3D Scan Conversion of CSG Models into Distance Volumes

David E. Breen[1] Sean Mauch[1] Ross T. Whitaker[2]

[1]California Institute of Technology
[2]University of Tennessee, Knoxville

Abstract

A distance volume is a volume dataset where the value stored at each voxel is the shortest distance to the surface of the object being represented by the volume. Distance volumes are a useful representation in a number of computer graphics applications. In this paper we present a technique for generating a distance volume with sub-voxel accuracy from one type of geometric model, a Constructive Solid Geometry (CSG) model consisting of superellipsoid primitives. The distance volume is generated in a two step process. The first step calculates the shortest distance to the CSG model at a set of points within a narrow band around the evaluated surface. Additionally, a second set of points, labeled the zero set, which lies on the CSG model's surface are computed. A point in the zero set is associated with each point in the narrow band. Once the narrow band and zero set are calculated, a *Fast Marching Method* is employed to propagate the shortest distance and closest point information out to the remaining voxels in the volume. Our technique has been used to scan convert a number of CSG models, producing distance volumes which have been utilized in a variety of computer graphics applications, e.g. CSG surface evaluation, offset surface generation, and 3-D model morphing.

1 Introduction

Volume graphics is a growing field which generally involves representing three dimensional objects as a rectilinear 3-D grid of scalar values, a volume dataset. Given this kind of representation numerous algorithms have been developed to process, manipulate and render volumes. Volume datasets may be generated in a variety of ways. Certain scanning devices, e.g. MRI and CT, generate a rectilinear grid of scalar values directly from their scanning process. The scalar values can represent the concentration of water or the density of matter at each grid point (voxel). Additionally, volume datasets can be generated from conventional geometric models, using a process called 3-D scan conversion.

When 3-D scan converting a geometric model to a volumetric representation it is not always clear what value should be stored at each voxel of the volume, and what that value should represent. Here, we propose the use of distance volumes. A distance volume is a volume dataset where the value stored at each voxel is the shortest distance to the surface of the object being represented by the volume. If the object is closed, a signed distance may be stored to provide additional inside-outside information. We store negative values inside the object and positive distances outside. In this paper we will show how to generate a distance volume with sub-voxel accuracy from one type of geometric model, a Constructive Solid Geometry (CSG) model, and we will also show that this type of volume representation is useful in a number of computer graphics applications, namely CSG surface evaluation, offset surface generation, and 3-D model morphing.

Constructive Solid Geometry (CSG) modeling is a well-developed technique that combines simple solid primitives using spatial boolean operations to produce complex three dimensional objects [15]. Some of the most commonly used primitives in CSG modeling are quadrics, superquadrics [1], and closed polygonal objects. These primitives can be added, subtracted, or intersected with each other to create a variety of solid geometric models. The structure that is used to represent a CSG model is ordinarily a binary tree. The leaf nodes of the tree contain solid primitives, superellipsoids in our case. A boolean operation is associated with each non-leaf node and a transformation matrix is associated with each arc of the tree. The CSG binary tree may also be derived from a directed acyclic graph.

While Constructive Solid Geometry is a powerful modeling paradigm, unfortunately its modeling representation cannot be directly displayed on today's graphics workstations. Additionally, it is a representation not suitable for many other types of modeling operations. Frequently the CSG tree or graph must first be evaluated and converted into a polygonal surface before it can be interactively displayed, processed or manipulated. We have found that first scan converting the CSG model into a distance volume allows us to perform several types of graphics operations on the model. Applying the Marching Cubes algorithm [12] to the distance volume and extracting the iso-surface at value zero produces a polygonal surface which approximates the evaluated CSG model. Extracting an iso-surface at a value other than zero produces offset surfaces to the CSG model. The distance volume may also be used to perform 3-D

[1]Computer Graphics Laboratory, Caltech MS 348-74, Pasadena, CA 91125, {david,sean}@cg.caltech.edu

[2]330 Ferris Hall, Department of Electrical Engineering, University of Tennessee, Knoxville, TN 37996-2100, rtw@utk.edu

model morphing. A deformable implicit model can utilize the distance information to change from one shape into another [21]. The derivative of the distance volume describes a field which effectively points in the direction of the embedded object's surface. Given an initial object and the distance volume representing a second object, forces are applied on each point of the initial model which push it towards the second object.

A distance volume is generated in a two step process. The first step calculates the shortest distance to the CSG model at a set of points within a narrow band around the evaluated surface. Additionally, a second set of points lying on the CSG model's surface, labeled the zero set, are computed. A point in the zero set is associated with each point in the narrow band. The narrow band and zero set are calculated with a modified version of the Constructive Cubes algorithm [3]. Once the narrow band and zero set are calculated, a *Fast Marching Method* similar to Sethian's [17], is employed to propagate the shortest distance and closest point information out to the remaining voxels in the volume. Sethian's approach has been used in the past to numerically solve partial differential equations, but we have modified it to use a heuristic rule for propagating closest point information instead of calculating distance with a finite difference scheme. The accuracy of our method depends on a discretization of the surface (resolution of the zero set) and is independent of the volume grid spacing. We therefore are able to calculate shortest distance at resolutions greater than the resolution of the final distance volume.

The original Constructive Cubes algorithm was developed to produce a polygonal approximation to a CSG model's surface. This is accomplished by first converting the CSG model into a volumetric representation, where the value stored at each voxel is a combination of the value of the inside-outside function for each of the model's primitives (superellipsoids) evaluated at the (x, y, z) location of the voxel. The inside-outside function of a superellipsoid is a non-linear function of (x, y, z), and is defined to be one on the surface of the primitive, less than one and greater than zero inside, and greater than one outside. The modifications made to the Constructive Cubes algorithm were designed to produce the initial closest point information near the CSG model's surface needed for the Fast Marching Method, which then calculates the shortest distance at the voxels away from the CSG model.

The first modification involves calculating the closest point to a single superellipsoid primitive. In general this is accomplished with an iterative minimization scheme. Given the closest points to separate geometric primitives (and therefore the shortest distances), a new set of combinations rules are applied to merge the distance values of the individual primitives to produce the closest point and shortest distance to the entire CSG model. Unfortunately, there are small regions near the CSG model where the combination rules generate invalid results, calculating a closest point which does not lie on the evaluated surface. These cases, which occur less than 1 percent of the time, can be easily detected and discarded by evaluating the closest points with the original Constructive Cubes algorithm.

The remainder of the paper first presents related work in 3-D scan conversion. The paper then details the steps required to produce a distance volume: generating the narrow band of points near the CSG model surface and zero set on the surface, followed by the propagation of the closest point information into the remaining voxels of the volume with the Fast Marching Method. The final section presents the results of our 3-D scan conversion method within three applications: CSG surface evaluation, offset surface generation, and 3-D model morphing.

2 Previous Work

3-D scan conversion takes a 3-D geometric model, a surface in 3-D or a solid model, and converts it into a 3-D volume data set [4, 9, 10, 11, 18], where voxels that contain the original surface or solid have a value of one. The remaining voxels have a value of zero. Using the volume-sampling methods of Wang and Kaufman [20] aliasing artifacts may be significantly reduced. These methods produce voxels with values between zero and one, where non-integer values represent voxels partially occupied by the original object. Scan converted primitives may then be rendered, or combined using CSG operations [7] with other scan converted primitives or acquired volume datasets. Payne and Toga [13] present a method for calculating distance volumes from a polygonal model. They use the distance volumes to perform a variety of surface manipulation tasks. Extensions to discrete distance transforms [2, 5], e.g. Chamfer methods, were considered for our work. They were deemed insufficient for our needs, because they do not provide sub-voxel accuracy.

Our algorithm differs from previous efforts to 3-D scan convert CSG models because we evaluate the parametric primitives directly and combine the results in object space, before scan conversion. This avoids the sampling errors produced when performing CSG operations on scan converted primitives, that are seen in other methods. If the primitives are first scan converted, then combined with CSG operations, errors may occur at the boundaries of the primitives, where exact surface information has been lost [20]. It is also possible to evaluate the CSG model to produce a polygonal approximation to the final object [14]. Payne and Toga's method may be used to then calculate a distance to the polygonal model. We preferred to make our calculations directly on the original model, and avoid the extra step of approximating the CSG model with polygons and the errors associated with calculating distance to a faceted model. Our approach also generates the additional closest point information, which may be used in a variety of graphics applications.

3 Generating the Distance Volume

This sections describes the two major components of our approach. The first step generates a set of closest points on the surface of the evaluated model. Additionally, it calculates the shortest distance to another set of points in a narrow band near the surface. The second step uses a Fast Marching Method to propagate this information to the remaining voxels of the distance volume.

3.1 Calculating Closest Points for the Narrow Band and Zero Set

The narrow band and zero set needed for the Fast Marching Method are generated with a modified version of the Constructive Cubes algorithm [3]. For each voxel, the algorithm involves traversing the CSG model's acyclic graph, evaluating each primitive's inside-outside function at the voxel location, and combining subcomponent values at each non-leaf node of the graph to produce a value which represents the inside-outside function for the complete model at a particular point. The original combination rules of Constructive Cubes are defined to produce a value of 1 for points on the surface of the evaluated CSG model. The Constructive Cubes algorithm was developed to calculate the final evaluated surface of a CSG model, which is produced by applying the Marching Cubes algorithm to the derived volume dataset with an iso-value of one. It was not developed to produce reasonable values away from the CSG model's surface.

The Constructive Cubes algorithm has been modified in two ways to generate the data needed for the fast marching algorithm. First, the step which evaluates the inside-outside function for each superellipsoid at a particular point has been replaced by a technique for calculating the closest point to the superellipsoid from an arbitrary point. Given the original evaluation point and the closest point on the superellipsoid, the shortest distance from the point to the superellipsoid can be calculated. The second modification involves formulating new shortest distance combination rules which are applied at each non-leaf node of the CSG graph. These are formulated in a way to produce the closest point and shortest distance to the entire CSG model from a point near the model surface.

3.1.1 Calculating the Closest Point to a Superellipsoid

The parametric equation for a superellipsoid is

$$\mathbf{S}(\eta,\omega) = \begin{bmatrix} a_1 cos^{\epsilon 1}(\eta) cos^{\epsilon 2}(\omega) \\ a_2 cos^{\epsilon 1}(\eta) sin^{\epsilon 2}(\omega) \\ a_3 sin^{\epsilon 1}(\eta) \end{bmatrix} \quad \begin{matrix} -\pi/2 \le \eta \le \pi/2 \\ -\pi \le \omega \le \pi \end{matrix} \qquad (1)$$

where η and ω are the longitudinal and latitudinal parameters of the surface, a_1, a_2, a_3 are the scaling factors in the x, y, and z directions, and $\epsilon 1$ and $\epsilon 2$ define the shape in the longitudinal and latitudinal directions [1].

The distance to the point on the surface of a superellipsoid defined at $[\eta, \omega]$ from an arbitrary point $\mathbf{P}$ is

$$d1(\eta,\omega) = ||\mathbf{S}(\eta,\omega) - \mathbf{P}||. \qquad (2)$$

Squaring and expanding Equation 2 gives

$$\begin{aligned} d2(\eta,\omega) \;=\; & (a_1 cos^{\epsilon 1}(\eta) cos^{\epsilon 2}(\omega) - P_x)^2 \\ & + (a_2 cos^{\epsilon 1}(\eta) sin^{\epsilon 2}(\omega) - P_y)^2 \\ & + (a_3 sin^{\epsilon 1}(\eta) - P_z)^2. \end{aligned} \qquad (3)$$

The closest point to the superellipsoid from an arbitrary point $\mathbf{P}$ can then be calculated by determining the values of $[\eta, \omega]$ which minimize Equation 3. In general Equation 3 is minimized with a gradient descent technique utilizing variable step-sizes. These values of $[\eta, \omega]$ may then be plugged into Equation 1 to give the closest point on the surface of the superellipsoid, which in turn may be used to calculate the shortest distance.

Several issues must be addressed when minimizing Equation 3. First, the special degenerate cases of the superellipsoid must be dealt with separately, because their surface normals are discontinuous. The most common cases are the cuboid ($\epsilon 1 = \epsilon 2 = 0$), the cylinder ($\epsilon 1 = 0, \epsilon 2 = 1$), the double cone ($\epsilon 1 = 2, \epsilon 2 = 1$), and the double pyramid ($\epsilon 1 = \epsilon 2 = 2$). The shortest distance to these primitives may be determined with non-iterative, closed form solutions.

Finding the values of η and ω at the closest point with a gradient descent technique involves calculating the gradient of Equation 3,

$$\nabla d2 = [\partial d2/\partial\eta, \partial d2/\partial\omega]. \qquad (4)$$

Unfortunately, superellipsoids have a tangent vector singularity near values of η or ω which are multiples of $\pi/2$. To overcome this problem, we reparameterize $\mathbf{S}$ by arc length [6]. That is,

$$\mathbf{S}(\eta,\omega) = \mathbf{S}(\eta(\alpha),\omega(\beta)) = \mathbf{S}(\alpha,\beta). \qquad (5)$$

where

$$\left\| \frac{\partial \mathbf{S}(\alpha,\beta)}{\partial\alpha} \right\| = 1 \text{ and } \left\| \frac{\partial \mathbf{S}(\alpha,\beta)}{\partial\beta} \right\| = 1. \qquad (6)$$

Given this we can say

$$\left\| \frac{\partial \mathbf{S}(\alpha,\beta)}{\partial\alpha} \right\| = \left\| \frac{\partial \mathbf{S}(\eta,\omega)}{\partial\eta} \right\| \cdot \left\| \frac{\partial\eta(\alpha)}{\partial\alpha} \right\| \qquad (7)$$

and

$$\left\| \frac{\partial \mathbf{S}(\alpha,\beta)}{\partial\beta} \right\| = \left\| \frac{\partial \mathbf{S}(\eta,\omega)}{\partial\omega} \right\| \cdot \left\| \frac{\partial\omega(\beta)}{\partial\beta} \right\|. \qquad (8)$$

If we assume that the arc-length parameterization is in the same direction as the original parameterization, we have

$$\frac{\partial\eta(\alpha)}{\partial\alpha} = \left\| \frac{\partial \mathbf{S}(\eta,\omega)}{\partial\eta} \right\|^{-1} \text{ and } \frac{\partial\omega(\beta)}{\partial\omega} = \left\| \frac{\partial \mathbf{S}(\eta,\omega)}{\partial\omega} \right\|^{-1}. \qquad (9)$$

Now we re-express our steepest descent (on d2) so that it is steepest with respect to the normalized parameters

$$\frac{\partial d2}{\partial\alpha} = \frac{\partial d2}{\partial\eta}\frac{\partial\eta}{\partial\alpha} = \frac{\partial d2}{\partial\eta}\left\| \frac{\partial \mathbf{S}(\eta,\omega)}{\partial\eta} \right\|^{-1} \qquad (10)$$

and

$$\frac{\partial d2}{\partial\beta} = \frac{\partial d2}{\partial\omega}\frac{\partial\omega}{\partial\beta} = \frac{\partial d2}{\partial\omega}\left\| \frac{\partial \mathbf{S}(\eta,\omega)}{\partial\omega} \right\|^{-1}. \qquad (11)$$

We now can use the gradient of the reparameterized $d2$,

$$\nabla d2' = [\partial d2/\partial\alpha, \partial d2/\partial\beta], \qquad (12)$$

to find the closest point with greater stability.

The general formulation of Equation 12 significantly simplifies for values of η and ω near multiples of $\pi/2$. Instead of deriving and implementing these simplifications for all regions of the superellipsoid we chose to only perform the calculation in the first octant ($0 \le \eta \le \pi/2, 0 \le \omega \le \pi/2$). Since a superellipsoid is 8-way symmetric, point $\mathbf{P}$ may be reflected into the first octant, the minimization performed, and the solution point reflected back into $\mathbf{P}$'s original octant.

3.1.2 Combining Shortest Distance Calculations

The CSG graph is processed in a depth-first manner. The closest point on and shortest distance to individual superellipsoids are calculated at the leaf nodes. The results from the non-leaf nodes' subcomponents (A and B) are then combined. Since the subcomponents may be combined with a variety of boolean operations (union, intersection and difference) just choosing the closest point to the subcomponents does not produce the correct result. Similar to CSG classification methods [19], a set of combination rules are utilized at each non-leaf node to evaluate the complete model, and are defined in Tables 1, 2, and 3. The rules are formulated for combining signed distance values which have no predefined limits. The values of A and B are negative inside an object and positive outside. Combination decisions are based on the signed distances computed from the non-leaf node's subcomponents. Additionally the closest point to the tested point is appropriately updated at each non-leaf node, until the complete model has been evaluated.

The entries in the tables have the following meanings. The IN conditions are used when the point being tested against subcomponent A or B is inside the subcomponent, and the shortest distance to that subcomponent is negative. The OUT conditions are used when the point being tested against subcomponent A or B is outside the subcomponent, and the shortest distance to that subcomponent is positive. The ON conditions are used when the point being tested against subcomponent A or B is on the subcomponent, and the shortest distance to that subcomponent is zero. MAX states that the two values may be combined by taking the maximum of the values

	A ∪ B	B IN	B OUT	B ON
	IN	MIN	A	A
A	OUT	B	MIN	B
	ON	B	A	A

Table 1: Union combination rules.

	A ∩ B	B IN	B OUT	B ON
	IN	MAX	B	B
A	OUT	A	MAX	A
	ON	A	B	A

Table 2: Intersection combination rules.

	A - B	B IN	B OUT	B ON
	IN	-B	MAX(A,-B)	B
A	OUT	MAX(A,-B)	A	A
	ON	-B	A	A

Table 3: Signed distance difference combination rules.

	A - B	B IN	B OUT	B ON
	IN	2-B	MAX(A,1/B)	B
A	OUT	MIN(A,2-B)	A	A
	ON	2-B	A	A

Table 4: Inside-outside difference combination rules.

returned by evaluating A and B. MIN states that the two values may be combined by taking the minimum of the two. 'A' states that the values of A and B are combined by taking the shortest distance to A. 'B' states that the values of A and B are combined by taking the shortest distance to B. '-B' states that the values of A and B are combined by taking the negative of B. MAX(A,-B) states that the combination is produced by taking the maximum of the value of A and the negative of B.

The combination rules for union and intersection are the same as the ones described in the original Constructive Cubes paper. A detailed explanation of these rules may be found in [3].

The combination rules for difference (A-B) have been changed to work with signed distances, and may be explained with Figure 1. Point P6 is the IN-IN condition. The shortest distance to the evaluated surface is the shortest distance to B. Since P6 is inside of B the shortest distance to B is negative. P6 is outside the evaluated model, and therefore must be negated to produce the correct signed distance. In the IN-OUT case, A is negative and B is positive. Therefore MAX(A,-B) compares two negative numbers, producing the number with the smallest absolute value. The correct answer for P1 is A, while the correct answer for P4 is -B. P5 is in A and on B. B or zero is the correct result for this combination. The OUT-IN combination rule is also MAX(A,-B). In this case A is positive and B is negative, and it compares two positive numbers, producing the distance with the largest absolute value. The correct answer at P7 is -B, recalling that B is negative, and must be negated to produce the correct signed result. The correct answer at P3 is A. P10 is the OUT-OUT condition, with A providing the closest point to the evaluated model. P12 is the OUT-ON condition, with A also being the correct answer. P8 represents the ON-IN condition. A is zero in this case, and B is negative. B is negated to produce the correct signed distance. P2 is the ON-OUT condition, which returns A, which is zero. The ON-ON case occurs at the intersection point of

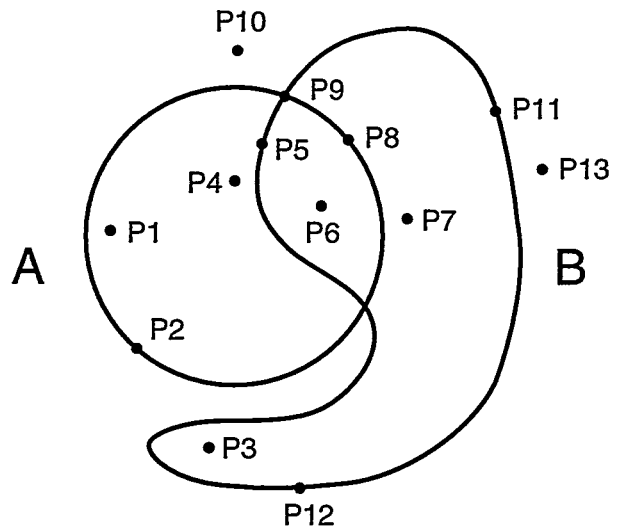

Figure 1: Evaluation points for a CSG model.

the two objects (P9), and returns A, which is zero.

It is not always possible to determine the closest point to a CSG model given the closest points to the primitives which comprise it. (From our experience this occurs in significantly less than one percent of the narrow band calculations.) As seen in Figure 1, no valid result can be calculated for P13, when evaluating A - B. Both of the closest points to A and B are not on the final evaluated model. Similarly, no valid solution can be generated at P6 when evaluating A ∪ B. The closest points to both A and B are on interior curves, which will not be a part of the final evaluated model. These invalid points which do not lie on the final evaluated CSG model can be easily removed from the zero set by evaluating them with the original Constructive Cubes algorithm. If the evaluation returns a value within a small ϵ of 1 (i.e., the point lies on the surface of the evaluated model), the point is retained. Otherwise it is discarded. Since the sampling of the zero set is quite dense, no adverse effects have been noted from discarding the occasional incorrect closest point. The first step of the algorithm produces a satisfactory distribution of closest points on the evaluated surface of the CSG model.

Even though the range of the superellipsoid's inside-outside $[0, \infty]$ is different than the signed distance $[-\infty, \infty]$, the inside-outside function combination rules for union and intersection used in the original Constructive Cubes algorithm are the same as the rules for combining signed distances, and are given in Tables 1 and 2. The inside-outside difference combination rules are different than the signed distance combination rules, and are given in Table 4.

3.2 Fast Marching Method For Computing Closest Points

We present a Fast Marching Method for computing the approximate closest point to a surface from the points in a regular grid. It is an approach based on the work of Sethian [16, 17]. His approach has been used in the past to numerically solve partial differential equations, but we have modified it to use a heuristic rule for propagating closest point information instead of calculating distance with a finite difference scheme. The accuracy of the method depends on a discretization of the surface and is independent of the volume grid spacing, allowing us to calculate distance to sub-voxel accuracy.

3.2.1 The Eikonal Equation and the Fast Marching Level Set Method

Let $u(x, y, z)$ denote the signed distance from the closed surface S. u satisfies the Eikonal equation,

$$|\nabla u| \equiv \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial z}\right)^2} = 1, \quad \text{subject to} \quad u\big|_S = 0. \qquad (13)$$

The characteristics of Equation 13 are straight lines that are normal to S and point in the direction of increasing distance. For each point (x, y, z) in space, there is a line segment from the surface to the point that is a characteristic of the entropy-satisfying solution of the Eikonal equation. The point (x, y, z) and the closest point on the surface S are the endpoints of this line segment.

Sethian [16, 17] has developed a Fast Marching Level Set Method to solve the Eikonal equation,

$$|\nabla u| f(x, y, z) = 1, \quad \text{subject to} \quad u\big|_S = g(x, y, z),$$

in the case that f is either always positive or negative. The method uses an upwind, viscosity solution, finite difference scheme to numerically solve this equation. For $f(x, y, z) = 1$ and $g(x, y, z) = 0$, the solution gives the signed distance from the surface S. The initial condition $u\big|_S = 0$ is specified by giving the value of u on a narrow band of points around the surface S. The distance values in the remainder of the volume are computed by pushing this narrow band outward.

3.2.2 Closest Point Calculation Overview

To calculate the closest points to a surface on a regular grid, we utilize Sethian's Fast Marching Method, but instead of using a finite difference scheme to compute distance, we use a heuristic algorithm to propagate closest points information. Instead of specifying the distance for the points in the narrow band as an initial condition, we specify the closest points to the surface. In one step of the closest points method:

1. The point gp with the smallest distance is removed from the narrow band and it's value is frozen.
2. Points are added to the narrow band to maintain unit thickness.
3. The closest points of the neighbors with larger distances than gp are recomputed using the closest point information from gp.

The closest points method is based on the following idea. The closest point on the surface to a point in the grid is usually close to one of the closest points of its neighbors in the grid. Thus if one knows the closest points of the neighbors of a grid point gp, one can compute an approximate closest point for gp by assuming that it is near one of the closest points of its neighbors. This is only a heuristic, and in Figure 2 we see cases in two dimensions for which the heuristic succeeds and fails. In the cases where the heuristic fails to determine the correct closest point, it still gives a reasonable approximation of the distance. The heuristic may fail if the characteristics from several different portions of the surface S intersect near gp. Fortunately, if the heuristic fails at a point, this mistake is usually not propagated outward to increasing distances. This is because "information" in the Eikonal equation and the closest points method is propagated along characteristics of Equation 13. Where characteristics collide, information goes into the shock and is lost.

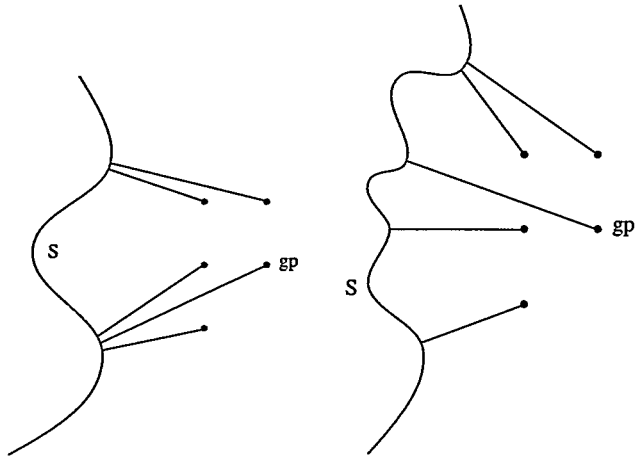

Figure 2: The Closest Point Heuristic.

3.2.3 Terminology

Let the *distance volume* be the $N \times N \times N$ grid that spans the space around the scan converted object. We will refer to points in the distance volume with (i, j, k) coordinates. Let the *zero set grid* be an $M \times M \times M$ uniform grid that spans the same Cartesian domain. We refer to the ratio M/N as the *super-sampling factor* of the zero set grid. In most cases the zero set grid is finer than the distance volume grid, providing distance calculations with sub-voxel accuracy. We will refer to points in the zero set grid with (I, J, K) coordinates. For any grid point, the closest point is defined as the Cartesian coordinates of the point on the CSG model surface that is closest to that grid point.

3.2.4 Initial Data

The fast marching algorithm takes as input: a set of points in the distance volume that forms a *narrow band* around the CSG model surface and a point sampling of the surface. The narrow band contains all the points in the distance volume having the property that a neighbor of the point has opposite inside/outside status.[1] We generate the narrow band by evaluating the inside/outside status [19] of all the grid points of the distance volume, and note where inside/outside transitions occur. For the points in the narrow band we must supply the (i, j, k) coordinates of the points and their inside/outside status. The narrow band is used as a starting point for propagating the closest point information outward and inward to the rest of the points in the distance volume. Note that specifying the inside/outside status of the points in the narrow band determines the inside/outside status of the other points in the grid.

During this stage of our calculations the CSG model surface is represented with a set of points that lie on the surface. This set of points will be called the *zero set*, as they are points lying on the isosurface of zero distance. The zero set is made by first constructing a thin band of points in the zero set grid that surrounds the CSG model surface. This set of grid points will be called the *zero band*. The zero set is the set of closest points on the model surface to the grid points in the zero band. The method used to calculate the zero set has been described in the previous section. Given a point p in the zero set that is closest to the grid point (i, j, k) in the zero band, one can determine all the points in the zero set that lie in

[1] In three dimensions *neighbor* means one of the 26 locations surrounding each grid point.

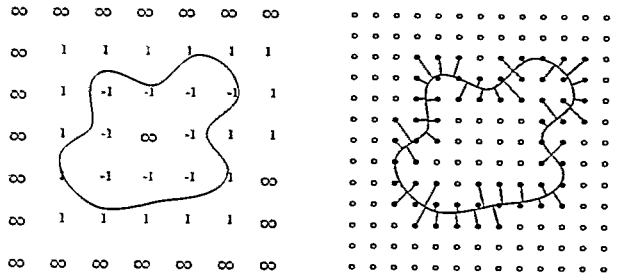

Figure 3: Initial Data for the Fast Marching Algorithm.

a neighborhood of p by determining all the points in the zero band in a neighborhood of (i, j, k). As input to the algorithm, we must supply the (I, J, K) coordinates of the grid points in the zero band and the corresponding (x, y, z) coordinates of the points in the zero set. In Figure 3 the initial data is shown pictorially in 2 dimensions.

3.2.5 Propagating the Closest Point Data

Initially, we have the closest points data in the zero band that surrounds the surface. We use the closest points data in the zero band to determine the closest points in the narrow band of the distance volume and then march the narrow band outward and inward to calculate the closest points in the rest of the distance volume. Consider a point gp that neighbors the band and whose closest point is unknown. The closest point of gp is probably close to one of the closest points of its neighbors in the band. Thus for each neighbor of gp in the band, we recompute the distance of gp by considering points that are near the closest points of that neighbor. First we will present the marching algorithm that moves the band outward and then inward. Next, we will show the algorithm for recomputing the distance at a point gp, given the closest point of one of its neighbors.

Let in_out_{ijk} denote the inside/outside status for a point in the distance volume; $+1$ for outside, -1 for inside. Let grid_{ijk} denote the computed distance at a distance volume grid point. A value of ∞ indicates that the distance has not yet been computed. Let source_{ijk} denote the point in the zero set Z from which this distance was computed.

Initially: The closest point to each (I, J, K) in the zero band is known. For each (i, j, k) in the narrow band $\text{grid}_{ijk} = \text{in\_out}_{ijk}$. For each point not in the narrow band grid_{ijk} and in_out_{ijk} are set to be undefined. The closest points of the zero band are used to generate approximate closest points for the narrow band. Below is the fast marching, closest points algorithm.

```
begin
// March forward to find positive distances.
put each point with a non-negative, finite
 grid_ijk in the set U;
while U ≠ ∅
   remove the grid point gp with the smallest
    distance from U;
   for each of the 26 neighbors of gp
      if the source of the neighbor is unknown
         add that neighbor to U;
      if the distance of the neighbor is
       larger than the distance of gp
         recompute the neighbor's distance
          using gp's source s;
end
```

Next the narrow band is marched backward to compute the closest points with negative distance. Below is the algorithm to recompute the distance grid_{ijk} to the distance grid point gp, using a zero set source s. Let (I, J, K) be the coordinates in the zero band for which s is the closest point. The user chooses the search radius parameter R. This is the radius of a cube around the point (I, J, K) in the zero band that defines a neighborhood on the surface around the point s. The parameter, $\sigma = 2 * R + 1$ is the diameter of the cube. When recomputing the distance, all the points in the zero set in a neighborhood around s are considered as possible closest points.

```
begin
for each grid point (l,m,n) in a σ×σ×σ cube
 surrounding (I,J,K)
   t ∈ Z is the closest point to (l,m,n);
   calculate the distance from gp to t;
grid_ijk = minimum of the σ^3 computed distances;
source_ijk = the source of this minimum distance,
 (an element of Z);
end
```

From experience we have found that for most surfaces, a search radius R of half the super-sampling factor of the zero set grid will provide satisfactory closest points information to the set Z. Finally, note that since the zero band is of small constant thickness, the number of points in the zero band in the $\sigma \times \sigma \times \sigma$ cube is $\mathcal{O}(\sigma^2)$.

3.3 Computational Complexity

There are N^3 grid points in the distance volume. Each distance grid point is removed from the narrow band once, giving us a factor of N^3. At any point in the algorithm, there are $\mathcal{O}(N^2)$ points in the narrow band. There are $2P$ nodes in the binary tree representing the CSG model, where P is the number of superellipsoids in the model. Each node of the model must be evaluated (in constant time) to determine if a particular grid point is inside or outside the model. Determining which grid points are in the initial narrow band requires $\mathcal{O}(N^3 P)$ operations. Determining the closest point on the CSG model from a particular grid point is also an $\mathcal{O}(P)$ operation. This is only computed on the points of the zero band. Calculating the zero set requires $\mathcal{O}(M^2 P)$ operations. Unfortunately it is difficult to characterize the amount of time needed to calculate the closest point to each superellipsoid, since each one is evaluated with an iterative technique. This calculation typically requires approximately 30 iterations in our variable step-size gradient descent routine.

The cost of adding and deleting elements from the narrow band is proportional to the logarithm of the number of points in the narrow band. This gives us a factor of $\mathcal{O}(\log N)$. The computational cost of recomputing the distance for a given grid point is proportional to the number of zero band points in a $\sigma \times \sigma \times \sigma$ cube neighborhood of a point s in the zero band. This gives us a factor of $\mathcal{O}(\sigma^2)$. Thus the overall computational complexity of the fast marching algorithm is $\mathcal{O}(N^3 \sigma^2 \log N)$.

4 Results

A number of moderately complex CSG models have been scan converted into distance volumes with our approach. Each of the CSG models consist of superellipsoids which have been unioned, intersected, and/or differenced to produce the final shapes. The resulting distance volumes have been used to generate an evaluated surface of the model, as well as offset surfaces. Additionally, the volumes have been utilized to morph one model into another [21]. Figure 7 presents an improved evaluated surface of a CSG model

Figure 4: Offset surface from the X-29 distance volume.

Figure 5: A 3-D morphing between an X-29 and a dart.

Figure 5: A 3-D morphing between an X-29 and a dart. (cont.)

similar to the one included in the first Constructive Cubes paper. The polygonal model is generated by applying the Marching Cubes algorithm [12] to a distance volume of dimension $195 \times 90 \times 120$, producing an iso-surface of value zero. The result presented here is superior to the one presented in the original paper. See Figure 6. Since the Marching Cubes algorithm linearly interpolates the iso-surface value between volume grid points (voxels), utilizing shortest Euclidean distance instead of the non-linear superellipsoid inside-outside functions values provides the linear relationship necessary for correctly calculating the iso-surface intersection point between each voxel in the Marching Cubes algorithm, and for properly combining subcomponent values in the Constructive Cubes algorithm.

Figure 8 presents the zero iso-surface of a model of an X-29 jet fighter, consisting of 38 primitives and also generated from a $96 \times 192 \times 240$ distance volume. Figure 9 presents the iso-surface of value zero of a dart model, consisting of 21 primitives, generated from a $96 \times 192 \times 240$ distance volume. These volume resolutions were chosen because they produced satisfactory results given the cost in time (several hours) and memory ($\sim$17 MBtyes) to produce them. The excessive time needed to produce our results is significantly affected by the message-passing overhead imposed by the object-oriented environment used to prototype our algorithms [8]. We believe that the processing times can be improved by at least an order magnitude if the algorithm is custom coded in a conventional programming environment. Figure 4 presents an offset surface to the X-29 model. This is the iso-surface at value 0.5

running through the X-29's distance volume. The colors in Figures 7, 8 and 9 were generated with the closest point information that is maintained with the shortest distance information. Discussion of this aspect of our work is beyond the scope of this paper. Figure 5 presents four intermediate shapes produced while morphing the X-29 model in Figure 8 into the dart model in Figure 9. The X-29 model follows the dart's shortest distance information to the surface of the dart [21].

5 Conclusion

We have described a technique for generating a distance volume with sub-voxel accuracy from one type of geometric model, a CSG model consisting of superellipsoid primitives. The distance volume is generated in a two step process. The first step calculates the shortest distance to the CSG model at a set of points within a narrow band around the evaluated surface. Additionally, a second set of points, labeled the zero set, which lies on the CSG model's surface are computed. A point in the zero set is associated with each point in the narrow band. Once the narrow band and zero set are calculated, a Fast Marching Method is employed to propagate the shortest distance and closest point information out to the remaining voxels in the volume. Our technique has been used to scan convert a number of CSG models, producing distance volumes which have been utilized in a variety of computer graphics applications, e.g. CSG surface evaluation, offset surface generation, and 3-D model morphing.

6 Acknowledgements

We would like to thank Dr. Alan Barr and the other members of the Caltech Computer Graphics Group for their support and assistance. Timothy Doyle created the model used in Figure 9. This work was financially supported by the National Science Foundation (ASC-89-20219), as part of the STC for Computer Graphics and Scientific Visualization; the National Institute on Drug Abuse, the National Institute of Mental Health and the NSF, as part of the Human Brain Project; the Office of the Director of Defense Research and Engineering, and the Air Force Office of Scientific Research (F49620-96-1-0471), as part of the MURI program; and the Volume Visualization Program of the Office of Naval Research (N00014-97-0227). Additional equipment grants were provided by Silicon Graphics, Hewlett-Packard, IBM, and Digital Equipment Corporation. This work was initially funded by the former shareholders of the European Computer-Industry Research Centre: Bull SA, ICL PLC, and Siemens AG.

REFERENCES

[1] A. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

[2] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.

[3] D. E. Breen. Constructive Cubes: CSG evaluation for display using discrete 3D scalar data sets. In Werner Purgathofer, editor, *Eurographics '91*, pages 127–142. North-Holland, September 1991.

[4] D. Cohen and A. Kaufman. Scan-conversion algorithms for linear and quadratic objects. In A. Kaufman, editor, *Volume Visualization*, pages 280–301. IEEE Computer Society Press, 1990.

[5] D. Cohen-Or, D. Levin, and A. Solomivici. Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, 1998.

[6] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, Englewood Cliffs, NJ, 1976.

[7] S. Fang and R. Srinivasan. Volumetric-CSG – a model-based volume visualization approach. In *Proceedings of the 6th International Conference in Central Europe on Computer Graphics and Visualization*, 1998.

[8] P. Getto and D. Breen. An object-oriented architecture for a computer animation system. *The Visual Computer*, 6(2):79–92, March 1990.

[9] M.W. Jones. The production of volume data from triangular meshes using voxelisation. *Computer Graphics Forum*, 15(5):311–318, 1996.

[10] A. Kaufman. An algorithm for 3D scan-conversion of polygons. In G. Marechal, editor, *Eurographics '87*, pages 197–208. North-Holland, August 1987.

[11] A. Kaufman. Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 171–179, July 1987.

[12] W.E. Lorensen and H.E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

[13] B. Payne and A. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, 1992.

[14] A. Requicha and H. Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(1):30–44, 1985.

[15] A. A. G. Requicha and H. B. Voelcker. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 2(2):9–22, March 1982.

[16] J.A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Science*, volume 93 of 4, pages 1591–1595, 1996.

[17] J.A. Sethian. *Level Set Methods*. Cambridge University Press, Cambridge, UK, 1996.

[18] N. Shareef and R. Yagel. Rapid previewing via volume-based solid modeling. In *Proceedings of the 3rd Symposium on Solid Modeling and Applications*, pages 281–292, May 1995.

[19] R. B. Tilove. Set membership classification: a unified approach to geometric intersection problems. *IEEE Trans. Comput.*, C-29:874–883, October 1980.

[20] S. Wang and A. Kaufman. Volume-sampled 3D modeling. *IEEE Computer Graphics and Applications*, 14(5):26–32, September 1994.

[21] R. Whitaker and D. Breen. Level-set models for the deformation of solid objects. In *Proceedings of the Third International Workshop on Implicit Surfaces*, pages 19–35. Eurographics Association, June 1998.