A New Solid Subdivision Scheme based on Box Splines

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

Department of Computer Science
State University of New York at Stony Brook

ABSTRACT

During the past twenty years, much research has been undertaken to study surface representations based on B-splines and box splines. In contrast, volumetric splines have received much less attention as an effective and powerful solid modeling tool. In this paper, we propose a novel solid subdivision scheme based on tri-variate box splines over tetrahedral tessellations in 3D. A new data structure is devised to facilitate the straightforward implementation of our simple, yet powerful solid subdivision scheme. The subdivision hierarchy can be easily constructed by calculating new vertex, edge, and cell points at each level as affine combinations of neighboring control points at the previous level. The masks for our new solid subdivision approach are uniquely obtained from tri-variate box splines, thereby ensuring high-order continuity. Because of rapid convergence rate, we acquire a high fidelity model after only a few levels of subdivision. Through the use of special rules over boundary cells, the B-rep of our subdivision solid reduces to a subdivision surface. To further demonstrate the modeling potential of our subdivision solid, we conduct several solid modeling experiments including free-form deformation. We hope to demonstrate that our box-spline subdivision solid (based on tetrahedral geometry) advances the current state-of-the-art in solid modeling in the following aspects: (1) unifying CSG, B-rep, and cell decomposition within a popular subdivision framework; (2) overcoming the shortfalls of tensor-product spline models; (3) generalizing both subdivision surfaces and free-form spline surfaces to a solid representation of arbitrary topology; and (4) taking advantage of triangle-driven, accelerated graphics hardware.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

General Terms

Algorithms, Design

[*]Email: {yusung|ktm|qin}@cs.sunysb.edu

Keywords

Representation conversion, Blends, sweeps, offsets & deformations, Multi resolution models, Geometric and topological representations, Reverse engineering, User interaction techniques.

1. INTRODUCTION

Although solid modeling is more desirable in many engineering and manufacturing applications, it has not yet gained popularity until recently due to both a lack of widespread standards and its strong need for more powerful computing resources. The past two decades have witnessed a significant growth in solid modeling, especially in the development of new solid representations. One such class of approaches employs implicit functions, such as CSG and blobby models [21]. They represent a solid as the solution set of an implicit function,

$$w = f(x, y, z).$$

In general, a level set

$$w = f(x, y, z),\ w = w_0$$

represents the boundary of the solid, and portions where $w < w_0$ comprise the interior. In implicit function representation, it is easy to differ the interior from the exterior and the boundary. Hence, they are very well suitable for algebraic operations on models [18]. However, there are a few disadvantages. For instance, there is no simple way to evaluate them in general cases [1] for rendering purpose. In addition, directly manipulating the level-set geometry is very challenging because the solid boundary is implicitly defined.

Parametric representations also have been developed and are well-studied because of their mathematical aspects, including Bernstein-Bézier solids [13], B-spline solids, and other tensor-product based [10, 16] approaches. Unlike implicit functions, parametric representations define a solid by

$$S(x, y, z) = \sum_i v_i\ N_i(x, y, z), \tag{1}$$

where the N_i's are basis functions that satisfy certain properties. Tensor-product based solids in particular have basis functions of the form

$$N_i(x, y, z) = N_{i_0}(x) \otimes N_{i_1}(y) \otimes N_{i_2}(z).$$

where $i = (i_0, i_1, i_2)$. Due to their tensor-product nature, their domains are restricted to a rectangular structure. In general, in contrast with implicit functions, the basis functions can be evaluated efficiently and robustly.

Since the pioneering work of Catmull and Clark [5] and Doo and Sabin [8] in the late 1970s, much research related to subdivision [9,

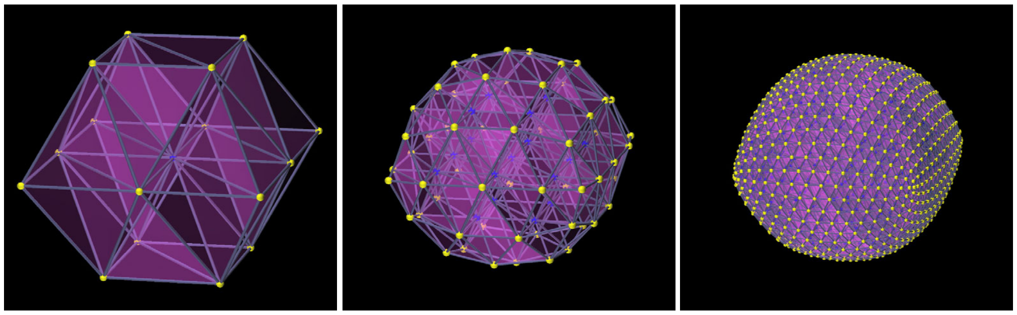
(a) (b) (c)

Figure 1: An example of our solid subdivision algorithm run on a simple model of 16 control points. (a) Initial control lattice. (b) Lattice after one level of subdivision. (c) Lattice after three levels of subdivision. The three lattices contain 24, 140, and 7280 cells, respectively. Faces are colored transparently to enhance the visibility of inside structures.

15, 12] and its analysis [19, 22] has been done. However, most of such work has been focused on surface representations, rather than solids. One exception is the work by MacCraken and Joy [16], which generalizes tri-cubic B-splines to solids of arbitrary topology. Bajaj and Warren [2] also suggested alternative solid subdivision rules for hexahedral meshes. Subdivision algorithms inherently have several key properties that make them very attractive in computer graphics, engineering, and manufacturing. Some of these advantages include:

- Uniformity of representation,
- Multiresolution analysis and levels of detail,
- Numerical efficiency and stability,
- Arbitrary topology or genus,
- Simplicity in implementation, and
- Hierarchical structure.

Our motivation is to combine the benefits of various existing solid modeling representations by employing subdivision as its foundation. We base our new subdivision scheme on volumetric box splines and take advantage of its strong mathematical foundation. In addition, we utilize non-tensor-product based tri-variate splines to ensure topological freedom. We introduce a new 3D regular structure that consists of two distinct types of polyhedra, which has been briefly addressed in [3] and utilized in [14] and [11], but has never been intensively employed previously in solid modeling field. The new mesh provides us with the subdivision masks for our scheme that share regular topology and simplicity. Simple affine combinations over the mesh enable an efficient and robust evaluation of the solid that can be expressed in the form of (1). By combining solid representations with subdivision techniques, our algorithm elegantly addresses many issues that current solid modeling techniques are confronted with. Its box spline foundation provides the advantage of high-order continuity without the need to have high-order degree basis functions. We also employ the well-known box spline based surface subdivision scheme [15] as our B-rep to facilitate the data exchange with current design and modeling requirements.

We demonstrate several application examples to illustrate the solid modeling potential of our novel subdivision scheme. Free-form deformation [20, 10, 16] is one of examples through which the advantage of our novel scheme is exhibited. The underlying tetrahedral mesh make it very easy to devise an efficient and accurate computation of the coordinates in the corresponding parametric domain. In addition, a few direct modeling sessions are presented to demonstrate how easily and effectively our approach can represent complex models of arbitrary topology.

2. TRI-VARIATE BOX SPLINE VOLUMES

2.1 3D Box Splines

There are many ways to define box splines, but one constructive way is by considering a *shadow* (or image) of a higher dimensional box in a lower dimensional space [4, 7]. More precisely, box splines are defined by the projection of n-dimensional hypercubes onto m-dimensional ($m < n$) affine space. Analytically, a box B of an n-dimensional affine space $\mathcal{A}^n$ is defined by

$$B_{(p,p_1,\cdots,p_n)} = \{\boldsymbol{v} \in \mathcal{A}^n \mid \boldsymbol{v} = \boldsymbol{p} + \sum_j c_j \boldsymbol{p}_j,\ c_j \in [0,1]\},$$

where $\boldsymbol{p}$ is a box vertex and $\boldsymbol{p}_j$ are linearly independent vectors in $\mathcal{A}^n$ that are representing n edges of an n-dimensional box. If all of the $\boldsymbol{p}_j$'s are of unit length, we call it a cube or *hypercube*.

An affine map $\pi : \mathcal{A}^n \longrightarrow \mathcal{A}^m$ denotes a projection onto an m-dimensional affine space $\mathcal{A}^m$. Consider the image of $B_{(p,p_1\cdots,p_n)}$ with respect to the map π. We have

$$\pi(B) = \{\boldsymbol{w} \in \mathcal{A}^m \mid \boldsymbol{w} = \boldsymbol{q} + \sum_j c_j \boldsymbol{q}_j,\ c_j \in [0,1]\},$$

where $\boldsymbol{q} = \pi(\boldsymbol{p})$ and $\boldsymbol{q}_j = \pi(\boldsymbol{p}_j)$.

Since $m < n$, it is obvious that the $\boldsymbol{q}_j$'s are not linearly independent. Hence, the pre-image of each point $\boldsymbol{w} \in \mathcal{A}^m$ forms a non-trivial affine subspace:

$$\pi^{-1}(\boldsymbol{w}) = \{\boldsymbol{v} \in \mathcal{A}^n \mid \pi(\boldsymbol{v}) = \boldsymbol{w}\}, \qquad (2)$$

which is called a *fibre* of the map π at $\boldsymbol{w}$. We can also derive the fibre by solving the linear system

$$c_1\boldsymbol{q}_1 + c_2\boldsymbol{q}_2 + \cdots + c_n\boldsymbol{q}_n = \boldsymbol{w} - \boldsymbol{q},$$

which has dimension $d = n - m$.

We now define a box spline as

$$M_B(\boldsymbol{w}) = \frac{vol\big(\pi^{-1}(\boldsymbol{w}) \cap B\big)}{vol\big(U(\boldsymbol{w})\big)},$$

where U is a fixed, d-dimensional unit box parallel to the fibre. Note that M_B has local support and satisfies C^d continuity inside the support. Finally, the normalized version N_B is obtained so that it forms a partition of unity over the lattice $\mathbb{Z}^m$. It satisfies C^{d-l} continuity over the space where l is the largest dimension of collapsing faces under π [6].

For instance, Loop's surface subdivision scheme [15] is based on the case of $n = 6$, $m = 2$. Our solid approach is the case of $n = 8$, $m = 3$. However, the lack of regular tessellation over the lattice $\mathbb{Z}^m$ (except cubic-grid) in 3D makes an analogy complicated. This aspect is addressed in the following sections in details.

2.2 3D Regular Mesh

Since $\boldsymbol{q}_j$ does not form a linearly independent set on the m-dimensional space, we are considering a *regular mesh* on the range space and allowing some of the edges to overlap. Unlike 2D space, 3D space does not have a regular mesh which consists of simplices, i.e., tetrahedra. Nonetheless, it is possible to fill the space with two types of polyhedra rather than a single type of equilateral tetrahedra. The initial form of the mesh can be seen by projecting a 4D hypercube onto 3D space through its longest diagonal, or the *main* diagonal, $(0,0,0,0) \rightarrow (1,1,1,1)$. Let the mapping be π_d. Then we have

$$\begin{aligned}\pi_d((1,0,0,0)) &= (1,0,0) = \boldsymbol{u}_1,\\ \pi_d((0,1,0,0)) &= (0,1,0) = \boldsymbol{u}_2,\\ \pi_d((0,0,1,0)) &= (0,0,1) = \boldsymbol{u}_3,\\ \pi_d((0,0,0,1)) &= (-1,-1,-1) = \boldsymbol{u}_4.\end{aligned}$$

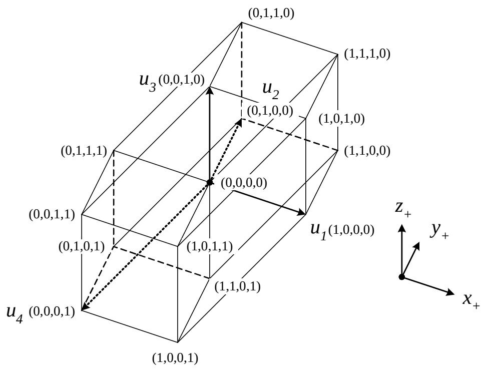

Figure 2: 4D hypercube projected on 3D space.

Figure 2 shows an image of 4D hypercube by π_d. The problem is that the image does not form a polyhedral structure in 3D space. We have to introduce additional edges to make it complete. At the same time, however, we do not want to introduce too many auxiliary edges which can cause unwanted complexity and irregularity. The simplest regular structure that consists of 4 mesh directions $\{u_1, u_2, u_3, u_4\}$ is called an *octet-truss*, which is shown in Figure 3. It may be noted that, this structure serves as a mesh for our subdivision scheme. In a nutshell, it consists of two distinct types of polyhedra – tetrahedra and octahedra. Each vertex has a valence of 14 in the regular case and shares the same topology as its neighbors. The mesh is denoted by $\mathcal{M}_{OT}(\mathbb{Z}^3)$.

3. SOLID SUBDIVISION RULES

3.1 Subdivision

One of the attractive properties of box splines is that they can be decomposed into an affine combination of splines with smaller support [3, 7]. We first consider a subdivision of the n-dimensional hypercube into 2^n cubes, or *sub-cells* with a half edge length. The im-

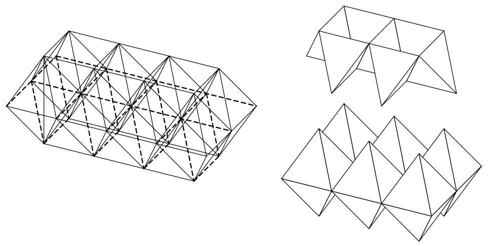

Figure 3: A typical example of the octet-truss $\mathcal{M}_{OT}(\mathbb{Z}^3)$ in 3D space. It consists of octahedral grids with tetrahedra in between.

ages of sub-cells also form box splines, but with smaller supports. The original box spline can be written as an affine combination of box splines of sub-cells. Since every cube has edge vectors of the same orientation, we represent a cube as C_i where $i \in \mathcal{M}_{OT}(\mathbb{Z}^3)$, which corresponds to the image of the projectional axis, i.e., the main diagonal of a cube. We write simply $S_i = S_{C_i}$. Also, we use a hat notation to represent sub-structures, i.e., $\hat{M}_{\hat{\imath}}$ means a box spline of a sub-cell $\hat{C}_{\hat{\imath}}$ where $\hat{\imath} \in \mathcal{M}_{OT}(\frac{1}{2}\mathbb{Z}^3)$. First, in the linear case ($n = 4$, $m = 3$), it is clear to notice

$$N_i = \frac{1}{2}\sum_{\hat{\jmath}} \alpha_{i,\hat{\jmath}}\ \hat{N}_{\hat{\jmath}}, \tag{3}$$

where

$$\alpha_{i,\hat{\jmath}} = \begin{cases} 2 & \text{if } i = \hat{\jmath} \\ 1 & \text{if } i \text{ is adjacent to } \hat{\jmath} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The adjacent information among the i's is controlled by the presence of edges in $\mathcal{M}_{OT}$. Furthermore, if our solid is given by

$$S = \sum_i v_i\ N_i, \tag{5}$$

where $\{v_i\}$ are control points in $\mathcal{A}^3$ and $i \in \mathcal{M}_{OT}(\mathbb{Z}^3)$, then, by applying (3) to (5), we obtain

$$\begin{aligned} S &= \sum_i \Big(v_i\Big(\frac{1}{2}\sum_{\hat{\jmath}} \alpha_{i,\hat{\jmath}}\ \hat{N}_{\hat{\jmath}}\Big)\Big) = \sum_{\hat{\jmath}} \Big(\Big(\frac{1}{2}\sum_i \alpha_{i,\hat{\jmath}}\ v_i\Big)\ \hat{N}_{\hat{\jmath}}\Big) \\ &= \sum_{\hat{\jmath}} w_{\hat{\jmath}}\ \hat{N}_{\hat{\jmath}} \end{aligned}$$

where

$$w_{\hat{\jmath}} = \frac{1}{2}\sum_i \alpha_{i,\hat{\jmath}} v_i.$$

Therefore, control points $w_{\hat{\jmath}}$ for the refined mesh $\mathcal{M}_{OT}(\frac{1}{2}\mathbb{Z}^3)$ can be expressed as affine combinations of the original control points, namely, the v_i's. By convention, N_i is denoted by $N_i^{1,1,1,1}$, since each mesh direction has only one edge from the hypercube projected onto it.

In the interest of generating high-order continuous solids, our proposed scheme is based on $n = 8$, or $N_i^{2,2,2,2}$. In this case, each mesh direction is projected twice by 8D hypercube edges. Unlike in the linear case, we need to count the image of sub-cells twice. Therefore, the N_i can be written as

$$N_i = \frac{1}{2}\sum_{\hat{\jmath}} \alpha_{i,\hat{\jmath}}\ \Big(\frac{1}{c}\sum_{\hat{k}} \alpha_{\hat{\jmath},\hat{k}}\ \hat{N}_{\hat{k}}\Big).$$

Once again, if we have a solid expressed by

$$S = \sum_i v_i \, N_i,$$

then,

$$S = \sum_j w_{\hat{j}} \, \hat{N}_{\hat{j}},$$

where

$$w_{\hat{j}} = \frac{1}{2c} \sum_i \sum_{\hat{k}} \alpha_{i,\hat{j}} \, \alpha_{\hat{j},\hat{k}} \, v_i \tag{6}$$

$$= \frac{1}{2c} \sum_i \sum_{\hat{k}} \beta_{i,\hat{j},\hat{k}} \, v_i. \tag{7}$$

The coefficient α is defined by (4). The variable c is a normalization factor to keep the summation equal to one. The value is $2^4 = 16$ in our case. Refined control points $w_{\hat{j}}$ can be specified as vertex, edge, or cell points, depending on different cases. We call them $v_{\hat{j}}, e_{\hat{j}}$, and $c_{\hat{j}}$, respectively.

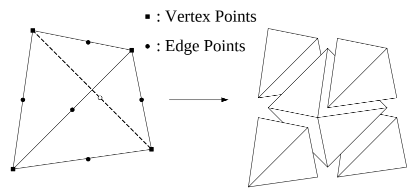

Figure 4: A tetrahedron is subdivided into an octahedron and 4 tetrahedra surrounding it.

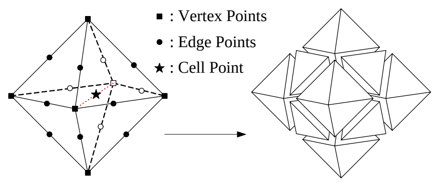

Figure 5: An octahedron is subdivided into 6 octahedra with 8 tetrahedra in between.

3.2 Subdivision Mask

We employ $\mathcal{M}_{OT}(\mathbb{Z}^3)$ as our regular mesh. In the subdivision process, elements in the mesh are subdivided into smaller ones. Figure 4 and Figure 5 explain the way in which the mesh elements are divided. The subdivided mesh again forms an octet-truss of half the size. Therefore, after the first level, any local irregularities are contained and only regular cases occur inside.

Even though the introduction of octahedra offers the benefit of simplicity and regularity in structure, we need to choose an orientation of the mesh due to asymmetry of our mask, which is based on 4D hypercube projection (Figure 6). We devise this orientation by choosing one of the diagonals inside an octahedron, which is denoted by a *major diagonal*. For each vertex point $v_{\hat{j}}$, we define $v_{i'}$ as an *adjacent vertex* if and only if there exists an edge or a major diagonal between j and i'. Also, an *edge neighbor* of $e_{\hat{j}}$ is defined by a vertex of a tetrahedral cell that shares an edge (i_0, i_j) on which j lies, or a vertex of an octahedron that shares the edge *and* and whose major diagonal joins the vertex and one of the edge's end-points. For cell points, *cell neighbors* are easily defined using the vertices that comprise the cell.

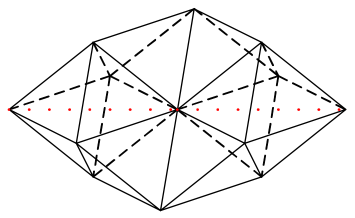

Figure 6: A regular subdivision mask. It is a projected image of a 4D hypercube that is visualized as an octet-truss. Major diagonals (red dotted lines) are introduced due to the asymmetrical aspect of the mask.

3.3 Subdivision Rules

As mentioned previously, each new control point $w_{\hat{j}}$ in $\mathcal{M}_{OT}(\frac{1}{2}\mathbb{Z}^3)$ is obtained by an affine combination of the v_i's in $\mathcal{M}_{OT}(\mathbb{Z}^3)$, whose coefficients are defined by (6). The evaluation of coefficients could be a tedious process in general. For instance, in the $\hat{j} = i$ case, the coefficient of v_i is $\frac{18}{32}$, because $\beta_{i,\hat{j},\hat{k}}$ is equal to 4 if $\hat{j} = \hat{k}$ and equal to 1 for all other adjacent $\hat{k}$'s, whose number is 14. We classify them into the following cases.

3.3.1 *Vertex Points*

This is the case when $\hat{j} = i$ for some i's. In the regular case, a new vertex point is computed by

$$v_{\hat{j}} = \frac{18}{32} \, v_i \; + \; \frac{1}{32} \sum_{i'} v_{i'}, \tag{8}$$

where $i = \hat{j}$ and i' is an adjacent index of $\hat{j}$ in $\mathcal{M}_{OT}(\mathbb{Z}^3)$. The number of adjacent vertices, $|\{i'\}|$, is equal to 14 in the regular case. For the general case with valence k, we can use

$$v_{\hat{j}} = \frac{18}{32} \, v_i \; + \; \frac{14}{32k} \sum_{i'} v_{i'}, \tag{9}$$

without much degeneration. It should be mentioned that the (9) only ensures convergence around irregular vertices ($k \neq 14$). However, the difference is hard to notice. Therefore, it is used in our applications because of its simplicity.

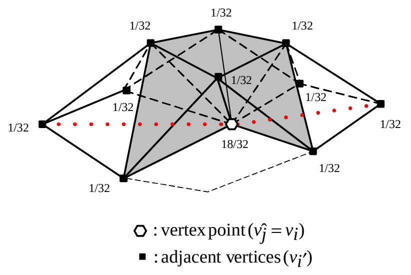
1/32
1/32
1/32
1/32
1/32
1/32
1/32
1/32
1/32
1/32
18/32
○ : vertex point ($v_{\hat{j}} = v_i$)
▪ : adjacent vertices ($v_{i'}$)

Figure 7: A vertex point and its mask in the regular case. Only the top half is shown. 4 more adjacent vertices are placed below the vertex point. Red dotted lines indicate major diagonals. Gray areas indicate faces which belong to tetrahedra.

3.3.2 *Edge Points*

For the case $\hat{j} \neq i$ for any i, there are two sub-cases. The first case is an edge point, which lies on an edge by i_0 and i_1 in $\mathcal{M}_{OT}(\mathbb{Z}^3)$ that is *not* a major diagonal of an octahedron. In the regular case, we have

$$e_{\hat{j}} = \frac{10}{32}\,(v_{i_0} + v_{i_1}) \;+\; \frac{2}{32}\,\sum_{i'} v_{i'}, \qquad (10)$$

where i' denotes an edge neighbor of $\hat{j}$. There are 6 edge neighbors, as shown in Figure 8. In general case, we may use

$$e_{\hat{j}} = \frac{10}{32}\,(v_{i_0} + v_{i_1}) \;+\; \frac{12}{32k}\,\sum_{i'} v_{i'}, \qquad (11)$$

where k is a number of edge neighbors.

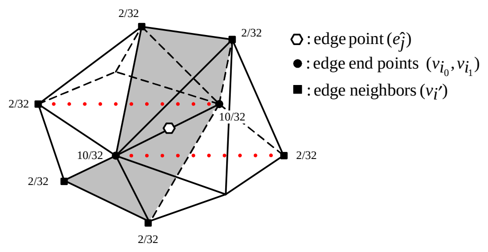
2/32
2/32
2/32
10/32
10/32
2/32
2/32
2/32
○ : edge point ($e_{\hat{j}}$)
● : edge end points (v_{i_0}, v_{i_1})
■ : edge neighbors ($v_{i'}$)

Figure 8: An edge point and its mask. Note that vertices on major diagonals (red dotted lines) are included. Gray areas indicate faces of tetrahedra.

3.3.3 Cell Points

When $\hat{j}$ lies on a major diagonal of an octahedron formed by i_0 and i_1, we use

$$c_{\hat{j}} = \frac{8}{32}\,(v_{i_0} + v_{i_1}) \;+\; \frac{4}{32}\,\sum_{i'} v_{i'}, \qquad (12)$$

where i' is a cell neighbor that belongs to the same octahedron (Figure 9). The number of neighbors is always 4.

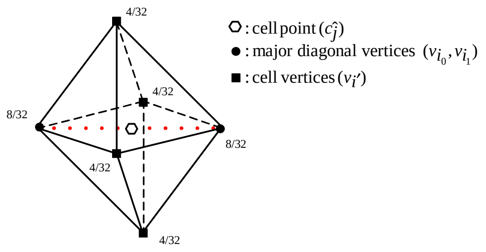
4/32
4/32
8/32
8/32
4/32
4/32
○ : cell point ($c_{\hat{j}}$)
● : major diagonal vertices (v_{i_0}, v_{i_1})
■ : cell vertices ($v_{i'}$)

Figure 9: A cell point and its neighbors. The vertices on the major diagonal have different weights.

3.4 Splitting and Reconnecting

Because our algorithm keeps octahedra that are created during the splitting of cells, it is obvious what the connectivity of subcells should be. Each tetrahedron is split into 4 tetrahedra, each of which consists of one vertex point and 3 edge points adjacent to it, and one octahedron that consist of 6 edge points that are generated by the 6 edges of the tetrahedron (Figure 4). An octahedron is divided into 6 octahedra and 8 tetrahedra. Each of the sub-octahedra comprises a vertex point that is from each vertex of the octahedra, 4 edge points from an adjacent edges, and one cell point. Each new tetrahedron is made by connecting 3 edge points from one face and the cell point (Figure 5).

3.5 Boundary Surface

The boundary requires special treatment. Since our solid subdivision scheme is based on box splines, it is natural to choose a box spline surface as its B-rep. We employ Loop's scheme [15] for this purpose. The weight used in the general case of valence k is based on a modified version by Warren [23]. For surface vertices, masks include only neighbors that are on the surface. This guarantees high-order continuity not only in the interior, but on the boundary as well. Attention also must be paid to the interface between the boundary and interior of a solid object. For surface subdivision algorithms, one can introduce modified rules [23] to acquire open (rather than closed) surfaces over 2D parametric spaces. However, the modified masks are often very complex. In solid subdivision, boundaries and interfaces are inevitable. There could be some visual irregularities in interfaces which have to be addressed in future research. Nonetheless, we can still assure convergence and C^0 continuity.

4. IMPLEMENTATION

4.1 Data Structure

Our subdivision scheme requires a data structure to handle two types of cells, even though our input data consist of only tetrahedra. We have implemented a flexible structure which can handle arbitrary type of faces and cells. Edges (i.e., adjacency information) and faces are reconstructed each time the subdivision is invoked. In the interest of memory efficiency, pointers are used to record adjacency information.

In each step, new vertices are generated by taking affine combinations of vertices from the previous level, formulated in (6). This can be expressed in the form of the subdivision matrix:

$$\begin{pmatrix} \vdots \\ w_{\hat{j}} \\ \vdots \end{pmatrix} = A \begin{pmatrix} \vdots \\ v_i \\ \vdots \end{pmatrix}.$$

We need to maintain the matrix A for several reasons. Even though doing so imposes a heavy memory requirement, this data is critical for our Free-Form Deformation (FFD) and other shape modeling applications to sustain real-time performance. It may be noted that matrices are required only when we are performing free-form deformation or direct manipulating on the models. Fortunately, since most of the matrix consists of zeros, sparse matrix storage schemes can be used to dramatically reduce memory consumption.

4.2 Diagonal Orientation

One problem with the mesh is that we need both to choose major diagonals for octahedra and to maintain their orientations. In the first level of subdivision, we can select diagonals for sub-octahedra arbitrarily since we make no assumptions about the input lattice. However, each time we choose diagonals, additional information must be recorded to recover orientations in the next level. Each sub-tetrahedron remembers sub-octahedra in the same parent cell as *orientation references*. Hence, the next time we subdivide the sub-cell, it refers to the linked cell to extract orientation information. In this way, major diagonals are maintained regularly during the process.

5. APPLICATIONS AND RESULTS

5.1 Free-Form Deformation

Free-form deformation (FFD) is one of the important applications to which our subdivision is directly applied. It plays a crucial role in graphics, design, and manufacturing. Usually, FFD involves generating parametric solids and translating model coordinates back to parametric space [20], so that changes to the solids can be reflected in the models. Our approach is similar to that of MacCracken and et al. [16]. However, unlike the tensor-product nature of Catmull-Clark solids and volumetric splines, our solid is much more flexible due to the tetrahedral structure of the mesh. The following is an overview of our implementation of FFD:

1. Generate an appropriate mesh that contains the model to be deformed.
2. Subdivide the mesh up to the user-specified level using our new solid subdivision scheme.
3. Calculate barycentric coordinates for each vertex in the model on the final level.
4. The user then interactively moves control points in any coarser level.
5. Recalculate the coordinates by following the subdivision matrices.

The barycentric coordinates are easily computed. Suppose a model vertex p lies within a tetrahedron (v_0, v_1, v_2, v_3). The coordinate (c_1, c_2, c_3) is given by

$$p = v_0 + c_1\, u_1 + c_2\, u_2 + c_3\, u_3, \qquad (13)$$

where $u_i = v_i - v_0$ for $i = 1, 2, 3$. Conditions $0 \le c_1, c_2, c_3 \le 1$ and $0 \le c_1 + c_2 + c_3 \le 1$ are required to be in the tetrahedron. We can solve the linear system

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots \\ u_1 & u_2 & u_3 \\ \vdots & \vdots & \vdots \end{pmatrix}^{-1} \begin{pmatrix} \vdots \\ p - v_0 \\ \vdots \end{pmatrix},$$

to obtain the coordinate. If a vertex is in an octahedral cell, we split the octahedron into 4 parts using a major diagonal and compute the barycentric coordinate within the corresponding tetrahedron.

Figure 10 and Figure 11 show two applications of FFD. The mesh does not necessarily have a simple topology (i.e., genus zero), as shown in 12. It is also possible that we manipulate objects locally (Figure 13), by simply assigning control meshes to certain region of an object that we want to deform. Note that all processes are done in real-time except barycentric coordinate computation. Even the coordinate computation can be done in a matter of seconds. Most of time is consumed by intersection checks between tetrahedra/octahedra and model vertices. We are working on several methods to accelerate these checks.

5.2 Direct Manipulation of Solids

Figure 1 and Figure 14 demonstrate examples of various models that can be obtained by our solid subdivision. The tetrahedral structure offers the greatest freedom to generate objects of arbitrary topology. Also, by simple user interaction, we can perform real-time modifications on solids by manipulating control points at arbitrary levels. We can also introduce discontinuities on vertices or along edges by assigning exceptional rules (i.e., simple bisection without weights) to desired parts of objects.

5.3 Performance

All of our results have been run on a wide-range of consumer level PCs which do not have any special hardware for volume visualization. We were able to perform some of the free-form deformations on a relatively low-end system (Intel Celeron 700MHz without hardware accelerated OpenGL rendering). It clearly demonstrates how efficient our subdivision algorithm is in many aspects. In most cases, the only requirement is a large amount of memory (desirably more than 256 MB), which could be further alleviated by optimizing the data structure. Even volume visualization can be done on PCs by using the OpenGL 3D texture implementation available in some recent video cards.

On a Pentium III 1 GHz machine with 1 GB RAM, the car model (Figure 10) required only 0.701 seconds to subdivide up to level 3 (7280 cells). The coordinate update, triggered by user input, takes roughly 0.03 seconds in each time, which guarantees real-time interaction. The filter model (Figure 11) consists of 24877 vertices and 49548 faces, takes 0.5 seconds to subdivide meshes (5200 cells). The interaction cost was comparable to that of the car model. Even though it is possible to use higher subdivision levels without much additional computational cost, the results are visually indistinguishable.

6. CONCLUSION AND FUTURE WORK

We have developed a novel solid subdivision scheme based on powerful box splines. The new solid subdivision scheme has a lot of potential, especially in solid modeling. Since it is founded on well-defined box splines, it is easy to integrate with current industrial standard models based on splines. It can also achieve high-order continuity with relatively low degree basis functions and numerical stability. Its B-rep, which is critical in rendering, is the well-known box spline surface. The tetrahedral mesh affords users much freedom in modeling and deformation, in such way that other tensor-product based or parametric solids can not represent. Moreover, the subdivision nature of the entire process offers fast evaluation without much numerical cost, provides a multiresolution solution, and facilitates real-time manipulation.

Our free-form deformation application demonstrates the robustness and efficiency of our novel scheme. The structural simplicity of our meshes gives rise to the critical robustness issue that is required to manipulate highly complicated models, and the rapid convergence rate makes real-time user interaction possible. Levels of details can be easily computed simply by choosing the subdivision level, and effects from one level are propagated by means of the subdivision matrix without any extra numerical cost. Therefore, our subdivision algorithm could offer a number of advantages to interactive solid modeling, for instance, Virtual Clay by McDonnell et al. [17].

Finally, even though the process does not generate any new extraordinary points, original control points may include non-regular cases which are not yet fully analyzed and addressed. Many researchers have analyzed extraordinary cases for surface subdivisions [19, 22], and these methods can be applied to our solid subdivision. We have already analyzed cases using subdivision matrix [8] and numerical techniques, and the values currently used in our experiments assure convergence and certain level of visual continuity around extraordinary vertices. A complete analysis is another topic for future research.

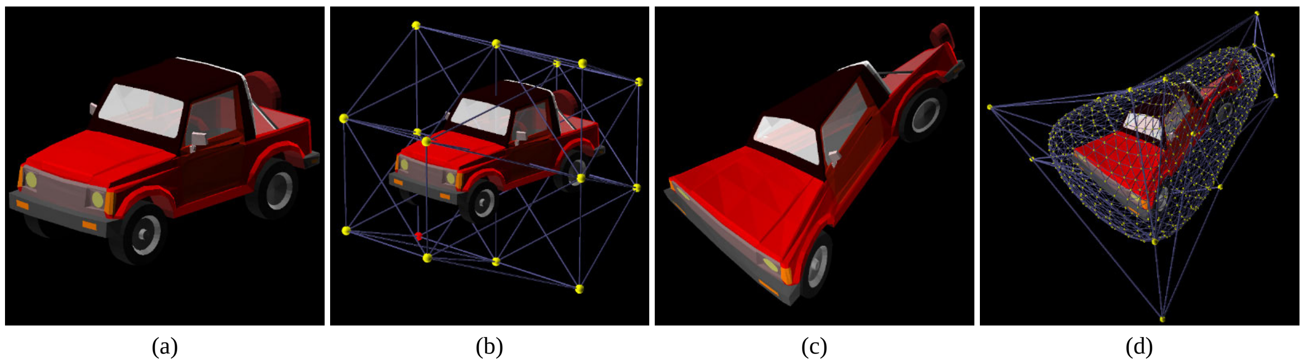

Figure 10: Free-form deformation of a car model. The model contains 2244 vertices with 21 different components. (a) The original model. (b) The subdivision mesh. (c) The deformed model. (d) Underlying subdivision solid. All deformation has been done in real-time.

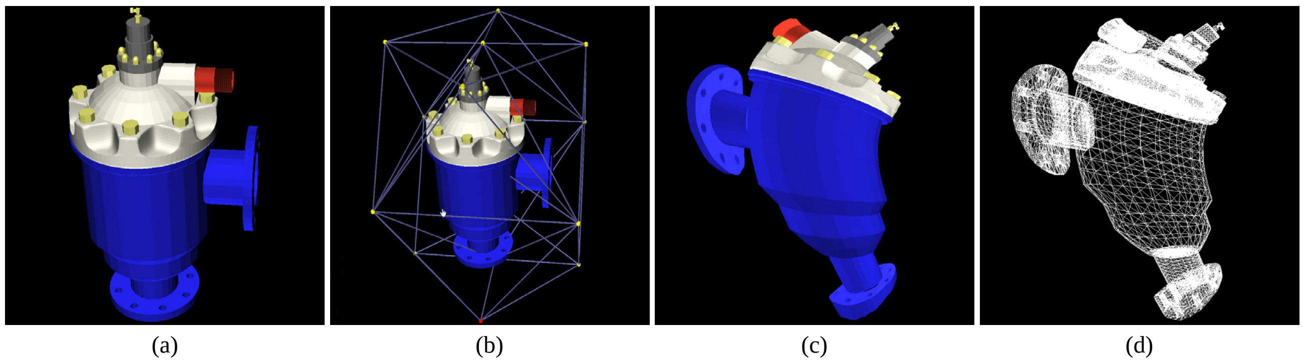

Figure 11: Another example of free-form deformation of an industrial filter block model. The model contains 24877 vertices with more than 49000 faces, and is converted from the B-spline surface model. (a) The original model. (b) The subdivision mesh. (c) The deformed model. (d) The deformed model in wireframe.

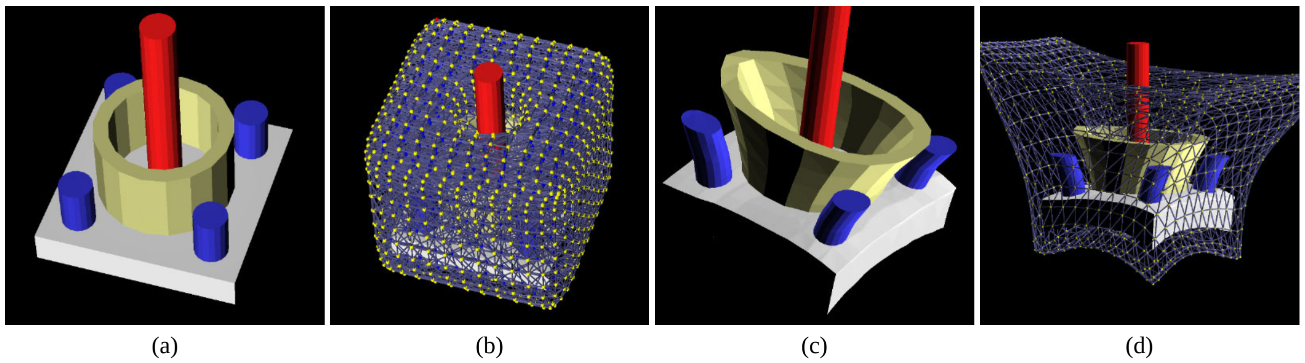

Figure 12: A mesh with non-trivial topology. In this case, our deformation mesh contains a hole. (a) The original model. (b) The subdivision mesh in level 2. (c) The part of the model (the central cylinder) that is inside the hole has not been changed. (d) The deformed subdivision mesh.

Acknowledgments

This research was supported in part by the NSF CAREER award CCR-9896123, the NSF grants IIS-0082035 and IIS-0097646, Honda Initiation Award, and Alfred P. Sloan Fellowship.

7. REFERENCES

[1] P. R. Atherton. A scanline hidden surface removal procedure for constructive solid geometry. *Computer Graphics (SIGGRAPH 83 Proceedings)*, 17(3):73–82, July 1983.

[2] C. Bajaj, J. Warren, and G. Xu. A smooth subdivision scheme for hexahedral meshes. *The Visual Computer*, 2001. To appear in the special issue on subdivision.

[3] W. Boehm. Subdividing multivariate splines. *Computer-Aided Design*, 15:345–352, Nov. 1983.

[4] W. Boehm. Calculating with box splines. *Computer Aided Geometric Design*, 1(2):149–162, 1984.

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

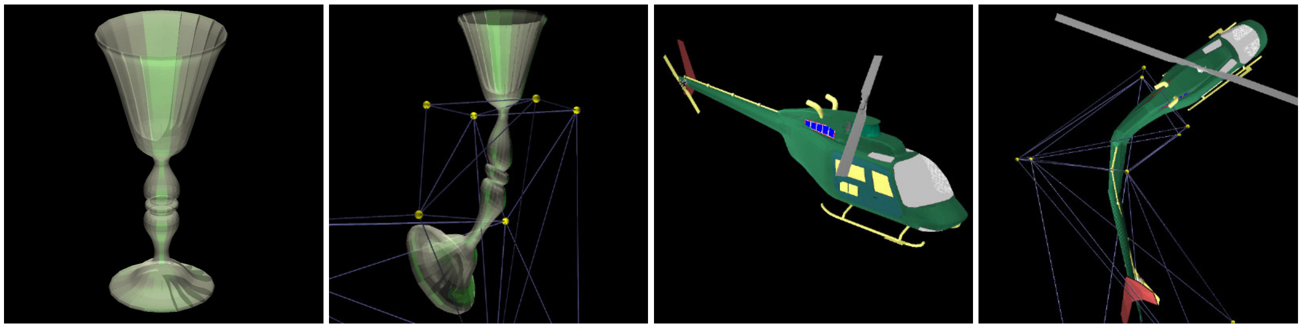

Figure 13: Localized free-form deformation. We can choose any region of the model and perform FFD. (a) The original model. (b) Locally deformed model. (c) The original model. (d) Locally deformed model. Each model contains 3760 and 7854 vertices, respectively.

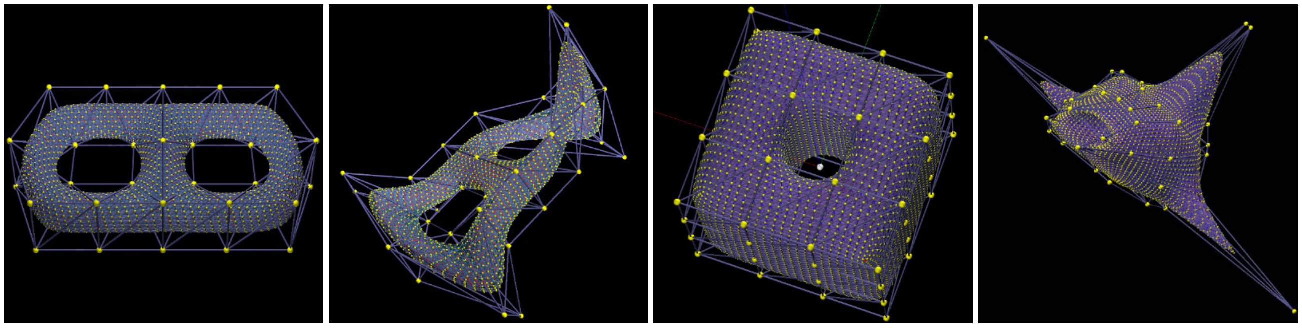

Figure 14: Examples of some models and their manipulations. (a) The original model with genus two. (b) The deformed model. (c) The original model with genus one. (d) The plane made from the model (c).

[6] C. de Boor and K. Höllig. B-splines from parallelepipeds. *J. Analyse Math.*, 42:99–115, 1982.

[7] C. de Boor, K. Höllig, and S. Riemenschneider. *Box Splines*. Springer-Verlag, New York, 1993.

[8] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, Sept. 1978.

[9] N. Dyn, D. Levin, and J. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, April 1990.

[10] J. Greissmair and W. Purgathofer. Deformation of solids with trivariate B-Splines. In *Proceedings of Eurographics '89*, pages 137–148, 1989.

[11] M. Hall and J. Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, 10(6):33–42, Nov. 1990.

[12] L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. In *Computer Graphics Forum (Proceedings of Eurographics '96)*, volume 15(3), pages 409–420, 1996.

[13] D. Lasser. Bernstein-bezier representation of volumes. *Computer Aided Geometric Design*, 2(1-3):145–150, 1985.

[14] E. Leitner and S. Selberherr. Mixed-element decomposition method for three-dimensional grid adaptation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(7):561–572, July 1998.

[15] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Dept. of Math., 1987.

[16] R. MacCracken and K. I. Joy. Free-Form deformations with lattices of arbitrary topology. In *SIGGRAPH '96 Computer Graphics Proceedings, Annual Conference Series*, pages 181–188, Aug. 1996.

[17] K. T. McDonnell, H. Qin, and R. A. Wlodarczyk. Virtual clay: A real-time sculpting system with haptic toolkits. In *Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics*, pages 179–190, March 2001.

[18] A. Pasko and V.Savchenko. Algebraic sums for deformation of constructive solids. In *Proceedings of Solid Modeling '95*, pages 403–408, May 1995.

[19] H. Prautzsch. Generalized subdivision and convergence. *Computer Aided Geometric Design*, 2(1-3):69–76, 1985.

[20] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Computer Graphics (SIGGRAPH 86 Proceedings)*, volume 20, pages 151–160, Aug. 1986.

[21] B. Wyvill, C. McPheeters, and G. Wyvill. Animating soft objects. *The Visual Computer*, 2(4):235–242, 1986.

[22] D. Zorin. Smoothness of stationary subdivision on irregular meshes. *Constructive Approximation*, 16:3, 2000.

[23] D. Zorin and P. Schröder. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*, 2000.