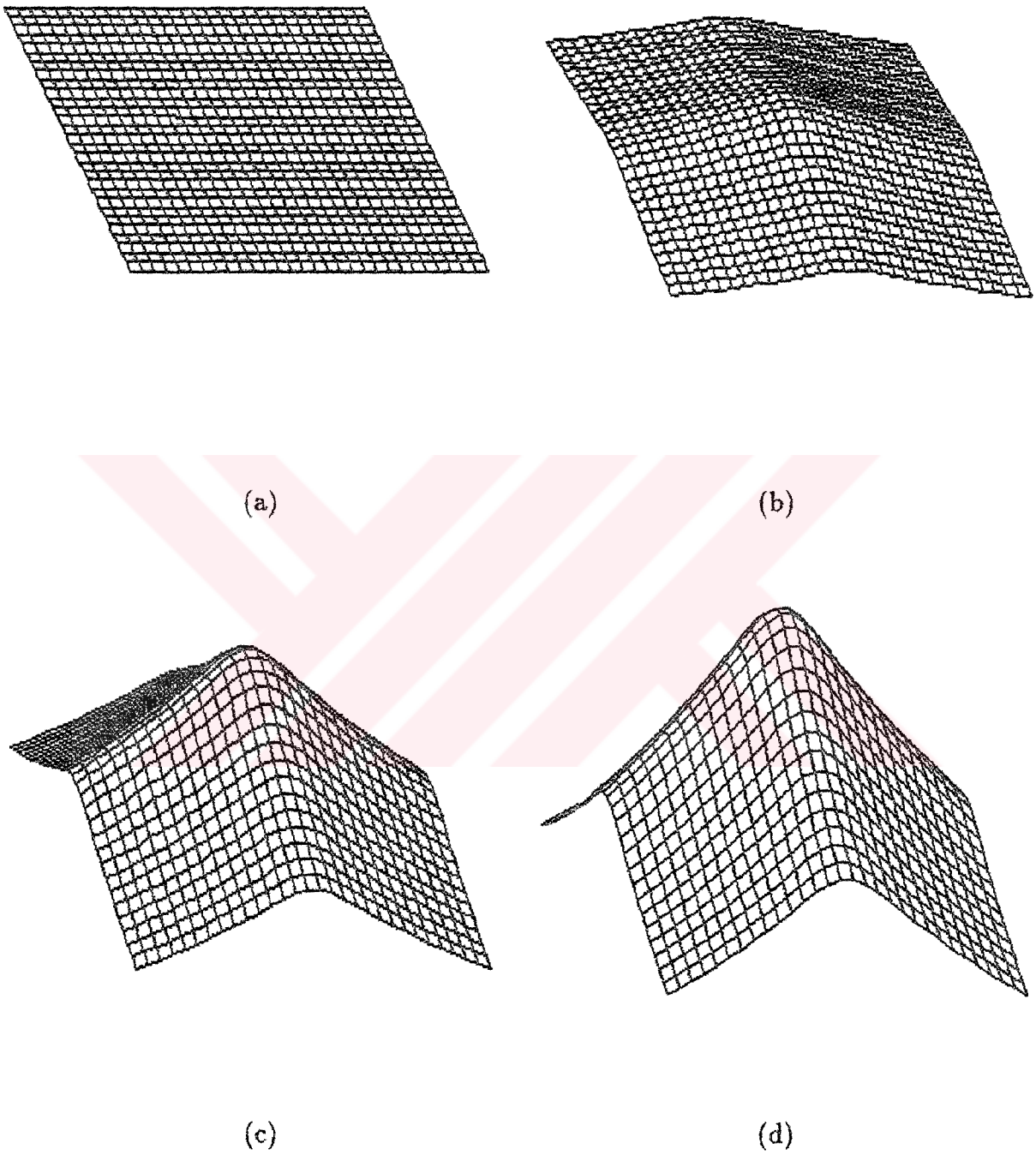
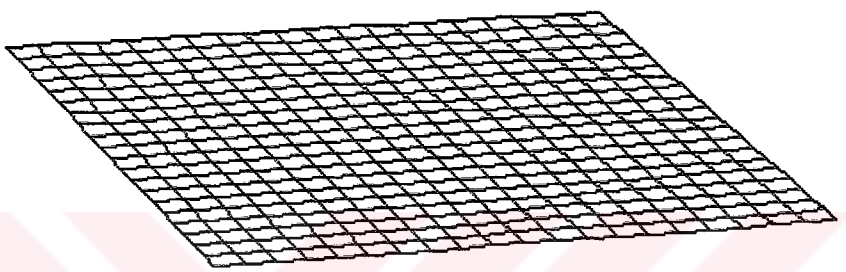
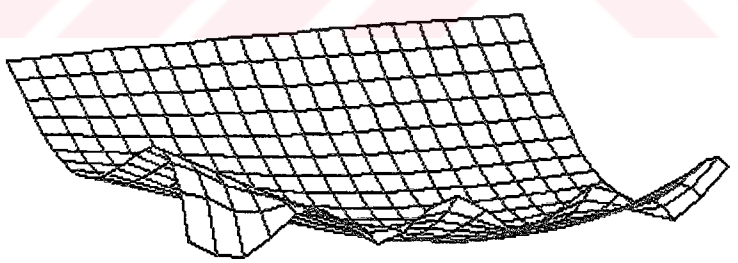


Figure 5.2: A highly nonrigid surface, constrained from its center of mass, falls.

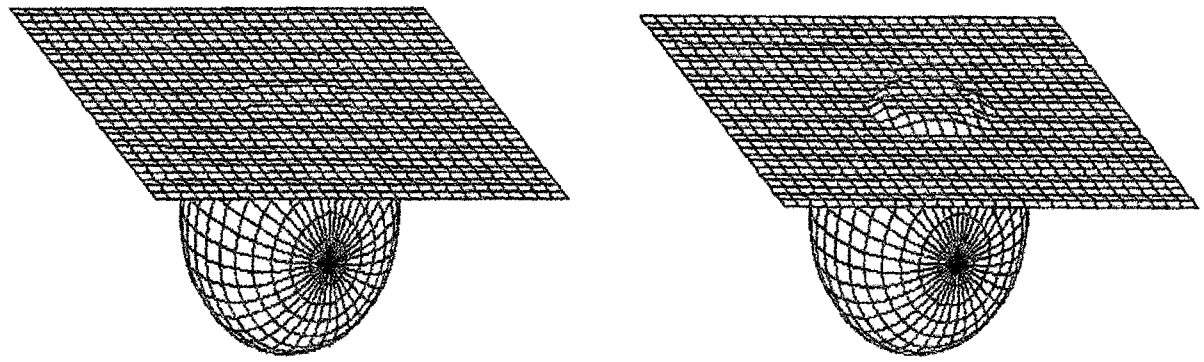


(a)



(b)

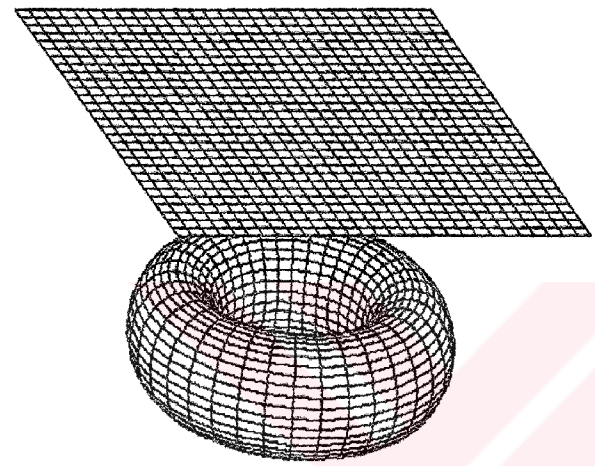
Figure 5.3: A piece of paper, constrained from three corners, applied a downward force.



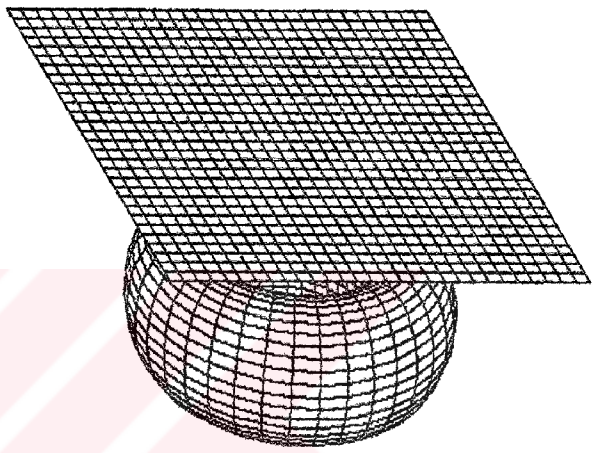
(a) (b)

(c) (d)

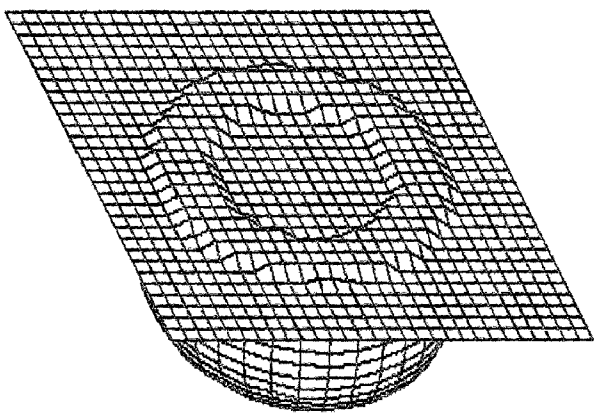
Figure 5.4: A highly nonrigid surface collides with an ellipsoid.



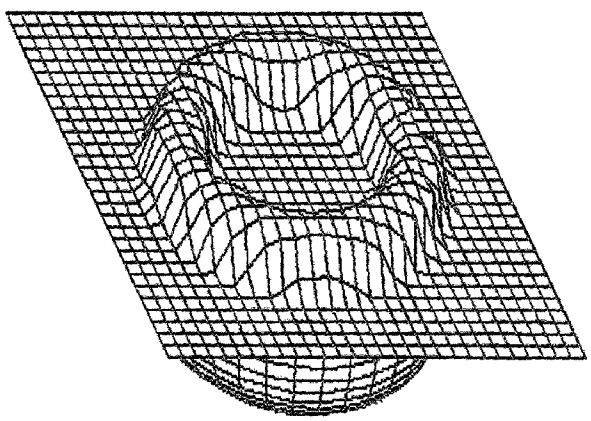
(a)



(b)



(c)



(d)

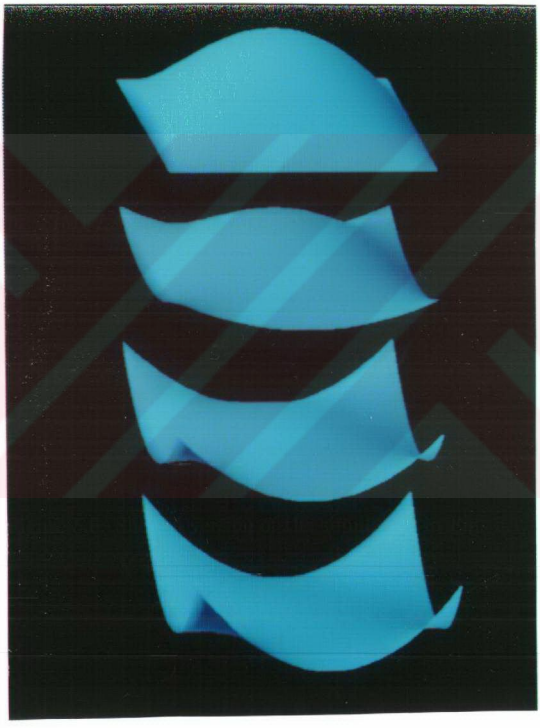


Figure 5.6: Shaded version of the simulation in Fig. 5.1.

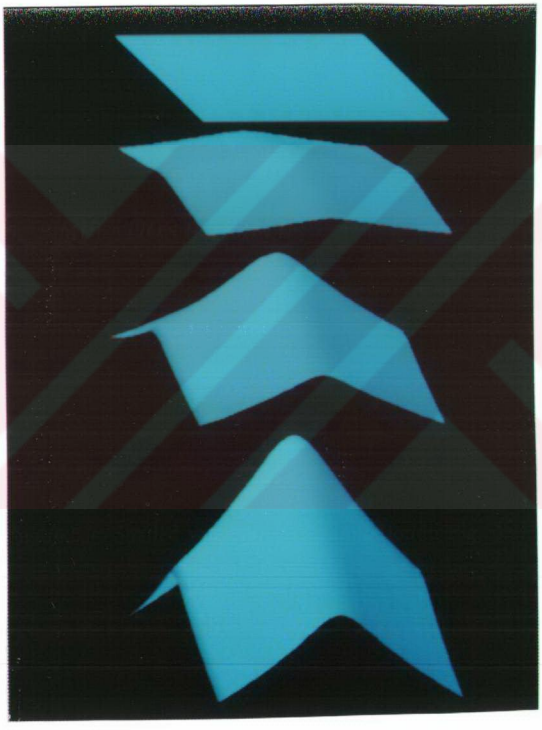


Figure 5.7: Shaded version of the simulation in Fig. 5.2.

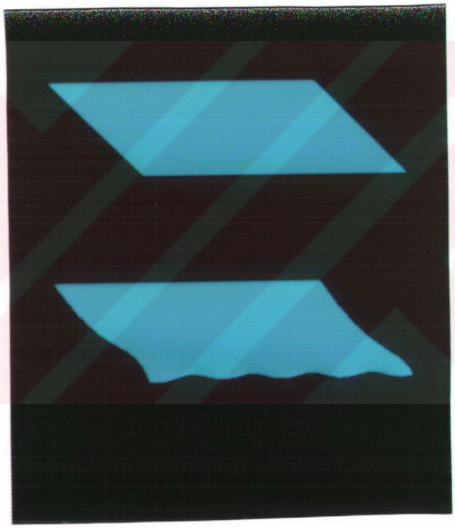


Figure 5.8: Shaded version of the simulation in Fig. 5.3.

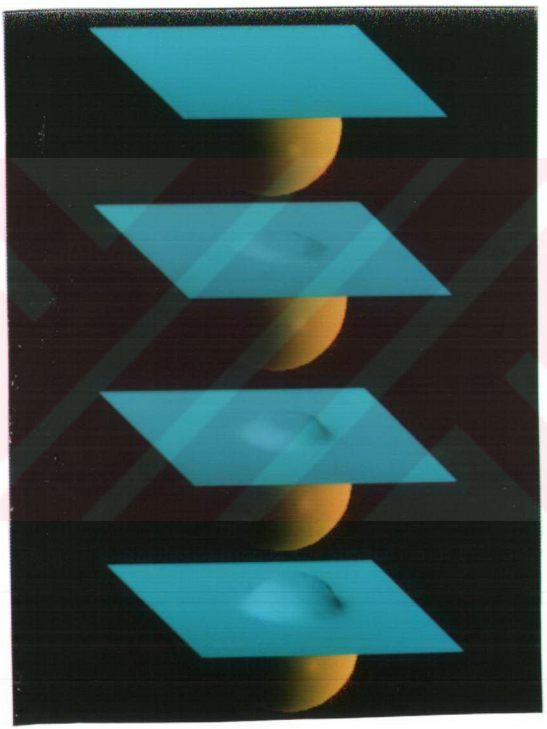


Figure 5.9: Shaded version of the simulation in Fig. 5.4.

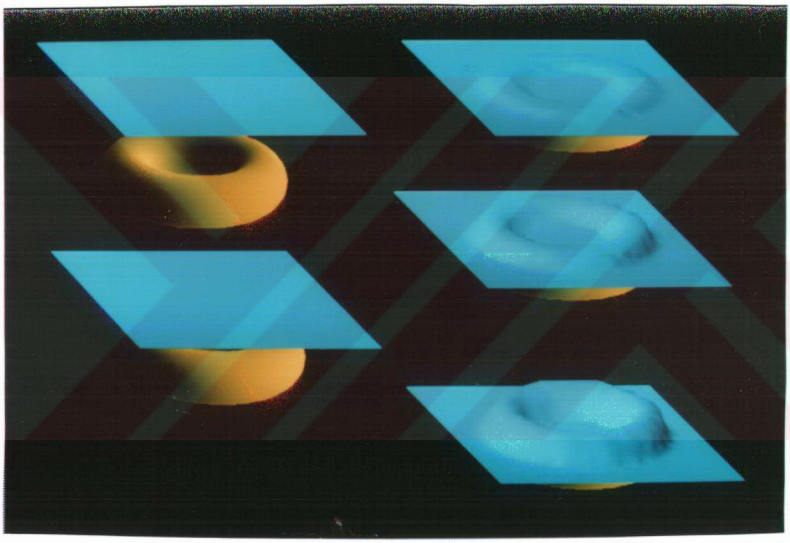
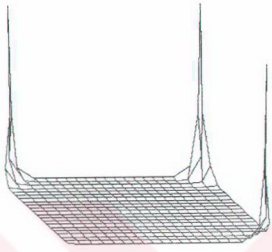
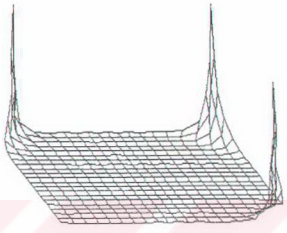


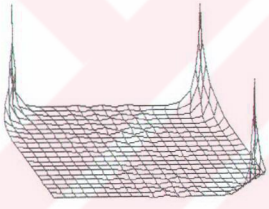
Figure 5.10: Shaded version of the simulation in Fig. 5.5.



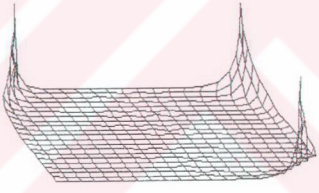
(a) k=1



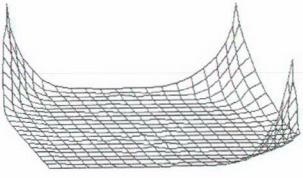
(b) k=5



(c) k=10

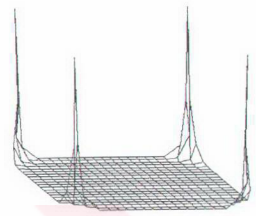


(d) k=20

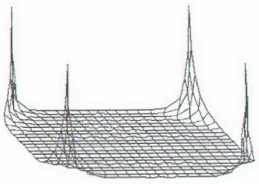


(e) k=30

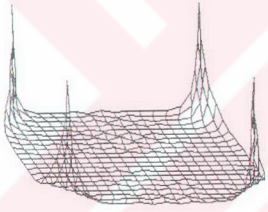
Figure 5.11: Different elastic surfaces, constrained from three corners, fall.



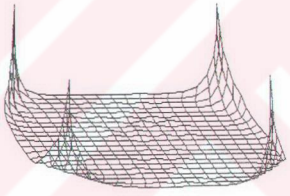
(a) k=1



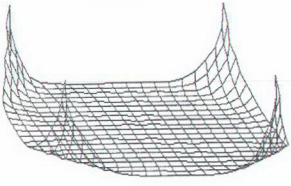
(b) k=5



(c) k=10

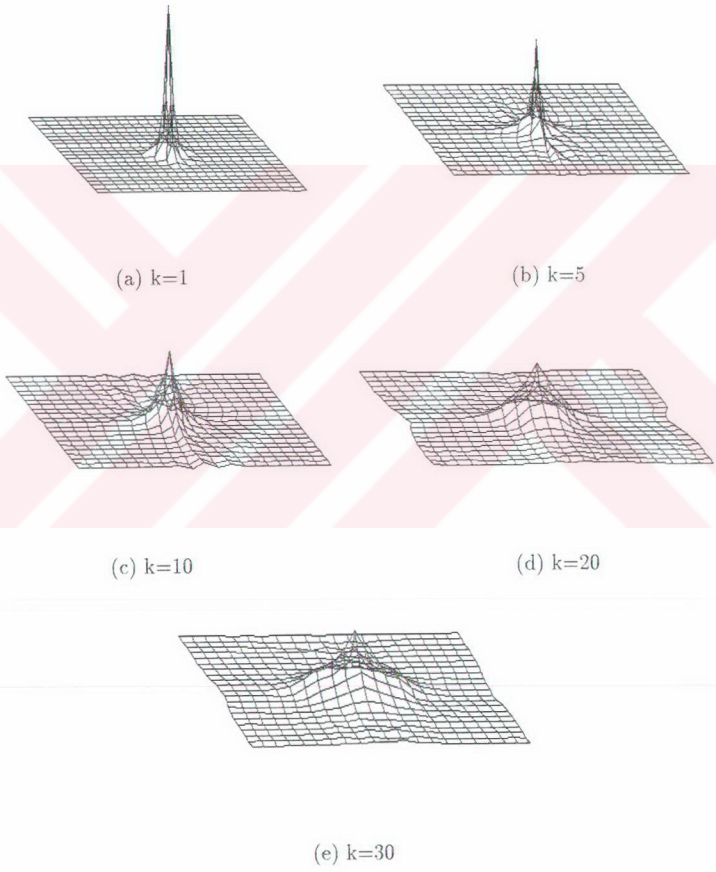


(d) k=20



(e) k=30

Figure 5.12: Different elastic surfaces, constrained from four corners, fall.



(a) k=1

(b) k=5

(c) k=10

(d) k=20

(e) k=30

Figure 5.13: Different elastic surfaces, constrained from the center of mass, fall.

Figure 5.14: A strechy sheet, constrained from its three corners, falls.

Figure 5.15: A strechy sheet, constrained from its four corners, falls.

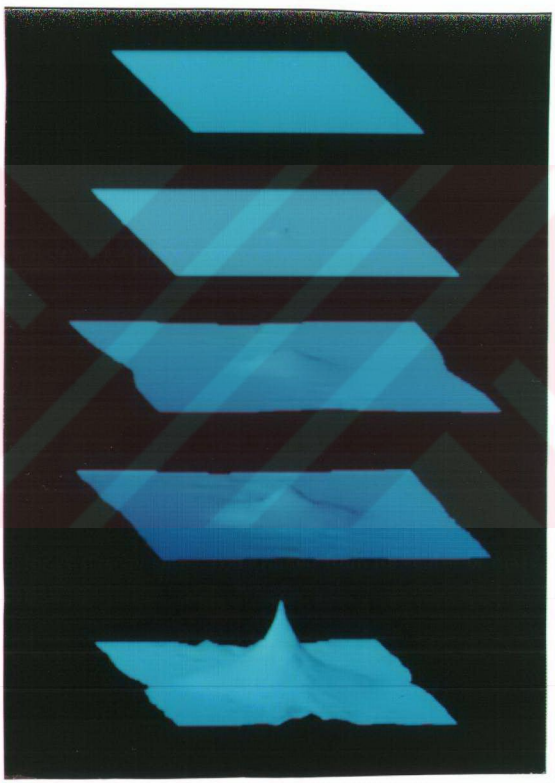


Figure 5.16: A strechy sheet, constrained from its center of mass, falls.

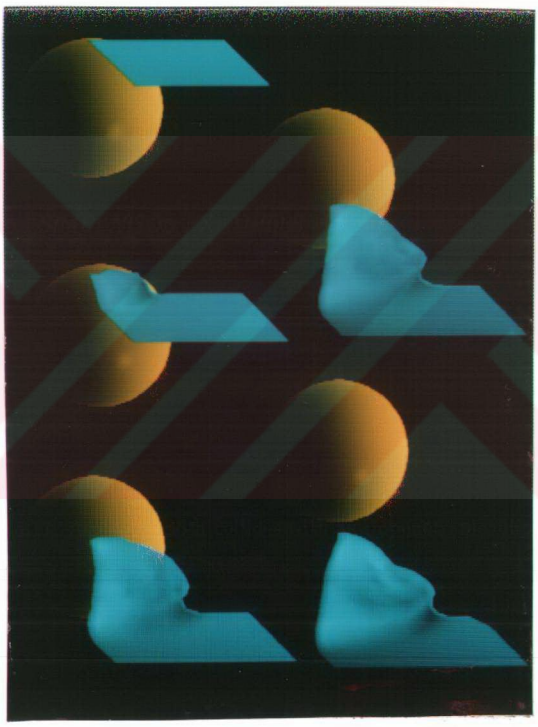


Figure 5.17: A piece a cloth collides with an impenetrable ellipsoid.

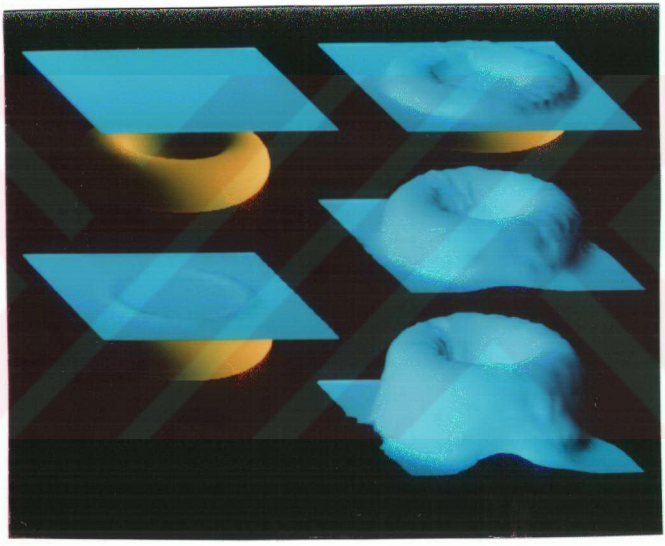


Figure 5.18: A strechy sheet drops over a toroid.

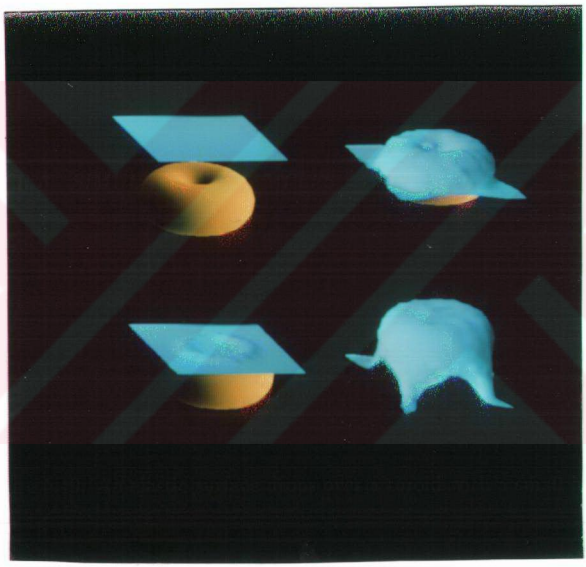


Figure 5.19: An elastic surface drops over a toroid with a small hole.

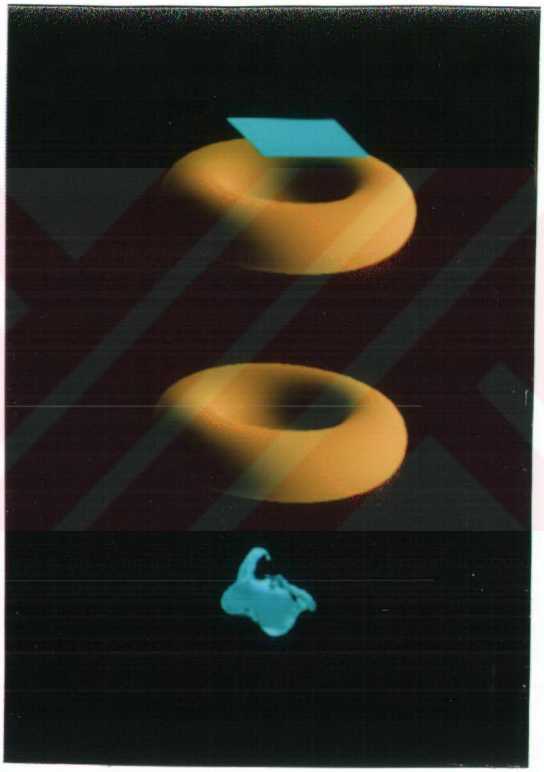


Figure 5.20: A small elastic surface passes through a toroid.

Chapter 6

Conclusions and Further Research Areas

In creating good computer animation, the focus is not on the problem of completing a given motion task, but more importantly on how this task is to be performed by the animated character. All the elements involved in an animated character must cooperate in synchronized harmony. Most of the animation systems built so far leave the burden of generating realistic animation to the animator. To remedy this problem, fundamental principles of traditional animation, such as *squash and stretch, exaggeration, follow through,* and *overlapping action* [20], should be formalized as high level constructs.

Physically-based modeling has emerged as a means of creating realistic animation. It proposes methods to create active models which react to applied forces, constraints, ambient media, or impenetrable obstacles, as one would expect from real physical objects. In this way, computer animators are unconcerned with the kinematic details of animations, knowing that physics will take care the low-level motions.

Physically-based modeling adds new levels of representation to object description in addition to geometry. Forces, torques, velocities, kinetic and potential energies, heat, and other physical quantities are used to control the creation and evolution of models. To construct the differential equations of the motion of the

models, different techniques (such as Lagrange equations, constraint-based methods) could be used. Constraint-based methods, which are highly suitable for this purpose, unify the creation of complex models with the control of the motion of the models. After constructing the equations of motion for the models, the equations should be solved using fast numerical methods.

In this thesis, we introduced a system for animating deformable models. Also a new formulation for animating deformable models, called the spring force formulation, was presented. The animation system uses three different formulations, namely the primal, hybrid, and spring force formulations, for animating the models so that the user could decide which one to use in an animation after considering the advantages and disadvantages of each formulation.

6.1 Contributions of the Thesis

The salient contributions of the thesis can be summarized as follows:

- In the spring force formulation that is presented, the elastic properties of the materials are represented as external spring forces, instead of using the stiffness matrix approach. In this way, the problem of automatically constructing the stiffness matrix is avoided.

- Since the stiffness matrix is not formed, models could be animated faster than the other approaches. The linear system of equations that should be solved to compute animation frames contains only mass and damping values which are the diagonal entries. This allows us to use simple linear system solving methods.

- The elastic properties of the materials could be given by setting the spring constants to proper values.

- Since the formulation models a deformable object using a finite number of grid points, it is possible to give different elastic properties to different parts of a model.

6.2 Future Research Directions

Future extensions to the research in this thesis could be summarized as follows:

- The equations of motion proposed for deformable models could be modified in such a way that new types of constraints will be taken into account by using external forces. This approach allows modeling and animating articulated bodies consisting of rigid and nonrigid parts by creating complex models from simpler primitives using point-to-point constraint. Also, other constraints, such as point-to-path and orientation, could be used to control the motion of the models.
- The animation system presented could be improved in rendering aspects. Animated frames could be rendered using sophisticated ray tracers, and texture mapping could be applied on deformable models.
- Aliasing artifacts (jaggies) noticeable in the animated frames could be eliminated by high-level visual processing.

Bibliography

[1] V. Akman, *Unobstructed Shortest Paths in Polyhedral Environments*, Lecture Notes in Computer Science, Vol. 251, Springer-Verlag, Berlin, 1987.

[2] W. W. Armstrong and M. W. Green, "The dynamics of articulated rigid bodies for purposes of animation," *The Visual Computer*, Vol. 1, No. 4, pp. 231–240, December 1985.

[3] B. Arnaldi, G. Dumont, and G. Hégron, "Dynamics and unification of animation control," *The Visual Computer*, Vol. 5, Nos. 1–2, pp. 22–31, January 1989.

[4] N. I. Badler, K. H. Manoochehri, and G. Walters, "Articulated figure positioning by multiple constraints," *IEEE Computer Graphics and Applications*, Vol. 7, No. 6, pp. 39–51, November 1987.

[5] D. Baraff, "Analytical methods for dynamic simulation of non-penetrating rigid bodies," *ACM Computer Graphics (Proc. SIGGRAPH'89)*, Vol. 23, No. 3, pp. 223–232, July 1989.

[6] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer Graphics and Applications*, Vol. 1, No. 1, pp. 11–23, January 1981.

[7] A. H. Barr, "Global and local deformations of solid primitives," *ACM Computer Graphics (Proc. SIGGRAPH'84)*, Vol. 18, No. 3, pp. 21–30, July 1984.

[8] R. Barzel and A. H. Barr, "A modeling system based on dynamic constraints," *ACM Computer Graphics (Proc. SIGGRAPH'88)*, Vol. 22, No. 4, pp. 179–188, August 1988.

[9] P. Bézier, *Numerical Control — Mathematics and Applications*, John Wiley and Sons, London, 1972.

[10] D. E. Breen, D. H. House, and P. H. Getto, "Physically-based particle model of woven cloth," *The Visual Computer*, Vol. 8, Nos. 5–6, pp. 264–277, June 1992.

[11] J. E. Chadwick, D. R. Haumann, and R. E. Parent, "Layered construction for deformable animated characters," *ACM Computer Graphics (Proc. SIGGRAPH'89)*, Vol. 23, No. 3, pp. 243–252, July 1989.

[12] M. F. Cohen, "Interactive spacetime control for animation," *ACM Computer Graphics (Proc. SIGGRAPH'92)*, Vol. 26, No. 2, pp. 293–302, July 1992.

[13] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[14] C. R. Feynman, *Modeling the Appearance of Cloth*, Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1986.

[15] M. Girard and A. Maciejewski, "Computational modeling for computer animation of legged figures," *ACM Computer Graphics (Proc. SIGGRAPH'85)*, Vol. 19, No. 3, pp. 263–270, July 1985.

[16] D. Haumann, *Modeling the Physical Behavior of Flexible Objects*, *SIGGRAPH* Tutorial 17 Notes, ACM, 1987.

[17] Sun Microsystems, Inc., *Sun View Programmer's Guide*, Mountain View, CA, 1986.

[18] P. Issacs and M. Cohen, "Controlling dynamic simulation with kinematic constraints, behavior functions, and inverse dynamics," *ACM Computer Graphics (Proc. SIGGRAPH'87)*, Vol. 21, No. 4, pp. 215–224, July 1987.

[19] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[20] J. Lasseter, "Principles of traditional animation applied to 3D computer animation," *ACM Computer Graphics (Proc. SIGGRAPH'87)*, Vol. 21, No. 4, pp. 35–44, July 1987.

[21] U. Güdükbay and B. Özgüç, "Free-form solid modeling using deformations," *Computers and Graphics*, Vol. 14, No. 3, pp. 491–500, September 1990.

[22] U. Güdükbay and B. Özgüç, "Animation of deformable models," in *Proc. of ATARV'93 (Advanced Techniques in Animation, Rendering and Visualization)*, pp. 193–206, Ankara, Turkey, July 1993. B. Özgüç, V. Akman (eds.) (A revised version of this paper will appear in a special issue of *Computer-Aided Design* in 1994.)

[23] U. Güdükbay, B. Özgüç, and V. Akman, "Physically-based modeling for realistic animation," in *Proc. of COMPUGRAPHICS'91 (First International Conference on Computational Graphics and Visualization Techniques)*, Vol. 2, pp. 37–46, Sesimbra, Portugal, September 1991. H. P. Santo (ed.).

[24] U. Güdükbay, B. Özgüç, and Y. Tokad, "An animation system for rigid and deformable models," *Computers and Graphics*, Vol. 17, No. 1, pp. 71–77, January-February 1993.

[25] U. Güdükbay and B. Özgüç, "Deformation of solid models," in *Proc. of ISCIS'89 (International Conference on Computer and Information Sciences)*, Vol. 1, pp. 525–534, İzmir, Turkey, October 1989. A. Doğaç, E. Gelenbe (eds.).

[26] U. Güdükbay, B. Özgüç, and V. Akman, "Constraint-based methods in realistic animation," in *Proc. of ISCIS'91 (International Conference on Computer and Information Sciences)*, Vol. 2, pp. 1195–1203, Antalya, Turkey, October 1991. Elsevier, Amsterdam. M. Baray, B. Özgüç (eds.).

[27] D. Metaxas and D. Terzopoulos, "Dynamic deformation of solid primitives with constraints," *ACM Computer Graphics (Proc. SIGGRAPH'92)*, Vol. 26, No. 2, pp. 309–312, July 1992.

[28] M. Moore and J. Wilhems, "Collision detection and response for computer animation," *ACM Computer Graphics (Proc. SIGGRAPH'88)*, Vol. 22, No. 4, pp. 289–298, August 1988.

[29] C. W. A. M. Overveld, "A technique for motion specification in computer animation," *The Visual Computer*, Vol. 6, No. 2, pp. 106–116, November 1990.

[30] A. Pentland and J. Williams, "Good vibrations: Modal dynamics for graphics and animation," *ACM Computer Graphics (Proc. SIGGRAPH'89)*, Vol. 23, No. 3, pp. 215–222, July 1989.

[31] C. B. Phillips, J. Zhao, and N. I. Badler, "Interactive real-time articulated figure manipulation using multiple kinematic constraints," *ACM Computer Graphics (Proc. SIGGRAPH'90)*, Vol. 24, No. 4, pp. 245–250, August 1990.

[32] J. Platt and A. H. Barr, "Constraint methods for flexible models," *ACM Computer Graphics (Proc. SIGGRAPH'88)*, Vol. 22, No. 4, pp. 279–288, August 1988.

[33] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, U.K., 1986.

[34] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *ACM Computer Graphics (Proc. SIGGRAPH'86)*, Vol. 20, No. 4, pp. 151–160, August 1986.

[35] D. Terzopoulos, "Regularization of inverse visual problems involving discontinuities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, pp. 413–424, 1986.

[36] D. Terzopoulos and K. Fleischer, "Modeling inelastic deformation: Viscoelasticity, plasticity, fracture," *ACM Computer Graphics (Proc. SIGGRAPH'88)*, Vol. 22, No. 4, pp. 269–278, August 1988.

[37] D. Terzopoulos, J. Platt, A. H. Barr, and K. Fleischer, "Elastically deformable models," *ACM Computer Graphics (Proc. SIGGRAPH'87)*, Vol. 21, No. 4, pp. 205–214, July 1987.

[38] D. Terzopoulos and A. Witkin, "Physically based models with rigid and deformable components," *IEEE Computer Graphics and Applications*, Vol. 8, No. 6, pp. 41–51, November 1988.

[39] N. M. Thalmann and D. Thalmann, "Six-hundred indexed references on computer animation," *The Journal of Visualization and Computer Animation*, Vol. 3, No. 3, pp. 147–174, July-September 1992.

[40] J. A. Thingvold and E. Cohen, "Physical modeling with B-spline surfaces for interactive design and animation," *ACM Computer Graphics (Proc. SIGGRAPH'90)*, Vol. 24, No. 4, pp. 129–137, August 1990.

[41] Y. Tokad, *Mühendislik Sistemlerinin Analizi — Kısım III*, Bilkent University, Ankara, Turkey, 1990.

[42] J. Weil, "The synthesis of cloth objects," *ACM Computer Graphics (Proc. SIGGRAPH'86)*, Vol. 20, No. 4, pp. 49–54, August 1986.

[43] J. Wilhelms, "Using dynamic analysis for realistic animation of rigid bodies," *IEEE Computer Graphics and Applications*, Vol. 7, No. 3, pp. 12–27, June 1987.

[44] A. Witkin, K. Fleischer, and A. H. Barr, "Energy constraints on parameterized models," *ACM Computer Graphics (Proc. SIGGRAPH'87)*, Vol. 21, No. 4, pp. 225–232, July 1987.

[45] A. Witkin, M. Gleischer, and W. Welch, "Interactive dynamics," *ACM Computer Graphics (Proc. SIGGRAPH'90)*, Vol. 24, No. 4, pp. 11–22, August 1990.

[46] A. Witkin and M. Kass, "Spacetime constraints," *ACM Computer Graphics (Proc. SIGGRAPH'88)*, Vol. 22, No. 4, pp. 159–168, August 1988.

[47] A. Witkin and W. Welch, "Fast animation and control of nonrigid structures," *ACM Computer Graphics (Proc. SIGGRAPH'90)*, Vol. 24, No. 4, pp. 243–252, August 1990.

Appendix A

The Implicit Functions for the Obstacles

Ellipsoid:

The implicit function for an ellipsoid is

$$f(x,y,z) = (x/a_1)^2 + (y/a_2)^2 + (z/a_3)^2, \tag{A.1}$$

where a_1, a_2, and a_3 are the scaling factors.

Hyperboloid of one piece:

The implicit function for an hyperboloid of one piece is

$$f(x,y,z) = (x/a_1)^2 + (y/a_2)^2 - (z/a_3)^2, \tag{A.2}$$

where a_1, a_2, and a_3 are the scaling factors.

Toroid:

The implicit function for a toroid is

$$f(x,y,z) = (((x/a_1)^2 + (y/a_2)^2))^{0.5} - a_4)^2 + (z/a_3)^2, \tag{A.3}$$

where a_1, a_2, and a_3 are the scaling factors, and $a_4 = \hat{a}/\sqrt{(a_1^2 + a_2^2)}$. Here, $\hat{a}$ is the torus radius.

$$\text{if} \quad \begin{cases} f(x_0,y_0,z_0) = 1, & (x_0,y_0,z_0) \text{ is on the surface.} \\ f(x_0,y_0,z_0) > 1, & (x_0,y_0,z_0) \text{ lies outside the surface.} \\ f(x_0,y_0,z_0) < 1, & (x_0,y_0,z_0) \text{ lies inside the surface.} \end{cases} \tag{A.4}$$

These inside-outside functions are given for the surfaces in standard positions and orientations (e.g., centered at origin). To describe obstacles in desired configurations, it is necessary to translate and rotate these objects. The rigid body transformations are invertible; thus, the original inside-outside function can be used after a function inversion. If the original surface is denoted by $\underline{x}$, and translated and rotated surface is denoted by $\hat{\underline{x}}$ then the new surface is given by,

$$\hat{\underline{x}} = \mathtt{M}\underline{x} + \underline{b}, \tag{A.5}$$

where $\mathtt{M}$ is a rotation matrix, and $\underline{b}$ is a displacement vector.

The new inside-outside function is calculated by inverting the transformation and substituting into the old inside-outside function; i.e.,

$$\hat{f}(\hat{x},\hat{y},\hat{z}) = f(x,y,z), \tag{A.6}$$

where

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathtt{M}^T \begin{bmatrix} \hat{x} - b_1 \\ \hat{y} - b_2 \\ \hat{z} - b_3 \end{bmatrix} \tag{A.7}$$

Note that $\mathtt{M}^{-1} = \mathtt{M}^T$ since the rotation matrices are orthogonal.

Appendix B

The External Spring Force Expressions

In this part, the external spring force expressions for the grid points that are on the boundaries and corners are given.

B.1 The External Spring Force Expressions for the Boundaries

If $i = 0$ and $j = 1, 2, \cdots, N-1$, then (see Fig. 4.7 (a))

$$\left.\begin{aligned} \mathbf{f}_{0,j} &= k\left[(\mathbf{x}_{0,j} - \mathbf{x}_{0,j-1}) - \ell \frac{\mathbf{x}_{0,j}-\mathbf{x}_{0,j-1}}{\|\mathbf{x}_{0,j}-\mathbf{x}_{0,j-1}\|}\right] \\ &+ k\left[(\mathbf{x}_{0,j} - \mathbf{x}_{0,j+1}) - \ell \frac{\mathbf{x}_{0,j}-\mathbf{x}_{0,j+1}}{\|\mathbf{x}_{0,j}-\mathbf{x}_{0,j+1}\|}\right] \\ &+ k\left[(\mathbf{x}_{0,j} - \mathbf{x}_{1,j}) - \ell \frac{\mathbf{x}_{0,j}-\mathbf{x}_{1,j}}{\|\mathbf{x}_{0,j}-\mathbf{x}_{1,j}\|}\right] \end{aligned}\right\} \tag{B.1}$$

If $i = M$ and $j = 1, 2, \cdots, N-1$, then (see Fig. 4.7 (b))

$$\left.\begin{aligned} \mathbf{f}_{M,j} &= k\left[(\mathbf{x}_{M,j} - \mathbf{x}_{M,j-1}) - \ell \frac{\mathbf{x}_{M,j}-\mathbf{x}_{M,j-1}}{\|\mathbf{x}_{M,j}-\mathbf{x}_{M,j-1}\|}\right] \\ &+ k\left[(\mathbf{x}_{M,j} - \mathbf{x}_{M,j+1}) - \ell \frac{\mathbf{x}_{M,j}-\mathbf{x}_{M,j+1}}{\|\mathbf{x}_{M,j}-\mathbf{x}_{M,j+1}\|}\right] \\ &+ k\left[(\mathbf{x}_{M,j} - \mathbf{x}_{M-1,j}) - \ell \frac{\mathbf{x}_{M,j}-\mathbf{x}_{M-1,j}}{\|\mathbf{x}_{M,j}-\mathbf{x}_{M-1,j}\|}\right] \end{aligned}\right\} \tag{B.2}$$

If $i = 1, 2, \cdots, M-1$ and $j = 0$, then (see Fig. 4.7 (c))

$$\left.\begin{aligned} \mathbf{f}_{i,0} &= k\left[(\mathbf{x}_{i,0} - \mathbf{x}_{i-1,0}) - \ell \frac{\mathbf{x}_{i,0}-\mathbf{x}_{i-1,0}}{||\mathbf{x}_{i,0}-\mathbf{x}_{i-1,0}||}\right] \\ &+ k\left[(\mathbf{x}_{i,0} - \mathbf{x}_{i+1,0}) - \ell \frac{\mathbf{x}_{i,0}-\mathbf{x}_{i+1,0}}{||\mathbf{x}_{i,0}-\mathbf{x}_{i+1,0}||}\right] \\ &+ k\left[(\mathbf{x}_{i,0} - \mathbf{x}_{i,1}) - \ell \frac{\mathbf{x}_{i,0}-\mathbf{x}_{i,1}}{||\mathbf{x}_{i,0}-\mathbf{x}_{i,1}||}\right] \end{aligned}\right\} \tag{B.3}$$

If $i = 1, 2, \cdots, M-1$ and $j = N$, then (see Fig. 4.7 (d))

$$\left.\begin{aligned} \mathbf{f}_{i,N} &= k\left[(\mathbf{x}_{i,N} - \mathbf{x}_{i-1,N}) - \ell \frac{\mathbf{x}_{i,N}-\mathbf{x}_{i-1,N}}{||\mathbf{x}_{i,N}-\mathbf{x}_{i-1,N}||}\right] \\ &+ k\left[(\mathbf{x}_{i,N} - \mathbf{x}_{i+1,N}) - \ell \frac{\mathbf{x}_{i,N}-\mathbf{x}_{i+1,N}}{||\mathbf{x}_{i,N}-\mathbf{x}_{i+1,N}||}\right] \\ &+ k\left[(\mathbf{x}_{i,N} - \mathbf{x}_{i,N-1}) - \ell \frac{\mathbf{x}_{i,N}-\mathbf{x}_{i,N-1}}{||\mathbf{x}_{i,N}-\mathbf{x}_{i,N-1}||}\right] \end{aligned}\right\} \tag{B.4}$$

B.2 The External Spring Force Expressions for the Corners

If $i = 0$ and $j = 0$, then (see Fig. 4.8 (a))

$$\left.\begin{aligned} \mathbf{f}_{0,0} &= k\left[(\mathbf{x}_{0,0} - \mathbf{x}_{0,1}) - \ell \frac{\mathbf{x}_{0,0}-\mathbf{x}_{0,1}}{||\mathbf{x}_{0,0}-\mathbf{x}_{0,1}||}\right] \\ &+ k\left[(\mathbf{x}_{0,0} - \mathbf{x}_{1,0}) - \ell \frac{\mathbf{x}_{0,0}-\mathbf{x}_{1,0}}{||\mathbf{x}_{0,0}-\mathbf{x}_{1,0}||}\right] \end{aligned}\right\} \tag{B.5}$$

If $i = 0$ and $j = N$, then (see Fig. 4.8 (b))

$$\left.\begin{aligned} \mathbf{f}_{0,N} &= k\left[(\mathbf{x}_{0,N} - \mathbf{x}_{0,N-1}) - \ell \frac{\mathbf{x}_{0,N}-\mathbf{x}_{0,N-1}}{||\mathbf{x}_{0,N}-\mathbf{x}_{0,N-1}||}\right] \\ &+ k\left[(\mathbf{x}_{0,N} - \mathbf{x}_{1,N}) - \ell \frac{\mathbf{x}_{0,N}-\mathbf{x}_{1,N}}{||\mathbf{x}_{0,N}-\mathbf{x}_{1,N}||}\right] \end{aligned}\right\} \tag{B.6}$$

If $i = M$ and $j = 0$, then (see Fig. 4.8 (c))

$$\left.\begin{aligned} \mathbf{f}_{M,0} &= k\left[(\mathbf{x}_{M,0} - \mathbf{x}_{M-1,0}) - \ell \frac{\mathbf{x}_{M,0}-\mathbf{x}_{M-1,0}}{||\mathbf{x}_{M,0}-\mathbf{x}_{M-1,0}||}\right] \\ &+ k\left[(\mathbf{x}_{M,0} - \mathbf{x}_{M,1}) - \ell \frac{\mathbf{x}_{M,0}-\mathbf{x}_{M,1}}{||\mathbf{x}_{M,0}-\mathbf{x}_{M,1}||}\right] \end{aligned}\right\} \tag{B.7}$$

If $i = M$ and $j = N$, then (see Fig. 4.8 (d))

$$\left.\begin{aligned} \mathbf{f}_{M,N} &= k\left[(\mathbf{x}_{M,N} - \mathbf{x}_{M-1,N}) - \ell \frac{\mathbf{x}_{M,N}-\mathbf{x}_{M-1,N}}{||\mathbf{x}_{M,N}-\mathbf{x}_{M-1,N}||}\right] \\ &+ k\left[(\mathbf{x}_{M,N} - \mathbf{x}_{M,N-1}) - \ell \frac{\mathbf{x}_{M,N}-\mathbf{x}_{M,N-1}}{||\mathbf{x}_{M,N}-\mathbf{x}_{M,N-1}||}\right] \end{aligned}\right\} \tag{B.8}$$