M.I.T. Media Laboratory Perceptual Computing Group Technical Report No. 184, August 1992,
Appears as a chapter in *Directions in Geometric Computing*, Ralph Martin, Editor, 1993,
Publisher: Information Geometers, U.K.

Physically-based Modeling for Graphics and Vision

Irfan A. Essa, Stan Sclaroff and Alex Pentland
Perceptual Computing Group, The Media Laboratory
Massachusetts Institute of Technology
Room E15-383, 20 Ames Street,
Cambridge MA 02139, U.S.A.

1 Introduction

The elastic properties of materials constrain the motion and dynamics of objects in the real world, hence modeling and simulating the physical characteristics of these objects is essential to obtain realistic computer modeling for graphics, vision and animation. This type of modeling is referred to as *physically-based modeling* and is the main focus of this chapter.

A major problem for physically-based modeling has been to establish a unified approach for representing geometry that allows efficient calculation of both geometric relations (e.g., surface-to-point distances) and the non-rigid physical response to imposed forces. This problem has been addressed by researchers in both computer graphics [7, 38, 31, 44], and machine vision [40, 12, 25, 41, 30, 28].

In computer graphics, the central goal is efficient detection of collisions and simulation of the resulting non-rigid motion. In machine vision the hope is to develop general purpose models that are capable of accurately representing the free-form shapes and nonrigid motions of natural objects. As part of this chapter, we will present a general overview of this past research and then present our approach to 3-D physical modeling in graphics and vision.

Our approach to geometric modeling is based on the use of parameterized implicit functions, particularly polynomial deformations of superquadric functions. The physical model is developed by discretizing the geometric model via the finite element method, making use of Galerkin polynomial approximations for shape interpolation. The geometric and physical models are then combined into a unified model by using the finite element modal shape functions as polynomial deformations applied to an underlying superquadric implicit function. These polynomial deformations are known as the object's "deformation modes" and are derived directly from the finite element equations.

This approach significantly reduces the computational cost of both geometric distance calculations and modeling of physical response. By describing the finite element equations in terms of the object's free vibration modes, the system of equations is decoupled into a sum of independent, frequency-ordered deformations. This not only allows closed-form solutions of the finite element equations, but also allows us to smoothly trade accuracy against computational cost. Similarly, by using an implicit function representation for geometry we can greatly reduce the computational cost of contact detection and collision response. By combining the vibration mode deformations with an underlying implicit function representation, we obtain both of these advantages.

A shorter and earlier version of this paper has appeared as "A Unified Approach for Physical and Geometric Modeling for Graphics and Animation", in *Computer Graphics Forum, The International Journal of the Eurographics Association*. 2(3), 1992, Editors: Alistar Kilgour and Lars Kjelldahl, Publisher: Blackwell, U.K.

After describing our approach to physically-based modeling, the final sections of this chapter will present examples of modeling and simulating complex objects. Examples of automatic generation of these models using computer vision techniques will also be presented.

All of the concepts presented in this chapter have been implemented in the ThingWorld Modeling System [27], which serves as the software base for our research in physically-based modeling for graphics, vision, and animation. The ThingWorld system is implemented in Lisp and C on Sun and DEC workstations, and is available free to degree-granting institutions.

2 Background

The interactive modeling and simulation of rigid and nonrigid object dynamics has long been a major goal in computer graphics and animation, starting with Sutherland's Sketchpad [37] and Borning's Thinglab [10]. In recent years there has been a profusion of new computer graphics research [7, 38, 25, 44] aimed at computationally efficient modeling of deformable geometric models. As the elastic properties of materials constrain the motion and dynamics of objects, modeling and simulation of these physical characteristics is required for realistic modeling.

Implicit and parametric function representations have long been common in the field of computer graphics because of their inherent advantages in contact detection, rendering, and ease in applying polynomial deformations. Because such representations can express a relatively limited range of geometry, researchers have concentrated on generalizing them by use of polynomial deformations. This includes both general free-form deformations as described by Sederberg *et. al.* [33] and Barr [7], and the simpler affine transformations of Witkin [44].

Barr's approach to generalizing implicit functions [7, 5, 6], which uses global and local deformations of a superquadric implicit function, has become widely used because it allows the hierarchical solid modeling operations useful in computer aided design. This type of representation was introduced into computer vision by Pentland [23], and since has been the basis of much work in object recognition [23, 24, 2, 25, 41, 20].

Terzopoulos *et. al.* [38, 39] introduced deformable elastic models to the graphics and vision communities, by applying the principles of elasticity found in mechanics to the problem of animating and describing 3-D objects. His model is based on a mesh-like finite-difference surface model. Terzopoulos [40], Witkin [44], Ayache and Cohen [12], and many others elaborated on this approach, by incorporating geometric and dynamic constraints and kinematic control. A major drawback of this approach is the huge computational expense required to calculate dynamic interactions, which prevented interactive simulations except in simple situations.

Pentland and Williams [31] used the finite element method of *modal analysis* to define a new set of polynomial deformations that has the advantage that it allows very efficient approximate physical simulation, because the polynomials are derived from the eigenvectors of the finite element equations themselves. This was the first attempt to combine the domains of geometric deformation and physical simulation in a way that reduced the computational complexity of the physically-based modeling approach [26]. More recently Pentland *et. al.* [30, 28, 29] have used this approach for machine vision, including shape extraction, object recognition, and tracking of nonrigid motion. Some extensions of this work are presented in this chapter.

This chapter extends Pentland and Williams' work by characterizing an object's deformations directly, using the Galerkin polynomial shape functions [34] which define the object's finite element model. As in Sclaroff and Pentland [32], these polynomial deformations will then be used to form a generalized implicit function representation, allowing for very efficient contact detection and collision response. Displacement

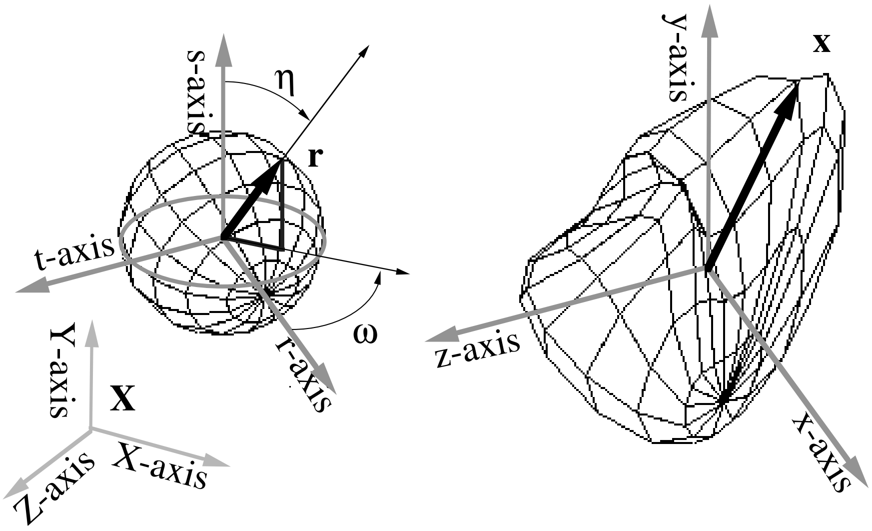


Figure 1: *Coordinate Systems:* $\mathbf{X} = [X, Y, Z]^T$ *is the global Cartesian coordinates,* $\mathbf{x} = [x, y, z]^T$ *is the object-centered local coordinates, and* $\mathbf{r} = [r, s, t]^T$ *object-centered local coordinates before parametric deformation. The vector* $\mathbf{r}$ *sweeps out the surface of a unit-sized superquadric.*

mapping for generalized implicit representations will also be integrated into this representation, thus allowing high-quality animations of complex 3-D objects.

Recent work by Terzopoulos and Metaxas [41] also employs superquadrics, with a finite element mesh wrapped around.

3 The Underlying Geometry: Parametric and Implicit Functions

With increasing interest in numerical methods for three dimensional design and representation, parametric shape representations are assuming greater importance in computer graphics, animation, vision and related disciplines. These parametric representations provide an intuitively appealing mathematical formulation that is easily adapted to imaging, finite element modeling, and other common numerical problems [5, 31, 43]. In this chapter we will use the *superquadric functions*, together with polynomial deformations based on the object's physical vibration modes, to describe object geometry.

3.1 Superquadric Representation

Superquadrics are a family of parametric shapes that were invented by a Danish designer Peit Hein as an extension of basic quadric surfaces and shapes [2, 5, 6, 24]. Superquadrics are a type of three-dimensional parametric shape function that is developed from two 2-D parametric curves,

$$\mathbf{h}(\omega) = \begin{bmatrix} h_1(\omega) \\ h_2(\omega) \end{bmatrix}; \quad \omega_0 \leq \omega \leq \omega_1 \quad , \quad \mathbf{m}(\eta) = \begin{bmatrix} m_1(\eta) \\ m_2(\eta) \end{bmatrix}; \quad \eta_0 \leq \eta \leq \eta_1 \ . \tag{1}$$

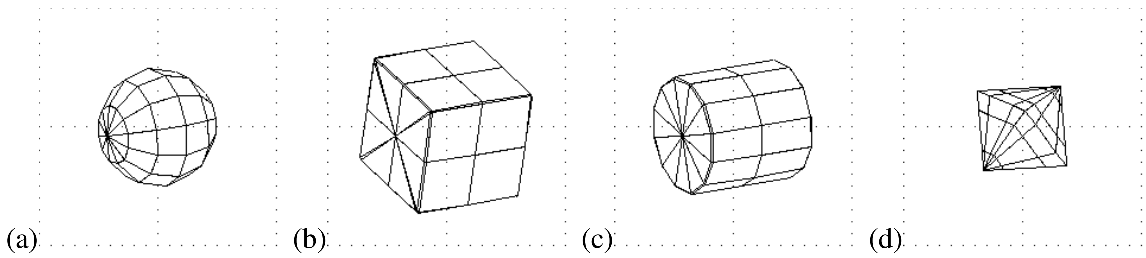


Figure 2: *Superquadrics with various squareness values (a)* $e1 = 1.0$, $e2 = 1.0$, *(b)* $e1 = 0.05$, $e2 = 0.05$, *(c)* $e1 = 1.0$, $e2 = 0.05$, *and (d)* $e1 = 2.0$, $e2 = 2.0$.

The spherical product of these two curves, $\mathbf{r} = \mathbf{m} \otimes \mathbf{h}$ gives the three dimensional surface. Geometrically, $\mathbf{h}(\omega)$ is a horizontal curve vertically modulated by $\mathbf{m}(\eta)$ where $m_1(\eta)$ changes the relative scale of $\mathbf{h}$ while $m_2(\eta)$ raises and lowers it. η is a north-south parameter, like latitude, and ω is a east-west parameter, like longitude, as illustrated in Figure 1. The spherical product can be scaled by a vector $\mathbf{A} = [a_1, a_2, a_3]^T$ that defines the sizes of the object in three directions [5, 6, 7].

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} a_1 m_1(\eta) h_1(\omega) \\ a_2 m_1(\eta) h_2(\omega) \\ a_3 m_2(\eta) \end{bmatrix}; \quad \begin{array}{c} \eta_0 \leq \eta \leq \eta_1 \\ \omega_0 \leq \omega \leq \omega_1 \end{array} . \tag{2}$$

The superquadric surface is defined by picking the following 2-D parametric curves,

$$\mathbf{h}(\omega) = \begin{bmatrix} \cos^{e_2}(\omega) \\ \sin^{e_2}(\omega) \end{bmatrix}; \quad -\pi \leq \omega \leq \pi \quad , \quad \mathbf{m}(\eta) = \begin{bmatrix} \cos^{e_1}(\eta) \\ \sin^{e_1}(\eta) \end{bmatrix}; \quad \frac{-\pi}{2} \leq \eta \leq \frac{\pi}{2} . \tag{3}$$

Therefore the unit superquadric surface is defined by:

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} r \\ s \\ t \end{bmatrix} = \begin{bmatrix} \cos^{e_1}(\eta)\cos^{e_2}(\omega) \\ \cos^{e_1}(\eta)\sin^{e_2}(\omega) \\ \sin^{e_1}(\eta) \end{bmatrix}; \quad \begin{array}{c} -\pi \leq \omega \leq \pi \\ \frac{-\pi}{2} \leq \eta \leq \frac{\pi}{2} \end{array} . \tag{4}$$

Where e_1 and e_2 are parameters that control the "squareness" of the surface's profile, as illustrated in Figure 2.

3.1.1 Surface normals of superquadric functions

For both geometric and physical modeling, the surface normals are required. Normals to the two parametric curves in Equation 3 are:

$$\mathbf{n_h}(\omega) = \begin{bmatrix} \cos^{2-e_2}(\omega) \\ \sin^{2-e_2}(\omega) \end{bmatrix} \quad ; \quad \mathbf{n_m}(\eta) = \begin{bmatrix} \cos^{2-e_1}(\eta) \\ \sin^{2-e_1}(\eta) \end{bmatrix} . \tag{5}$$

Again using a spherical product: $\mathbf{N} = \mathbf{n_m} \otimes \mathbf{n_h}$ we get the surface normals of the superquadric functions:

$$\mathbf{n}(\eta, \omega) = \begin{bmatrix} \cos^{2-e1}(\eta)\cos^{2-e2}(\omega) \\ \cos^{2-e1}(\eta)\sin^{2-e2}(\omega) \\ \sin^{2-e1}(\eta) \end{bmatrix} = \begin{bmatrix} \frac{1}{r}\cos^2(\eta)\cos^2((\omega) \\ \frac{1}{s}\cos^2(\eta)\sin^2((\omega) \\ \frac{1}{t}\sin^2(\eta) \end{bmatrix} . \tag{6}$$

3.2 Implicit Function Representation

Eliminating the parameters ω and η from Equation 2 using the trigonometric identity $\sin^2 r + \cos^2 r = 1$, we get the superquadric surface's implicit equation: [2].

$$\left[r^{\frac{2}{e_2}} + s^{\frac{2}{e_2}} \right]^{\frac{e_2}{e_1}} + t^{\frac{2}{e_1}} = 1.0. \tag{7}$$

This equation can then be written as an *inside-outside* function

$$f(\mathbf{r}) = f(r,s,t) = \left[r^{\frac{2}{e_2}} + s^{\frac{2}{e_2}} \right]^{\frac{e_2}{e_1}} + t^{\frac{2}{e_1}}. \tag{8}$$

This inside-outside function acts as a distance metric that is similar to the standard $\mathbf{L}_2$ distance metric [22, 32].

A three-dimensional solid defined in this way can be easily positioned and oriented with respect to the global coordinates by the following transformation,

$$\mathbf{X} = \mathcal{R}\mathbf{r} + \mathbf{b}, \tag{9}$$

where $\mathcal{R}$ is a rotation matrix, and $\mathbf{b}$ is a translation vector.

3.3 Deformations

As in Barr [7, 6], this basic set of functions can be generalized further by defining an appropriate set of global deformations $\mathcal{D}$ with parameters $\mathbf{m}$. For particular values of $\mathbf{m}$ the new deformed surface is defined using a deformation matrix $\mathcal{D}_{\mathbf{m}}$:

$$\mathbf{X} = \mathcal{R}\mathcal{D}_{\mathbf{m}}\mathbf{r} + \mathbf{b} \tag{10}$$

In the ThingWorld modeling system [27] deformations are described by a 3 x 3 deformation matrix $\mathcal{D}_{\mathbf{m}}$, referred to as the *modal deformation matrix*, whose entries are polynomials which mimic the free vibration modes of the object [27, 31]:

$$\mathcal{D}_{\mathbf{m}} = \begin{bmatrix} d_{00} & d_{01} & d_{02} \\ d_{10} & d_{11} & d_{12} \\ d_{20} & d_{21} & d_{22} \end{bmatrix}. \tag{11}$$

Because low-order modes change slowly as a function of object shape, the matrix can be used for a very wide range of shapes, and thus may be precomputed.

The computation of these polynomial deformations will be described in detail in the following sections. However for applications that do not require accurate simulation, such as user interfaces and animation for entertainment, it is reasonable to use a single set of particularly simple deformations derived using idealized elasticity properties. By using these idealized deformations to animate compact solid objects, visually pleasing physical animations can be obtained. This set of idealized deformations is described in detail in Appendix A.

3.4 Displacement Maps

The class of implicit functions can be generalized further by defining the surface as the set of points for which $f(\mathbf{r}) = d$ for some displacement function $d(\eta, \omega)$, where η, ω are the point's coordinates in the

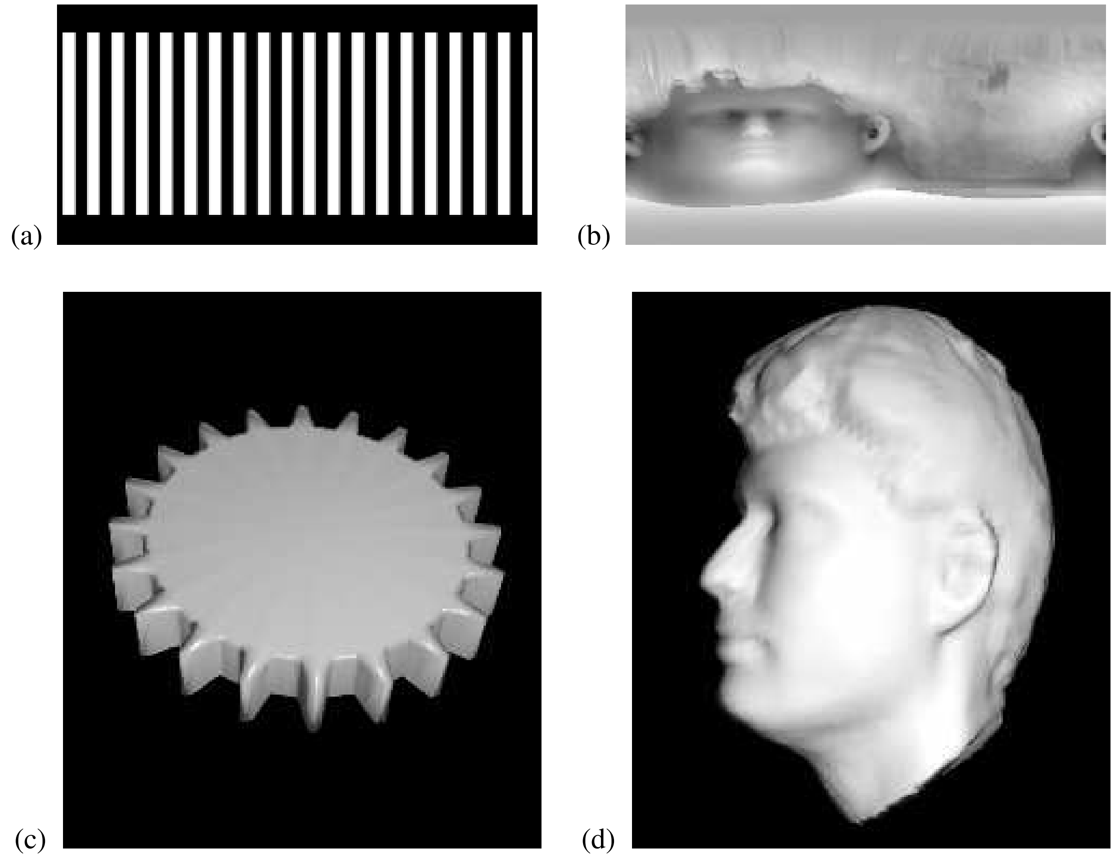


Figure 3: *Superquadrics with Displacement Maps (a) displacement map for a gear (b) displacement map for a head (c) gear and (d) head*

surface's two-dimensional parametric space. We define $\tilde{\mathbf{r}}(\eta, \omega)$ to be the displaced surface point before rotation, deformation or translation:

$$\tilde{\mathbf{r}} = \mathbf{r} + d\mathbf{n}. \tag{12}$$

This will have the effect of displacing the surface along the surface normal in the original undeformed space before applying the deformation [32].

The displacement map function $d(\eta, \omega)$ is defined by bilinear interpolation between point samples. If a smoother interpolation is desired, standard spline or pyramid-based multiresolution methods [11] can be employed.

Given a point $\mathbf{r}(\eta, \omega)$ and normal $\mathbf{n}(\eta, \omega)$ on the undeformed implicit surface, deformation transform $\mathcal{D}_{\mathbf{m}}$, and scalar displacement map function $d(\eta, \omega)$, we define $\bar{\mathbf{X}}$ to be the position vector including deformations and displacement map:

$$\bar{\mathbf{X}} = \mathcal{R}\mathcal{D}_{\mathbf{m}}\tilde{\mathbf{r}} + \mathbf{b}. \tag{13}$$

We also need to define $\bar{\mathbf{n}}$ to be the normal vector including deformations and displacement. To find $\bar{\mathbf{n}}$, it is first necessary to find $\tilde{\mathbf{n}}$, the surface normal for the displaced surface before deformation, and then apply deformations to get $\bar{\mathbf{n}}$. The normal $\tilde{\mathbf{n}}(\eta, \omega)$ is derived by taking the cross product of the partial derivatives for the undeformed displaced surface function, $\tilde{\mathbf{r}}(\eta, \omega)$:

$$\frac{\partial \tilde{\mathbf{r}}}{\partial \eta} = \frac{\partial \mathbf{r}}{\partial \eta} + d\frac{\partial \mathbf{n}}{\partial \eta} + \mathbf{n}\frac{\partial d}{\partial \eta} \quad \text{and} \quad \frac{\partial \tilde{\mathbf{r}}}{\partial \omega} = \frac{\partial \mathbf{r}}{\partial \omega} + d\frac{\partial \mathbf{n}}{\partial \omega} + \mathbf{n}\frac{\partial d}{\partial \omega}. \tag{14}$$

In our current implementation, the partials are calculated by finite differences. Figure 3 shows some examples of superquadrics with displacement maps.

4 Modeling Physics

In the previous section we developed a generalized implicit function representation. This representation allows us to describe a wide range of geometries, while at the same time retaining the advantage of having extremely efficient surface-to-point and surface-to-surface distance calculation. However, a good representation for physically-based modeling must be efficient for physical simulation as well as for geometric calculations. We can accomplish this by incorporating physical material properties, described by a set of differential equations, in our implicit function representation.

Several methods are available in engineering analysis for formulating these differential equations; however, the most common numerical approach is the *finite element method*. The major advantage of the finite element method is that it uses the Galerkin method of surface approximation. This provides us with an analytic characterization of shape and material properties over the whole surface, rather than just at the sampling points (nodes) as in the finite difference method [13, 8]. Mass, stiffness and damping matrices are then calculated by integrating local material properties over the volume by using the Galerkin shape functions to describe the object's geometry. The following section will describe this approach in some detail.

4.1 The Finite Element Method

Consider a finite element model, with global displacements in $\mathbf{X} = [X, Y, Z]^T$ denoted by $\mathbf{U} = [U, V, W]^T$ and $\mathbf{u} = [u, v, w]^T$ for displacements in object centered local coordinates $\mathbf{x}$. For a more detailed analysis on the importance of these axes for nonrigid modeling see [22].

Most of the formulation for a single finite element will be carried out in object's coordinate system; however, the general finite element equations will be formulated in the global coordinate system, so as to maintain the generality of the equations, because the assemblage of complex shapes leading to a combination of individual objects is required. We also need to introduce the forces applied on these objects. Consider the following notation:

$$\mathbf{f}^B = \begin{bmatrix} f_x^B & f_y^B & f_z^B \end{bmatrix}^T, \quad \mathbf{f}^S = \begin{bmatrix} f_x^S & f_y^S & f_z^S \end{bmatrix}^T, \quad \mathbf{F}^i = \begin{bmatrix} F_x^i & F_y^i & F_z^i \end{bmatrix}^T$$

where, $\mathbf{f}^B$ are the body forces, like mass, $\mathbf{f}^S$ are the surface forces, like buoyancy forces, and $\mathbf{F}^i$ are the concentrated forces, due to attachments. The strains ($\boldsymbol{\epsilon}$) due to $\mathbf{u}$ and the corresponding stresses (τ) are:

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} & \epsilon_{yy} & \epsilon_{zz} & \gamma_{xy} & \gamma_{xz} & \gamma_{yz} \end{bmatrix}^T$$

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{yy} & \tau_{zz} & \tau_{xy} & \tau_{xz} & \tau_{yz} \end{bmatrix}^T$$

In this framework, the equations of motion can be derived by using a weighted residual method, Lagrange's principle, Hamilton's principle or the principle of virtual work [22]. We will use the principle of virtual work as described below:

4.1.1 Principle of Virtual Work

The equilibrium of a body requires that for any compatible, small virtual displacements, which satisfy the essential boundary conditions, imposed on the body, the total virtual work is equal to the total external virtual work [8].

$$\int_V \bar{\boldsymbol{\epsilon}}\boldsymbol{\tau} dV = \int_V \bar{\mathbf{U}}^{BT}\mathbf{f}^B dV + \int_V \bar{\mathbf{U}}^{ST}\mathbf{f}^S dS + \sum_i \bar{\mathbf{U}}^{iT}\mathbf{F}^i \tag{15}$$

This simply means that provided the boundary conditions are satisfied, the work done to change the external energy (right hand side) is always equal to the work done to change the internal energy (left hand side). Note that the bar on the vector signifies "virtual"

For the above to be true the following conditions as described by Bathe [8] have to be satisfied:

- displacements should be compatible and continuous.
- must satisfy the displacement boundary conditions.
- must satisfy constitutive relationships (i.e., stresses can be evaluated from strains).

4.2 Interpolation Functions

Using the Galerkin Method for finite element discretization [34], we can set up a system of polynomial shape functions that relate the displacement of a single point to the relative displacements of all the other nodes of an object. This set of polynomial shape functions describes an *isoparametric finite element*. Using these polynomials we can then calculate the deformations which spread uniformly over the body as a function of its constitutive parameters [13, 8, 34, 1]. The polynomial shape function for each element is written in vector form as:

$$\mathbf{x}(x, y, z) = \mathbf{H}(x, y, z)\hat{\mathbf{x}}, \tag{16}$$

where, $\mathbf{x} = [x, y, z]^T$ is the shape polynomial, $\mathbf{H}(x, y, z)$ is a $(3 \times 3n)$ polynomial interpolation matrix and $\hat{\mathbf{x}}$ is a $(3n \times 1)$ vector of local coordinates of the element's nodes. Writing the above equation in matrix form for an n node element, we have:

$$\mathbf{x}(x, y, z) = \begin{bmatrix} h_0 & 0 & 0 & \ldots & h_{n-1} & 0 & 0 \\ 0 & h_0 & 0 & \ldots & 0 & h_{n-1} & 0 \\ 0 & 0 & h_0 & \ldots & 0 & 0 & h_{n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ \vdots \\ x_{n-1} \\ y_{n-1} \\ z_{n-1} \end{bmatrix}. \tag{17}$$

The following section details the method used to formulate these polynomial interpolation functions.

4.2.1 Determination of Polynomial Shape Functions

The fundamental properties of these polynomial shape functions h_i are:

- their value in the natural coordinate system is unity at node i and zero at all other nodes
- the sum of all interpolation functions $\sum_{i=0}^{n-1} h_i = 1.0$ at any point on the object

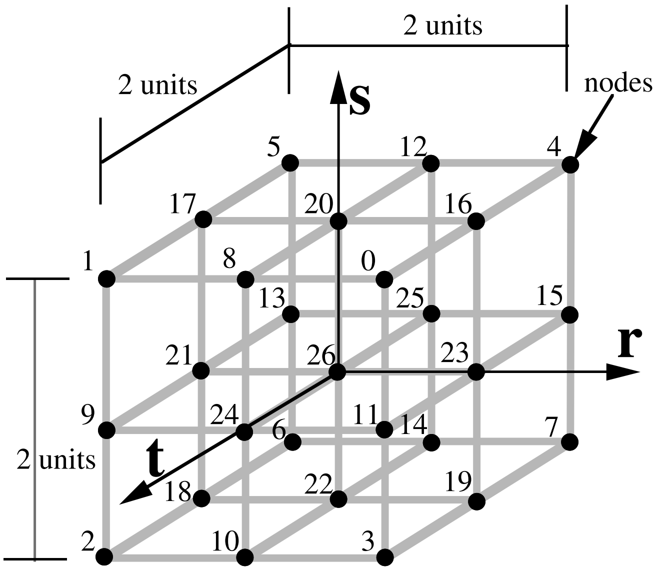


Figure 4: *An Isoparametric FEM model, in r, s, and t coordinates and in x, y, and z coordinates.*

Using these properties, the polynomial shape functions for any shape can be determined in a systematic manner; however, for simpler shapes it is much easier to construct these functions by inspection. For clarity of exposition, we will generate these functions for simple, undeformed unit shapes and then map them onto more complex shapes using global deformations defined in terms of the same polynomials. We will use a unit superquadric as described in Section 3.2, as our undeformed model. These undeformed coordinates in local space are r, s, and t. Hence, the above Equation 16 and a similar equation for displacements can be rewritten as:

$$\mathbf{x}(x, y, z) = \mathbf{H}(r, s, t)\hat{\mathbf{x}} \quad ; \quad \mathbf{u}(x, y, z) = \mathbf{H}(r, s, t)\hat{\mathbf{u}}, \tag{18}$$

where $\mathbf{u} = [u, v, w]^T$ is the displacement polynomial and $\hat{\mathbf{u}} = [u_0, v_0, w_0, \ldots, u_{n-1}, v_{n-1}, w_{n-1}]^T$ the nodal displacement vector similar to $\hat{\mathbf{x}}$. The process of shape description using Galerkin interpolant functions is illustrated in Figure 5.

Table 1 shows the polynomial shape functions for a three dimensional cube for up to 27 nodes. This table can also be used to generate functions for one and two dimensional isoparametric elements.

4.2.2 Strains due to Polynomial Deformations

For physical simulation it is necessary to calculate the *strain* due to deformation. Strain (ϵ) is defined as the ratio of displacement to the actual length, or simply the ratio of the change in length. For a three dimensional finite element model $\boldsymbol{\epsilon} = [\epsilon_{xx}, \epsilon_{yy}, \epsilon_{zz}, \gamma_{xy}, \gamma_{xz}, \gamma_{yz}]^T$, where:

$$\epsilon_{xx} = \frac{\partial u}{\partial x}; \quad \epsilon_{yy} = \frac{\partial v}{\partial y}; \quad \epsilon_{zz} = \frac{\partial w}{\partial z}; \quad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}; \quad \gamma_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}; \quad \gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}$$

Values of the Interpolation Coefficients	
$h_0 = g_0 - \frac{1}{2}(g_8 + g_{11} + g_{16}) - \frac{1}{4}(g_{20} + g_{23} + g_{24}) - \frac{1}{8}g_{26}$	$h_{14} = g_{14} - \frac{1}{2}(g_{22} + g_{25}) - \frac{1}{4}g_{26}$
$h_1 = g_1 - \frac{1}{2}(g_8 + g_9 + g_{17}) - \frac{1}{4}(g_{20} + g_{21} + g_{24}) - \frac{1}{8}g_{26}$	$h_{15} = g_{15} - \frac{1}{2}(g_{23} + g_{25}) - \frac{1}{4}g_{26}$
$h_2 = g_2 - \frac{1}{2}(g_9 + g_{10} + g_{18}) - \frac{1}{4}(g_{21} + g_{22} + g_{24}) - \frac{1}{8}g_{26}$	$h_{16} = g_{16} - \frac{1}{2}(g_{20} + g_{23}) - \frac{1}{4}g_{26}$
$h_3 = g_3 - \frac{1}{2}(g_{10} + g_{11} + g_{19}) - \frac{1}{4}(g_{22} + g_{23} + g_{24}) - \frac{1}{8}g_{26}$	$h_{17} = g_{17} - \frac{1}{2}(g_{20} + g_{21}) - \frac{1}{4}g_{26}$
$h_4 = g_4 - \frac{1}{2}(g_{12} + g_{15} + g_{16}) - \frac{1}{4}(g_{20} + g_{23} + g_{25}) - \frac{1}{8}g_{26}$	$h_{18} = g_{18} - \frac{1}{2}(g_{21} + g_{22}) - \frac{1}{4}g_{26}$
$h_5 = g_5 - \frac{1}{2}(g_{12} + g_{13} + g_{17}) - \frac{1}{4}(g_{20} + g_{21} + g_{25}) - \frac{1}{8}g_{26}$	$h_{19} = g_{19} - \frac{1}{2}(g_{22} + g_{23}) - \frac{1}{4}g_{26}$
$h_6 = g_6 - \frac{1}{2}(g_{13} + g_{14} + g_{18}) - \frac{1}{4}(g_{21} + g_{22} + g_{25}) - \frac{1}{8}g_{26}$	$h_{20} = g_{20} - \frac{1}{2}g_{26}$
$h_7 = g_6 - \frac{1}{2}(g_{14} + g_{15} + g_{19}) - \frac{1}{4}(g_{22} + g_{23} + g_{25}) - \frac{1}{8}g_{26}$	$h_{21} = g_{21} - \frac{1}{2}g_{26}$
$h_8 = g_8 - \frac{1}{2}(g_{20} + g_{24}) - \frac{1}{4}g_{26}$	$h_{22} = g_{22} - \frac{1}{2}g_{26}$
$h_9 = g_9 - \frac{1}{2}(g_{21} + g_{24}) - \frac{1}{4}g_{26}$	$h_{23} = g_{23} - \frac{1}{2}g_{26}$
$h_{10} = g_{10} - \frac{1}{2}(g_{22} + g_{23}) - \frac{1}{4}g_{26}$	$h_{24} = g_{24} - \frac{1}{2}g_{26}$
$h_{11} = g_{11} - \frac{1}{2}(g_{23} + g_{24}) - \frac{1}{4}g_{26}$	$h_{25} = g_{25} - \frac{1}{2}g_{26}$
$h_{12} = g_{12} - \frac{1}{2}(g_{20} + g_{25}) - \frac{1}{4}g_{26}$	$h_{26} = g_{26}$
$h_{13} = g_{13} - \frac{1}{2}(g_{21} + g_{25}) - \frac{1}{4}g_{26}$	

Where:

$$g_i = \begin{cases} 0 & \text{if node } i \text{ is not included} \\ G(r, r_i)G(s, s_i)G(t, t_i) & \text{otherwise} \end{cases}$$

$$G(\beta, \beta_i) = \begin{cases} \frac{1}{2}(1 + \beta\beta_i) & \text{for } \beta_i = \pm 1 \\ (1 - \beta^2) & \text{for } \beta_i = 0 \end{cases} \qquad \beta = r, s, t$$

Table 1: *The interpolation functions of eight to twenty variable number nodes for three dimensional elements.*

The polynomial shape functions could be used to calculate the strains (ϵ) over the body provided the displacements at the node points are known. Using this fact we obtain the strain displacement matrix:

$$\boldsymbol{\epsilon}(x, y, z) = \mathbf{B}(x, y, z)\hat{\mathbf{u}} \tag{19}$$

where $\mathbf{B}$ is a $(6 \times 3n)$ matrix written as follows:

$$\mathbf{B} = \begin{bmatrix} b_0^x & 0 & 0 & \dots & b_{n-1}^x & 0 & 0 \\ 0 & b_0^y & 0 & \dots & 0 & b_{n-1}^y & 0 \\ 0 & 0 & b_0^z & \dots & 0 & 0 & b_n^z \\ b_0^y & b_0^x & 0 & \dots & b_{n-1}^y & b_{n-1}^x & 0 \\ 0 & b_0^z & b_0^y & \dots & 0 & b_{n-1}^z & b_{n-1}^y \\ b_0^z & 0 & b_0^x & \dots & b_{n-1}^z & 0 & b_{n-1}^x \end{bmatrix} \tag{20}$$

where:

$$b_i^x = \frac{\partial}{\partial x}h_i(r, s, t), \quad b_i^y = \frac{\partial}{\partial y}h_i(r, s, t), \quad b_i^z = \frac{\partial}{\partial z}h_i(r, s, t)$$

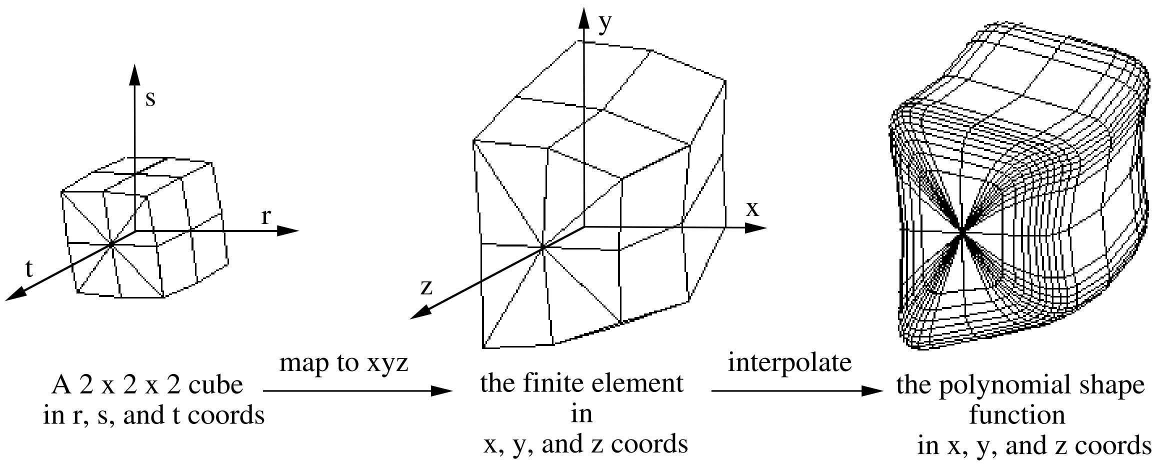


Figure 5: *Shape as defined by the polynomial.*

4.2.3 Jacobians for Superquadrics

Note that $\mathbf{H}$ in the previous equation, is a function of r, s, and t, not, x, y and z. Hence we need to use the chain rule as follows to determine $\mathbf{B}$:

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial}{\partial s}\frac{\partial s}{\partial x} + \frac{\partial}{\partial t}\frac{\partial t}{\partial x}, \tag{21}$$

similarly relationships for $\frac{\partial}{\partial y}$ and $\frac{\partial}{\partial z}$ are also possible. All of these three can be combined to form a *Jacobian Matrix* for the shape interpolation functions.

$$\begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial s} \\ \frac{\partial}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} & \frac{\partial z}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix}, \tag{22}$$

In matrix form:

$$\frac{\partial}{\partial \mathbf{r}} = \mathbf{J}\frac{\partial}{\partial \mathbf{x}}, \tag{23}$$

Since we require $\frac{\partial}{\partial \mathbf{x}}$:

$$\frac{\partial}{\partial \mathbf{x}} = \mathbf{J}^{-1}\frac{\partial}{\partial \mathbf{r}}, \tag{24}$$

so from above we can determine the partials as follows:

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial s} \\ \frac{\partial}{\partial t} \end{bmatrix}. \tag{25}$$

This Jacobian will be especially useful in inverse mapping for geometric and physical modeling; its usage is described in Section 6.1.

4.3 Stiffness and Load Matrices

By using Equations 16 and 19, m finite elements and their stiffness matrices can be assembled into one global stiffness matrix. This is known as the *direct stiffness method* [8]. The relationship between strains and initial stresses for the m-th finite element is:

$$\boldsymbol{\tau}^{(m)} = \mathbf{E}^{(m)}\boldsymbol{\epsilon}^{(m)} + \boldsymbol{\tau}^{I(m)}. \tag{26}$$

where m is the number of the finite element, $\mathbf{E}$ is the elasticity matrix and $\boldsymbol{\tau}^I$ is the initial stress. The elasticity matrix for a three dimensional isoparametric finite element model is

$$\mathbf{E} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix}. \tag{27}$$

where E is the Young's modulus and ν is the Poisson's ratio. Writing the Principle of Virtual work into element formulation for m elements:

$$\begin{aligned} \sum_m \int_{V^{(m)}} \bar{\boldsymbol{\epsilon}}^{(m)}\boldsymbol{\tau}^{(m)} dV^{(m)} &= \sum_m \int_{V^{(m)}} \bar{\mathbf{U}}^{(m)BT}\mathbf{f}^{(m)B} dV^{(m)} \\ &+ \sum_m \int_{V^{(m)}} \bar{\mathbf{U}}^{(m)ST}\mathbf{f}^{(m)S} dS^{(m)} \\ &+ \sum_m \sum_i \bar{\mathbf{U}}^{iT}\mathbf{F}^i, \end{aligned} \tag{28}$$

where the left hand side is the internal strain energy due to a virtual strain ($\bar{\epsilon}$) integrated over the volume and the right hand side is the sum of energies due to body, surface, and concentrated forces integrated over the model. Now substituting Equations 16 and 19 into Equation 28 and comparing with the following equilibrium equation [8]:

$$\mathbf{KU} = \mathbf{R}. \tag{29}$$

We get:

$$\mathbf{K} = \sum_m \int_{V^{(m)}} \mathbf{B}^{(m)T}\mathbf{E}^{(m)}\mathbf{B}^{(m)} dV^{(m)}, \tag{30}$$

also,

$$\mathbf{R} = \mathbf{R}^B + \mathbf{R}^S - \mathbf{R}^I + \mathbf{R}^C, \tag{31}$$

$$\mathbf{R}^B = \sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}\mathbf{f}^{(m)B} dV^{(m)}, \tag{32}$$

$$\mathbf{R}^S = \sum_m \int_{S^{(m)}} \mathbf{H}^{(m)T}\mathbf{f}^{(m)S} dS^{(m)}, \tag{33}$$

$$\mathbf{R}^I = \sum_m \int_{V^{(m)}} \mathbf{B}^{(m)T}\boldsymbol{\tau}^{(m)I} dV^{(m)}, \tag{34}$$

$$\mathbf{R}^C = \mathbf{F}. \tag{35}$$

4.3.1 Inertia

In the above equations of motion, the inertia effects were neglected. If the loads are to be applied rapidly, then the inertia forces have to be considered using d'Alambert's principle [8]. We can simply include the element inertia forces as a part of the body forces. Assuming that the element accelerations are in the same directions as the element displacements, the contribution of the inertia force to the load vector, with $\boldsymbol{\rho}$ as mass density is:

$$\mathbf{R}^B = \sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}[\mathbf{f}^{(m)B} - \boldsymbol{\rho}^{(m)}\mathbf{H}^{(m)}\ddot{\mathbf{U}}] dV^{(m)}. \tag{36}$$

Comparing with

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{R}, \tag{37}$$

we get,

$$\mathbf{M} = \sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}\rho^{(m)}\mathbf{H}^{(m)} dV^{(m)}. \tag{38}$$

4.3.2 Damping

Subsequently, in measuring time dependent response, it is observed that the energy is dissipated which in time dependent response is taken into account by velocity dependent damping forces. Based on this, with κ as the velocity damping coefficient:

$$\mathbf{R}^B = \sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}[\mathbf{f}^{(m)B} - \rho^{(m)}\mathbf{H}^{(m)}\ddot{\mathbf{U}} - \kappa^{(m)}\mathbf{H}^{(m)}\dot{\mathbf{U}}] dV^{(m)}. \tag{39}$$

Comparing:

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{C}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{R}, \tag{40}$$

and hence:

$$\mathbf{C} = \sum_m \int_{V^{(m)}} \mathbf{H}^{(m)T}\kappa^{(m)}\mathbf{H}^{(m)} dV^{(m)}. \tag{41}$$

5 Dynamics and Simulations

In the previous section we developed the equations of motion for elastic bodies. In this section we will describe how these equations can be integrated over time to obtain a simulation of the object's dynamic behavior.

The linear dynamic response of a system of finite elements is given by Equation 40, obtained by considering the equilibrium relationships at some time t. Rewriting this equation:

$$\mathbf{F}_I(t) + \mathbf{F}_D(t) + \mathbf{F}_E(t) = \mathbf{R}(t), \tag{42}$$

where $\mathbf{F}_I(t)$ are the inertia forces, $\mathbf{F}_D(t)$ are the damping forces and $\mathbf{F}_E(t)$ are elastic forces at time t.

Mathematically, Equation 40 represents a system of linear differential equations of second order. The solution could be obtained by standard methods of solving differential equations with constant coefficients [8, 35]. However, these methods are computationally very expensive; hence, more efficient techniques are required. Two methods of approaching this problem are discussed below.

5.1 Direct Integration Methods

In direct integration, Equation 40 is integrated using a numerical iterative procedure. By direct integration, it is meant that prior to numerical integration, no transformation of equation into a different form is carried out. The direct integration techniques are aimed at satisfying the system Equation 40 only at discrete time intervals Δt apart, rather then at any time t and that the variation of displacement, velocities and accelerations within a time step is assumed [8]. It is this assumed variation of displacements, velocities, and accelerations that determines the accuracy, stability, and cost efficiency of the system. Finite difference methods are used for these integrations. See Central Difference Method and Newmark Method in Bathe [8] for complete details on these methods.

5.2 Mode Superposition Method

In the mode superposition method, a change of basis to *generalized displacements* is undertaken so as to decouple the system of equations at the initial stage. Because the system of equations is decoupled, it can be solved much more efficiently than is possible by use of standard direct integration. The change of basis from the finite element nodal point displacements $\mathbf{U}$ to *modal displacements* $\mathbf{q}$ is accomplished by a transformation matrix $\mathbf{P}$:

$$\mathbf{U}(t) = \mathbf{P}\mathbf{q}(t), \tag{43}$$

where $\mathbf{q}(t)$ is a time-dependent vector of generalized displacements. Equation 43 can also be written as:

$$\mathbf{U}(t) = \sum_{i=0}^{p-1} \mathbf{P}_i q_i(t), \tag{44}$$

where $\mathbf{P}_i$ is the ith vector of $\mathbf{P}$. Substituting Equation 43 into Equation 40 and premultiplying by $\mathbf{P}^T$, we get:

$$\tilde{\mathbf{M}}\ddot{\mathbf{q}} + \tilde{\mathbf{C}}\mathbf{q} + \tilde{\mathbf{K}}\mathbf{q} = \tilde{\mathbf{R}}, \tag{45}$$

where:

$$\tilde{\mathbf{M}} = \mathbf{P}^T\mathbf{M}\mathbf{P}; \quad \tilde{\mathbf{C}} = \mathbf{P}^T\mathbf{C}\mathbf{P}; \quad \tilde{\mathbf{K}} = \mathbf{P}^T\mathbf{K}\mathbf{P}; \quad \tilde{\mathbf{R}} = \mathbf{P}^T\mathbf{R} \ . \tag{46}$$

Now we can express the elemental displacements in terms of generalized displacements for any finite element, using Equation 18, as:

$$\mathbf{u}(x, y, z, t) = \mathbf{HPq}(t). \tag{47}$$

With this transformation of basis, a new system of stiffness, mass and damping matrices is obtained which has a smaller bandwidth than the original system. Usually, the transformation matrix is established using the displacement solutions of free vibration equilibrium equations with damping neglected [8]. Hence, assuming no damping, the system of equations is:

$$\mathbf{M\ddot{U}} + \mathbf{KU} = \mathbf{R}. \tag{48}$$

The solution to this equation is of the form:

$$\mathbf{U} = \boldsymbol{\phi} \sin\left[\omega(t - t_0)\right], \tag{49}$$

where $\boldsymbol{\phi}$ is a $(3n \times 1)$ vector, t is the time variable, t_0 a time constant and ω the frequency of vibration of vector $\boldsymbol{\phi}$. From this a generalized eigenvalue problem could be set up as follows:

$$\mathbf{K}\boldsymbol{\phi} = \omega^2 \mathbf{M}\boldsymbol{\phi}, \tag{50}$$

from which $\boldsymbol{\phi}$ and ω can be determined.

The generalized eigenvalue problem in Equation 50 yields the p eigensolutions ($p \leq 3n$):

$$(\omega_0^2, \boldsymbol{\phi}_0), (\omega_1^2, \boldsymbol{\phi}_1), \ldots, (\omega_{p-1}^2, \boldsymbol{\phi}_{p-1})$$

where all the eigenvectors are $\mathbf{M}$-orthonormalized [8, 35, 36]. Hence we have:

$$\boldsymbol{\phi}_i^T \mathbf{M} \boldsymbol{\phi}_j \begin{cases} = & 1; \quad i = j \\ = & 0; \quad i \neq j \end{cases}, \tag{51}$$

and

$$0 \leq \omega_0^2 \leq \omega_1^2 \leq \omega_2^2 \leq \ldots \leq \omega_{p-1}^2. \tag{52}$$

The vector $\boldsymbol{\phi}_i$ is the ith-mode shape vector and ω_i is the corresponding frequency of vibration. Now defining a $(3n \times p)$ matrix $\boldsymbol{\Phi}$, which has for its columns the eigenvectors $\boldsymbol{\phi}_i$, and a diagonal $(p \times p)$ matrix $\boldsymbol{\Omega}^2$, with the eigenvalues ω_i^2 on its diagonal:

$$\boldsymbol{\Phi} = \left[\boldsymbol{\phi}_0,\ \boldsymbol{\phi}_1,\ \boldsymbol{\phi}_2,\ \ldots,\ \boldsymbol{\phi}_{p-1}\right]; \quad \boldsymbol{\Omega}^2 = \begin{bmatrix} \omega_0^2 & & & & & \\ & \omega_1^2 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & \omega_{p-1}^2 \end{bmatrix}. \tag{53}$$

Using Equation 53 we can write Equation 50 as:

$$\mathbf{K}\boldsymbol{\Phi} = \mathbf{M}\boldsymbol{\Phi}\boldsymbol{\Omega}^2 \tag{54}$$

and since the eigenvectors are **M**-orthonormal:

$$\mathbf{\Phi}^T \mathbf{K} \mathbf{\Phi} = \mathbf{\Omega}^2 \qquad \mathbf{\Phi}^T \mathbf{M} \mathbf{\Phi} = \mathbf{I}. \tag{55}$$

From the above formulations it becomes apparent that matrix $\mathbf{\Phi}$ is an appropriate transformation matrix $\mathbf{P}$; therefore, replacing in Equation 43 and Equation 45 we get:

$$\ddot{\mathbf{q}} + \mathbf{\Phi}^T \mathbf{C} \mathbf{\Phi} \mathbf{q} + \mathbf{\Omega}^2 \mathbf{q} = \mathbf{\Phi}^T \mathbf{R}(t). \tag{56}$$

Equation 56 shows that if the damping matrix is not included in the analysis, then the finite element equations are decoupled on using a transformation matrix $\mathbf{\Phi}$ with the free vibration mode shapes of FEM system to be its columns [31, 8]. Figure 6 illustrates some of the low-frequency mode shapes of a cube. Now we will briefly touch on the concepts of systems with or without damping.

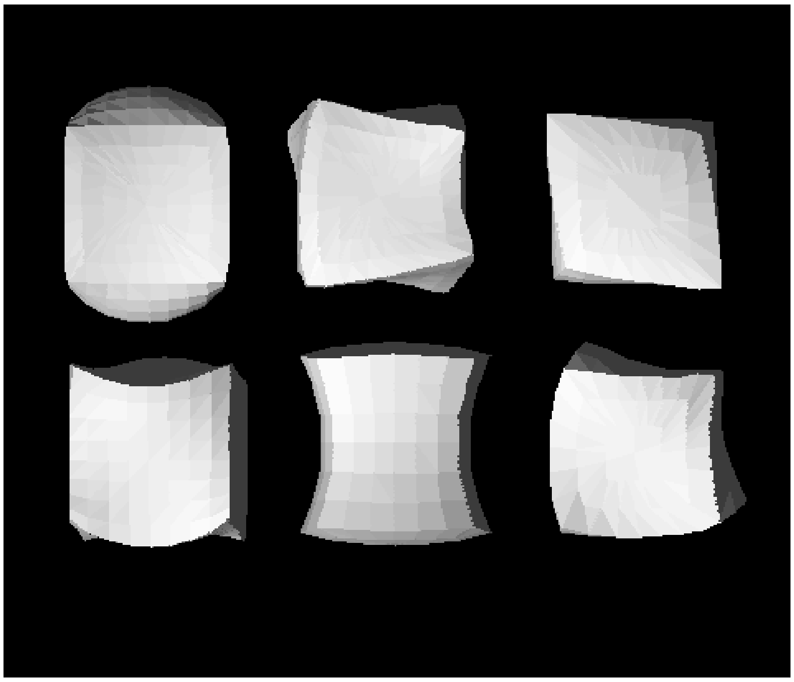


Figure 6: *Few of the vibrations mode shapes of a 27 node isoparametric element.*

Analysis without Damping: For systems in which damping effects are negligible and could be neglected the equations of equilibrium for analysis are reduced to:

$$\ddot{\mathbf{q}} + \mathbf{\Omega}^2 \mathbf{q} = \mathbf{\Phi}^T \mathbf{R}(t), \tag{57}$$

or to p independent and individual equations of the form:

$$\ddot{q}_i(t) + \omega_i^2 q_i(t) = r_i(t), \tag{58}$$

where $r_i(t) = \boldsymbol{\phi}_i^T \mathbf{R}(t)$ for $i = 0, \ldots, p-1$. Note here that in the above Equation 58, each equation is an equilibrium equation of single degree of freedom with a unit mass and stiffness of ω_i^2. The solution to each equation in Equation 58 can be obtained by any numerical integration method or by using the following *Duhamel integral*.

$$q_i(t) = \frac{1}{\omega_i}\int_0^t r_i(\tau)\sin\omega_i(t-\tau)d\tau + \alpha_i \sin\omega_i t + \beta_i \cos\omega_i t, \tag{59}$$

where α_i and β_i are constants depending on the initial conditions. For complete response all the p-equations above are solved and finite element nodal point displacements are obtained by superposition of response in each mode [8], using

$$\mathbf{U}(t) = \sum_{i=0}^{p-1} \boldsymbol{\phi}_i q_i(t) \tag{60}$$

Analysis with Damping: In the analysis of systems in which the effects of damping cannot be neglected, it would be preferable, if at all possible, to deal with the same functionality of decoupled equilibrium equations as in the undamped case. The damping matrix estimates the overall energy dissipation during the system response, and hence, cannot be generally obtained from element assemblage, like the mass and stiffness matrices. If it is assumed that damping is proportional to response then

$$\boldsymbol{\phi}_i^T \mathbf{C}\boldsymbol{\phi} = 2\omega_i^2 \xi_i \delta_{ij}, \tag{61}$$

where ξ_i is a modal damping parameter and δ_{ij} is the *Kronecker delta* ($\delta_{ij} = 1$ for $i = j$, $\delta_{ij} = 0$ for $i \neq j$). Now using Equation 61 and assuming that the eigenvectors $\boldsymbol{\phi}_i$ are $\mathbf{C}$-orthogonal, Equation 56 becomes:

$$\ddot{q}_i(t) + 2\omega_i\xi_i \dot{q}_i(t) + \omega_i^2 q_i(t) = r_i(t), \tag{62}$$

where $r_i(t) = \boldsymbol{\phi}_i^T \mathbf{R}(t)$ for $i = 0, \ldots, p-1$. Compare this equation with Equation 58 and note that an additional term is introduced. This is the damping term, with a damping ratio, ξ_i, of the single degree of freedom system in the modal coordinates. The *Duhamel integral* now becomes.

$$q_i(t) = \frac{1}{\bar{\omega}_i}\int_0^t r_i(\tau)e^{-\xi_i\omega_i(t-\tau)}\sin\bar{\omega}_i(t-\tau)d\tau + e^{-\xi_i\omega_i(t)}\left\{\alpha_i \sin\bar{\omega}_i t + \beta_i\cos\bar{\omega}_i t\right\}, \tag{63}$$

where $\bar{\omega}_i = \omega_i\sqrt{1-\xi_i^2}$. Note that if $\xi_i = 0$ Equation 58 becomes (62).

For a system with damping it is important to consider the following two points.

1. The total damping of the structure is equal to the sum of individual damping of each mode.

   AND

2. In modal analysis for numerical solution of the system, using the decoupled equations, only stiffness $\mathbf{K}$ and mass $\mathbf{M}$ matrices are calculated from the elemental assemblage. Damping $\mathbf{C}$ is computed as follows.

The damping matrix that satisfies the relation in Equation 61 is obtained by using the *Caughey series* [8],

$$\mathbf{C} = \mathbf{M}\sum_{k=0}^{c-1} a_k \left[\mathbf{M}^{-1}\mathbf{K}\right]^k, \tag{64}$$

where the coefficients a_k, $k = 0, \ldots, c-1$, are calculated from c simultaneous equations. At $c = 2$ Equation 64 reduces to *Rayleigh damping* ($\mathbf{C} = a_1\mathbf{M} + a_2\mathbf{K}$). If $c > 2$ then matrix $\mathbf{C}$ is, in general, a full matrix, which increases the cost of computation by big factor, therefore, *Rayleigh damping* is assumed. A limitation of this assumption is that higher modes are considered more damped than the lower modes. The magnitudes of the *Rayleigh damping* coefficients, a_1 and a_2, are to a large extent dependent on the energy dissipation characteristics of the structural materials.

5.3 Determination of Critical Time Step

In the preceding section the numerical solution of the dynamic equations of motion was discussed. An important observation is that the cost of this solution is directly proportional to the number of time steps. It follows from this that the selection of an appropriate time step in these integration techniques is extremely important. The time step should be small enough to obtain the desired accuracy and not smaller then necessary — too small a time step makes the solution more costly.

The aim of this section is to discuss how the technique of modal analysis can be used to dramatically reduce the computational cost of dynamic simulations. The analysis of stability and accuracy characteristics of the various integration methods will provide an essential guideline for selection of the appropriate modes and time step.

In any numerical integration scheme the time step Δt must be smaller than the critical value Δt_{cr}, which can be calculated from the mass and stiffness properties of the finite element assemblage. For a valid solution it is required that [8]:

$$\Delta t \leq \Delta t_{cr} = \frac{T_n}{\gamma}, \tag{65}$$

where $T_n = \frac{2\pi}{\omega_n}$ is the smallest period of the finite element assemblage with n degrees of freedom and γ is a constant depending on the integration method in use (*e.g.* $\gamma = \pi$ for Central Difference Method [8]).

Using either mode superposition or direct integration procedures, the solution is still obtained by numerical integration, over time, of Equations 40 or 56. In the mode superposition method, however, a change of basis from finite element coordinates to an eigenvector basis is performed prior to time integration (see Section 5.2). Equation 56 consists of p decoupled equations and the periods of vibration for the p degrees of freedom, $T_i, i = 0, 1, \ldots, p-1$ are known, where, $T_i = \frac{2\pi}{\omega_i}$. From these periods of vibration, we can choose an appropriate time step that assures a required level of accuracy for the numerical integration of Equation 56.

An important advantage of the mode superposition solution is that often only a small fraction of the total number of decoupled equations need to be considered to obtain a good approximate solution to the actual response. This is because modes associated with high resonance frequencies normally have little effect on object shape. For a fixed excitation energy, the displacement amplitude for each mode is inversely proportional to the *square* of the mode's resonance frequency. Thus relatively little error is incurred by discarding the small-amplitude, high-frequency modes, and using only the p largest-amplitude, low-frequency modes.

One effect of discarding small-amplitude modes is, of course, to reduce the number of equations that must be considered within each time step. However, because the maximum allowable time step is inversely proportional to the highest resonance frequency in the system of equations, a more important effect of discarding high-frequency modes is that we can use much larger time steps. In typical situations we have found that the savings from fewer equations and larger time steps can reduce computation time by up to two orders of magnitude, while at the same time producing a usefully accurate, realistic-looking simulation.

Further, unless very exacting monotonic convergence conditions are satisfied, finite element analysis only approximates the lowest exact frequencies, and little or no accuracy can be expected in approximating the higher frequencies and mode shapes. These exacting convergence conditions are rarely satisfied in computer graphics or vision applications, so that there is little justification in including the dynamic response of high frequencies in the analysis.

When only p modes are included in the analysis, the loss of accuracy due to truncation is referred to as *modal truncation*. Standard techniques are available for choosing the number of modes required to obtain a desired accuracy [17, 16]. Denoting by $\mathbf{U}_p$ the response predicted by mode superposition with p modes, an indication of the accuracy for the analysis at any time t is obtained by calculating the error measure e_p:

$$e_p(t) = \frac{\left|\mathbf{R}(t) - \left[\mathbf{M}\ddot{\mathbf{U}}_p(t) + \mathbf{K}\mathbf{U}_p(t)\right]\right|}{|\mathbf{R}(t)|}, \tag{66}$$

assuming that $|\mathbf{R}(t)| \neq 0$. If a good approximate solution for the equilibrium systems has been obtained, then $e_p(t)$ will be small at any time t. Note that $\mathbf{U}_p(t)$ must be obtained by accurate calculations at the response in each of the p modes considered, so that the error is only due to modal truncation. For a more detailed discussion on stability and accuracy of integration methods described here see Strang [35], Bathe [8], or Craig [18].

6 Combining Geometric and Physical Modeling

In complex, multi-body simulations the ability to efficiently detect and characterize collision response is extremely important. Unfortunately, the vertex or polygon representations standardly used in the finite element method are known to be very inefficient for collision detection [9], having a computational complexity is $\mathcal{O}(nm)$ operations, where n is the number of points and m the number of polygons.

However by using implicit function representations and their associated inside-outside function (see Section 3.2), one can perform collision detection and collision response characterization much more efficiently. In this section we explore methods that combine the geometric representations and physical models that were developed in the preceding sections, focusing on the problems of contact detection and response.

6.1 Contact Detection

In Section 3.2 we presented the inside-outside function of a superquad using its implicit function representation. The implicit function's positioned and oriented (rigid) inside-outside function becomes (using Equation 9):

$$f(\mathbf{r}) = f\left(\mathcal{R}^{-1}(\mathbf{X} - \mathbf{b})\right). \tag{67}$$

To detect a collision between a point $\mathbf{X} = [X_p, Y_p, Z_p]$ and the volume bounded by this surface, one simply substitutes the coordinates of $\mathbf{X}$ into the function f. If the result is negative, then the point is inside the surface and a collision has occurred. Generalizations of this basic operation may be used to find line-surface intersections or surface-surface intersections [42].

Similarly, for nonrigid deformations the inside-outside function could also be determined using Equation 10:

$$f(\mathbf{r}) = f\left(\mathcal{D}_{\mathbf{m}}^{-1}\mathcal{R}^{-1}(\mathbf{X} - \mathbf{b})\right). \tag{68}$$

In Section 4.2.1 we introduced a new set of polynomial deformations, described by Equation 18. Hence using these shape functions we get the following relationship for our deformation matrix.

$$\mathcal{D}_{\mathbf{m}} = \mathbf{H}(\mathbf{r})\hat{\mathbf{x}}. \tag{69}$$

The inside-outside function, with nonrigid deformations becomes (using Equations 68 and 69):

$$f(\mathbf{r}) = f\left(\mathbf{H}(\mathbf{r})^{-1}\hat{\mathbf{x}}\mathcal{R}^{-1}(\mathbf{X}-\mathbf{b})\right). \tag{70}$$

This inside-outside function is valid as long as the inverse polynomial mapping $\mathbf{H}(\mathbf{r})^{-1}$ exists. In cases where a set of deformations has no closed-form inverse mapping, Newton-Raphson and other numerical iterative techniques have to be used. This is the subject of discussion in Section 6.2

This method of defining geometry, therefore, provides an inherently more efficient mathematical formulation for contact detection than geometric representations such as polygons or splines, and a more exact formulation for physical simulation than was possible using the previously published modal techniques. See Moore [21] and Pentland [26] for comparisons and Pentland [26], Sclaroff [32] for a discussion of the computational complexity of contact detection algorithms.

For the case of an implicit function generalized by a displacement map, the associated inside-outside function is

$$f(\mathbf{r}) = f\left(\mathbf{H}(\mathbf{r})^{-1}\hat{\mathbf{x}}\mathcal{R}^{-1}(\bar{\mathbf{X}}-\mathbf{b}) - d\mathbf{n}\right). \tag{71}$$

It is difficult to evaluate this inside-outside function because we cannot know d or $\mathbf{n}$ in Equation 71 before we know $\mathbf{r}$, and *vice versa*. We therefore need a parametric projection function $P(\tilde{\mathbf{r}}) = (\eta, \omega)$, which can project an undeformed, displacement-mapped $\tilde{\mathbf{r}}$ point back onto the original, undisplaced implicit surface. These parameters can then be used to determine $\mathbf{n}$, $\mathbf{r}$, and d. The projection should be *normal projection* since the displacement map displaces surface points along the surface normal.

In the case of superquadric ellipsoids, for example, the projection function $P(\tilde{\mathbf{r}}) = (\eta, \omega)$ is computed as follows. We first find ω by observing:

$$\frac{\tilde{s}}{\tilde{r}} = \frac{\cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega}{\cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega} = \tan^{\epsilon_2}\omega \tag{72}$$

where $(\tilde{r}, \tilde{s}, \tilde{t})^T = \tilde{\mathbf{r}}$ is the undeformed, displaced surface point. From Equation 72, we see that $\omega = \arctan^{1/\epsilon_2}(\tilde{s}/\tilde{r})$. The remaining parameter, η, is determined by either $\eta = \arctan^{1/\epsilon_1}((\tilde{t}\cos^{\epsilon_2}\omega)/\tilde{r})$ or $\eta = \arctan^{1/\epsilon_1}((\tilde{t}\sin^{\epsilon_2}\omega)/\tilde{s})$ depending on whether $\tilde{r}$ or $\tilde{s}$ is larger.

6.2 From Deformed Space to Undeformed Space

The polynomial shape functions shown in Table 1 (see Section 4.2.1) are dependent on undeformed space $\mathbf{r}$, so that the inverse deformation problem requires inverting the (3×3) Jacobian of polynomials calculated in Section 4.2.3. This can be accomplished efficiently by use of either the Newton-Raphson or Secant Methods, as follows.

The goal is to calculate $\mathbf{r}$ from $\mathbf{X}$, using Equation 10 and 18. As $\mathcal{D}_{\mathbf{m}} = \mathbf{H}(\mathbf{r})\hat{\mathbf{x}}$ is itself a function of $\mathbf{r}$, simply inverting $\mathcal{D}_{\mathbf{m}}$ does not solve the problem. Essentially we have a system of three polynomials:

$$X = g_1(r,s,t)\,, \quad Y = g_2(r,s,t)\,, \quad Z = g_3(r,s,t)$$

where X, Y and Z are already known, as are the constants

$$G_1(r,s,t) = 0\,, \quad G_2(r,s,t) = 0\,, \quad G_3(r,s,t) = 0$$

A solution is then found using Newton's Method [35]

$$\mathbf{J}^k(\mathbf{X}^{k+1} - \mathbf{X}^k) + \mathbf{G}^k = \mathbf{0} \tag{73}$$

where k represents iteration, and $\mathbf{G} = [G_1, G_2, G_3]^T$.

6.3 Collision Response and Friction

After contact detection, the second part of collision characterization is collision response. The methods presented above are well suited to contact detection. In this section we will present methods to determine the physical response of a collision, with special emphasis on efficient computation.

Part of the problem of determining the collision response is to characterize the type of contact. The contact could be of the following the three types:

- Instantaneous contact, contact does not last for more then an instant.
- continuous contact, sliding, *etc.*

In the case of an instantaneous contact, collision response is a discontinuous impulse force that acts instantaneously on the involved bodies. In the case of continuous contact, it is a continuous response shared by the contacting bodies, formed the moment they come into contact. This response needs to be calculated analytically at the initial moment of collision to prevent bodies from interpenetrating. The case of continuous contact is particularly difficult, and typically requires use of a very small time step.

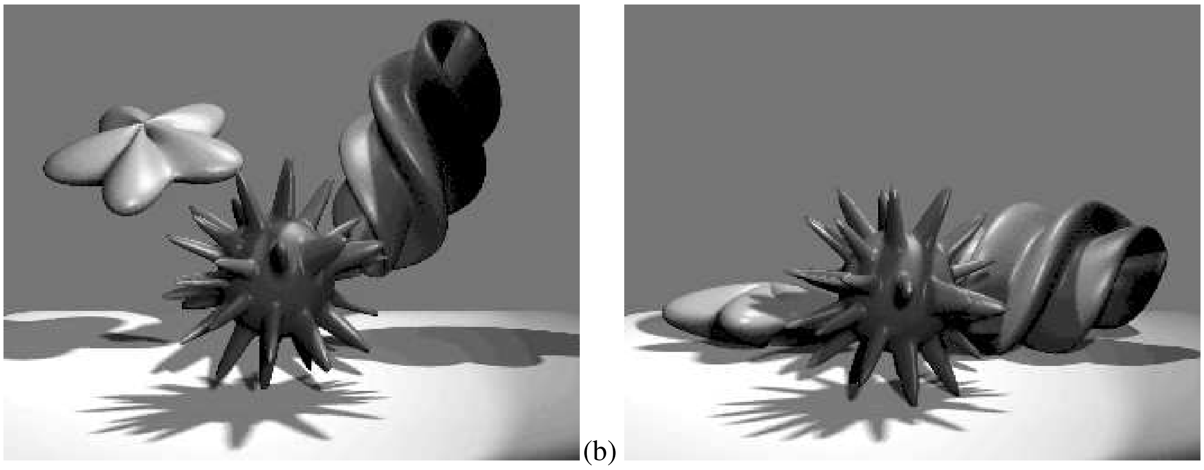
Moore and Wilhelms [21] used a method for rigid body collisions which is based on calculating the impulse for the colliding bodies at a single point. Laws of conservation of angular and linear momentum were used to determine the collision response. This force was then applied by introducing springs at the collision points. Hahn [15] presented similar methods for rigid-body contacts using non-holonomic and holonomic constraints [19, 35]. "Penalty" methods that introduce restraining forces against penetrations were presented by Terzopoulos [38] for rigid and non-rigid collisions. Continuous contact remains a particularly difficult problem for all of these approaches. Moore and Wilhelms [21] modeled simultaneous collisions as staggered series of single collisions, while Hahn [15] modeled them as a series of frequently occurring collisions.

In our approach collisions are characterized as impulse forces. The object's inside-outside function, however, provides an efficient analytical method for determining the penetration due to collision. Using Equation 19 with the inside-outside function of the undeformed superquad, and the stress-strain relationship of Equation 26, we can compute the stress at any point of an object due to penetration. This stress is then used to determine the restraining forces against penetration. Thus, by combining the ideas of geometric representation by implicit function with the isoparametric finite element model, an analytic characterization of the collision response is achieved.

Continuous contact is still a problem in this scenario, as it is modeled as simultaneous collisions requiring a very small time step. In our system this problem is lessened by adding a Kalman estimation process in the dynamics equations, as described by Essa and Pentland [14], to obtain efficient approximation to the dynamic simulation. Friction is handled by using the parametric representation of geometry to determine the tangent vectors at the point of collision; this together with Coulomb's Law of friction allows the frictional resistance to be computed. For more details on modeling of friction see Essa [13], Bathe *et. al.* [9] and Baraff [4, 3].

7 Applications

In the following section we present some applications of the methods described above. First we will present an example of a dynamic simulation of complex shapes. Next, we will describe some machine vision applications that allow automatic creation and animation of 3-D computer models starting from video



(a) (b)

Figure 7: *Complex shapes (sea shells) generated by applying deformations and displacement maps modeled under gravity force dropping through viscous medium, with contact detection and collision response from the ground.*

imagery. Finally, we will discuss an example in which the modal representation developed here is used for object recognition.

7.1 Modeling and Physics of Complex Shapes

Figure 7 shows two frames from a physically-based animation in which three seashell-like shapes drop through a viscous medium (*e.g.*, seawater), hit the sea bottom, bump into each other, and then come to rest. The simulation was conducted using the ThingWorld system [27].

The seashell shapes were modeled as superquadric ellipsoids with displacement maps. Each displacement map consisted of a 100 x 100 uniformly spaced grid generated by combinations of sines and cosines. The shells were polygonalized for display and simulation purposes — approximately 2300 polygons for the spike seashell, and 576 polygons for the other seashells. Bounding boxes were also computed for the objects. During the simulation, if these bounding boxes crossed, then polygon vertices were plugged into the offending objects' inside-outside functions to test for collisions.

Execution time for the three active objects was 0.05 seconds per time step during the initial frames of the animation (before any contact), and 0.081 seconds per time step during the final few frames, when the three seashells were colliding with the seabed (which is not planar), and the shells were colliding with each other. Subtracting the pre-contact time from the execution time during contact, we find it took approximately 0.031 seconds per time step for contact detection and calculation of the non-rigid dynamics. Contact detection, physical simulation, and geometric updates were computed on a Sun 4/330, with a TAAC board performing rendering.

7.2 Vision Applications: Automatic Creation and Animation

It is useful to be able to fit a generalized implicit function representation to three-dimensional point data. This allows objects in the real world to be measured at several points, and then models of them to be automatically created in the computer. To fit point data with a generalized implicit function requires determining both the deformation polynomials $\mathbf{x}$ (Equation 16) and a displacement map.

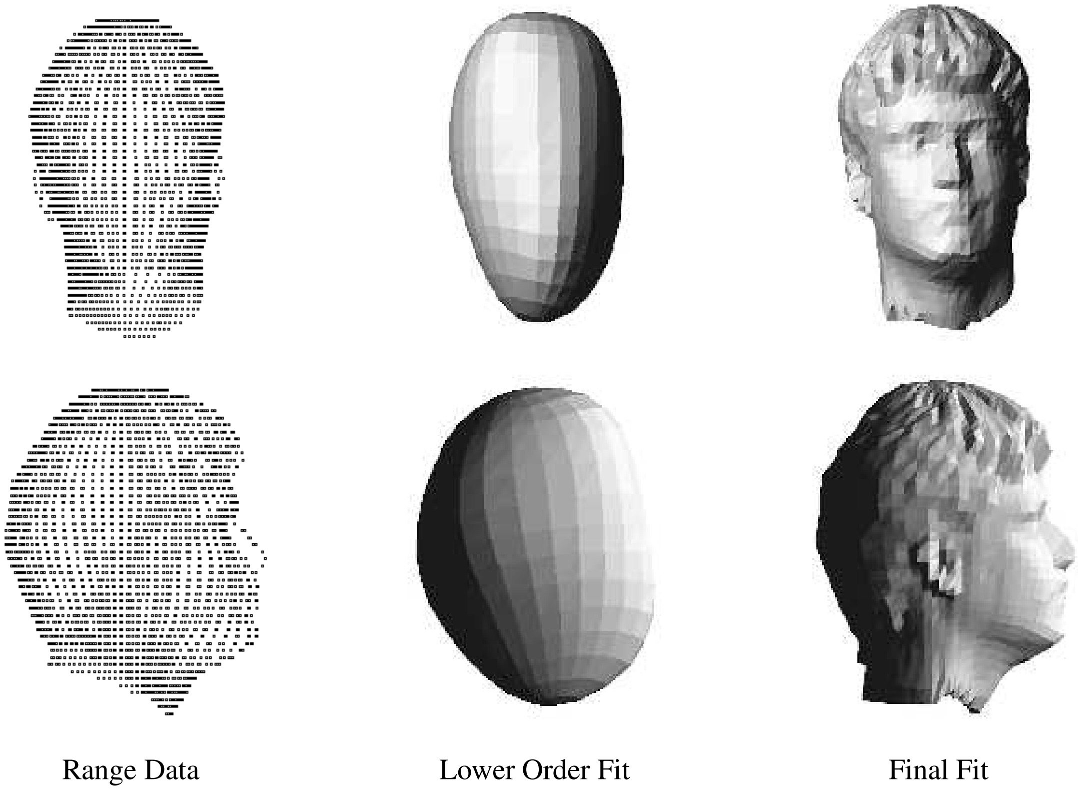


Figure 8: *Recovery of heads using range data.*

7.2.1 3-D Shape From 3-D Sensor Measurements

Let us first assume that we are given m three-dimensional sensor measurements $\mathbf{X}^w$ (in the global coordinate system) that originate from the surface of a single object

$$\mathbf{X}^w = [x_0^w, y_0^w, z_0^w, \cdots, x_{m-1}^w, y_{m-1}^w, z_{m-1}^w]^T.$$

Following Terzopoulos *et. al.* [40], we then attach virtual springs between these sensor measurement points and particular nodes on our deformable model. This defines an equilibrium equation whose solution $\mathbf{U}$ is the desired fit to the sensor data. Consequently, for n nodes with corresponding sensor measurements, we can calculate the virtual loads $\mathbf{R}$ exerted on the undeformed object while fitting it to the sensor measurements. For node k these loads are simply $[r_{3k}, r_{3k+1}, r_{3k+2}]^T = [x_k^w, y_k^w, z_k^w]^T - [x_k, y_k, z_k]^T$ where $x_0, y_0, z_0, \cdots, x_{n-1}, y_{n-1}, z_{n-1}$ are the nodal coordinates of the undeformed object in the object's coordinate frame.

To fit a deformable solid to the measured data, we solve the equilibrium governing equation $\mathbf{KU} = \mathbf{R}$. The difficulty in calculating this solution is the large dimensionality of $\mathbf{K}$, so that iterative solution techniques are normally employed. However, a closed-form solution is available simply by converting this equation to the modal coordinate system as in Equation 43. This is accomplished by substituting $\mathbf{U} = \mathbf{\Phi q}$ and premultiplying by $\mathbf{\Phi}^T$, so that the equilibrium equation becomes

$$\mathbf{\Phi}^T \mathbf{K} \mathbf{\Phi} \mathbf{q} = \tilde{\mathbf{K}} \mathbf{q} = \mathbf{\Phi}^T \mathbf{R} \tag{74}$$

where $\tilde{\mathbf{K}} = \mathbf{\Phi}^T \mathbf{K} \mathbf{\Phi}$ is a *diagonal* matrix as shown in Section 5.2. Again, note that the calculation of $\mathbf{\Phi}$ needs to be performed only once as a precomputation, and then stored for all future applications. Further, it is normally not desirable to use all of the eigenvectors (as explained previously), so that the $\mathbf{\Phi}$ matrix remains of manageable size even when using large numbers of nodes. In our implementation $\mathbf{\Phi}$ is normally a 81 x $3n$ matrix, where n is the number of nodes.

The solution to the fitting problem, therefore, is obtained by inverting the diagonal matrix $\tilde{\mathbf{K}}$.

$$\mathbf{q} = \tilde{\mathbf{K}}_6^{-1} \mathbf{\Phi}^T \mathbf{R} \tag{75}$$

where $\tilde{\mathbf{K}}_6 = \tilde{\mathbf{K}} + \mathbf{I}_6$ and $\mathbf{I}_6$ is a matrix whose first six diagonal elements are ones, and remaining elements are zero. Inclusion of the matrix $\mathbf{I}_6$ into Equation 75 may be interpreted as adding an external force that constrains the solution to have no residual translational or rotational stresses.

The results of this point fitting technique are illustrated in Figure 8. The input data was range data gathered from a 360° laser rangefinder built by Cyberware corporation. The 3-D model for the head was recovered with the method described above. It took about 5 seconds on a Sun 4/330 to recover the deformations and displacement map from 2500 data points, including graphic display. For additional details see Pentland and Sclaroff [30].

A color map was then computed at higher resolution, using about 10,000 data points. The color map was computed by projecting the color points onto the surface using Gaussian interpolation for smoothing. Figure 9 shows two frames (front and side view) from a physically-based animation in which the head model recovered in Figure 8 is struck with a wooden mallet.

7.2.2 3-D Shape from 2-D Contours

In the case where we are given only 2-D contour information we can still employ the same equations to estimate shape, however we must generalize Equation 75 to reflect the uncertainty we have about the z coordinate of each contour point. This can be accomplished by altering Equation 75 to reflect the fact that some sensor measurements are more certain than others.

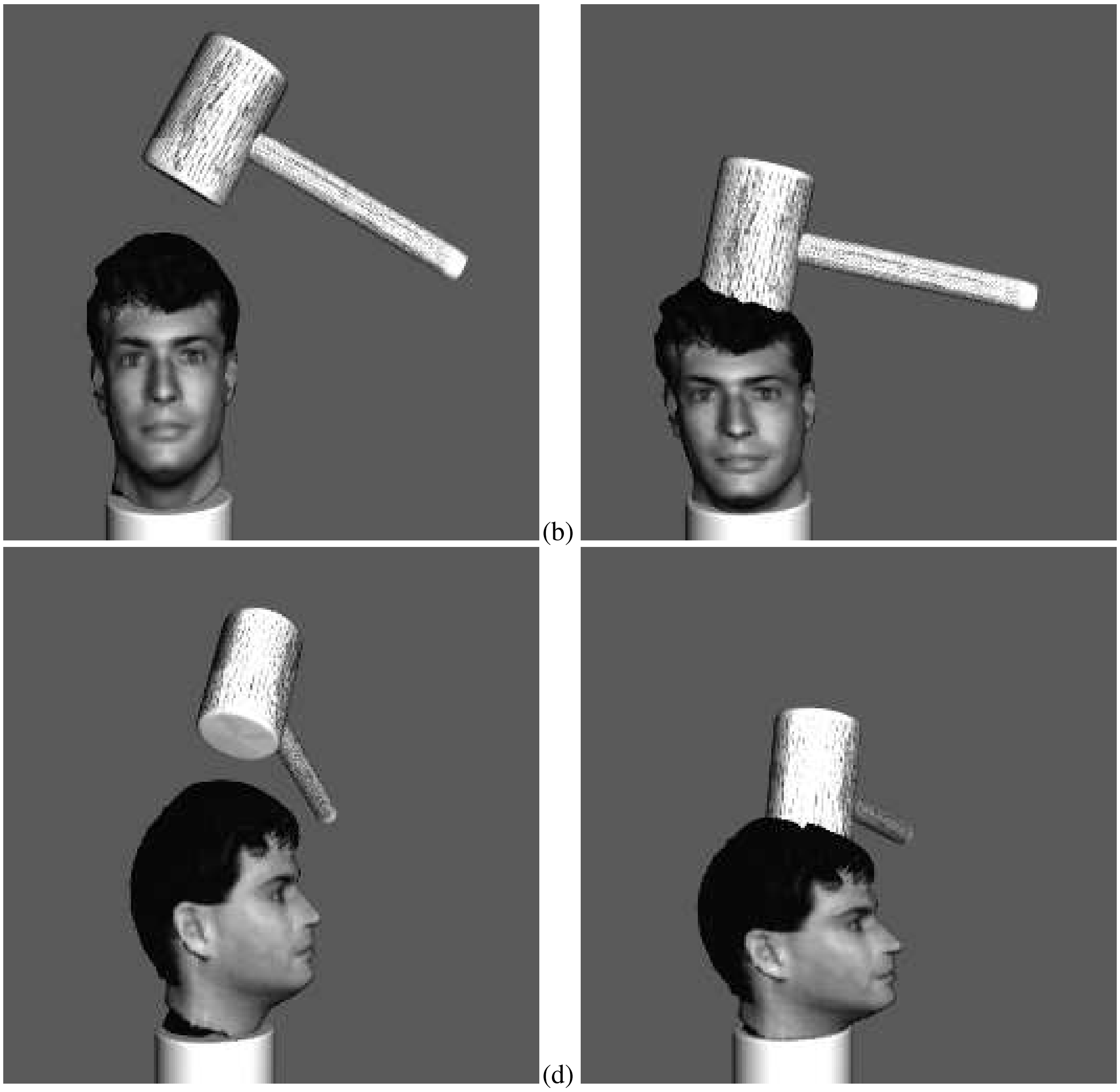
We accomplish this by introducing a $3n$ x $3n$ diagonal weighting matrix $\mathbf{W}$:

$$\mathbf{q} = \tilde{\mathbf{K}}_6^{-1} (\mathbf{W} \mathbf{\Phi})^T \mathbf{R} \tag{76}$$

The diagonal entries of $\mathbf{W}$ are inversely proportional to the uncertainty (variance) of the data associated with each of the nodal coordinates. The effect of $\mathbf{W}$ is to make the strength of the virtual springs associated with each data point reflect the uncertainty of the measurement.

7.2.3 Nonrigid Motion

Figure 10(a) shows one frame from a sequence of X-ray images, with the zero-crossing edge contours overlayed. From these contours the 3-D shape was estimated using Equation 76. The resulting shape is shown in Figure 10(b) as a 3-D wireframe overlayed on the original X-ray data. Figure 10(c) shows the recovered model from the side. Because only bounding contour information was available, the shape estimated along the z axis (shown in Figure 10(c)) is determined by finding the minimum stress state that



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

Figure 9: *A head generated using a displacement map, struck with a wooden mallet.*

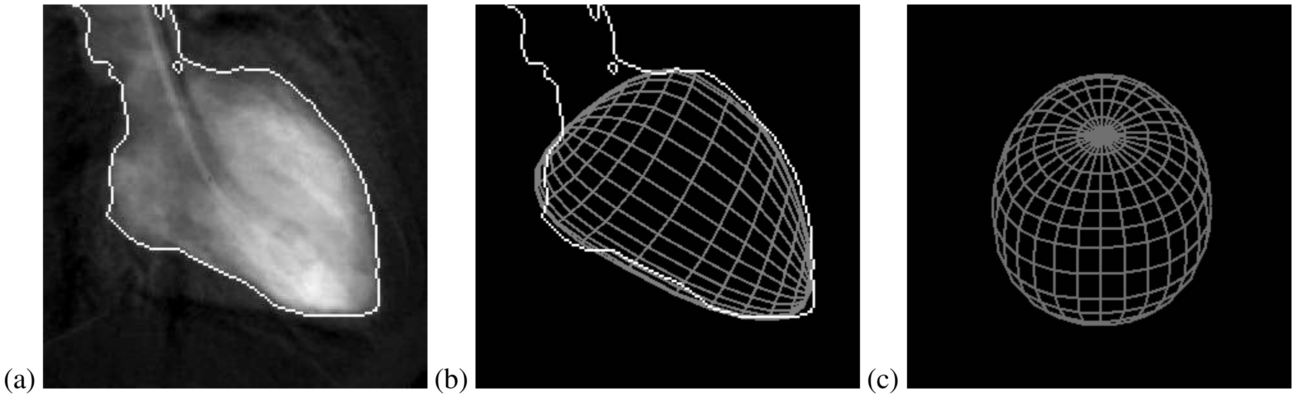


Figure 10: *An example showing the use of a 2-D image contour to recover a 3-D deformable solid model. The original image and contour are shown in (a). The model is recovered from the contour as shown in (b). An orthogonal side view is shown in (c).*

still fits the bounding contour. The z-axis shape cannot, therefore, be regarded as accurate but only as plausible and consistent. Note that use of a minimum stress criterion for solution means that symmetric and mirror symmetric shapes are preferred. Execution time was approximately one second on a standard Sun 4/330. For additional detail see Pentland and Sclaroff [30].

Figure 11 shows an example of recovering non-rigid motion from contour information. The 3-D shape and motion of the heart ventricle was tracked over time using the contour information shown at the top within each box of Figure 11. For each frame Equation 76 was used to obtain an estimate of 3-D shape from the contour information (see Figure 10). These shape estimates were then integrated in a Kalman filter formulation, as described in Pentland and Horowitz [28]. As in the single-image case, deformations along the z axis are determined by finding the minimum stress state that still fits the bounding contour. The z-axis deformations cannot, therefore, be regarded as accurate but only as plausible. Execution time was approximately one second per frame on a standard Sun 4/330.

7.2.4 Recognition

Perhaps the major drawback of physically-based models in machine vision applications is that that they have not been very useful for recognition, comparison or other database tasks. This is primarily because they normally have more degrees of freedom than there are sensor measurements, so the recovery process is underconstrained. Therefore, although heuristics such as smoothness and symmetry can be used to obtain a solution, they do not produce a stable unique solution [30]. The modal representation (Section 5.2), assuming all the modes are employed, decouples the degrees of freedom within the nonrigid dynamic system of Equation 40, but it does not by itself reduce the total degrees of freedom.

However by discarding enough of the high-frequency modes we can obtain an overconstrained estimate of the shape. Moreover, because the modes form an orthogonal basis set, there is only one way to represent an object that is in canonical position [30]. Consequently, to recognize an object we need only to compare the modal amplitudes of the recovered shape description to a database of known objects. This comparison can be accomplished by a simple normalized inner product, so that even very large databases can be searched quickly.

Figure 12 illustrates an object recognition experiment using this technique. In this experiment we

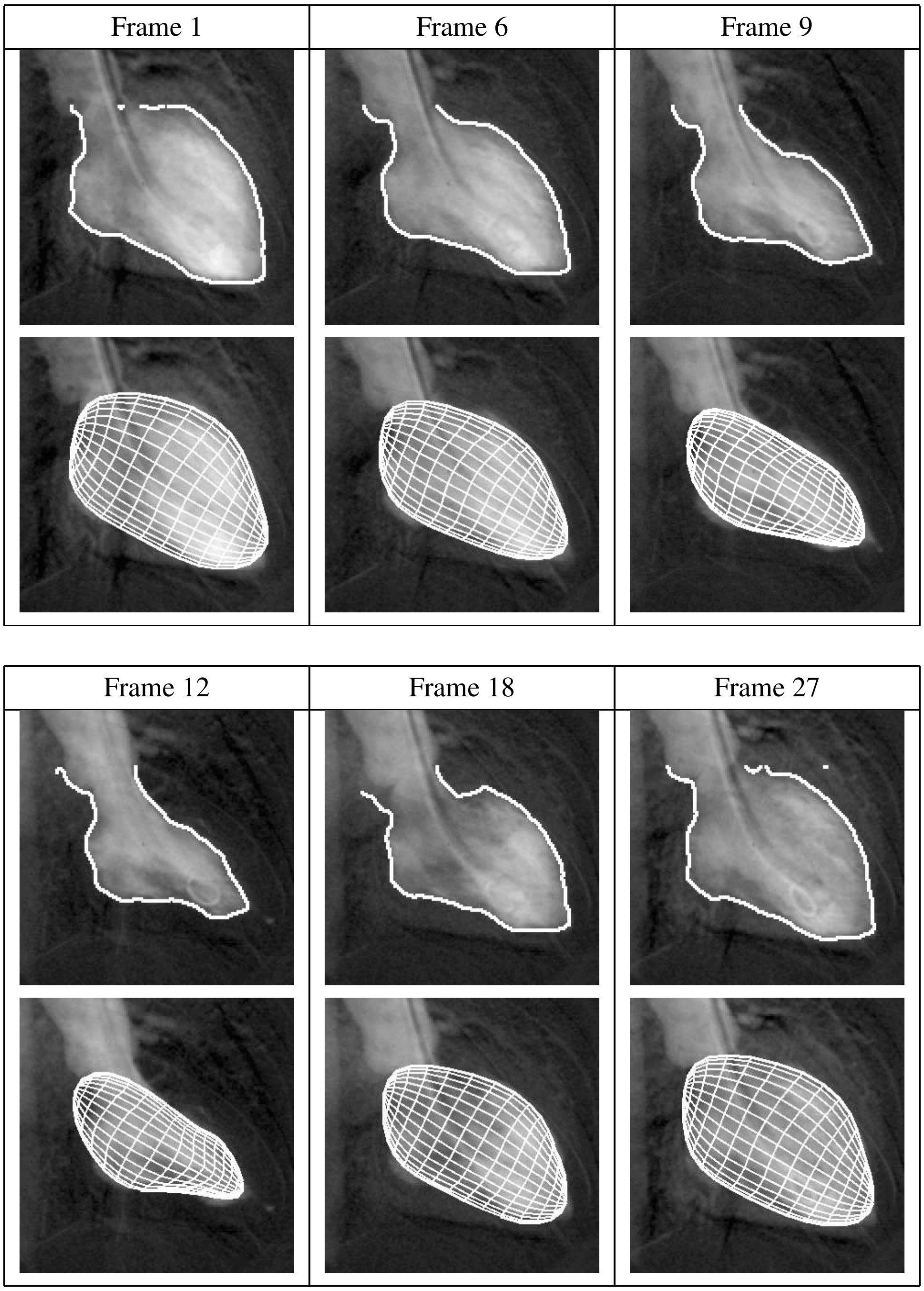


Figure 11: *A heart's nonrigid motion as it was recovered from contours taken from a motion sequence. The contours, extracted via a simple threshold and zero crossing scheme, are shown along the top. The deformable model recovered from these contours is shown in wireframe.*

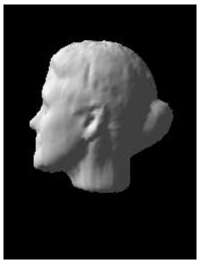
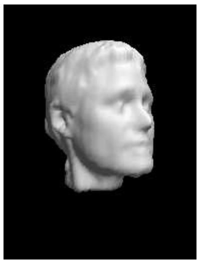
	-90^o	-45^o	0^o	45^o	90^o
					
a	0.16	-0.19	-0.24	-0.19	-0.12
b	0.26	0.28	0.35	0.37	0.30
c	**0.88**	**0.99**	**0.99**	**0.98**	**0.99**
d	0.03	0.15	0.21	0.21	0.11
e	0.06	-0.13	-0.10	-0.04	-0.06
f	0.58	0.44	0.46	0.46	0.50
g	0.53	0.53	0.52	0.50	0.58
h	0.42	0.47	0.53	0.49	0.50

Figure 12: *Recognition using 3-D Range Data.*

obtained laser rangefinder data of eight people's heads from a five different viewing directions: the right side (-90^o), halfway between right and front (-45^o), front (0^o), halfway between front and left (45^o), and the left side (90^o). In each case the range data was from the forward-facing, visible surface only. Recognition was accomplished by first recovering a 3-D model from the visible-surface range data, and then comparing the recovered mode values to the mode values stored for each of the known head models. The known model producing the largest normalized inner product was declared to be the recognized object. The rigid-body modes were not used, so that the recognition process was translation, rotation, and scale invariant.

In this experiment and overall recognition accuracy of 92.5% was obtained. Figure 12 illustrates typical results from this experiment. The top row of Figure 12 illustrates the five models recovered from range data from each of the five viewpoints. The recovered mode values were then compared to each of the stored head models; the inner products obtained are shown below each recovered head model. In this figure all of the input data were views of head model **C**. As can be seen, the inner products between recovered 3-D model and model **C** are much larger than for other modes, so that a correct recognition was obtained. For more detail see Pentland and Sclaroff [30].

8 Conclusion

We have presented a unified approach for geometric and physical modeling for use in computer graphics, machine vision, and animation. By combining polynomial shape functions with a parametric implicit function representation we obtain a geometric model that is simple, efficient, and quite general. Modal analysis methods were then employed to determine the physical vibration shapes and frequencies, in order to efficiently perform dynamic simulations.

This combination of physical and geometric models provides many benefits. The implicit function representation affords the ability to do fast inside-outside checks for collision. The modal shape polynomials provide the ability to accurately model physical deformations, and provide an easy method for determining physical strains and stresses.

Acknowledgments. We would like to thank the ThingWorld programming posse: Martin Friedmann, Brad Horowitz, and Thad Starner.

References

[1] Amir Wadi Al-Khafaji and John R. Tooley. *Numerical Methods in Engineering Practice*. Holt, Rinehart and Winston Inc., 1986.

[2] Ruzena Bajcsy and Franc Solina. Three dimensional object representation revisited. *Proceedings First International Conference on Computer Vision*, pages 231–240, 1987.

[3] David Baraff. Coping with friction for non-penetrating rigid body simulation. *ACM SIGGRAPH Conference Proceedings*, 25(4):31–40, 1991.

[4] David Baroff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *ACM SIGGRAPH Conference Proceedings*, 23(4):223–232, 1989.

[5] Alan H. Barr. Faster calculations of superquadric shapes. *IEEE Computer Graphics and Applications*, 1(1):101–122, 1981.

[6] Alan H. Barr. Superquadrics and angle-preserving transforms. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

[7] Alan H. Barr. Global and local deformations of solid primitives. *ACM SIGGRAPH Conference Proceedings*, 18(3):21–30, 1984.

[8] Klaus-Jürgen Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, 1982.

[9] Klaus-Jürgen Bathe and Anil Chaudhary. A Solution method for Axisymmetric Contact Problems. *International Journal for Numerical Methods in Engineering*, 21:65–88, 1985.

[10] A Borning. A constraint oriented simulation laboratory. Technical Report SSL-79-3, Xerox PARC, Palo Alto, CA, 1979.

[11] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.

[12] Laurent D. Cohen and Issac Cohen. A finite element method applied to new active contour models and 3d reconstruction from cross sections. In *Third International Conference on Computer Vision*, pages 587–591. IEEE Computer Society, 1990.

[13] Irfan A. Essa. Contact detection, collision forces and friction for physically based virtual world modeling. Master's thesis, Department of Civil Engineering, Massachusetts Institute Technology, June 1990.

[14] Irfan A. Essa and Alex Pentland. Estimated physics: Dynamic simulation in a resource limited environment. Technical Report 182, M.I.T. Media Laboratory, Vision and Modeling Group, 1991.

[15] James K. Hahn. Realistic animation of rigid bodies. *ACM SIGGRAPH Conference Proceedings*, 22(4):299–308, 1988.

[16] Gordon G. Hastings and Wayne J. Book. A linear dynamic model for flexible robotic manipulators. *IEEE Control Systems Magazine*, 7(1):61–64, 1987.

[17] P. C. Hughes. Modal identities for elastic bodies, with application to vehicle dynamics and control. *Journal of Applied Mechanics*, 47:177–184, March 1980.

[18] Roy R. Craig Jr. *Structural Dynamics; An Introduction to Computer Methods*. John Wiley and Sons, 1981.

[19] Donald E. Kirk. *Optimal Control Theory*. Prentice-Hall, 1970.

[20] Dimitri Metaxas and Demetri Terzopoulos. Recursive estimation of shape and nonrigid motion. In *IEEE Workshop on Visual Motion*, pages 306–311. IEEE Computer Society, 1991.

[21] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *ACM SIGGRAPH Conference Proceedings*, 22(4):289–298, 1988.

[22] G. N. Pande, G. Beer, and J. R. Williams. *Numerical Methods in Rock Mechanics*. John Wiley and Sons, 1990.

[23] Alex Pentland. Perceptual organization and representation of natural form. *Artificial Intelligence*, 28(3):293–331, 1986.

[24] Alex Pentland. Towards an ideal 3-D CAD system. *SPIE Conference on machine vision and man-machine interface*, 1987.

[25] Alex Pentland. Automatic extraction of part deformable models. *International Journal of Computer Vision*, 4:107–126, 1990.

[26] Alex Pentland. Computational complexity versus virtual worlds. *ACM SIGGRAPH Proceedings of Symposium on Interactive 3D Graphics*, 24(2):185–192, 1990.

[27] Alex Pentland, I. A. Essa, M. Friedmann, B. Horowitz, and S. E. Scalaroff. The thingworld modeling system: Virtual sculpting by modal forces. *ACM SIGGRAPH Proceedings of Symposium on Interactive 3D Graphics*, 24(2):143–144, 1990.

[28] Alex Pentland and Bradley Horowitz. Recovery of nonrigid motion and structure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):730–742, July 1r991.

[29] Alex Pentland, Bradley Horowitz, and Stan Sclaroff. Non-rigid motion and structure from contour. In *IEEE Workshop on Visual Motion*, pages 288–293. IEEE Computer Society, 1991.

[30] Alex Pentland and Stan Sclaroff. Closed form solutions for physically based shape modeling and recovery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):715–729, July 1991.

[31] Alex Pentland and J. R. Williams. Good vibrations: Modal dynamics for graphics and animation. *ACM SIGGRAPH Conference Proceedings*, 23(4):215–222, 1989.

[32] Stan Sclaroff and Alex Pentland. Generalized implicit functions for computer graphics. *ACM SIGGRAPH Conference Proceedings*, 25(4):247–250, 1991.

[33] Thomas Sederberg and Scott Parry. Free-form deformations of solid geometric models. *ACM SIGGRAPH Conference Proceedings*, 20(4):151–160, 1986.

[34] Larry J. Segerlind. *Applied Finite Element Analysis*. John Wiley and Sons, 1984.

[35] Gilbert Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge, 1986.

[36] Gilbert Strang. *Linear Algebra and its Applications*. Wellesley-Cambridge, Third edition, 1986.

[37] I. Sutherland. A man-machine graphical graphical communication system, in interactive computer graphics. In *1963 Spring Joint Computer Conference Proceedings*, 1963.

[38] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *ACM SIGGRAPH Conference Proceedings*, 21(4):205–214, 1987.

[39] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models for 3-d object reconstruction. *International Journal of Computer Vision*, 1(3):211–221, 1987.

[40] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[41] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[42] B. von Herzen, A. Barr, and H. Zatz. Geometric Collisions for Time-Dependent Parametric Surfaces. *ACM SIGGRAPH Conference Proceedings*, 24(4):39–48, 1990.

[43] J. R. Williams and Alex Pentland. Superquadrics and modal dynamics for discrete elements in concurrent design. *First U.S. Discrete Element Conference Proceedings*, 1989.

[44] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. *ACM SIGGRAPH Conference Proceedings*, 24(4):243–252, 1990.

Appendix A: Simple Idealized Deformation Modes

For applications that do not require accurate simulation, such as user interfaces and animations for entertainment, we have found that it is adequate to use a single set of particularly simple deformations derived using idealized elasticity properties. By using these idealized deformations to animate compact solid objects, visually pleasing physical animations can be obtained. The entries of the deformation matrix

Values of the Modal Deformation Matrix
$d_{00} = m_6 + sm_{12} + tm_{15} - (m_{13} + m_{16})sgn(r) - m_{14} - m_{17}$
$d_{01} = m_{11} + 2s(m_{13} + sgn(r)m_{14})$
$d_{02} = m_{10} + 2t(m_{16} + sgn(r)m_{17})$
$d_{10} = m_{11} + 2r(m_{19} + sgn(s)m_{20})$
$d_{11} = m_7 + rm_{18} + tm_{21} - (m_{19} + m_{22})sgn(s) - m_{20} - m_{23}$
$d_{12} = m_9 + 2t(m_{22} + sgn(s)m_{23})$
$d_{20} = m_{10} + 2r(m_{25} + sgn(t)m_{26})$
$d_{21} = m_9 + 2s(m_{28} + sgn(t)m_{29})$
$d_{22} = m_8 + rm_{24} + sm_{27} - (m_{25} + m_{28})sgn(t) - m_{26} - m_{29}$
where: $\mathbf{r} = [r,\ s,\ t]^T$ = Undeformed Point

Table 2: *Modal Deformation Matrix Values in terms of Modal Amplitudes*

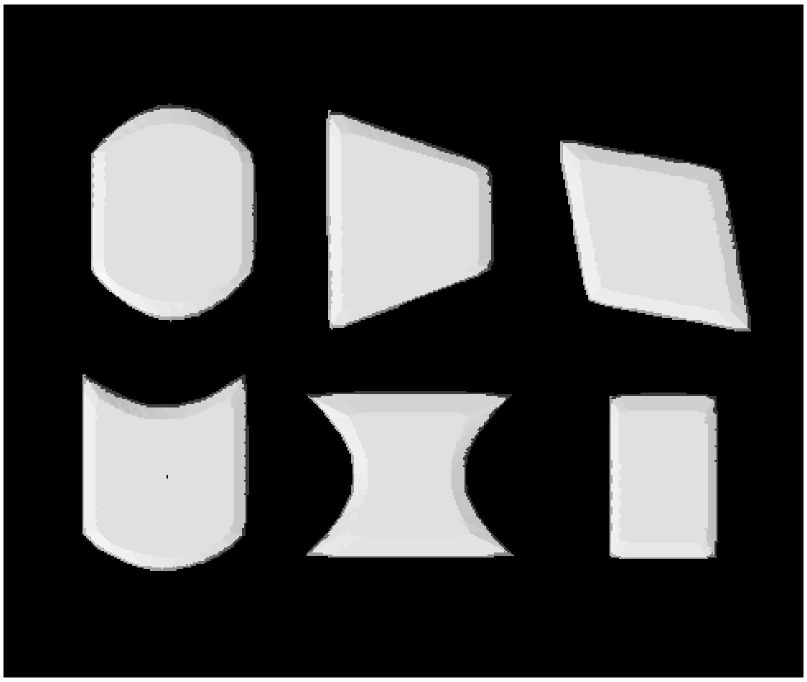


Figure 13: *Few of the vibrations mode shapes of a cube, using an idealized deformation matrix* $\mathcal{D}_{\mathbf{m}}$

$\mathcal{D}_{\mathbf{m}}$ for these idealized modes are shown in Table 2, where $\mathbf{m} = [m_0, m_1, \ldots, m_{p-1}]^T$ is a p x 1 vector of the *modal amplitudes*.

The modal amplitudes m_i formulated in this way have an intuitive meaning. Modal amplitudes m_0 - m_5 are the rigid body modes of translation and rotation, m_6 - m_8 are the x, y, and z sizes, m_9 - m_{11} are shears about the x, y, and z axes and the rest are bends, tapers and pinches in various axes. Figure 13 illustrates a few of these idealized deformation modes for a cube; the reader should compare this figure to Figure 6.