Available online at www.sciencedirect.com
SCIENCE DIRECT®

Graphical Models 67 (2005) 347–369

Graphical Models

www.elsevier.com/locate/gmod

DigitalSculpture: a subdivision-based approach to interactive implicit surface modeling

Kevin T. McDonnell[*], Yu-Sung Chang, Hong Qin

Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-4400, USA

Received 30 June 2004; received in revised form 1 November 2004; accepted 13 November 2004
Available online 22 January 2005

Abstract

This paper presents *DigitalSculpture*, an interactive sculpting framework founded upon iso-surfaces extracted from recursively subdivided, 3D irregular grids. Our unique implicit surface model arises from an interpolatory, volumetric subdivision scheme that is C^1 continuous across the domains defined by arbitrary 3D irregular grids. We assign scalar coefficients and color to each control vertex and allow these quantities to participate in the volumetric subdivision of irregular grids. In the subdivision limit, a virtual sculpture is obtained by extracting the zero-level from the volumetric, scalar field defined over the irregular grid. This novel shape geometry extends concepts from solid modeling, recursive subdivision, and implicit surfaces; facilitates many techniques for interactive sculpting; permits rapid, local evaluation of iso-surfaces; and affords level-of-detail control of the sculpted surfaces.
© 2004 Elsevier Inc. All rights reserved.

Keywords: Implicit surfaces; Interactive sculpting; Subdivision algorithms; Solid modeling

1. Introduction and motivation

This paper presents a new implicit surface modeling technique founded upon *implicit subdivision solids*, which define scalar fields over 3D irregular grids. We have

[*] Corresponding author. Present address: Department of Mathematics and Computer Science, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA. Fax: +1 631 244 1033.
E-mail address: mcdonnek@dowling.edu (K.T. McDonnell).

1524-0703/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2004.11.001

developed an accompanying sculpting system called *DigitalSculpture* that employs implicit subdivision solids as the underlying shape primitive. In our new shape design approach, the boundary of an object is given by a level surface of a 3D function defined over an arbitrary, volumetric region (spanned by hexahedral meshes) of complex topology. The space is uniquely defined by our new interpolatory, solid subdivision algorithm [21]. (By *interpolatory* we mean that vertices are preserved or *interpolated* between subdivision levels.) Our subdivision scheme is founded upon the cubic, Lagrange interpolating polynomial and can be applied over arbitrary hexahedral meshes. The scalar field from which the iso-surface arises is constructed by first assigning a scalar coefficient to each vertex of the control mesh. Then the subdivision algorithm refines both the positions and the scalar coefficients of the control vertices with the same set of rules. After subdivision of these *control coefficients*, an iso-surface can be extracted from the subdivided scalar field. The surface can be modified by directly changing the control coefficients.

Our approach exhibits several capabilities in a *single, integrated framework* which, to our best knowledge, cannot be accommodated by other existing approaches. First, our framework supports sculpting of implicit surfaces over non-rectilinear working spaces (grids). Such flexibility can help to address the well-known blending problem as well as provide a control lattice appropriate for free-form deformation [35]. Second, blending is guaranteed to be at least C^1 throughout the volumetric domain, thereby permitting the definition of smooth shapes. Finer features like corners and cusps can be approximated by increasing the grid resolution. Third, our approach exhibits many of the desirable properties of subdivision-based schemes. For instance, the subdivision algorithm's continuity guarantees smoothness even when the working space is deformed, which can be manipulated in a global or local fashion using FFD-like techniques. It also affords local control, straightforward geometric manipulation, topological flexibility, and level-of-detail control. Fourth, our framework is very general and can be extended to other volumetric subdivision algorithms [2,6–8] or even volumetric spline-based approaches. It can be extended to accommodate newer volumetric models as they appear and also to sophisticated parametric approaches, such as simplex splines [16]. Last, our approach is not limited only to the representation of geometric quantities like position. Indeed, the *DigitalSculpture* framework can be generalized to support the specification of material and physical attributes [27,22] and thereby broaden the applications of our work even further.

2. Related work

This work is strongly motivated by a number of disparate areas of research, including volumetric sculpting [1,30,39], implicit and functional modeling [4,29], and volumetric subdivision schemes [2,6,20]. To our best knowledge, no one has yet proposed the idea of defining an implicit surface as the iso-level of a smooth scalar field defined by recursive *volumetric* subdivision algorithms. Cani and Hornus [5] proposed the idea of defining an implicit surface by a skeleton of subdivision curves. This reliance on skeletons limits the ranges of definable shapes and the types of interaction that can be used to manipulate such objects. Dewaele and Cani [11] present an interesting implicit surface sculpting system that employs a pseudo-physical model to simulate deformable materials. Their approach is best suited for medium-scale and large-scale deformations and cannot represent the kinds of detail our framework can. Museth et al. [24] recently proposed a framework for editing implicit surfaces based on the traditional level-set method. Their model is particularly well-suited for performing large-scale operations on scanned models, but its computational expense precludes its use in interactive sculpting. Turk et al. [37] present a framework for modeling implicit surfaces that interpolate a given set of constraints. They employ a global solution scheme that can accommodate at most a few thousand coefficients. Our work, in contrast, can support working spaces consisting of over 10^5 control vertices. Reuter et al. [32] employ point-based modeling and radial basis functions to represent and render implicit surfaces. As in Turks et al's approach, however, their method can accommodate a limited amount of detail since at most a few thousand coefficients can be employed in real-time. Pasko et al. [28], Wyvill et al. [41], and works by later researches fuse the concepts of boolean operators and blending functions into a single model. We present a subdivision-based alternative that supports similar warping, blending, and boolean operations in a single, subdivision-based framework. Ferley et al. [13,14] developed a multiresolution framework for volume sculpting of scalar fields. Their approach requires a rectilinear grid, however, and cannot accommodate the large deformations supported by our model.

Unlike commonly used spatial decomposition algorithms such as quadtree or octree construction, our volumetric subdivision scheme generalizes popular subdivision surfaces to the construction of solid or volumetric models. That is, our volumetric subdivision scheme, in the mathematical limit of the recursive application of our subdivision rules over control meshes, unambiguously defines a smooth, solid object with provable continuity and geometric properties. The volumetric model that most closely resembles our approach is the scalar B-spline approach investigated by Raviv and Elber [31], Schmitt et al. [33,34], and Hua and Qin [17]. In this approach, a smooth, volumetric, scalar field is defined by continuously blending a set of scalar coefficients with B-spline basis functions. An iso-surface extracted from the field defines the virtual sculpture. Usually, these scalar fields are topologically rectangular, employ computationally expensive polynomials, and are modified *indirectly* by the user. Our work does not suffer from these shortcomings because we employ a simple, yet powerful, direct approach founded upon recursive subdivision solid schemes. The spaces in which our implicit surfaces are defined can have any geometric and topological shape, can be modified directly, and can be sampled at any global level of refinement.

Our *DigitalSculpture* system is also related to our earlier *Virtual Clay* sculpting system [23]. A key distinction is that the *Virtual Clay* system defines solid objects with the subdivision solid geometry itself. Also, the earlier model is physically based and cannot change topology very easily. Furthermore, sculptures created in the *Virtual Clay* modeling system cannot represent details without extensive adaptive refinement. In contrast, our newer modeling approach uses subdivision solids only as a convenient mechanism for defining a volumetric space. The sculpted models themselves are actually implicit surfaces defined over that space. Compared to *Virtual Clay* models, objects sculpted in our *DigitalSculpture* system can undergo topological changes more easily, can represent fine features and details more readily, and can exhibit a much wider range of shapes.

3. Subdivision of irregular grids

This section reviews volumetric subdivision algorithms and briefly describes our new scheme for volumetric, interpolatory subdivision solids. The details of the subdivision algorithm, including its derivation, can be found in [21].

3.1. Background of subdivision solid schemes

Subdivision solids have recently emerged as a new solid modeling approach and interactive deformation technique. In comparison with established modeling techniques associated with subdivision surfaces, subdivision solid formulations transcend surface-based approaches by defining geometry and topology both in the interior and on the boundary of solid objects. To our best knowledge, all existing subdivision solid schemes but two are approximating in nature. One of the exceptions is the interpolatory algorithm by Pascucci and Bajaj [26], which is a tensor-product generalization of Dyn et al's four-point scheme [12] to rectilinear, volumetric grids. The other scheme is an algorithm we recently published [7] for recursive subdivision of meshes organized around octet-truss structures. With an approximating subdivision scheme, each application of the subdivision algorithm causes the geometry to "shrink" away from the initial control mesh.

The first documented volumetric subdivision algorithm, that of MacCracken and Joy [20], generalizes cubic B-spline solids to meshes of arbitrary topology. Bajaj et al. [2] have proposed an alternate 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 C^1 subdivision solid scheme based on box splines that must be applied over hybrid tetrahedral/octahedral meshes. Other recent work includes the investigation of wavelet decompositions of subdivi-

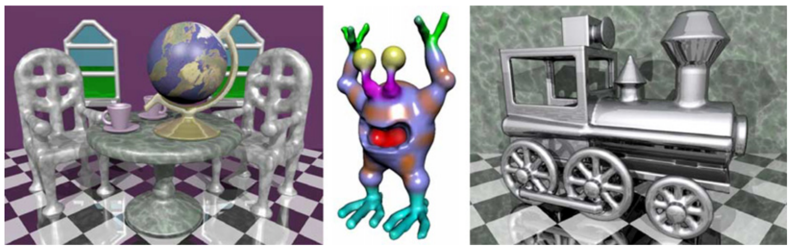
Fig. 1. Virtual sculptures created in our *DigitalSculpture* modeling environment.

sion volumes [3], hierarchical representation of time-varying data [19], and physically based animation and volumetric sculpting [23].

3.2. Our volumetric subdivision scheme

We employ a new subdivision scheme we recently developed in order to define volumetric scalar fields over arbitrary 3D grids. In the interest of brevity, we provide only an overview of the scheme and provide the subdivision masks in Figs. 2 and 3 without derivation. (The interested reader is encouraged to see [21], which discusses the algorithm in detail and provides its complete derivation.) Then we will describe how the scheme is used to define smooth, volumetric scalar fields.

Our interpolatory, volumetric subdivision algorithm is derived from a generalization of the tri-cubic, interpolating Lagrange polynomial. As we shall see later, the interpolatory nature of our algorithm is critical for defining implicit surfaces in our framework. Given a control mesh consisting of hexahedral cells, quadrilateral faces, edges, and vertices, the algorithm recursively subdivides the space enclosed by the mesh. The algorithm is designed to represent cubic polynomials in the regular case while reproducing C^1 continuous volumetric spaces under most non-regular

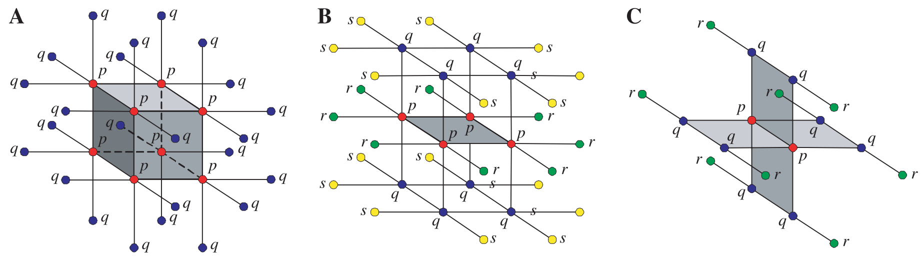

Fig. 2. Subdivision masks for the scheme over regular topology. (A) Cell-point weights: $w_p = (6w + 1)/8$, $w_q = -(w/4)$. (B) Face-point weights: $w_p = (2w + 1)/4$, $w_q = w/4$, $w_r = -w/4$, $w_s = -(w/8)$. (C) Edge-point weights: $w_p = 1/2$, $w_q = w/4$, $w_r = -(w/4)$.

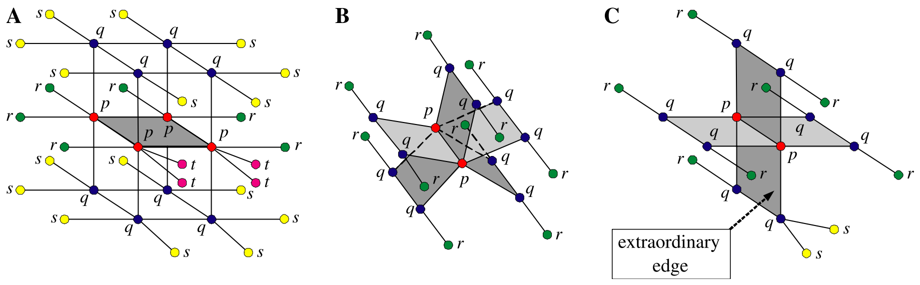

Fig. 3. Subdivision masks for the scheme over non-regular topology. (A) Face-point weights: $w_p = (2w + 1)/4$, $w_q = w/4$, $w_r = -(w/4)$, $w_s = -(w/8)$, $w_t = -(w/8)$. (B) Edge-point weights for extraordinary edge: $w_p = 1/2$, $w_q = w/5$, $w_r = -(w/5)$. (C) Edge-point weights for edge with an *adjacent* extraordinary edge: $w_p = 1/2$, $w_q = w/4$, $w_r = -(w/4)$, $w_s = -(w/8)$.

topological conditions. (Non-regular topology meshes are those containing vertices of valence not equal to six.) A subdivision weight, which can be specified to meet design constraints, is similar to the tension parameter of Butterfly subdivision surfaces. The standard weight is 1/16, which can be obtained from the derivation of the regular case of the algorithm and which we use in the *DigitalSculpture* system.

The algorithm takes as input and produces as output a hexahedral mesh. Tetrahedral meshes can be converted into hexahedra with a Catmull–Clark-like split [2]. In fact, any mesh containing cells that have vertices of valence three can be converted into a hexahedral mesh with this technique. The number of hexahedra created from such a cell is equal to its number of vertices.

3.3. Matrix representation

Like all procedural subdivision algorithms, our subdivision solid can be expressed as a global matrix multiplication:

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

where $\mathbf{p}$ is a column vector of the positions of the control points; the matrix $\mathbf{A}$ is a sparse, rectangular, global subdivision matrix that contains weights given by the subdivision rules; and the column vector $\mathbf{d}$ gives the positions of the subdivided mesh. (We will refer to such points as *subdivided points*.) Hence, whenever the control geometry changes, it is easy to reevaluate the subdivided geometry simply by performing a matrix multiplication. Note that matrix $\mathbf{A}$ is a global subdivision matrix, and that local changes to the subdivision rules can be expressed by changing a few rows. In practice, however, we typically forego assembly of the matrix $\mathbf{A}$ and instead employ local subdivision matrices.

4. From subdivision solids to implicit surfaces

In our framework, an implicit subdivision solid is defined by adding a fourth coordinate, a scalar coefficient, to the control vertices of a subdivision solid mesh. We refer to the control mesh and its corresponding scalar coefficients (control coefficients) as the *working space* in this paper, since virtual sculptures are created over this space. The scalar coefficients are subdivided with the same rules as the geometry to produce a smooth volumetric field. The strong relationship between the subdivision volume algorithm and underlying iso-surfaces can be seen clearly in Fig. 4. The train sculpture—defined implicitly by the scalar values stored inside the solid—deforms as a result of geometric deformation of the control mesh. This interplay between subdivision solid geometry and a corresponding scalar field forms the fundamental representation used in this work. The iso-surfaces in the figure were evaluated by first subdividing the control mesh (i.e., working space) once, and by then polygonizing the iso-level seen in the figure. The zero-level lies within the green region of the scalar field and is the iso-level we use in our *DigitalSculpture* to represent virtual sculptures. Since our sculpting tools have local support, regions of the

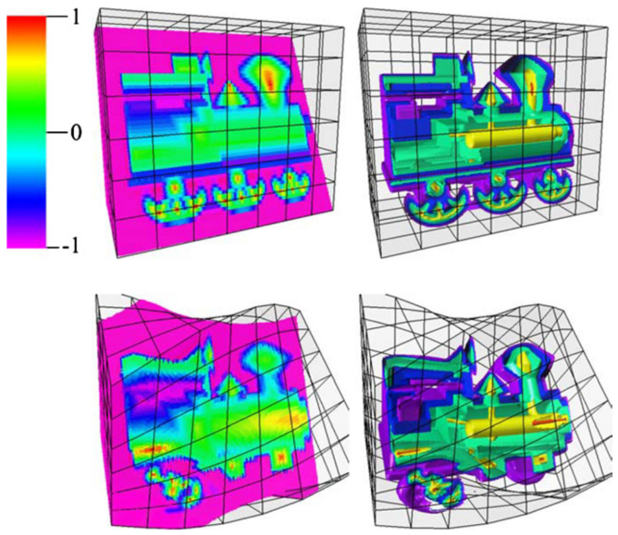

Fig. 4. A train model, defined as the zero-level of a scalar field associated with the subdivision geometry, deforms as a consequence of working space deformation. First row: undeformed working space; second row: deformed working space. First column: subdivided coefficients; second column: cut-away views of level sets $\varphi \in [-0.8, -0.5, 0.0, 0.6, 0.9]$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

working space that do not affect the zero-level are assigned -1 and are shaded purple in the figure.

The global subdivision operation can now be rewritten as

$$\hat{\mathbf{d}} = \widehat{\mathbf{A}}\hat{\mathbf{p}}, \tag{2}$$

where $\hat{\mathbf{d}}$ and $\hat{\mathbf{p}}$ include the (x, y, z) components of $\mathbf{d}$ and $\mathbf{p}$, respectively, as well as a fourth component that represents the scalar coefficients. Note that the size and contents of $\mathbf{A}$ are adjusted to accommodate the scalar values appended to $\mathbf{d}$ and $\mathbf{p}$. We denote this modified subdivision matrix as $\widehat{\mathbf{A}}$. The vector $\hat{\mathbf{d}}$ can be considered an approximation of the continuous scalar field that would be obtained after an infinite number of subdivisions of the working space ($\hat{\mathbf{p}}$). Throughout the remainder of this discussion, we assume that all vertices (including both the control vertices and the subdivided vertices) are assigned scalar coefficients. The term *subdivided coefficients* will refer to the scalar values of the subdivided points (i.e., vertices in the subdivided mesh).

Now, a scalar field function is generally characterized by some (continuous) function $f(x, y, z)$. A surface defined by the function is a level-set:

$$\{(x, y, z) \mid f(x, y, z) = \varphi\}. \tag{3}$$

By collecting all the level sets whose values are greater (or less) than a given threshold, we define an implicit solid as

$$\{(x, y, z) \mid f(x, y, z) \leqslant \varphi\}. \tag{4}$$

By convention, we assume in this work that $\varphi = 0$, and that f is negative inside the solid and positive outside the solid.

Hence, the fundamental contribution of this work is a new way of defining the field function, $f(x,y,z)$, which we approximate with $\hat{\mathbf{p}}$ after several levels of subdivision:

$$f(x,y,z) \approx \hat{\mathbf{d}} = \widehat{\mathbf{A}}\hat{\mathbf{p}}. \tag{5}$$

Rather than represent the scalar field as a collection of algebraic functions, we define the scalar field as a collection of coefficients obtained via volumetric subdivision of a smaller set of coefficients. Since the subdivision algorithm has provable smoothness properties, the implicit surfaces are guaranteed to be at least C^1 continuous for most topological configurations. (The specification of C^0 features would require a wavelet-based decomposition [3], which we are presently investigating for future incorporation into *DigitalSculpture*.) The surfaces are modified *directly* by changing the scalar values stored at the control vertices. In contrast, traditional implicit surface representations are often modified by changing the polynomial coefficients, which usually have no intuitive or geometric meaning. The user modifies the scalar field through a graphical interface and sculpting tools.

In practice, it can be very computationally expensive to evaluate Eq. 5 over the entire domain, even when $\widehat{\mathbf{A}}$ is stored in sparse matrix format. Therefore, we instead compute $\hat{\mathbf{d}}$ on a hexahedron-by-hexahedron basis. If the same subdivision rules are employed throughout the entire 3D space and the control mesh is regular, a single, local subdivision matrix can be used:

$$\hat{\mathbf{d}}_i = \mathbf{S}\hat{\mathbf{p}}_i, \tag{6}$$

where $\hat{\mathbf{d}}_i$ represents a local block of the subdivided points and subdivided coefficients, $\hat{\mathbf{p}}_i$ represents the $4 \times 4 \times 4$ group of control points and coefficients that influence $\hat{\mathbf{d}}_i$, and $\mathbf{S}$ is the local subdivision matrix itself. Each control cell of eight vertices will generate $(2\ell + 1)^3$ subdivided vertices, where ℓ is the subdivision level. Because our subdivision algorithm has local support, the subdivision matrix has dimension $(2\ell + 1)^3 \times 56$. (The 8 corner vertices of a $4 \times 4 \times 4$ block of vertices are not required.) For non-regular topological settings, several local subdivision matrices must be maintained. Since this can become very complicated to implement, our system instead uses global subdivision matrices for non-regular topologies.

Cells on the boundary of the control mesh may not be subdivided by the standard rules since not enough vertices are present to define the subdivision masks. In these regions we simply use linear interpolation to define the edge-points, face-points, and cell-points. In such cases, the system employs a local subdivision matrix of size $(2\ell + 1)^3 \times 8$, and we write

$$\hat{\mathbf{d}}_i = \mathbf{L}\hat{\mathbf{p}}_i, \tag{7}$$

where $\hat{\mathbf{d}}_i$, $\hat{\mathbf{p}}_i$, and $\mathbf{L}$ have definitions similar to the corresponding terms in Eq. (6). In this case, $\hat{\mathbf{p}}_i$ consists of a smaller $2 \times 2 \times 2$ block of control points.

The subdivided points and scalar coefficients can now be written as a collection of local computations:

$$\hat{\mathbf{d}} = \sum_{i=1}^{N_s} \mathbf{C}_i \mathbf{S} \hat{\mathbf{p}}_i + \sum_{i=1}^{N_l} \mathbf{C}_i \mathbf{L} \hat{\mathbf{p}}_i, \tag{8}$$

where N_s and N_l are the numbers of interior and boundary cells, respectively. The equation sums over all local $4 \times 4 \times 4$ and $2 \times 2 \times 2$ patches. The $\hat{\mathbf{p}}_i$ represent subsets of the global $\hat{\mathbf{p}}$ vector. The $\mathbf{C}_i$ are *selection matrices* that ensure that the contributions of the control coefficients are properly added to the subdivided coefficients. Based on the topology of the control mesh, they select which control coefficients affect the subdivided coefficients in a local region. That is, an entry $\mathbf{C}_i(j,k)$ is 1 if control point p_k of block $\hat{\mathbf{p}}_i$ influences the subdivided point d_j, and is 0, otherwise. The structure of each $\mathbf{C}_i$ varies as a function of both topology and vertex ordering. One selection matrix is needed for each local block of control coefficients; different matrices are needed for the linear subdivisions performed along the boundary. These matrices are computed directly from the edge connectivity of the control mesh. Under a regular topological setting, it is possible to avoid their assembly if the vertices are stored in a predetermined order. In the above notation we assume the topology is regular. As mentioned earlier, for non-regular topology, a different $\mathbf{S}$ matrix would be needed for each distinct topological configuration. The $\hat{\mathbf{p}}_i$ would also necessarily have different sizes.

By employing local subdivision matrices, the system is able to subdivide only those control coefficients whose values are changed as the result of user interaction. This has large ramifications for a potential sculpting system, since many operations can be localized. Since the subdivided points are influenced by a small neighborhood of control coefficients, it is necessary to recompute all subdivided points affected by a change in a single control coefficient. Fortunately, since the refinement masks overlap considerably, the cost of recomputing a subdivided point is, in large part, shared by its neighbors.

5. Interactive sculpting of implicit surfaces

5.1. Direct editing and modification

Any sculpting system founded upon our subdivision-based, implicit surface approach must allow modification of the control coefficients that define the scalar field. Since our algorithm is interpolatory in nature, scalar coefficients may be assigned values directly. This is a critical aspect of the subdivision scheme. If the algorithm were approximating in nature, the inverse or pseudo-inverse of the subdivision matrix would be required to create a specified shape. For a real-time application like virtual sculpting, it would be impractical to employ the inverse of the global subdivision matrix, $\widehat{\mathbf{A}}$. Local inverse matrices could, in theory, be used, but it is unclear how large a subset of the control mesh would have to be used. More seriously, such "macro-patches" of control vertices would necessarily destroy the smoothness properties of the algorithm. Their use would also complicate the implementation of the sculpting system.

Since we have the freedom to manipulate the control coefficients directly, it remains to define what functionality is desirable. In the computer graphics literature there exist many ways in which users interact with virtual models. For instance, constructive solid geometry (CSG) provides a very intuitive mechanism for defining certain kinds of shapes, especially mechanical objects and other models that can be described as the addition and subtraction of solid primitives. Feature-based design of this sort can provide significant functionality for a modeling system and expedite the design process. In light of the desirability and popularity of CSG, constructive solid geometry operators are supported in our sculpting system and are represented as volumetric field functions. They can be expressed as

$$\hat{\mathbf{d}} = \bigoplus_{i=1}^{N_s} \mathbf{C}_i\mathbf{S}\hat{\mathbf{p}}_i + \bigoplus_{i=1}^{N_l} \mathbf{C}_i\mathbf{L}\hat{\mathbf{p}}_i, \tag{9}$$

where $\oplus$ is a Boolean operator. Hence, a CSG operator is defined simply as a direct modification of the control coefficients followed by subdivision. The iso-levels are then updated implicitly as a result of modifying the control coefficients. Several examples can be seen in Fig. 5. "Copy-and-paste" functionality (Fig. 6) can be implemented in a similar fashion to CSG union. Given a 3D region of scalar values stored in a volumetric "clipboard," CSG union may be used to update the working space:

$$\hat{\mathbf{d}} = \bigcup_{i=1}^{N_s} \mathbf{C}_i\mathbf{S}\mathbf{T}\hat{\mathbf{r}}_i + \bigcup_{i=1}^{N_l} \mathbf{C}_i\mathbf{L}\mathbf{T}\hat{\mathbf{r}}_i, \tag{10}$$

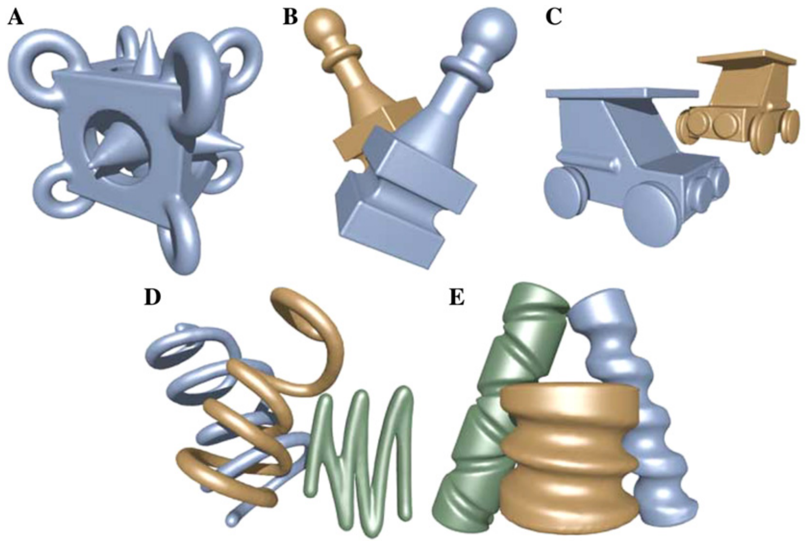

Fig. 5. (A–C) Surfaces created with CSG operators in our implicit subdivision solid approach. (D and E) Objects created by curve sweeping—addition and subtraction of material, respectively.

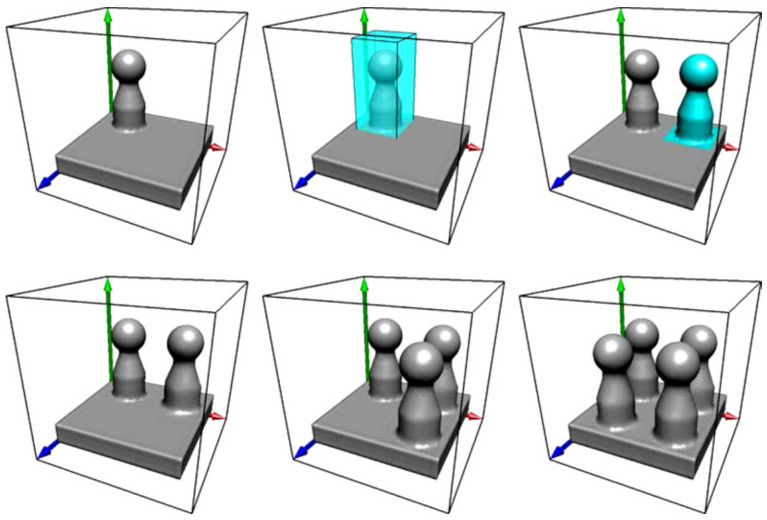
Fig. 6. Copy-and-paste functionality.

where ∪ indicates union and **T** is a transformation operator that translates the scalar components of the clipboard, $\hat{\mathbf{r}}$, to the desired position in the working space. The $\hat{\mathbf{r}}_i$ vector represents a local $4 \times 4 \times 4$ subset of the clipboard's control coefficients. Resampling is necessary if the coordinate positions of the coefficients in the clipboard and those of the working space do not coincide. This can cause details to be lost if the underlying grids are not of the resolution required to preserve desired features. Such problems are inherent in sampled representations such as ours, and can be avoided only by using denser grids or multiresolution approaches. During the paste operation, only the scalar coefficients are modified. The positions of the control coefficients remain unchanged. The coefficients lying on the boundary of the clipboard region are also filtered in order to avoid aliasing.

DigitalSculpture also features a spray painting tool (Fig. 7) to add color to sculptures. Unlike typical spray paint, however, our virtual paint covers a volumetric region of space. In this manner the user actually defines a 3D texture over the entire working space. Each control coefficient is assigned an (r, g, b) triple to store the red, green, and blue color components at that position. The colors are subdivided with the same subdivision rules as the control vertices and control coefficients. During polygonization of the iso-surface, the colors are linearly interpolated across edges and are assigned to triangle vertices. Typically, models are painted after all other sculpting operations have completed, although this is not a requirement.

The spray painting tool in *DigitalSculpture* is a special case of what could be termed "material editing." Material properties need not be limited to position, color, or scalar value. Physical properties such as mass, damping, stiffness, pressure, temperature, and other quantities could also be associated with and computed by the subdivision algorithm. Such a physical description would support physically based

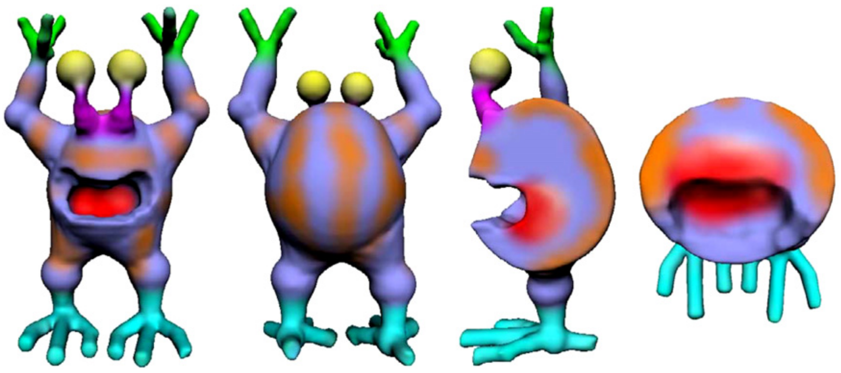

Fig. 7. Spray-painted alien model showing both the colored boundary surface and the interior regions. Cutting away a portion of the sculpture reveals that the texture is actually solid in nature.

animation, force-based interaction, dynamic modeling, finite element modeling, etc. These ideas are currently under investigation.

5.2. Local deformations

Local deformations may be applied to implicit surfaces by slowly modulating the scalar control coefficients according to a given function. We use what is essentially a combination of dilations and erosions to achieve rapid, local deformations. Such interaction supports the sculpting of organic-looking shapes, similar to those created with the Teddy design system [18]. A scalar factor can be specified that controls how quickly the regions of the space are modified by the function. For instance, our inflation tool, described below, slowly increases the scalar values in a local region. A modulation of the scalar field can be expressed as

$$\hat{\mathbf{d}} = \bigcup_{i=1}^{N_s} \rho \mathbf{C}_i \mathbf{S} \hat{\mathbf{p}}_i + \bigcup_{i=1}^{N_l} \rho \mathbf{C}_i \mathbf{L} \hat{\mathbf{p}}_i, \tag{11}$$

where ρ is the rate of change. For our inflation tool, $0 < \rho \leqslant 1$, and for deflation, $-1 \leqslant \rho < 0$.

If repeated CSG operations cause aliasing to arise in the scalar field, it may be necessary to perform local smoothing of the working space. This can be achieved in a similar manner to the deformation tools by applying a filter of small support. In our system, our sanding tool achieves this goal by employing an approximation of a Gaussian filter. The control coefficients are filtered and then locally subdivided to obtain the new scalar field.

5.3. Large-scale deformations

In addition to local deformation, implicit surfaces defined with our approach can also undergo global and large-scale deformation, as shown in Figs. 8 and 9.

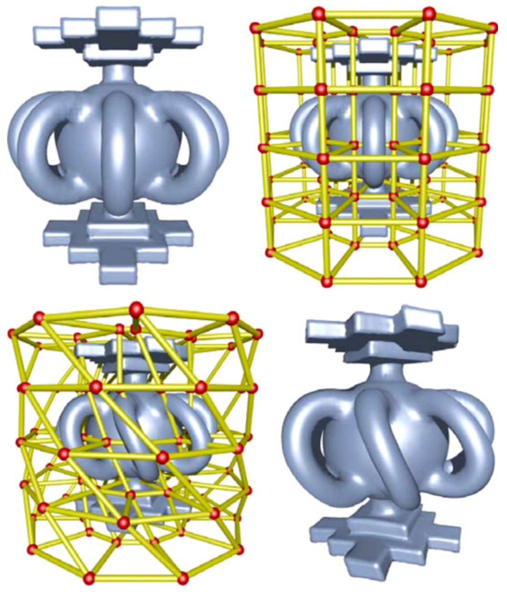
Fig. 8. The sculpture depicted in Fig. 5B undergoes FFD-based twisting with a non-regular, cylindrical subdivision mesh.

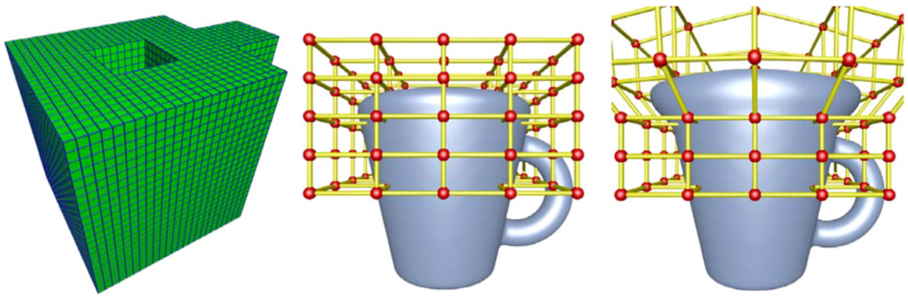
Fig. 9. The top portion of a mug sculpted over a non-regular working space is deformed with a local FFD mesh. Note that the mug's bottom portion remains unchanged.

However, whereas the local deformations are effected by modifying the *scalar values* of the control vertices, large-scale deformations are achieved by modifying the *positions* of the control vertices. Since the control mesh of the working space represents a 3D space, this kind of interaction is essentially a variant of free-form deformation (FFD) [35]. Our FFD technique is unusual, however, in that it essentially is part of the model. That is, our implicit subdivision solid approach supports free-form deformation without the use of a second representation, such as a Bézier or B-spline volume. It also permits the use of complicated FFD meshes without the need for patching operations, as is common with extended FFD, or E-FFD [9].

The user can achieve FFD effects either by deforming the working space directly, or by using an auxiliary subdivision solid mesh. Since the former approach can be very time-consuming, we generally employ the latter method instead and embed working spaces in simpler subdivision meshes containing fewer control points. In this manner we provide a unified framework used for defining, manipulating, and deforming virtual sculptures. The system parameterizes each control coefficient of the working space inside the continuous 3D space defined by the FFD mesh. The FFD mesh is first subdivided one or more times to approximate the continuous space covered by the mesh. When the user moves one of the FFD control points, the control coefficients of the working space move in response. After the deformed working space is re-subdivided by reevaluating Eq. (2) for the new value of $\hat{\mathbf{p}}$, the new iso-surface can be extracted.

For regular meshes, each hexahedron is divided into six tetrahedra. Control coefficients of the working space are then parameterized in the tetrahedra by their barycentric coordinates. For non-regular meshes, it may not be possible to refine each hexahedral cell into so few tetrahedra without causing degeneracies or intersecting triangles. To avoid these situations, we refine each cell in a non-regular, subdivided FFD mesh into 24 tetrahedra by splitting the quadrilateral faces into four triangles each. Then the control coefficients are parameterized as in the regular case of six tetrahedra. Our algorithm for supporting global and large-scale deformations is summarized in Algorithm Listing 1.

Algorithm 1. Our specialized FFD algorithm for deforming implicit subdivision solids.

```
1: The user moves the control points of the FFD mesh ($\mathbf{q}$).
2: The subdivision matrix ($\mathbf{F}$) of the FFD mesh is used to reposition the subdivided
   points in the FFD mesh ($\mathbf{e}$) through $\mathbf{e} = \mathbf{Fq}$.
3: The control coefficients of the working space ($\hat{\mathbf{p}}$) are repositioned based on their
   parameterization inside the subdivided FFD mesh.
4: The subdivided coefficients are repositioned by Eq. 2 ($\hat{\mathbf{d}} = \widehat{\mathbf{A}}\hat{\mathbf{p}}$).
5: The new iso-surface is extracted and displayed on the screen.
```

As Coquillart [9], and MacCracken and Joy [20] have clearly demonstrated, it is often the case that FFD of an object can be achieved more effectively by using a deformation mesh that is not topologically rectangular. By defining an FFD mesh that conforms to the overall shape of the deforming object, the user can enjoy more intuitive and flexible shape control. Since our interpolatory subdivision algorithm can support meshes of arbitrary topology, it can be employed for performing FFD over arbitrary 3D domains. We see an example in Fig. 8, for instance, in which a sculpture defined over a rectangular working space was deformed with a cylindrical subdivision mesh.

5.4. Tool representation

Most tools in the *DigitalSculpture* system are represented as field functions. Given the (x, y, z) location of a control coefficient, the shortest distance to the boundary of a tool is computed with an evaluation function. To avoid aliasing, each CSG tool features a transition region inside of which the tool linearly blends the control coefficients from -1 to $+1$. The maximal value is assigned to the center of the tool and is linearly blended outwards. In voxel-based representations, a transition region of 2.5 voxels is generally recommended to avoid artifacts [38]. Since our subdivided vertices sometimes are not at unit distance from each other, our transition region is typically set to $\lceil 2.5\Delta s \rceil$, where Δs is the average inter-vertex distance.

The local deformation tools are evaluated in a similar manner. The inflation and deflation tools affect regions of the same overall shapes as those of the CSG tools (e.g., sphere, cylinder, box, etc.) This simplifies the implementation by permitting extensive code reuse. The sanding tool essentially filters the control coefficients according to an approximation of the Gaussian distribution. The filter is discretized and has local support in order to avoid reevaluation of the entire scalar and to avoid a potentially expensive global subdivision.

5.5. Tool/sculpture interaction

All sculpting tools operate directly on the control vertices, either by changing their scalar coefficients or by modifying their positions. In this respect, several sculpting operations are implemented in a similar fashion to that of traditional voxel-based sculpting systems. The position, shape, orientation, and other parameters of the tool are specified by the user through the graphical interface. Once the tool has been activated and the control coefficients updated, the system first determines which portions of the subdivided coefficients need to be recomputed. This can be calculated based on the active sculpting tool and its parameters (e.g., radius, length, width, etc.). If a CSG or local deformation tool is applied, only a local region of the working space needs to be re-subdivided. If FFD is performed, most—if not all—of the working space may need to be re-subdivided, which is expensive if the control mesh is very large. Once the required portions of the subdivided coefficients have been updated, the system is then able to reevaluate the scalar function, extract an iso-surface, and update the display.

5.6. Iso-surface extraction

Once the subdivided scalar field is defined, it becomes possible to determine the geometry of the implicit surfaces. Since the mesh may be deformed arbitrarily, we employ the Marching Tetrahedra algorithm [36] to extract iso-surfaces. When the working space is topologically regular, cells in the subdivided space are divided into six tetrahedra. For non-regular working spaces, the hexahedra can be divided into 24 tetrahedra. For performance reasons, we make the simplifying assumption that the subdivided cells are not deformed by the user to such an extent that they become self-intersecting. In practice, very large deformations are required to cause self-intersec-

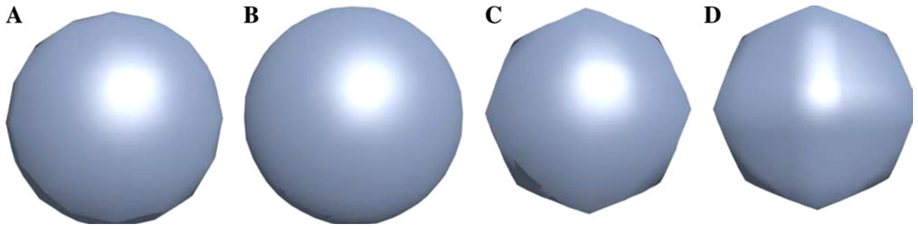

Fig. 10. Control coefficients are assigned values on a $20 \times 20 \times 20$ grid to represent a sphere of radius 15 and are then scaled by a factor of 10. (A and B) Our algorithm after 1 and 2 levels of subdivision, respectively. (C and D) Linear interpolation applied at the same resolution grids.

tions, because the movement of the refined mesh is constrained by the subdivision structure.

It is important to note that, based on the geometry of the control mesh, the subdivided vertices may be spaced at non-unit intervals or distributed very irregularly. In fact, there may be large gaps between adjacent vertices in the subdivided mesh. However, since we use a subdivision algorithm that produces a smooth scalar field—*regardless of its distribution*—we can always guarantee that the implicit surface will be smooth in the limit of subdivision. Moreover, our approach does not require resampling onto a regular voxel raster, which would introduce error. Although the process of iso-surface extraction does employ linear interpolation along tetrahedron edges, which has the potential to introduce sampling artifacts, we have found that two levels of volumetric subdivision suffice to approximate the smooth limit iso-surface and to avoid noticeable artifacts.

Since traditional voxel-based sculpting systems [1,30,39] employ tri-linear interpolation, they exhibit aliasing when the voxel grid is scaled or deformed. Aliasing must then be eliminated by filtering most or all of the voxel grids. We avoid this problem by using what is essentially a higher-order interpolation algorithm (i.e., C^1 subdivision). Surfaces represented in our implicit subdivision solid approach will remain smooth even if the control mesh is arbitrarily scaled or even deformed. Fig. 10 shows an example in which our algorithm produces a smooth surface even after scaling, whereas linear interpolation fails to improve surface quality even with denser grids.

5.6.1. Vertex normal estimation

A key aspect of iso-surface extraction is normal vector computation. With traditional polynomial-based implicit surfaces, it may be possible to compute normals analytically. In voxel-based approaches, the normalized gradient is usually employed as an approximation of the normal and can be computed by central differences. Since a subdivided mesh in our framework may exhibit non-regular geometry and topology, central differences cannot always be used. While local resampling and subsequent central difference computation could be employed, we have devised the following simpler method that works well in practice.

Suppose we are computing the gradient for a subdivided vertex d, located at (x, y, z), whose scalar coefficient is φ_d. Since we use the Marching Tetrahedra algorithm, we compute gradients only at subdivided vertices. Therefore, we do not need to estimate gradients at the subdivided cells, faces or edges. Let d_i indicate the position of a neighboring vertex and φ_i its corresponding scalar coefficient. The components of the (un-normalized) gradient $\nabla d = (g_x, g_y, g_z)$ at d can be approximated by

$$g_x = \sum_{i=1}^{N} (\varphi_d - \varphi_i) \frac{d_x - d_{i,x}}{(| d_x - d_{i,x} |)^n}, \tag{12}$$

$$g_y = \sum_{i=1}^{N} (\varphi_d - \varphi_i) \frac{d_y - d_{i,y}}{(| d_y - d_{i,y} |)^n}, \tag{13}$$

$$g_z = \sum_{i=1}^{N} (\varphi_d - \varphi_i) \frac{d_z - d_{i,z}}{(| d_z - d_{i,z} |)^n}, \tag{14}$$

where $n \geqslant 1$, N is the number of vertices adjacent to d, and d_x indicates the x component of d. When $n = 1$, this formulation constitutes a linear, radial-basis approach and provides a good estimate of the gradient at a point. It can be shown that for regularly distributed points, this formulation—after normalization—reproduces the central differences algorithm for computing gradients over rectilinear voxel grids.

During polygonization of the iso-surface, edges of the tetrahedra are split to create triangle vertices. The gradient of the vertex introduced at some position e along that edge can be approximated by

$$\nabla e = (1 - \alpha)\nabla d_0 + \alpha \nabla d_1, \tag{15}$$

where ∇d_0 and ∇d_1 indicate the respective gradients of the end-points, d_0 and d_1. The parameter α is also computed during polygonization and is given by

$$\alpha = \frac{\varphi - \varphi_0}{\varphi_1 - \varphi_0}, \tag{16}$$

for the iso-level φ. The values φ_0 and φ_1 indicate the scalar coefficients at the two end-points. The vertex's normal vector can then be approximated by $\nabla e/|\nabla e|$.

6. The *DigitalSculpture* environment

6.1. Sculpting functionality

We demonstrate the power of our new surface modeling approach by incorporating implicit subdivision solids into a new sculpting environment we have developed called *DigitalSculpture*. The sculpting system features traditional CSG sculpting tools that permit the addition and subtraction of material in the shape of a sphere, cylinder, rectangular box, torus, cylinder or cone. It also features local and large-scale deformation tools, a sanding tool, copy-and-paste functionality, a spray paint-

Table 1
Sculpting functionality offered in our *DigitalSculpture* environment

CSG tools:	Union and subtraction of spheres, boxes, cylinders, cones, tori
Local deformations:	Inflation, deflation, sanding
Large deformations:	FFD via control point manipulation
Other functionality:	Spray painting, curve sweeping, copy-and-paste, model import

ing tool, and functionality to import polygonal and volumetric data-sets. The capabilities of our system are summarized in Table 1. The user interacts with the system by employing a standard 2D mouse with the dominant hand and a 3D input device with the other. The latter is used to specify the (x, y, z) position of the sculpting tools. Tool parameters are specified in the GUI. Local subdivision and iso-surface extraction are performed at run-time.

6.2. Example digital sculptures

We have created numerous virtual sculptures with our new implicit solid sculpting environment. Several sophisticated examples can be found in Fig. 1 and throughout the remainder of the paper. Figs. 5A–C depict several objects modeled with only CSG operators. Examples of curve sweeping are given in Figs. 5D and E. To perform curve sweeping, the user draws a spatial curve and specifies a radius. The system samples the curve and adds a sphere of the given radius at each sample location. Fig. 11 shows example applications of the inflation and deflation tools. As can be seen in the figure, the deflation tool provides a convenient method for rounding, denting, and making bulges. The sanding tool, used extensively in the creature in Fig. 1, can be employed to smooth bumps and ridges. In Fig. 8 we performed FFD of an embedded sculpture by using a cylindrical control mesh. The top portion was given a 90° clockwise turn. The cylindrical mesh was used for FFD only, while the underlying object was defined over a regular, rectangular mesh. Fig. 9 shows a mug sculpted over a non-regular working space that contains a hole. The bottom of the mug was deformed with a localized FFD mesh. Fig. 12 shows several other models sculpted over irregular volumetric domains.

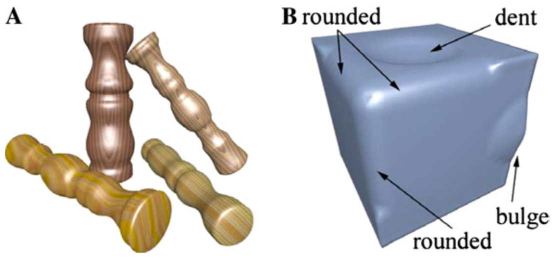

Fig. 11. Virtual sculptures created by the inflation and deflation tools. (A) Wood-turning simulated by deforming cylinders. Surface textures were applied afterwards. (B) A rectangular solid showing rounded edges, bulges, dents, and depressions created by the deformation tools.

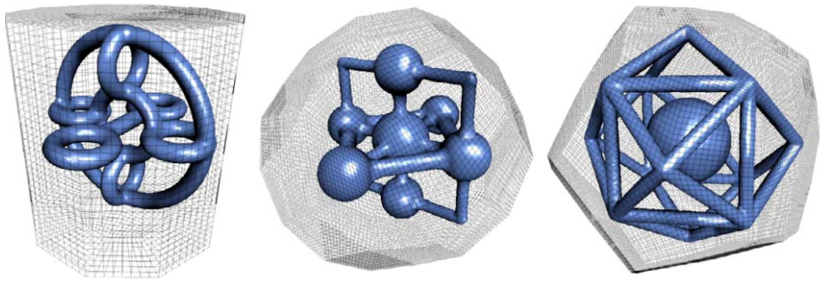
Fig. 12. Sculptures defined over irregular working spaces. Even the cylindrical working space on the left contains several extraordinary edges and vertices.

6.3. *Import of existing models*

Existing polygonal and voxel-based models can be imported into *DigitalSculpture* by converting them into distance fields. Some examples can be seen in Fig. 13. Polygonal meshes can be voxelized at a resolution in accordance with user requirement [10]. Voxelized data-sets may be resampled if they are too large to be feasibly stored in memory and manipulated in real-time. The voxel densities are rescaled and thereby treated as approximate signed distances. Note that we typically pad an imported model with two layers of voxels in order to ensure that there is sufficient support for the subdivision masks. The distance field must be scaled and truncated in order to bring each control coefficient into the range $[-1, +1]$. For more details about distance field construction from polygonal meshes see [15,38,42].

6.4. *Data structures*

For a topologically rectangular solid control mesh, all data can be stored in matrices and multi-dimensional arrays. The local subdivision matrices are represented

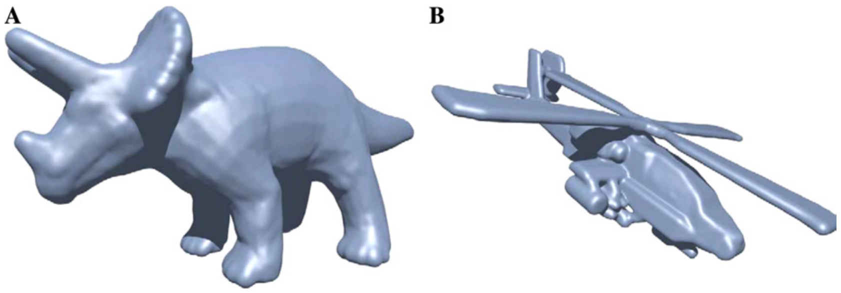

Fig. 13. (A) Polygonal mesh converted into a distance field and incorporated into our sculpting environment. (B) Voxelized Apache helicopter model imported into our system.

with a sparse matrix storage scheme to reduce subdivision time. When the control mesh is non-regular, we are required to store the connectivity information. For this purpose we employ a modified version of the radial-edge data structure [25,40], which is a generalization of winged-edge data structure to arbitrary manifolds. Our implementation consists of four lists to store the cells, faces, edges and vertices. Each topological entity (cell, face, edge, or vertex) contains several short lists that represent its local topological neighborhood. For instance, in our implementation, a face object consists of an ordered list of directed edges and a pair of pointers to the cells that share the face. Auxiliary information pertaining to the subdivision scheme is also stored, including a flag indicating whether an entity is in the interior or on the boundary, the coordinate positions of the vertices, the scalar coefficients, etc.

6.5. Hardware configuration

The *DigitalSculpture* environment features a dual-machine hardware configuration and uses message passing to facilitate inter-process communication. A photograph of the system in use is given in Fig. 14. A Haptics Server process manages a Sensable Technologies PHANToM and sends the (x, y, z) cursor position of the PHANToM to the Sculpting Station machine. The PHANToM is physically connected to the Haptics Server machine. The Sculpting Station acts as a client to the Haptics Server by polling the Server periodically for the 3D position of the PHANToM. The 3D cursor provided by the PHANToM is used to position the sculpting tools that can be activated by the user. Tool interaction, rendering, and all other functionality of *DigitalSculpture* run on the Sculpting Station, which, in our implementation, is a desktop PC with a 2.2 GHz CPU and 1.0 GB RAM. The Haptics Server and Sculpting Station software were implemented in C++. OpenGL is used

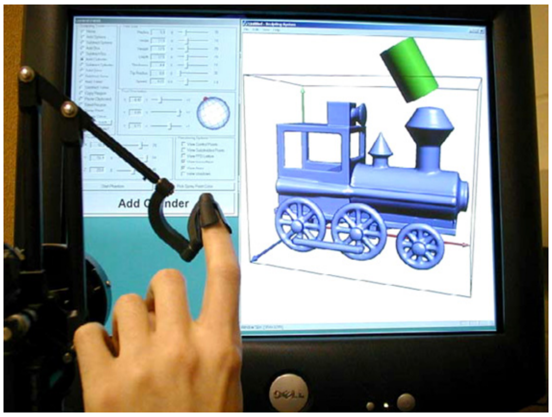

Fig. 14. The user interface of the *DigitalSculpture* system consists of a Sensable Technologies PHANToM, standard mouse, and on-screen GUI controls.

for run-time rendering virtual sculptures. POV-Ray (www.povray.org) was used for the most of the high-quality renderings shown in the paper.

7. Conclusions and future work

We have developed a new model for representing, modifying, and interacting with implicit surfaces. Our approach is to employ *implicit subdivision solids* over arbitrary 3D irregular grids, a new framework which uniquely extends concepts and generalizes techniques from implicit surfaces, recursive subdivision algorithms, and volumetric sculpting. The interpolatory solid scheme is at least C^1 continuous for a wide range of control meshes and can be applied over arbitrary hexahedral meshes. Our implicit model offers the unique advantage of supporting a wide variety of shape design techniques in a single, integrated framework. Furthermore, implicit surfaces designed in our approach can be evaluated and polygonized rapidly, which makes our model useful in real-time sculpting systems. We have applied our new framework in an interactive sculpting environment called *DigitalSculpture* that supports CSG operations and local and large-scale deformation techniques, and that can import existing polygonal and voxel data-sets.

Although our approach is very powerful, it exhibits a few limitations that we are confident can be overcome with additional effort. Since the underlying scalar field is ultimately a discrete representation, our approach occasionally exhibits some of the undesirable behavior of traditional voxel models. For instance, aliasing can arise in iso-surfaces if the working space is not of sufficient resolution to represent a given feature. We envision that a local refinement algorithm can be used to overcome such difficulties, although the underlying subdivision scheme must be modified to accommodate such an algorithm. A local refinement algorithm would also help to reduce the memory needs of volumetric subdivision algorithms, whose storage requirements grow quickly when applied over very large meshes. This problem could be solved or at least lessened dramatically via local subdivision. Finally, although we can almost represent sharp features, a hierarchical or wavelet-based decomposition technique would be necessary to represent hard edges and 90° corners exactly.

Future research involving the new interpolatory scheme for subdivision solids includes application to other domains such as volume visualization, data representation and compression, and dynamic simulation. A local, adaptive, subdivision (refinement) algorithm could find wide application, but would require significant changes to the subdivision scheme and data structures. We are also currently investigating other volumetric subdivision schemes and their analysis. We plan to associate other quantities with our implicit subdivision solid framework, especially physical properties, and use our geometric model in finite element simulation and analysis. We envision a number of extensions to the *DigitalSculpture* system, including multiresolution sculpting, incorporation of an algorithm to handle local refinement and T-vertices, automatic construction of control meshes from volumetric data, and the incorporation of haptic I/O.

Acknowledgments

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, and the GAANN Grant P200A9703199.

References

[1] J.A. Baerentzen, N.J. Christensen, Volume sculpting using the level-set method, in: Proceedings of the 2002 International Conference on Shape Modeling and Applications, 2002, pp. 175–182.
[2] C. Bajaj, S. Shaefer, J. Warren, G. Xu, A subdivision scheme for hexahedral meshes, Visual Comput. 18 (5–6) (2002) 343–356.
[3] M. Bertram, Biorthogonal wavelets for subdivision volumes, in: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications, June 2002, pp. 72–82.
[4] J. Bloomenthal, Introduction to Implicit Surfaces, Morgan Kaufmann Publishers, 1997.
[5] M.-P. Cani, S. Hornus, Subdivision curve primitives: a new solution for interactive implicit modeling, in: Shape Modelling International, May 2001, pp. 82–88.
[6] Y. Chang, K.T. McDonnell, H. Qin, A new solid subdivision scheme based on Box splines, in: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications (Solid Modeling 2002), June 2002, pp. 226–233.
[7] Y. Chang, K.T. McDonnell, H. Qin, An interpolatory subdivision for volumetric models over simplicial complexes, in: Proceedings of the International Conference on Shape Modeling and Applications (SMI 2003), Seoul, Korea, May 2003, pp. 143–152.
[8] Y. Chang, H. Qin, A framework for multi-dimensional adaptive subdivision objects, in: Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications (Solid Modeling 2004), June 2004, pp. 123–134.
[9] S. Coquillart, Extended free-form deformation: A sculpturing tool for 3D geometric modeling, in: Computer Graphics (Proceedings of SIGRRAPH 90), August 1990, pp. 187–196.
[10] F. Dachille IX , A. Kaufman, Incremental triangle voxelization, in: Proceedings of Graphics Interface 2000, May 2000, pp. 205–212.
[11] G. Dewaele, M.-P. Cani. Interactive global and local deformations for virtual clay, in: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2003), 2003, pp. 131–140.
[12] N. Dyn, D. Levin, J. Gregory, A four-point interpolatory subdivision scheme for curve design, Comput. Aided Geometric Des. 4 (4) (1987) 257–268.
[13] E. Ferley, M.-P. Cani, J.-D. Gascuel, Practical volumetric sculpting, Visual Comput. 16 (7) (2000) 469–480.
[14] E. Ferley, M.-P. Cani, J.-D. Gascuel, Resolution adaptive volume sculpting, Graphic. Models (Special Issue on Volume Modeling) 63 (2002) 459–478.
[15] S.F. Frisken-Gibson, Using distance maps for accurate surface reconstruction in sampled volumes, in: Proceedings of the 1998 IEEE Symposium on Volume Visualization, 1998, pp. 23–30.
[16] J. Hua, Y. He, H. Qin, Multiresolution heterogeneous solid modeling and visualization using trivariate simplex splines, in: Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications, 2004, pp. 47–58.
[17] J. Hua, H. Qin, Haptic sculpting of volumetric implicit functions, in: Proceedings of the Ninth Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2001), Tokyo, Japan, Oct. 2001, pp. 254–264.
[18] T. Igarashi, S. Matsuoka, H. Tanaka. Teddy: a sketching interface for 3D freeform design, in: Computer Graphics (Proceedings of SIGRRAPH 99), 1999, pp. 409–416.

[19] L. Linsen, V. Pascucci, M.A. Duchaineau, B. Hamann, K.I. Joy. Hierarchical representation of time-varying volume data with $\sqrt[4]{2}$ subdivision and quadrilinear B-spline wavelets, in: Proceedings of Tenth Pacific Conference on Computer Graphics and Applications, 2002, pp. 346–355.
[20] R. MacCracken, K.I. Joy. Free-form deformations with lattices of arbitrary topology, in: Computer Graphics (Proceedings of SIGGRAPH 96), August 1996, pp. 181–188.
[21] K.T. McDonnell, Y. Chang, H. Qin, Interpolatory, solid subdivision of unstructured hexahedral meshes, Visual Comput. 20 (6) (2004) 418–436.
[22] K.T. McDonnell, H. Qin, FEM-based subdivision solids for dynamic and haptic interaction, in: Proceedings of Sixth ACM Symposium on Solid Modeling and Applications (Solid Modeling 2001), Ann Arbor, Michigan, 2001, pp. 312–313.
[23] K.T. McDonnell, H. Qin, R.A. Wlodarczyk, Virtual Clay: a real-time sculpting system with haptic toolkits, in: Proceedings of 2001 ACM Symposium on Interactive 3D Graphics, Research Triangle Park, North Carolina, 2001, pp. 179–190.
[24] K. Museth, D.E. Breen, R.T. Whitaker, A. Barr, Level set surface editing operators, in: Computer Graphics (Proceedings of SIGGRAPH 2002), July 2002, pp. 330–338.
[25] M.J. Muuss, L.A. Butler, Combinatorial solid geometry, boundary representations, and n-manifold geometry, in: D.F. Rogers, R.A. Earnshaw (Eds.), State of the Art in Computer Graphics: Visualization and Modeling, Springer-Verlag, Berlin, 1991, pp. 185–223.
[26] V. Pascucci, C. Bajaj, Time critical isosurface refinement and smoothing, in: Proceedings of the 2000 IEEE Symposium on Volume Visualization, 2000, pp. 33–42.
[27] A. Pasko, V. Adzhiev, B. Schmitt, C. Schlick, Constructive hypervolume modeling, Graph. Models 63 (6) (2001) 413–442.
[28] A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, Function representation in geometric modeling: concepts, implementation and applications, Visual Comput. 11 (8) (1995) 429–446.
[29] A. Pasko, V. Savchenko, A. Sourin, Synthetic carving using implicit surface primitives, Comput. Aided Des. 33 (5) (2001) 379–388.
[30] R.N. Perry, S.F. Frisken, Kizamu: a system for sculpting digital characters, in: Computer Graphics (Proceedings of SIGGRAPH 2001), 2001, pp. 47–56.
[31] A. Raviv, G. Elber, Three dimensional freeform sculpting via zero sets of scalar trivariate functions, Comput. Aided Des. 32 (8/9) (2000) 513–526.
[32] P. Reuter, I. Tobor, C. Schlick, S. Dedieu, Point-based modelling and rendering using radial basis functions, in: Proceedings of ACM GRAPHITE 2003, 2003, pp. 111–118.
[33] B. Schmitt, A. Pasko, V. Savchenko, Extended space mapping with bezier patches and volumes, in: Proceedings of Implicit Surfaces '99, Eurographics/ACM SIGGRAPH Workshop, 1999, pp. 25–31.
[34] B. Schmitt, A. Pasko, C. Schlick, Constructive modeling of FRep solids using spline volumes, in: Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications, 2001, pp. 321–322.
[35] T.W. Sederberg , S.R. Parry, Free-form deformation of solid geometric models, in Computer Graphics (Proceedings of SIGRRAPH 86), volume 20, pages 151–160, Aug. 1986..
[36] P. Shirley, A. Tuchman, A polygonal approximation to direct scalar volume rendering, in: Computer Graphics (San Diego Workshop on Volume Visualization), November 1990, pp. 63–70.
[37] G. Turk, H.Q. Dinh, J. O'Brien, G. Yngve, Implicit surfaces that interpolate, in: Proceedings of Shape Modelling International 2001, 2001, pp. 62–71.
[38] M. Šramek, A. Kaufman, Alias-free voxelization of geometric objects, IEEE Trans. Visualization Comput. Graph. 5 (3) (1999) 251–267.
[39] S.W. Wang, A.E. Kaufman, Volume sculpting, in: Proceedings of the 1995 Symposium on Interactive 3D Graphics, April 1995, pages 151–156.
[40] K.J. Weiler, Topological structures for geometric modeling. PhD thesis, Rensselaer Polytechnic Institute, August 1986.
[41] B. Wyvill, E. Galin, A. Guy, Extending the CSG tree. Warping, blending and boolean operations in an implicit surface modeling system, Comput. Graph. Forum 18 (2) (1999) 149–158.
[42] G. Yngve, G. Turk, Robust creation of implicit surfaces from polygonal meshes, IEEE Trans. Vizualization Comput. Graph. 8 (4) (2002) 346–359.