MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

A Survey of Deformable Modeling in Computer Graphics

Sarah F. F. Gibson Brian Mirtich

TR-97-19 November 1997

Abstract

This paper presents a survey of the work done in modeling deformable objects within the computer graphics research community. The research has a long history and a wide variety of approaches have been used. This paper organizes the diversity of research by the technique used rather than by the application, although applications are discussed throughout. This paper presents some purely geometric approaches for modeling deformable objects, but focuses on physically based approaches. In the latter category are mass-spring models, finite element models, approximate continuum models, and low degree of freedom models. Special emphasis is placed on finite element models, which offer the greatest accuracy, but have seen limited use in computer graphics. The paper also suggests important areas for future research.

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Information Technology Center America; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Information Technology Center America. All rights reserved.

Copyright © Mitsubishi Electric Information Technology Center America, 1997
201 Broadway, Cambridge, Massachusetts 02139

1. First printing, TR97-19, October 1997

1 Introduction

Computers have become an indispensable tool in modeling and simulation. As computational power increases, users and applications are demanding ever increasing levels of realism in these domains. This trend is particularly apparent in computer graphics where more sophisticated geometric shapes and physical objects are being modeled in the context of complex physical environments.

In particular, the ability to model and manipulate deformable objects is essential to many applications. Approaches for modeling object deformation range from non-physical methods—where individual or groups of control points or shape parameters are manually adjusted for shape editing and design—to methods based on continuum mechanics—which account for the effects of material properties, external forces, and environmental constraints on object deformation.

Deformable object modeling has been studied in computer graphics for more than two decades, across a range of applications. In computer-aided design and computer drawing applications, deformable models are used to create and edit complex curves, surfaces, and solids. Computer aided apparel design uses deformable models to simulate fabric draping and folding. In image analysis, deformable models have been used to segment images and to fit curved surfaces to noisy image data. Deformable models have been used in animation and computer graphics, particularly for the animation of clothing, facial expression, and human or animal characters. Finally, surgical simulation and training systems demand both real-time and physically realistic modeling of complex, non-linear, deformable tissues. For summaries of the use of deformable models in image analysis and cloth modeling applications, the reader is referred to [MT96] and [NG96], respectively. This survey focuses on mathematical and computational techniques used for modeling deformable objects in computer graphics applications.

2 Non-physical models

Most of the models discussed in this paper use some sort of physical principles to compute the shapes or motions of deformable objects. However, many applications, particularly in design, employ purely geometric techniques. Generally, these techniques are computationally efficient, and they rely on the skill of the designer rather than on physical principles.

2.1 Splines and patches

Many early efforts in modeling deformable objects arose in the field of computer aided geometric design (CAGD). Designers needed ways to numerically specify curves and surfaces, and intuitive ways to modify and refine these objects. From this need came Bezier curves, and subsequently many other methods of specifying curves with a small vector of numbers, including: double-quadratic curves,

B-splines, rational B-splines, non-uniform rational B-splines (NURBS), and β-splines. These methods can represent both planar and 3D curves and have related 2D patches for specifying surfaces.

In these representations, the curve or surface is represented by a set of control points. The designer adjusts the shape of the objects by moving control points to new positions, by adding or deleting control points, or by changing their weights. For example, consider the cubic spline curve of Figure 1. The curve is specified mathematically by the locations of its four control points. The curve interpolates the control points **a** and **d**, and is tangent to two of the sides of the control polygon. Moving the control points changes the curve in a predictable manner.

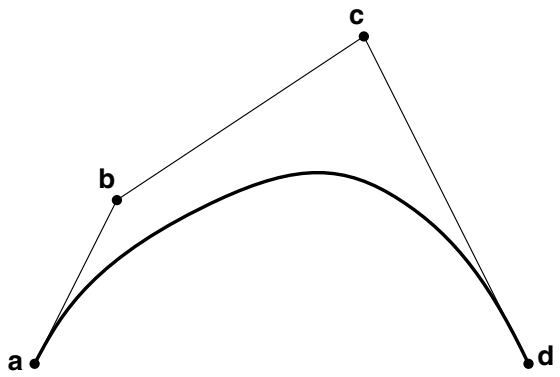

Figure 1: *A cubic spline curve. The shape of the curve is edited by moving control points* **a**, **b**, **c**, *or* **d**.

This parameter-based object representation is computationally efficient and supports interactive modification. It also affords subtle control of object shape when objects are represented by many control points. However, this level of control is sometimes a disadvantage: precise specification or modification of curves or surfaces can be laborious. Even a perceptually simple change may require adjustment of many control points.

Textbooks by Bartels *et. al.* and Farin provide comprehensive coverage of curve and surface modeling with splines [BBB87, Far90]. Extensive research results have been reported in journals devoted to computer aided design and CAGD. Some examples include [JV84, Pie89a, Pie89b, HM90, CR94, GP89]. Parent and Allen *et. al.* present methods that move local neighbors of a selected point together with point itself, based on some measure of proximity [Par77, AWW89]. Bartels and Beaty describe ways to edit the shape by directly moving points on a B-spline curve, rather than by manipulating control points [BB89]. Finkelstein and Salesin devised a multi-resolution wavelet-based representation for cubic B-spline curves. By selectively choosing which wavelet components are affected, the designer specifies whether control point modifications change the curve sweep (overall shape) or the curve character (local texture) [FS94].

2.2 Free-form deformation

Free-form deformation (FFD) is a general method for deforming objects that provides a higher and more powerful level of control than adjusting individual control points. FFD's change the shape of an object by deforming the space in which the object lies. This technique can be applied to many different graphical representations, including: points, polygons, splines, parametric patches, and implicit surfaces.

Barr's early work in this area examined deformation in terms of geometric mappings of three-dimensional space. For example, consider the map $f : \Re^3 \rightarrow \Re^3$ given by:

$$f(\mathbf{p}) = \begin{bmatrix} \cos p_z & -\sin p_z & 0 \\ \sin p_z & \cos p_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}.$$

This mapping causes objects to twist about the z-axis. Related mappings generate rigid motion, tapering, bending, and so on. More complex deformations can be constructed by composing mappings. Barr studied the transformation of surface tangent and normal vectors under such mappings, and the hierarchical arrangement of such mappings [Bar84].

Barr's space-warping method provides a powerful design tool, but the possible regions and types of deformations are limited, and the user's control of deformation is not very intuitive. Sederberg and Parry, who introduced the term free-form deformation, generalized Barr's approach by embedding an object in a lattice of grid points of some standard geometry, such as a cube or cylinder. Manipulating nodes of the grid induces deformations on the space inside the grid, and these deformations transform the underlying graphics primitives that form the object. If $\{U_i\}$ is the set of three-dimensional cells defined by the grid, a free-form deformation can be thought of as a collection of mappings of the form $f_i : U_i \rightarrow \Re^3$. The f_i mappings define how different object representations are affected by the deformation. For example, vertex or control points are directly mapped to new positions, enabling planes to be moved and warped. Splines and spline patches are modified by mapping control points and their derivatives. For parametric and implicit surfaces, the map defines a change of variables so that the represented objects are moved and warped while still remaining parametric and implicit respectively. Sederberg and Parry advocate the use of trivariate Bernstein polynomials for the f_i, which provide for intuitive manipulation through the control point mesh. These maps have convenient upper and lower bounds on the determinants of their Jacobians, which measure local volume changes [SP86].

The basic FFD method has been extended by several others. Coquillart provides a toolkit of lattices with different sizes, resolutions and geometries that can be positioned over the object for selective control of sub-regions of the surface [Coq90]. Hsu *et. al.* allow direct manipulation of surface or curve points by converting the desired movement of these points to equivalent grid point movement. This underconstrained problem is solved by choosing the grid point movement with the minimum least-squares energy that produces the desired

object manipulation [HHK92]. MacCracken and Joy perform FFD on lattices with arbitrary topology, using a subdivision algorithm to refine the lattice and provide the desired shape. Special lattice points along sharp edges and corners are handled separately [MJ96].

3 Mass-spring models

Non-physical methods for modeling deformation are limited by the expertise and patience of the user. Deformations must be explicitly specified and the system has no knowledge about the nature of the objects being manipulated. Using these tools alone, modeling an object as complex as the human face, for example, is a daunting task. As desktop computing power and graphics capabilities increased during the 1980's, the graphics community began exploring *physically based* methods for animation and modeling. These methods use physical principles and computational power for realistic simulation of complex physical processes that would be difficult or impossible to model with purely geometric techniques.

Mass-spring systems are one physically based technique that has been used widely and effectively for modeling deformable objects. An object is modeled as a collection of point masses connected by springs in a lattice structure (Figure 2). The spring forces are often linear (Hookean), but nonlinear springs can

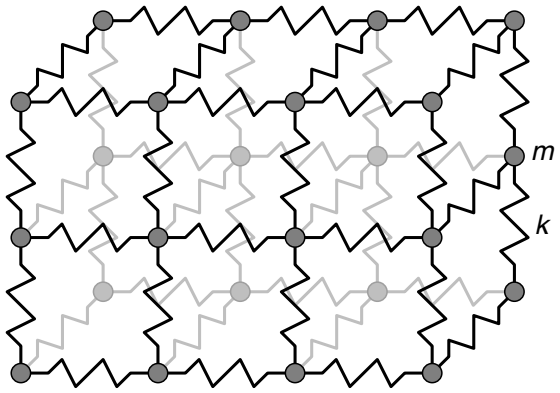

Figure 2: *A portion of a mass-spring model. Springs connecting point masses exert forces on neighboring points when a mass is displaced from its rest positions.*

be used to model tissues such as human skin that exhibit inelastic behavior. In a dynamic system, Newton's Second Law governs the motion of a single mass point in the lattice:

$$m_i \ddot{\mathbf{x}}_i = -\gamma_i \dot{\mathbf{x}}_i + \sum_j \mathbf{g}_{ij} + \mathbf{f}_i.$$

Here, m_i is the mass of the point, $\mathbf{x}_i \in \Re^3$ is its position, and the terms on the

right-hand side are forces acting on the mass point. The first right-hand term is a velocity-dependent damping force, $\mathbf{g}_{ij}$ is the force exerted on mass i by the spring between masses i and j, and $\mathbf{f}_i$ is the sum of other external forces, (e.g. gravity or user applied forces), acting on mass i.

The equations of motion for the entire system are assembled from the motions of all of the mass points in the lattice. Concatenating the position vectors of the N individual masses into a single 3N-dimensional position vector $\mathbf{x}$, one obtains:

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{f}. \tag{1}$$

Here, $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ are the $3N \times 3N$ mass, damping, and stiffness matrices, respectively. Although large, these matrices are typically quite sparse. $\mathbf{M}$ and $\mathbf{C}$ are diagonal matrices and $\mathbf{K}$ is banded because it encodes spring forces which are functions of distances between neighboring mass points only. The vector $\mathbf{f}$ is a $3N$-dimensional vector representing the total external forces on the mass points. The system is evolved forward through time by re-expressing (1) as a system of first-order differential equations:

$$\begin{aligned} \dot{\mathbf{v}} &= \mathbf{M}^{-1}\left(-\mathbf{C}\mathbf{v} - \mathbf{K}\mathbf{x} + \mathbf{f}\right) \\ \dot{\mathbf{x}} &= \mathbf{v} \end{aligned}$$

where $\mathbf{v}$ is the velocity vector of the system of mass points. A variety of numerical integration techniques are available to compute $\mathbf{x}$ and $\mathbf{v}$ as functions of time.

3.1 Applications

Mass-spring systems have been used widely in facial animation. The goal of these applications is to model subtle human facial expressions, in applications ranging from computerized expression of American Sign Language to storytelling. Early work of Platt and Badler used *tension nets*, static versions of mass-spring systems, which essentially solve the static system:

$$\mathbf{K}\mathbf{x} = \mathbf{f}.$$

The face is modeled as a two-dimensional mesh of points warped around an ovoid and connected by linear springs. Muscle actions are represented by the application of a force to a particular region of nodes, inducing node displacements that propagate outward to adjacent nodes [PB81]. This approach was expanded by Waters, who developed more sophisticated models for node displacements in response to muscle forces. In Water's approach, muscles directly displaced nodes within *zones of influence* which were parameterized by radius, fall-off coefficients, and other parameters [Wat87].

Terzopoulos and Waters were the first to apply dynamic mass-spring systems to facial modeling [TW90]. They constructed a three-layer mesh of mass points based on three anatomically distinct layers of facial tissue: the dermis, a layer of subcutaneous fatty tissue, and the muscle layer. The upper surface of the dermal

layer forms the epidermis, and the muscles are attached to rigid bone below and to the fascia above (Figure 3). Like the earlier work in facial modeling, the actuating muscles correspond to actual muscles in the human face.

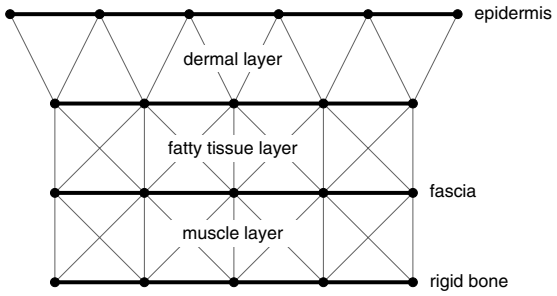

Figure 3: *The three-layer skin mesh used by Terzopoulos and Waters.*

Different spring constants were used to model the different layers based on tissues properties. Because some of the tissues are incompressible, and because the standard mass-spring model cannot easily enforce this constraint, additional forces are applied to mass points in order to maintain constant volume. These forces are computed by comparing the volumes of the regions between the nodes to their rest volumes. A second-order Runge-Kutta method is used to integrate the equations of motion. The facial model has 6500 springs, and is animated at interactive rates.

The previously mentioned work uses hand-crafted models of a generic face. Waters and Terzopoulos later developed a technique to generate facial models for particular individuals from a radial laser-scanned image data. The scanned data provides a texture map for the top layer of the mesh and the initial geometry of the mesh. A preprocessing step adjusts spring constants to increase the node density over image features with high information content. Real-time animation is achieved with a simplified two-layer skin model and a Euler integration method [WT91]. Later, Waters used computer tomography (CT) data to generate the three-dimensional facial model [Wat92]. Lee, Terzopoulos, and Waters used a mesh adaptation algorithm that tailors a generic mesh to the individual's features by locating these features in a laser-scanned depth image. For improved realism, their formulation also includes constraint forces to prevent muscles and fascia nodes from penetrating the skull [LTW93, LTW95].

Koch *et. al.* use a mass spring model of facial tissue to predict the post-operative appearance of patients whose underlying bone structure has been changed during cranio-facial surgery. Spring stiffnesses for the system are derived from tissue densities recorded by 3D Computed Tomography (CT) imaging.

Mass-spring systems have been used extensively for animation. Chadwick *et. al.* combined mass-spring models with free form deformations to animated muscles in human character animation. The muscles are embedded in a lattice of 8-node mass-spring elements and deformed by applying forces to the lattice

node points. The dynamic deformation of the muscle model is calculated by interpolating the motion of the lattice points [CHP89].

Terzopoulos *et. al.* describe a mass-spring model for deformable bodies that experience state transitions from solid to liquid [TPF89]. Nodes are connected by springs in a hexahedral lattice. Each node has an associated temperature as well as a position, and the spring stiffnesses between nodes are dependent on temperature. A discretized form of the heat equation is used to compute the diffusion of heat through the material, and the changes in nodal temperatures. Increased temperatures decrease the spring stiffnesses; when the melting point is reached, the stiffness is set to 0, severing the bond. When all bonds between a node and its neighbors are severed, it becomes an independent "glop" of fluid. Glops are simulated using a discrete fluid model.

Tu and Terzopoulos have used mass-spring systems with full dynamics to generate "artificial fish." The fish model comprises 23 mass points and 91 springs. Muscles actuate the body, creating hydrodynamic forces that propel the fish. An implicit Euler integration method is used for added stability in the presence of large forces. Aquatic plants are also modeled with mass-spring systems [TT94].

Christensen *et. al.* also embedded objects in a cubic, 8-node mass-spring lattice and used dynamic simulation and FFD's to animate the embedded objects [CMN97]. By changing the spring stiffness matrix to correspond to vibrational modes such as twisting, scaling, and shearing at specific times in the animation sequence, they obtain whimsical but physical behavior of the animated objects.

3.2 Advantages and limitations

Mass-spring systems are a simple physical model with well understood dynamics. They are easy to construct, and can be animated at rates not possible with some of the continuum methods discussed below. Interactive and even real-time simulation of mass-spring systems is possible with today's desktop systems. Mass-spring systems are well suited to parallel computation as well, due to the local nature of the interactions between nodes. They have demonstrated their utility in animating deformable objects that would be very difficult to animate by hand.

Mass-spring systems have some drawbacks. The discrete model is a significant approximation of the true physics that occurs in a continuous body. The lattice is tuned through its spring constants, and proper values for these constants are not always easy to derive from measured material properties. In addition, certain constraints are not naturally expressed in the model. For example, incompressible volumetric objects or thin surfaces that are resistant to bending are difficult to model as mass-spring systems. These and other phenomena can sometimes be modeled using additional springs with an increase in computational cost.

Finally, mass-spring systems sometimes exhibit a problem referred to as "stiffness" which can occur when spring constants are large. Large spring constants are used to model objects that are nearly rigid, or to model hard

constraints due to physical interactions, such as a non-penetration constraint between a deformable object and a rigid object. Stiff systems are problematic because they have poor stability, requiring the numerical integrator to take small time steps, even when the interesting modes of motion occur over much longer time intervals. The result is a slow simulation. A good description of this problem is in [BW92].

4 Continuum models and finite element methods

Mass-spring models start with a discrete object model. More accurate physical models treat deformable objects as a continuum: solid bodies with mass and energies distributed throughout. In making this distinction, it is important to separate the model from the method used to solve it. Models can be discrete or continuous but the computational methods used for solving the models in computer simulations are ultimately discrete. In the analysis of dynamic systems, numerical integration techniques approximate the system at discrete time steps. Furthermore, even a continuum model must be parameterized by a finite state vector. For deformable object modeling, this state vector often comprises the positions and velocities of representative points within the material. However, unlike the discrete mass-spring models, continuum models are derived from equations of continuum mechanics.

The full continuum model of a deformable object considers the equilibrium of a general body acted on by external forces. The object deformation is a function of these acting forces and the object's material properties. The object reaches equilibrium when its potential energy is at a minimum.

The total potential energy of a deformable system is denoted by Π, and is given by:

$$\Pi = \Lambda - W, \tag{2}$$

where Λ is the total strain energy of the deformable object, and W is the work done by external loads on the deformable object.[1] The strain energy is the energy stored in the body as material deformation. The work done by applied loads is due to three sources: concentrated loads applied at discrete points, loads distributed over the body, such as gravitational forces, and loads distributed over the surface of the object, such as pressure forces.

In order to determine the equilibrium shape of the object, both Λ and W are expressed in terms of the object deformation, which is represented by a function of the material displacement over the object. The system potential energy reaches a minimum when the derivative of Π with respect to the material

[1] The $-W$ term in (2) deserves some explanation. The deformable *system* comprises the deformable object, plus the loads that act on it. Work that a load does on the deformable object decreases the load's own potential energy and must be subtracted from the total potential energy of the system. Some of this energy may be recouped as strain energy in the deformable object, as when a weight suspended from a spring stretches the spring.

displacement function is zero. This approach leads to a continuous differential equilibrium equation that must be solved for the material displacement.

Because is not always possible to find a closed-form analytic solution of this differential equation, a number of numerical methods are used to approximate the object deformation. As discussed previously, mass-spring methods approximate the object as a finite mesh of points and discretize the equilibrium equation at the mesh points. Finite element methods, FEM, divide the object into a set of elements and approximate the continuous equilibrium equation over each element. In the following, a general approach to FEM is presented. The reader is referred to textbooks such as [Bat96] or [Seg84] for more detailed treatment.

4.1 General finite element methods

FEM is used to find an approximation for a continuous function that satisfies some equilibrium expression such as the deformation equilibrium expression described above. In FEM, the continuum, or object, is divided into elements joined at discrete node points. A function that solves the equilibrium equation is found for each element. The solution is subject to constraints at the node points and the element boundaries so that continuity between the elements is achieved. Unlike mass-spring methods, where the equilibrium equation is discretized and solved at finite mass points, in FEM, the system is discretized by representing the desired function within each element as a finite sum of element-specific interpolation, or *shape*, functions.

In the case when the desired function is a scalar function $\Phi(x, y, z)$, the value of Φ at the point (x, y, z) is approximated by:

$$\Phi(x, y, z) \approx \sum_i h_i(x, y, z)\Phi_i,$$

where the h_i are the interpolation functions for the element containing (x, y, z), and the Φ_i are the values of $\Phi(x, y, z)$ at the element's node points. Solving the equilibrium equation becomes a matter of determining the finite set of node values Φ_i that minimize the total potential energy in the body.

4.2 Displacement-based finite element methods

In displacement-based FEM, an equilibrium equation is derived from the goal of minimizing the system potential energy of (2), with respect to the material displacement over the object. Unlike the scalar function in the example above, material displacement is a 3D vector function over the object. However, the same discretizing method is applied. The basic steps in using FEM to compute object deformations are:

1. Derive an equilibrium equation from the potential energy equation of (2) in terms of material displacement over the continuum.

2. Select appropriate finite elements and corresponding interpolation functions for the problem. Subdivide the object into elements.

3. For each element, re-express the components of the equilibrium equation in terms of the interpolation functions and the element's node displacements.

4. Combine the set of equilibrium equations for all of the elements in the object into a single system. Solve the system for the node displacements over the whole object.

5. Use the node displacements and the interpolation functions of a particular element to calculate displacements or other quantities of interest (such as internal stress or strain) for points within the element.

4.2.1 Expressing the equilibrium equation in terms of object deformation

From (2), the potential energy of a body is the sum of the total strain energy and the work done on the body by external forces. In order to determine the object deformation that will minimize Π, Λ and W are expanded in terms of the material displacements.

The strain energy Λ is derived from an integral expression over the volume of the material stress, $\boldsymbol{\sigma}$, and strain, $\boldsymbol{\varepsilon}$, components:

$$\Lambda = \frac{1}{2}\int_V \boldsymbol{\sigma}^T \boldsymbol{\varepsilon}\ dV = \frac{1}{2}\int_V \boldsymbol{\varepsilon}^T \mathbf{D}\boldsymbol{\varepsilon}\ dV, \tag{3}$$

where $\mathbf{D}$ is the linear matrix which relates the stress and strain components for an elastic system (from the generalized Hooke's law), and $\boldsymbol{\sigma}^T = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{yz}, \sigma_{zx}, \sigma_{xy})$ and $\boldsymbol{\varepsilon}^T = (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, \varepsilon_{yz}, \varepsilon_{zx}, \varepsilon_{xy})$ are vectors of the stress and strain components. In an elastic system, the material strain $\boldsymbol{\varepsilon}$ is related to the displacement vector $\mathbf{u} = (u, v, w)^T$ by a set of differential equations:

$$\begin{array}{lcllcl} \varepsilon_{xx} & = & \frac{\partial u}{\partial x} & \varepsilon_{yz} & = & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \varepsilon_{yy} & = & \frac{\partial v}{\partial y} & \varepsilon_{zx} & = & \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \\ \varepsilon_{zz} & = & \frac{\partial w}{\partial z} & \varepsilon_{xy} & = & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{array} \tag{4}$$

These strain relationships are used to expand (3) in terms of material displacement. This is done explicitly once the interpolation functions have been chosen.

The work done by an external force $\mathbf{f}(x, y, z)$ is computed as the dot product of the applied force and the material displacement $\mathbf{u}$ integrated over the object volume:

$$W = \int_V \mathbf{u} \cdot \mathbf{f}\ dV \tag{5}$$

The forces applied to a deformable body include: distributed body forces, such as gravity; distributed surface forces, such as pressure or viscous drag; and concentrated loads. This leads to the expansion of (5) to yield:

$$W = \int_V \mathbf{u} \cdot \mathbf{f}_b\ dV + \int_\Gamma \mathbf{u} \cdot \mathbf{f}_s\ dS + \sum_i \mathbf{u}_i \cdot \mathbf{p}_i, \tag{6}$$

where: $\mathbf{f}_b(x, y, z)$ are body forces applied to the object volume V, $\mathbf{f}_s(x, y, z)$ are surface forces applied to the object surface Γ, and $\mathbf{p}_i$ are concentrated loads acting at the points (x_i, y_i, z_i).

4.2.2 Choosing the element and interpolation functions

Once the object geometry and the required accuracy are known, an appropriate set of elements and interpolation functions are chosen. The best choices of elements and interpolation functions depend on the object shape, convergence requirements, degrees of freedom, and trade-offs between accuracy and computational requirements. In general, using elements that have more nodes and more complex interpolation functions require fewer elements for the same degree of accuracy. However, because higher order interpolation functions require more sophisticated numerical integration techniques, they require more computation per element. Figure 4 illustrates some common 2D and 3D elements and their node points.

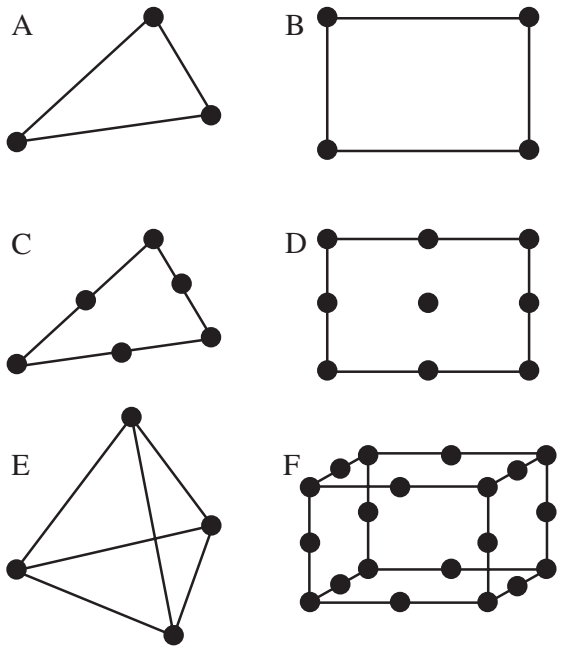

Figure 4: *Some common 2D and 3D elements used in FEM: (A) linear triangular element with 3 nodes, (B) linear rectangular element with 4 nodes, (C) quadratic triangular element with 6 nodes, (D) Lagrangian element with 9 nodes, (E) tetrahedral element with 4 nodes, and (F) 20-node brick element.*

Each element has an interpolation equation that describes how quantities vary continuously within the element. Usually, the interpolation equation is polynomial. The interpolation equation must satisfy specified values at each of

the element's M nodes, so if it is polynomial, it must have at least M coefficients to provide the necessary degrees of freedom. The order of the polynomial is chosen to be the lowest one possible that provides enough coefficients. For example, the linear triangular element is planar, and therefore uses a polynomial in x and y for its interpolation equation. Since there are three nodes, the general first order polynomial, with three undetermined coefficients, provides sufficient degrees of freedom:

$$\Phi = a_1 + a_2 x + a_3 y.$$

To derive the interpolation functions from the interpolation equation, one constrains the interpolated value of Φ at each node point (x_i, y_i) to equal the node value Φ_i. For the linear triangular element,

$$\begin{aligned} \Phi_1 &= a_1 + a_2 x_1 + a_3 y_1 \\ \Phi_2 &= a_1 + a_2 x_2 + a_3 y_2 \\ \Phi_3 &= a_1 + a_2 x_3 + a_3 y_3. \end{aligned}$$

Solving this system for the coefficients a_i yields:

$$\begin{aligned} a_1 &= [(x_2 y_3 - x_3 y_2)\Phi_1 + (x_3 y_1 - x_1 y_3)\Phi_2 + (x_1 y_2 - x_2 y_1)\Phi_3]/2A \\ a_2 &= [(x_2 - x_3)\Phi_1 + (x_3 - x_1)\Phi_2 + (x_1 - x_2)\Phi_3]/2A \\ a_3 &= [(x_3 - x_1)\Phi_1 + (x_1 - x_3)\Phi_2 + (x_2 - x_1)\Phi_3]/2A \end{aligned}$$

where A is the area of the triangular element. Finally, substituting a_1, a_2, and a_3 back into the interpolation equation and rearranging, the unknown value $\Phi(x, y)$ can be expressed in terms of the node values Φ_1, Φ_2, and Φ_3, and the interpolation functions, h_1, h_2, and h_3:

$$\Phi = h_1\Phi_1 + h_2\Phi_2 + h_3\Phi_3,$$

where the interpolation functions are:

$$\begin{aligned} h_1 &= [(x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y]/2A \\ h_2 &= [(x_3 y_1 - x_1 y_3) + (y_3 - y_1)x + (x_1 - x_3)y]/2A \\ h_3 &= [(x_1 y_2 - x_2 y_1) + (y_1 - y_2)x + (x_2 - x_1)y]/2A \end{aligned} \tag{7}$$

In general, as exhibited in (7), interpolation functions have the following properties:

- They are locally applied only to the given element and are treated as zero outside of the element.
- Each interpolation function has a value of one at its corresponding node, and vanishes at the other nodes in the element.
- They sum to one everywhere within the element.
- They have the same order or form as the interpolation equation.

Examples of common 2D and 3D elements, their interpolation equations, and some interpolation functions are shown in Table 1.

element type	# nodes	interpolation equation	interpolation functions
linear triangular area A	3	$\Phi = a_1 + a_2x + a_3y$	$h_1 = [(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y]/2A$ $h_2 = [(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y]/2A$ $h_3 = [(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y]/2A$
bilinear rectangular width w height h area A	4	$\Phi = a_1 + a_2x + a_3y + a_4xy$	$h_1 = (w + x_1 - x)(h + y_1 - y)/A$ $h_2 = (x - x_1)(h + y1 - y)/A$ $h_3 = (w + x_1 - x)(y - y_1)/A$ $h_4 = (x - x_1)(y - y1)/A$
quadratic triangular	6	$\Phi = a_1 + a_2x + a_3y$ $+a_4xy + a_5x^2 + a_6y^2$	*see FEM text*
Lagrangian	9	$\Phi = a_1 + a_2x + a_3y$ $+a_4xy + a_5x^2 + a_6y^2$ $+a_7x^2y + a_8y^2x + a_9x^2y^2$	*see FEM text*
tetrahedral	4	$\Phi = a_1 + a_2x + a_3y + a_4z$	*see FEM text*
20-node brick	20	*see FEM text*	*see FEM text*

Table 1: *Some common 2D and 3D elements and their interpolation functions.*

4.2.3 Expressing the potential energy as a function of node displacements

In the previous section, the interpolation functions were derived for a 2D triangular element. These functions express the value Φ at an element point (x, y) in terms of the values at the element nodes. For displacement-based FEM the same approach is used to express the 3D displacement vector $\mathbf{u}$ of a point (x, y, z) as a linear combination of interpolation functions applied to the node displacements:

$$\mathbf{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{HU}, \tag{8}$$

where

$$\mathbf{U} = (u_1, v_1, w_1, u_2, v_2, w_2, \ldots, u_N, v_N, w_N)^T.$$

N is the number of nodes in the element, $\mathbf{H}$ is matrix of dimension $3 \times 3N$ composed from the interpolation functions, and $\mathbf{U}$ is the composite vector of the node displacements.[2]

The exact form of $\mathbf{H}$ depends on the choice of the element and interpolation functions. However, from (7), the interpolation functions are a function (often polynomial) of the point location, (x, y, z). By applying the differential stress strain relationships of (4) to (8), the strain $\boldsymbol{\varepsilon}$ at (x, y, z), can be expressed in terms of the node displacements and the interpolation functions. The general form of the result is:

$$\boldsymbol{\varepsilon} = \mathbf{BU},$$

[2]The use of a capital, boldface $\mathbf{U}$ to indicate this composite vector is commonly used in FEM to distinguish the $3N \times 1$ composite vector from the 3×1 node displacement vectors. This notation is also adopted here.

where $\mathbf{B}$ is $6 \times 3N$. According to (4), the first row of $\mathbf{B}$ is obtained by differentiating the equation for u from (8) with respect to x, the second row of $\mathbf{B}$ is obtained by differentiating the equation for v from (8) with respect to y, and so on.

Ignoring thermal strain and using (3), the strain energy in the element can be written:

$$\begin{aligned} \Lambda &= \frac{1}{2}\int_V \mathbf{U}^T\mathbf{B}^T\mathbf{D}\mathbf{B}\mathbf{U}\ dV \\ &= \frac{1}{2}\mathbf{U}^T\left(\int_V \mathbf{B}^T\mathbf{D}\mathbf{B}\ dV\right)\mathbf{U} \end{aligned} \tag{9}$$

Equation (9) expresses the element strain energy as a function of the node displacements $\mathbf{U}$. Similarly, using (8), the work term (6) is expanded in terms of the displacements:

$$\begin{aligned} W &= \int_V \mathbf{U}^T\mathbf{H}^T\mathbf{f}_b\ dV + \int_\Gamma \mathbf{U}^T\mathbf{H}^T\mathbf{f}_s\ dV + \mathbf{U}^T\mathbf{P} \\ &= \mathbf{U}^T\left(\int_V \mathbf{H}^T\mathbf{f}_b\ dV + \int_\Gamma \mathbf{H}^T\mathbf{f}_s + \mathbf{P}\right) \\ &= \mathbf{U}^T(\mathbf{F}_b + \mathbf{F}_s + \mathbf{P}) \end{aligned} \tag{10}$$

where $\mathbf{P}$ is a $3N \times 1$ vector derived from $\mathbf{H}$ and the concentrated loads $\mathbf{p}_i$, and $\mathbf{F}_b$ and $\mathbf{F}_s$ are the $3N \times 1$ body and surface force vectors integrated over the object volume and surface respectively.[3] In general, the integral expressions for $\mathbf{F}_b$ and $\mathbf{F}_s$ are approximated using numerical integration.

Substituting (9) and (10) into (2), the potential energy is:

$$\Pi = \frac{1}{2}\mathbf{U}^T\left(\int_V \mathbf{B}^T\mathbf{D}\mathbf{B}\ dV\right)\mathbf{U} + \mathbf{U}^T(\mathbf{F}_b + \mathbf{F}_s + \mathbf{P}), \tag{11}$$

which is a quadratic function of the node displacement vector, $\mathbf{U}$. Minimizing the potential energy with respect to $\mathbf{U}$, by setting the partial derivatives $\partial\Pi/\partial U_i$ equal to zero, yields a linear equation of the form: $\mathbf{K}\mathbf{U} = \mathbf{F}$, where $\mathbf{K}$, the stiffness matrix, is numerically integrated over the volume, and F is the sum of the distributed body forces, surface forces, and concentrated loads.

The above analysis was performed for a single element. To model an object consisting of many elements, the equilibrium expression is derived for each element, and the resulting linear systems for all of the elements are assembled in to a single large, but sparse, linear system of the same form. This linear system can be solved by a number of methods, including Gaussian elimination, or sparse matrix methods.

[3] $\mathbf{F}_b, \mathbf{F}_s$, and $\mathbf{P}$ can be thought of as equivalent forces acting at the FEM node points. As with $\mathbf{U}$, the use of a capital, boldfaced vector representation indicates that the equivalent force vectors are composited over all of the element nodes.

4.3 Dynamic deformation systems

In the discussion of displacement-based finite element methods, a static analysis of the object was considered. The goal of the analysis was to find the new shape of an object subject to a number of distributed and point forces. Animation and other graphics applications often require the dynamic trajectory, that is, the motion of the object as it moves towards its equilibrium shape. In this case, the effects of inertial body forces and energy dissipation through velocity-dependent damping forces are considered, resulting in second order differential equation for the node displacements that is similar to Equation (1):

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F},$$

where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are the mass, damping, and stiffness matrices respectively for the entire object, $\mathbf{F}$ is the composite vector of equivalent applied forces, and $\mathbf{U}$ is the composite vector of node displacements.

The stiffness matrix $\mathbf{K}$ is the assembled matrix of element stiffness matrices described at the end of the preceding section. Similarly, $\mathbf{M}$ is assembled by compositing the mass matrices of individual elements. The mass matrix for each element is defined by expressing the object density in terms of the interpolating functions and integrating over the element volume as was done with the distributed body forces in (10). The resultant mass matrix for a single element is defined by:

$$\mathbf{M} = \int_V \rho \mathbf{H}^T \mathbf{H} \, dV.$$

The damping matrix $\mathbf{C}$ can be calculated similarly by assembling contributions from element damping parameters. However, in general, it difficult to determine these damping parameters and the damping matrix is often constructed as a linear combination of the $\mathbf{M}$ and $\mathbf{K}$ matrices. The dynamic system is usually solved using a numerical integrator.

4.4 Applications

The use of FEM in computer graphics has been limited because of the computational requirements. In particular, it has proven difficult to apply FEM in real-time systems. Because the force vectors and the mass and stiffness matricies are computed by integrating over the object, they must, in theory, be re-evaluated as the object deforms. This re-evaluation is very costly and is frequently avoided by assuming that objects undergo only small deformations.

Celniker and Gossard applied FEM to shape editing in computer-aided design [CG91]. They used 2D triangular surface elements and Hermite polynomials functions (3rd order cubic polynomials) as interpolation functions to model the 3D displacements of the triangle vertices for deforming surfaces. User-controlled external forces are applied to edit the object shape.

Collier *et. al.* use a flat shell element in fabric modeling [CCOS91]. The element they use is a 4-node element with 5 degrees of freedom (position and

in-plane rotation) at each node. This element has been used in material sciences applications for modeling membranes with bending.

Gouret *et. al.* use FEM to model interactions between the soft tissues in a human hand and a deformable object. They use 3D elements with linear interpolation functions and a dynamic formulation to animate the interaction [GMTT89].

Chen and Zeltzer [CZ92] use 20-node brick elements with parabolic interpolation functions to model deformation of muscles and other objects. For muscles, external forces are applied where tendons connect the muscles to rigid, moving bones. Chen and Zeltzer use a small number of brick elements per object (2 elements per muscle, 4 elements for a plasticine head model) and embed the object geometry into the rest state of the elements. When forces are applied, object vertex displacements are calculated from the FEM node displacements. In order to speed up the computation, Chen and Zeltzer use a modal decomposition of the dynamic system as described in Section 6.1.

Essa *et. al.* use dynamic FEM to model shape fitting and motion tracking in image analysis. They use a 27 node unit superquadric FEM element and modal decomposition of the dynamic system to represent the object motion as a function of the vibrational modes of the object [ESP92, ESP93].

Bro-Nielsen, Cotin *et. al.* apply FEM for modeling human tissue deformation for surgical simulation [BN97, BNC96, CDC+96, CDBN+96]. They use a tetrahedral element with linear interpolation functions. In order to accomplish real-time simulation, they perform a number of pre-processing steps. Bro-Nielsen partitions the problem into interior and surface points and solves for deformations only at surface points. In addition, he inverts the stiffness matrix during pre-processing, and tailors the solution method to the assumption that only a small number of contact loads are applied (so that most elements of the composite force vector, F, are zero). Cotin *et. al.* assume superposition and linearity of object deformations. For every surface node, they pre-calculate and store the deformations of all node points and the reflected force at the given node when it is subjected to an infinitesimal load. During the simulation, the applied forces are expressed as linear sums of these infinitesimal loads, and the stored displacements and reflected forces are superimposed to estimate the object deformation. The methods used by Bro-Nielsen and Cotin greatly improve the speed of the simulation with the cost of significant pre-processing and reduced flexibility for objects whose shape or topology change significantly.

4.5 Advantages and limitations

Finite element methods provide a more physically realistic simulation than mass-spring methods with fewer node points, hence requiring the solution of a smaller linear system. However, applied forces must be converted to their equivalent force vectors, which can require numerically integrating distributed forces over the volume at each time step. Because mass and stiffness matrices are derived by numerical integration over the elements, this can lead to significant pre-processing time for finite element methods. If the topology of the object

changes during the simulation, or if the object shape changes beyond small deformation limits, the mass and stiffness matrices must be re-evaluated during the simulation.

The linear elastic theory used to derive the potential energy equation assumes small deformations of the object. Traditional FEM is applied to materials such as metals, where the amount of deformation is limited to less than 1% of the object dimensions. In materials such as human tissue, object dimensions can deform by more than 100% so that the small deformation assumption no longer holds. In particular, as the object deforms, the volumes over which equivalent force and the mass and stiffness matrix integrations are performed, will change. One method to address this problem is to consider a quasi-static system, where it is assumed that mass and stiffness matrices remain constant over a single time interval but they are re-evaluated at each time step. While this provides more physically realistic results, the amount of computation required at each time step is greatly increased.

5 Approximate Continuum Models

The continuum models discussed in this section are termed approximate because certain physical quantities, particularly the deformation energy, are formulated to afford efficient simulation algorithms or to achieve certain desired effects. The models are physically motivated, but adhere less strictly to the laws of physics than some the finite element methods discussed above.

5.1 Snakes

Kass, Witkin, and Terzopoulos introduced active contour models, or "snakes" for solving various problems in computer vision and image analysis [KWT87]. Snakes are one-dimensional deformable curves that are often used to deform or define edges or contours, or to track motion in a moving image. Snakes respond interactively to internal forces that resist stretching and bending, to local image-generated forces, and to user-applied forces.

If $\mathbf{v}(s)$ is the parameterized position of the snake, the internal deformation energy of the snake is expressed as

$$V = \frac{1}{2} \int \left[\alpha(s) \left\| \frac{d\mathbf{v}}{ds}(s) \right\|^2 + \beta(s) \left\| \frac{d^2\mathbf{v}}{ds^2}(s) \right\|^2 \right] ds.$$

The first and second derivative terms correspond to axial and bending deformations, respectively. The terms $\alpha(s)$ and $\beta(s)$ are weighting parameters. Setting $\boldsymbol{\beta}(s_0) = 0$ permits bending at s_0, allowing the snake to develop a corner there. Image energies are integrated over the length of the snake and attract the snake towards features of interest. For example, an image force that attracts the snake towards high gradients will fit the snake around image edges. User-applied external forces allow the user to move the snake interactively away from a local

minima and towards a more desirable solution. If the user attaches a spring from point $\mathbf{v}(s_0)$ on the snake to point $\mathbf{p}$ in the image, a user-applied energy term of the form $k\|\mathbf{v}(s_0) - \mathbf{p}\|^2$ is added to the energy equation.

For a numerical solution, the snake is discretized as a set of points $\{\mathbf{v}_i\}$ along the parametrized curve. Deformation of the snake is simulated using both implicit and explicit Euler integration techniques that minimize the total energy. The derivatives in the energy equation are computed using *finite differences*, a general technique used often when discretizing continuum models:

$$\begin{aligned}\left(\frac{d\mathbf{v}}{ds}\right)_i &= \frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{h} \\ \left(\frac{d^2\mathbf{v}}{ds^2}\right)_i &= \frac{\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}}{h}.\end{aligned}$$

2D snakes have been generalized to active surfaces and active volumes in [CC89, CEO+93, BN95]. These extensions allow smooth surfaces or a 3D mesh of grid points to be fit to 3D image data. These methods have been used for segmentation, model fitting and motion tracking in medical imaging.

5.2 Discretized deformation energy

Terzopoulos *et. al.* developed continuum models for deformable curves, surfaces, and solids for animation applications [TPBF87]. Their method uses a continuum model for the potential energy due to deformation based on fundamental forms. Consider a three-dimensional body parameterized by the coordinate $\mathbf{s} = (s_1, s_2, s_3)$, so $\mathbf{r}(\mathbf{s})$ is the actual position of a particular point of the body in space. If $\mathbf{s}_0$ parameterizes a point on the body, and $\mathbf{s}_1 = \mathbf{s}_0 + d\mathbf{s}$ parameterizes a differentially displaced point, then the squared Euclidean distance between the points is given by

$$\|\mathbf{r}(\mathbf{s}_0) - \mathbf{r}(\mathbf{s}_1)\|^2 = d\mathbf{s}^T\ \mathbf{G}(\mathbf{s}_0)\ d\mathbf{s},$$

where the 3×3 matrix $\mathbf{G}(\mathbf{s}_0)$ is defined by

$$G_{ij}(\mathbf{s}_0) = \frac{\partial \mathbf{r}}{\partial s_i}(\mathbf{s}_0) \cdot \frac{\partial \mathbf{r}}{\partial s_j}(\mathbf{s}_0). \tag{12}$$

$\mathbf{G}$ is called the *first fundamental form* or *metric tensor* of the solid. It is defined at each point in the solid. If two solids have identical metric tensors as functions of $\mathbf{s}$, then the two solids have the same shape and differ only by a rigid body motion. Indeed, rigid-body motions can be defined as exactly those transformations that preserve distance between all pairs of points. Hence, the fundamental form is a way of parameterizing the deformation of an object.

Denoting the metric tensor of the undeformed shape by $\mathbf{G}^0$, the internal potential energy due to deformation can be defined as the functional:

$$V(\mathbf{r}) = \int \|\mathbf{G} - \mathbf{G}^0\|_\alpha^2\ ds_1\ ds_2\ ds_3, \tag{13}$$

where α is a weighted matrix norm, and the integration is over the entire body. This formulation of potential energy does not follow directly from physical principles. Rather, it is a "reasonable" measure that is minimized when the body is in its undeformed state; Terzopoulos *et. al.* were most interested in animation applications, where complete physical accuracy is not required. Using tools of variational calculus and some additional approximations, they use this form of potential energy to derive expressions for the internal forces due to deformation.

In practice, the object is spatially discretized into a mesh of N points, with $3N$ degrees of freedom. At each point, the fundamental form is computed by approximating (12) with finite differences. The integral (13) becomes a finite sum, and the internal deformation forces are computed at each point. The kinetic energy and damping forces are also computed in terms of these points, which are assigned masses based on the object density. Ultimately, Equation (1) is used to compute the dynamics; the main difference between this approach and the mass-spring approaches is the method used to compute the internal forces. This approach can also handle two-dimensional surfaces and one-dimensional space curves. For these objects, there are additional terms in the potential energy expression from other fundamental forms. See [TPBF87] for details.

5.3 Hybrid models

The discrete equations of motion derived from (13) become increasingly ill-conditioned as the model becomes more rigid; this is similar to the stiffness problem with mass-spring systems. Another problem with many deformable body representations is that the state vector is very large, making operations such as shape recognition and pose estimation difficult. Hybrid models combat these problems by breaking a deformable object into two components, one rigid and one deformable.

Terzopoulos and Witkin represented a deformable body by a rigid reference body that captures the rigid-body motion, and a discretized displacement function that gives the location of the mesh nodes relative to their positions on the undeformed reference body [TW88]. The reference body's motion is computed from standard rigid-body dynamics. The reference body eliminates the need to capture the reference shape in the potential energy functional (13) via $\mathbf{G}_0$. Rather, a "reasonable" potential energy can be expressed as a weighted sum of squared magnitudes of low order partial derivatives of the displacement function. This potential energy functional is minimized when the displacement function is identically zero, corresponding to no deformation, and increases monotonically as deformation increases. Using this hybrid approach, the discrete equations of motion become better conditioned as the rigidity of the object increases.

Terzopoulos and Fleischer generalized this approach to handle viscoelasticity, plasticity, and fracture in deformable bodies. This is accomplished by using more elaborate models for internal forces due to deformation. For viscous behavior, the internal forces depend on the time derivatives of the displacement function, that is, the velocities of the mesh points, as well as on the spatial partial derivatives. Plastic behavior considers the time history of the deformations, and

fracture is modeled by breaking connections between mesh points by removing interdependencies in the equations of motion. Details are in [TF88b, TF88a].

Terzopoulos and Metaxas proposed deformable superquadric ellipsoids as useful models for image analysis [TM91, Met97]. They first parameterize the shape of a (rigid) superquadric ellipsoid by six parameters: a scale parameter a, aspect ratio parameters a_1, a_2, and a_3, and squareness parameters ε_1 and ε_2. The rigid portion of the representation also includes the standard rigid body degrees of freedom. As in [TW88], the positions of model points are the vector sum of their positions on the rigid body model, and an offset described by a displacement function. The complete state of a deformable superquadric ellipsoid is thus given by a vector $\mathbf{q} = (\mathbf{q}_c, \mathbf{q}_\theta, \mathbf{q}_s, \mathbf{q}_d)$, where $\mathbf{q}_c$ and $\mathbf{q}_\theta$ are the translational and rotational rigid body degrees of freedom, $\mathbf{q}_s$ is the vector of global shape parameters, and $\mathbf{q}_d$ parameterizes the displacement function in terms of a basis of functions. The kinetic and potential energies of the object are expressed in terms of the generalized coordinates $\mathbf{q}$. If $\mathbf{d}(u, v)$ is the displacement function over the surface of the ellipsoid, then the potential energy of deformation is approximated by a spline deformation energy functional:

$$V(\mathbf{d}) = \int \left[w_1 \left(\left(\frac{\partial \mathbf{d}}{\partial u} \right)^2 + \left(\frac{\partial \mathbf{d}}{\partial u} \right)^2 \right) + w_0 \mathbf{d}^2 \right] \, du \, dv,$$

where w_0 and w_1 and are weighting functions that scale the energy terms due to magnitude and variation of the deformation. Once the kinetic and potential energies are specified, the methods of Lagrangian dynamics (see, for example, [Gol80]) generate the equations of motion of the system, in the form given by (1). For computer vision and model fitting applications, the forces acting on the dynamic model come from features of a two-dimensional image or three-dimensional range map. Through dynamic simulation, the parameters of the deformable superquadric ellipsoid evolve until they reach a minimum energy state, fitting the deformable model to the data. Several example experiments are presented in [TM91]. Metaxas and Terzopoulos extended this basic approach by using Kalman filtering methods in conjunction with the dynamic deformable superquadric ellipsoids [MD91]. With the Kalman filtering, it was possible to track shapes in noisy, time-varying image data at interactive rates. Finally, [MT92] describes how to add constraints to the basic model for animation applications.

6 Low degree of freedom models

Discretization of the physically based models discussed above leads to systems with many degrees of freedom since the object's state is characterized by the positions and velocities of a large number of node points. While these general systems support a rich variety of deformations, the systems are slow to simulate, limiting their use in interactive and real time settings. Alternative approximate continuum models have been proposed that restrict the deformable object to many fewer degrees of freedom, sacrificing generality for speed.

6.1 Modal analysis

Pentland and Williams developed a simplified expression for the dynamics of deformable bodies using modal analysis [PW89]. Given the mass, damping, and stiffness matricies, $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ of Equation (1), there exists a matrix $\boldsymbol{\phi}$ and a diagonal matrix $\boldsymbol{\lambda}$ such that

$$\boldsymbol{\phi}\boldsymbol{\lambda} = \mathbf{M}^{-1}\mathbf{K}\boldsymbol{\phi}.$$

The columns of $\boldsymbol{\phi}$ are the generalized eigenvectors of $\mathbf{M}$ and $\mathbf{K}$, and the diagonal elements of $\boldsymbol{\lambda}$ are the generalized eigenvalues. Because $\mathbf{M}$ and $\mathbf{K}$ are normally symmetric positive definite, they can be simultaneously diagonalized by $\boldsymbol{\phi}$:

$$\begin{aligned} \boldsymbol{\phi}^T\mathbf{M}\boldsymbol{\phi} &= \tilde{\mathbf{M}} \\ \boldsymbol{\phi}^T\mathbf{K}\boldsymbol{\phi} &= \tilde{\mathbf{K}} \end{aligned}$$

where $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{K}}$ are diagonal. When $\mathbf{C}$ is a linear combination of $\mathbf{M}$ and $\mathbf{K}$, which is typical, it can also be diagonalized by $\boldsymbol{\phi}$. Multiplying through Equation (1) by $\boldsymbol{\phi}^T$, and using the transformation $\mathbf{x} = \boldsymbol{\phi}\tilde{\mathbf{x}}$, yields:

$$\boldsymbol{\phi}^T\mathbf{M}\boldsymbol{\phi}\ddot{\tilde{\mathbf{x}}} + \boldsymbol{\phi}^T\mathbf{C}\boldsymbol{\phi}\dot{\tilde{\mathbf{x}}} + \boldsymbol{\phi}^T\mathbf{K}\boldsymbol{\phi}\tilde{\mathbf{x}} = \boldsymbol{\phi}^T\mathbf{f},$$

and defining $\tilde{\mathbf{f}} = \boldsymbol{\phi}^T\mathbf{f}$ gives

$$\tilde{\mathbf{M}}\ddot{\tilde{\mathbf{x}}} + \tilde{\mathbf{C}}\dot{\tilde{\mathbf{x}}} + \tilde{\mathbf{K}}\tilde{\mathbf{x}} = \tilde{\mathbf{f}}.$$

In this form, the system equations are linearly independent and each equation describes a vibrational mode of the object. Each element of $\tilde{\mathbf{x}}$ is a modal displacement corresponding to a combination of displacements in $\mathbf{x}$. The diagonal elements of $\boldsymbol{\lambda}$ are proportional to the resonant frequencies of the vibration modes.

In 3D systems, 6 of the vibrational modes account for rigid body motion (object position and orientation). Additional modes account for linear strain, quadratic strain and higher order deformations. In most animation and computer graphics applications, it is not necessary to simulate all of the vibrational modes. Higher order modes generally have smaller displacement amplitudes and their resonant frequencies can be higher than animation frame rates. In order to reduce the number of degrees of freedom in the system to enable faster simulation, one can often disregard higher frequency modes. It is also possible to order the vibrational modes so that, in situations where speed is critical, only the rigid body, linear strain, and quadratic strain modes are simulated, and, in situations where accuracy is important, additional motion detail can be added systematically by including more higher frequency modes in the analysis. Thus, a modal representation is ideal for controlling level of detail in deformable object simulation.

6.2 Dynamic global deformations

Witkin and Welch developed a technique for animating and controlling deformable bodies that experience global deformations [WW90]. These deformations are similar to Barr's space-warping functions (Section 2.2) in that they are characterized by a map from $\Re^3$ to $\Re^3$. However, Witkin and Welch also introduce a time dependence into the deformation in order to animate deformable objects. Specifically, if $\mathbf{x}$ is the location of a point on the undeformed object, the location of the point on the deformed object is

$$f(\mathbf{x}, t) = \mathbf{R}(t)\mathbf{p}(\mathbf{x}). \tag{14}$$

The vector $\mathbf{p}$ is some function of the position coordinate $\mathbf{x}$ and does not depend on time. An example is $\mathbf{p}(\mathbf{x}) = (x_1, x_1^2, x_2 x_3, 1)$. The elements of the matrix $\mathbf{R}$ form the generalized coordinates of the deformable object. These vary with time and are what cause the object to move over time. If one expresses the object's kinetic and potential energies in terms of these coordinates, the Lagrangian formulation provides a straightforward method of computing the dynamics of the object, that is, how the coordinates change over time. Details of the method are in [Gol80].

The kinetic energy is simple to compute. Differentiating (14) with respect to time gives the velocity $\dot{\mathbf{x}}$ of a point $\mathbf{x}$, and summing the $m_i\|\dot{\mathbf{x}}_i\|^2/2$ terms over a set of discretized mass points gives an approximation of the kinetic energy. The kinetic energy can also be computed exactly using an integral involving the mass density and velocity of an arbitrary point.

The potential energy formulation is derived in a less formal manner, depending on what sort of behavior is desired; different potential energy functions give rise to different deformation dynamics. Witkin and Welch give some examples of potential energy functions that can be used when (14) is of a special form. One example is a potential energy function that is zero when the deformation is volume preserving.

Within restricted classes of deformation functions, the dynamics computations can be accelerated by precomputing certain quantities. There are also ways to incorporate constraints into the formulation, which provide a high level way to manipulating the objects. See [WW90] for details. Baraff and Witkin used this basic approach in simulating combinations of rigid and deformable bodies with non-penetration constraints [BW92].

6.3 Minimal Energy surfaces

Several researchers have explored ways to add physical behavior to the traditional geometric modeling primitives, particularly parametric surface patches. This is done by minimizing an energy functional defined on the surface. If $\mathbf{w}(u, v)$ is a parametric surface defined over the domain Γ, one reasonable energy functional based on local stretching and bending is

$$E = \int_\Gamma (\alpha_{11}\|\mathbf{w}_u\|^2 + 2\alpha_{12}\mathbf{w}_u \cdot \mathbf{w}_v + \alpha_{22}\|\mathbf{w}_v\|^2 + \beta_{11}\|\mathbf{w}_{uu}\|^2 + 2\beta_{12}\|\mathbf{w}_{uv}\|^2 + \beta_{22}\|\mathbf{w}_{vv}\|^2 - 2\mathbf{f} \cdot \mathbf{w})\, du\, dv, \quad (15)$$

where the α_{ij} and β_{ij} are weighting coefficients, and the subscripts of $\mathbf{w}$ indicate partial derivatives. The vector $\mathbf{f}$ encodes user-applied sculpting forces that modify the surface in predictable ways. The user may also add constraints for the surface to satisfy, such as a requirement that the surface interpolate certain points or curves in space.

One restricts the surface to some class specified by a finite set of parameters, such as control point locations and weights. These parameters form the state vector $\mathbf{x}$, and the methods of constrained optimization are used to find a state vector that minimizes the energy functional while satisfying the constraints. There are significant computational advantages when the constraints are linear in the state variables.

Celniker and Welch present a formulation that uses B-spline tensor product surfaces. The constrained minimum of (15) occurs at the value of $\mathbf{x}$ that solves an equation of the form $\mathbf{Kx} = \mathbf{f}$. This solution is the final equilibrium position of the surface, however dynamic effects are easily incorporated by adding mass and damping terms. Then one obtains Equation (1) with diagonal $\mathbf{M}$ and $\mathbf{C}$ matrices. That equation can be numerically integrated to cause the surface to evolve over time [CW92]. Welch and Witkin further developed this idea by adding automatic refinement to the algorithm. Nonuniform subdivision is applied to the B-spline surface to ensure that the constraints are met, and to enforce error bounds. The user manipulates a surface that appears "infinitely malleable" by defining control points and curves that the surface interpolates [WW92]. Terzopoulos and Qin developed D-NURBS: physically based, dynamic versions of NURBS curves and surfaces. Their approach also entails minimizing surface energy in the presence of applied forces and constraints. D-NURBS curves can be used to trim D-NURBS surfaces [TQ94].

7 Conclusions

This paper has presented many of the techniques that have been used in computer graphics for modeling deformable objects. These techniques range from the manual editing of individual vertex or control points, to finite element methods that approximate the full continuum mechanics model of object deformation. These methods have been applied to many areas of computer graphics, including: shape editing, cloth modeling, object and character animation, image analysis, and surgical simulation. While significant advances have been made, there are a number of important areas for future work. These include: validation of physically accurate deformation, achieving realistic real-time deformation of complex objects, and integrating deformable modeling techniques into broader contexts.

Many papers have recognized the importance of obtaining realistic material parameters and of validating the results of simulated deformations. While applications such as character animation may tolerate or even desire unrealistic deformations, most applications require the best accuracy possible for the available computational power. In applications such as surgical simulation, where it is envisioned that computer simulators will eventually replace animal models for surgical training, it is especially important that tissue modeling be accurate. However, it is often difficult to obtain accurate material properties and it is even more difficult to conduct the kinds of experiments required to verify the accuracy of computer simulations. Material properties have been obtained in fabric modeling by mechanical testing, for example by Breen *et. al.*[BHG92, BHW94]. Hunter *et. al.* acquire material properties of the eye for FEM using both mechanical and light interference techniques [HDL+93]. In most of the work described in the literature, careful validation was not performed, although there are exceptions. In apparel design, Okabe *et. al.* compare simulation results to the shape of real draped fabric [OITN92]. Pinsky and Datye measure some material properties of the eye and compare simulation results to the results predicted by a number of surgeons [PD91]. Keeve *et. al.* compare the results of a surgical simulation of the deformation of soft facial tissues after cranio-facial surgery to actual post-operative appearance [KGPG96]. While such qualitative verification methods are useful, quantitative comparisons between simulation and reality are necessary for evaluating and refining the various models for deformable objects.

Many computer graphics applications for deformable objects require real time speeds. Applications such as surgical simulation also require highly accurate deformation. Because human tissues often undergo large deformations and their material properties are complex, continuum models that can handle nonlinear, large-scale deformation may be required. For this reason, there is need to develop techniques for deformable object modeling that are fast and accurate. As discussed in Section 4.4, Bro-Nielson and Cotin use several pre-processing methods to enable real-time interaction with their 3D FEM models. Cotin has reported obtaining user-controlled deformation of an 8000 node FEM system at up to 15 frames per second[4]. However, many hours of pre-processing are required to attain these rates. If the object topology changes (for example by cutting the tissue), if the environmental constraints change, or if shape changes are larger than small-deformation limits, then the system must be re-analyzed and the pre-processing step must be repeated. Promising methods for improving the speed of deformable modeling include parallel hardware systems, intelligent model simplification, and hierarchical techniques. Hierarchical techniques approximate the deformation from a coarse to fine scale to improve convergence or response time without sacrificing too much realism. Examples of these hierarchical techniques are: modal analysis [PW89, ESP92, CZ92]; multigrid finite element methods, where the object deformation is analyzed from a coarse to fine grid; and combined kinematic-dynamic models. An example of the latter is Gibson's *ChainMail* algorithm, in which kinematic inequality constraints de-

[4] personal communication, March, 1997

termine the coarse motion of a discretized deformable object, and a dynamic mass-spring model is then used to refine the final shape of the object [Gib97]. The algorithm supports real-time deformation of an object with as many as 125,000 elements.

Finally, there is a need to integrate deformable objects into broader simulation contexts. While great strides have been made in deformable object modeling, the deformable object is not an isolated system. In surgery, deformable tissue might be constrained by a rigid clamp. In engineering analysis, a thin membrane might be deformed by a pressure difference. In reality, a deformable object might contact many other types of objects: other deformable objects, virtually rigid objects, fluids, and so on. Important problems exist at the boundaries where these various objects meet. Collision detection is an important problem that is required for any type of interaction between objects; efficient collision detection algorithms are needed for deformable objects. Modeling forces between deformable objects and their environments also needs further study. Some systems have taken initial steps toward solving these challenging problems, for example [BW92, PW89, TT94], however, faster and more accurate algorithms are still needed. The solutions to these problems will make existing simulation systems more flexible and powerful than they are today. The solutions will also offer the user new interaction modalities, significantly enhancing the computer's utility as a design and modeling tool for deformable objects.

References

[AWW89] J. Allen, B. Wyvil, and I. Witten. A method for direct manipulation of polygon meshes. In *Proceedings of Computer Graphics International*, pages 451–469, 1989.

[Bar84] A. Barr. Global and local deformations of solid primitives. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 84, pages 21–30. ACM SIGGRAPH, 1984.

[Bat96] K-J. Bathe. *Finite Element Proceedures.* Prentice Hall, Englewood Cliffs, NJ, 1996.

[BB89] R. Bartels and J. Beatty. A technique for the direct manipulation of spline curves. In *Proceedings Graphics Interface*, 1989.

[BBB87] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann, Los Altos, 1987.

[BHG92] D. Breen, D. House, and P. Getto. A physically-based particle model of woven cloth. *The Visual Computer*, 8:264–277, 1992.

[BHW94] D. Breen, D. House, and M. Wozny. Predicting the drape of woven cloth using interacting particles. In *Computer Graphics Proceed-*

ings, Annual Conference Series, Proceedings of SIGGRAPH 94, pages 365–372. ACM SIGGRAPH, 1994.

[BN95] M. Bro-Nielsen. Modeling elasticity in solids using active cubes - application to simulated operations. *Computer Vision, Virtual Reality and Robotics in Medecine*, pages 535–541., 1995. Vol. 905 of Lecture Notes in Computer Science.

[BN97] M. Bro-Nielsen. Fast finite elements for surgery simulation. *Medicine Meets Virtual Reality V*, 1997.

[BNC96] M. Bro-Nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proceedings of Eurographics*, volume 15, pages 57–66, 1996.

[BW92] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 92, pages 303–308. ACM SIGGRAPH, 1992.

[CC89] L. Cohen and I. Cohen. Finite element methods for active contour models and balloons for 2D and 3D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, 1989.

[CCOS91] J. Collier, B. Collier, G. O'Toole, and S. Sargand. Drape prediction by means of finite-element analysis. *Journal of the Textile Institute*, 82(1):96–107, 1991.

[CDBN+96] S. Cotin, H. Delingette, M. Bro-Nielsen, N. Ayache, J. Clement, V. Tassetti, and J. Marexcaus. Geometric and physical representations for a simulator for hepatic surgery. In H. Sieburg, S. Weghorst, and K. Morgan, editors, *Health Care in the Information Age*, pages 139–151. IOS Press and Ohmsha, 1996.

[CDC+96] S. Cotin, H. Delingette, J. Clement, L. Soler, N. Ayache, and J. Marescaux. Geometrical and physical representations for a simulator of hepatic surgery. In *Proceedings of Medicine Meets Virtual Reality IV*, 1996.

[CEO+93] S. Cover, N. Ezquerra, J. O'Brian, R. Rowe, T. Gadacz, and E. Palm. Interactively deformable models in surgical simulation. *IEEE Computer Graphics and Applications*, pages 68–75, 1993.

[CG91] G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 91. ACM SIGGRAPH, 1991.

[CHP89] J. Chadwick, D. Haumann, and R. Parent. Layered construction for deformable animated characters. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 89, pages 243–252. ACM SIGGRAPH, 1989.

[CMN97] J. Christensen, J. Marks, and J. T. Ngo. Automatic motion synthesis for 3D mass-spring models. *The Visual Computer*, 13:20–28, 1997.

[Coq90] S. Coquillart. Extending free-form deformation: a sculpting tool for 3D geometric modeling. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 90, pages 187–196. ACM SIGGRAPH, 1990.

[CR94] Y. Chang and A. Rockwood. A generalized de Casteljau approach to 3D free-form deformation. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 94, pages 257–260. ACM SIGGRAPH, 1994.

[CW92] G. Celniker and W. Welch. Linear constraints for non-uniform B-spline surfaces. In *Proceedings of Symposium on Interactive 3D Graphics*, 1992.

[CZ92] D. Chen and D. Zeltzer. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 92, pages 89–98. ACM SIGGRAPH, 1992.

[ESP92] I. Essa, S. Sclaroff, and A. Pentland. A unified approach for physical and geometric modeling for graphics and animation. In *Proceedings of Eurographics*, volume 11, pages C129–C138, 1992.

[ESP93] I. Essa, S. Sclaroff, and A. Pentland. Physically-based modeling for graphics and vision. In R. Martin, editor, *Directions in Geometric Computing*, U.K., 1993. Information Geometers.

[Far90] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Inc., San Diego, second edition, 1990.

[FS94] A. Finkelstein and D. Salesin. Multiresolution curves. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 94, pages 262–268. ACM SIGGRAPH, 1994.

[Gib97] S. Gibson. 3Dchainmail: a fast algorithm for deforming volumetric objects. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 149–154, 1997.

[GMTT89] J.P. Gourret, N. Magnenat-Thalmann, and D. Thalmann. Simulation of object and human skin deformations in a grasping task. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 89, pages 21–30. ACM SIGGRAPH, 1989.

[Gol80] H. Goldstein. *Classical Mechanics*. Addison Wesley, Reading, MA, second edition, 1980.

[GP89] J. Griessmair and W. Purgathofer. Deformation of solids with trivariate B-splines. In *Proceedings of Eurographics*, pages 20–27, 137–148. North-Holland, 1989.

[HDL+93] I. Hunter, T. Doukoglou, S. Lafontaine, P. Charette, L. Jones, M. Sagar, G. Mallinson, and P. Hunter. A teleoperated microsurgical robot and associated virtual environment for eye surgery. *Presence*, 2(4):265–280, 1993.

[HHK92] W. Hsu, J. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 92, pages 177–184. ACM SIGGRAPH, 1992.

[HM90] B. Hinds and J. McCartney. Interactive garmet design. *The Visual Computer*, 6:53–61, 1990.

[JV84] G. Jared and T. Varady. Synthesis of volume modeling and sculptured surfaces in BUILD. In *Proceedings of Computers in Design Engineering*, pages 481–495, 1984.

[KGPG96] E. Keeve, S. Girod, P. Pfeifle, and Bernd Girod. Anatomy-based facial tissue modeling using the finite element method. In *Proceedings of IEEE Visualization*, pages 21–28, 1996.

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International J. Computer Vision*, 1(4):321–332, 1987.

[LTW93] Y. Lee, D. Terzopoulos, and K. Waters. Constructing physics-based facial models of individuals. In *Proceedings of Graphics Interface*, pages 1–8, May 1993.

[LTW95] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 95, pages 55–62. ACM SIGGRAPH, 1995.

[MD91] D. Metaxas and D.Terzopoulos. Recursive estimation of shape and nonrigid motion. In *Proceedings of IEEE Workshop on Visual Motion*, pages 306–311. IEEE Computer Society Press, 1991.

[Met97] D. Metaxas. *Physics-based Deformable Models: Applications to Computer Vision, Graphics, and Medical Imaging.* Kluwer Academic, Boston, 1997.

[MJ96] R. MacCracken and K. Joy. Free-form deformations with lattices of arbitrary topology. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 96, pages 181–188. ACM SIGGRAPH, 1996.

[MT92] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 92, pages 309–312. ACM SIGGRAPH, 1992.

[MT96] T. McInerny and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[NG96] H. Ng and R. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics and Applications*, 16(5):28–41, 1996.

[OITN92] H. Okabe, H. Imaoka, T. Tomiha, and H. Niwaya. Three dimensional apparel CAD system. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 92, pages 105–110. ACM SIGGRAPH, 1992.

[Par77] R. Parent. A system for sculpting 3D data. *Computer Graphics*, 11:138–147, 1977.

[PB81] S. Platt and N. Badler. Animating facial expressions. *Computer Graphics*, 15(3):245–252, 1981.

[PD91] P. Pinsky and D. Datye. A microstructurally based finite element model of the incised human cornea. *Journal of Biomechanics*, 24(10):907–922, 1991.

[Pie89a] L. Piegle. Modifying the shape of rational B-splines. part 1: Curves. *Computer Aided Design*, 21(8):509–518, 1989.

[Pie89b] L. Piegle. Modifying the shape of rational B-splines. part 2: Surfaces. *Computer Aided Design*, 21(9):538–546, 1989.

[PW89] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 89, pages 215–222. ACM SIGGRAPH, 1989.

[Seg84] L. Segerlind. *Applied Finite Element Analysis.* John Wiley and Sons, New York, 1984.

[SP86] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 86, pages 151–160. ACM SIGGRAPH, 1986.

[TF88a] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[TF88b] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 88, pages 269–278. ACM SIGGRAPH, 1988.

[TM91] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: deformable superquadrics. *IEEE Transactions on Patern Analysis and Machine Intelligence*, 13(7):703–714, 1991.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 87, pages 205–214. ACM SIGGRAPH, 1987.

[TPF89] D. Terzopoulos, J. Platt, and K. Fleischer. From gloop to glop: heating and melting deformable objects. In *Proceedings of Graphics Interface*, pages 219–226, 1989.

[TQ94] D. Terzopoulos and H. Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(3):103–136, 1994.

[TT94] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 94, pages 43–50. ACM SIGGRAPH, 1994.

[TW88] D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, pages 41–51, November 1988.

[TW90] D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis, and animation. *Journal of Visualization and Computer Animation*, 1:73–80, 1990.

[Wat87] K. Waters. A muscle model for animating three-dimensional facial expression. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 87, pages 17–24. ACM SIGGRAPH, 1987.

[Wat92] K. Waters. A physical model of facial tissue and muscle articulation derived from computer tomography. In *Proceedings of Visualization in Biomedical Computing*, volume 1808, pages 574–583. SPIE, 1992.

[WT91] K. Waters and D. Terzopoulos. Modeling and animating faces using scanned data. *Visualization and Computer Animation*, 2:123–128, 1991.

[WW90] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 90, pages 243–252. ACM SIGGRAPH, 1990.

[WW92] W. Welch and A. Witkin. Variational surface modeling. In *Computer Graphics Proceedings, Annual Conference Series*, Proceedings of SIGGRAPH 92, pages 157–166. ACM SIGGRAPH, 1992.