*From the proceedings of ACM SIGGRAPH 2002*

# Interactive Skeleton-Driven Dynamic Deformations

Steve Capell Seth Green Brian Curless Tom Duchamp Zoran Popović

University of Washington

## Abstract

This paper presents a framework for the skeleton-driven animation of elastically deformable characters. A character is embedded in a coarse volumetric control lattice, which provides the structure needed to apply the finite element method. To incorporate skeletal controls, we introduce line constraints along the bones of simple skeletons. The bones are made to coincide with edges of the control lattice, which enables us to apply the constraints efficiently using algebraic methods. To accelerate computation, we associate regions of the volumetric mesh with particular bones and perform locally linearized simulations, which are blended at each time step. We define a hierarchical basis on the control lattice, so for detailed interactions the simulation can adapt the level of detail. We demonstrate the ability to animate complex models using simple skeletons and coarse volumetric meshes in a manner that simulates secondary motions at interactive rates.

**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

## 1 Introduction

Physical simulation is central to the process of creating realistic character animations. In the film industry, animators require detailed control of the motion of their characters, but creating physically-based secondary motions is difficult and time consuming to do by hand. Recently, techniques have been developed for automatically simulating these secondary motions. These methods are built atop skin, muscle, and bone models and can generate detailed, dynamic motions. However, constructing these models is time consuming, and the simulations are computationally expensive.

By contrast, in video game or virtual reality applications where interactivity is critical, character animation is built atop much simpler models. The shapes are composed of convenient primitives and are controlled by line segment based skeletons. Deformations of body parts are purely kinematically driven, using, e.g., blended coordinate frames. Incorporating realistic physically-based dynamics using the film industry's approach is currently impractical.



In this paper, we attempt to bring dynamic simulation into the realm of real-time, skeleton-driven animation. The challenge is to find the right combination of physical principles, geometric modeling, computational tools, and simplifying assumptions that yield compelling animations at interactive rates.

Our approach is based on the equations of motion of elastic solids, simulated in a finite element setting. The volumetric finite element mesh need only be specified coarsely, subject to the requirement that it encompass the geometric model on which simulation will be performed. This last requirement is necessary in order to ensure complete integration over the interior of the object. In fact, as long as the interior of the object is well-defined, simulation of its elastic deformation is possible regardless of the the surface representation or complexity.

The volumetric mesh we choose is not restricted to a regular grid; rather, it is comprised of elements such as tetrahedra and hexahedra. This flexibility permits construction of meshes that conform better to the surface of the object, improving simulation quality. In addition, to support adaptive level of detail during simulation, we construct a hierarchical basis, which allows detail to be introduced or removed as needed.

Since our ultimate goal is simulation of skeletally controlled characters, our framework supports line constraints, where lines correspond to bones. In order to incorporate these constraints easily, we require the volumetric mesh to contain edges coincident with the bones. Finally, to achieve interactive rates, we linearize the equations of motion, solve them over volumetric regions associated with each bone, and blend the deformations where regions overlap.

We believe that this work makes a number of contributions. Our crafting of the function space in order to make constraint handling easier is, to our knowledge, novel. We introduce blended local linearization of nonlinear equations, in the context of deformable animated characters. We generalize a method of solving constraints using linear subspace projection. We also introduce a constraint that allows one-dimensional bones to behave as three-dimensional bones. Finally, we believe our most important contribution is putting together a collection of techniques that allows us to interactively animate arbitrary shapes with skeletal controls while generating realistic dynamic deformations.

## 2 Related Work

Probably the most common technique for deforming articulated characters is to define the position of the surface geometry as a function of an underlying skeletal structure or set of control parameters. Recent advances in this area can be found in the work of Lewis et al. [2000], Singh and Kokkevis [2000], and Sloan et al. [2001]. Our work builds on the notion of skeletal control, but within a physically-based framework.

In the late 1980's, Terzopoulos et al. pioneered the field of physically-based deformable models for computer graphics. Using Lagrangian equations of motion and finite differences they simulated elastic [1987] and inelastic [1988] behaviors, combined with a rigid body motion term to compensate for instabilities with stiff bodies [1988].

Much of the research that followed sought to add more sophisticated constraint solvers, accelerate the solutions under a variety of approximations, and add stability to permit larger timesteps. Platt and Barr [1988] improved on existing constraint methods (e.g., the penalty method) by introducing reaction constraint and augmented Lagrangian constraint approaches. Applying these constraints to already complex simulations, however, was not a step toward interactivity.

Pentland and Williams [1989] simplified the problem by solving for the vibrational modes of a body and keeping only the lower frequency modes and obtained realtime simulations of physically plausible deforming bodies. Witkin and Welch [1990] used Lagrangian dynamics to solve for low-order polynomial, global deformations, coupled with constraints enforced through Lagrange multipliers. Baraff and Witkin [1992] later extended this method for better handling of non-penetration constraints. Finally, Metaxas and Terzopoulos [1992] combined Witkin and Welch's global deformation framework with local finite element surface deformations and Lagrange multiplier constraints to animate superquadric surfaces. In each of these examples, while achieving interactive rates, the deformations were substantial approximations to detailed volumetric deformations and were not demonstrated on complex shapes.

To accelerate computations, some hierarchical methods have also been employed. Terzopoulos et al. [1988] use a multigrid solver for a surface-based inelastic simulation. The approach of Metaxas and Terzopoulos [1992] is analogous to a two level simulation that uses global deformations at the coarse level and finite elements for finer surface deformations. More recently, Debunne et al. [1999] built an octtree of particles that interact according to Lamé's equation, resulting in interactive simulations. The particles are simulated using an explicit Euler solver that steps each particle adaptively in time and at differing spatial resolutions. To animate a surface, the particles are linked to each surface point by a weighting scheme. Their approach was recently extended to use unstructured tetrahedral hierarchies [Debunne et al. 2001].

To add stability to computations, implicit solvers have proven to be quite effective. Terzopoulos et al. [1987; 1988] used semi-implicit solvers in their initial work. Baraff and Witkin [1998] used an implicit scheme to permit large timesteps in notoriously unstable cloth simulations. Desbrun et al. [1999], also working with cloth models, showed that the implicit solution method acts as a filter that stabilizes stiff systems. They also add a rotation term to preserve angular momentum and a correction term after each time step to simulate nonlinear elasticity.

Free-form deformation (FFD), introduced by Sederberg and Parry [1986], is also closely related to our work. FFD involves embedding an object in a domain that is more easily parameterized than the object itself. The main advantages of FFD are that arbitrary objects can be easily deformed and the space of deformations can be crafted independently of the representation and resolution of the object. Since its introduction, the flexibility of FFD has been improved by introducing lattices of arbitrary topology [MacCracken and Joy 1996], and dynamic free-form deformation has been introduced to apply FFD to animation [Faloutsos et al. 1997]. Our framework builds on FFD by also embedding the object in a coarse control lattice. But unlike the work of Faloutsos et al., where a diagonal stiffness matrix was used, we use the principles of continuum elasticity to compute the dynamics of the object being deformed.

Another approach to fast, physically-based deformations is to solve quasi-static solutions, i.e., compute the equilibrium state of the system given forces and constraints, and then animate by adjusting the forces and constraints over time. Gourret et al. [1989] explored such a technique for volumetric finite elements, and James and Pai [1999] developed an interactive boundary element solution under the assumption of constant material properties inside the volume. Other quasi-static approaches have also been favored for surgical planning and simulation [Bro-Nielsen and Cotin 1996; Koch et al. 1996; Roth et al. 1998]. Quasi-static solutions, however, are approximations that do not capture the true dynamics of motion.

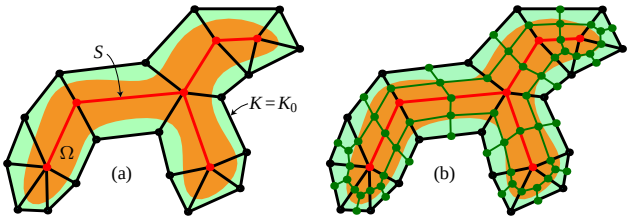


Figure 1: (a) An object $\Omega \subset \mathbb{R}^3$ instrumented with a skeletal complex $S$ and a control lattice $K = K_0$. Functions on $\Omega$ are defined in terms of $K$, which forms a neighborhood of the object. A trilinear basis function is associated with each vertex of $K$ (including the red vertices on the skeleton). (b) The control lattice subdivided once to form $K_1$ (which includes the black, red, and green vertices and edges). Additional basis functions are associated with newly introduced vertices (shown in green).

In some cases, interactivity has been achieved without resorting to significant simplifications in the dynamic model. Recently, Picinbono et al. [2000] described an interactive surgery simulation using a nonlinear finite element method. They were able to achieve interactive rates for a virtual liver composed of about 2000 tetrahedra. However, more complex objects such as the ones used in our work would require many more tetrahedra.

There are also a number of papers that approach the problem of deforming humans and animals using anatomical modeling [Aubel and Thalmann 2000; Wilhelms and Gelder 1997]. They differ from our approach in that we are more concerned with interactivity and the appearance of realism than actual anatomical modeling.

Another possibility for deformation is to perform thin-shell computations. Recent work by Cirak and Ortiz [2001] demonstrates the use of subdivision elements for computing the dynamics of thin shells. However, thin shells are insufficient for modeling the interior of solid objects.

In the next section we introduce the basic mathematical and physical formulation that underlies our framework. We then describe our simulation and control methodology (Section 4), discuss results (Section 5), and conclude (Section 6).

## 3 Formulation

Each object that we wish to animate is represented as a domain $\Omega \subset \mathbb{R}^3$. We make no assumptions about $\Omega$ other than we know its interior. We instrument each object with a skeleton, i.e. an annotated transformation hierarchy suitable for animation. We refer to each transformation in the hierarchy, as well as its associated origin point, as a *joint*. The skeleton defines a graph $S$, whose vertices correspond to joints and whose edges correspond to line segments between two joints that share a parent-child relationship. Joints and bones are located interior to the object, so the graph $S$ is a piecewise linear subset $S \subset \Omega$, that we call a *skeletal complex* (see Figure 1).

Motion of the object is represented by a time dependent function

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

Let

$$\mathbf{p}_S : S \times \mathbb{R} \to \mathbb{R}^3 \tag{2}$$

denote the restriction of the map to $S$. Rigidity of the bones implies that $\mathbf{p}_S$ is an isometry on each edge of $S$. In particular, $\mathbf{p}_S$ is a piecewise linear function on $S$.

Our goal is to solve for the dynamic motion of the object given the motion of the skeleton. Since we model the object as an elastic

body, the function $\mathbf{p}(\mathbf{x}, t)$ is then the solution of a system of partial differential equations, subject to the constraint $\mathbf{p}(\mathbf{x}, t) = \mathbf{p}_S(\mathbf{x}, t)$ for $\mathbf{x} \in S$.

To solve the system numerically, we apply the finite element method (see, e.g., [Prenter 1975]). We separate the map $\mathbf{p}(\mathbf{x}, t)$ into a constant *rest state* $\mathbf{r}(\mathbf{x})$ and a dynamic *displacement* $\mathbf{d}(\mathbf{x}, t)$, each of which is represented as a finite sum. The rest state of the object is given by the identity map $\mathbf{r} : \Omega \to \mathbb{R}^3$, which has the expansion:

$$\mathbf{r}(\mathbf{x}) = \sum_a \mathbf{r}_a \, \phi^a(\mathbf{x}) = \mathbf{r}_a \, \phi^a(\mathbf{x}) = \mathbf{x} \tag{3}$$

where the functions $\phi^a(\mathbf{x})$ are elements of a finite basis $\mathcal{B}$, and $\mathbf{r}_a \in \mathbb{R}^3$. As demonstrated in the above equation, we use the Einstein summation convention throughout this paper: whenever a term contains the same index as both a subscript and a superscript, the term implies a summation over the range of that index. The displacement is expanded similarly:

$$\mathbf{d}(\mathbf{x}, t) = \mathbf{q}_a(t) \, \phi^a(\mathbf{x}) \, . \tag{4}$$

where $\mathbf{q}_a(t) \in \mathbb{R}^3$ are the dynamically evolving coefficients that determine the deformation of the object over time. The state of the system is simply the sum of the rest state and the displacement:

$$\mathbf{p}(\mathbf{x}, t) = (\mathbf{r}_a + \mathbf{q}_a(t))\phi^a(\mathbf{x}) \tag{5}$$

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 the coefficient $\mathbf{q}_a(t)$ in Equation (4), and we model the dynamics of the body as a system of second order ordinary differential equations. The system is obtained by applying the finite element method to the Lagrangian formulation of the equations of elasticity (see, e.g., [Shabana 1998]). In the remainder of the section we describe the basis $\mathcal{B}$ and formulate the finite element problem.

### 3.1 The Hierarchical Basis

In order to allow our simulations to adapt to local conditions, we employ a hierarchical basis. Such bases are well established as useful tools for numerical computation (see, e.g., [Bank 1996]). Our construction mirrors that of what are referred to as *lazy wavelets* in [Stollnitz et al. 1996]. The basis $\mathcal{B}$ is defined in terms of repeated subdivision of a control lattice surrounding the object (see Figure 1). It is desirable that the control lattice conform to the shape of the object while being as coarse as possible. An advantage of using an unstructured lattice instead of a regular grid to define the deformation function space is that the lattice can be tailored to fit the object. More precisely a *control lattice* $K$ is a finite union $K = \cup_i C_i$ of convex cells $C_i$ satisfying the following conditions:

(i) For all $i, j$, $i \neq j$ the intersection $C_{i,j} = C_i \cup C_j$ is either empty or a face, edge, or vertex of both $C_i$ and $C_j$.
(ii) The edges of $S$ are edges of cells of $K$.
(iii) The domain $\Omega$ is contained in the interior of $K$.
(iv) For all $i$, each vertex of $C_i$ has valence 3 (within $C_i$).

Condition (iv) still allows a variety of cell shapes including hexahedra, tetrahedra and triangular prisms.

We now show how to construct a collection of functions $\mathcal{B} = \{\phi^a\}$ on $K$ whose restriction to $\Omega$ is a linearly independent set of continuous functions on $\Omega$. Let $V_0 \subset V_1 \subset V_2 \subset \ldots$ be the nested sequence of function spaces described in appendix A.1. The set $V_J$ consists of the piecewise trilinear functions on the complex $K_J$ obtained from $K$ by $J$ hexahedral subdivisions. For each vertex $a$ of $K$, positioned at $\mathbf{x}_a$, let $\phi^a$ denote the unique function in $V_0$ such that $\phi^a(\mathbf{x}_a) = 1$ and $\phi^a(\mathbf{x}_b) = 0$ for $b \neq a$ a vertex of $K$. Include $\phi^a$ in $\mathcal{B}$ if the restriction of $\phi^a$ to $\Omega$ is non-zero. Proceed inductively as follows. Let $a$ be a vertex of $K_{J+1}$ that is not a vertex of $K_J$ and let $\phi^a$ be the unique function in $V_{J+1}$ such that $\phi^a(a) = 1$ and that vanishes at all other vertices of $K_{J+1}$. Include $\phi^a$ in $\mathcal{B}$ if its restriction to $\Omega$ is non-zero *and* if its restriction to $S$ *is* zero.

Although the elements of $\mathcal{B}$ are defined on all of $K$, we are only interested in their values on $\Omega$; we will, therefore, interpret $\mathcal{B}$ as collection of functions on $\Omega$. One can show that the set $\mathcal{B}$ is linearly independent set of functions, which we call the *hierarchical basis*.

By construction $\phi^a \in \mathcal{B}$ for each joint vertex $a \in S$, and the restriction of $\phi^a$ to $S$ is linear on each bone of $S$. Moreover if $\phi^a \in \mathcal{B}$, for $a$ not a joint vertex, then $\phi^a$ vanishes identically on $S$. Consequently, the function $\mathbf{p}_S(\mathbf{x}, t)$ can be written in the form

$$\mathbf{p}_S(\mathbf{x}, t) = \sum_{a \in S} (\mathbf{r}_a + \mathbf{q}_a(t))\phi^a(\mathbf{x}) \, . \tag{6}$$

and because $\phi^a(a) = 1$, the vector $(\mathbf{r}_a + \mathbf{q}_a(t))$ is the location of the joint vertex $a$ at time $t$.

### 3.2 Equations of Motion

By virtue of Equation (5), we can express the kinetic energy $T$ and elastic potential energy $V$ as functions of $\dot{\mathbf{q}}$ and $\mathbf{q}$, respectively, where $\dot{\mathbf{q}}$ denotes the time derivative of $\mathbf{q}$. The equations of motion are then the *Euler-Lagrange* equations

$$\frac{d}{dt}\left(\frac{\partial T(\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}}\right) + \frac{\partial V(\mathbf{q})}{\partial \mathbf{q}} + \mathbf{Q}^{ext} - \mu\dot{\mathbf{q}} = 0 \tag{7}$$

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* arising from external body forces, such as gravity. The last term is a generalized dissipative force, added to simulate the effect of friction. We will now derive each of the first three terms of Equation 7, ultimately yielding a system of ODEs to be solved in generalized coordinates.

The kinetic energy of a moving body is a generalization of the familiar $\frac{1}{2}mv^2$:

$$T = \frac{1}{2}\int_\Omega \rho(x)\, \dot{\mathbf{p}} \cdot \dot{\mathbf{p}} \, d\Omega = \frac{1}{2} M^{ab} \, \dot{\mathbf{q}}_a \cdot \dot{\mathbf{q}}_b \tag{8}$$

where $\rho(x)$ is the mass density of the body, and $\mathbf{M}^{ab} = \int_\Omega \rho \, \phi^a \phi^b \, d\Omega$. Equation (8) yields the formula

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\mathbf{q}}}\right) = \mathbf{M}\,\ddot{\mathbf{q}} \, . \tag{9}$$

The matrix $\mathbf{M}$ composed of the elements $\mathbf{M}^{ab}$ is called the *mass matrix*. We discuss its computation in Section 3.3.

The elastic potential energy of a body captures the amount of work required to deform the body from the rest state into the current configuration. It is expressed in terms of the *strain tensor* and *stress tensor*. Strain is the degree of metric distortion of the body. A standard measure of strain is Green's strain tensor:

$$e_{ij} = \frac{\partial d^i}{\partial x^j} + \frac{\partial d^j}{\partial x^i} + \delta_{kl}\frac{\partial d^k}{\partial x^i}\frac{\partial d^l}{\partial x^j} \tag{10}$$

The diagonal terms of the strain tensor represent normal deformations while those off the diagonal capture shearing. Forces acting on the interior of a continuum appear in the form of the *stress tensor*, which is defined in terms of strain:

$$\tau_{ij} = 2G\left\{\frac{\nu}{1-2\nu} tr(e)\delta_{ij} + e_{ij}\right\} \tag{11}$$

where $tr(e) = \delta^{ij} e_{ij}$. The constant $G$, called the *shear modulus* or *modulus of rigidity*, determines how hard the body resists deformation. The coefficient $\nu$, called *Poisson's ratio*, determines the extent to which strains in one direction are related to those perpendicular to it. This gives a measure of the degree to which the body preserves volume. The elastic potential energy $V(\mathbf{q})$, which is analogous to the familiar definition of work as force times distance, is given by the formula

$$V = G \int_\Omega \left\{ \frac{\nu}{1-2\nu} tr^2(e) + \delta^{ij}\delta^{kl} e_{ik} e_{jl} \right\} d\Omega \qquad (12)$$

By combining Equations 4, 10, and 12 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. Details are described in appendix A.2. The matrix $\mathbf{S} = \frac{\partial^2 V}{\partial \mathbf{q} \partial \mathbf{q}}$ is referred to as the *stiffness matrix*.

To add realism, we include the force of gravity in our formulation. 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

$$V_g = \int_\Omega \rho\, \mathbf{g} \cdot \mathbf{p} = \int_\Omega \rho \phi^a \mathbf{g} \cdot \mathbf{q}_a \,. \qquad (13)$$

The *generalized gravitational force* is the gradient

$$\mathbf{Q}_a^g = \frac{\partial V_g}{\partial \mathbf{q}_a} = \left( \int_\Omega \rho \phi^a \right) \mathbf{g} \qquad (14)$$

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.

### 3.3 Numerical Integration

In order to compute the gravity terms and the mass and stiffness matrices we precompute the integrals in equations (8), (14), and (29). The integration is done numerically using the following steps:

1. Subdivide $K$ to the desired level for numerical integration.
2. Compute the values of the basis functions at each vertex.
3. Tetrahedralize the domain. After subdividing once, the domain is composed of only hexahedral cells. We then divide each of these cells into tetrahedra 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. If all four vertices of a tetrahedron fall outside the surface of the object, its contribution to the integrals is neglected.

With the integrals computed, equation (7) can now be solved using a nonlinear Newton-Raphson solver.

## 4 Skeletal Simulation

The fully nonlinear elastic formulation described in the previous section is computationally expensive, and does not take into consideration the skeleton. In this section we introduce a set of techniques, tailored for fast skeleton-driven animation, that approximate the nonlinear dynamics.

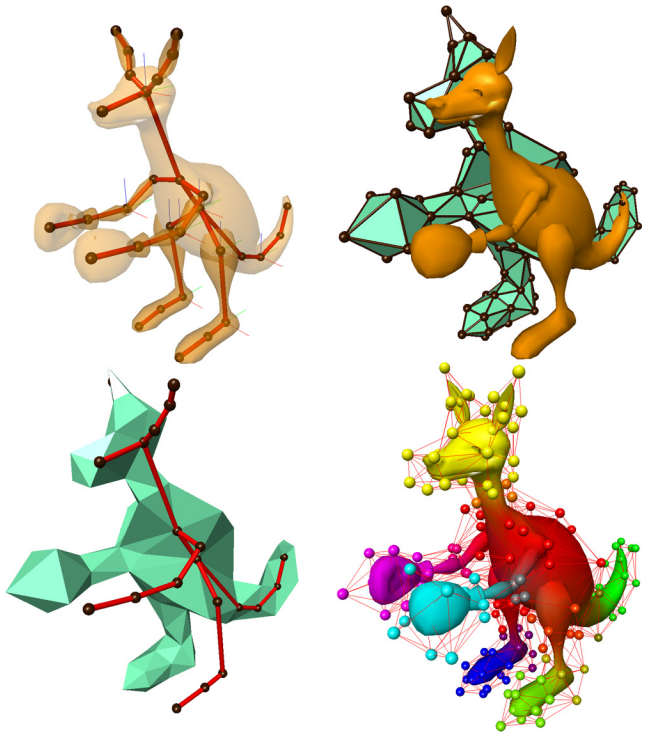

Figure 2: The upper left image shows an input model instrumented with a skeleton and local coordinate systems. The upper right image shows the model embedded in (half of) a control lattice. The lower left image shows how the skeleton coincides with edges and vertices of the control lattice. The lower right image shows the entire control lattice, as well as the division of the object into regions for local linearization. Each region is associated with one of the local coordinate systems in the upper left image. Note the color blending where regions overlap.

### 4.1 Instrumentation

Prior to simulation, a model must be instrumented with a skeleton and control lattice. Although recent work by Teichmann and Teller addresses automated skeleton construction [1998], we currently let the animator specify the skeleton in order to achieve the desired level of control. We have implemented a simple system that allows a skeleton to be constructed manually in just a few minutes. The user creates a joint by clicking on the object with the mouse. If the ray through the mouse point (from the camera projection center) intersects the object at least twice, a joint is placed midway between the first two intersections. This positioning scheme produces joints that are centrally located inside the object. Two joints can be selected to define a bone, and with the selection of a root joint, a transformation hierarchy can be created automatically.

We currently use a constructive procedure that allows the user to build the control lattice interactively by adding cells incrementally and repositioning the control vertices as needed. Several hours are required for an experienced user to create a moderately complex control lattice. The abundance of volumetric meshing schemes suggests that automatic creation of the control lattice is possible, and we hope to address this problem in the future. Figure 2 shows the skeleton and control mesh for a kangaroo model.

### 4.2 Solving the System

Due to the computational expense of solving the full nonlinear equations of elasticity, we seek simplifications that make the equations easier to solve. One possibility is to linearize the equations of motion at the beginning of each timestep as was done by Baraff

and Witkin in their work on cloth simulation [1998]. In our experience, simulations using this method are essentially indistinguishable from results obtained using a nonlinear implicit method to solve the system, as long as the timestep is not so large as to allow radical shape change during a single step. After applying their implicit solver to our formulation, the resulting equations are:

$$\Delta\mathbf{q} = h(\dot{\mathbf{q}} + \Delta\mathbf{v}) \quad (15)$$

$$(\mathbf{M} - h\mu\mathbf{I} + h^2\mathbf{S})\Delta\mathbf{v} = h\left(\mu\dot{\mathbf{q}} - \frac{\partial V}{\partial \mathbf{q}} - \mathbf{Q}^{ext} - h\mathbf{S}\dot{\mathbf{q}}\right) \quad (16)$$

where $h$ is the timestep, $\mu$ is the damping coefficient, $\mathbf{I}$ is the identity matrix, $\Delta\mathbf{v}$ is the change in the velocity $\dot{\mathbf{q}}$ during the timestep, $\Delta\mathbf{q}$ is the change in $\mathbf{q}$ during the timestep, $\mathbf{M}$ is the mass matrix, and $\mathbf{S}$ is the stiffness matrix. All quantities are evaluated at the beginning of the timestep. Equation (16) is a sparse linear system that can be solved for $\Delta\mathbf{v}$ using a Conjugate Gradients (CG) solver. Then $\Delta\mathbf{v}$ is substituted into equation (15) to obtain $\Delta\mathbf{q}$.

### 4.3 Bone Constraints

In our framework the skeleton is controlled directly by keyframe data or some other source external to the dynamic simulation. From the viewpoint of the simulation, the skeleton is simply a complicated constraint. Because we have restricted the bones to lie along edges in the control lattice, and the basis is interpolating, it is especially easy to handle the bone constraints algebraically. Each control point that lies on a bone corresponds to a component of $\Delta\mathbf{v}$ that is known *a priori*, rather than having to be computed. Simplifying equation (16) to the form $\mathbf{A}\Delta\mathbf{v} = \mathbf{b}$, we can sort the variables into known ($\Delta\mathbf{v}_k$ and $\mathbf{b}_k$) and unknown quantities ($\Delta\mathbf{v}_u$ and $\mathbf{b}_u$) and form the following system:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{v}_k \\ \Delta\mathbf{v}_u \end{pmatrix} = \begin{pmatrix} \mathbf{b}_u \\ \mathbf{b}_k \end{pmatrix} \quad (17)$$

The reason that some components of the vector $\mathbf{b}$ are now unknown arises from the fact that the external forces required to enforce the bone constraints are unknown, and they appear on the r.h.s. of equation (16). In order to solve for $\Delta\mathbf{v}_u$ we simply solve the system:

$$\mathbf{A}_{22}\Delta\mathbf{v}_u = \mathbf{b}_k - \mathbf{A}_{21}\Delta\mathbf{v}_k \quad (18)$$

The advantage of this approach is that adding skeletal constraints actually reduces the computational cost by shrinking the system that must be solved.

### 4.4 Linear Subspace Constraints

Because we would like our objects to interact with other objects, position constraints are also important. The framework of Baraff and Witkin [1998] provides an elegant solution for particle systems. During each internal step of a CG solver, they project out certain components of $\Delta\mathbf{v}$ corresponding to constrained particles. Here we show that this technique can be extended to include position constraints at any point in a continuous body. Position constraints in our framework are of the form:

$$\mathbf{d}_c(t) = \mathbf{q}_a\phi^a(\mathbf{x}_c) \quad (19)$$

which simply says that the displacement at $\mathbf{x}_c$ conforms to some known function $\mathbf{d}_c$. Evaluating equation (15) at $\mathbf{x}_c$ results in:

$$\Delta\mathbf{v}_a\phi^a(\mathbf{x}_c) = \frac{\mathbf{d}_c(t+h) - \mathbf{d}_c(t)}{h} - \dot{\mathbf{q}}_a\phi^a(\mathbf{x}_c) \quad (20)$$

The r.h.s. of the above equation is simply a constant $\mathbf{a}_c$ that can be computed at the beginning of each timestep. If we accumulate the $x$, $y$, and $z$ components of the 3-vectors $\mathbf{a}_c$ into the $n$-vectors $\mathbf{a}^\alpha$, where $\alpha \in \{x, y, z\}$, define the matrix $\mathbf{C}_{ac} = \phi^a(\mathbf{x}_c)$, and separate $\Delta\mathbf{v}$ into its $x$, $y$, and $z$ components $\Delta\mathbf{v}^\alpha$, equation (20) becomes:

$$\mathbf{C}^T\Delta\mathbf{v}^\alpha = \mathbf{a}^\alpha, \ \alpha \in \{x, y, z\} \quad (21)$$

So each constraint requires that $\Delta\mathbf{v}$ be constant along three particular directions in $\mathbb{R}^{3n}$. Maintaining the constraints involves the following steps:

1. At the beginning of each timestep, $\Delta\mathbf{v}$ is initialized so that equation (21) holds. This is accomplished by computing the QR-decomposition of $\mathbf{C}$ and transforming equation (21) into $\mathbf{R}^T\mathbf{b}^\alpha = \mathbf{a}^\alpha$, $\Delta\mathbf{v}^\alpha = \mathbf{Q}\mathbf{b}^\alpha$, from which $\Delta\mathbf{v}$ can be easily computed. Although QR-decomposition of an $n \times m$ matrix requires $O(nm^2)$ time, in our case the number of constraints $m$ is typically small, so the computational cost is low.
2. Each column $\mathbf{c}$ of $\mathbf{C}$ has an associated *projection matrix* $\mathbf{P} = \mathbf{I} - \mathbf{c}\mathbf{c}^T/\mathbf{c}^T\mathbf{c}$, which, when applied to a vector, eliminates the component in the direction of $\mathbf{c}$. These projectors are applied during CG such that incremental updates to $\Delta\mathbf{v}$ are orthogonal to the vectors $\mathbf{c}$, ensuring that equation (21) remains true (for details see [Baraff and Witkin 1998]).

In our current framework, conflicting constraints can be detected during QR-decomposition and removed. In the future we hope to augment this method to solve over-constrained systems more elegantly, as was done for FFD by Hsu et al. [1992].

### 4.5 Blended Local Linearization

A major bottleneck in our system is the computation of the stiffness matrix at the beginning of each timestep (the elastic potential is a quartic function of $\mathbf{q}$). A well-known simplification is to linearize the strain tensor by dropping the last term in equation (10), which results in a quadratic elastic potential and thus a constant stiffness matrix (which is composed of the first three addends in equation (28)). As compared to other simplifications such as using a mass-spring-based elastic potential, linearization of strain has the advantage that it is a very good approximation, but only when the deformation is small; for large deformations, severe distortions occur.

A notable case for the linear strain model is when the object undergoes a large rigid rotation, coupled with a small deformation. While the elastic potential based on nonlinear strain does not penalize rotations, the linear strain model does, while failing to penalize certain shearing deformations. Terzopoulos et al. [1988] addressed this case by modeling the deformation relative to a frame of reference that follows the gross motion of the object. Since the relative deformation is assumed to be small, the linearized strain is a reasonable approximation. This approach is common practice in the engineering literature, such as in the textbook of Shabana [1998], in which multibody systems composed of interconnected parts are considered. In such systems, the deformation of each part can be measured from a local reference configuration that factors in the rotation of the part. As long as the deformation of each part is small relative to its rotated reference configuration, the linear strain model is a good approximation.

To apply these ideas to articulated characters, we first recognize that the soft tissues of vertebrates do not typically undergo large deformations *relative to nearby bones*. Based on this assertion, our approach is to divide the object into regions, each of which can be simulated using the linear strain model.

The user divides the object into regions by assigning weights to the control vertices, forming a partition of unity over the object. A piece of the object can belong to a single region or can be divided fractionally among several regions. We encode the weights

for region $i$ in a diagonal square matrix $W^i$, where $W^i_{aa}$ is the weight associated with vertex $a$ in region $i$. The lower right image in Figure 2 shows a partitioned object, colored according to the region assignments. Our current system requires that the user select individual weights for each control vertex, but a more intuitive painting interface would be straightforward to implement. It would also be helpful to automate the task of region assignment (recent work by Li et al. [2001] may be adaptable to our problem domain).

From the region assignments we form a cell complex $K^i$ corresponding to region $i$:

$$K^i = \{C \in K : \forall\, v_a \in v(C),\ W^i_{aa} > 0\} \tag{22}$$

where $v(C)$ is the set of control vertices on cell $C$. Each region has an associated function space:

$$\mathcal{B}^i = \{\, \phi^a|_{K^i} : \phi^a \in \mathcal{B},\ \phi^a|_{K^i} \not\equiv 0\} \tag{23}$$

where $\phi^a|_{K^i}$ denotes the restriction of $\phi^a$ to $K^i$. We define a rectangular matrix $\mathbf{Q}^i$ to select the basis functions that have nonzero restrictions to region $i$. The element $\mathbf{Q}^i_{ab} = 1$ if and only if $\phi^a \in \mathcal{B}$ corresponds to $\phi^b \in \mathcal{B}^i$. The pseudocode for taking a single simulation step is:

```
   foreach region i do
1  |  [r^i, q^i, q̇^i] ⇐ [Q^i r, Q^i q, Q^i q̇]
   |  foreach a do
2  |  |  q^i_a := q^i_a − T^i(r^i_a) + r^i_a
   |  end
3  |  Construct A^i and b^i from equation (18)
4  |  Solve A^i Δv^i = b^i
   end
5  Δv ⇐ Σ_i W^i Q^{iT} Δv^i
6  q̇ ⇐ q̇ + Δv
7  q ⇐ q + h q̇
```

Line 1 extracts the regional variables from the global system. Line 2 converts $\mathbf{q}^i$ so that it corresponds not to displacement from the rest state, but to displacement from the rest state transformed according to the transformation of the bone coordinate system. The homogeneous transformation $\mathbf{T}^i$, extracted from the current configuration of the skeleton, represents the transformation of the bone from its rest position to its current position. But it is not enough to simply transform $\mathbf{q}^i$, because the transformation itself must be subtracted from $\mathbf{q}^i$. A displacement field $\mathbf{d_T}$ that transforms the object $\mathbf{x}$ according to the transformation $\mathbf{T}(\mathbf{x})$, has the following form:

$$\mathbf{d_T} + \mathbf{x} = \mathbf{T}(\mathbf{x}) \tag{24}$$

It is from the above expression that line 2 is derived. Line 3 builds the linear system required to solve for the local equations of motion, including the extraction of bone constraints, and line 4 solves the linear system using CG. Line 5 merges the solutions from each region, each weighted according to the user-assigned weights in $\mathbf{W}^i$. Finally, the state of the global system is updated in lines 6 and 7.

### 4.6 Twist Constraint

In natural creatures with three-dimensional bones, the flesh cannot twist (i.e. rotate) around the axis of the bone without causing the flesh to deform. Such deformations are resisted by emergent elastic forces, so the twisting is limited. But flesh can rotate about a line constraint without deforming. To avoid such unnaturally free movement, we introduce a soft constraint to penalize all displacement (not just deformation) within a fixed radius of the bones. We denote this region $\Omega_\beta \subset \Omega$. The following potential describes the constraint:

$$U = \frac{1}{2}\int_{\Omega_\beta} \mathbf{d}\cdot\mathbf{d}\, d\Omega \tag{25}$$

The above potential is quadratic, so its Hessian is simply a constant that can be added to the stiffness matrix:

$$\frac{\partial^2 U}{\partial \mathbf{q}_a \partial \mathbf{q}_b} = \mathbf{I}\int_{\Omega_\beta} \phi^a \phi^b \tag{26}$$

where $\mathbf{I}$ is a $3 \times 3$ identity matrix. The above constraint must be computed relative to the rigidly transformed bone, which fits well into our local computation framework.

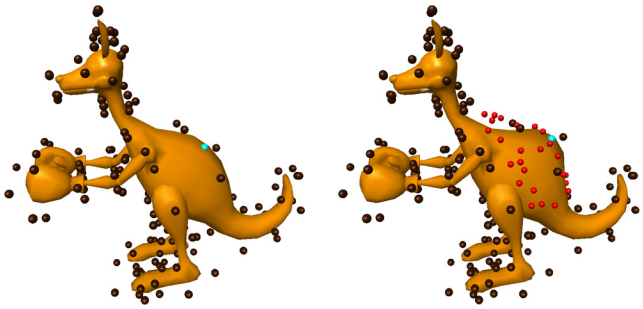

Figure 3: The left image shows the kangaroo at rest. Brown spheres represent active basis functions. The cyan sphere represents a position constraint. On the right, the position constraint has been moved causing adaptation of the basis. The red spheres in the right image represent newly introduced detail coefficients.

### 4.7 Adaptation

Because we use a hierarchical basis, our simulator can add detail where needed. We apply the simple heuristic that detail is more helpful where there are large deformations (similar to, e.g., [Debunne et al. 2001]). If the object is sufficiently deformed over the support of a particular basis function, then all of the basis functions in the next finer level with support overlapping the area of high distortion are introduced into the simulation. Likewise, basis functions are removed when there is little deformation in their support. Each level of the basis has an associated threshold for determining when to refine and another for determining when to coarsen. As noted in [Debunne et al. 2001], a lower threshold is required for coarsening than refining in order to prevent the simulation from oscillating between levels of resolution.

Regardless of the criteria employed, adapting the basis is straightforward in our framework. Most of this simplicity comes from refining the basis, not the geometry, as was done by Gortler and Cohen [1995] and recently generalized by Grinspun et al. [2002]. For some fixed number of basis levels we precompute the mass and stiffness matrices and store them in a sparse data structure. Adapting the basis simply corresponds to extracting and relinquishing certain components from these matrices, which can be done very quickly. The resultant subsets of the basis are linearly independent regardless of which basis functions we choose. Figure 3 shows adaptation of the kangaroo model. For more details regarding our adaptation methodology see [Capell et al. 2002].

## 5 Results

The accompanying video shows the results of applying our framework to two triangle meshes that we acquired from the Internet.

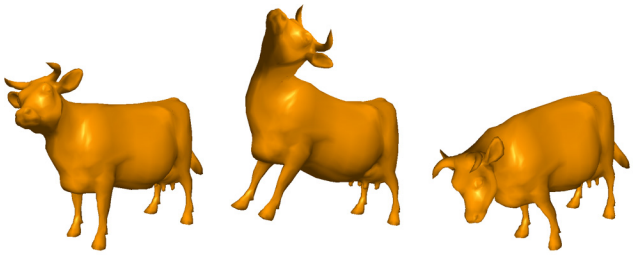

Figure 4: Frames from an interactive animation. There is no noticeable warping due to strain linearization, and the different materials (e.g., ears, horns) behave distinctly.

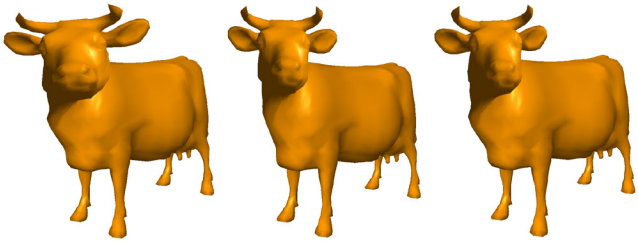

Figure 5: On the left is the global linear solution, which shows significant warping when the cow turns its head to one side. In the center is the fully nonlinear solution. On the right is the blended local linear solution, which shows no noticeable warping of the head. A slight protrusion can be seen in the neck of the right image due to region blending.

The control mesh for the kangaroo model has 448 cells and 177 vertices; the cow control mesh has 572 cells and 214 vertices. On a 1 Ghz PC, both the cow and kangaroo animated at about 100 Hz using only the coarse basis functions, which is clearly within range for interactive applications (with adaptation, simulation time varies depending on the degree of adaptation required). Figure 4 shows frames of an animation of the cow model (using the coarse basis), which demonstrates the ability of our system to handle variable material properties; the ears flop around realistically while the horns stay rigid. This feature is possible to do interactively because the control mesh can be carefully crafted to respect material boundaries, and because our computation of the stiffness matrix takes variable material properties into account.

For our datasets, the blended local linear and global linear solutions required about the same amount of computation time. Yet the blended local linear solution produced much more pleasing results, as demonstrated in Figure 5. The blended local linear solution looks similar to the fully nonlinear solution, while the global linear solution is badly warped.

# 6 Conclusion

We have introduced a method for interactive simulation of deformable bodies controlled by an underlying skeleton. By choosing a volumetric mesh that aligns with the bones, we are able to meet the bone constraints rapidly. We extend a fast constraint solver that works directly within an iterative solver. We also introduce a twist constraint that mimics the effects of three-dimensional bones when only one-dimensional bones are being modeled. Our method performs with the speed of simple linear-strain models of elasticity, but does not suffer from distortions arising from global linearity.

There are many avenues for future work. We would like to automatically generate skeletons and especially control lattices, the latter being the most labor intensive aspect of our framework. Our assumptions about small deformations break down near the joints. It may be possible to address this problem by using nonlinear elasticity near the joints. The deformations near joints might also be improved by specifically tailoring adaptation to the problem. Finally, it would be convenient to include dynamic, not just fully constrained, bones.

**Acknowledgments** The authors would like thank Chris Twigg, Mira Dontcheva and Samantha Michel for their instrumental work in creating and converting Maya skeletal animations, and Shawn Bonham and Sean Smith for additional help. This work was supported by the Animation Research Labs, Microsoft Research, NSF grants DMS-9803226 and CCR-0092970, and an Intel equipment donation.

# A Appendix

## A.1 Review of Trilinear Functions

A *trilinear function* on the standard unit cube $C^3 = \{\mathbf{x} = (x,y,z) : 0 \le x,y,z \le 1\}$ is a function of the form

$$f(x,y,z) = a_0 + a_1x + a_2y + a_3z + a_4xy + a_5xz + a_6yz + a_7xyz.$$

The function $f$ is determined by its values at the vertices of $C^3$: let $\hat{\phi}(s)$ denote the *hat function*

$$\hat{\phi}(s) = \begin{cases} 1 - |s| & \text{for } |s| \le 1 \\ 0 & \text{for } |s| > 1. \end{cases}.$$

and let $\phi_0(x,y,z) = \hat{\phi}(x)\hat{\phi}(y)\hat{\phi}(z)$. Then

$$f(\mathbf{x}) = \sum_{0 \le i,j,k \le 1} f_{i,j,k}\phi_0(x-i, y-j, z-k)$$

where $f_{i,j,k} = f(i,j,k)$. It is easy to check that trilinear functions satisfy the following interpolation or *hexahedral subdivision* rules:

(i) The value of $f$ at the midpoint of an edge of $C^3$ is the average of its values at the endpoints of the edge.
(ii) The value of $f$ at the centroid of a face of $C^3$ is the average of its values at the corners of the face.
(iii) The value of $f$ at the centroid of $C^3$ is the average of its values at the eight vertices of $C^3$.

If we subdivide the unit cube into 8 sub-cubes in the standard way, we can use these subdivision rules to determine the value of $f$ at the vertices of each sub-cube. Repeatedly subdividing and applying the subdivision rules yields the value of $f$ at each diadic point $(i/2^J, j/2^J, k/2^J)$ of $C^3$. Because the diadic points are dense in $C^3$, the subdivision rules completely determine $f$ from its values at the vertices of $C^3$. More generally, starting with values of a function at the vertices of the standard cubic tiling of $\mathbb{R}^3$ and applying the subdivision rules to each cubic cell determines a piecewise trilinear function on $\mathbb{R}^3$.

We can generalize this construction to define *piecewise trilinear* functions on any control lattice in which the vertices of each 3-cell of $K$ have valence 3. Starting with the values of $f$ at the vertices of $K$, we infer its values at the centroid of every edge, face and 3-cell of $K$. This gives values of $f$ at every vertex of the refined complex $K_1$ obtained by subdivision (see [MacCracken and Joy 1996] for details). Because the vertices of each 3-cell of $K$ have valence 3, the subdivided complex $K_1$ has only hexahedral cells, so after one subdivision, the subdivision process behaves just as for cubes in $\mathbb{R}^3$.

There is a corresponding nested sequence of function spaces

$$V_0 \subset V_1 \subset V_2 \subset \ldots$$

defined on $K$. To define $V_J$, subdivide $J$-times to obtain the complex $K_J$ and specify values at each vertex of $K_J$. The subdivision rules then determine a function on all of $K$. Thus, each function in $V_J$, for $J = 0, 1, 2 \ldots$, is determined by its values at the vertices of $K$.

### A.2 Derivatives of Elastic Potential

The gradient and Hessian of $V$ from equation (12) are:

$$\frac{\partial V}{\partial \mathbf{q}_c} = \begin{array}{l} 2A_1^{ca}\mathbf{q}_a + A_2^{ac}\mathbf{q}_a + B^{ac}\mathbf{q}_a \\ +2\mathbf{q}_d\left(\mathbf{q}_a \cdot C_1^{adc}\right) + (\mathbf{q}_a \cdot \mathbf{q}_b)\, C_1^{cab} \\ +\mathbf{q}_a\left(\mathbf{q}_d \cdot C_2^{acd}\right) + \mathbf{q}_a\left(\mathbf{q}_d \cdot C_2^{cad}\right) + (\mathbf{q}_a \cdot \mathbf{q}_b)\, C_2^{bac} \\ +\mathbf{q}_d\,(\mathbf{q}_a \cdot \mathbf{q}_b)\, D_1^{abcd} + \mathbf{q}_a\,(\mathbf{q}_d \cdot \mathbf{q}_e)\, D_2^{adce} \end{array} \tag{27}$$

$$\frac{\partial^2 V}{\partial \mathbf{q}_c \partial \mathbf{q}_f} = \begin{array}{l} 2A_1^{cf} + A_2^{fc} + IB^{fc} + 2I\left(\mathbf{q}_a \cdot C_1^{afc}\right) \\ +2\mathbf{q}_d \otimes C_1^{fdc} + 2C_1^{cfb} \otimes \mathbf{q}_b + I\left(\mathbf{q}_d \cdot C_2^{fcd}\right) \\ +\mathbf{q}_a \otimes C_2^{acf} + I\left(\mathbf{q}_d \cdot C_2^{cfd}\right) + \mathbf{q}_a \otimes C_2^{caf} \\ +C_2^{fac} \otimes \mathbf{q}_a + C_2^{bfc} \otimes \mathbf{q}_b + I\,(\mathbf{q}_a \cdot \mathbf{q}_b)\, D_1^{abcf} \\ +2\,(\mathbf{q}_d \otimes \mathbf{q}_a)\, D_1^{afcd} + I\,(\mathbf{q}_d \cdot \mathbf{q}_e)\, D_2^{fdce} \\ +(\mathbf{q}_a \otimes \mathbf{q}_d)\, D_2^{adcf} + (\mathbf{q}_a \otimes \mathbf{q}_e)\, D_2^{afce} \end{array} \tag{28}$$

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

$$\begin{aligned} A_1^{ab} &= \textstyle\int_\Omega \frac{4G\nu}{1-2\nu}\left(\frac{\partial \phi^a}{\partial \mathbf{x}} \otimes \frac{\partial \phi^b}{\partial \mathbf{x}}\right) d\Omega \\ A_2^{ab} &= \textstyle\int_\Omega 4G\left(\frac{\partial \phi^a}{\partial \mathbf{x}} \otimes \frac{\partial \phi^b}{\partial \mathbf{x}}\right) d\Omega \\ B^{ab} &= \textstyle\int_\Omega 4G\left(\frac{\partial \phi^a}{\partial \mathbf{x}} \cdot \frac{\partial \phi^b}{\partial \mathbf{x}}\right) d\Omega \\ C_1^{abc} &= \textstyle\int_\Omega \frac{4G\nu}{1-2\nu}\frac{\partial \phi^a}{\partial \mathbf{x}}\left(\frac{\partial \phi^b}{\partial \mathbf{x}} \cdot \frac{\partial \phi^c}{\partial \mathbf{x}}\right) d\Omega \\ C_2^{abc} &= \textstyle\int_\Omega 4G\frac{\partial \phi^a}{\partial \mathbf{x}}\left(\frac{\partial \phi^b}{\partial \mathbf{x}} \cdot \frac{\partial \phi^c}{\partial \mathbf{x}}\right) d\Omega \\ D_1^{abcd} &= \textstyle\int_\Omega \frac{4G\nu}{1-2\nu}\left(\frac{\partial \phi^a}{\partial \mathbf{x}} \cdot \frac{\partial \phi^b}{\partial \mathbf{x}}\right)\left(\frac{\partial \phi^c}{\partial \mathbf{x}} \cdot \frac{\partial \phi^d}{\partial \mathbf{x}}\right) d\Omega \\ D_2^{abcd} &= \textstyle\int_\Omega 4G\left(\frac{\partial \phi^a}{\partial \mathbf{x}} \cdot \frac{\partial \phi^b}{\partial \mathbf{x}}\right)\left(\frac{\partial \phi^c}{\partial \mathbf{x}} \cdot \frac{\partial \phi^d}{\partial \mathbf{x}}\right) d\Omega \end{aligned} \tag{29}$$

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

## References


AUBEL, A., AND THALMANN, D. 2000. Realistic deformation of human body shapes. In *Proceedings of Computer Animation and Simulation 2000*, 125–135.

BANK, R. E. 1996. *Hierarchical bases and the finite element method*, vol. 5 of *Acta Numerica*. Cambridge University Press, Cambridge, 1–43.

BARAFF, D., AND WITKIN, A. 1992. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2, 303–308.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, 43–54.

BRO-NIELSEN, M., AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum (Proceedings of Eurographics '96) 15*, 3, 57–66.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. *University of Washington, Department of Computer Science and Engineering, Technical Report 02-04-02*.

CIRAK, F., AND ORTIZ, M. 2001. Fully $c^1$-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering 51*, 7 (July), 813–833.

DEBUNNE, G., DESBRUN, M., BARR, A., AND CANI, M.-P. 1999. Interactive multiresolution animation of deformable models. *Eurographics Workshop on Animation and Simulation*.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of SIGGRAPH 2001*, 31–36.

DESBRUN, M., SCHRÖDER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. *Graphics Interface '99* (June), 1–8.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 1997. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics 3*, 3 (July–Sept.), 201–214.

GORTLER, S. J., AND COHEN, M. F. 1995. Hierarchical and variational geometric modeling with wavelets. *Symposium on Interactive 3D Graphics*, 35–42.

GOURRET, J.-P., THALMANN, N. M., AND THALMANN, D. 1989. Simulation of object and human skin deformations in a grasping task. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3 (July), 21–30.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. Charms: A simple framework for adaptive simulation. *To appear in the Proceedings of SIGGRAPH 2002*.

HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July), 177–184.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo - accurate real time deformable objects. *Proceedings of SIGGRAPH 99* (August), 65–72.

KOCH, R. M., GROSS, M. H., CARLS, F. R., VON BÜREN, D. F., FANKHAUSER, G., AND PARISH, Y. 1996. Simulating facial surgery using finite element methods. *Proceedings of SIGGRAPH 96* (August), 421–428.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of SIGGRAPH 2000*, 165–172.

LI, X., WOON, T. W., TAN, T. S., AND HUANG, Z. 2001. Decomposing polygon meshes for interactive applications. In *ACM Symposium on Interactive 3D Graphics*, 35–42.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. *Computer Graphics (Proceedings of SIGGRAPH 96) 30*, 181–188.

METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July), 309–312.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proceedings of SIGGRAPH 89) 23*, 3 (July), 215–222.

PICINBONO, G., DELINGETTE, H., AND AYACHE, N. 2000. 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*, 643–652.

PLATT, J. C., AND BARR, A. H. 1988. Constraint methods for flexible models. *Computer Graphics (Proceedings of SIGGRAPH 88) 22*, 4 (August), 279–288.

PRENTER, P. M. 1975. *Splines and Variational Methods*. John Wiley and Sons.

ROTH, S. H. M., GROSS, M. H., TURELLO, S., AND CARLS, F. R. 1998. A bernstein-bézier based approach to soft tissue simulation. *Computer Graphics Forum 17*, 3, 285–294.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. *Computer Graphics (Proceedings of SIGGRAPH 86) 20*, 4 (Aug.), 151–160.

SHABANA, A. 1998. *Dynamics of Multibody Systems*. Cambridge University Press.

SINGH, K., AND KOKKEVIS, E. 2000. Skinning characters using Surface-Oriented Free-Form deformations. In *Proceedings of the Graphics Interface 2000*, 35–42.

SLOAN, P.-P. J., ROSE, C. F., AND COHEN, M. F. 2001. Shape by example. In *Symposium on Interactive 3D Graphics*, 135–144.

STOLLNITZ, E. J., DEROSE, T. D., AND SALESIN, D. H. 1996. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA.

TEICHMANN, M., AND TELLER, S. 1998. Assisted articulation of closed polygonal models. In *Computer Animation and Simulation '98*, 87–101.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics (Proceedings of SIGGRAPH 88) 22*, 4 (August), 269–278.

TERZOPOULOS, D., AND WITKIN, A. 1988. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications 8*, 6 (Nov.), 41–51.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH 87) 21*, 4 (July), 205–214.

WILHELMS, J., AND GELDER, A. V. 1997. Anatomically based modeling. In *Proceedings of SIGGRAPH 97*, 173–180.

WITKIN, A., AND WELCH, W. 1990. Fast animation and control of nonrigid structures. *Computer Graphics (Proceedings of SIGGRAPH 90) 24*, 4 (August), 243–252.