Visual Comput (2007)
DOI 10.1007/s00371-007-0096-9

ORIGINAL ARTICLE

Kevin T. McDonnell
Hong Qin

A novel framework for physically based sculpting and animation of free-form solids

K.T. McDonnell (✉)
Dowling College, Department of Mathematics and Computer Science, Idle Hour Blvd., Oakdale, NY, 11769-1999, USA
mcdonnek@dowling.edu

H. Qin
Department of Computer Science, Stony Brook University, Stony Brook, NY, 11794-4400, USA
qin@cs.sunysb.edu

Abstract This paper presents a new, physically based model for performing finite element simulation of deformable objects in which all quantities – strain, stress, displacement, etc. – are computed entirely in local frames of reference. In our framework, subdivision solids with non-homogeneous material properties, such as mass and deformation distributions, can be defined throughout continuous, volumetric domains. This capability enables an animator or virtual sculptor to exert fine-level control over deforming objects and to define a wide variety of physical behaviors. Furthermore, since all quantities pertinent to physical simulation are computed locally, our model facilitates both large-scale and small-scale deformations, as well as rigid or near-rigid transformations. We demonstrate applications of our framework in animation and interactive sculpting and show that interactive simulation of non-trivial, volumetric shapes is possible with our methodologies.

Keywords Physically based modeling · Subdivision algorithms · Virtual sculpting · Animation

1 Introduction and motivation

The goal of this paper is to articulate a new physically based model for manipulating, deforming and animating volumetric objects. Our framework marries the dynamic behaviors afforded by finite element models (FEM) with subdivision solid geometry of complicated topology. Finite element models permit the specification of a wide variety of physical behaviors, while subdivision solids facilitate the definition of topologically complex geometric objects. The major contribution of our work is a new model in which all quantities pertinent to physical simulation – stretching, shearing, stress, strain, etc. – are computed with respect to moving local reference frames. The use of local frames permits the finite elements to move through space freely, and the run-time solver need not reassemble the stiffness matrix with each time-step. Such attributes of a deformable model are very desirable in interactive sculpting and animation applications.

Our primary motivation for employing a finite element model over a mass-spring model is to permit an animator or virtual sculptor to exert fine-level control over the material distributions of sculpted objects. While in a mass-spring system it is possible to specify mass and stiffness distributions at mass points and springs, a finite element-based approach permits the assignment of material distributions throughout the continuous domain of the deforming object. Second, although the accuracy of physical simulation is not our primary interest in this work, FEM does have the desirable advantage over mass-spring models of increased realism by way of continuous material variation. This permits the animator to create deformations that are more life-like than can be achieved in the absence of finite elements. Third, a finite element-based approach can support downstream applications like deformation and interactive manipulation of volumetric data. Hence, we anticipate that our new model will be of interest to practitioners in fields other than geometric and physically based modeling.

Figure 1 illustrates examples of the design, manipulation and simulation processes facilitated by our novel solid modeling approach. First, the designer must construct an initial control mesh (Fig. 1a) by using sculpting tools like those briefly discussed in Sect. 4.1 and detailed in a previous paper [15]. Second, the model is subdivided at least once, and the resulting mesh of polyhedra is decomposed into a set of finite elements (Fig. 1b). Third, the designer employs physically based tools and/or runs an offline physical simulation to deform the object (Fig. 1c). Last, if it is required, a boundary representation of the model may be extracted and rendered (Fig. 1d) by determining which polygonal faces lie on the boundary surface of the model.

2 Background review

2.1 Subdivision solids

Subdivision solid algorithms take as input a polyhedral mesh and produce a refined mesh containing many more, smaller polyhedra. Typically, the algorithms are designed so as to reproduce a volumetric spline of certain type after an infinite number of subdivisions. The major advantage they possess over volumetric splines is the capability to define holes, handles and other non-trivial topological features in volumetric shapes. The first documented volumetric subdivision algorithm, that of MacCracken and Joy [13], generalizes cubic B-spline solids to meshes of arbitrary topology. Later, Bajaj and colleagues [1] proposed an alternative to the MacCracken–Joy algorithm that also reproduces cubic B-spline volumes under regular topological conditions but is easier to analyze mathematically. Chang et al. [6] derived a suite of C^1 subdivision solid schemes for application over hybrid tetrahedral/octahedral meshes. Other work includes the investigation of wavelet decompositions of subdivision volumes [3], hierarchical representation of time-varying data [12], applications of interpolatory subdivision volumes [4, 14, 19], and physically based animation and volumetric sculpting [5, 15].

2.2 Physically based modeling

Physically based models integrate physical properties with geometric representations to support applications in animation, shape design, mechanical simulation and fluid dynamics. Dynamic solid models were introduced to the modeling and computer graphics communities by Terzopoulos and colleagues [25]. Closely related to our work is that of Faloutsos et al. [8], who developed dynamic free-form deformations. These models facilitate physically based manipulation of objects embedded in a 3D space. Our work also bears resemblance to the models of Qin and colleagues [21, 22], who derived FEM-based dynamic models for direct manipulation of spline-based and subdivision-based surfaces, and to those of James and Pai [11], who developed a dynamic surface model based on the boundary element method (BEM). Also related are various techniques for muscle deformation, such as Ng-Thow-Hing's work [18] to develop physically based models of muscle deformations using B-spline volumes. Like Ng-Thow-Hing, we employ the volumetric B-spline as the primitive for simulating deformable shapes. Fedkiw's group has also extensively studied issues related to muscle deformation, such as the simulation of very large deformations [10], algorithms for generating finite element meshes suitable for simulating large deformations [24], level-set-based approaches [16], and models for achieving anatomically accurate deformations derived from real data [23]. Although our work does not focus on muscle deformation, we too must consider many of the issues addressed by Fedkiw et al., particularly with regard to large deformations and finite element mesh generation.

Debunne et al. [7] proposed an efficient time and space adaptive deformable model using finite elements that guarantees user-specified frame rates. Müller and colleagues [17] presented a dynamic model that employs non-rotating reference frames at the mesh vertices for computing elastic forces. In contrast, our model employs frames of reference that can rotate and deform arbitrarily. This gives our method the ability to simulate deforming objects that move freely through space. To our best knowledge, ours is the only volumetric, deformable model

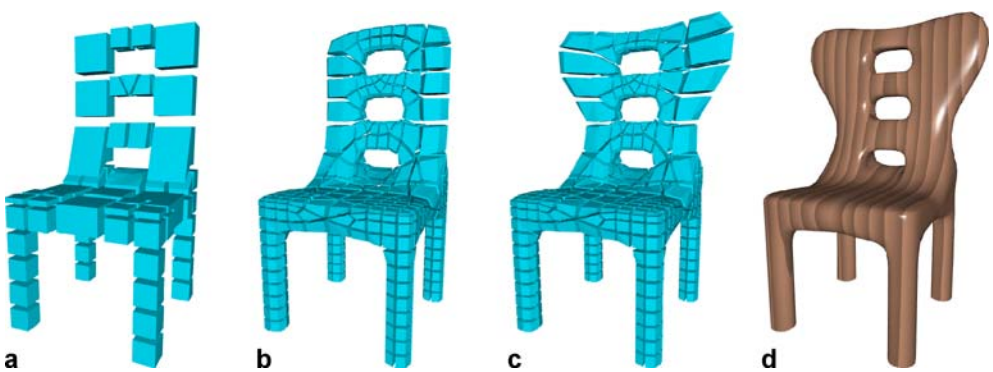


Fig. 1a–d. The design, manipulation and simulation processes featured in our dynamic solid modeling approach. **a** Design rough shape of object. **b** Subdivide model and discretize into finite elements. **c** Sculpt model and perform further modifications. **d** Extract B-rep and render final object. In this paper we are concerned mainly with the deformable models used in Step **c**

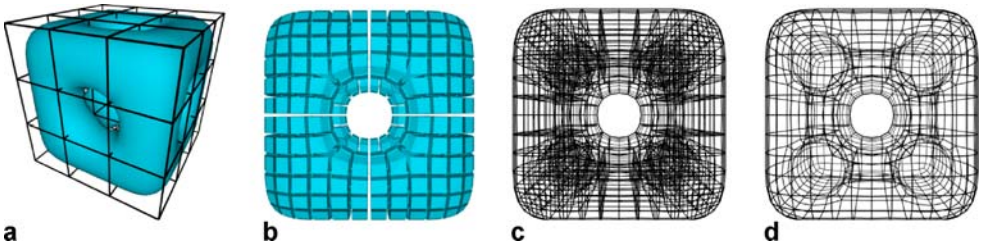


Fig. 2. a A subdivision solid after two levels of subdivision along with its control mesh. **b** Interior cell structure. **c** Wireframe rendering of subdivided solid. **d** Wireframe of boundary only

that employs local reference frames in this fashion. Hauth et al. [9] introduced a tetrahedral finite element model for simulation of biological tissues. Recently, Capell and colleagues [5] addressed the problem of how to perform hierarchical free-form deformation of embedded models.

3 FEM-based subdivision solids

3.1 Geometric representation

Throughout our discussion of FEM-based subdivision solids, we use MacCracken–Joy solids [13] for the underlying subdivision scheme. This algorithm, as well as most other procedural subdivision solid algorithms, can be expressed as a matrix multiplication:

$$\boldsymbol{d} = \boldsymbol{A}\boldsymbol{p}, \tag{1}$$

where $\boldsymbol{p}$ is a $3N$-vector consisting of the concatenated x, y and z positional components of the N control points; $\boldsymbol{A}$ is a sparse matrix whose entries are determined by the subdivision rules, and $\boldsymbol{d}$ is a $3M$-vector that concatenates all the components of the M nodal points that are used to approximate the limit solid after a certain number of subdivisions. An example of a MacCracken–Joy subdivision solid is depicted in Fig. 2. As one can see, the object produces a fairly uniform, volumetric decomposition of the initial control mesh and can accommodate features like holes and handles. In the interest of brevity, we do not provide the subdivision rules in this paper; instead, we refer the interested reader to MacCracken and Joy's original paper [13].

3.2 Dynamic representation

Conventional FEM procedures solve equations whose unknowns are displacements of nodal points from their initial, undeformed positions. Our FEM-based formulation starts from the following discrete form of the equation of motion:

$$\boldsymbol{M_x}\ddot{\boldsymbol{x}} + \boldsymbol{D_x}\dot{\boldsymbol{x}} + \boldsymbol{K_x}\delta_{\boldsymbol{x}} = \boldsymbol{f_x}, \tag{2}$$

where the $\boldsymbol{M_x}$, $\boldsymbol{D_x}$ and $\boldsymbol{K_x}$ matrices represent the mass, damping and stiffness distributions, respectively, of a modeled object. Note that $\dot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{x}}$ are the respective velocity and acceleration of the discretized object $\boldsymbol{x}$. $\delta_{\boldsymbol{x}}$ is the companion displacement vector derived from $\boldsymbol{x}$. In the new model described in this paper, $\delta_{\boldsymbol{x}}$ is calculated with respect to a moving local reference frame. Last, the vector $\boldsymbol{f_x}$ collects the total external forces acting on $\boldsymbol{x}$.

We associate with the FEM equation of motion the geometric and topological quantities of a subdivision solid. The physics of the subdivision solid is governed by:

$$\boldsymbol{M}\ddot{\boldsymbol{d}} + \boldsymbol{D}\dot{\boldsymbol{d}} + \boldsymbol{KRd} = \boldsymbol{f_d}, \tag{3}$$

where the $\boldsymbol{R}$ operator characterizes deformation due to stretching and shearing, and $\boldsymbol{d}$ is a vector of the concatenated x, y and z positional components of the FEM nodes. Essentially, the $\boldsymbol{R}$ operator measures local displacements along edges and changes in angles inside a polygonal face of a finite element. In practice, it is not necessary to assemble $\boldsymbol{R}$ since its effect on $\boldsymbol{d}$ can be computed by traversing local cell topology. Its structure is discussed in Sect. 3.2.2.

3.2.1 Mass and damping matrices

To define local $\boldsymbol{M}_i$, $\boldsymbol{D}_i$ and $\boldsymbol{K}_i$ matrices for element i, let us first define the matrix $\boldsymbol{J}$, which consists of the shape functions for the finite elements in a solid:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{\Gamma} & & \\ & \boldsymbol{\Gamma} & \\ & & \boldsymbol{\Gamma} \end{bmatrix} \tag{4}$$

and where $\boldsymbol{\Gamma} = [B_0B_0B_0 \;\; B_0B_0B_1 \;\; \cdots \;\; B_3B_3B_3]$ consists of the 64 uniform, tricubic B-spline basis functions.[1] The mass matrix can then be computed as $\boldsymbol{M}_i = \int_V \mu \boldsymbol{J}^\top \boldsymbol{J}\, \mathrm{d}V$, where $\mu(u, v, w)$ is the mass density function of one element of the solid. Similarly, damping matrix is computed as $\boldsymbol{D}_i = \int_V \gamma \boldsymbol{J}^\top \boldsymbol{J}\, \mathrm{d}V$, where $\gamma(u, v, w)$ is the damping density function.

3.2.2 Stiffness matrix

Since the subdivision solid has no global parameterization, deformations must be approximated through local

[1] We use the uniform B-spline basis functions for the shape functions because the MacCracken–Joy algorithm reproduces uniform tricubic B-spline solids in the limit of subdivision. Our formulation can be easily generalized to accommodate other subdivision solid schemes.

computations on individual cell geometry. Furthermore, we depart from traditional finite element methods [2] by computing *all* quantities relative to local frames of reference, rather than to a single global frame of reference. Strains along edges correspond to stretching deformations and can be computed as the changes in edge lengths in the hexahedral cells. Strains across faces correspond to shearing deformations and can be approximated by finding the change in internal face angles of the cells, as illustrated in Fig. 3. The shearing is estimated by finding the angle θ between two adjacent edges ($\boldsymbol{s}_1$ and $\boldsymbol{s}_2$) of a polyhedron: $\theta = \cos^{-1}\left(\frac{\boldsymbol{s}_1 \cdot \boldsymbol{s}_2}{|\boldsymbol{s}_1||\boldsymbol{s}_2|}\right)$. Both deformations are captured by the quantity $\boldsymbol{Rd}$. In particular, the i-th row of $\boldsymbol{Rd}$ consists of the sum two quantities: (i) the sum of all signed changes in edge lengths (from their rest lengths) of those edges incident on node i, and (ii) the sum of all vectors $a' - a$ and $b' - b$ induced by changes in angles (from their rest angles) of those faces of which node i is a member (see Fig. 3). As one can immediately discern, it is much simpler to compute and assemble $\boldsymbol{Rd}$ on-the-fly, rather than assemble $\boldsymbol{R}$ separately from $\boldsymbol{d}$.

During simulation, rather than assemble the global $\boldsymbol{K}$ matrix explicitly, it is easier to pre-assemble local stiffness matrices and multiply them against the local displacements given by $\boldsymbol{Rd}$. We shall use cubic finite elements throughout the discussion, but our approach can be applied to finite element shape functions of other degrees. Let $\boldsymbol{K}_i$ denote the local 192×192 stiffness matrix for cubic element i, and $\widehat{\boldsymbol{d}_i} = \boldsymbol{R}_i \boldsymbol{d}_i$ denote the 192-vector of concatenated displacements of the 64 nodes due to stretching and shearing. Conceptually, the matrix $\boldsymbol{R}_i$ encapsulates the computations required to calculate the local stretching and shearing displacements. As mentioned earlier, it need not be assembled explicitly. In each finite element we store, for each node, a pointer that indicates the node's corresponding row in the $\boldsymbol{Rd}$ vector. As the vector $\boldsymbol{R}_i \boldsymbol{d}_i$ is computed, its entries are added to the corresponding rows of $\boldsymbol{Rd}$. Given $\widehat{\boldsymbol{d}_i}$, we can compute $\boldsymbol{K}_i \widehat{\boldsymbol{d}_i}$, which corresponds to the forces contributed by elements to the entire solid as a result of stretching. Using pointers stored in the element, the vector $\boldsymbol{K}_i \widehat{\boldsymbol{d}_i}$ is added to the global vector $\boldsymbol{KRd}$, which represents the total internal forces introduced by stretching and shearing. The same process is carried out for all elements.

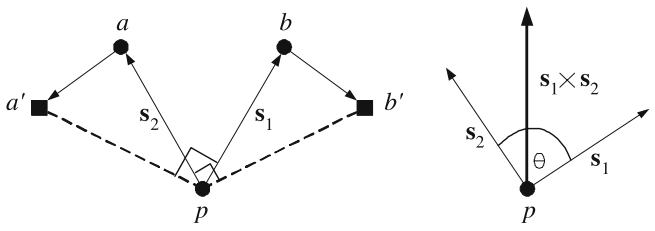


Fig. 3. To compute the shearing energy as defined by internal angles, we begin by computing where end-points a and b would be if there were no shearing (given by a' and b'). The points are rotated about the nodal point using the axis of rotation (determined by the cross product $\boldsymbol{s}_1 \times \boldsymbol{s}_2$). Using these virtual positions we compute a pair of displacements (aa' and bb'). In this example we assume the rest angle is 90 degrees

Now, we know from the study of mechanics that the finite element stiffness matrix of a solid body is written as

$$\boldsymbol{K} = \int_V \boldsymbol{B}^\top \boldsymbol{C} \boldsymbol{B} \, \mathrm{d}V, \tag{5}$$

where $\boldsymbol{B}$ is the stress-strain displacement matrix and whose rows are obtained by appropriately differentiating and summing the rows of the displacement interpolation matrix, $\boldsymbol{J}$. The $\boldsymbol{C}$ is the stress-strain material matrix for solid bodies and is defined as [2]:

$$\boldsymbol{C} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \times \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & & & & & \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & & & & & \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & & & & & \\ & & & \frac{1-2\nu}{2(1-\nu)} & & & & \\ & & & & \frac{1-2\nu}{2(1-\nu)} & & & \\ & & & & & \frac{1-2\nu}{2(1-\nu)} & & \\ & & & & & & & \end{bmatrix}, \tag{6}$$

where E is Young's modulus, ν is Poisson's ratio, and the matrix entries not shown are zeroes. It is also known that the strains $\boldsymbol{\epsilon} = [\epsilon_{uu} \ \epsilon_{vv} \ \epsilon_{ww} \ \gamma_{uv} \ \gamma_{vw} \ \gamma_{wu}]^\top$ generate stresses $\boldsymbol{\tau} = [\tau_{uu} \ \tau_{vv} \ \tau_{ww} \ \tau_{uv} \ \tau_{vw} \ \tau_{wu}]^\top$ according to $\boldsymbol{\tau} = \boldsymbol{C}\boldsymbol{\epsilon} + \boldsymbol{\tau}'$, where $\boldsymbol{\tau}'$ denotes initial stresses. By computing strains with respect to local coordinate frames we find $\epsilon_{uu} = \frac{\partial x}{\partial u}$, $\epsilon_{vv} = \frac{\partial y}{\partial v}$, $\epsilon_{ww} = \frac{\partial z}{\partial w}$, $\gamma_{uv} = \frac{\partial y}{\partial u} + \frac{\partial x}{\partial v}$, $\gamma_{vw} = \frac{\partial z}{\partial v} + \frac{\partial y}{\partial w}$ and $\gamma_{wu} = \frac{\partial x}{\partial w} + \frac{\partial z}{\partial u}$, where $u - v - w$ defines a local reference frame for a finite element (Figs. 4 and 5). Assuming there are no initial stresses, $\boldsymbol{\tau}^\top = \boldsymbol{\epsilon}^\top \boldsymbol{C}$ and we express the principle of virtual work as [2]:

$$\int_V \boldsymbol{\epsilon}^\top \boldsymbol{C} \boldsymbol{\epsilon} \, \mathrm{d}V = \int_V \widehat{\boldsymbol{d}}^\top \boldsymbol{f_p} \, \mathrm{d}V. \tag{7}$$

Using Eq. 5 and Eq. 7 we seek to obtain an explicit expression for $\boldsymbol{B}$, which will lead to the formulation for $\boldsymbol{K}$. In the interest of brevity, we will derive $\boldsymbol{B}$ for a cubic element, but the derivations for other element types follow suit. Let

$$x(u, v, w) = \boldsymbol{\Gamma} \boldsymbol{d}_{i,x} \tag{8}$$

describe the continuous x component of a cubic element's position as a function of the 64 nodal positions for element i. Let $\boldsymbol{d}_{i,x}$, $\boldsymbol{d}_{i,y}$ and $\boldsymbol{d}_{i,z}$ indicate the x, y and z

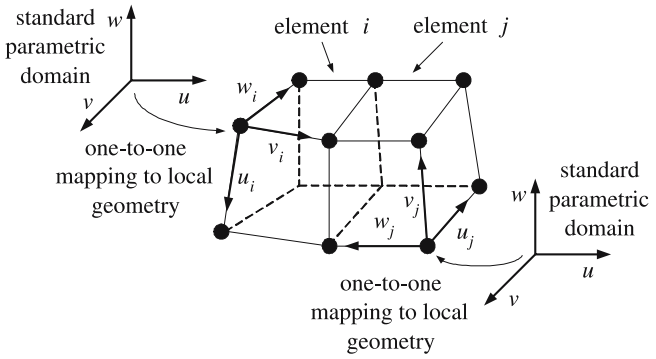


Fig. 4. The lack of a global parameterization requires each element to have its own local reference frame. The standard, orthogonal $u-v-w$ coordinate system is mapped to the local subdivision solid geometry to parameterize the element shape functions and material density distribution functions

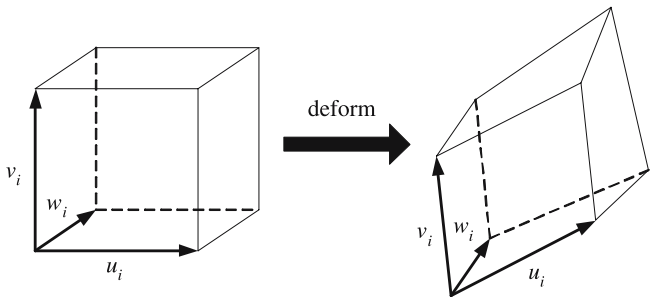


Fig. 5. As an object experiences external forces, the local reference frames stretch, shear and move with the finite elements

coordinates, respectively, of the nodes, and $\boldsymbol{\Gamma}$ indicate the 64 uniform, tricubic B-spline basis functions. Define $y(u, v, w)$ and $z(u, v, w)$ in a similar fashion. To simplify notation in the following discussion, a product of B-spline basis functions $B_j B_k B_l$ will be denoted $\widetilde{B}_m$, where $m = 16j + 4k + l$.

Now, let $\widehat{\boldsymbol{d}}_{i,x}$, $\widehat{\boldsymbol{d}}_{i,y}$ and $\widehat{\boldsymbol{d}}_{i,z}$ row-vectors represent the x, y and z components, respectively, of the 64 relative nodal displacements for a cubic element as computed by $\boldsymbol{R}_i$. Differentiating both sides of Eq. 8 with respect to u yields $\frac{\partial x}{\partial u} = \left[\frac{\partial \widetilde{B}_1}{\partial u} \cdots \frac{\partial \widetilde{B}_{64}}{\partial u}\right]\widehat{\boldsymbol{d}}_{i,x}$, which we will rewrite using matrix notation as $\frac{\partial x}{\partial u} = \boldsymbol{\Gamma}_u \widehat{\boldsymbol{d}}_{i,x}$, where the subscript u on $\boldsymbol{\Gamma}_u$ indicates differentiation of the basis functions with respect to u. Following a similar process yields expressions for the other partial derivatives and permits us to express the components of $\boldsymbol{\epsilon}$ as

$$\begin{aligned}
\epsilon_{uu} &= \boldsymbol{\Gamma}_u \widehat{\boldsymbol{d}}_{i,x}, \quad \epsilon_{vv} = \boldsymbol{\Gamma}_v \widehat{\boldsymbol{d}}_{i,y}, \quad \epsilon_{ww} = \boldsymbol{\Gamma}_w \widehat{\boldsymbol{d}}_{i,z}, \\
\gamma_{uv} &= \boldsymbol{\Gamma}_u \widehat{\boldsymbol{d}}_{i,y} + \boldsymbol{\Gamma}_v \widehat{\boldsymbol{d}}_{i,x}, \quad \gamma_{vw} = \boldsymbol{\Gamma}_v \widehat{\boldsymbol{d}}_{i,z} + \boldsymbol{\Gamma}_w \widehat{\boldsymbol{d}}_{i,y}, \\
\gamma_{wu} &= \boldsymbol{\Gamma}_w \widehat{\boldsymbol{d}}_{i,x} + \boldsymbol{\Gamma}_u \widehat{\boldsymbol{d}}_{i,z}.
\end{aligned}$$

After collecting and re-ordering the terms, we can express the strains $\boldsymbol{\epsilon}$ as

$$\begin{aligned}
\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{uu} \\ \epsilon_{vv} \\ \epsilon_{ww} \\ \gamma_{uv} \\ \gamma_{vw} \\ \gamma_{wu} \end{bmatrix} &= \begin{bmatrix} \frac{\partial \widetilde{B}_1}{\partial u} \cdots \frac{\partial \widetilde{B}_{64}}{\partial u} \\ 0 \cdots 0 \\ 0 \cdots 0 \\ \frac{\partial \widetilde{B}_1}{\partial v} \cdots \frac{\partial \widetilde{B}_{64}}{\partial v} \\ 0 \cdots 0 \\ \frac{\partial \widetilde{B}_1}{\partial w} \cdots \frac{\partial \widetilde{B}_{64}}{\partial w} \end{bmatrix} \widehat{\boldsymbol{d}}_{i,x} + \begin{bmatrix} 0 \cdots 0 \\ \frac{\partial \widetilde{B}_1}{\partial v} \cdots \frac{\partial \widetilde{B}_{64}}{\partial v} \\ 0 \cdots 0 \\ \frac{\partial \widetilde{B}_1}{\partial u} \cdots \frac{\partial \widetilde{B}_{64}}{\partial u} \\ \frac{\partial \widetilde{B}_1}{\partial w} \cdots \frac{\partial \widetilde{B}_{64}}{\partial w} \\ 0 \cdots 0 \end{bmatrix} \widehat{\boldsymbol{d}}_{i,y} \\
&+ \begin{bmatrix} 0 \cdots 0 \\ 0 \cdots 0 \\ \frac{\partial \widetilde{B}_1}{\partial w} \cdots \frac{\partial \widetilde{B}_{64}}{\partial w} \\ 0 \cdots 0 \\ \frac{\partial \widetilde{B}_1}{\partial v} \cdots \frac{\partial \widetilde{B}_{64}}{\partial v} \\ \frac{\partial \widetilde{B}_1}{\partial u} \cdots \frac{\partial \widetilde{B}_{64}}{\partial u} \end{bmatrix} \widehat{\boldsymbol{d}}_{i,z}, \qquad (9)
\end{aligned}$$

which we will denote as $\boldsymbol{\epsilon} = \widetilde{\boldsymbol{B}}_x \widehat{\boldsymbol{d}}_{i,x} + \widetilde{\boldsymbol{B}}_y \widehat{\boldsymbol{d}}_{i,y} + \widetilde{\boldsymbol{B}}_z \widehat{\boldsymbol{d}}_{i,z}$. Thus, the strains can be concisely expressed as

$$\boldsymbol{\epsilon} = \begin{bmatrix} \widetilde{\boldsymbol{B}}_x & & \\ & \widetilde{\boldsymbol{B}}_y & \\ & & \widetilde{\boldsymbol{B}}_z \end{bmatrix} \widehat{\boldsymbol{d}}_i = \boldsymbol{B}\widehat{\boldsymbol{d}}_i, \qquad (10)$$

where the empty entries in $\boldsymbol{\epsilon}$ are zeroes.

Substituting the right-hand side of Eq. 10 into the left side of Eq. 7 and carrying out the multiplication yields an expression of nine terms, the following five of which affect the x component of the work performed by the internal stresses:

$$\begin{aligned}
&\widehat{\boldsymbol{d}}_{i,x}^{\top}\left(\int_V \widetilde{\boldsymbol{B}}_x^{\top} \boldsymbol{C} \widetilde{\boldsymbol{B}}_x \, \mathrm{d}V\right)\widehat{\boldsymbol{d}}_{i,x} + \widehat{\boldsymbol{d}}_{i,x}^{\top}\left(\int_V \widetilde{\boldsymbol{B}}_x^{\top} \boldsymbol{C} \widetilde{\boldsymbol{B}}_y \, \mathrm{d}V\right)\widehat{\boldsymbol{d}}_{i,y} \\
&+ \widehat{\boldsymbol{d}}_{i,x}^{\top}\left(\int_V \widetilde{\boldsymbol{B}}_x^{\top} \boldsymbol{C} \widetilde{\boldsymbol{B}}_z \, \mathrm{d}V\right)\widehat{\boldsymbol{d}}_{i,z} + \widehat{\boldsymbol{d}}_{i,y}^{\top}\left(\int_V \widetilde{\boldsymbol{B}}_y^{\top} \boldsymbol{C} \widetilde{\boldsymbol{B}}_x \, \mathrm{d}V\right)\widehat{\boldsymbol{d}}_{i,x} \\
&+ \widehat{\boldsymbol{d}}_{i,z}^{\top}\left(\int_V \widetilde{\boldsymbol{B}}_z^{\top} \boldsymbol{C} \widetilde{\boldsymbol{B}}_x \, \mathrm{d}V\right)\widehat{\boldsymbol{d}}_{i,x}. \qquad (11)
\end{aligned}$$

The y and z components of the work can be defined in an similar fashion.

Differentiating the internal virtual work (Eq. 7) with respect to $\widehat{\boldsymbol{d}}_x$ and combining like terms gives the expres-

sion for the internal stresses in x:

$$\frac{\partial}{\partial \widehat{\boldsymbol{d}}_{i,x}} \int_V \boldsymbol{\epsilon}^\top \boldsymbol{C} \boldsymbol{\epsilon} \, \mathrm{d}V = 2\Big(\int_V \widetilde{\boldsymbol{B}}_x^\top \boldsymbol{C} \widetilde{\boldsymbol{B}}_x \, \mathrm{d}V \Big) \widehat{\boldsymbol{d}}_{i,x} + 2\Big(\int_V \widetilde{\boldsymbol{B}}_x^\top \boldsymbol{C} \widetilde{\boldsymbol{B}}_y \, \mathrm{d}V \Big) \widehat{\boldsymbol{d}}_{i,y} + 2\Big(\int_V \widetilde{\boldsymbol{B}}_x^\top \boldsymbol{C} \widetilde{\boldsymbol{B}}_z \, \mathrm{d}V \Big) \widehat{\boldsymbol{d}}_{i,z}, \tag{12}$$

which can be rewritten more concisely in matrix form as:

$$\frac{\partial}{\partial \widehat{\boldsymbol{d}}_{i,x}} \int_V \boldsymbol{\epsilon}^\top \boldsymbol{C} \boldsymbol{\epsilon} \, \mathrm{d}V = \boldsymbol{K}_x^x \widehat{\boldsymbol{d}}_{i,x} + \boldsymbol{K}_y^x \widehat{\boldsymbol{d}}_{i,y} + \boldsymbol{K}_z^x \widehat{\boldsymbol{d}}_{i,z}, \tag{13}$$

where $\boldsymbol{K}_x^x = 2 \int_V \widetilde{\boldsymbol{B}}_x^\top \boldsymbol{C} \widetilde{\boldsymbol{B}}_x \, \mathrm{d}V$, $\boldsymbol{K}_y^x = 2 \int_V \widetilde{\boldsymbol{B}}_x^\top \boldsymbol{C} \widetilde{\boldsymbol{B}}_y \, \mathrm{d}V$ and $\boldsymbol{K}_z^x = 2 \int_V \widetilde{\boldsymbol{B}}_x^\top \boldsymbol{C} \widetilde{\boldsymbol{B}}_z \, \mathrm{d}V$.

Given similar expressions for the stresses in the y and z directions, we arrive at the following expression for the internal stresses:

$$\begin{aligned} &\frac{\partial}{\partial \widehat{\boldsymbol{d}}_{i,x}} \int_V \boldsymbol{\epsilon}^\top \boldsymbol{C} \boldsymbol{\epsilon} \, \mathrm{d}V + \frac{\partial}{\partial \widehat{\boldsymbol{d}}_{i,y}} \int_V \boldsymbol{\epsilon}^\top \boldsymbol{C} \boldsymbol{\epsilon} \, \mathrm{d}V + \frac{\partial}{\partial \widehat{\boldsymbol{d}}_{i,z}} \int_V \boldsymbol{\epsilon}^\top \boldsymbol{C} \boldsymbol{\epsilon} \, \mathrm{d}V \\ &= \big(\boldsymbol{K}_x^x \widehat{\boldsymbol{d}}_{i,x} + \boldsymbol{K}_y^x \widehat{\boldsymbol{d}}_{i,y} + \boldsymbol{K}_z^x \widehat{\boldsymbol{d}}_{i,z} \big) \\ &\quad + \big(\boldsymbol{K}_x^y \widehat{\boldsymbol{d}}_{i,x} + \boldsymbol{K}_y^y \widehat{\boldsymbol{d}}_{i,y} + \boldsymbol{K}_z^y \widehat{\boldsymbol{d}}_{i,z} \big) \\ &\quad + \big(\boldsymbol{K}_x^z \widehat{\boldsymbol{d}}_{i,x} + \boldsymbol{K}_y^z \widehat{\boldsymbol{d}}_{i,y} + \boldsymbol{K}_z^z \widehat{\boldsymbol{d}}_{i,z} \big). \end{aligned} \tag{14}$$

Now let $\boldsymbol{K}_x = \boldsymbol{K}_x^x + \boldsymbol{K}_x^y + \boldsymbol{K}_x^z$ and likewise define $\boldsymbol{K}_y$ and $\boldsymbol{K}_z$. Finally, the product $\boldsymbol{K}_i \widehat{\boldsymbol{d}}_i$ can be evaluated from Eq. 14 by collecting the terms as follows:

$$\boldsymbol{K}_i \widehat{\boldsymbol{d}}_i = \begin{bmatrix} \boldsymbol{K}_x & & \\ & \boldsymbol{K}_y & \\ & & \boldsymbol{K}_z \end{bmatrix} \widehat{\boldsymbol{d}}_i. \tag{15}$$

In practice we do not assemble the global $\boldsymbol{K}$ since only its effect on $\widehat{\boldsymbol{d}}$ is required to solve the equation of motion (Eq. 20). (Rather, we compute the local $\boldsymbol{K}_i \widehat{\boldsymbol{d}}_i$ vectors and add them to the global stresses in $\boldsymbol{KRd}$ in an incremental, element-by-element fashion using the pointers stored in the elements.)

3.3 Numerical solver

Equation 3 can be integrated numerically through time using an implicit solver, where:

$$\dot{\boldsymbol{p}}(t + \Delta t) = \frac{\boldsymbol{p}(t + \Delta t) - \boldsymbol{p}(t - \Delta t)}{2 \Delta t} \tag{16}$$

and

$$\ddot{\boldsymbol{p}}(t + \Delta t) = \frac{\boldsymbol{p}(t + \Delta t) - 2\boldsymbol{p}(t) + \boldsymbol{p}(t - \Delta t)}{\Delta t^2}, \tag{17}$$

where t denotes time.

Now, multiplying both sides of Eq. 3 by $\boldsymbol{A}^\top$ and applying Eq. 1 yields:

$$\widetilde{\boldsymbol{M}} \ddot{\boldsymbol{p}} + \widetilde{\boldsymbol{D}} \dot{\boldsymbol{p}} + \widetilde{\boldsymbol{K}} \boldsymbol{p} = \tilde{\boldsymbol{f}}_{\boldsymbol{p}}, \tag{18}$$

where $\widetilde{\boldsymbol{M}} = \boldsymbol{A}^\top \boldsymbol{M} \boldsymbol{A}$, $\widetilde{\boldsymbol{D}} = \boldsymbol{A}^\top \boldsymbol{M} \boldsymbol{A}$, $\widetilde{\boldsymbol{K}} = \boldsymbol{A}^\top \boldsymbol{K} \boldsymbol{R} \boldsymbol{A}$ and $\tilde{\boldsymbol{f}}_{\boldsymbol{p}} = \boldsymbol{A}^\top \boldsymbol{f}_{\boldsymbol{d}}$. Substituting Eq. 16 and Eq. 17 into Eq. 18 yields

$$\widetilde{\boldsymbol{M}} \Big(\frac{\boldsymbol{p}(t + \Delta t) - 2\boldsymbol{p}(t) + \boldsymbol{p}(t - \Delta t)}{\Delta t^2} \Big) + \widetilde{\boldsymbol{D}} \Big(\frac{\boldsymbol{p}(t + \Delta t) - \boldsymbol{p}(t - \Delta t)}{2 \Delta t} \Big) + \widetilde{\boldsymbol{K}} \boldsymbol{p}(t) = \tilde{\boldsymbol{f}}_{\boldsymbol{p}}. \tag{19}$$

Last, multiplying both sides of Eq. 19 by $2\Delta t^2$ and rearranging terms produces

$$\begin{aligned} &(2\widetilde{\boldsymbol{M}} + \Delta t \widetilde{\boldsymbol{D}}) \boldsymbol{p}(t + \Delta t) \\ &= 2\Delta t^2 \big(\tilde{\boldsymbol{f}}_{\boldsymbol{p}} - \widetilde{\boldsymbol{K}} \boldsymbol{p}(t) \big) + 4\widetilde{\boldsymbol{M}} \boldsymbol{p}(t) - (2\widetilde{\boldsymbol{M}} - \Delta t \widetilde{\boldsymbol{D}}) \boldsymbol{p}(t - \Delta t). \end{aligned} \tag{20}$$

It is straightforward to employ the conjugate gradient method [20] to obtain an iterative solution for $\boldsymbol{p}(t + \Delta t)$. We have observed that only two to four iterations typically suffice to converge the system to a residual error of less than 10^{-4} because we employ cubic and linear shape functions. More than two are necessary when the physical parameters are changed during simulation. In such cases, the system automatically performs extra iterations to meet the error tolerance.

The generalized external force vector $\boldsymbol{f}_{\boldsymbol{d}}$ is obtained through the principle of virtual work done by the applied force distribution $\boldsymbol{f}(u, v, w, t)$ and is expressed as $\boldsymbol{f}_{\boldsymbol{d}} = \int_V \boldsymbol{J}^\top \boldsymbol{f}_{\boldsymbol{p}}(u, v, w, t) \mathrm{d}V$ where t is time. In theory, forces could be exerted at any continuous parametric values in an element. However, in practice, we restrict the application of forces to discrete quadrature points so as to simplify the implementation and to avoid an expensive root-finding procedure that would otherwise be required to find the appropriate values of (u, v, w). We believe this provides a reasonable trade-off between system performance and usability.

The shape functions, the material distribution functions, and the external force function are integrated using the technique of Gaussian quadrature [20]. This numerical integration technique approximates a definite integral with a summation, such as:

$$\int_{u_o}^{u_1} \int_{v_o}^{v_1} \int_{w_o}^{w_1} g(u, v, w) \mathrm{d}u \, \mathrm{d}v \, \mathrm{d}w \approx \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{L} q_i q_j q_l \, g(u_i, v_j, w_k), \tag{21}$$

where the weights q_i, q_j and q_k and parametric values u_i, v_j, and w_k are provided in look-up tables. These look-up tables are pre-computed and depend on the bounds of the parametric domains and the degree of the finite element shape functions. The accuracy of Gaussian quadrature is dependent on the number of samples (N, M, L) taken from the parametric domains. In our implementation we use only four samples in each parametric domain since most of the shape functions are cubic.

3.4 Finite elements

3.4.1 Regular cells

After several subdivisions of the initial control mesh, the vast majority of cells in the subdivided mesh are hexahedral (if we employ MacCracken–Joy subdivision). In order to compute the elemental mass, damping and stiffness matrices, we assign one finite element to each hexahedral cell that appears in the subdivided mesh. Since a MacCracken–Joy subdivision solid has no global parameterization, each element must be parameterized independently (see Figs. 4 and 5). Each element consists of the 8 vertices that comprise the cell geometry as well as 58 other vertices that complete the 64-node control mesh of the local B-spline solid. Together, the 64 vertices are used to characterize the FEM mass, damping and stiffness shape functions. In this manner, the shape functions define continuous material distributions across the regular cells. Local deformations are also stored for assembly later.

3.4.2 Extraordinary cells

While many of the cells in the subdivided MacCracken–Joy solid exhibit regular topology and can be parameterized using B-splines, a certain percentage of the hexahedral cells always requires special treatment. Such *extraordinary* cells include those on the boundary of the solid, as well as those within the 1-neighborhood of extraordinary vertices (i.e., interior vertices with more than or fewer than six neighbors). For each extraordinary cell, we can assign a linear, hexahedral element and employ traditional, time-tested finite element methodologies. In our experiments we did not observe any significant deformation artifacts caused by the mixing of linear and cubic elements in a single object.

3.4.3 Irregular cells

The presence of non-hexahedral cells in the control mesh results in the creation of a small number of *irregular* cells in the subdivided solid. Each irregular cell has an even number of quadrilateral faces. Irregular cells present a challenge for incorporation into the physical model since, to our knowledge, there are no standard finite elements that can handle octahedra, decahedra, and so forth. Our model employs pyramidal elements formed through the introduction of virtual nodes in the centers of each cell. The pyramidal elements are treated as degenerate, linear hexahedral elements [2] and participate fully in the simulation. This approach unavoidably introduces a small amount of error into the system. During our experiments we did not perceive any significant negative visual impact from the inclusion of these elements.

3.4.4 Element assignment

Given the three types of cells that exist in the dynamic solid model (regular, extraordinary and irregular), the system must be able to distinguish between the three types in order to assign finite elements. Irregular cells are trivial to discern and can be immediately subdivided into pyramids. Then the system must locate hexahedral cells that can be assigned cubic elements. The radial-edge data structure [26] provides invaluable assistance in this process. Our algorithm for detecting and assembling cubic elements proceeds as follows:

1. First, verify that (see Fig. 6a) the cell has 6 faces; each of the cell's faces is *not* on the boundary; each of the cell's 8 vertices is *not* on the boundary; each of the

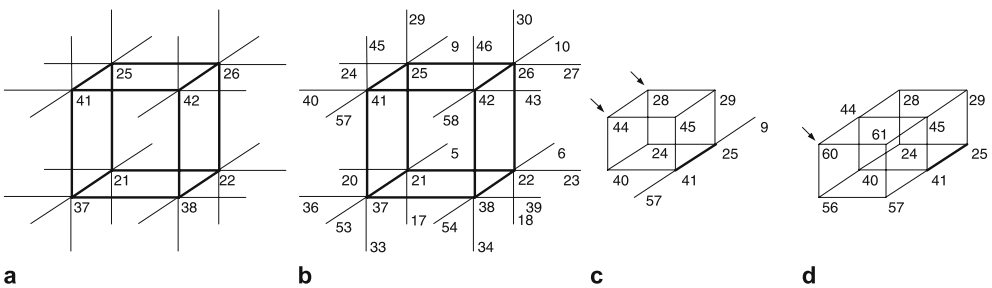


Fig. 6a–d. Assembly of cubic elements for a particular cell. The vertex numbers are used to compute the B-spline basis functions. **a** The cell itself. **b** The 24 vertices located using face-adjacency information. **c** Locating the two vertices (#28 and #44) of the cell that is edge-adjacent along the edge defined by vertices numbered 25 and 41. **d** Locating the vertex (#60) of the cell that is vertex-adjacent by vertex 41

Table 1. Details for some of the sculpted models discussed in this paper. #CC = Number of control cells. #FE = Number of finite elements. FPS = Frames per second. MSPI = milliseconds per iteration. %BE = percentage of elements that are cubic B-spline elements. %LE = percentage that are linear elements. %PE = percentage that are pyramidal elements. (Percentages may not sum to 100% due to rounding.) Timings were recorded on a 2.8 GHz Pentium 4-based computer with 1 GB RAM

Model	#CC	#FE	FPS	MSPI	%BE	%LE	%PE
Soccer player	24	1536	7.4	135	24%	76%	0%
Sea monster	38	320	21.3	47	94%	3%	4%
3^3 block	27	1728	3.2	310	58%	42%	0%
Chair	76	984	11.1	90	68%	3%	29%
4^3 block	64	296	11.9	84	58%	33%	0%
Head	123	1241	9.1	110	82%	1%	17%
Bridge	150	1200	4.7	213	95%	5%	0%
Nozzle	99	792	13.3	75	100%	0%	0%

cell's vertices has valence 8; and each of the cell's 26 neighbors is a hexahedron.
2. For each face, find the four vertices of the cell adjacent to the current cell that is *face-adjacent* to the current cell (Fig. 6b).
3. For each edge, find the two vertices of the cell adjacent to the current cell that is *edge-adjacent* to the current cell (Fig. 6c).
4. For each vertex, find the cell adjacent to the current cell that is *vertex-adjacent* to the current cell (Fig. 6d).

For sake of clarity, Fig. 6 shows how to locate only a subset of the vertices in Steps 3 and 4. The numbers assigned to each vertex are used to maintain a proper ordering and orientation of the vertices in memory so that the B-spline basis functions can be evaluated correctly. The adjacency information in the radial-edge data structure records, among other quantities, face orientation data, which makes it straightforward to identify and label the 64 vertices.

3.4.5 Element data structures

Each finite element's data structure contains the geometric and physical properties associated with the element, including pointers to appropriate components of the global vector $\boldsymbol{d}$ as well as the nodal displacements, $\widehat{\boldsymbol{d}}$. We also allocate in each element its elemental mass, damping and stiffness matrices, and include in the data structure the quantities needed to compute these matrices. These quantities include the continuous mass ($\mu(u, v, w)$) and damping ($\gamma(u, v, w)$) density functions, which are represented as parametric arrays of sample values.

4 Applications

We have applied our new FEM-based solid modeling approach to several domains, including haptics-based volumetric sculpting and animation of soft, solid bodies.

4.1 Shape design

Our dynamic, haptics-based sculpting system features numerous sculpting tools that permit users to design, simulate and probe virtual objects. One can also perform simulation and analysis of models to examine the behavior of virtual objects and to compute various material distributions. The hardware platform for both applications includes a generic PC and a Sensable Technologies PHANToM 1.0 3D haptic input/output device. Table 1 provides some timing information and details about the sculptures featured in this paper.

The design of dynamic subdivision solids typically begins with a series of topological and geometric modifications of a base shape, such as a cube. Our current implementation permits the user to delete material, create protrusions by extruding material from the surface, merge disconnected portions of a model, subdivide the model locally to generate detailed features,

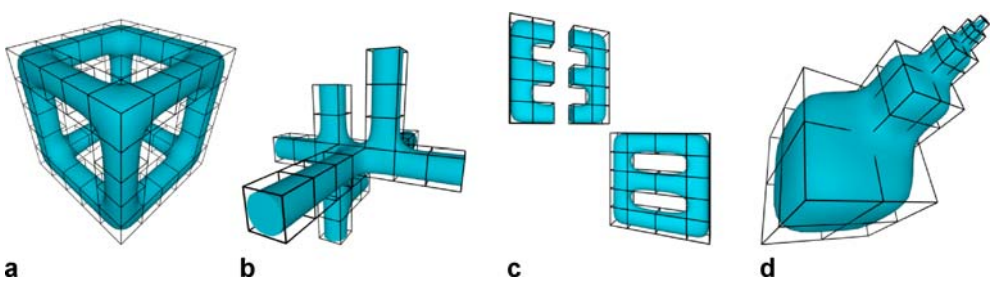


Fig. 7a–d. Topology-based sculpting tools: **a** deletion, **b** extrusion and local changes to subdivision rules, **c** merging, **d** local subdivision. The initial control mesh of each model is displayed in wireframe

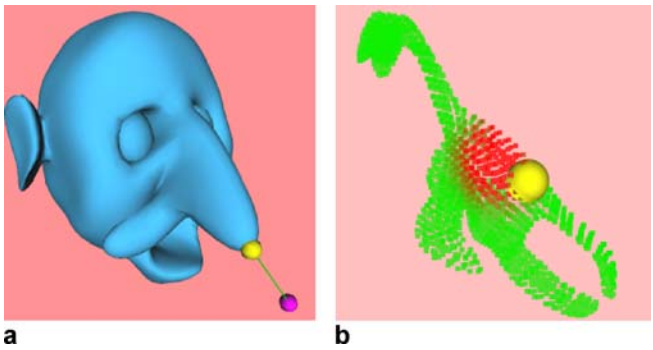


Fig. 8a,b. Haptics-based sculpting tools. **a** The user pulls on the man's nose. **b** The user probes the stiffness distribution of a sea monster model. Low stiffness = green, high stiffness = red. The color saturation of the background indicates the strength of the haptic forces felt by the user. These two objects were created with a previous system [15] but are shown here being manipulated with our new approach

and make local changes to the subdivision rules to create sharp features. These operations are illustrated in Fig. 7.

Our sculpting system also features several physically based tools that rely on forces to deform subdivision solids. The user may pull and push on the model, inflate and deflate objects, perform curve-based manipulation, and modify the stiffness and mass distributions at runtime. Figures 8 and 9 demonstrate these capabilities.

4.2 Animation

Our finite element formulation of dynamic subdivision solids can also be used to generate animations of solid bodies. Our finite element approach provides greater control and flexibility than a mass-spring model since the animator can specify forces and material quantities anywhere within the volumetric space occupied by an object. Figures 10–12 show several physically based animations. It is also possible to impose boundary constraints to change the outcome of the simulation, as can be seen in Figs. 13 and 14. Such constraints can be imposed via the penalty method (i.e., through strong spring forces) or through the dimension reduction technique. We used the latter in the referenced examples.

5 Conclusion

We have developed a new finite element modeling technique for physically based deformation and simulation of subdivision solid objects. Our approach employs a novel deformation model that is able to compute stretching

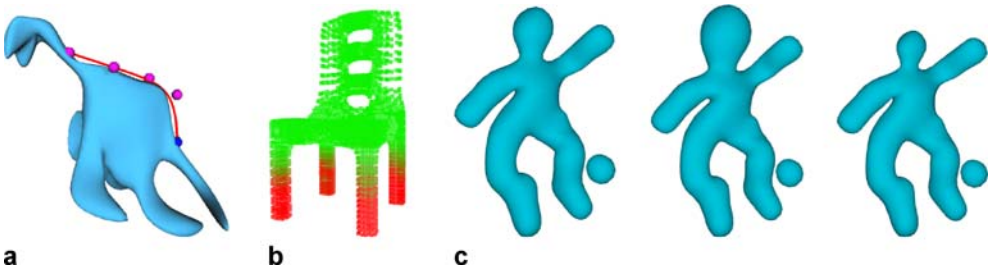


Fig. 9a–c. Force-based sculpting tools: **a** Curve-based manipulation, **b** stiffness painting. The red color in the legs of the chair indicate regions of high stiffness. **c** Inflation/deflation

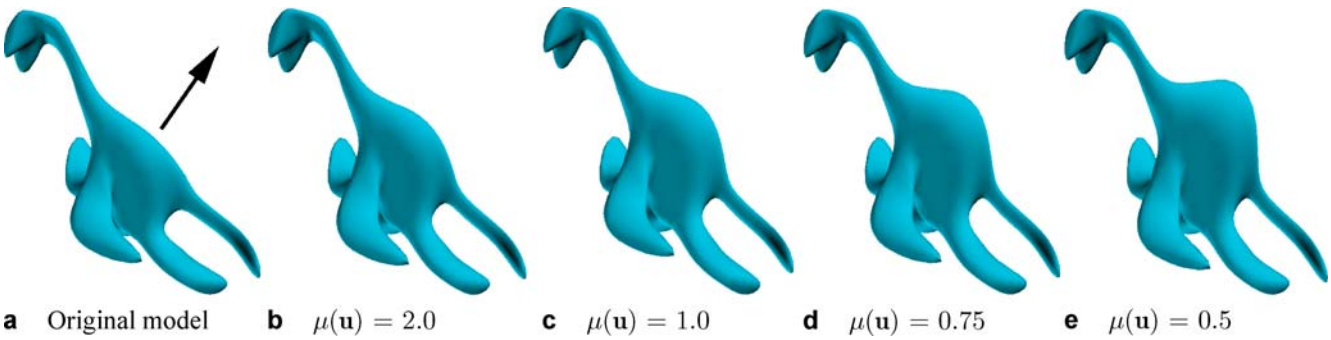


Fig. 10a–e. A model in which the mass distribution has been modified globally, but on which the same force has been applied. **a** Original model with indicated force to be applied. **b–e** Model with varying mass distributions after the external force has been applied. Here, $\boldsymbol{u} = (u, v, w)$

Fig. 11. A head sculpture is animated with our new finite element model

Fig. 12. A new chair is designed via force-based tools. Boundary constraints were imposed on the bottom portions of the legs to fix their positions

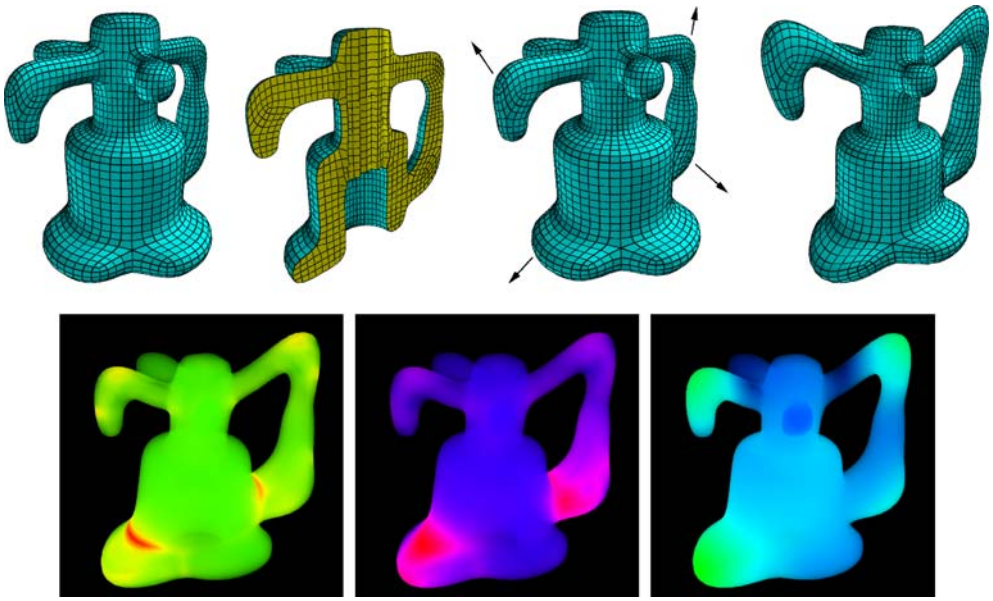


Fig. 13. A nozzle CAD model experiences external forces in the indicated directions. A cut-away view of the model shows the interior geometry. The three volume renderings show, respectively, the internal strains, the displacements of nodes from their initial positions, and the volumetric distortions of the finite elements

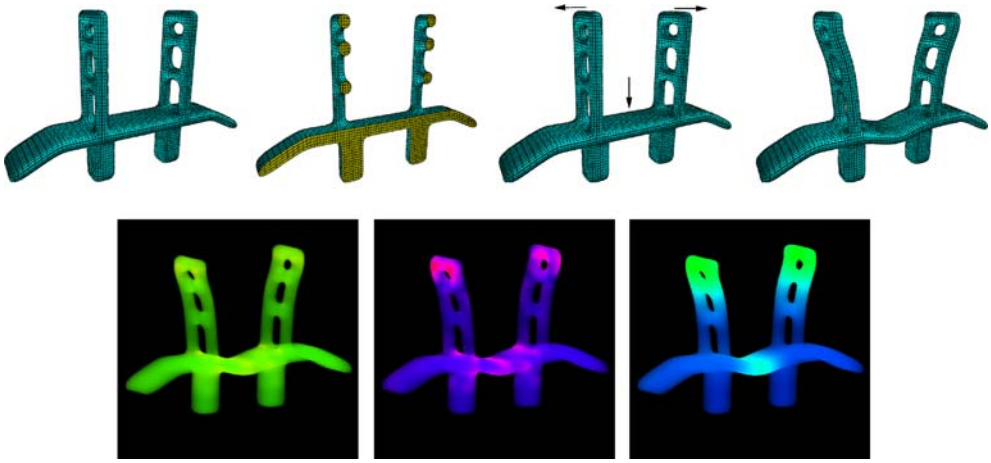


Fig. 14. A bridge CAD model experiences external forces in the indicated directions. Gaps in the vertical parts of the bridge result in a buckling effect. Note that boundary constraints were imposed on the lower parts of the pylons to prevent them from moving

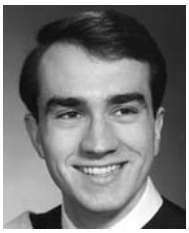
and shearing energies for elements moving freely through space. As a result, our approach marries some of the benefits of traditional finite element models – such as decomposition of solid bodies into continuous elements – with the ability to simulate large deformations and displacements of solid objects. We have shown that our new model is well-suited for practical use in interactive sculpting and animation, two applications that typically require free and unhindered movement of deforming shapes. Our framework can be implemented in a straightforward manner and supports simulation of non-trivial models at interactive rates on a modern PC platform.

Acknowledgement This research was supported in part by the NSF CAREER award CCR-9896123, the NSF grant DMI-9896170, the NSF ITR grant IIS-0082035, the NSF grant IIS-0097646, a Honda Initiation Award, an Alfred P. Sloan Fellowship, the GAANN grant P200A9703199, and a Dowling College equipment grant.

References

1. Bajaj, C., Shaefer, S., Warren, J., Xu, G.: A subdivision scheme for hexahedral meshes. Visual Comput. **18**(5–6), 343–356 (2002)
2. Bathe, K.-J.: Finite Element Procedures. Prentice Hall, Englewood Cliffs, NJ (1996)
3. Bertram, M.: Biorthogonal wavelets for subdivision volumes. In: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, pp. 72–82 (2002)
4. Bertram, M.: Volume refinement fairing isosurfaces. In: Proceedings of IEEE Visualization 2004, pp. 449–456 (2004)
5. Capell, S., Green, S., Curless, B., Duchamp, T., Popovic, Z.: A multiresolution framework for dynamic deformations. In: Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 41–47 (2002)
6. Chang, Y., Qin, H.: A framework for multi-dimensional adaptive subdivision objects. In: Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications, pp. 123–134 (2004)
7. Debunne, G., Desbrun, M., Cani, M., Barr, A. H.: Dynamic real-time deformations using space and time adaptive sampling. In: Computer Graphics (Proceedings of ACM SIGGRAPH 2001), pp. 31–36 (2001)
8. Faloutsos, P., van de Panne, M., Terzopoulos, D.: Dynamic free-form deformations for animation synthesis. IEEE T. Vis. Comput. Graph. **3**(3), 201–214 (1997)
9. Hauth, M., Gross, J., Strasser, W.: Interactive physically based solid dynamics. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 17–27 (2003)
10. Irving, G., Teran, J., Fedkiw, R.: Tetrahedral and hexahedral invertible finite elements. Graph. Models **68**(2), 66–89 (2006)
11. James, D. L., Pai, D. K.: ARTDEFO: Accurate Real Time Deformable Objects. In: Computer Graphics (Proceedings of ACM SIGGRAPH '99), pp. 65–72, 1999
12. Linsen, L., Pascucci, V., Duchaineau, M. A., Hamann, B., Joy, K. I.: Hierarchical representation of time-varying volume data with $\sqrt[4]{2}$ subdivision and quadrilinear B-spline wavelets. In: Proceedings of the Tenth Pacific Conference on Computer Graphics and Applications, pp. 346–355 (2002)
13. MacCracken, R., Joy, K. I.: Free-form deformations with lattices of arbitrary topology. In: Computer Graphics (Proceedings of ACM SIGGRAPH '96), pp. 181–188 (1996)
14. McDonnell, K. T., Chang, Y., Qin, H.: Interpolatory, solid subdivision of unstructured hexahedral meshes. Visual Comput. **20**(6), 418–436 (2004)
15. McDonnell, K. T., Qin, H., Wlodarczyk, R. A.: Virtual clay: A real-time sculpting system with haptic toolkits. In: Proceedings of 2001 ACM Symposium on Interactive 3D Graphics, pp. 179–190 (2001)
16. Molino, N., Bridson, R., Teran, J., Fedkiw, R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In: Proceedings of the 12th International Meshing Roundtable, pp. 103–114 (2003)
17. Müller, M., Dorsey, J., McMillan, L., Jagnow, R., Cutler, B.: Stable real-time deformations. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA) 2002, pp. 49–54 (2002)
18. Ng-Thow-Hing, V., Fiume, E.: Application-specific muscle representations. In: Proceedings of Graphics Interface 2002, pp. 107–115 (2002)
19. Pascucci, V., Bajaj, C.: Time critical isosurface refinement and smoothing. In: Proceedings of the 2000 IEEE Symposium on Volume Visualization, pp. 33–42 (2000)
20. Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P.: Numerical Recipes in C++: The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (2002)

21. Qin, H., Mandal, C., Vemuri, B. C.: Dynamic Catmull-Clark subdivision surfaces. IEEE T. Vis. Comput. Graph. **4**(3), 215–229 (1998)
22. Qin, H., Terzopoulos, D.: D-NURBS: A physics-based geometric design framework. IEEE T. Vis. Comput. Graph. **2**(1), 85–96 (1996)
23. Teran, J., Molino, N., Fedkiw, R., Bridson, R.: Adaptive physics based tetrahedral mesh generation using level sets. Eng. Comput. **21**(1), 2–18 (2005)
24. Teran, J., Sifakis, E., Blemker, S., Ng-Thow-Hing, V., Lau, C., Fedkiw, R.: Creating and simulating skeletal muscle from the Visible Human data set. IEEE T. Vis. Comput. Graph. **11**(3), 317–328 (2005)
25. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. Computer Graphics (Proceedings of ACM SIGGRAPH '87) **21**(4), 205–214 (1987)
26. Weiler, K. J.: Topological Structures for Geometric Modeling. PhD thesis, Rensselaer Polytechnic Institute, 1986



PROFESSOR KEVIN T. MCDONNELL is an assistant professor of Computer Science at Dowling College in Oakdale, NY. Previously, he was a research scientist in the Department of Computer Science at Stony Brook University, where he was also a fellow of the Center for Visual Computing (CVC). He received his Ph.D. in Computer Science at Stony Brook University in Computer Science in 2003, where he also earned his M.S. degree. He graduated summa cum laude with a B.S. in both Computer Science and Applied Mathematics from Stony Brook in 1998. In 1998 he was awarded both a University Graduate Research Fellowship and a GAANN fellowship (Graduate Assistance in Areas of National Need). His research interests include physically based modeling, geometric modeling, scientific visualization, and interactive 3D graphics. He is a member of Phi Beta Kappa, ACM, and the IEEE Computer Society.

DR. HONG QIN is a full professor (with tenure) of Computer Science in the Department of Computer Science at Stony Brook University. He received his B.S. degree and his M.S. degree in Computer Science from Peking University in Beijing, China. He received his Ph.D. (1995) degree in Computer Science from the University of Toronto. During his years at the University of Toronto (UofT), he received a UofT Open Doctoral Fellowship. In 1997, Professor Qin was awarded an NSF CAREER Award from the National Science Foundation (NSF). In December 2000, Professor Qin received a Honda Initiation Grant Award. In February 2001, Professor Qin was selected as an Alfred P. Sloan Research Fellow by the Sloan Foundation. Currently, he is an associate editor of IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG) and he is also on the editorial board of The Visual Computer (International Journal of Computer Graphics). His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer aided geometric design, human-computer interaction, visualization, and scientific computing. Detailed information about Dr. Hong Qin can be found on his website: http://www.cs.sunysb.edu/∼qin.