3D Scan Conversion of CSG Models into Distance, Closest-Point and Colour Volumes

D. Breen and S. Mauch and R. Whitaker

April 6, 2000

1 Introduction

Volume graphics is a growing field that generally involves representing three dimensional objects as a rectilinear 3D 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 3D scan conversion.

When 3D 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 chapter we will describe an algorithm for generating a distance volume with subvoxel accuracy from one type of geometric model, a Constructive Solid Geometry (CSG) model, and we will show that this type of volume representation is useful in a number of computer graphics applications, namely CSG surface evaluation, offset surface generation, and 3D 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 [1]. Some of the most commonly used primitives in CSG modeling are quadrics, superquadrics [2], 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 [3] to the distance volume and extracting the isosurface at value zero produces a polygonal surface which approximates the evaluated CSG model. Extracting an isosurface at a value other than zero produces offset surfaces to the CSG model. The distance volume may also be used to perform 3D model morphing. An active implicit model can utilise the distance information to change from one shape into another [4]. Given a volumetric representation of an initial object and the distance volume representing a second object, the surface of the initial object may be deformed into the surface of the second object.

Most volume datasets contain scalar values at each voxel, but multi-dimensional volumetric models are becoming more prevalent. In these models a vector or tensor is stored at each grid point. This multi-dimensional data can also be obtained from scanning devices, and generated from numerical simulations, similar to their scalar counterparts. Additionally, specific types of vector data can be produced during the 3D scan conversion process, in our case closest-point and colour data. A closest-point volume stores, at each voxel, the \([X, Y, Z]\) coordinates of the closest point on the scan-converted object’s surface from the voxel. A colour volume contains, at each voxel, the \([R, G, B]\) colour of the object at the \([X, Y, Z]\) closest point stored in the same voxel in the closest-point volume. The closest-point and colour volume data may be used to colour-shade polygonal surfaces extracted from the associated distance volume. The derived polygonal surface may be colour shaded by conceptually imbedding it within the colour and closest-point volumes. When the polygonal surface is rendered, the colour value at any location \([X, Y, Z]\) on the model's surface may be retrieved as the interpolated value in the colour volume at \([X, Y, Z]\). If the colour is not constant within a small region around \([X, Y, Z]\) in the colour volume, the information in the closest-point volume may be used to supersample the colour in the associated surface region on the original CSG model in order to produce an average colour value for the point \([X, Y, Z]\) on the polygonal model.

Distance, closest-point and colour volumes may be generated in a two-step process. The first step begins by computing a set of points lying on the CSG model's surface, labeled the zero set. The colour of the CSG model at each point is also calculated. The closest point in the zero set and its colour is associated with each grid point in a one-voxel narrow band around the evaluated CSG surface. The shortest distance from each narrow band grid point to the CSG model may now be computed. Once the narrow band and zero set are calculated, a Fast Marching Method similar to Sethian's [5], is employed to propagate the shortest-distance, closest-point and colour 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 and colour information instead of calculating distance with a finite difference scheme. The accuracy of our method depends on a discretisation of the surface (resolution of the zero set) and is independent of the final volume grid spacing. We therefore are able to calculate shortest distance at resolutions greater than the resolution of the final distance volume.

The zero set points are produced by performing closest-point calculations from a grid of user-specified resolution within the narrow band. We utilise a variation of the Constructive Cubes algorithm [6] and Tikove's CSG classification algorithm [7] to perform this computation. The first step of the CSG closest-point computation involves calculating the closest point to a single superellipsoid primitive, as well as the colour at the closest point. In general this is accomplished with an iterative minimisation scheme. Given the closest points to separate geometric primitives (and therefore the shortest distances), a set of combinations rules are applied to merge the distance values of the individual primitives to produce the closest point and colour on the evaluated CSG model. Unfortunately, there are small regions near the CSG model surface where this "calculate-and-combine" approach generates invalid results. These cases can be easily detected and discarded.

The remainder of the chapter first presents related work in 3D scan conversion and colour shading
polygonal models. The chapter then details the steps required to produce a distance, closest-point and
colour volumes: 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 and colour information into the remaining
voxels of the volume with the Fast Marching Method. We then describe our colour shading method
which utilises the closest-point and colour volumes to calculate surface colour on the polygonal models
extracted from the distance volumes. The final section presents the results of our 3D scan conversion
method within three applications: CSG surface evaluation, offset surface generation, and 3D model
morphing.

2 Previous Work

3D scan conversion takes a 3D geometric model, a surface in 3D or a solid model, and converts it into
a 3D volume data set [8, 9, 10, 11, 12], 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 [13] aliasing artefacts 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
[14] with other scan converted primitives or acquired volume datasets. Payne and Toga [15] 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 [16, 17],
e.g. Chamfer methods, were considered for our work. They were deemed insufficient for our needs,
because they do not provide subvoxel accuracy.

Our scan conversion algorithm differs from previous efforts to 3D 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 [13]. It is also possible to evaluate the CSG model to produce a
polygonal approximation to the final object [18]. 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 and colour information, which may be used in a variety of graphics applications.

The relevant colour-shading work focuses on determining the colour of surfaces with complex or no
parameterisation. Much of this work deals with texture mapping implicit or complex, unparameterised
surfaces. Maillot et al. [19] define a deformation process based on energy minimisation that lessens the
distortion of texture maps interactively placed on complicated surfaces. Litwinowicz and Miller [20]
provide additional improvements to this work. Agrawala et al. [21] present an interactive technique for
painting unparameterised 3D polygonal meshes. Pedersen describes a technique for applying texture
tmaps to implicit surfaces [22], as well as a general framework for texturing other kinds of surfaces
[23]. Smets-Solanes [24] places texture maps on animated implicit objects by wrapping them in a
virtual skin and deforming the skin with vector fields as the underlying objects change over time.
Both Tigges and Wyvill [25], and Zonenschein et al. [26] trace particles through a gradient vector
field in order to produce the parameterisation needed for texture mapping implicit surfaces. Shibolet
and Cohen-Or [27] extend Bier and Sloan’s Two-Part Texture Mapping technique [28] with a discrete
dilation process in order to texture map concave voxel-based objects. Our work is similar to many
of these techniques, except that we do not just use 2D textures to colour our models. We utilise the
original 3D CSG model along with solid colour, 2D image textures and 3D procedural textures defined
on the model to calculate the anti-aliased colour of a polygonal surface derived from the CSG model's
distance volume.

3 Generating the Distance, Closest-Point and Colour Volumes

This section describes the two major components of our scan conversion algorithm. The first step generates a set of closest points and associated colours on the surface of the evaluated CSG 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, closest-point and colour volumes.

3.1 Calculating Closest Points, Shortest Distances and Colours 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 [6] and Tilove’s CSG classification algorithm [7]. The algorithm involves traversing the CSG model’s acyclic graph, calculating each primitive’s closest-point and colour value at the voxel location, and combining sub-component values based on the shortest distance at each non-leaf node of the graph to produce the closest point, shortest distance and colour for the complete model at a particular voxel.

3.1.1 Calculating the Closest Point to a Superellipsoid

The parametric equation for a superellipsoid is

\[
\mathbf{S}(\eta, \omega) = \begin{bmatrix}
    a_1 \cos^\epsilon_1(\eta) \cos^\epsilon_2(\omega) \\
    a_2 \cos^\epsilon_1(\eta) \sin^\epsilon_2(\omega) \\
    a_3 \sin^\epsilon_1(\eta)
\end{bmatrix}
\]

\[-\pi/2 \leq \eta \leq \pi/2, \quad -\pi \leq \omega \leq \pi\]

where \(\eta\) and \(\omega\) 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 [2].

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

\[
d_1(\eta, \omega) = |\mathbf{S}(\eta, \omega) - \mathbf{P}|.
\]

Squaring and expanding Equation 2 gives

\[
d_2(\eta, \omega) = (a_1 \cos(\eta) \cos^\epsilon_2(\omega) - P_x)^2 \\
+ (a_2 \cos(\eta) \sin^\epsilon_2(\omega) - P_y)^2 \\
+ (a_3 \sin(\eta) - P_z)^2.
\]

The closest point to the superellipsoid from an arbitrary point \(\mathbf{P}\) can then be calculated by determining the values of \([\eta, \omega]\) which minimise Equation 3. In general Equation 3 is minimised with a gradient descent technique utilising 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 minimising 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 ($e_1 = e_2 = 0$), the cylinder ($e_1 = 0, e_2 = 1$), the double cone ($e_1 = 2, e_2 = 1$), and the double pyramid ($e_1 = e_2 = 2$). The shortest distance to these primitives may be determined with non-iterative, closed form solutions.

Finding the values of $\eta$ and $\omega$ at the closest point with a gradient descent technique involves calculating the gradient of Equation 3,

$$\nabla d^2 = \left[ \frac{\partial d^2}{\partial \eta}, \frac{\partial d^2}{\partial \omega} \right].$$

Unfortunately, superellipsoids have a tangent vector singularity near values of $\eta$ or $\omega$ which are multiples of $\pi/2$. To overcome this problem, we reparameterise $\mathbf{S}$ by arc length [29]. That is,

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

where

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

Given this we can say

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

and

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

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

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

Now we re-express our steepest descent (on $d^2$) so that it is steepest with respect to the normalised parameters

$$\frac{\partial d^2}{\partial \alpha} = \frac{\partial d^2}{\partial \eta} \frac{\partial \eta}{\partial \alpha} = \frac{\partial d^2}{\partial \eta} \left\| \frac{\partial \mathbf{S}(\eta, \omega)}{\partial \eta} \right\|^{-1}$$

and

$$\frac{\partial d^2}{\partial \beta} = \frac{\partial d^2}{\partial \omega} \frac{\partial \omega}{\partial \beta} = \frac{\partial d^2}{\partial \omega} \left\| \frac{\partial \mathbf{S}(\eta, \omega)}{\partial \omega} \right\|^{-1}.$$  

We now can use the gradient of the reparameterised $d^2$,  

$$\nabla d^2' = [\partial d^2/\partial \alpha, \partial d^2/\partial \beta],$$

to find the closest point with greater stability.

The general formulation of Equation 12 significantly simplifies for values of $\eta$ and $\omega$ 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 ($0 \leq \eta \leq \pi/2, 0 \leq \omega \leq \pi/2$). Since a superellipsoid is 8-way symmetric, point $\mathbf{P}$ may be reflected into the first octant, the minimisation performed, and the solution point reflected back into $\mathbf{P}$’s original octant.

Once the closest point is calculated, the colour at that point may be determined by three different methods. The superellipsoid may be assigned a constant colour, which is the colour assigned to all closest points on the primitive. The $[\eta, \omega]$ values may be used to access a 2D texture map or the $[X, Y, Z]$ value of the closest point may be used to calculate a colour from a 3D procedural texture map [30].
\[
\begin{array}{c|ccc}
A \cup B & B & \text{IN} & \text{OUT} & \text{ON} \\
\hline
\text{IN} & \text{MIN} & A & A \\
A & \text{OUT} & B & \text{MIN} & B \\
\text{ON} & B & A & A \\
\end{array}
\]

Table 1: Union combination rules.

\[
\begin{array}{c|ccc}
A \cap B & B & \text{IN} & \text{OUT} & \text{ON} \\
\hline
\text{IN} & \text{MAX} & B & B \\
A & \text{OUT} & \text{A} & \text{MAX} & A \\
\text{ON} & \text{A} & \text{B} & A \\
\end{array}
\]

Table 2: Intersection combination rules.

\[
\begin{array}{c|c|ccc}
A - B & B & \text{IN} & \text{OUT} & \text{ON} \\
\hline
\text{IN} & -B & \text{MAX}(A-B) & B \\
A & \text{OUT} & \text{MAX}(A-B) & A & A \\
\text{ON} & -B & A & A \\
\end{array}
\]

Table 3: Signed distance difference combination rules.

\[
\begin{array}{c|c|ccc}
A - B & B & \text{IN} & \text{OUT} & \text{ON} \\
\hline
\text{IN} & 2-B & \text{MAX}(A,1/B) & B \\
A & \text{OUT} & \text{MIN}(A,2-B) & A & A \\
\text{ON} & 2-B & A & A \\
\end{array}
\]

Table 4: Inside-outside difference combination rules.

### 3.1.2 Combining Shortest-Distance Calculations

The CSG graph is processed in a depth-first manner. The closest point and colour 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 based on the shortest-distance values. 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 [7], a set of combination rules are utilised 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 and colour are 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 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.

Even though the range of the superellipsoid’s inside-outside [0, ∞] is different than the signed distance [−∞, ∞], the rules for combining signed distances are the same as the inside-outside combination rules for union and intersection used in the original Constructive Cubes algorithm, and are given in Tables 1 and 2. A detailed explanation of these rules may be found in [6]. The inside-outside difference combination rules (Table 4) are different than the signed distance combination rules given in Table 3.

The Constructive Cubes combination rules for difference (A-B) have been changed to work with signed distances rather than inside-outside values, 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, 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 the two objects (P9), and returns A, which is zero.

3.1.3 Detecting Invalid Results

It is possible that at any node in the CSG graph one or both of the closest points to its subcomponents do not lie on the final evaluated surface. For example in Figure 1, the closest points to both A and B
from point $P6$ do not lie on the final evaluated surface of $A \cup B$. Additionally, the closest points to both $A$ and $B$ from point $P13$ do not lie on the final evaluated surface of $A - B$. When generating colour information it is crucial to detect these invalid combinations when they first occur, and to flag them so they do not propagate invalid results up through the CSG graph during the closest-point/shortest-distance/colour calculations.

A test for invalid combination results is performed at each non-leaf node of the CSG graph during the depth-first traversal of the “Calculate-and-Combine” step of the closest-point calculation. The closest point calculated at each node is tested to determine if it lies on the surface of the model defined by the CSG subtree rooted at the current node. If it does not, the closest point calculated for that node is marked as invalid; otherwise it is marked as valid. The node’s closest point is evaluated with the Constructive Cubes (CC) algorithm [6] to produce this determination. The CC algorithm is a variation of Tilove’s point classification algorithm for CSG models [7], which returns a non-negative value less than one when the point is inside the CSG model, one when the point is on the model’s surface, and a value greater than one when outside the model. The CC algorithm is used in order to test for a value of $1 \pm \epsilon$, which allows the user to account for round-off errors in the calculations.

The combination rules defined in Tables 1, 2, and 3 are only applied at a particular node if more than one of the closest points calculated from its subcomponents are valid. The validation test is applied at a particular node if at least one of its subcomponent’s closest points is valid. If only one subcomponent closest point is valid the combination rules are not applied, and the valid point and the colour associated with the point are returned to the next higher level of the CSG graph. If none of the
subcomponents are valid, no closest-point/colour information is returned, and the node is marked as invalid. Ultimately this process produces a valid closest point to the complete model and the colour at that point, or a notification that no valid closest point can be calculated for the given input point.

The process may be further explained with Figure 2. The closest point calculation is always valid for leaf nodes (superquadrics in our case) of the CSG graph. Given that valid closest points are returned for both nodes C and D to node 5, these results are combined at node 5 using the combination rules in Tables 1, 2, and 3. The closest-point result at node 5 is now tested for validity by evaluating the closest point with the Constructive Cubes algorithm against the CSG submodel defined at node 5. Let's assume that node 5’s closest point is invalid, i.e. the closest point does not lie on the evaluated surface defined by node 5. Now a closest point from one of node 3’s subcomponents is invalid (node 5’s) and one is valid (node C’s). The combination rules are not applied at node 3, because it only has one valid subcomponent. C’s closest point becomes the closest point for node 3. It is now tested for validity against the submodel defined at node 3.

3.2 Fast Marching Method For Computing Closest Points and Colours

We present a Fast Marching Method for computing the approximate closest point to a surface from the points in a regular grid, as well as the colour at the closest point. Shortest-distance information may then derived from the closest-point information. The accuracy of the method depends on a discretisation of the surface and is independent of the volume grid spacing, allowing us to calculate distance to subvoxel 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 $S$. $u$ is a weak solution of the Eikonal equation,

$$|\nabla u| \equiv \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial z}\right)^2} = 1,$$

subject to $u|_S = 0$. (13)

The characteristics of Equation 13 are straight lines that are normal to $S$. 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 [31, 5] has developed a Fast Marching Level Set Method to solve the Eikonal equation,

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

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 $u|_S = 0$ 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 and Colour Calculation Overview

To calculate the closest points to a surface and the colour at those points on a regular grid, we utilise Sethian’s Fast Marching Method, but instead of using a finite difference scheme to compute distance, we use a heuristic algorithm to propagate closest-point and colour 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. The closest points at the remaining voxels are then computed to the point samples on the original surface. In one step of the closest-point and colour method:

1. The point $gp$ with the smallest distance is removed from the narrow band and it’s shortest-distance, closest-point and colour value are frozen.

2. Points are added to the narrow band to maintain unit thickness.

3. The closest points of the neighbours with larger distances than $gp$ are recomputed using the closest-point information from $gp$.

The closest-point and colour 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 neighbours in the grid. Thus if one knows the closest points of the neighbours 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 neighbours. This is only a heuristic, and in Figure 3 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.
3.2.3 Terminology

Let the distance, closest-point and colour volumes be represented by \( N \times N \times N \) grids that span the space around the scan converted object. We will refer to points in these volumes 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 volume grid, providing distance calculations with subvoxel 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 grid points in an initial working volume that forms a narrow band around the CSG model surface and a point sampling of the surface. This second set of points will be called the zero set, since they are points lying on the isosurface of zero distance. The narrow band contains all the points in the working volume having the property that a neighbour\(^1\) of the point has opposite inside/outside status. We generate the narrow band by evaluating the inside/outside status \([7]\) of all the grid points of the working 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 and colour information outward and inward to the rest of the points in the working 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, the zero set. 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 Section 3.1. 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 a neighbourhood of \( p \) by determining all the points in the zero band in a neighbourhood 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 4 the initial data is shown graphically in two dimensions.

\(^1\)In three dimensions neighbour means one of the 26 locations surrounding each grid point.
3.2.5 Propagating the Closest-Point and Colour Data

Initially, we have the closest-point and colour data in the zero band that surrounds the surface. We use the closest-point and colour data in the zero band to determine the closest points and colours in the narrow band of the working volume and then march the narrow band outward and inward to calculate the closest points and colours in the rest of the working volume. Consider a point $gp$ that neighbours the band and whose closest point is unknown. The closest point of $gp$ is probably close to one of the closest points of its neighbours in the band. Thus for each neighbour of $gp$ in the band, we compute the distance of $gp$ by considering zero set points that are near the closest points of that neighbour. 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 neighbours.

Let $in\,out_{ijk}$ denote the inside/outside status for a point in the working volume; $+1$ for outside, $-1$ for inside. Let $dist_{ijk}$ denote the computed distance at a working volume grid point. A value of $\infty$ 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 $dist_{ijk} = in\,out_{ijk}$. For each point not in the narrow band $dist_{ijk}$ 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-point/colour algorithm.

\begin{verbatim}
begin
  // March forward to find positive distances.
  put each point with a non-negative, finite
  dist_{ijk} in the set U;
  while U \neq \emptyset
    remove the grid point gp with the smallest
    distance from U;
    for each of the 26 neighbours of gp
      if the source of the neighbour is unknown
        add that neighbour to U;
      if the distance of the neighbour is
        larger than the distance of gp
        recompute the neighbour’s distance
        using gp’s source s;
end
\end{verbatim}

Next, the narrow band is marched backward to compute the closest points and colours with negative distance. Once the closest point, shortest distance and colour have been computed for all the grid points of the working volume, separate closest-point, distance and colour volumes are written to individual files.

Below is the algorithm to recompute the distance $dist_{ijk}$ to the working 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 neighbourhood on the surface around the point $s$. The parameter, $\sigma = 2 \times R + 1$ is the diameter of the cube. When recomputing the distance, all the points in the zero set in a neighbourhood around $s$ are considered as possible closest points.
begin
for each grid point \((l,m,n)\) in a \(\sigma \times \sigma \times \sigma\) cube
surrounding \((I,J,K)\)
\(t \in Z\) is the closest point to \((l,m,n)\);
calculate the distance from \(gp\) to \(t\);
dist\(_{ijk}\) = minimum of the \(\sigma^3\) 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 \(O(\sigma^2)\).

### 3.3 Computational Complexity

There are \(N^3\) grid points in the working volume. At any point in the algorithm, there are \(O(N^2)\) 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 \(O(N^3P)\) operations. Determining the closest point on the CSG model from a particular grid point is also an \(O(P)\) operation. This is only computed on the points of the zero band. Calculating the zero set requires \(O(M^2P)\) operations, recalling that the zero set grid has resolution \(M \times M \times M\). Unfortunately it is difficult to characterise 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.

Each working grid point is removed from the narrow band once, giving us a factor of \(N^3\). 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 \(O(\log N)\). 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 neighbourhood of a point \(s\) in the zero band. This gives us a factor of \(O(\sigma^2)\). Thus the overall computational complexity of the fast marching algorithm is \(O(N^3\sigma^2 \log N)\).

### 4 Colour Shading Polygonal IsoSurfaces

A polygonal approximation of an isosurface imbedded in the distance volume may be generated with the Marching Cubes algorithm [3]. The colour and closest-point volumes may then be used to colour shade the isosurface. In this approach the colour at any point on the isosurface is defined as the colour at the closest point on the original CSG model from the isosurface point.

When a colour value at \([X,Y,Z]\) of the polygonal surface is needed during rendering, a user-defined number of tri-linearly interpolated samples are taken from the colour volume around the \([X,Y,Z]\) location. The colour value is calculated by

\[
C = C_{\text{eff}}(1-u)(1-v)(1-w) + C_{\text{eff}}(u)(1-v)(1-w) +
\]
\[ C_{\text{ref}}(1 - u)v(1 - w) + C_{\text{sef}}(u)(v)(1 - w) + \\
C_{\text{frc}}(1 - u)(1 - v)(w) + C_{\text{euc}}(u - v)(w) + \\
C_{\text{rec}}(1 - u)(v)(w) + C_{\text{exc}}(u)(v)(w) \]

where \( u = (X - \text{floor}(X)) \), \( v = (Y - \text{floor}(Y)) \), and \( w = (Z - \text{floor}(Z)) \). \( C_{\text{xyz}} \) defines the colour value stored at a specific voxel in the colour volume. 'f' and 'c' designate the floor or ceiling of the \( X \), \( Y \), or \( Z \) value used to reference the colour. The distance between adjacent voxel locations is defined to be 1. For example, if the point on the polygonal surface being shaded is \([34.27, 129.78, 56.45]\) the colour at that point would be computed by

\[
C = C_{34.129, 56}(1 - 0.27)(1 - 0.78)(1 - 0.45) + \\
C_{35.129, 56}(0.27)(1 - 0.78)(1 - 0.45) + \\
C_{34.130, 56}(1 - 0.27)(0.78)(1 - 0.45) + \\
C_{35.130, 56}(0.27)(0.78)(1 - 0.45) + \\
C_{34.129, 57}(1 - 0.27)(1 - 0.78)(0.45) + \\
C_{35.129, 57}(0.27)(1 - 0.78)(0.45) + \\
C_{34.130, 57}(1 - 0.27)(0.78)(0.45) + \\
C_{35.130, 57}(0.27)(0.78)(0.45),
\]

where \( C_{34.129, 56} \) signifies the value stored in the colour volume at \([34, 129, 56]\).

In the examples in this chapter we take samples at the corners of the cube centered around \([X, Y, Z]\), as well as at \([X, Y, Z]\), with the length of the edges of the cube equal to one, the distance between voxels. If all of the colour values are the same, the constant colour value is returned as the colour of the polygonal surface at \([X, Y, Z]\). If any of the sampled colour values are different, a tri-linearly interpolated closest-point value is calculated from the closest-point volume at the same sample locations. Since these interpolated points do not necessarily lie on the surface of the original CSG model, the closest point from each of the interpolated points to the CSG model and the associated colour are calculated using the “calculate-and-combine” step of the method described in Section 3.1. The colours calculated at all of the samples are averaged together to produce the final colour for \([X, Y, Z]\).

Our ray-tracer automatically supersamples and anti-aliases those regions of the image where large colour changes take place. Therefore, our approach quickly renders those parts of the model where there are constant colour values, and only performs expensive closest-point calculations along edges of differing colours. Since the supersampling of the closest-point volume is driven by the rendering process, only the extra closest-point calculations necessary for proper anti-aliasing are performed, saving the computation time and storage space that would be needed to supersample the entire volume.

The complete process of colour shading a distance-volume-derived polygonal surface is summarised in Figure 5. Distance, colour and closest-point volumes are generated from a CSG model using the 3D scan conversion algorithm (step 1). A polygonal surface is generated from the distance volume using the Marching Cubes algorithm (step 2). The resulting surface is rendered (step 3). In order to determine the colour at a specific point on the surface, the colour volume is sampled around that location (step 4). If all of the samples are the same colour, the colour is used to shade the point on the surface. If the colours are different, the closest-point volume is sampled at the same locations as the colour volume (step 5). The closest points on the original CSG model are calculated for the interpolated sample points, as well as the colour using the “calculate-and-combine” algorithm (step 6). The colours calculated in step 6 are combined and the average colour is used when rendering the specific point on the polygonal surface (step 7).
5 Results

A number of moderately complex CSG models have been scan converted into distance, closest-point and colour 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 scan-converted volumes have been used to generate an evaluated surface of the model, as well as offset surfaces. Additionally, the volumes have been utilised to morph one model into another [4]. The results contain rendered polygonal surfaces which have been extracted from the distance volumes using the Marching Cubes Algorithm. The surfaces have been colour shaded using the algorithms described in Section 4.

Figure 6 presents an evaluated CSG surface of an X-29 jet fighter, consisting of 38 primitives and generated from a $96 \times 192 \times 240$ distance volume. Additionally three offset surfaces are generated by applying the Marching Cubes algorithm with an isovalue greater than zero. Figure 7 presents the same polygonal models with the colour shading algorithm applied to the surfaces. Figure 8 presents five colour-shaded isosurfaces generated from the scan-converted volumes of a dart model, consisting of 21 primitives and sampled at a resolution of $96 \times 192 \times 240$. Figure 7 presents four colour-shaded isosurfaces generated from the scan-converted volumes of a CSG part, consisting of 5 primitives and sampled at a resolution of $195 \times 90 \times 120$. Figure 10 presents a morphing sequence where the dart model transforms into the X-29 model. The morphing process involves manipulating and changing voxel values of the dart distance volume based on the values in the X-29 distance volume [4]. Since the basic model representation of the morphing object is volumetric, the colour and closest-point volumes associated with the initial and final models may be used to colour shade the intermediate shapes. For a given point $[X, Y, Z]$ on the morphing surface, the associated colour is calculated from both the initial and final models' colour and closest-point volumes. These two colours are linearly interpolated based on the time parameter of the morphing sequence to produce an average colour for $[X, Y, Z]$ on the intermediate shape.

These volume resolutions were chosen because they produced satisfactory results given the cost in time (several hours) and memory ($\sim$17 MBbytes) 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 [30]. 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.
6 Conclusion

We have described a technique for generating distance, closest-point and colour volumes with subvoxel accuracy from one type of geometric model, a CSG model consisting of superellipsoid primitives. The volumes are 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 and contains colour information 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, closest-point and colour information out to the remaining voxels in the volume. In addition, we have described a colour-shading technique that utilises the closest-point and colour volumes to calculate the surface colours on polygonal surfaces extracted from the distance volumes. Our techniques have been used to scan convert a number of CSG models, producing volumes and colour-shaded polygonal surfaces which have been utilised in a variety of computer graphics applications, e.g. CSG surface evaluation, offset surface generation, and 3D model morphing.

7 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 dart model used in Figures 8 and 10. 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; 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


Figure 6: Offset surfaces from the X-29 distance volume.

Figure 7: Anti-aliased colour-shaded offset surfaces from the X-29 distance, colour, and closest-point volumes.
Figure 8: Anti-aliased colour-shaded offset surfaces from the dart distance, colour, and closest-point volumes.

Figure 9: Anti-aliased colour-shaded offset surfaces from the CSG part distance, colour, and closest-point volumes.
Figure 10: Anti-aliased colour-shaded morphing surfaces.