# A Multiresolution Framework for Dynamic Deformations

Steve Capell Seth Green Brian Curless Tom Duchamp Zoran Popović

University of Washington
Department of Computer Science and Engineering
Technical Report #02-04-02

## Abstract

We present a novel framework for dynamic simulation of elastically deformable solids. Our approach combines classical finite element methodology with subdivision wavelets to meet the needs of computer graphics applications. We represent deformations using a wavelet basis constructed from volumetric Catmull-Clark subdivision. Catmull-Clark subdivision solids allow the domain of deformation to be tailored to objects of arbitrary topology. The domain of deformation can correspond to the interior of a subdivision surface or can enclose an arbitrary surface mesh. Within the wavelet framework we develop the equations of motion for elastic deformations in the presence of external forces and constraints. We solve the resulting differential equations using an implicit method, which lends stability. Our framework allows trade-off between speed and accuracy. For interactive applications, we accelerate the simulation by adaptively refining the wavelet basis while avoiding visual "popping" artifacts. Off-line simulations can employ a fine basis for higher accuracy at the cost of more computation time. By exploiting the properties of smooth subdivision we can compute less expensive solutions using a trilinear basis yet produce a smooth result that meets the constraints.

## 1 Introduction

Physical simulation of dynamic, deformable bodies has numerous applications in computer graphics, including animation in film, video games, and surgical simulation. In this paper, we explore a new framework for simulating the dynamics of elastically deformable solids that is appropriate for interactive and off-line simulations.

Our simulator supports a number of desirable properties:

*Physically-based.* Because the mechanical behaviors of solids are derived from differential properties of continuous media, our framework is based on continuum mechanics.

*Dynamics with constraints.* For realistic animation, our framework supports dynamics using the Lagrangian formulation. Constraints, which are necessary to simulate behaviors like collisions and manipulations, are supported using Lagrange multipliers.

*Support for arbitrarily shaped objects.* For objects with interesting shapes, it is important to ensure that the deformation has the appearance that one would expect. We accomplish this through the use of parameterization and embedding objects in custom control lattices.

*Support for objects with spatially varying material properties.* Deformable solids can be, for example, firmer in some places and more gelatinous in others. Our framework accommodates varying material properties.

*Principled decoupling of base geometry and deformations.* In order to simulate complex objects interactively (for example) it is important that the resolution of the deformation not be tied to the complexity of the underlying surface. In our framework, the resolution of the surface and simulation are decoupled.

*Fast, stable solution.* The naive solutions for deformable bodies require many iterations per time step and then must take small time steps in order to achieve stability. We use an implicit method that ensures stability at the cost of some artificial damping.

*Speed-accuracy trade-off.* While accuracy is often desirable, it is not always essential. Our framework allows the user to determine the trade-off between speed and accuracy. Coarse grids can be used to achieve the fastest simulation. For higher accuracy the coarse grid needs only to be refined using subdivision.

*Adaptation.* As a simulation runs, not all of the detail is necessary to achieve the desired accuracy. By using a multiresolution basis, our framework can support schemes that adapt the mesh resolution in order to concentrate the computational effort where it is likely to have the most impact.

We represent our objects and deformation domains as Catmull-Clark (or trilinear) subdivision volumes (see Figure 1). The surfaces inherit all of the desirable properties of Catmull-Clark surfaces (e.g., sharp features and straightforward handling of arbitrary topologies), while providing a volumetric domain for representing solid deformations which are explicitly defined both inside the volume and at every point on the surface. Using the principles of subdivision, we construct a multiresolution, lazy wavelet basis for fast computation of solid deformations. We then formulate the equations of motion for a dynamically deforming elastic solid, in terms of the wavelet basis. This formulation defines the temporal behavior of the wavelet coefficients in the presence of body forces and constraint forces. The wavelet 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. Our simulation method is described in Section 4 and a variety of extensions are described in Section 5. The extensions include realtime simulation, quasi-linearization of the equations of motion, use of trilinear basis function, adaptation, and the embedding of complex objects in simple domains. In section 6 we describe some resulting interactive simulations built on our framework, and we conclude with a discussion in section 7.

## 2 Related Work

Early work on deformations focused on non-dynamic techniques. For example, the introduction of free-form deformations by Sederberg et al. [28] allowed objects to be deformed independent of their structure by embedding them in easily-parameterized domains. Two important extensions were the use of unstructured lattices by MacCracken and Joy [18] and the introduction of dynamics by Faloutsos et al. [10]. Our framework builds on both of these extensions, using volumetric Catmull-Clark lattices as in [18], and embedding objects in dynamic free-form lattices as in [10]. But unlike [10], where a diagonal stiffness matrix is employed, we simulate the dynamics of the embedded object.

The use of physically-based deformable models in graphics was pioneered by Terzopoulos et al. [32]. The original work applied the Lagrangian equations of motion using a finite difference scheme to simulate elastic objects with regular parameterizations. This framework was extended to include inelastic behaviors [31], and to handle stiff rotating bodies using linearized equations [33].

Following their introduction, physically-based deformations were extended in many ways. Platt and Barr [24] introduced better constraint handling via Lagrange multipliers. Pentland and Williams [23] obtained realtime simulations by using only a few vibration modes. Witkin and Welch [35] introduced the use of low-order polynomial deformations to achieve fast deformations. Baraff and Witkin [1] added non-penetration constraints to this framework. Metaxas and Terzopoulos [21] combined global deformations with local finite element deformations.

Implicit solvers are enjoying a renaissance in dynamic deformations. Terzopoulos et al. [32][31] used semi-implicit solvers in their initial work. Baraff and Witkin [2] used a fully implicit scheme to greatly improve the speed and stability of cloth simulation. Desbrun et al. [9] used a semi-implicit scheme to stabilize stiff systems.

Hierarchical methods have also been used to speed up simulations and allow more detail in interactive systems. Terzopoulos et al. [31] employed a multigrid solver on a rectangular domain. Debunne et al. [7] created interactive simulations using an adaptive octtree representation, adaptive in both space and time. To animate a surface, the surface points are linked to the grid by a weighting scheme. This framework was later extended to use finite elements over an unstructured hierarchy of tetrahedral meshes [6].

For some application, dynamic motion has not been deemed necessary, so static and quasi-static methods have been employed [13] [14] [3] [15] [27]. Since our interest is in realistic motion, we build on dynamic methods.

As noted in the introduction, we employ a wavelet scheme based on 3D subdivision. In the last decade, subdivision and wavelets have enjoyed wide use in computer graphics as a tool for efficient solution of many problems, including, e.g., modeling [11] and rendering [12]. 3D subdivision was first developed by MacCracken and Joy [18] as an extension to Catmull-Clark subdivision surfaces for the purpose of defining free-form deformations of arbitrary topology. More recently, Weimer and Warren [34] employed 3D subdivision to solve PDEs associated with fluid flow. Cirak et al. [5] 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 [19] and then applied the finite-element methodology to the problem [20]. Our framework builds on many of these methods, but differs from each of them significantly.

## 3 Formulation

We have chosen to work within the framework of Catmull-Clark subdivision volumes as introduced by MacCracken and Joy [18], which are a generalization of Catmull-Clark subdivision surfaces [4]. An example of applying volumetric Catmull-Clark subdivision to a simple cube is shown in Figure 1. Subdividing indefinitely results in a solid ball that is parameterized by the original cube. Our framework supports only a subset of all possible Catmull-Clark control lattices: those that result in hexahedral meshes after one subdivision step. This restriction is required in order to ensure that

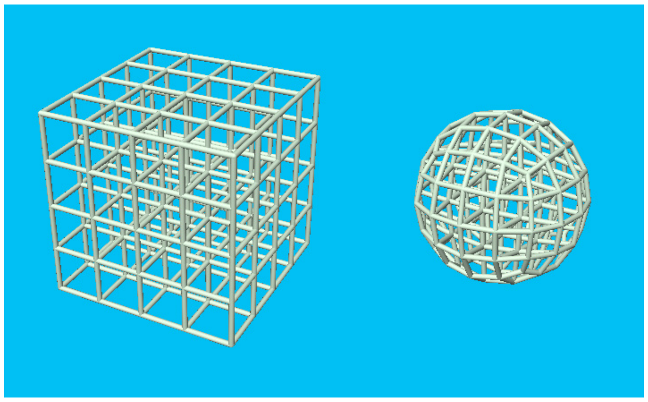

Figure 1: A ball resulting from Catmull-Clark subdivision (on the right) and its associated parametric domain (on the left), both shown after two levels of subdivision.

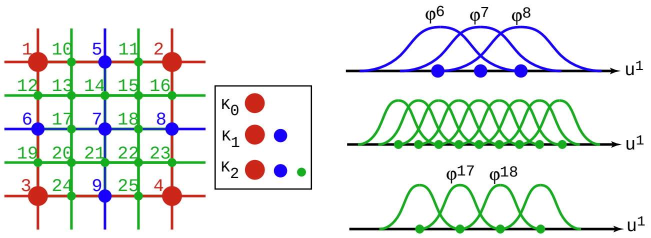

Figure 2: On the left is a visualization of a complex $K = K_0$ in red and two finer subdivisions in blue and green. In our framework, the complex is actually three dimensional, but we show the two dimensional case for convenience. Centered at each vertex of $K_0$ is a Catmull-Clark basis function. After subdividing $K_0$ once, we introduce new vertices that belong to $K_1$ (in addition to the vertices of $K_0$, as shown in the key next to the grid). We could place Catmull-Clark basis functions (half as big in each dimension with respect to the coarser level) at all vertices of $K_1$. Instead, we trim this overrepresentation by only placing the new functions at the newly created vertices (shown in blue). The process continues recursively (e.g., new basis functions at the green vertices). Note that, since only one basis function is centered at each vertex, the index of that vertex uniquely identifies the basis function. On the right we see a one dimensional visualization of a slice through the basis functions along the line from vertex 6 to vertex 8 as the parameter $u^1$ varies. On the top, we see one set of basis functions. In the middle, we see an overcomplete basis consisting of the next level scaling functions. On the bottom we see the lazy wavelets formed by discarding every other basis function.

all objects are parameterized by the control lattice in the obvious way. It still allows a variety of cell shapes including hexahedra, tetrahedra and triangular prisms.

### 3.1 Volumetric Subdivision Wavelets

Subdivision schemes give rise to wavelets [17]. We build a wavelet basis based on the construction of *lazy wavelets* found in [30], except that our scaling functions are not piecewise-linear. At the coarsest level are the basis functions that correspond to the original vertices in the control lattice. We denote the Catmull-Clark basis functions by $\varphi_0^a$, where the index $a$ ranges over the vertices of $K_0 = K$.

While the Catmull-Clark basis functions serve as the scaling functions for our wavelet framework, we have some latitude in choosing the wavelets themselves. For simplicity and efficiency we have chosen to use a lazy wavelet basis (see Figure 2).

In particular, repeated subdivision of $K$, introduces an increasingly fine sequence of complexes $K_k$ (see Figure 1). Applying Catmull-Clark subdivision to the complex $K_k$ gives rise to basis functions at level $k$. The *lazy wavelets at level* $k$ are the Catmull-

Clark basis functions $\varphi_k^a$ on $K_k$ where $a$ ranges over the set of *odd vertices* of $K_k$ (vertices introduced when we subdivide $K_{k-1}$). 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 $\varphi^a$ instead of $\varphi_k^a$.

## 3.2 Wavelet Representation of Deformations

The objects that we consider can be described at rest using a *lazy wavelet expansion*[1]:

$$\mathbf{r}(u) = \sum_a \mathbf{r}_a \varphi^a(u) = \mathbf{r}_a \varphi^a(u) \tag{1}$$

where $\mathbf{r}_a$ is a 3-dimensional vector, and $\mathbf{r}(u)$ is a homeomorphism between a complex $K$ and $\Omega$, a subset of $\mathbf{R}^3$:

$$\mathbf{r} : K \longrightarrow \Omega \subset \mathbf{R}^3 \;:\; u \mapsto \mathbf{r}(u) \tag{2}$$

Objects described by simple Catmull-Clark subdivision have trivial lazy wavelet expansions in which only the coarsest basis functions have non-zero coefficients [2].

Because $\mathbf{r}(u)$ is a homeomorphism, any function of $K$ is also a function of $\Omega$ and vice-versa. In particular, our wavelets $\varphi^a$ can be treated as functions of $\Omega$. This is a great convenience because the equations of motion for elastic bodies are easiest to describe in Euclidean coordinates. In particular, the displacement of an object is most naturally thought of as a function of the rest coordinates of the object. We represent the displacement in wavelet expanded form:

$$\mathbf{d}(x,t) = \mathbf{q}_a(t)\varphi^a(x) \tag{3}$$

where $\mathbf{q}_a(t)$ is a time-dependent 3-dimensional vector. Our goal is to compute the value of the coefficients $\mathbf{q}_a$ which describe the displacement of the object as it deforms over time. Finally, the shape of the object over time is:

$$\mathbf{p}(x,t) = (\mathbf{r}_a + \mathbf{q}_a(t))\varphi^a(x) \tag{4}$$

Figure 3 illustrates the relationship between the complex $K$ composed of polyhedral cells (such as $C$) and the configuration of the object in Euclidean coordinates.

## 3.3 Equations of Motion

We model the dynamics of the deformable body as a system of second order ordinary differential equations that is obtained by applying the finite element method to the Lagrangian formulation of the equations of elasticity (see [29][22][25]). 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 $\mathbf{q}_a(t)$ is a 3-dimensional vector.

Due to equation (4), we can express both kinetic energy $T$ and potential energy $V$ in the form

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

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

$$\frac{d}{dt}\left(\frac{\partial T(\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}}\right) + \frac{\partial V(\mathbf{q})}{\partial \mathbf{q}} + \mathbf{Q}^{ext} - \mu\dot{\mathbf{q}} = 0 \tag{5}$$

[1]Throughout this paper, the Einstein summation convention is in force: whenever a term contains an index as both a subscript and a superscript, the term implies a summation over the range of that index

[2]Not all objects described by subdivision correspond to homeomorphisms; we limit ourselves to those that do, i.e. natural objects that do not intersect themselves.

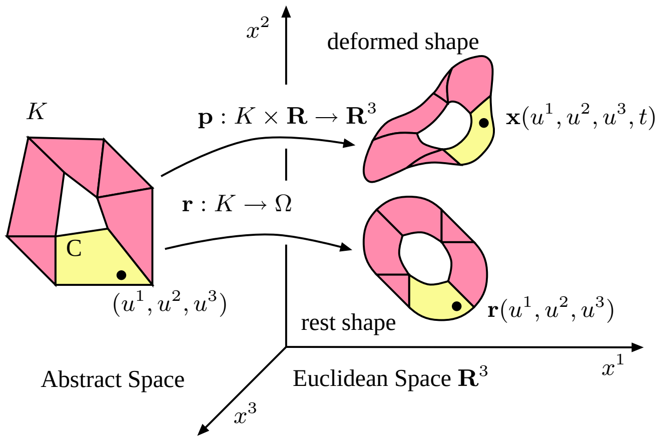
$x^2$
deformed shape
$K$
$\mathbf{p} : K \times \mathbf{R} \rightarrow \mathbf{R}^3$
$\mathbf{x}(u^1, u^2, u^3, t)$
$\mathbf{r} : K \rightarrow \Omega$
C
$(u^1, u^2, u^3)$
$\mathbf{r}(u^1, u^2, u^3)$
rest shape
Abstract Space
Euclidean Space $\mathbf{R}^3$
$x^1$
$x^3$

Figure 3: Visualization of a cell complex, rest shape, and deformed shape. The complex $K$ consists of three dimensional polyhedral cells (represented figuratively here as polygons), such as the one highlighted in yellow and labeled $C$. Each cell has a three dimensional embedding into the rest shape $\mathbf{r}$ and deformed shape $\mathbf{p}$ of an object. Within a cell of the complex, we can address a point in terms of its affine coordinates $(u^1, u^2, u^3)$ which have corresponding embeddings in the rest and deformed shapes.

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 internal damping (a more physically-based damping force would be straight-forward).

The derivation of the terms of equation 5 that follows will yield a system of ODEs to be solved in generalized coordinates.

## 3.4 Kinetic Energy

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

$$T = \frac{1}{2}\int_\Omega \rho(x)\, \dot{\mathbf{p}} \cdot \dot{\mathbf{p}}\, d\Omega = \frac{1}{2} M^{ab}\, \dot{\mathbf{q}}_a \cdot \dot{\mathbf{q}}_b \tag{6}$$

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

$$M^{ab} = \int_\Omega \rho\, \varphi^a \varphi^b\, d\Omega. \tag{7}$$

Equation (6) yields the formula

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) = M\,\ddot{\mathbf{q}}\,. \tag{8}$$

We call the matrix $M$ composed of the elements $M^{ab}$ the *mass matrix*. We discuss its computation in Section 4.1.

## 3.5 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 that handles large deformations:

$$e_{ij} = \frac{\partial d^i}{\partial x^j} + \frac{\partial d^j}{\partial x^i} + \delta_{kl}\frac{\partial d^k}{\partial x^i}\frac{\partial d^l}{\partial x^j} \tag{9}$$

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:

$$\tau_{ij} = 2G\left\{\frac{\nu}{1-2\nu} tr(e)\delta_{ij} + e_{ij}\right\} \tag{10}$$

where $tr(e) = \delta^{ij} e_{ij}$. The scalar $G$, called the *shear modulus*, determines how easily the body deforms, and the scalar $\nu$, 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:

$$V = G \int_{\Omega} \left\{ \frac{\nu}{1-2\nu} tr^2(e) + \delta^{ij}\delta^{kl} e_{ik} e_{jl} \right\} d\Omega \qquad (11)$$

By combining equations (3), (9), and (11) 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. The exact form of these derivatives can be found in appendix A.1.

## 3.6 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 $\mathbf{Q}^{ext} = \mathbf{Q}^g + \mathbf{Q}^c$.

### 3.6.1 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

$$V_g = \int_K \rho\, \mathbf{g} \cdot \mathbf{p} = \int_K \rho \varphi^a \mathbf{g} \cdot \mathbf{q}_a \,.$$

The *generalized gravitational force* is the gradient

$$\mathbf{Q}^g_a = \frac{\partial V_g}{\partial \mathbf{q}_a} = \left( \int_K \rho \varphi^a \right) \mathbf{g} \qquad (12)$$

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.

### 3.6.2 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:

$$\mathbf{0} = \mathbf{C}(\mathbf{q}) = P - P_0 = \mathbf{q}_a \varphi^a(u_0) - P_0 \qquad (13)$$

## 3.7 System of Equations

Collecting together the various terms computed above, substituting them into the Euler-Lagrange equation (5), and applying Baumgarte stabilization (see [21]) to our constraints yields the system of equations

$$\begin{bmatrix} \mathbf{M} & \frac{\partial \mathbf{C}}{\partial \mathbf{q}}^T \\ \frac{\partial \mathbf{C}}{\partial \mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mu\dot{\mathbf{q}} - \mathbf{Q}^{ext} - \mathbf{Q}^e(\mathbf{q}) \\ -\alpha\dot{\mathbf{C}} - \beta\ddot{\mathbf{C}} \end{bmatrix} \qquad (14)$$

where $\mathbf{Q}_e(\mathbf{q})$ is the force due to the elastic potential (see appendix A.1). Due to the non-linearity of $\mathbf{Q}_e$ our system of equations is not linear.

# 4 Simulation

In this section we describe in more detail the computational aspects of solving equation (14) efficiently.

## 4.1 Numerical Integration

In order to compute the gravity terms and the mass and stiffness matrices we precompute the integrals in equations (7), (18), and (12). The integration is done numerically using the following steps:

1. Subdivide the domain to the desired level for numerical integration (at least once).

2. Compute the values of all of the basis functions at each of the vertices.

3. Tetrahedralize the domain. After subdividing at least once, the domain is composed of only hexahedral cells. We then divide each of these cells into tetrahedral cells. This step is performed in order to approximate functions on the domain as piecewise linear.

4. Compute the integrals over each domain tetrahedron using piecewise linear approximations to the basis functions. Since at every vertex the rest 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 $K$.

## 4.2 Solving the ODEs

Once we have precomputed the mass and stiffness terms, we are prepared to solve the system (14) 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. Baraff and Witkin give an excellent discussion on the use of implicit methods in [2].

We desire a fast stable solution, so we chose to use an implicit method to solve our system of equations. Applying the method of [2], adapted to our constrained system, results in the following nonlinear system of equations:

$$\begin{bmatrix} \mathbf{M} & \frac{\partial \mathbf{C}}{\partial \mathbf{q}}^T \\ \frac{\partial \mathbf{C}}{\partial \mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{v} \\ \lambda \end{bmatrix} = h \begin{bmatrix} \mathbf{Q}^{all}(\mathbf{q}_0 + h(\mathbf{v}_0 + h\Delta\mathbf{v}), \mathbf{v}_0 + \Delta\mathbf{v}) \\ -\alpha\dot{\mathbf{C}} - \beta\ddot{\mathbf{C}} \end{bmatrix} \qquad (15)$$

where $\mathbf{Q}^{all} = \mu\dot{\mathbf{q}} - \mathbf{Q}^{ext} - \mathbf{Q}^e$. 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 gradient and Hessian of the elastic potential $V$, which are required to compute the value and gradient of equation (15). In most cases we find it acceptable to only perform one iteration of Newton-Raphson. This corresponds to linearization of equation (15) at each timestep (like Baraff and Witkin did), but should not be confused with the commonly used linearization of strain, which is only valid for infinitesimal rotations. Our system handles large rotations nicely. The linear systems that need to be solved when performing Newton-Raphson on equation (15) are symmetric but indefinite, so we solve them using the iterative *minres* algorithm (using sparse matrices, see [26]).

### 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 wavelets. 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 wavelet synthesis step. We perform a wavelet 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 triangles refer to the vertices in the volumetric lattice, which 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 3.6.2.

## 5 Extensions

Now that our basic framework is in place we describe some extensions to the framework that improve its performance for interactive applications. These include real-time simulation, quasi-linearization, the use of trilinear basis functions while maintaining smoothness, adaptation of the wavelet basis, and support for objects that are not parameterized by a cell complex.

### 5.1 Real-time 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 time steps we can remedy this problem by adjusting the timestep to stay in sync with actual time. For example, when wavelets are added because a new constraint is placed, a 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 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 always set the timestep to the amount of physical time that lapsed during the previous iteration of simulation and display.

### 5.2 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. For these cases we support two traditional approximations. If rotations of the object are not required, and the deformations are modest, the nonlinear terms of strain (equation (9)) can be dropped, resulting in a quadratic (instead of quartic) elastic potential, and thus a constant stiffness matrix. If rotations are required but deformations are modest the strain can be linearized about a floating frame of reference that roughly tracks the orientation of the object (see [33]). 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. [33] (and similar formulations in the engineering literature, e.g. [29]) 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.

Our approach is simple and avoids the drifting problem, while addressing the fundamental issue, which is to ensure that the deformation is measured with respect to a rest state that has been rotated to align with the (mildly) deformed body. Instead of tracking a rotating frame of reference that has complicated (yet irrelevant) behavior, we simply choose an appropriate orientation for the undeformed object at the beginning of each simulation step. Our choice of orientations is simple. At the beginning of the simulation we choose a reference point on the interior of the object. Before each step we rotate the *reference* rest shape so that the displacement field contains no rotational component at the reference point. The displacement field is adjusted to be relative to the rotated rest state. For small deformations, the appropriate rotation can be obtained by measuring the curl of the displacement field (which is only accurate for infinitesimal rotations, but is self-correcting over multiple time steps). This method has the desired effect of reducing the mismeasure of potential energy due to linearization and rotation, and eliminating it completely in the presence of only rigid body motion.

### 5.3 Trilinear Basis

Our framework as described above uses smooth Catmull-Clark basis functions. This is an advantage for graphics applications because the deformed state of the object is always smooth. However, the computation involving smooth basis functions is expensive (due to overlap between basis functions) and our differential equations only require the existence of first derivatives. Trilinear basis functions are an alternative that fits nicely into our framework; they can be easily generalized to our case where cells are oddly shaped but subdivide into hexahedra after one subdivision step. But trilinear basis functions produce unappealing non-smooth deformations. Since the structure of the control lattice provides for smoothing via subdivision, it is tempting to solve the PDEs using trilinear basis functions and then smooth the results. But the constraints will not be met using this approach. In order to meet the constraints we can use the Catmull-Clark basis function for the constraint computations in equation (13) while using the trilinear basis functions for the actual simulation. In this manner we can achieve a smooth deformation that meets the constraints, at the reduced cost (and reduced accuracy) of using the trilinear basis. Figure 4 shows a simple example using this method.

### 5.4 Adapting the Wavelet Basis

Since our basis is multiresolution, it is possible to adapt the basis 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, “*how* to adapt?”, is easily answered in our framework. We precompute the mass and stiffness terms for the entire basis that may *potentially* be used and store them in sparse data structures. When a decision is made to add or remove a basis function, we need only to add or remove terms from the current set of mass and stiffness matrices (and in the nonlinear case, higher order terms). This simplicity comes from the fact that we establish the basis *a priori* rather than constructing a new basis based on

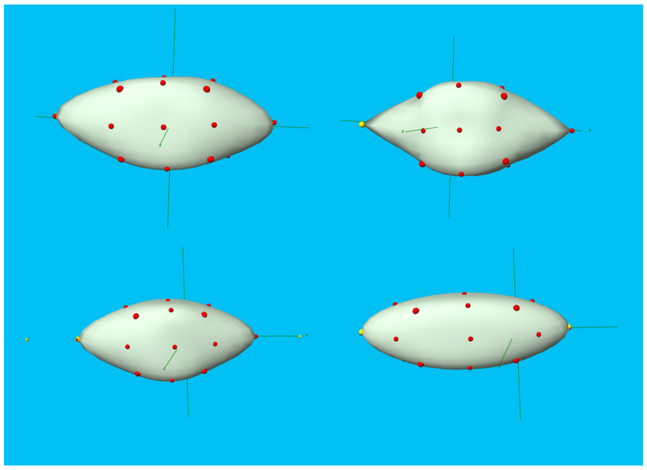

Figure 4: The deformation in the upper left uses 2 levels of smooth Catmull-Clark basis functions, requiring about 0.1 seconds of simulation per frame (using linear strain). The deformation in the upper right uses trilinear basis functions, giving a less pleasing result due to discontinuities, but using about half the computation time. The deformation in the lower right uses the trilinear basis for simulation but reconstructs the result using the Catmull-Clark basis, but the constraint are not met (the two yellow spheres away from the surface indicate where the surface should be). Finally, the deformation in the lower right uses the trilinear basis to compute physical properties but uses the smooth reconstruction to compute the constraints. Note that the constraints imposed in the above examples were not all identical, but were roughly equivalent.

an adapted mesh. In our framework it is the basis that is adapted, not the mesh. We note that the idea of adapting the basis rather than the geometry is not new (see, e.g.,[11]), and has recently been generalized by Krysl et al. [16].

For the second question, "*when/where* to adapt?", we can look to prior work by other researchers. Gortler and Cohen [11] adapted near wavelets having large coefficients. Debunne et al. [6] adapted in areas of high curvature. We have experimented with these schemes, in addition to a scheme by which we simply introduce wavelets that have support overlapping with constraints. In all of these cases disturbing "popping" artifacts appear because the basis changes suddenly during the simulation, instantaneously allowing the simulation more (or less) freedom.

We address the popping problem by gradually introducing and removing basis functions. We begin by simulating the degrees-of-freedom (DOFs) that correspond to the finest active wavelets using simple *harmonic oscillator* dynamics ("simple DOFs"). As we adapt the basis we maintain the property that the DOFs that are being simulated using the full dynamic model ("full DOFs") are "protected" by a layer of simple DOFs. If the magnitude of a simple DOF is above a user-specified threshold, it is upgraded to a full DOF and its children (finer wavelets sharing its support) are activated as simple DOFs. Likewise, if a simple DOF has low magnitude and is not part of the protective layer it is disabled.

We distinguish two types of transition: the introduction or removal of an simple DOF, and the transition between a simple DOF and a full DOF. When an simple DOF is introduced or disabled it first passes through a transition state for a fixed length of simulation time, parameterized by $\beta_a \in [0..1]$. During this time the stiffness coefficient $k_a$ is replaced by $\beta_a k_a + (1 - \beta_a)k_\infty$, where $k_\infty$ represents a very large stiffness. Thus when an simple DOF is introduced it is initially not affected by constraints but over the transition interval it becomes more pliant. Likewise, when a simple DOF is disabled the transition to $k_\infty$ pulls it toward zero.

Transitions between full DOFs and simple DOFs are more complicated because we must interpolate between two physical models. In this case we also parameterize the transition by $\beta_a$. If a DOF is in transition, we divide it into $\mathbf{q}_a^{full} = \beta_a \mathbf{q}_a$ and $\mathbf{q}_a^{simple} = (1 - \beta_a)\mathbf{q}_a$. We then combine the full non-linear mass and stiffness terms computed using $\mathbf{q}_a^{full}$ with the simple DOF mass and stiffness terms computed using $\mathbf{q}_a^{simple}$. The results of this scheme can be seen in the accompanying videos.

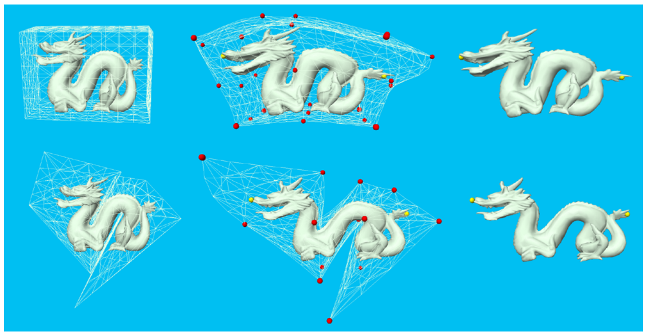

Figure 5: Two deformations of an embedded dragon. Yellow spheres represent constraints and red spheres represent active control points. In the top row the dragon is embedded in a regular grid and in the bottom row the dragon is embedded in a custom control lattice. The left images show the rest state, the center images show the deformed lattice and object, and the right images show only the deformed object. Notice that the bottom example has fewer control vertices but produces a much more realistic deformation that respects the cracks and crevices in the dragon.

## 5.5 Simulating Embedded Objects

Until now we have only considered objects that are exactly parameterized by a cell complex. This is an unfortunate restriction because volumetric parameterization is very difficult; but it is also an unnecessary restriction. Our framework can be generalized to handle situations in which the object is embedded in a control lattice, similar to the free-form deformation (FFD) of MacCracken and Joy [18] combined with the dynamic FFD of Faloutsos et al. [10]. The primary difference between our method and the method of Faloutsos et al. is that they use a diagonal stiffness matrix to describe the dynamics of the system. By following the methodology of the finite element method and Lagrangian dynamics, we can ensure that the motion of the object truly conforms to the shape and constitution of the embedded object. We simply need to make sure that the kinetic and potential energies used in the Euler-Lagrange equation 5 correspond to the embedded object. This is accomplished by computing the integrals in equations (7) and (18) only over the interior of the object. We approximate these integrals by subdividing the control lattice finely and then discarding all tetrahedra that fall completely outside the object. Two examples of deforming an embedded object appear in Figure 5, one using a regular grid and one using a custom control lattice. The example using a custom control lattice produces a much more realistic deformation than the example using a regular grid because the custom lattice is able to avoid incorrectly correlating distant parts of the object.

# 6 Results

In order to test various features of our framework, we implemented a number of interactive dynamic simulations. We show the realtime interaction with the each of these simulations in the accompanying videos. In this section, we describe the setup and the implementation details for each example.

**Sharp Features.** DeRose and Kass [8] added rules for sharp features to the Catmull-Clark subdivision framework. Since the

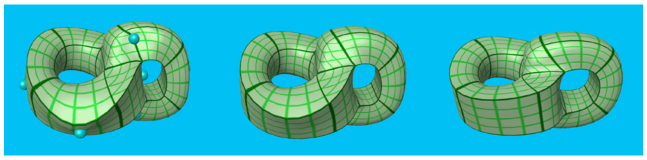

Figure 6: Upon releasing the constraints (represented as blue spheres in the first frame), a chain-like object dynamically oscillates and eventually returns to its rest shape.

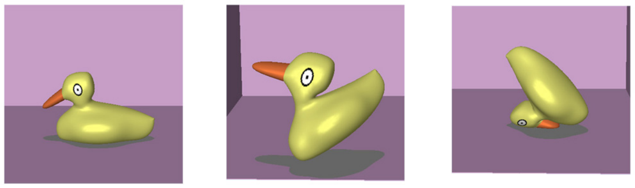

Figure 7: A collection of frames from the dynamic interaction with the duck constrained with the non-penetrating floor and wall constraints.

boundaries of Catmull-Clark volumes are Catmull-Clark surfaces, we can easily include sharp features in our framework. [3] Figure Figure 6 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 7 a duck is being tossed about in a box.

**Varying Material Properties.** Another feature of our system is the ability to vary material properties both spatially and temporally. Material properties are incorporated during the computation of the stiffness and mass matrices. During the quadrature phase, the values for $\nu$, $G$, and $\rho$ 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 associated basis function are factored in when computing the mass and stiffness matrices. As a result, when we use a subset of the basis for simulation, the material properties of the entire object are still being taken into consideration.

As a convenient way to specify material properties over the entire body, we make use of the subdivision basis. We simply specify the material properties at the coarse vertices and use the subdivision rules to generate material property values at each fine vertex, which can then be used when the stiffness and mass matrices are constructed. The object in Figure 8 varies in $G$ along its length. When the middle is grabbed and shaken, one end wobbles like soft rubber while the other remains almost rigid.

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

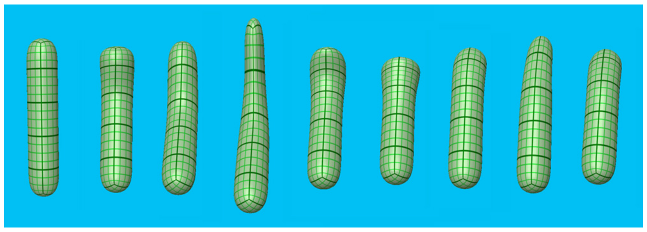

Figure 8: A cucumber-like object a with longitudinally varying shear modulus $G$. The cucumber is being shaken by the firmer bottom, while the top deforms drastically.

We also allow $G$ to be scaled at runtime, which does not require any re-computation because it simply scales the entire stiffness matrix. $G$ is arguably the most intuitive physical property of linear elastic bodies because it corresponds to how easily an object deforms.

For the examples shown, which ran interactively, we typically used a three-level basis. Precomputation required about 10 minutes per model. Simulation times averaged about 10ms per simulation step using only the base mesh, but were longer depending on the extent of adaptation required.

## 7 Conclusion

In this paper we present a framework for the simulation of elastically deformable solids. We represent solid objects and their deformations using a subdivision wavelet basis which provides a speed-accuracy trade-off and support for objects of arbitrary topology. The wavelet representation allows us to adaptively refine the basis, which we do in a way that mitigates popping artifacts. Our framework also supports complex objects embedded in a coarse control lattice, without sacrificing the dynamics of the underlying object. In this framework, we have been able to simulate elastic deformable models of moderate complexity and having spatially varying material properties at interactive rates. Our framework also supports the use of trilinear basis functions, which produce faster simulations, and we can smooth the results while maintaining constraints.

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 sharp *internal* features. 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 more sophisticated wavelet bases could be useful in this framework. We have experimented with using *lifted* wavelets, 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.

Finally, there are a number of theoretical issues regarding the appropriateness of volumetric subdivision schemes for simulation that should be explored.

## A Appendix

### A.1 Derivatives of Elastic Potential

In order to solve the system of equations in (14) we require the gradient and Hessian of the elastic potential given in equation (11):

$$\frac{\partial V}{\partial \mathbf{q}_c} = \begin{array}{l} 2A_1^{ca}\mathbf{q}_a + A_2^{ac}\mathbf{q}_a + B^{ac}\mathbf{q}_a \\ +2\mathbf{q}_d\left(\mathbf{q}_a \cdot C_1^{adc}\right) + \left(\mathbf{q}_a \cdot \mathbf{q}_b\right) C_1^{cab} \\ +\mathbf{q}_a\left(\mathbf{q}_d \cdot C_2^{acd}\right) + \mathbf{q}_a\left(\mathbf{q}_d \cdot C_2^{cad}\right) + \left(\mathbf{q}_a \cdot \mathbf{q}_b\right) C_2^{bac} \\ +\mathbf{q}_d\left(\mathbf{q}_a \cdot \mathbf{q}_b\right) D_1^{abcd} + \mathbf{q}_a\left(\mathbf{q}_d \cdot \mathbf{q}_e\right) D_2^{adce} \end{array} \quad (16)$$

$$\frac{\partial^2 V}{\partial \mathbf{q}_c \partial \mathbf{q}_f} = \begin{array}{l} 2A_1^{cf} + A_2^{fc} + IB^{fc} + 2I\left(\mathbf{q}_a \cdot C_1^{afc}\right) \\ +2\mathbf{q}_d \otimes C_1^{fdc} + 2C_1^{cfb} \otimes \mathbf{q}_b + I\left(\mathbf{q}_d \cdot C_2^{fcd}\right) \\ +\mathbf{q}_a \otimes C_2^{acf} + I\left(\mathbf{q}_d \cdot C_2^{cfd}\right) + \mathbf{q}_a \otimes C_2^{caf} \\ +C_2^{fac} \otimes \mathbf{q}_a + C_2^{bfc} \otimes \mathbf{q}_b + I\left(\mathbf{q}_a \cdot \mathbf{q}_b\right) D_1^{abcf} \\ +2\left(\mathbf{q}_d \otimes \mathbf{q}_a\right) D_1^{afcd} + I\left(\mathbf{q}_d \cdot \mathbf{q}_e\right) D_2^{fdce} \\ +\left(\mathbf{q}_a \otimes \mathbf{q}_d\right) D_2^{adcf} + \left(\mathbf{q}_a \otimes \mathbf{q}_e\right) D_2^{afce} \end{array} \tag{17}$$

where $I$ is a $3 \times 3$ identity matrix and

$$\begin{array}{l} A_1^{ab} = \int_\Omega \frac{4G\nu}{1-2\nu}\left(\frac{\partial\varphi^a}{\partial \mathbf{x}} \otimes \frac{\partial\varphi^b}{\partial \mathbf{x}}\right) d\Omega \\ A_2^{ab} = \int_\Omega 4G\left(\frac{\partial\varphi^a}{\partial \mathbf{x}} \otimes \frac{\partial\varphi^b}{\partial \mathbf{x}}\right) d\Omega \\ B^{ab} = \int_\Omega 4G\left(\frac{\partial\varphi^a}{\partial \mathbf{x}} \cdot \frac{\partial\varphi^b}{\partial \mathbf{x}}\right) d\Omega \\ C_1^{abc} = \int_\Omega \frac{4G\nu}{1-2\nu}\frac{\partial\varphi^a}{\partial \mathbf{x}}\left(\frac{\partial\varphi^b}{\partial \mathbf{x}} \cdot \frac{\partial\varphi^c}{\partial \mathbf{x}}\right) d\Omega \\ C_2^{abc} = \int_\Omega 4G\frac{\partial\varphi^a}{\partial \mathbf{x}}\left(\frac{\partial\varphi^b}{\partial \mathbf{x}} \cdot \frac{\partial\varphi^c}{\partial \mathbf{x}}\right) d\Omega \\ D_1^{abcd} = \int_\Omega \frac{4G\nu}{1-2\nu}\left(\frac{\partial\varphi^a}{\partial \mathbf{x}} \cdot \frac{\partial\varphi^b}{\partial \mathbf{x}}\right)\left(\frac{\partial\varphi^c}{\partial \mathbf{x}} \cdot \frac{\partial\varphi^d}{\partial \mathbf{x}}\right) d\Omega \\ D_2^{abcd} = \int_\Omega 4G\left(\frac{\partial\varphi^a}{\partial \mathbf{x}} \cdot \frac{\partial\varphi^b}{\partial \mathbf{x}}\right)\left(\frac{\partial\varphi^c}{\partial \mathbf{x}} \cdot \frac{\partial\varphi^d}{\partial \mathbf{x}}\right) d\Omega \end{array} \tag{18}$$

Note that $A_i^{ab}$ is a $3 \times 3$ matrix, $C_i^{abc}$ is a 3-dimensional vector, and $B^{ab}$ and $D_i^{abcd}$ are scalar quantities.

## References

[1] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):303–308, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

[3] 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. ISSN 1067-7055.

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

[5] 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.

[6] 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.

[7] Gilles Debunne, Mathieu Desbrun, Alan Barr, and Marie-Paule Cani. Interactive multiresolution animation of deformable models. *Computer Animation and Simulation '99*, September 1999. ISBN 3-211-83392-7. Held in Milano, Italy.

[8] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. *Computer Graphics*, 32(Annual Conference Series):85–94, August 1998.

[9] Mathieu Desbrun, Peter Schröder, and Al Barr. Interactive animation of structured deformable objects. *Graphics Interface '99*, pages 1–8, June 1999. ISBN 1-55860-632-7.

[10] 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.

[11] 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. ISBN 0-89791-736-7.

[12] Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Proceedings of SIGGRAPH 93*, pages 221–230, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California.

[13] 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. Held in Boston, Massachusetts.

[14] Doug L. James and Dinesh K. Pai. Artdefo - accurate real time deformable objects. *Proceedings of SIGGRAPH 99*, pages 65–72, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[15] 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. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[16] P. Krysl, E. Grinspun, and P. Schröder. Natural hierarchical refinement for finite element methods. *draft*, 2001.

[17] 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.

[18] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. *Computer Graphics*, 30(Annual Conference Series):181–188, 1996.

[19] 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.

[20] 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.

[21] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):309–312, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois.

[22] 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.

[23] 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. Held in Boston, Massachusetts.

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

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

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

[27] 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. ISSN 1067-7055.

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

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

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

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

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

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

[34] Henrik Weimer and Joe Warren. Subdivision schemes for fluid flow. *Proceedings of SIGGRAPH 99*, pages 111–120, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

[35] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):243–252, August 1990. ISBN 0-201-50933-4. Held in Dallas, Texas.