Free-Form Deformations With Lattices of Arbitrary Topology

Ron MacCracken
Kenneth I. Joy

Computer Graphics Research Laboratory
Department of Computer Science
University of California, Davis[1]

Abstract

A new free-form deformation technique is presented that generalizes previous methods by allowing 3-dimensional deformation lattices of arbitrary topology. The technique uses an extension of the Catmull-Clark subdivision methodology that successively refines a 3-dimensional lattice into a sequence of lattices that converge uniformly to a region of 3-dimensional space. Deformation of the lattice then implicitly defines a deformation of the space. An underlying model can be deformed by establishing positions of the points of the model within the converging sequence of lattices and then tracking the new positions of these points within the deformed sequence of lattices. This technique allows a greater variety of deformable regions to be defined, and thus a broader range of shape deformations can be generated.

1 Introduction

Efficient and intuitive methods for three-dimensional shape design, modification, and animation are becoming increasingly important areas in computer graphics. The model-dependent techniques initially used by designers to modify surfaces required each primitive type to have different parameters and/or control points that defined its shape. Model designers had to consider this mathematical model when making the desired modifications, and shape design could be difficult – making simple changes to the surface required the modification of many surface parameters. The process grew more difficult when local changes, such as adding arbitrarily shaped bumps, or global changes, such as bending, twisting, or tapering were necessary.

The free-form deformations [5, 6, 9, 19] were designed to deal with some of these problems. These methods embed an object in a deformable region of space such that each point of the object has a unique parameterization that defines its position in the region. The region is then altered, causing recalculation of the positions based upon their initial parameterization. If a deformable space can be defined with great flexibility and with few control points relative to the number in the surface model, then complex models comprised of thousands of control points can be deformed in many interesting ways with very little user-interaction.

[1] Department of Computer Science, University of California, Davis CA 95616. Email: {maccrack,joy}@cs.ucdavis.edu

Barr [1] first introduced deformations by creating operations for stretching, twisting, bending and tapering surfaces around a central axis (x, y, or z). Operations that involved moving many control points could now be accomplished with the altering of as little as one parameter. However, this technique limits the possible definitions of the deformable space to that of a single coordinate system about an axis, and restricts the ways in which the space can be altered - the axis can not be modified and the deformable space can only be modified by a few parameters.

Barr's deformations were followed by a more generalized approach to the problem, the Free-Form Deformations (FFDs) of Sederberg and Parry [19]. This method imposes an initial deformation lattice on a parallelepiped, and defines the deformable space as the trivariate Bézier volume defined by the lattice points. The parallelepiped form of the lattice allows points of an embedded object to be quickly parameterized in the space of the lattice, and as the lattice is deformed, the deformed points can be calculated by simple substitution into the defining equations of the trivariate volume. This method is widely used because of its ease of use and power to create many types of deformations with little user-interaction. Griessmair and Purgathofer [9] extended this technique by utilizing a trivariate B-Spline representation. Although both methods give the user many controls to alter the deformable space, both Sederberg and Parry's FFDs and Griessmair and Purgathofer's deformation techniques handle only a specific type of space definition, that defined initially by a parallelepiped lattice.

In order to generate free-form deformations for a more general lattice structure, Coquillart introduced Extended Free-Form Deformations (EFFD) [5, 6]. This method uses the initial lattice points to define an arbitrary trivariate Bézier volume, and allows the combining of many lattices to form arbitrary shaped spaces. Modifying the points of the defining lattice creates a deformation of space where one trivariate volume is deformed into another. This extension allows a greater inventory of deformable spaces, but loses some of the flexibility and stability of Sederberg and Parry's FFDs: While the corner control points of the joined lattices are user-controllable, the internal control points are constrained to preserve continuity; and, calculating the parameterization of a point embedded in the initial trivariate volume requires numerical techniques.

A recent deformation technique developed by Chang and Rockwood [4] generalizes Barr's technique in a different manner. Instead of defining the space in a free-form manner, Chang's approach deals with increasing the flexibility of an axis-based approach by allowing modifications to the axis during the deformation. The technique allows the user to define the axis as a Bézier curve with two user-defined axes at each control point of the curve. Repeated affine transformations using a generalized deCasteljau approach are used to define the deformable space. This technique is very powerful, intuitive, and efficient, but again restricts the ways in which the space surrounding the curve can be altered.

This paper introduces a further extension to these techniques by establishing deformation methods defined on lattices of arbitrary topology. In this case, the deformable space is defined by using a volume analogy of subdivision surfaces [2, 7, 8]. In these subdivision methods, the lattice is repeatedly refined, creating a sequence of lattices that converge to a region in three-dimensional space. This refinement procedure is used to define a pseudo-parameterization of an embedded point. As the points of the lattice are modified a deformation of the space is created, and the embedded points can be relocated within the deformed space.

This method has been found to be quite intuitive for the designer and dramatically increases the inventory of lattices that can be considered in a free-form deformation. The twists and bends of Barr [1] and the cylindrical lattices of Coquillart [5] can be easily simulated. By allowing meshes of arbitrary topology, the continuity problems of adjoining lattices virtually disappear.

In section 2, we give an overview of the subdivision methods that are used to define the deformable space from the lattice. In our case these methods are based upon an extension of the Catmull-Clark refinement rules for surfaces [2]. In section 3, we modify the Catmull-Clark procedure to control the boundary surfaces and curves of the deformable region. This produces a deformable region that can be intuitively defined from the lattice. In section 4 we discuss the methods that give a correspondence between a point embedded in the deformable space and the sequence of lattices generated by the refinement procedure. In section 5 we present an overview of the complete deformation procedure. Implementation details of the algorithm are discussed in section 6 and results are given in section 7.

2 Defining the Deformable Space from the Lattice

A *lattice* $\mathcal{L}$ is defined to be a set of vertices $\{\mathbf{P}_0, \mathbf{P}_1, ..., \mathbf{P}_n\}$ and an associated simplicial complex which specifies the connectivity of the vertices[2]. A *subdivision method* applied to a lattice is a function from the set of lattices into itself. A subdivision method is usually implemented as a set of *refinement rules* which define how to generate the vertices of the resulting lattice, and also how to connect these new vertices. A set of refinement rules can be repeatedly applied to a lattice $\mathcal{L}$, creating a sequence of lattices $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, ...\}$. In many cases, this sequence can be made to converge to a region of 3-dimensional space.

To describe the components of a lattice, we will use the following terms:

- An edge of the lattice is defined by two vertices that are connected in the simplicial complex of the lattice.
- A face of the lattice is defined by a minimal connected loop of vertices.
- a cell of the lattice is the region of space bounded by a set of faces.

A control polygon has vertices and edges, a control mesh has vertices, edges and faces, and a control lattice has vertices, edges, faces and cells. In the bivariate B-spline case, each face of the control mesh is defined by four vertices, and each vertex of the mesh has connectivity four (four edges radiating from the vertex). In the trivariate case, each cell of the control lattice is defined by six faces and each face by four vertices. Each vertex has connectivity six.

[2] For consistency, we will refer to a set of points that generates a volume as a *lattice*. The set of points generating a surface will be called a *mesh*. The set of points generating a curve will be called a *control polygon*.

figures/fig1.tif

Figure 1: Lattice Structures

We allow lattices of arbitrary topology with the following properties:

- The lattice is well-connected, i.e. no vertex lies on an edge not containing that vertex.
- All cells are closed, meaning the faces comprising the cells do not form any holes. For example, a cube with one face missing is not a valid cell.
- No two cells of the lattice intersect – that is, we will not consider self-intersecting lattices.

Figure 1 illustrates two sample lattices, one based upon a parallelepiped structure and one based upon a cylindrical structure.

The uniform B-spline curves, surfaces and volumes can be defined by subdivision methods. In the curve case, the refinement rules were first presented by George Chaikin [3]. Riesenfeld [18] proceeded to show that Chaikin's curves were uniform quadratic B-spline curves. Doo and Sabin [7, 8] extended Chaikin's method to uniform quadratic B-spline surfaces and then extended the refinement rules for the quadratic case to meshes of an arbitrary topology. Catmull and Clark [2] developed a similar technique for the uniform cubic B-spline case. These methods have now come into widespread use in geometric modeling. They have been used for interpolation and fairing [10], approximation [12], and multiresolution design [16].

In this paper, we consider lattices of arbitrary topology and develop a set of refinement rules that subdivide this lattice to generate a deformable region in three-dimensional space. To generate the deformable regions, we utilize an extension of the Catmull-Clark subdivision method to volumes. In the following sections, we summarize the Catmull-Clark refinement rules for the uniform B-spline volume, along with the extensions of these methods to lattices of arbitrary topology. The complete details of the development of these refinement rules can be found in [13].

2.1 Subdivision Methods for Trivariate Cubic Uniform B-Spline Volumes

Given a control lattice $\mathcal{L}$ that defines a trivariate uniform B-spline volume, the subdivision method generates a new control lattice $\mathcal{L}_1$ which consists of the union of all the vertices generated by a binary figures/fig2.tif

Figure 2: Type-3, 4 and 5 cells generated by the subdivision process.

figures/fig3.tif

Figure 3: The type-4 cells generated by repeated subdivision.

subdivision of the trivariate volume. These points can be classified into

1. cell points – these points are the average of the vertices in the lattice that make up the cell.

2. face points – these points can be written as

$$\mathbf{F} = \frac{\mathbf{C}_0 + 2\mathbf{A} + \mathbf{C}_1}{4}$$

   where $\mathbf{C}_0$ and $\mathbf{C}_1$ are the cell points of the two adjacent hexahedral cells that contain the face and $\mathbf{A}$ is the face centroid (the average of the vertices that surround the face).

3. edge points – these points can be written as

$$\mathbf{E} = \frac{\mathbf{C}_{avg} + 2\mathbf{A}_{avg} + \mathrm{M}}{4}$$

   where $\mathbf{C}_{avg}$ is the average of the cell points for those hexahedral cells that contain this edge, $\mathbf{A}_{avg}$ is the average of the face centroids for those faces that contain this edge, and M is the midpoint of the edge.

4. vertex points – these points can be written as

$$\mathbf{V} = \frac{\mathbf{C}_{avg} + 3\mathbf{A}_{avg} + 3\mathrm{M}_{avg} + \mathbf{P}}{8}$$

   where $\mathbf{C}_{avg}$ is the average of the cell points for each of the hexahedral cells that contain this vertex, $\mathbf{A}_{avg}$ is the average of the face centroids for the faces that contain this vertex, M_{avg} is the average of the midpoints for the edges that radiate from the vertex, and $\mathbf{P}$ is the vertex itself.

At each subdivision step, a cell point is inserted into the lattice for each cell according to the first rule, a face point is inserted for each face according to the second rule, an edge point is inserted for each edge according to the third rule, and each vertex's position is recalculated according to the fourth rule. To reconnect the lattice after these rules have been applied, we first connect each new cell point to the new face points generated for the faces defining the cell. Each new face point is connected to the new edge points of the edges defining the original face. Each new edge point is connected to the two vertex points defining the original edge.

2.2 Catmull-Clark Volumes

Extension of the above rules to lattices of arbitrary topology is straightforward, using an extension of the bivariate Catmull-Clark subdivision strategy[2]. We can classify the points of the refinement into four types:

1. cell points – these points are the average of the vertices of the lattice that bound the cell.

2. face points – these points can be written as

$$\mathbf{F} = \frac{\mathbf{C}_0 + 2\mathbf{A} + \mathbf{C}_1}{4}$$

   where $\mathbf{C}_0$ and $\mathbf{C}_1$ are the cell points of the two cells that contain the face and $\mathbf{A}$ is the face centroid.

3. edge points – these points can be written as

$$\mathbf{E} = \frac{\mathbf{C}_{avg} + 2\mathbf{A}_{avg} + (n-3)\mathrm{M}}{n}$$

   where $\mathbf{C}_{avg}$ is the average of the cell points for those cells that contain this edge, $\mathbf{A}_{avg}$ is the average of the face centroids for those faces contain this edge, and M is the midpoint of the edge. n is the number of faces that contain the edge.

4. vertex points – these points can be written as

$$\mathbf{V} = \frac{\mathbf{C}_{avg} + 3\mathbf{A}_{avg} + 3\mathrm{M}_{avg} + \mathbf{P}}{8}$$

   where $\mathbf{C}_{avg}$ is the average of the cell points for each of the cells that contain this vertex, $\mathbf{A}_{avg}$ is the average of the face centroids for the faces that contain this vertex, M_{avg} is the average of the midpoints for the edges that radiate from the vertex, and $\mathbf{P}$ is the vertex itself.

These refinement rules can be applied to a lattice of arbitrary topology creating a new set of vertex points, edge points, face points and cell points. The reconnection strategy is identical to that of the trivariate uniform B-spline lattice: the new cell point are connected to the new face points generated for the faces defining the cell; the new face points are connected to the new edge points from the edges figures/fig4.tif

Figure 4: Catmull-Clark Volumes defined by a rectangular and cylindrical lattice.

defining the original face; and, each new edge point is connected to the two new vertex points from the original edge.

To describe the cell structure of the subdivided lattice, we define the *valence* of a vertex $\mathbf{V}$ within a cell $\mathbf{C}$ to be the number of edges in $\mathbf{C}$ that contain the vertex $\mathbf{V}$. Given a cell $\mathbf{C}$ of a lattice $\mathcal{L}$, consider a vertex $\mathbf{V}$ of the cell $\mathbf{C}$ of valence n. The refinement process creates a new cell from $\mathbf{V}$ that contains $2n$ 4-sided faces, 2 vertices of valence n and $2n-2$ vertices of valence 3 (see figure 2). We call these characteristic cells type-n cells. After the first subdivision, all cells can be classified as type-n cells.

In the subdivision process, a type-n cell is refined into two type-n cells and $2n-2$ type-3 cells. The type-3 cell is a hexahedral cell with 4-sided faces and 3-valence vertices. After a few subdivisions, the bulk of the lattice will consist of type-3 cells arranged in a regular pattern – that of the trivariate uniform B-spline case – except around a finite number of type-n cells where this regularity is disturbed. For $n > 3$, the number of type-n cells doubles in each application of the subdivision algorithm (see figure 3).

Figure 4 illustrates the Catmull-Clark volumes generated from the lattices shown in figure 1.

3 Boundary Control of the Subdivision Volume

Designing a lattice that represents a particular region of space is a difficult task. The free-form deformations of Sederberg and Parry [19] were based upon an initial lattice that was formed on a parallelepiped, with the deformable space filling the lattice completely. In our case, the region of space resulting from applying the trivariate Catmull-Clark subdivision method to an arbitrary lattice does not conform closely to the general shape of the lattice – shrinking away from the boundary substantially. In figure 4 for example, the cylindrical lattice does not refine into a cylindrical region of space. This feature creates an unusual burden on the designer and limits the usefulness of the technique.

To solve this problem and create a tool that will construct regions of space that are intuitive to the designer, we modify the refinement rules for those areas of the lattice that correspond to boundary surfaces, sharp edges, and corner vertices. These rules are summarized as follows:

figures/fig5.tif

Figure 5: Catmull-Clark volumes with boundary and edge control. The corner vertices are yellow, the sharp edges are red, the boundary edges are green and the internal edges are blue.

- Corner vertices are identified as those incident to only one cell of the lattice. In the refinement procedure, the position of a corner vertex does not change.
- Sharp edges are those edges joining vertices that are incident to only one or two cells of the lattice, including corner vertices. The edge and vertex points along the sharp edges of the lattice are calculated according to subdivision rules for uniform cubic B-spline curves[13].
- All other vertex, edge, and face points on the boundary are generated according to the Catmull-Clark rules for surfaces[2].
- All internal points are calculated using the Catmull-Clark rules for volumes.

Given a lattice based on a cube, these methods will generate a region of space that is the cube. In the case of the lattice approximating a cylinder, the region is flat at either end of the cylinder and rounded along the length of the cylinder as one would expect. These are shown in figure 5. The corner vertices are yellow, the sharp edges are red, the boundary edges are green and the internal edges are blue.

These techniques have been previously used by Nasri [17] for Doo-Sabin surfaces, and are similar to the techniques used by Hoppe et al. [12] in defining edges, creases, corners and darts on Loop Surfaces [15]. When added to the subdivision procedure, these new rules generate deformable regions of space that closely represent their lattice.

4 Calculating the Location of Vertices Embedded in the Deformable Space

Sederberg and Parry [19] impose the initial lattice on a parallelepiped in space and calculate the parameterization of a point within the deformable space by using the local coordinates of a point within the parallelepiped. The location of the point under the deformation is calculated by substituting these local coordinate values into the defining equation for the trivariate Bézier volume. Coquillart [5] uses a similar method, but numerical iteration is required to calculate the local coordinate, as her initial lattices are not formed as parallelepipeds. In both these cases, the cells of the lattice are hexahedral. In the case that the lattice is of an arbitrary topology and the above subdivision procedure is used, a simple trivariate parameterization is not available. Fortunately, the subdivision procedure itself can be used to establish a correspondence between points in the deformable region and points in the deformed region.

Given an initial lattice $\mathcal{L}$, the subdivision procedure generates a sequence of lattices $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4 ..., \}$ that converge to the deformable region. Each lattice in the sequence induces a partitioning of the deformable space by its cells. We select a lattice, say $\mathcal{L}_i$, that has the property that the maximum volume of the individual cells of $\mathcal{L}_i$ is small. We then identify the cell of $\mathcal{L}_i$ that contains a given point $\mathbf{P}$ and assume that $\mathbf{P}$ is deformed to a position within the corresponding cell of the deformed lattice – in the same relative position in the cell. Whereas this is an approximation, it can be made arbitrarily close to the actual deformation.

To determine the relative position of a point in the cell, we take advantage of the fact that after the first refinement all cells are type-n cells, and most are type-3 (hexahedral). In the type-3 case we can calculate a trilinear approximation of the position of the point in the cell and use this trilinear parameterization to adjust the position of the point in the deformed cell. In the type-n case, we can calculate a piecewise trilinear approximation, by partitioning the cell into tetrahedra, and use this to adjust the position in the deformed cell.

5 The Deformation Process

To deform an object, we follow the 4-step procedure outlined by Coquillart in [5]. First, the user must construct the lattice. This is normally done by utilizing an inventory of lattices and a set of tools to merge and build new lattices from this inventory. A common tool is the extrusion tool that takes a mesh and extrudes it in a specified direction to become a lattice (the cylinder of figure 1 was generated in this manner.) At the lowest level, the user is allowed to to create cells one by one, attaching them face by face to form the lattice. Boundary surfaces, sharp edges and corner vertices can be marked automatically (as in section 3), or a manual marking procedure can determine them. Once the lattice has been constructed, the user must place the lattice around the object, or the part of the object, to be deformed.

When the lattice is oriented properly, it is frozen to the object. At this time, the lattice is refined, and the number of refinement steps n is retained.

Each point embedded in the deformable region can be “tagged” with a pointer to the cell of the refined lattice that contains the point, and a finite number of parameters that defines its position in the cell.

- For a type-3 hexahedral cell, the parameters consist of a (u, v, w) triple which defines the point’s trilinear parameterization within the cell.

- For a type-n cell, a new cell point is calculated and the cell is partitioned into $4n$ tetrahedra about this cell point, each face contributing two tetrahedra to the partition (see figure 6). The parameters consist of an index into the tetrahedra containing the point and a (u, v, w) triple which defines the point’s parameterization within the tetrahedra.

Finally, the original lattice is deformed by moving one or more of its vertices. The deformed lattice is then refined n times and the tag on each point is used to obtain the corresponding cell in the deformed lattice. The vertices of this cell are used to calculate a position for the deformed point according to the parameters associated with the original.

figures/fig6.tif

Figure 6: Partitioning of a type-n cell for approximation. The green edges represent the original edges of the cell. The blue edges are generated by the tetrahedral partition.

6 Implementation Details

The data structure holding the lattice is implemented as an extension of the half-edge data structure for surfaces – much like the radial-edge structure of Weiler [20]. The primary difference between halfedge structure for a mesh representing a surface and a lattice representing a volume is that the lattice structure may have several faces that contain each edge – the mesh structure will have at most two. In addition, a real-time deformation algorithm requires that the sequence of lattices be stored hierarchically. The n subdivision steps are then executed only once during the initialization (or freezing) phase of the deformation. Subsequent deformations of the lattice then require only the recalculation of the refinement rules for the vertices in lattices $\mathcal{L}_i$ to $\mathcal{L}_n$, without recreating the lattice structure. With each subdivision, the size of the data structure nearly triples and so deformations where the user desires a very small cell size can be quite memory intensive.

Many numerical algorithms exist to generate the trilinear approximation of a point in a type-3 cell. We have utilized an adaptation of an algorithm presented by Hamann, et al. [11]. Given a point $\mathbf{P}$ in a cell, we generate a point $\mathbf{P}_0$ as the trilinear point defined by $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. We then obtain

$$(u_0, v_0, w_0) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) + (\mathbf{P} - \mathbf{P}_0)\tilde{J}^{-1}$$

where $\tilde{J}$ is the transpose of a local approximate of the Jacobian at $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ obtained by interpolating the estimates of the Jacobians at the vertices of the cell. In general,

$$(u_i, v_i, w_i) = (u_{i-1}, v_{i-1}, w_{i-1}) + (\mathbf{P} - \mathbf{P}_i)\tilde{J}_i^{-1}$$

where $\tilde{J}_i$ is obtained by performing trilinear interpolation at $(u_{i-1}, v_{i-1}, w_{i-1})$ of the Jacobians at the vertices of the cell.

The procedure is repeated until the distance between $\mathbf{P}_i$ and $\mathbf{P}$ is sufficiently small. We found that few iterations were needed as the algorithm converges quickly and the cells are generally small.

In the case of a type-n cell, we partition the cell into $4n$ tetrahedra by utilizing the cell point (average of the vertices of the cell). A simple interpolation function for tetrahedral cells can be written as

$$\mathbf{P}(u, v, w) = \mathbf{P}_0 + u(\mathbf{P}_1 - \mathbf{P}_0) + v(\mathbf{P}_2 - \mathbf{P}_0) + w(\mathbf{P}_3 - \mathbf{P}_0)$$

where $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$, are the four vertices of the tetrahedra. This can be put into matrix form and solved directly [14].

With this implementation, we have found that the algorithm executes in real time on an SGI Indigo^2 Extreme.

7 Results

The primary motivation for moving from the hexahedral topological lattices of the trivariate Bézier and B-spline representations of [5, 9, 19] was to increase the inventory of available lattices and thus the number of possible deformations. Figures 7 through 9 show the results of this algorithm with a variety of meshes and shapes.

Figure 7 exhibits a deformation by a cylindrical lattice, resulting in a surface deformation in the form of a star.

Figure 8 illustrates a complex lattice in the shape of a barbell. This lattice was generated by creating a mesh in the shape of a barbell and extruding the shape to form the lattice. The subdivision methodology automatically handles the continuity between the segments of the lattice.

Figure 9 shows a deformation applied to the arm of the "mobster" which causes the arm to lift upward, and the hand to twist toward the viewer. This is an excellent example of our technique, as it was necessary to construct the lattice about the appendage that was to be moved.

8 Conclusions

We have described a new free-form deformation technique that generalizes previous methods by allowing 3-dimensional deformation lattices of arbitrary topology. The technique uses an extension of the Catmull-Clark subdivision methodology to successively refine a 3-dimensional lattice into a sequence of lattices that converge uniformly to a region of 3-dimensional space. Deformation of the lattice then implicitly defines a deformation of this region. An underlying model can be deformed by establishing positions of the points of the model within the converging sequence of lattices, establishing the cell of the lattice that contains the point, establishing an approximation of the position of the point within the cell, and using this information to establish the new positions of these points within the deformed lattice.

This method is very powerful in that it can be applied to virtually any geometric model, as it directly modifies the vertices that define the model. The variety of lattices that can be used with this technique greatly increases the number of deformations that can be accomplished.

We have only discussed positional data of the embedded object in this paper. It is clear that the lattice could hold additional parameters. For example, we could store, in the lattice points the parameters of a solid texture. As the lattice is deformed, the texture would be deformed along with the object.

The careful reader will notice that, for the vertex points of the Catmull-Clark volume, we utilize the form

$$\mathbf{V} = \frac{\mathbf{C}_{avg} + 3\mathbf{A}_{avg} + 3\mathbf{M}_{avg} + \mathbf{P}}{8}$$

which does not contain an adjustment for the number of edges radiating from a vertex. We found that the Catmull-Clark surface methodology directly generalizes to edge points, but not to the vertex points of the refinement rules. It was our purpose to use this refinement to generate a partitioning of the deformable space by its cells, and for this purpose, this calculation appears to work very well. A detailed theoretical analysis of the continuity of the derivatives of these volumes at the extraordinary points [2, 8]. will have to be addressed in a future paper.

9 Acknowledgments

We are very grateful to the anonymous referees for their many helpful comments on the first version of this paper. We would also like to thank Bernd Hamann for pointing us toward the simple trivariate schemes that make this algorithm possible. Special thanks go to Justin Legakis for many useful critiques of the research, and who assisted us in the production of our final images. The data for the "mobster" in figure 9 was contributed via the avalon cite at *www.viewpoint.com*. The research reported here was partially supported by a grant through the University of California MICRO Program.

References

[1] Alan H. Barr. Global and local deformations of solid primitives. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 21–30, July 1984.

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

[3] G. Chaikin. An algorithm for high speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.

[4] Yu–Kuang Chang and Alyn P. Rockwood. A generalized de Casteljau approach to 3D free–Form deformation. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 257–260.

[5] Sabine Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 187–196, August 1990.

[6] Sabine Coquillart and Pierre Jancéne. Animated free-form deformation: An interactive animation technique. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 23–26, July 1991.

[7] D. Doo. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proceedings of the Int'l Conf. Interactive Techniques in Computer Aided Design*, pages 157–165, 1978.

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

[9] Josef Griessmair and Werner Purgathofer. Deformation of solids with trivariate B-splines. In *Eurographics '89*, pages 137–148. North-Holland, September 1989.

[10] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 35–44, August 1993.

[11] Bernd Hamann, Donhua Wu, and Robert J. Moorhead II. On particle path generation based on quadrilinear interpolation and Bernstein-Bézier polynomials. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):210–217, 1995.

[12] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29,*

1994), Computer Graphics Proceedings, Annual Conference Series, pages 295–302.

[13] Kenneth I. Joy and Ron MacCracken. The refinement rules for Catmull-Clark solids. Technical Report CSE-96-1, Department of Computer Science, University of California, Davis, January 1996.

[14] David N. Kenwright and Davis A. Lane. Optimization of time-dependent particle tracing using tetrahedral decomposition. In *Proceedings of Visualization '95*, pages 321–328. IEEE Computer Society, 1985.

[15] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.

[16] Mike Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Department of Computer Science and Engineering, University of Washington, Seattle, WA, June 1994.

[17] A. Nasri. Polyhedral subdivision methods for free-form surfaces. *ACM Transactions on Graphics*, 6:29–73, 1987.

[18] R. Riesenfeld. On Chaikin's algorithm. *Computer Graphics and Image Processing*, 4(3):304–310, 1975.

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

[20] Kevin J. Weiler. *Topological structures for geometric modeling*. PhD thesis, Rensselaer Polytechnic Institute, August 1986.

figures/fig7a.tif

figures/fig7b.tif

figures/fig7c.tif

Figure 7: Deforming a disk with a star-shaped lattice.

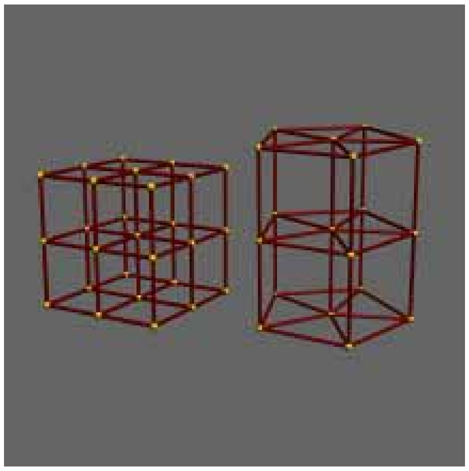

Figure 1: Lattice Structures

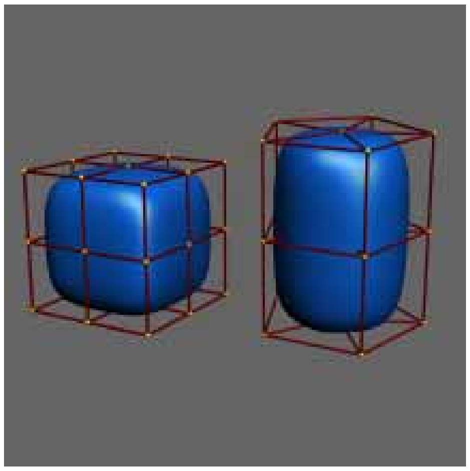

Figure 4: Catmull-Clark Volumes defined by a rectangular and cylindrical lattice.

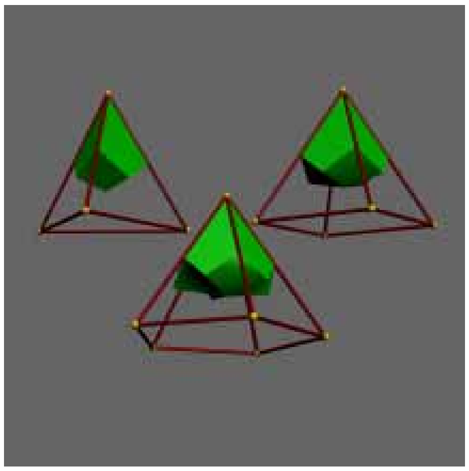

Figure 2: Type-3, 4 and 5 cells generated by the subdivision process.

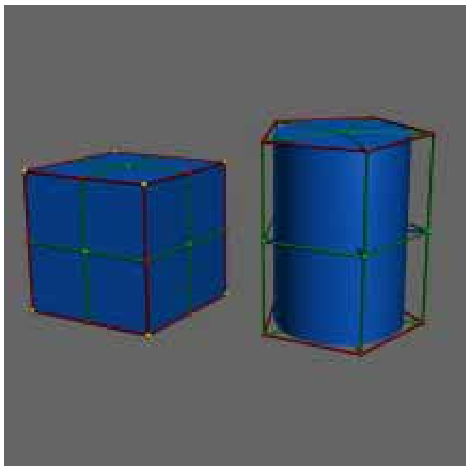

Figure 5: Catmull-Clark volumes with boundary and edge control. The corner vertices are yellow, the sharp edges are red, the boundary edges are green and the internal edges are blue.

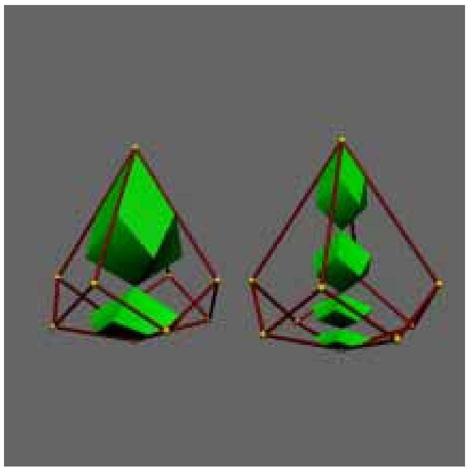

Figure 3: The type-4 cells generated by repeated subdivision.

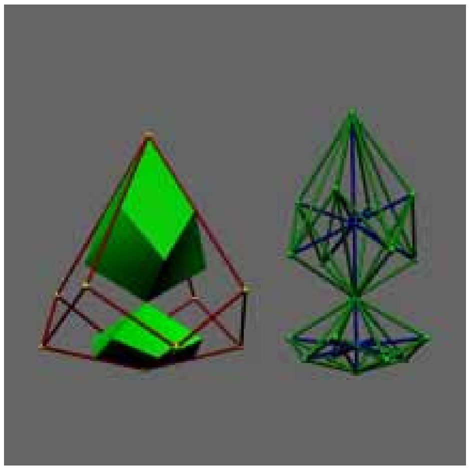

Figure 6: Partitioning of a type-*n* cell for approximation. The green edges represent the original edges of the cell. The blue edges are generated by the tetrahedral partition.

High-resolution TIFF versions of these images can be found on the CD-ROM in:
`S96PR/papers/joy`

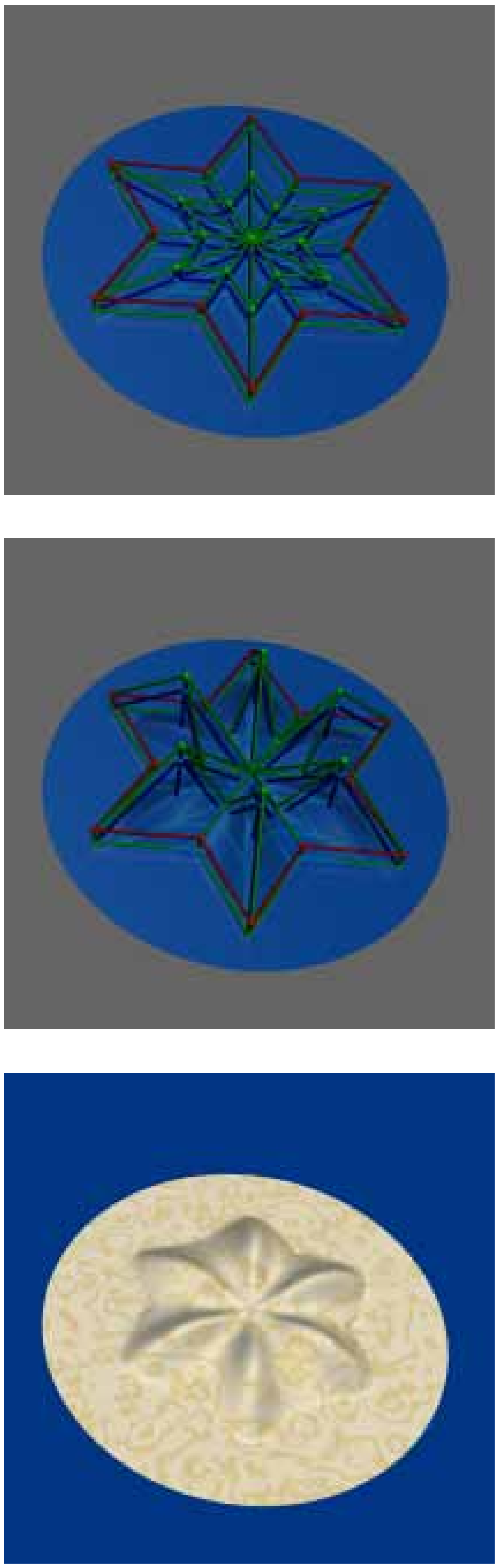

Figure 7: Deforming a disk with a star-shaped lattice.

High-resolution TIFF versions of these images can be found on the CD-ROM in:
`S96PR/papers/joy`

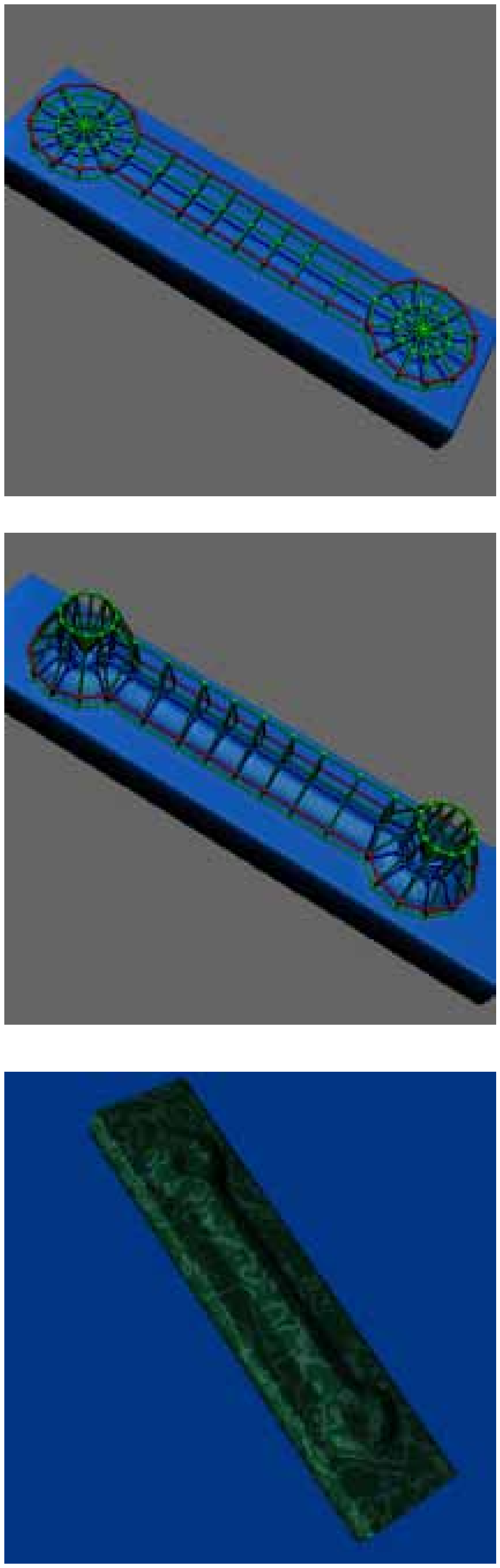
Figure 8: Deforming a block with a barbell-shaped lattice.

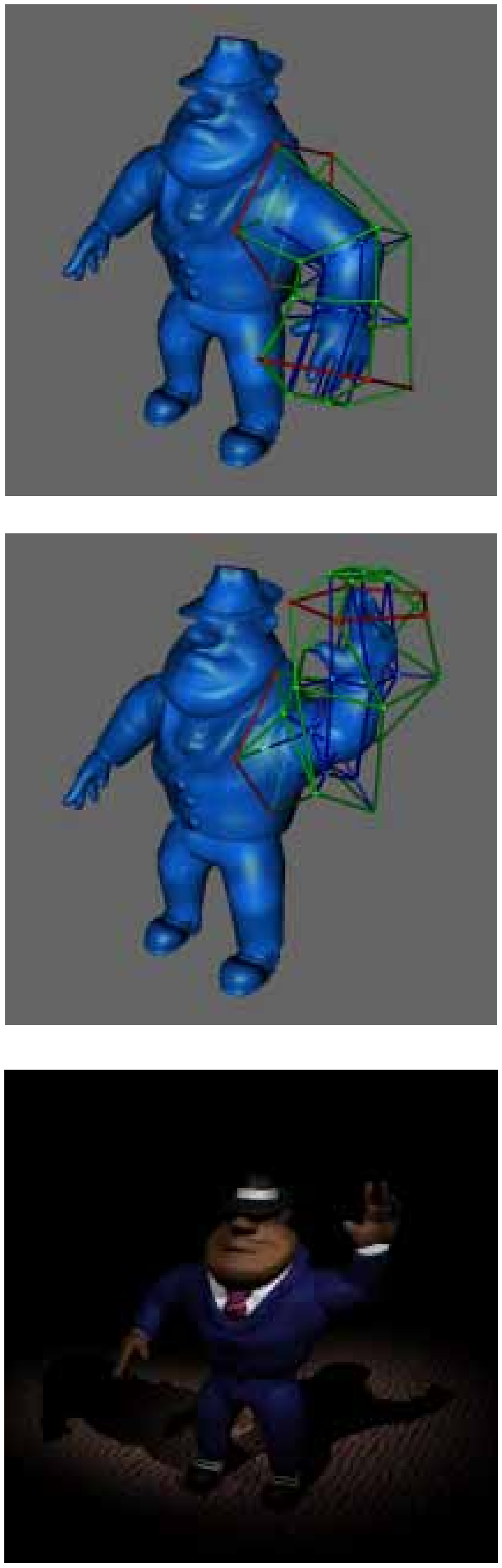
Figure 9: Deforming the mobster’s arm.

High-resolution TIFF versions of these images can be found on the CD-ROM in:
`S96PR/papers/joy`