Dynamic Sculpting and Animation of Free-form Subdivision Solids

Kevin T. McDonnell and Hong Qin

Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400
{ktm|qin}@cs.sunysb.edu

Abstract

This paper presents a sculptured solid modeling system founded upon dynamic Catmull-Clark subdivision-based solids of arbitrary topology. Our primary contribution is that we integrate the geometry of sculptured free-form solids with the powerful physics-based modeling framework by augmenting pure geometric entities with material properties such as mass, damping, and stiffness distributions and with physical behaviors such as elasticity, plasticity, and natural deformation under external forces. Our novel dynamic model of free-form solids frees users from having to deal with low-level control point operations and permits them to interact with subdivision-based virtual clay in a more natural and intuitive fashion via "forces."

1. Introduction

To date, the vast majority of popular solid modeling approaches as well as commonly-used solid modeling systems are built upon the following geometric foundations: constructive solid geometry (CSG), boundary representation (B-reps), and cell decomposition. When the goals are to interactively sculpt solid objects, deform the solid geometry with ease in real-time, modify the solid topology, and conduct kinematic and dynamic analysis of physical solids, prior representations and the current state-of-the-art in solid modeling fall short in offering designers an array of flexible and powerful modeling and sculpting tools.

Free-form solid modeling provides modelers with a more flexible interface for designing a much wider range of objects than the aforementioned approaches. However, free-form solids typically offer users more degrees of freedom (*i.e.*, control points, weights, etc.) than what they can actually handle. In addition, free-form solids based on parametric geometry are constrained to modeling topologically regular shapes.

Subdivision-based solid modeling, such as the Catmull-Clark subdivision scheme for volumetric construction, allows users to create sculptured solids of arbitrary topology from user-specified initial control lattices. Despite their modeling superiority, subdivision solids also suffer from a large number of control points and an associated complex topological structure of their control lattices.

In this paper, we systematically develop a physics-based modeling framework for free-form solids that can overcome many of the limitations associated with conventional solid modeling techniques. Within our novel dynamic modeling framework, free-form solids are equipped with mass and damping distributions, internal deformation energies, and other material properties. Consequently, users can sculpt solids in a physically plausible and accurate manner as if they are manipulating real-world *physical clay*. Figure 1 illustrates several such objects that were sculpted using our system. We have developed a prototype solid modeling application in which the real-time deformation and sculpting processes can be easily facilitated through a set of intuitive virtual tools and through the use of efficient numerical algorithms.

The remainder of this paper is organized as follows. We discuss the motivation and present our contributions in Section 2. In Section 3, we detail the geometry of free-form solids and review prior research in relevant areas. In Section 4 we present our dynamic formulation and algorithms. Section 5 discusses the details of our implementation and sculpting system and then presents our experimental results with time performance. Finally, we conclude the paper and outline future research directions in Section 6.

2. Motivation and contribution

Solid modeling has recently emerged as a very powerful paradigm that can greatly enhance existing surface modeling techniques because of its unique advantages over curve and surface modeling. Despite many advances in solid modeling during the past decade, conventional solid modeling techniques based on algebraic geometry can be rather

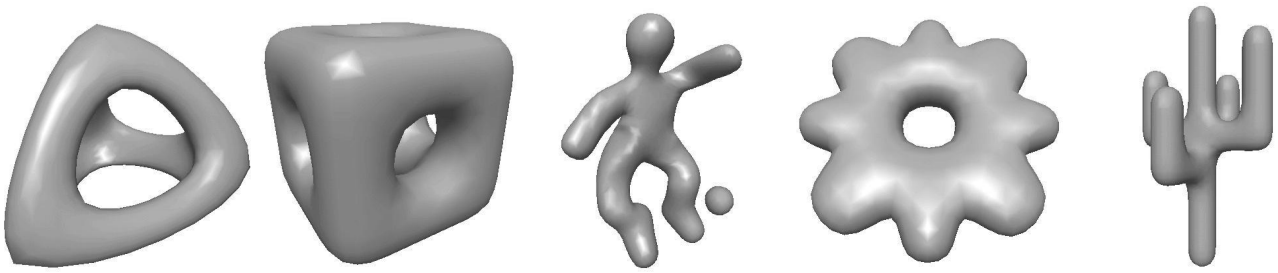

Figure 1. Several dynamic subdivision-based solids: tetrahedron with holes, cube with holes, soccer player, gear-like object, cactus.

rigid and inflexible. Free-form solids are a superior modeling candidate to CSG, B-reps and cell decomposition because (1) they have great potential to model a much wider range of real-world objects; (2) they combine the benefits of free-form surface boundaries and interior geometry within a unified representation scheme; and (3) they facilitate efficient algorithms for both interior interrogation and boundary evaluation.

Nevertheless, state-of-the-art free-form solids can be very difficult to use due to their bewildering number of degrees of freedom and their dependency on unintuitive control point manipulation. Physics-based modeling attempts to overcome such shortcomings of geometric modeling through the integration of material attributes and physical behaviors with powerful geometric modeling techniques. This approach alleviates the user's burden of managing large sets of degrees of freedom, which are typically required for the design of large, complicated objects. Furthermore, a purely geometric representation of a solid does not permit users to effectively validate physically-relevant tests such as finite element analysis, kinematic simulation, and material property calculation. Physics-based modeling approaches allow users to focus more attention on the object and to more easily create a large array of shape variations permitted by the underlying solid mathematics. Based on our physics-based modeling methodology for surface design, in this paper we forge ahead to tackle the challenging problem of integrating dynamic modeling algorithms with free-form solids in order to facilitate volumetric modeling, synthesis, and manipulation. Our primary contributions are as follows:

- We integrate material attributes with Catmull-Clark subdivision solids and formulate a set of equations of motion for arbitrary free-form solids.
- We develop a simple yet effective real-time numerical solver that employs a finite difference approximation for the finite element formulation of free-form solids.
- We discretize Catmull-Clark subdivision solids into a set of cells bounded by a three-dimensional lattice. A mass-spring discretization provides users with the intuitive mechanism for physically manipulating solid geometry.
- The mathematics of parametric geometry for free-form solids always constrains the behavior of the mass-spring lattice, enforces the spline structure, and synchronizes two different representations throughout the sculpting task.
- We implement a sculpting system for Catmull-Clark solids that should appeal to both engineering designers and to non-technical users because it frees users from the need to understand the underlying complicated mathematics of free-form solids.

3. Background

3.1. Catmull-Clark subdivision solids

Unlike subdivision curves and surfaces, little research has been published on modeling solids through subdivision techniques. MacCracken and Joy extended the subdivision rules for Catmull-Clark surfaces to generate a volumetric model from a lattice of control vertices [3]. However, their goal for using subdivision techniques was to contrive a space in which an arbitrary object could be deformed (*i.e.*, free-form deformation (FFD)). In this paper, we treat the volumetric Catmull-Clark subdivision scheme as a novel free-form spline solid that is obtained in the limit through recursive application of subdivision rules on a user-specified lattice of control vertices.

Given an initial *control lattice* in 3-space, the Catmull-Clark solid subdivision rules recursively subdivide the lattice and refine the three-dimensional space occupied by the lattice. The lattice consists of a set of closed cells that are defined by a collection of their constituent faces. The faces in the lattice are comprised of an ordered list of edges that are defined by their end vertices. Hence, there are four types of geometric entities in the lattice: cells, faces, edges and points. Figure 2 shows an example of a Catmull-Clark solid.

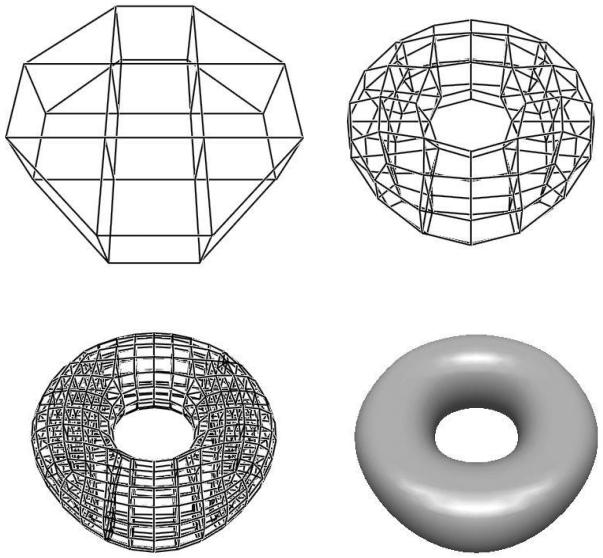

Figure 2. A torus-like solid object that has been subdivided a few times using the Catmull-Clark solid subdivision rules. The complex topology is particularly evident in the first image.

The subdivision scheme recursively applies a set of four rules, one for each type of element, in order to achieve successively finer representations of the original lattice. Each element in the lattice produces a new vertex that must be subsequently incorporated into the next finer level of the subdivided lattice.

The Catmull-Clark solid subdivision rules include:

- Cell points: for each cell, the cell point is its centroid.
- Face points: for each face, the face point is the weighted average: $\mathbf{f} = \frac{\mathbf{c}_0+2\mathbf{a}+\mathbf{c}_1}{4}$, where $\mathbf{a}$ is the face's centroid and $\mathbf{c}_0$ and $\mathbf{c}_1$ are the centroids of the cells on either side of the face.
- Edge points: for each edge, the edge point is the weighted average: $\mathbf{e} = \frac{\mathbf{c}_{avg}+2\mathbf{a}_{avg}+(n-3)\mathbf{m}}{n}$, where $\mathbf{c}_{avg}$ is the average of the centroids of the cells that contain the edge, $\mathbf{a}_{avg}$ is the average of the centroids of the faces that contain the edge, $\mathbf{m}$ is the midpoint of the edge, and n is the number of faces that contain the edge.
- Vertex points: for each vertex $\mathbf{p}$, the vertex point is the weighted average: $\mathbf{v} = \frac{\mathbf{c}_{avg}+3\mathbf{a}_{avg}+3\mathbf{m}_{avg}+\mathbf{p}}{8}$, where $\mathbf{c}_{avg}$ is the average of the centroids of the cells that contain the point, $\mathbf{a}_{avg}$ is the average of the centroids of the faces that contain the point, and $\mathbf{m}_{avg}$ is the average of the midpoints of the edges that contain the point.

Note that these rules are a straightforward extension of the subdivision rules for surfaces originally described by Catmull and Clark [1].

The new lattice is then assembled as follows. Cell points are connected to the new face points of the faces that defined the cell; face points are connected to the new edge points of the edges that defined the face; and edge points are connected to the new vertex points of the vertices that defined the edge. In the limit of this recursive subdivision process, a smooth free-form solid $\mathbf{s}$ can be obtained:

$$\mathbf{s}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{p}_i B_i(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ is a parametric value whose domain is a 3-space occupied by the initial lattice. $\mathbf{p}_k$ is a control point, which is one of the original lattice points, $B_k(\mathbf{x})$ is a basis function, and n is the number of the initial vertices defined by the original lattice. Note that the affine rules explained above naturally establish one-to-one point correspondences between lattices in two consecutive levels within the subdivision hierarchy. As with Catmull-Clark surfaces [8], the basis functions $B_k(\mathbf{x})$ of Catmull-Clark solids in the limit can be defined explicitly over the original lattice. However, it is non-trivial to derive the closed-form analytic equation for $B_i(\mathbf{x})$, given an initial control lattice of arbitrary connectivity and topology. In contrast, when the initial lattice is regular, a certain basis function $B_h(\mathbf{x})$ can be computed from three univariate B-spline basis functions: $B_h(\mathbf{x}) = B_{i,q}(u)B_{j,r}(v)B_{k,s}(w)$, where $B_{i,q}(u)$, $B_{j,r}(v)$, and $B_{k,s}(w)$ are the piecewise B-spline polynomials. This is because the Catmull-Clark solid of a regular lattice reduces to a standard tri-cubic B-spline solid.

3.2. Physics-based modeling

Although free-form modeling approaches are powerful for representing smooth volumetric shapes, they constitute a purely geometric representation. In addition, conventional geometric modeling may be inconvenient for representing complicated solids, because modelers are faced with the tedium of indirect shape modification and refinement through time-consuming operations on a large number of control vertices. In contrast, physics-based models respond to externally applied forces in a very intuitive manner. The dynamic formulation marries the model geometry with time, mass, damping and constraints via a force-balance equation. Dynamic models produce smooth, natural motions that are intuitive to control. In addition, they facilitate interaction –

especially direct manipulation – of complex geometries and topologies.

Free-form deformable models were introduced to computer graphics by Terzopoulos *et al.* [11] and further developed by Terzopoulos and Fleischer [10], Pentland and Williams [7], and Metaxas and Terzopoulos [5]. Qin and Terzopoulos introduced D-NURBS surfaces, an extension to traditional NURBS that permits more natural control of the geometry of the surface [9, 12]. Later, Qin *et al.* extended such ideas to dynamic subdivision surfaces [8, 4]. Most recently, Dachille *et al.* combined haptic interaction with dynamic B-spline surfaces to provide a very natural user interface for deformation [2]. This paper incorporates spline-based and subdivision-based solid modeling into the dynamic framework.

4. Formulation and algorithms

4.1. Dynamic Catmull-Clark solids

In order to associate material properties with a Catmull-Clark solid, we begin by expressing the subdivision process as a matrix-vector multiplication:

$$\mathbf{d} = \mathbf{A}\mathbf{p} \tag{2}$$

where $\mathbf{p}$ contains the positions of the lattice points at the initial, coarsest level. As shown in Equation 1, such points are called the "control points" of a Catmull-Clark solid in analogy with parametric solids. Vector $\mathbf{d}$ contains the positions of the points in the lattice at the current subdivision level. We shall call these vertices "data points" or "mass points," since each of them is assigned a mass. We shall use the discretized point set $\mathbf{d}$ to approximate the continuous Catmull-Clark solid in the interest of the simplicity and efficiency of dynamic simulation and manipulation. Note that after several levels of subdivision are conducted, $\mathbf{d}$ can be considered a good approximation with high precision. Matrix $\mathbf{A}$ contains weights given by the subdivision rules and defines how to obtain the data point positions in $\mathbf{d}$ from $\mathbf{p}$.

During the subdivision process, each new data point is defined by a certain affine combination of the existing points as computed by the subdivision rules. We can store this collection of weights for each point in $\mathbf{A}$. We can then perform the matrix multiplication $\mathbf{d} = \mathbf{A}\mathbf{p}$ and obtain the positions of the points in the new lattice.

4.2. Dynamics equations

Now we set the stage to define how the dynamic system evolves over time. A dynamic solid is characterized by its position $\mathbf{s}(\mathbf{x}, t)$, velocity $\dot{\mathbf{s}}(\mathbf{x}, t)$ (which stands for $\frac{\partial \mathbf{s}(\mathbf{x},t)}{\partial t}$), and acceleration $\ddot{\mathbf{s}}(\mathbf{x}, t)$ (*i.e.,* $\frac{\partial^2 \mathbf{s}(\mathbf{x},t)}{\partial t^2}$) along with material properties including mass density $\mu(\mathbf{x})$, damping density $\gamma(\mathbf{x})$, and internal energy functional $E(\mathbf{s})$. The continuous form of Lagrangian equations of motion can be written as

$$\mu\ddot{\mathbf{s}} + \gamma\dot{\mathbf{s}} + \frac{\partial E(\mathbf{s})}{\partial \mathbf{s}} = \mathbf{f} \tag{3}$$

where $\mathbf{f}(\mathbf{s})$ is the sum of all external force distribution acting on $\mathbf{s}(\mathbf{x})$. A large variety of physical behavior of $\mathbf{s}$ (*e.g.,* elasticity and/or plasticity) results from different mathematical formulations of energy functionals E [10].

The discretization of free-form solids will transform Equation 3 to Lagrangian dynamics of all mass points:

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{D}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{f_d} \tag{4}$$

Since the data point set of a solid is constrained by control points, we obtain a new set of motion equations whose unknowns are $\mathbf{p}$:

$$\mathbf{A}^\top\mathbf{M}\mathbf{A}\ddot{\mathbf{p}} + \mathbf{A}^\top\mathbf{D}\mathbf{A}\dot{\mathbf{p}} + \mathbf{A}^\top\mathbf{K}\mathbf{A}\mathbf{p} = \mathbf{A}^\top\mathbf{f_d} \tag{5}$$

Therefore, we can directly compute the acceleration of the control points ($\ddot{\mathbf{p}}$) based on the forces acting on the lattice of the current data points $\mathbf{d}$:

$$\ddot{\mathbf{p}} = (\mathbf{A}^\top\mathbf{M}\mathbf{A})^{-1}(\mathbf{A}^\top\mathbf{f_d} - \mathbf{A}^\top\mathbf{D}\dot{\mathbf{d}} - \mathbf{A}^\top\mathbf{K}\mathbf{d}). \tag{6}$$

4.3. Numerical integration

In order to achieve real-time, interactive performance in our sculpting system, we employ an explicit integration method to steer our physical simulation. At each time-step, the state of the system is advanced based on the preceding states. The summarized forces on the discretized lattice are applied, and then the accelerations of the control points are computed using Equation 6.

The velocities $\dot{\mathbf{p}}$ of the control points are updated according to the applied forces and material quantities (*e.g.,* mass, damping, and stiffness). The control points are moved to their new positions:

$$\dot{\mathbf{p}}_{i+1} = \dot{\mathbf{p}}_i + \ddot{\mathbf{p}}_i\Delta t \qquad \mathbf{p}_{i+1} = \mathbf{p}_i + \dot{\mathbf{p}}_i\Delta t$$

where subscripts denote time, and Δt stands for one time-step. The updated control point positions $\mathbf{p}_{i+1}$ and their velocities are further used to update the discretized model defined by $\mathbf{d}_{i+1} = \mathbf{A}\mathbf{p}_{i+1}$ and $\dot{\mathbf{d}}_{i+1} = \mathbf{A}\dot{\mathbf{p}}_{i+1}$.

5. Sculpting system

We have developed a prototype sculpting system that allows users to interactively model, design, and deform Catmull-Clark solids in real-time. This section details the implementation issues.

5.1. Forces and constraints

The discretized dynamic model has material properties such as mass, stiffness and damping distributions. The mass points are connected to their immediate neighbors through springs. These springs seek to maintain the Euclidean distances between all neighboring points. However, nearest-neighbor connections would not sufficiently constrain the overall deformation of the solid. For this reason, we incorporate an additional set of *virtual springs* into conventional mass-spring systems. These virtual springs, whose stiffnesses may be different from normal springs, are intended to help maintain internal angles of each face, and hence help minimize the volumetric distortion of a solid object. Note that since the stiffness matrix $\mathbf{K}$ (derived from E) is a function of $\mathbf{s}$ and is extremely complex due to the non-quadratic nature of E, it would be nearly impossible to achieve real-time sculpting performance if the system were to assemble it at each time-step based on the energy functional E. Therefore, our mass-spring configuration explained above only intends to achieve the objective of minimizing the volumetric distortion. More accurate implementations based on the finite element method would be more desirable for certain applications such as engineering design and analysis. Such approaches are currently under investigation.

Figure 3 illustrates an example of four virtual springs for a single face element. For each point $\mathbf{m}$ of each face in the finest lattice, we introduce a corresponding *virtual point*, $\mathbf{v}$, whose location is defined as the midpoint of the two points adjacent to point $\mathbf{m}$ in the same face. One end of the virtual spring is attached to $\mathbf{m}$ and the other end to the virtual point. The location of the virtual point is updated dynamically as the accompanying positions of the two points evolve over time. The virtual springs resist any changes in their corresponding internal angles for each face and thereby assist in minimizing local distortions. Note that unlike traditional "diagonal" springs, one end-point of a virtual spring is a mass point and the other a massless point whose position is determined by geometry. Figure 4 shows a typical cell without and with virtual springs. In the interest of clarity, normal springs are drawn as straight lines.

Figure 5 provides a global view of the major steps in our system architecture. After an initial preprocessing stage, the simulation runs in a loop and continuously updates the state of the dynamic solid object in real-time. At any time during the simulation the user may modify the object's topology, and the system will rebuild the necessary data structures to effect the change.

5.2. Fixed regions and local modifications

For large control lattices our model can potentially be computationally expensive due to a very large number of

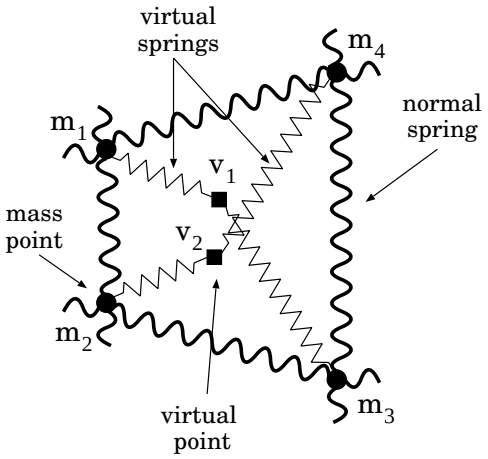

Figure 3. The virtual and normal springs associated with a typical face of a lattice. Mass points are drawn as filled circles, virtual points as filled squares. The position of mass point $\mathbf{m_1}$'s virtual point, $\mathbf{v_1}$, is defined as the average of the positions of $\mathbf{m_1}$'s neighboring mass points: $\mathbf{v_1} = \frac{1}{2}(\mathbf{m_2} + \mathbf{m_4})$.

mass points. In order to improve system performance, our model can be modified to constrain a set of selected control points that are attributed to certain fixed regions. This reduces the amount of data that needs to be processed by the system and thereby increases the simulation speed. Such modifications also enable the designer to make changes only in the localized region(s) of his/her interest without affecting a large portion of the solid object.

The necessary modifications to the matrices are fairly straightforward. We break the free and fixed portions of the model into two matrix-vector multiplications that we can then add together. Specifically, we separate each of $\mathbf{p}$ and $\mathbf{A}$ into two matrices: $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{A}_1$ and $\mathbf{A}_2$, where $\mathbf{p}_1$ and $\mathbf{p}_2$ contain the positions of the free and fixed control points, respectively, and $\mathbf{A}_1$ and $\mathbf{A}_2$ contain the weights for the free and fixed control points, respectively. The familiar $\mathbf{d} = \mathbf{A}\mathbf{p}$ is replaced with $\mathbf{d} = \mathbf{A}_1\mathbf{p}_1 + \mathbf{A}_2\mathbf{p}_2$ so that the positions of the fixed points in $\mathbf{p}_2$ correctly influence the positions in $\mathbf{d}$. Figure 6 illustrates this change. Vector $\mathbf{d}_1$ contains the positions of the free points from $\mathbf{d}$. Note that there is no need to assemble a second matrix for the fixed points of $\mathbf{d}$ since they will not appear in any matrix multiplication.

5.3. Data structures

A Catmull-Clark solid requires a somewhat sophisticated data structure to store the adjacency and connectivity information of the lattice. For this purpose we used a

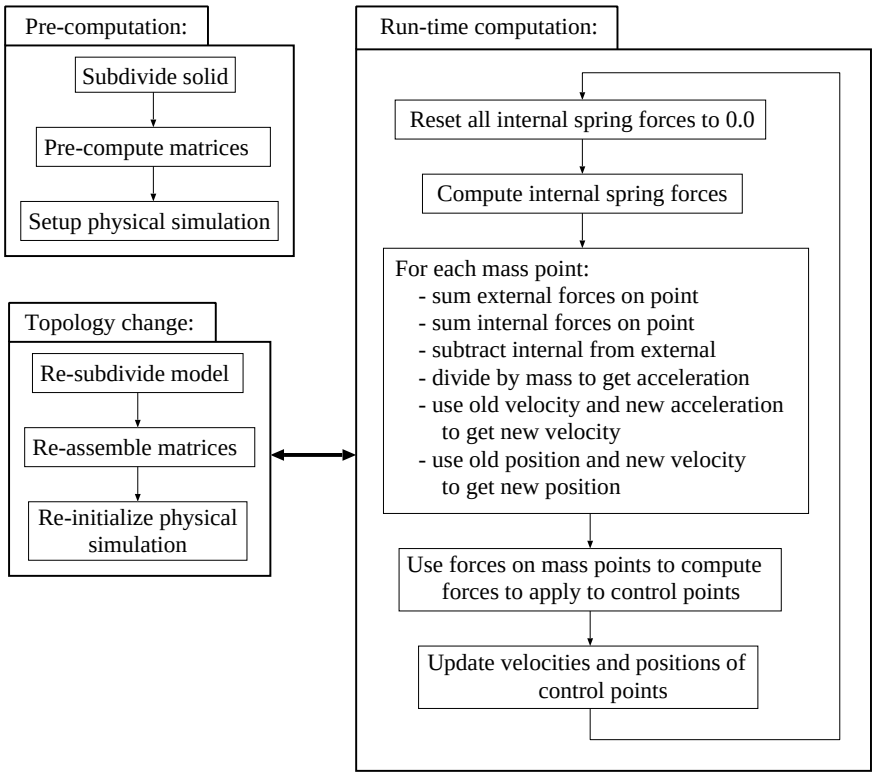

Figure 5. System architecture of the physical simulation.

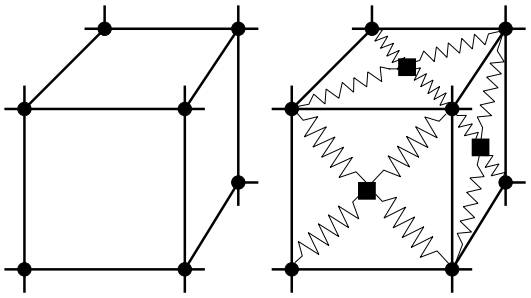

Figure 4. A typical regular cell of a mass-spring lattice both without and with virtual springs. Normal springs are drawn as straight lines in order to avoid cluttering the figure. Mass points are drawn as filled circles, virtual points as filled squares.

$$\begin{bmatrix}\mathbf{d}\end{bmatrix} = \begin{bmatrix}\mathbf{A}\end{bmatrix}\begin{bmatrix}\mathbf{p}\end{bmatrix} \rightarrow \begin{bmatrix}\mathbf{d}_1\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1\end{bmatrix}\begin{bmatrix}\mathbf{p}_1\end{bmatrix} + \begin{bmatrix}\mathbf{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{p}_2\end{bmatrix}$$

Figure 6. Matrices d, A and p are partitioned to speed up the simulation.

modified version of the radial-edge data structure [13, 6]. Given a point, edge, face or cell, this data structure can help one determine which components comprise which elements, which elements are adjacent to which elements, etc. This functionality is achieved through the use of numerous arrays of pointers.

5.4. Sculpting tools

Figures 7–10 demonstrate applications of some of the sculpting tools in our system. The main sculpting tool in our system is a non-compressible *rope tool* that permits the user to interactively deform the solid. Through the use of a 3D input device, the user can select any vertex of the discretized lattice and drag it along any direction in three dimensions. Forces acting on the given mass point result in a solid deformation that behaves in a physically plausible manner. We undergo a simple linear search to determine which point in the lattice is the closest to the current location of the 3D cursor.

We have implemented an extrusion tool for the Catmull-Clark solid that lets the user grow protrusions from the existing solid. After using the input device to select a face

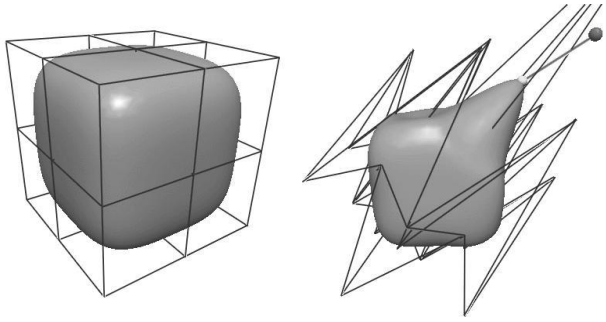

Figure 7. Original cube-like object before and after deformation by the rope tool.

on the surface of the object, the user presses a button in the GUI, and the system creates a new cell in the control lattice and affixes it to the highlighted face. The size of the new cell is calculated based on the information about the existing cell to which it is attached, and it grows along the normal direction of the selected face.

Trimming is supported for the subdivision solid by allowing the user to select any cell in the control lattice and to subsequently remove it. This operation, as well as the extrusion process, entails reconstructing most of the data structures that represent both the geometry and physical properties of the object. The topology, on the other hand, can be updated relatively easily because of the radial-edge data structure.

We have also implemented a useful and simple feature that permits the user to instruct the system to reset all of the rest lengths of the springs to their current lengths. This has the effect of re-defining the rest state of the system to its current state. In this way we can deform an elastic object but force it to take on a new rest shape at any time during the simulation. Essentially, this causes a normally elastic object to become temporarily plastic for the purpose of sculpting. Note that the rest lengths of the virtual springs must also be updated to their current lengths.

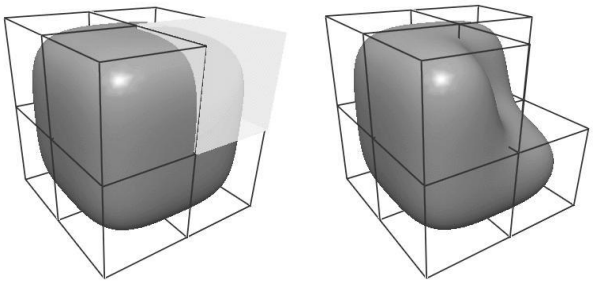

Figure 8. The user highlights and then deletes a cell of the control lattice.

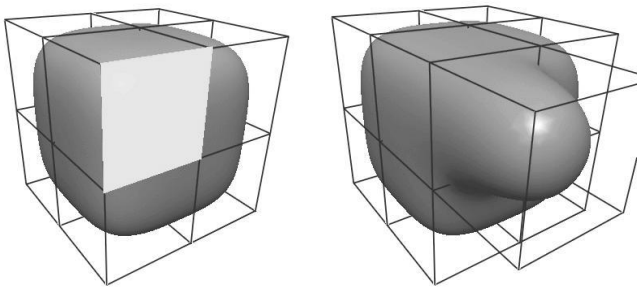

Figure 9. The user highlights a face and then extrudes material.

Control points can be fixed at run-time by moving the cursor to the desired point and pressing a button in the GUI. The matrices and other data structures are re-assembled on the fly. Other parameters, such as spring stiffness and damping factor, can be modified at run-time through the GUI.

5.5. Results and time performance

We now document various solids sculpted in our system and report their time performances in Table 1. The test platform was a Microsoft Windows NT PC with a single Intel Pentium III 550 MHz CPU and 512 megabytes RAM. As one would expect, the update time grows linearly with an increase in the number of control points since a mass-spring lattice lies at the core of the physical model.

6. Conclusions

We have presented a novel dynamic framework for deforming and sculpting free-form solid objects. We have devised a dynamic algorithm and implemented a system that permits physically plausible deformation and manipulation of *virtual clay* expressed by Catmull-Clark subdivision solids. Using the dynamic modeling approach, graphic

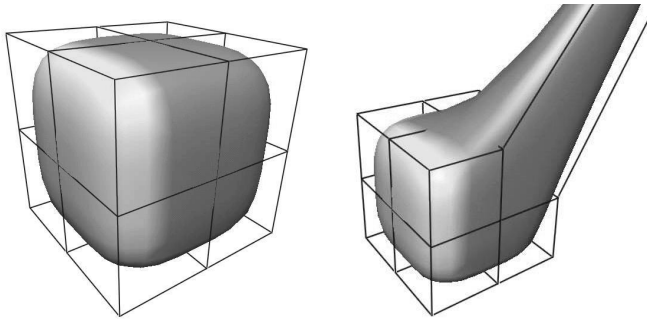

Figure 10. The user fixes the left side and underside of the solid and then deforms it.

Model Name	Sub. level	Initial Points	Initial Edges	Initial Faces	Initial Cells	Data Points	Data Edges	Data Faces	Data Cells	Update Time (ms)
Cube (3^3)	2	27	54	36	8	729	1944	1728	512	30
Cube (3^3)	3	27	54	36	8	4913	13872	13056	4096	189
Cube (4^3)	1	64	144	108	27	343	882	756	216	20
Cube (4^3)	2	64	144	108	27	2197	6084	5616	1728	121
Torus	2	24	52	36	8	680	1784	1552	448	28
Torus	3	24	52	36	8	4464	12464	11584	3584	163
Gear	2	56	116	76	16	1480	3864	3344	960	71
Tetrahedron w/ holes	3	16	42	22	1	2505	6808	6048	1744	87
Cube w/ holes	2	64	144	96	20	1900	5040	4416	1280	103
Soccer player	2	104	200	122	24	2450	6320	5408	1536	151
Cactus	2	108	212	131	26	2625	6800	5840	1664	169
Cylinder	3	15	34	28	8	3553	10144	9664	3072	110

Table 1. Various subdivision solids sculpted in our system, their sizes, and their time performance. "Initial points", "initial edges", etc. refer to the number of geometric elements in the control lattice; "data points", "data edges", etc. refer to the number of elements in the finest subdivided lattice.

modelers can naturally enforce various functional and aesthetic requirements on a free-form solid without the need to explicitly manipulate the control vertices. Our formulations, algorithms, and system techniques are very general in the sense that they can be applied to arbitrary parametric or subdivision solids in a hierarchical and adaptive fashion. We envision several generalizations resulting from our current results in the near future, including extensions to other subdivision schemes and to other physical domains, such as fluid dynamics and heat transfer.

Acknowledgments

The authors wish to thank Frank Dachille IX and Jihad El-Sana for providing some of the code used in implementing this system. This research was supported in part by the NSF CAREER award CCR-9896123, the NSF grant DMI-9896170, the GAANN grant P200A97030199 and a research grant from Ford Motor Company.

References

[1] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topology. *Computer-Aided Design*, 10(6):350–355, November 1978.

[2] F. Dachille IX, H. Qin, A. Kaufman, and J. El-Sana. Haptic sculpting of dynamic surfaces. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, pages 103–110, 1999.

[3] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, pages 181–188, August 1996.

[4] C. Mandal, H. Qin, and B. C. Vemuri. A novel FEM-based dynamic framework for subdivision surfaces. In *Proceedings of the Fifth Symposium on Solid Modeling*, pages 191–202, 1999.

[5] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. In *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, pages 309–312, July 1992.

[6] M. J. Muuss and L. A. Butler. Combinatorial solid geometry, boundary representations, and n-manifold geometry. In D. F. Rogers and R. A. Earnshaw, editors, *State of the Art in Computer Graphics: Visualization and Modeling*, pages 185–223. Springer-Verlag, 1991.

[7] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, pages 215–222, 1989.

[8] H. Qin, C. Mandal, and B. C. Vemuri. Dynamic Catmull-Clark subdivision surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215–229, July 1998.

[9] H. Qin and D. Terzopoulos. D-NURBS: a physics-based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, Mar. 1996.

[10] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, Dec. 1988.

[11] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH*, July 1987.

[12] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994.

[13] K. J. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Institute, August 1986.