From the proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation

A Multiresolution Framework for Dynamic Deformations

Steve Capell Seth Green Brian Curless Tom Duchamp Zoran Popović

University of Washington

Abstract

We present a novel framework for the dynamic simulation of elastic deformable solids. Our approach combines classical finite element methodology with a multiresolution subdivision framework in order to produce fast, easy to use, and realistic animations. We represent deformations using a hierarchical basis constructed using volumetric subdivision. The subdivision framework provides topological flexibility and the hierarchical basis allows the simulation to add detail where it is needed. Since volumetric parameterization is difficult for complex models, we support the embedding of objects in domains that are easier to parameterize.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: animation, deformation, physically-based animation, physically-based modeling, multiresolution, hierarchical basis

1 Introduction

With advances in computer animation over the past few decades, people have come to expect a high degree of realism. A common way to achieve such realism is to painstakingly craft each keyframe, but this requires a tremendous amount of labor and skill. An alternative is to endow animated objects with physical properties and allow them to move automatically according to physical laws. This approach is especially appealing because it promises increased realism while simultaneously reducing the human workload. But such simulations are computationally expensive and difficult to set up.

In this paper we introduce a new framework for dynamic deformations that is fast and easy to use, and produces realistic results. Our method can be applied to complex objects, and does not depend on the representation of the object or its surface.

In our framework, the domain of deformation is described by volumetric subdivision. Applying the principles of subdivision, we construct a hierarchical basis to represent displacements. We then formulate the equations of motion for a dynamically deforming elastic solid in terms of the hierarchical basis. This formulation defines the temporal behavior of the basis coefficients in the presence of body forces and constraint forces. The mathematical framework and equations of motion are described in Section 3.

Within this framework, we construct a robust dynamic simulator that employs an implicit solver, permitting us to take large timesteps without succumbing to instabilities. Spatially varying material properties are accommodated, regardless of the resolution of the simulation. External constraints are handled using Lagrange multipliers. Using the hierarchical basis, our simulation can adapt in order to concentrate computational resources where they are needed. We introduce a new method of quasi-linearization of the equations of motion that is effective but simpler than the standard method. Our simulator is described in Section 4.

In Section 5 we describe some resulting interactive simulations built using our framework, and we conclude with a discussion in Section 6.

2 Related Work

Our work builds on previous work in the areas of static and dynamic deformations, hierarchical methods, and subdivision.

Early work on deformations focused on non-dynamic techniques. The introduction of free-form deformation (FFD) by Sederberg et al. [38] allowed objects to be deformed independent of their structure by embedding them in easily-parameterized domains. MacCracken and Joy developed three-dimensional lattice subdivision, an extension of Catmull-Clark subdivision surfaces, for the purpose of easing the topological restrictions of FFD. Another important extension to FFD was the introduction of dynamics by Faloutsos et al. [15]. Our framework builds on both of these extensions, using a flexible class of control lattices as in [24], and embedding objects in *dynamic* free-form lattices as in [15]. But unlike [15], where a diagonal stiffness matrix is employed, we use the finite element method to simulate the dynamics of the embedded object.

The use of physically-based deformable models in graphics was pioneered by Terzopoulos et al. [42]. The original work applied the Lagrangian equations of motion using a finite difference scheme to simulate elastic objects with regular parameterizations. Their framework was extended to include inelastic behaviors [41], and to handle stiff rotating bodies using linearized equations [44].

Physically-based deformations have since been extended in many ways. Platt and Barr [33] introduced better constraint handling via Lagrange multipliers. Pentland and Williams [31] obtained realtime simulations by using only a few vibration modes. Witkin and Welch [46] introduced the use of low-order polynomial deformations to achieve fast deformations, to which Baraff and Witkin [3] added non-penetration constraints. Metaxas and Terzopoulos [28] combined global deformations with local finite element deformations. O'Brien and Hodgins [29] computed realistic fractures of viscoelastic bodies using a finite element framework in which objects are approximated by a fine tessellation of tetrahedra. A similar framework was used by Picinbono et al. [32] to achieve interactive simulations of a virtual liver composed of about 2000 tetrahedra. But we are interested in interactively deforming objects that cannot be approximated well by so few tetrahedra.

Deformable surfaces have also been used for geometric modeling. Celniker and Gossard [9] applied the finite element method to minimize surface energy while meeting constraints. These ideas

were later extended to NURBS by Terzopoulos and Qin [43] and to Catmull-Clark subdivision surfaces by Qin et al. [36].

For some applications dynamic motion has not been deemed necessary so static and quasi-static methods have been employed [6, 18, 21, 22, 37]. Since our interest is in realistic motion, we build on dynamic methods.

Implicit solvers have been enjoying a renaissance in dynamic deformations for computer graphics in recent years. Terzopoulos et al. [41, 42] used semi-implicit solvers in their initial work. Baraff and Witkin [4] used an implicit scheme to greatly improve the speed and stability of cloth simulation. Desbrun et al. [14] used a semi-implicit scheme to stabilize stiff systems. Hauth and Etzmuss [20] recently showed that the implicit method BDF(2), which factors in one state of history in addition to the current and future states, produces fast and accurate results for cloth simulations.

As noted in the introduction, we simulate using a hierarchical basis. Such bases have been widely studied in the field of numerical analysis (see, e.g., [2]). Our work was thus inspired, but our focus is on building a useful framework for computer animation.

In the field of computer graphics, hierarchical methods have been used to solve many problems including rendering [17], geometric modeling [16], and deformable model simulations. Terzopoulos et al. [41] employed a multigrid solver on a rectangular domain. Debunne et al. [11] created interactive simulations using an octree representation, adaptive in both space and time. To animate a surface, the surface points are linked to the grid by a weighting scheme. Our framework is similar in that we embed objects in domains that are easier to parameterize than the object itself. But since we do not require that the domain be a parallelepiped, we can fit it more closely to the underlying object. Later, Debunne et al. [12] developed an adaptive framework for deformable models employing an unstructured hierarchy of tetrahedral meshes [12]. At each level of the hierarchy the object is approximated by a tetrahedral mesh. In our framework, coarse simulations do not require that the object be coarsely approximated. Our coarse simulations factor in the detailed shape and material properties of the entire object, as approximated at a fine level of detail during the preprocessing stage. Recently, Grinspun et al. [19] have developed a general method for organizing hierarchical simulations involving refinable function bases.

Subdivision schemes have also been used for simulation. Weimer and Warren [45] employed 3D subdivision to solve PDEs associated with fluid flow. Cirak et al. [10] employed subdivision surfaces to solve thin shell finite element problems, exploiting the smoothness of subdivision basis functions to satisfy the integrability requirements of thin shell elements. McDonnell et al. simulated volumetric subdivision objects using a mass-spring model [26] and then applied the finite-element methodology to the problem [27].

3 Formulation

In this section we describe our framework for computing the elastic dynamics of an elastic body. We model the dynamics of the deformable body as a system of second-order ordinary differential equations obtained by applying the finite element method (FEM) to the Lagrangian formulation of the equations of elasticity (see [39, 30, 34]).

To establish notation, we begin with a quick review of the method. Consider a body whose rest state is defined by a domain $\Omega \subset \mathbb{R}^3$. The trajectory of the body over time is represented by a function

$$\mathbf{p} : \Omega \times \mathbb{R} \to \mathbb{R}^3 \; : \; (\mathbf{x}, t) \mapsto \mathbf{p}(\mathbf{x}, t) \tag{1}$$

It is convenient to decompose $\mathbf{p}(\mathbf{x}, t)$ as the sum of the rest state and a displacement

$$\mathbf{p}(\mathbf{x}, t) = \mathbf{x} + \mathbf{d}(\mathbf{x}, t) \, . \tag{2}$$

The function $\mathbf{d}$ is the solution of a system of second-order partial differential equations, which we want to approximate by a (finite) system of second-order ordinary differential equations.

This is done via a *hierarchical basis* [2] of piecewise smooth functions on Ω. Roughly speaking, a collection $\mathcal{B} = \{\phi^a(\mathbf{x}) \; : a = 1, 2, \dots\}$ is called a hierarchical basis if the diameter of the support of ϕ^a decreases as a increases.[1] Expressing the state of the body in terms of $\mathcal{B}$ yields the expansion

$$\mathbf{p}(\mathbf{x}, t) = \sum_a \left(\mathbf{r}_a + \mathbf{q}_a(t)\right) \, \phi^a(\mathbf{x}) = \left(\mathbf{r}_a + \mathbf{q}_a(t)\right) \, \phi^a(\mathbf{x}) \tag{3}$$

where $\mathbf{r}_a$ and $\mathbf{q}_a(t)$ are elements of $\mathbb{R}^3$ and

$$\mathbf{x} = \mathbf{r}_a \phi^a(\mathbf{x}) \tag{4}$$

is the expansion of the identity map. We use the Einstein summation conventions throughout this paper: any term that contains the same index as both a subscript and a superscript implies a summation over that index.

We represent the state of the body at time t as a column vector of generalized coordinates $\mathbf{q} = \mathbf{q}(t)$ whose a-th component is $\mathbf{q}_a(t) \in \mathbb{R}^3$. Thus both the kinetic energy T and the potential energy V are functions of $\mathbf{q}$:

$$T = T(\dot{\mathbf{q}}) \text{ and } V = V(\mathbf{q}) \tag{5}$$

where $\dot{\mathbf{q}}$ denotes the time derivative of $\mathbf{q}$. The equations of motion are then the *Euler-Lagrange* equations

$$\frac{d}{dt}\left(\frac{\partial T(\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}}\right) + \frac{\partial V(\mathbf{q})}{\partial \mathbf{q}} + \mathbf{Q}^{ext} - \mu \dot{\mathbf{q}} = 0 \tag{6}$$

where $\partial T / \partial \dot{\mathbf{q}}$ and $\partial V / \partial \mathbf{q}$ denote gradients with respect to $\dot{\mathbf{q}}$ and $\mathbf{q}$, respectively. The term $\mathbf{Q}^{ext}$ is a *generalized force* corresponding to external body forces, such as gravity. The last term is a generalized dissipative force, added to simulate the effect of friction. A more realistic damping term could easily be added (see, e.g., [29]).

In the following subsections, we discuss in detail our construction of a hierarchical basis and the derivation of the terms of Equation (6).

3.1 The Hierarchical Basis

Our construction of a hierarchical basis of functions on Ω is a generalization to subdivision functions of the embedding methods commonly used in the finite element community under the headings *fictitious component* or *fictitious domain* methods [5, 25]. Rather than attempting to construct a basis of functions on the (perhaps irregular) region Ω, one instead views Ω as embedded in a regular region N (for instance, a cube in $\mathbb{R}^3$) on which a hierarchical basis of functions is known. Restriction to Ω then induces a collection of functions on the original domain. Color Plates 1(a) and 1(d) illustrate the fictitious domain approach applied to a dragon-shaped region.

More generally, consider a (piecewise smooth) homeomorphism

$$\mathbf{h} : K \to N \subset \mathbb{R}^3 \; : \; u \mapsto \mathbf{h}(u) \tag{7}$$

from a three-dimensional hexahedral complex K, which we call a *control lattice*, onto a superset $N \supset \Omega$. Denote the pre-image of Ω under $\mathbf{h}$ by $D = \mathbf{h}^{-1}(\Omega) \subseteq K$. Figure 1 illustrates the relationship between an object Ω and a hexahedral complex K. The homeomorphism $\mathbf{h}$ can then be used to transfer any hierarchical construction on K to one on N.

[1] It is also desirable that $\mathcal{B}$ be well-behaved with respect to the L^2-inner product on $L^2(\Omega)$.

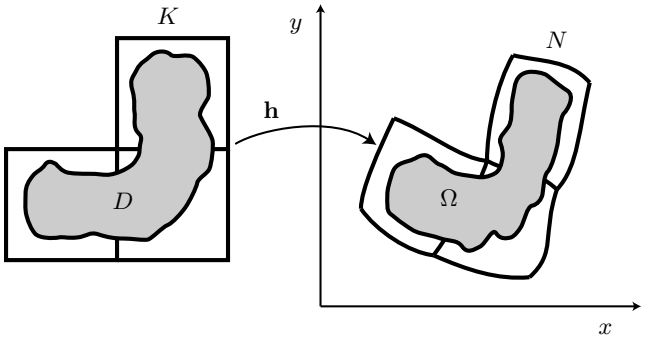


Figure 1: Visualization of the relationship between an object Ω and a control lattice K. The lattice K parameterizes N, which is a superset of Ω, via the function $\mathbf{h}$. The pre-image of Ω under $\mathbf{h}$ is D.

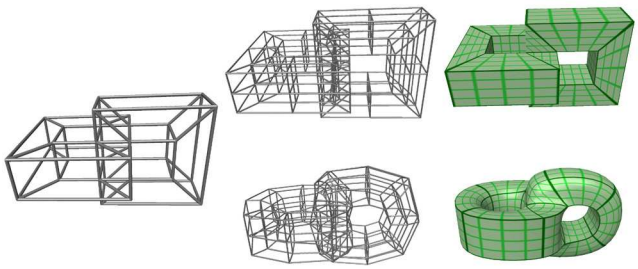


Figure 2: On the left is a control lattice. In the top row the control lattice has been subdivided once (center) and infi nitely (right) using trilinear subdivision. In the bottom row, the MacCracken-Joy scheme (augmented with the sharp surface rules of [13]) has been used. In the rightmost images the object has been textured to indicate the surface of a volumetric parameterization.

We use subdivision rules on the control lattice K to define both the homeomorphism $\mathbf{h}$ and a hierarchical basis $\mathcal{B}$ of functions on Ω. The subdivision framework gives topological flexibility and is intrinsically hierarchical. Our current implementation, based on hexahedral subdivision, supports trilinear subdivision [7], the subdivision solids introduced by MacCracken and Joy [24], which are a generalization of Catmull-Clark subdivision surfaces [8], and the scheme introduced recently by Bajaj et al. [1]. In all three cases, we support only those control lattices that result in hexahedral complexes after one subdivision step. This restriction is required in order to ensure that all objects are parameterized by the control lattice in the obvious way. It still allows a variety of polyhedral cells, including hexahedra, tetrahedra and triangular prisms. While less has been proven about the smoothness of the MacCracken and Joy scheme than that of Bajaj et al., we have found that the former produces surfaces that are more visually pleasing. Therefore we used the trilinear and MacCracken-Joy schemes for the examples shown in this paper.

Both trilinear and MacCracken-Joy subdivision schemes give rise to infinite sequences of nested function spaces [23] spanned by the elements of a hierarchical basis $\mathcal{B} = \{\phi^a(u)\}$. Each finite dimensional space is spanned by a set of functions, one corresponding to each vertex of the subdivided control lattice. Using a standard construction (see, e.g., [40]), we form a hierarchical basis by selecting a linearly independent subset of these basis functions.

At the coarsest level are the basis functions that correspond to the original vertices in the control lattice. We denote the set of basis functions at the coarsest level by $\mathcal{B}_0 = \{\phi_0^a\}$, where the index a ranges over the vertices of $K_0 = K$. Repeated subdivision of K introduces an increasingly fine sequence of complexes K_k (see Figure 2). Applying subdivision to the complex K_k gives rise to basis functions at level k. The basis $\mathcal{B}_k$ is inductively formed from $\mathcal{B}_{k-1}$ by appending the basis functions ϕ_k^a on K_k where a ranges over the set of *odd vertices* of K_k (vertices introduced when we subdivide K_{k-1}). The hierarchical basis $\mathcal{B}$ is the union of the nested sequence

$$\mathcal{B}_0 \subset \mathcal{B}_1 \subset \mathcal{B}_2 \subset \dots \tag{8}$$

Notice that the index k is redundant because each vertex a appears at a unique subdivision level k. We, therefore, drop the index k, writing ϕ^a instead of ϕ_k^a.

The homeomorphism $\mathbf{h}$ is formed by taking a linear combination of the coarse basis functions:

$$\mathbf{h}(u) = \mathbf{h}_a \phi_0^a \tag{9}$$

where $\mathbf{h}_a \in \mathbb{R}^3$. We note that not every set of coefficients $\{\mathbf{h}_a\}$ defines a homeomorphism. Unfortunately, we do not know of an easy way to detect overlap in $\mathbf{h}$, so we resort to numerical approximation to verify that $\mathbf{h}$ is indeed a homeomorphism. Figure 2 shows examples of a trilinear and a MacCracken-Joy subdivision solid that are parameterized by a control lattice. Color Plate 2(b) shows another example of a region parameterized using trilinear subdivision.

Although the elements of $\mathcal{B}$ are defined on all of K, we are only interested in their values on D, so we view them as functions on D, deleting from $\mathcal{B}$ all functions whose support is disjoint from D. Finally, by composing with $\mathbf{h}$, the elements of $\mathcal{B}$ can be treated as functions on Ω. This is convenient because the equations of elasticity are easiest to describe in Euclidean coordinates. In Color Plate 2(c) the vertices of a subdivided control lattice have been colored to indicate the structure of the hierarchical basis.

3.2 Derivation of the Equations of Motion

The kinetic energy $T(\dot{\mathbf{q}})$, the potential energy $V(\mathbf{q})$, and the generalized forces, can all be expressed as integrals over the domain Ω. To derive the equations of motion, we need only express each integral in terms of integrals involving the basis functions ϕ^a.

3.2.1 Kinetic Energy

The standard definition of the kinetic energy of a moving body is:

$$T = \frac{1}{2}\int_\Omega \rho(x)\, \dot{\mathbf{p}} \cdot \dot{\mathbf{p}}\, dV = \frac{1}{2} M^{ab}\, \dot{\mathbf{q}}_a \cdot \dot{\mathbf{q}}_b \tag{10}$$

where $\rho(x)$ is the mass density of the body, and

$$M^{ab} = \int_\Omega \rho\, \phi^a \phi^b\, dV. \tag{11}$$

Equation (10) yields the formula

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) = M\, \ddot{\mathbf{q}}\,. \tag{12}$$

The matrix M, composed of the elements IM^{ab}, where I is a 3×3 identity matrix, is commonly referred to as the *mass matrix*. We discuss its computation in Section 4.1.

3.2.2 Potential Energy

The potential energy of an elastic body is based on measuring the *strain* or distortion present in the body. Green's strain tensor is a common measure of strain:

$$e_{ij} = \frac{\partial d^i}{\partial x^j} + \frac{\partial d^j}{\partial x^i} + \delta_{kl}\frac{\partial d^k}{\partial x^i}\frac{\partial d^l}{\partial x^j}\,. \tag{13}$$

A related concept is that of *stress* (also a tensor), which measures the forces present in a continuous body. For linear (stress is proportional to strain) and isotropic bodies, stress has the following relation to strain:

$$\tau_{ij} = 2G\left\{\frac{\nu}{1-2\nu}tr(e)\delta_{ij} + e_{ij}\right\} \tag{14}$$

where $tr(e) = \delta^{ij}e_{ij}$. The scalar G, called the *shear modulus*, determines how easily the body deforms, and the scalar ν, called *Poisson's ratio*, determines how strains in perpendicular directions relate.

The potential energy V, analogous to computing *work* as force times distance, is computed by taking the componentwise product of the stress and strain tensors:

$$V = \int_\Omega G\left\{\frac{\nu}{1-2\nu}tr^2(e) + \delta^{ij}\delta^{kl}e_{ik}e_{jl}\right\} dV\,. \tag{15}$$

By combining Equations 3, 13, and 15 we can express the elastic potential V and its derivatives (with respect to $\mathbf{q}$) as polynomial functions of $\mathbf{q}$. The coefficients of these polynomials are integrals that can be precomputed. We refer to these coefficients as the *stiffness integrals*. Their exact form can be found in the appendix of [7]. The matrix $\frac{\partial^2 V}{\partial \mathbf{q}\partial \mathbf{q}}$, which is needed during the simulation, is commonly referred to as the *stiffness matrix*.

3.2.3 External Forces

We address two specific types of external forces: gravity ($\mathbf{Q}^g$) and constraints ($\mathbf{Q}^c$). We add their generalized force contributions to compute the aggregate generalized force $\mathbf{Q}^{ext} = \mathbf{Q}^g + \mathbf{Q}^c$.

Gravity. Gravity is an example of a *body force* that affects all points inside the body. We treat gravity as a constant acceleration field specified by the vector $\mathbf{g}$. The gravitational potential energy is then the integral

$$V_g = \int_\Omega \rho\,\mathbf{g}\cdot\mathbf{p}\,dV = \int_\Omega \rho\,\phi^a\mathbf{g}\cdot\mathbf{q}_a\,dV\,. \tag{16}$$

The *generalized gravitational force* is the gradient

$$\mathbf{Q}^g_a = \frac{\partial V_g}{\partial \mathbf{q}_a} = \left(\int_\Omega \rho\,\phi^a\,dV\right)\mathbf{g}\,. \tag{17}$$

The above force can be interpreted as the familiar $m\mathbf{g}$ except that the mass term represents all of the mass associated with a particular basis function. Generalized forces for other conservative force fields can be derived similarly.

Constraints. Using Lagrange multipliers, we support standard constraints that can be described by equations of the form $\mathbf{C} = \mathbf{0}$. For example, we can constrain a body point P with coordinates u_0 to coincide with the arbitrary point P_0 as follows:

$$\mathbf{0} = \mathbf{C}(\mathbf{q}) = P - P_0 = \mathbf{q}_a\phi^a(u_0) - P_0\,. \tag{18}$$

3.2.4 System of Equations

Choosing a finite *active basis*

$$\mathcal{B}_A \subset \mathcal{B}, \tag{19}$$

collecting together the various terms computed above, substituting them into Equation (6), and applying Baumgarte stabilization (see [28]) to our constraints yields the system of ordinary differential equations

$$\begin{bmatrix} \mathbf{M} & \frac{\partial \mathbf{C}}{\partial \mathbf{q}}^T \\ \frac{\partial \mathbf{C}}{\partial \mathbf{q}} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mu\dot{\mathbf{q}} - \mathbf{Q}^{ext} - \frac{\partial V}{\partial \mathbf{q}} \\ -\alpha\dot{\mathbf{C}} - \beta\ddot{\mathbf{C}} \end{bmatrix}. \tag{20}$$

As explained in Section 4.5, the elements of $\mathcal{B}_A$, and consequently the system of equations, adaptively change with each time step of numerical solution of Equation (20).

The Baumgarte stabilization parameters α and β control a damped spring that acts to restore the constraints when they are not being met. Due to the non-linearity of $\frac{\partial V}{\partial \mathbf{q}}$ our system of equations is not linear.

4 Simulation

In this section we describe the computational aspects of solving Equation (20) efficiently.

4.1 Numerical Integration

To speed up the computation of the terms in Equation (20), we precompute the mass, stiffness, and gravity integrals using numerical quadrature. We first subdivide to the desired level and compute the values of all of the basis functions at each of the vertices. After subdividing at least once, the domain is composed of hexahedral cells, which we proceed to split into tetrahedra. We then compute the integrals over each domain tetrahedron using piecewise linear approximations to the basis functions. Since at every vertex the Euclidean coordinates are known, we can compute the spatial derivatives of the basis functions directly without using knowledge about the parameterization of the object by the complex K.

As described in Section 3.1, we represent functions on Ω by restricting functions on K to Ω. The restriction is approximated during numerical quadrature at the level of the tetrahedra mentioned in the previous paragraph. When computing integrals over Ω, we do not integrate over tetrahedra that are deemed to be outside the object. We accomplish this by regularly sampling each tetrahedron and testing whether the sampled point is outside the surface. If all of the sampled points are outside the surface then the tetrahedron is discarded. In our current implementation, tetrahedra that straddle the surface contribute to the computed integrals (an improvement would be to weight the integral over a tetrahedron according to the fraction of sample points inside the object). Color Plate 2(d) shows the tetrahedra that were used to approximate the interior of the dragon model.

The computed integrals involve products of as many as four basis functions (see [7]), so it is important to know which basis functions are nonzero over a given tetrahedron; otherwise, all 4-tuples would be integrated. We accomplish this by storing the basis heights as sparse vectors at each vertex. Our subdivision scheme operates directly on the sparse vectors to compute the basis heights. By examining the sparse vectors of basis heights at each vertex of a tetrahedron, the integrator knows which basis functions are nonzero.

4.2 Solving the ODEs

Once we have precomputed the mass and stiffness terms, we are prepared to solve the system in Equation (20) together with initial values for $\mathbf{p}$ and $\dot{\mathbf{p}}$ (and thus $\mathbf{q}$ and $\dot{\mathbf{q}}$). Solution techniques typically start with known values for $\mathbf{q}$ and $\dot{\mathbf{q}}$ and proceed to compute the values of these variables at a sequence of subsequent points in time.

There are two common classes for solving such systems of differential equations. Explicit techniques compute the future state of the system using information about the state of the system at the current and previous timesteps. Forward Euler and Runga-Kutta are examples of such explicit methods. Implicit techniques express the future state in terms of quantities evaluated at the end of the timestep, in addition to previously known quantities. Implicit methods are much more stable for large timesteps than explicit methods because rather than jumping blindly forward, the conditions at the

future state are taken into consideration. For discussions of implicit methods see [4, 20].

We desire a fast stable solution, so we chose to use an implicit method to solve our system of equations. Applying the method of [4], adapted to our constrained system, results in the following nonlinear system of equations:

$$\begin{bmatrix} \mathbf{M} & \frac{\partial \mathbf{C}}{\partial \mathbf{q}}^T \\ \frac{\partial \mathbf{C}}{\partial \mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{v} \\ \lambda \end{bmatrix} = h \begin{bmatrix} \mathbf{Q}(\mathbf{q}_0 + h(\mathbf{v}_0 + h\Delta\mathbf{v}), \mathbf{v}_0 + \Delta\mathbf{v}) \\ -\alpha\dot{\mathbf{C}} - \beta\ddot{\mathbf{C}} \end{bmatrix} \quad (21)$$

where $\mathbf{Q} = \mu\dot{\mathbf{q}} - \mathbf{Q}^{ext} - \mathbf{Q}^{e}$. We solve the above system of equations for $\Delta\mathbf{v}$ using the Newton-Raphson root-finding method. The cost of this comes primarily from computing the stiffness matrix, which is required to compute the gradient of Equation (21). For our simulations we have found it acceptable to only perform one iteration of Newton-Raphson. This corresponds to linearization of Equation (21) at each timestep (as in [4]), but should not be confused with the commonly used linearization of strain, which is only accurate for small deformations (we discuss such linearization in Section 4.4). The linear systems that need to be solved when performing Newton-Raphson on Equation (21) are symmetric but indefinite, so we solve them using the iterative *minres* algorithm (using sparse matrices, see [35]).

4.3 Runtime Details

Although the interpretation of our object as a lattice is not needed by the ODE solver, we still store a complete lattice at the level of the finest basis functions. This is convenient because of the one-to-one correspondence between lattice vertices and basis functions. For each vertex we store the sparse vector of basis values, which allows us to evaluate functions without using a global synthesis step. In the case where the object being simulated is described by subdivision ($\Omega = N$), we perform a synthesis on the surface of the object whenever it needs to be displayed. In order for the user to be able to click on the surface and set a constraint, we store the surface of the object as a triangle mesh. The triangle vertices store basis function information. This allows us to quickly compute the parametric location of the chosen surface point, select the relevant subset of basis functions, and set up a constraint at that point as in Section 3.2.3.

4.4 Quasi-linearization

For complex models in which there are many basis functions, the full nonlinear equations of elasticity are too expensive to solve interactively because even evaluating the stiffness matrix once per simulation step is costly, and the equations of motion must be linearized. We support two methods of linearization, in addition to the nonlinear solver.

If the deformations are small, the nonlinear terms of strain (Equation (13)) can be dropped, resulting in the traditional quadratic (instead of quartic) elastic potential, and thus a constant stiffness matrix. If the deformations are large but differ only slightly from a rigid motion then the strain can be linearized about a floating frame of reference that roughly tracks the orientation of the object (see [44]). If large deformations are required and significant error is unacceptable, then the full non-linear formulation is necessary.

Our approach to the large-rotation small-deformation scenario deserves further comment. Terzopoulos et al. [44] (and similar formulations in the engineering literature, e.g., [39]) integrate a moving frame of reference into the dynamic equations, adding greatly to the complexity of the exposition and implementation. The frame of reference attempts to track the configuration of the object as if it were a rigid body. Besides the added complexity, another problem is that over time, due to numerical error, the frame of reference will drift out of alignment with the deforming body.

Roughly speaking, the error due to linearizing strain is large when $\nabla\times\mathbf{d}$ is large. Our approach is to choose an orientation of the rest state at the beginning of each timestep in order to minimize $\nabla\times\mathbf{d}$. To avoid the computational expense of optimizing globally, we minimize $\nabla\times\mathbf{d}$ at a single point $\mathbf{x}_0$, located near the rest center of mass of the object. The orientation of the rest state is given by a rotation $\mathbf{R}(\mathbf{x})$ and is determined by solving the following equations:

$$\mathbf{x} + \mathbf{d}(\mathbf{x}) = \mathbf{x} + \mathbf{D}(\mathbf{x}) + \mathbf{R}(\mathbf{x}) \quad (22)$$

$$\nabla\times\mathbf{D}(\mathbf{x}_0) = \mathbf{0} \quad (23)$$

where $\mathbf{D}(\mathbf{x}) = \mathbf{d}(\mathbf{x}) - \mathbf{R}(\mathbf{x})$ is a displacement field relative to the new rest state $\mathbf{x} + \mathbf{R}(\mathbf{x})$. To first order, $\mathbf{R}(\mathbf{x})$ is a rotation by $\frac{1}{2}\|\nabla\times\mathbf{d}\|$ radians about the $\nabla\times\mathbf{d}$ axis. At the beginning of each timestep we rotate the rest state by $\mathbf{R}$ and replace $\mathbf{d}$ by $\mathbf{D}$. Since the rotation during one timestep is typically small, this first-order approximation to $\mathbf{R}(\mathbf{x})$ performs well.

4.5 Adapting the Basis

Because the basis $\mathcal{B}$ is hierarchical, it is possible to adaptively choose the subbasis $\mathcal{B}_A$ in Equation (19) so that detail is added where needed. There are two pertinent questions regarding adaptation: "*how* to adapt?" and "*when/where* to adapt?"

The first question is easily answered in our framework. We choose *a priori* a maximum allowable subdivision level k and precompute the mass and stiffness terms needed to form the system in Equation (20) when $\mathcal{B}_A = \mathcal{B}_k$. These terms are stored in sparse data structures. When a decision is made to add or remove a basis function from the active basis $\mathcal{B}_A$, it is only necessary to add or remove (precomputed) terms from the current mass and stiffness matrices (and in the nonlinear case, to modify $\frac{\partial V}{\partial \mathbf{q}}$ by adding or subtracting higher order terms). We note that the idea of adapting the basis is not new (see, e.g., [16]), and has recently been generalized by Grinspun et al. [19].

We address the second question by a heuristic similar to the one used in [12]: areas of higher deformation require more detail. The elements of $\mathcal{B}_k$ are organized into a tree, with a parent-child relationship between basis functions at adjacent subdivision levels and having intersecting support. Each level of the hierarchy has two separate thresholds for determining when to refine or coarsen. If the deformation is above the *activation threshold* in the region of support of a basis function ϕ_a then the children of ϕ_a are activated. A basis function is deactivated if the deformation is below the *deactivation threshold* in its region of support. As noted in [12], a lower deactivation threshold discourages the system from immediately removing newly introduced basis functions.

4.6 Realtime Simulation

In order to have the appearance of realism, it is important that the simulation be not only fast enough to be interactive, but also to proceed at a consistent pace. Adaptively changing the basis introduces variation in the amount of time required to compute a single timestep. Since our simulator can take large timesteps we can remedy this problem by adjusting the timestep to stay in sync with actual time. For example, when basis functions are added each step of the simulation will take longer due to the increased number of degrees of freedom in the system. We compensate by integrating over a longer period of virtual time during each timestep.

So why not take arbitrarily large timesteps? First, interactive applications demand high frame rates. It is best to display a new state

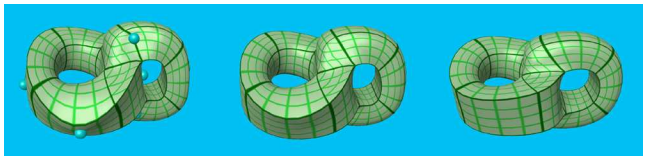


Figure 3: The deformation of an object described by MacCracken-Joy subdivision with sharp surface rules. Upon releasing the constraints (represented as spheres in the first frame), the object dynamically vibrates and eventually returns to its rest shape.

of the system at each video refresh cycle. Second, implicit integration exacts payment for its improved stability. Large timesteps result in unrealistic damping. For these reasons we typically set the timestep to the amount of physical time that lapsed during the previous iteration of simulation and display.

5 Results

We now describe the results of implementing our framework and running a variety of simulations. All of the simulations were performed interactively on a standard desktop PC (Athlon 1.4 GHz, 256 MB ram).

Comparison with Embedding in a Regular Grid. Color Plate 1 shows a comparison between our method and one in which the object is embedded in a regular grid. In the simulation, position constraints were used to stretch the dragon (by pulling on its front and back) and open its mouth. Despite having more degrees of freedom, and thus requiring more computation time, the simulation based on a regular grid is less convincing. Rather than having the mouth open and the body uncoil, as we would expect, the mouth and body seem to stretch uniformly. This effect is caused by basis functions whose support spans the empty regions adjacent to distinct parts of the dragon, thus correlating the motion of parts that would not naturally move together. In contrast, our method produces a more natural deformation because the grid can be made to fit the object much more closely.

Adaptation. Color Plate 2 illustrates our adaptive simulation algorithm applied to the dragon model. The coarsest level basis $\mathcal{B}_0$ has 88 basis functions, while the finest level basis $\mathcal{B}_2$ has 2245 elements. Using the quasi-linear solver, the simulation took about 0.02 seconds per frame when only the coarse basis is active. At the level of adaptation shown, each frame required about 0.1 seconds.

Sharp Features. DeRose and Kass [13] added rules for sharp features to the Catmull-Clark subdivision framework. Since the boundaries of MacCracken-Joy solids are Catmull-Clark surfaces, we can easily include sharp features in our framework.[2] Figure 3 shows a simulation involving an object with sharp surface features.

Virtual Environments. We have implemented a rudimentary collision detection scheme to demonstrate the feasibility of placing our objects in a virtual environment. We use surface constraints to stop vertices on the model from passing through walls in the environment. In Figure 4 a duck is being tossed about in a box interactively. Our quasi-linearization scheme performed as expected; as long as deformations are modest the results appear realistic.

Varying Material Properties. Our system supports material properties that vary both spatially and temporally. Material properties are incorporated during the computation of the stiffness and mass matrices. During the quadrature phase, the values for ν, G, and ρ need not be constant. In addition, because our basis is hierarchical, material properties are smoothly factored into the the mass and stiffness matrices at all levels. For a particular generalized coordinate, the material properties at all points in the support of its

[2] We are not sure what the limitations are of adding sharp features to the surfaces of MacCracken-Joy solids, but it works well in practice.

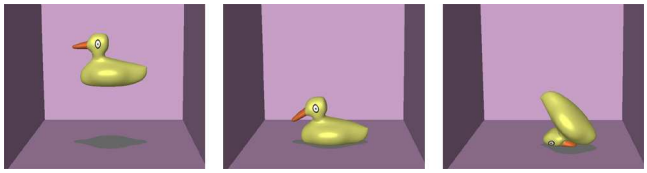


Figure 4: A collection of frames from an interaction with a duck model. The duck is modeled directly as a MacCracken-Joy subdivision solid. The motion of the duck is limited by the floor and wall constraints.

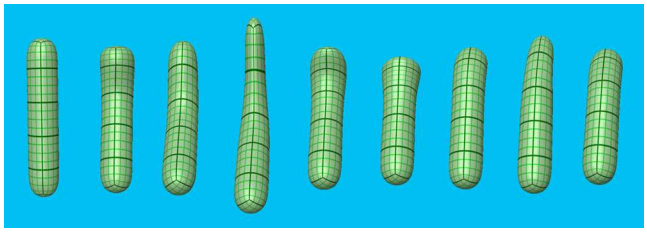


Figure 5: A cucumber-like object (modeled as a MacCracken-Joy subdivision solid), with a longitudinally varying shear modulus G. The object is being shaken by the bottom using a position constraint. The bottom is firm and maintains its shape, while the top deforms drastically.

associated basis function are factored into the computation of the mass and stiffness matrices. Consequently, the material properties of the object are correctly modeled, even when only a subset of the basis is used in the simulation. For instance, the shear modulus G of the object in Figure 5 varies along its length. When it is shaken, one end wobbles like soft rubber while the other remains almost rigid.

An easy way to represent varying material properties over the body is to use the coarse subdivision basis. Then we need only to specify control values at the coarse level vertices. The subdivision rules generate values throughout the object.

We also allow the shear modulus G to be scaled globally at runtime. This does not require re-computation because it uniformly scales the entire stiffness matrix. Scaling G globally makes the object seem more or less firm.

6 Conclusion

In this paper we have presented a framework for the simulation of elastic deformable solids. Our framework is easy to use and creates fast realistic simulations of complex objects. By working within a subdivision framework we inherit topological flexibility and a hierarchical basis. In this framework, we have been able to simulate elastic deformable models of moderate complexity and having spatially varying material properties at interactive rates.

There are many directions for future work. Because interactive simulations are easier to experiment with, our examples involve relatively simple simulations. We plan to apply this framework to more complex scenarios in the future. Another area of future work is to discover uses for the sharp *internal* features supported by the scheme of Bajaj et al. [1]. Real objects do have sharp internal discontinuities, such as at the boundaries of bones. We have also not addressed the problem of self collision, which is important for more general simulations.

Another interesting issue is whether a more sophisticated function basis could be useful in this framework. We have experimented with applying the lifting scheme to our bases, but in our current framework the benefits are unclear and the costs are significant (due to increased density of the mass and stiffness matrices). The choice

of basis would probably be much more critical if we were using a hierarchical solver such as multigrid, another area for future work. A major improvement over the current framework would be to implement an adaptive hierarchical solver with provable error bounds.

Acknowledgments This work was supported by the Animation Research Labs, NSF grants DMS-9803226 and CCR-0092970, an Intel equipment donation, and Microsoft Research.

References

[1] Chandrajit Bajaj, Scott Schaefer, Joe Warren, and Guoliang Xu. A subdivision scheme for hexahedral meshes. *draft*, 2001.

[2] R. E. Bank. *Hierarchical bases and the finite element method*, volume 5 of *Acta Numerica*, pages 1–43. Cambridge University Press, Cambridge, 1996.

[3] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):303–308, July 1992.

[4] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[5] C. Börgers and O. Widlund. On finite element domain imbedding methods. *SIAM J. Num. Anal.*, 27:145–162, 1990.

[6] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, August 1996.

[7] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. Interactive skeleton-driven dynamic deformations. *To appear in the Proceedings of SIGGRAPH 2002*, 2002.

[8] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, September 1978.

[9] George Celniker and Dave Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics*, 25(4):257–266, July 1991.

[10] Fehmi Cirak, Michael Ortiz, and Peter Peter Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–72, April 2000.

[11] Gilles Debunne, Mathieu Desbrun, Alan Barr, and Marie-Paule Cani. Interactive multiresolution animation of deformable models. *Computer Animation and Simulation '99*, September 1999.

[12] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. *Proceedings of SIGGRAPH 2001*, 2001.

[13] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. *Proceedings of SIGGRAPH 98*, pages 85–94, August 1998.

[14] Mathieu Desbrun, Peter Schröder, and Al Barr. Interactive animation of structured deformable objects. *Graphics Interface '99*, pages 1–8, June 1999.

[15] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, July–September 1997.

[16] Steven J. Gortler and Michael F. Cohen. Hierarchical and variational geometric modeling with wavelets. *1995 Symposium on Interactive 3D Graphics*, pages 35–42, April 1995.

[17] Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Proceedings of SIGGRAPH 93*, pages 221–230, August 1993.

[18] Jean-Paul Gourret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):21–30, July 1989.

[19] E. Grinspun, P. Krysl, and P. Schröder. Charms: A simple framework for adaptive simulation. *To appear in the Proceedings of SIGGRAPH 2002*, 2002.

[20] Michael Hauth and Olaf Etzmuss. A high performance solver for the animation of deformable objects using advanced numerical methods. *Computer Graphics Forum (Proceedings of Eurographics 2001)*, 20(3), 2001.

[21] Doug L. James and Dinesh K. Pai. Artdefo - accurate real time deformable objects. *Proceedings of SIGGRAPH 99*, pages 65–72, August 1999.

[22] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. Parish. Simulating facial surgery using finite element methods. *Proceedings of SIGGRAPH 96*, pages 421–428, August 1996.

[23] Michael Lounsbery, Tony D. DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, January 1997.

[24] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. *Computer Graphics (Proceedings of SIGGRAPH 96)*, pages 181–188, 1996.

[25] G.I. Marchuk, Y.A. Kuznetsov, and A.M. Matsokin. Fictitious domain and domain decomposition methods. *Sov. J. Numer. Anal. Math Modeling*, 1, 1986.

[26] K. McDonnell and H. Qin. Dynamic sculpting and animation of free-form subdivision solids. In *Proceedings of the Conference on Computer Animation*, pages 126–133. IEEE Press, 2000.

[27] Kevin T. McDonnell and Hong Qin. FEM-based subdivision solids for dynamic and haptic interaction. In *Proceedings of the Sixth Symposium on Solid Modeling and Application*, pages 312–313. ACM Press, 2001.

[28] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):309–312, July 1992.

[29] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. *Proceedings of SIGGRAPH 99*, pages 137–146, 1999.

[30] J. T. Oden and J. N. Reddy. *An Introduction to the Mathematical Theory of Finite Elements*. John Wiley and Sons, Ltd., New York, London, Sydney, 1982.

[31] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):215–222, July 1989.

[32] G. Picinbono, H. Delingette, and N. Ayache. Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model. In *Proceedings of the Third International Conference on Medical Robotics, Imaging and Computer Assisted Surgery: MICCAI 2000*, pages 643–652, 2000.

[33] John C. Platt and Alan H. Barr. Constraint methods for flexible models. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):279–288, August 1988.

[34] P. M. Prenter. *Splines and Variational Methods*. John Wiley and Sons, 1975.

[35] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C, 2nd. edition*. Cambridge University Press, 1992.

[36] Hong Qin, Chhandomay Mandal, and Baba C. Vemuri. Dynamic catmull-clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215–229, 1998.

[37] S. H. Martin Roth, Markus H. Gross, Silvio Turello, and Friedrich R. Carls. A bernstein-bézier based approach to soft tissue simulation. *Computer Graphics Forum*, 17(3):285–294, 1998.

[38] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, August 1986.

[39] A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, 1998.

[40] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgann Kaufmann, San Francisco, CA, 1996.

[41] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):269–278, August 1988.

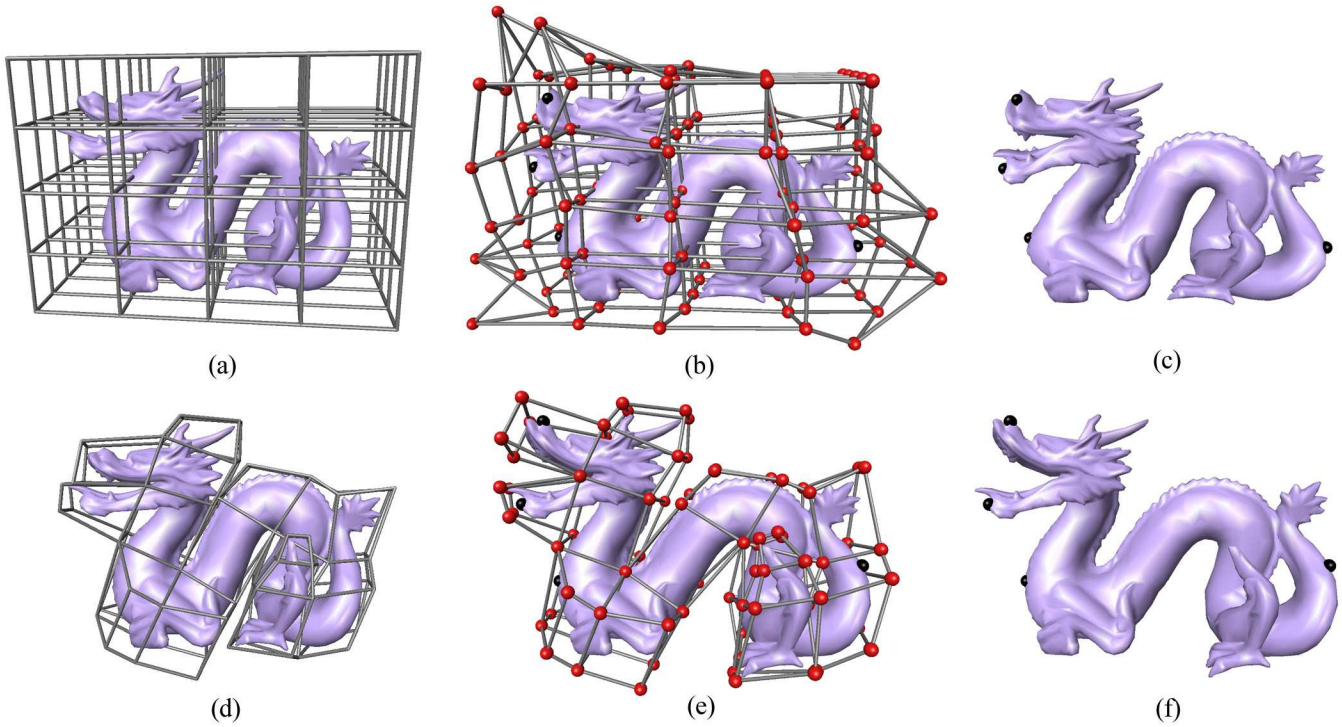
[42] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87)*, 21(4):205–214, July 1987.

[43] Demetri Terzopoulos and Hong Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994.

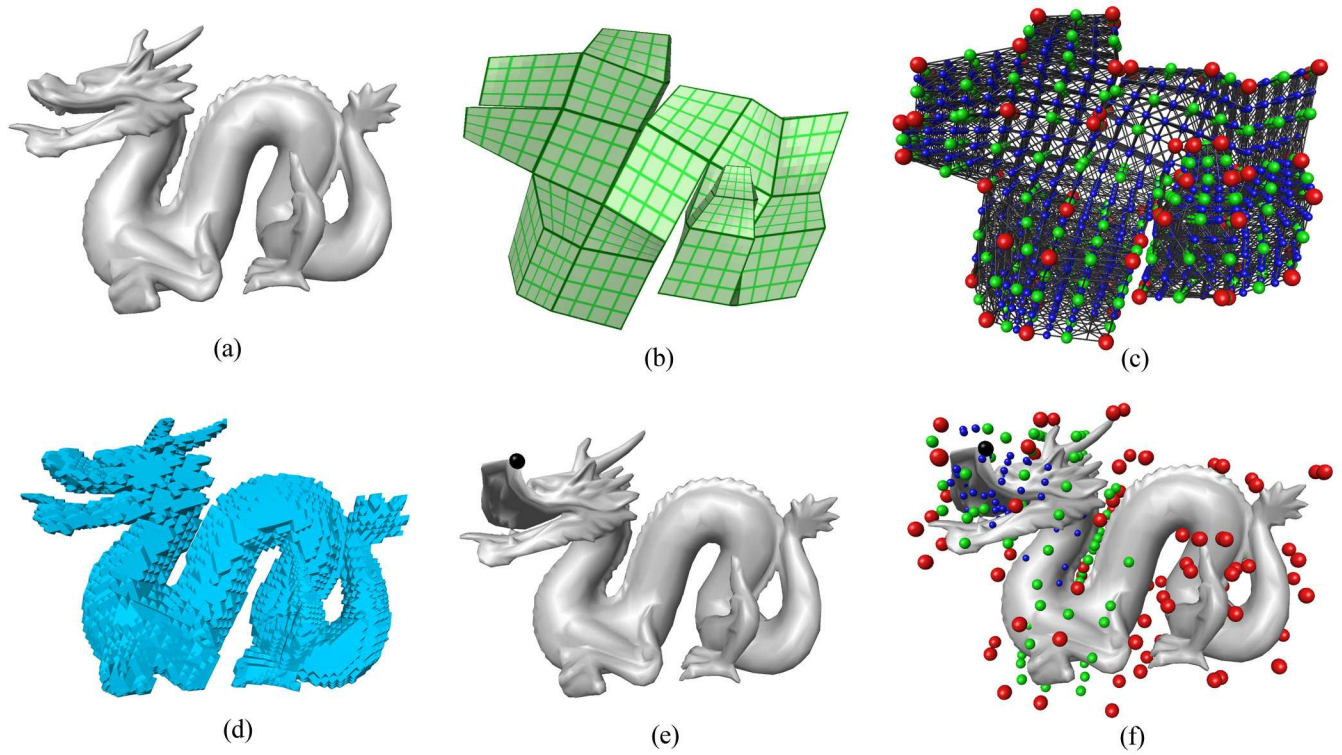
[44] Demetri Terzopoulos and Andrew Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, 8(6):41–51, November 1988.

[45] Henrik Weimer and Joe Warren. Subdivision schemes for fluid flow. *Proceedings of SIGGRAPH 99*, pages 111–120, August 1999.

[46] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):243–252, August 1990.



Color Plate 1: A comparison of simulations using a regular grid (top row) and a subdivision control lattice (bottom row). The left images show the dragon in its rest state surrounded by (a) a regular grid (375 degrees of freedom) and (d) a subdivision control lattice (264 degrees of freedom). The center images show a simulation in which position constraints have been used to stretch the dragon and open its mouth. Black spheres represent position constraints and red spheres represent the degrees of freedom of the system. In both cases, trilinear basis functions were used. The rightmost images show the deformed state of the dragon.



Color Plate 2: (a) The dragon in its rest state. (b) A trilinear subdivision volume that surrounds the dragon. The surface has been textured to indicate how the volume is parameterized. (c) The hierarchical structure of the subdivision volume. Red spheres correspond to level 0 basis functions, green to level 1, and blue to level 2. (d) The tetrahedral approximation of the dragon that is used to compute integrals over its interior (i.e., for numerical quadrature). (e) The dragon being deformed by a position constraint pulling on the upper lip. (f) The basis functions introduced by the adaptive solver.