A Technique for Volumetric CSG based on Morphology

Andreas Bærentzen and Niels Jørgen Christensen

IMM, Technical University of Denmark, {jab|njc}@imm.dtu.dk

Abstract. In this paper, a new technique for volumetric CSG is presented. The technique requires the input volumes to correspond to solids which fulfill a voxelization suitability criterion. Assume the CSG operation is union. The volumetric union of two such volumes is defined in terms of the voxelization of the union of the two original solids.
The theory behind the new technique is discussed, the algorithm and implementation are presented. Finally, we present images and timings.

1 Introduction

Many techniques in *volume graphics* are easily designed for binary volumes, but turn out to be quite difficult to generalize to *grey-level* volumes. A good example of this is *Constructive Solid Geometry*. Constructive solid geometry (CSG) [1] is a powerful paradigm for composing more complex shapes from simpler ones, and at first sight it seems to be very simple to use this paradigm in volume graphics. Indeed, for binary volumes, it is simple, since a CSG operation can be implemented as a block operation between the two input volumes. For each voxel location the new voxel value is calculated as a boolean operation between the old values.

For volumes where the voxel values are scalar and not boolean, CSG has, so far, also been implemented using block operations, but it is less clear what operations should be used to combine two voxels. In fact, it is not clear that it is at all possible to define a block traversal based CSG operation on scalar volumes. To clarify where the problem lies, consider the case where the voxel value represents the geometric distance to the solid. The distance to two objects from a given voxel location is not always in itself enough to estimate the distance to the new solid which results from the CSG operation. Although it may be perfectly feasible to visualize the resulting object, it is problematic that most of the voxels in the resulting object will have a value that corresponds to a geometric property while others will not. Put differently, the problem is that no volumetric CSG operation has so far been proposed that ensures consistency with respect to the type of 3D scalar function from which the original volumes were sampled. This may not be a problem in some of the application areas of volumetric CSG (e.g. for highlighting regions of interest in medical volume data), but for volumetric CSG in the context of shape modeling, it is a problem.

In this paper, we present a new technique for volumetric CSG. The input is two volumes which must be voxelized from solids that fulfill the openness–closedness criterion [2]. This criterion (which is defined in terms of mathematical morphology) can be explained, intuitively, to mean that it must be possible to roll a sphere on both the interior and exterior of the surface of a solid. Features such as sharp edges, corners and surface components that are too close together make it difficult to sample and reconstruct the solid with adequate precision. If the openness–closedness criterion is fulfilled, we know that these features are not present.

We surmise that a volumetric CSG operation must be as close as possible to a CSG operation on the reconstructed solids (i.e point sets). Hence, the following operation, would yield the desired result: (a) Reconstruct the original solids from their volumetric representation, (b) perform the CSG operation on the reconstructed solids (c), modify the result to ensure that the result fulfills the openness-closedness criterion, and then (d) voxelize once more to obtain a volumetric representation. This scheme cannot be implemented directly, though, and, consequently, our technique operates quite differently, but produces the same result as the above.

We use the morphological operators open and close to ensure that our solids do not violate certain constraints on the shape. In the context of non–voxel based solid modelling, morphological operations are sometimes used to remove non–manifold points. Hence, there is a big difference between the use of morphology in ordinary CSG and in the present context.

1.1 Notation and definitions

In the following, S_x will denote solids. ∂S_x the boundary of a solid, G_x will denote voxel grids. The word volume will be used interchangeably with voxel grid. $G_x[\mathbf{p}]$ where $\mathbf{p} \in N^3$ is the value of G at a grid point, i.e. a voxel value. vu means voxel unit – the smallest distance between two voxels.

For the sake of brevity, we shall discuss only one CSG operation, namely union. We are also interested in CSG difference and intersection, but these may be defined in terms of union and complement: $S_1 \bigcap S_2 = (S_1^c \bigcup S_2^c)^c$ and $S_1 \backslash S_2 = (S_1^c \bigcup S_2)^c$.

A V–model [3] of a solid S is a function that is smooth in a *transition region* on both sides of the boundary ∂S. In addition to smoothness, we require of a V–model that it should contain the boundary of the solid as an iso–surface. The operator V will be used to denote voxelization, and $G = V(S)$ means that the volume G is sampled from the V–model of S. The un–sampled V–model of a solid is denoted $\mathcal{V}(S)$.

2 Previous Work

Previous approaches to volumetric CSG [4–7] have in common that they are block operations where the new value at each grid point is calculated using only

the voxel values for this grid point from each of the volumes being combined. This mode of operation is sometimes called *voxblt* (Voxel Block Transfer) [8].

$$G_{new}\left[\mathbf{p}\right] = G_1\left[\mathbf{p}\right] \cup_v G_2\left[\mathbf{p}\right] \tag{1}$$

Where G_x are volumes and $\mathbf{p}$ is a grid point in G_{new}. In some of the approaches the voxel grids on the right hand side (rhs) of (1) may themselves be defined by the same equation [5, 6]. In this way, the recursive application of (1) defines a CSG tree where the leaf nodes are volumes. To evaluate the value at a given grid point, we traverse the CSG tree, performing binary per–voxel operations at each node until we reach a leaf.

Other authors [4, 7] let one of the volumes on the rhs be *object* and the other *tool*, and let the value of (1) be assigned to the object volume.

The approaches also differ in the exact nature of the $\cup_v$ operator. Some authors [9–11, 7] prefer to use min:

$$G_1\left[\mathbf{p}\right] \cup_v G_2\left[\mathbf{p}\right] = \min(G_1\left[\mathbf{p}\right], G_2\left[\mathbf{p}\right]) \tag{2}$$

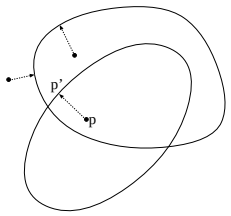

Fig. 1. Closest points on $S_1 \bigcup S_2$ using min distance.

If the V–model represents the *signed shortest distance* to the solid [2, 12] (where the sign is negative in the interior), (2) yields the correct shortest distance to the surface of the union in most cases. In fact (2) only fails to yield correct results for some points that are interior with respect to both solids, because in this case, the point which corresponds to the minimum of the distances to either solid, may be an interior point in the combined solids. This is illustrated in figure 1 where $\mathbf{p}'$ is the point corresponding to the minimum of the distances from $\mathbf{p}$ to the boundaries of the two solids. We see that $\mathbf{p}'$ is interior in the combined solid.

It is worth noting that it is not possible to bound the error: A CSG operation according to (2) might yield a value that is very close to 0 (i.e. the point should be close to the surface) while the point is in fact very far from the surface. An extreme example is the union of two half spaces delimited each by a plane of infinite extent. If the planes are parallel, point toward each other, and if the half spaces overlap, then the union is all of space, and the distance to the surface

should be $-\infty$ everywhere. Hence, (2) is wrong everywhere – except infinitely far from the original planes.

In [4] the authors argue that the following operator is better

$$G_1[\mathbf{p}] \cup_v G_2[\mathbf{p}] = G_1[\mathbf{p}] + G_2[\mathbf{p}] - G_1[\mathbf{p}]\, G_2[\mathbf{p}] \tag{3}$$

(largely) because the result is smoother. Unfortunately, this version of $\cup_v$ is not perfect, either, since $G \bigcup G \neq G$. By design, this operator does not yield the distance to the union but rather a smooth "pseudo–distance".

To sum up, (3) and (2) both fail to produce sensible results in some cases. At this point it is useful to consider what we should demand from a *correct* technique for volumetric union. The ideal is obviously that

$$V(S_1) \bigcup\nolimits_v V(S_2) = V(S_1 \bigcup S_2) \tag{4}$$

where $\bigcup_v$ and $\bigcup$ denote union of voxel grids and solids, respectively. Unfortunately, $S_1 \bigcup S_2$ will, in general, contain sharp edges. Such features cannot be represented volumetrically. To remedy this fact, we propose, as an attainable goal that

$$V(S_1) \bigcup\nolimits_v V(S_2) = V(\mathcal{F}(S_1 \bigcup S_2)) \tag{5}$$

where $\mathcal{F}$ is a shape filtering operator. $\mathcal{F}$ changes a solid so that it becomes representable, i.e. so that its V–model can be sampled and reconstructed with adequate precision.

3 Theory

In [2] the authors propose a criterion for determining whether solids are suitable for voxelization. The criterion assumes the *clamped signed distance* V–model. This V–model is a function $\mathcal{V} : R^3 \to R$ associated with a solid S which yields the signed shortest distance to ∂S from its argument $\mathbf{p}$. The value is clamped to the interval $[-r, r]$. Formally,

$$\mathcal{V}(S)(\mathbf{p}) = \min(\max(d_S(\mathbf{p}), -r), r) \tag{6}$$

where

$$d_S(\mathbf{p}) = \begin{cases} -\min_{\forall q \in \partial S}(|\mathbf{p} - \mathbf{q}|) & \mathbf{p} \in S \backslash \partial S \\ 0 & \mathbf{p} \in \partial S \\ \min_{\forall q \in \partial S}(|\mathbf{p} - \mathbf{q}|) & \mathbf{p} \notin S \end{cases} \tag{7}$$

The authors show, that if a solid is voxelized using this V–model, it is possible to bound the trilinear reconstruction error, provided that the solid fulfills a suitability criterion.

The essence of this criterion is that it must be possible to roll a sphere of a given radius on the surface of the solid (both inside and outside). Formally, we can state this criterion as the simple condition that the solid must have the

openness and *closedness* properties. This means that the operations open and close do not change the solid:

$$S = O(S, s_r) \wedge S = C(S, s_r) \tag{8}$$

where s_r is the structuring element, and, consequently, the smallest representable sphere. The radius r of s_r is also the thickness of the transition region, and the value of the V–model, (6), is clamped to $[-r, r]$.

The value of r should be chosen according to the desired tolerance. If r is large relative to the grid, the smallest representable "blob" is also relatively large. If r is small, more detail can be represented but the reconstruction errors will be greater. Usually, it is desirable that $r \geq \sqrt{6}$. The reader is referred to [2] for the details.

Solids that do not fulfill (8) need to be changed, for instance by actually applying open and close:

$$S_{\text{good}} = C(O(S_{\text{bad}}, s_r), s_r) \tag{9}$$

Unfortunately, for some solids, close will ruin openness and vice versa. For instance, in some cases where two solids are separate but very near each other, close will bridge the gap and the bridge will be removed by open. This means that such configurations are not representable and that there is no obvious way to change them so that they will conform to (8).

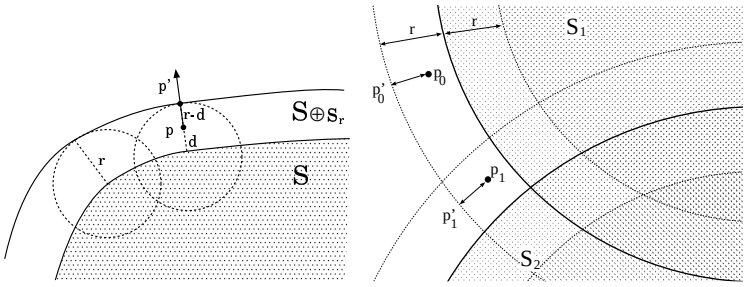

Fig. 2. Closest points on surfaces of dilated solids (left), and ∂S and $\partial(S \oplus s_r)$ (right).

We shall develop an algorithm for $\bigcup_v$ that adheres to (5) using (9) as the shape filter $\mathcal{F}$. Thus, the $\bigcup_v$ operation should take as input two volumes that both correspond to solids which have the openness and closedness properties. The output is a new volumetric model which also corresponds to a solid which has the openness and closedness properties.

Until section 4 the approach will be discussed theoretically and only in terms of un–sampled V–models. Hence, the operands of $\bigcup_v$ are V–models (e.g. $\mathcal{V}(S_1)$ and $\mathcal{V}(S_2)$). It is assumed that these V–models correspond to two solids (e.g. S_1 and S_2) which fulfill (8).

The union of these two solids $S_1 \bigcup S_2$ might not, however, fulfill the criterion. Hence, we need to filter the rhs using (9) before V–model construction

$$\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2) = \mathcal{V}(C(O(S_1 \bigcup S_2, s_r), s_r)) \tag{10}$$

Since it is easy to show that union preserves openness

$$\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2) = \mathcal{V}(C(S_1 \bigcup S_2, s_r), s_r)) \tag{11}$$

and using the facts that close is dilation followed by erosion and that dilation distributes over union [13], we obtain

$$\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2) = \mathcal{V}(((S_1 \oplus s_r)\bigcup(S_2 \oplus s_r)) \ominus s_r) \tag{12}$$

The principle behind the algorithm is to reconstruct the dilated solids from the V–models (on the lhs) and subsequently to construct the V–model of their eroded union – which is tantamount to the rhs of (12).

3.1 Finding the value of $\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2)$ at a point

It is clear from the definition of dilation [13] that when dilating with a spherical structuring element s_r of radius r, the distance from a point on ∂S to the closest point on $\partial(S \oplus s_r)$ is always r (see figure 2). Consequently, for any point in a distance field V–model (6), we can find the closest point on the surface of the dilated solid using the simple equation:

$$\mathbf{p}' = (r - d)\mathbf{n} + \mathbf{p} \tag{13}$$

where d is the distance to the (un-dilated) surface, and $\mathbf{n}$ is the normal direction. This is illustrated in figure 2.

The above equation can be used to generate points on the boundary of the dilated solids ∂S_1 and ∂S_2. If we throw away points that are in the interior of the other dilated solid (such as $\mathbf{p}_1'$ in figure 2), the remaining generated points must belong to the set D of surface points defined by:

$$S_{\text{ud}} = (S_1 \oplus s_r)\bigcup(S_2 \oplus s_r) \tag{14}$$

$$D = \partial S_{\text{ud}} \tag{15}$$

$$S_{\text{iud}} = S_{\text{ud}} - D \tag{16}$$

D is shown as a heavy curve in figure 4. Now, we can use the fact that points on the surface of a solid eroded with s_r are at a distance r from the closest point on the surface of the original solid (just like in the case of dilation). The surface D of the original solid S_{ud} in this case. Assuming we have a means of finding $d_{S_{\text{ud}}}$, we can find the distance function belonging to $((S_1 \oplus s_r)\bigcup(S_2 \oplus s_r)) \ominus s_r$ and, consequently, we have obtained the V–model on the rhs of (12)

$$\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2)(\mathbf{p}) = \min(\max(d_{S_{\text{ud}}}(\mathbf{p}) + r, -r), r) \tag{17}$$

To compute $d_{S_{\text{ud}}}$ (the clamped distance function for the union of the dilated solids) we can apply the ordinary CSG equation (2):

$$d_{S_{\text{ud}}} = \min(d_1, d_2) \tag{18}$$

where

$$d_1 = d_{S_1 \oplus s_r}(\mathbf{p}) = \mathcal{V}(S_1)(\mathbf{p}) - r \tag{19}$$
$$d_2 = d_{S_2 \oplus s_r}(\mathbf{p}) = \mathcal{V}(S_2)(\mathbf{p}) - r \tag{20}$$

(Observe that $d_1, d_2 \in [-2r, 0]$). Again, (18) only holds if the distance corresponds to a point that actually belongs to D. The following case analysis reveals where more work is needed. Without loss of generality, we will assume that $d_1 \leq d_2$ in the following.

- For points where $d_1 = -2r \vee d_2 = -2r$ it follows that $d_{S_{\text{ud}}} = -2r$ because the union operation cannot cause the distance from an interior point to the boundary of the solid to decrease. Furthermore, the V–model dictates that the value should be clamped, so no more work is needed.
- For points where $d_1 = 0 \wedge d_2 = 0$ it follows that $d_{S_{\text{ud}}} = 0$. This is clear because if the point is on the boundary or exterior of both solids, it cannot be interior in the union; again, due to clamping, no more work is needed.
- For the remaining points, we know that $-2r < d_1 < 0$ and $d_1 \leq d_2$. Now, if $\mathbf{p}_1' \in D$ where $\mathbf{p}_1' = -d_1\mathbf{n} + \mathbf{p}$, then (18) also holds. This is (nearly) trivial, because $\mathbf{p}_1'$ is already the closest point belonging to $\partial(S_1 \oplus s_r)$ and there can be no closer point in D since union cannot cause the distance from the interior point $\mathbf{p}$ to the boundary to decrease.

If $\mathbf{p}_1' \notin D$ it is not immediately clear which point belonging to D that is closest to $\mathbf{p}$. To compute $d_{S_{\text{ud}}}$ in this case, we need a proposition, but first (in order to simplify notation) some helpful definitions

$$D_1 = \partial(S_1 \oplus s_r) \tag{21}$$
$$D_2 = \partial(S_2 \oplus s_r) \tag{22}$$

(The point sets are illustrated in 3)

Proposition 1. *Given a point* $\mathbf{p}$ *where* $-2r < d_1 < 0$ *and* $d_1 \leq d_2 \leq 0$ *and the point* $\mathbf{p}_1' \in D_1$ *so that*
$d_1 = -||\mathbf{p}_1' - \mathbf{p}||$.
If $\mathbf{p}_1' \notin D$ *then either* $d_{S_{\text{ud}}}(\mathbf{p}) \leq -2r$ *or*
$d_{S_{\text{ud}}}(\mathbf{p}) = -||\mathbf{p}'' - \mathbf{p}||$ *where* $\mathbf{p}'' \in I$ *and*

$$I = \partial(S_1 \oplus s_r) \bigcap \partial(S_2 \oplus s_r) \subset D \tag{23}$$

Swap 1 and 2 if $d_2 < d_1$.

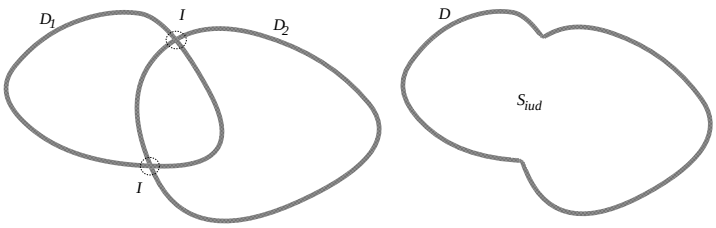

Fig. 3. $D, D_1, D_2, S_{\text{iud}}, I$

For a proof se [14]. The proposition is complicated, but its application is simple. For each voxel, we simply check whether the point $\mathbf{p}'$ corresponding to the shortest distance and computed according to (13) belongs to D. If that is the case, the value of $d_{S_{\text{ud}}}$ is computed using (18). Otherwise, the value of $d_{S_{\text{ud}}}$ is either equal to the shortest distance to I or $d_{S_{\text{ud}}} \leq -2r$ in which case the voxel value is clamped. Finally, the new voxel value is computed using (17).

In general, I is a curve[1]. In fact, we can view the entire process of performing the close operation as adding constant curvature blending to the CSG operation. If we see the operation as a blending, we can construe the curve I as the locus of the centre of the blending sphere.

3.2 Examples

In order to find the value of $\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2)$ at a given point, $\mathbf{p}$, the first step is to classify $\mathbf{p}$ according to the rules in table 1. Any point whose signed distance to the (un–dilated) solid is greater than r is called exterior. Any point whose distance is smaller than $-r$ is called interior. As the table indicates, points

state	I	T	E	
I	I	I	I	I=interior
T	I	T‖I	T‖I	T=transition
E	I	T‖I	E	E=exterior

Table 1. Transition rules for volumetric union

that are exterior to both solids remain exterior, and points that are interior with respect to either solid become interior. For these points the value of (17) is unchanged. For instance, $\mathbf{p}_2$ and $\mathbf{p}_3$ in figure 4 are exterior and interior, respectively. The values of the V–model at these points are $\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2)(\mathbf{p}_2) = r$ and $\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2)(\mathbf{p}_3) = -r$.

For points in the transition region of one solid which are simultaneously in the transition region or exterior to the other solid, more work is required. If

[1] In 2D, I is one or more points. I is shown in figure 3

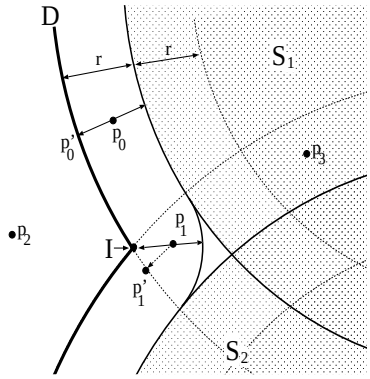

Fig. 4. Point classification

the corresponding point on the surface of the dilated solid (say S_1) is exterior to the other dilated solid (S_2), we simply use the value of $\mathcal{V}(S_1)$. This case is exemplified by $\mathbf{p}_0$ in figure 4 where $\mathcal{V}(S_1)\bigcup_v \mathcal{V}(S_2)(\mathbf{p}_0) = \mathcal{V}(S_1)(\mathbf{p}_0)$.

If the corresponding point on the surface of the dilated solid is an interior point in the other dilated solid, the problem is less trivial. We will call such points (or voxels) *inconsistent* in the following (in figure 4 $\mathbf{p}_1$ is inconsistent). For inconsistent points, we need to find the distance to $I \subset D$ according to our proposition.

In summary, our algorithm should work like previous volume CSG algorithms based on (2), except that for some points we need to estimate the distance to I. Hence, the main difficulties lie in representing I and finding the inconsistent points.

4 Algorithm

In the previous sections, we ignored the fact that we are dealing with discrete voxel grids, but now we shall look at how the algorithm actually operates. First of all, the algorithm works on two operands: The voxel grid being modified, $G = V(S_1)$, and the tool, $\mathcal{V}(S_2)$, which is an un–sampled V–model since sampling it before the CSG operation would only complicate matters[2]. The result of the CSG operation is assigned to the grid:

$$G \leftarrow G \bigcup_v \mathcal{V}(S_2) \tag{24}$$

The algorithm works in two passes, and both passes traverse all voxels.

First pass For each voxel two operations are performed during the first pass:

It is determined which solid has the smallest distance d to the voxel and then the corresponding point $\mathbf{p}'_1$ on the dilated solids surface is found using

[2] Although it would, of course, be useful for cut and paste operations.

(13). Our volume representation contains gradient information, which speeds up this process, although the gradient could also have been estimated. If $\mathbf{p}_1'$ is an interior point in $(S_1 \oplus s_r) \bigcup (S_2 \oplus s_r)$, d is no longer correct, and the voxel is tagged as being inconsistent.

The point $\mathbf{p}_1' \in \partial(S_1 \oplus s_r)$ closest to the voxel is found. If $\mathbf{p}_1'$ is within $\frac{1}{2}$ vu distance to $\partial S_2 \oplus s_r$, we estimate the closest point in I by assuming that $\partial(S_1 \oplus s_r)$ and $\partial(S_2 \oplus s_r)$ are planes and finding the point on the intersection of these plans that is closest to $\mathbf{p}_1'$. The point is added to a set of points that make up our estimate of I.

Second pass We loop once more over all voxels and perform a case analysis according to table 1. For voxels that are in the transition region of one solid and either exterior or in the transition region of the other, we check if they are tagged as inconsistent.

For untagged voxels the new value is simply the minimum of the V–models $\mathcal{V}(S_1)$ and $\mathcal{V}(S_2)$.

For inconsistent voxels, the closest point in the set of points representing $I \subset D$ is found, and the distance to that point is used to calculate the new voxel value according to (17).

Estimating the closest point in I is one of the trickiest parts of the algorithm, and it cannot be done simply by finding the closest point in the set of I–estimate points generated during the first pass, because these estimates may be as far apart as 1 vu if the surfaces of both S_1 and S_2 are parallel to coordinate axes in the grid. Hence, the distance to an I–estimate would sometimes deviate too much from the true distance to I. To solve this problem, we store an estimated tangent direction vector with (nearly) all points in the point set representing I. To estimate the distance from a voxel to I, we find the closest I–estimate and the projection of the voxel onto the line defined by the I–estimate and the associated direction vector. The projected point is used as the estimate of our closest point in I, and this point is used to update both the distance and normal direction of the voxel.

In some cases, the union of two solids may contain a corner. In these cases, I bends sharply and the direction vector associated with the I–estimate at the bend would be misleading. The solution is to find the closest I–estimate before and after any given I–estimate. If the angle defined by a given estimate and its two neighbours is above a given threshold (set to 0.52 rad) we assign the null–vector to the direction vector of the I–sample.

It should be observed that in places where the angle between S_1 and S_2 is too small, we do not find I–estimates. Since there are no inconsistent voxels in regions where ∂S_1 and ∂S_2 are parallel, this is not a problem.

While the distance between I–estimates can be great, there are also cases where the I–estimates cluster. To speed up searching for the nearest I–estimates these clusters are merged.

4.1 Alternative algorithm

A weakness of the described algorithm is the complexity in finding I–estimates. A much simpler, albeit potentially more computationally demanding, approach would be to find the nearest I–estimate independently for each voxel.

Using the alternative approach, for a point $\mathbf{p}$ we find the closest point $\mathbf{p}'_1$ on D_1. For $\mathbf{p}'_1$ we find the point $\mathbf{p}'_2 \in D_2$ closest to $\mathbf{p}'_1$. We repeat the process until convergence or some threshold has been exceeded.

This approach is slower, but it is simpler and for small CSG tools and reasonably thin transition regions, the difference is negligible.

5 Implementation

The algorithm has been implemented in C++ on a Linux platform and incorporated in a framework for voxelization, volume manipulation and visualization.

In the following, implementation specific details will be discussed. These pertain mostly to data structures and methods of visualization.

The volume is stored in a two–level hierarchical grid. For each voxel, the distance is stored as a two byte fixed point number. The unit length gradient is stored as two angles, each coded using two bytes. An additional flag byte is used to store information about whether the voxel is interior, transitional, or exterior, and whether it is tagged. Alignment causes the total size of a voxel to be eight bytes.

To keep the implementation simple, only CSG union has been implemented directly, but, as previously mentioned, intersection and difference can be expressed in terms of union and complement. Therefore, a voxel inversion function has been implemented. If flips the direction of the normal and the sign of the distance, and using this function the two other operations are possible.

By its nature, the intersection operation can change the volume far from the solid used as CSG tool. Fortunately, the same is not true in the case of union and subtraction. This is utilized by restricting the effect of the CSG operation to a *region of effect* about the CSG tool. The size of this region depends on the value of r. For intersection, the region of effect is simply the entire volume.

The representation of I is simply a set of points and associated tangent direction vectors. Since we need to search for the closest I–estimate, we have elected to store the I–estimates in a k–d tree, k-d trees being well suited to nearest neighbour queries [15].

Two methods for visualization have been implemented. An OpenGL based point rendering technique allows for fast rendering while sculpting. An image order iso–surface renderer has also been implemented. This renderer is too slow for interactive purposes, but produces images of higher quality.

Two applications have been written. A command line tool that generates CSG models serve as starting points for interactive sculpting. In addition, there is a tool for volume visualization and sculpting. So far this tool only enables the user to add or subtract spheres, but many other features are envisaged.

6 Results

The voxelization tool has been used to generate the following solids:

- SpheresA (Colour plate a) is a model consisting of a sphere of radius 80 from which a sphere of radius 66 has been subtracted, forming a bowl–like shape.
- Ellipsoid (Colour plate b) is an ellipsoid $x^2/a^2 + y^2/b^2 + z^2/c^2 - 1 = 0$ where $a = 80$, $b = 60$ and $c = 20$. A plane cuts off part of the ellipsoid.
- Cube (Colour plate c) is constructed from a voxelized plane. By taking the intersection of this model and five additional planes, the cube is created.
- SpheresB (Colour plate e) is a the union of a sphere of size 50 and a sphere of size 15.

The timings were performed on an 800MHZ Intel PIII based system running Linux. The timings are shown in table 2. The last column shows the timings for the alternative algorithm. The two columns in front of it, show the timings for the first pass and both passes of the normal algorithm.

A few of the details in table 2 are noteworthy. First of all, it is obvious that the run–time of the algorithm depends heavily on the choice of r. The CSG operation for the solid SpheresA takes almost twice as long for $r = 6$ as for $r = 2.5$.

Note also that a simple operation like adding a small sphere (SpheresB) takes no more than a second in both implementations. This is important, since the CSG operations have been implemented in an interactive system.

The alternative algorithm is somewhat slower than the standard algorithm, especially in the case of intersection. As mentioned, all voxels are visited when performing intersection. This magnifies any difference in performance and explains the issue. For small operations our experience (corroborated by the timings) is that the impact on interactivity is negligible. Hence, both algorithms are acceptable for interactive sculpting.

In the colour plate, a number of ray traced images are shown. Most of these have already been mentioned, except d and f which show sculpted models generated using our interactive program. The d image is also shown in figure 5 along with an image of the I–curves that were generated during the first passes of the CSG algorithm.

In some cases the implicit close in the volumetric union operation creates an object that does not have the openness property. In this case, the algorithm produces an object that looks correct until the surface collapses (see figure 5). The main problem is that such objects do not have the required volumetric properties to be used in further volumetric CSG operations.

7 Conclusions

A new technique for volumetric CSG has been proposed. Unlike previous methods for volumetric CSG, the result of a CSG operation is a volume where all voxels correspond to the signed shortest distance to the surface.

Model	Operation	Grid	r	Voxelization	Pass 1	Passes 1+2	Alt. alg.
SpheresA	$\setminus_v$	$256 \times 256 \times 256$	2.5	8.1	7.8	16.1	26.4
		$256 \times 256 \times 256$	6	9.1	13.0	29.3	89.8
Cube	$\bigcap_v$	$256 \times 256 \times 256$	4	1.3	1.7	3.8	44.4
					3.0	6.2	75.2
					1.7	3.7	44.4
					1.7	3.6	43.1
					2.9	6.1	73.1
Ellipsoid	$\bigcap_v$	$256 \times 256 \times 256$	4	8.1	32.6	67.8	72.6
SpheresB	$\bigcup_v$	$256 \times 256 \times 256$	4	2.2	0.4	0.8	1.0
			vu	seconds	seconds	seconds	seconds

Table 2. Timings

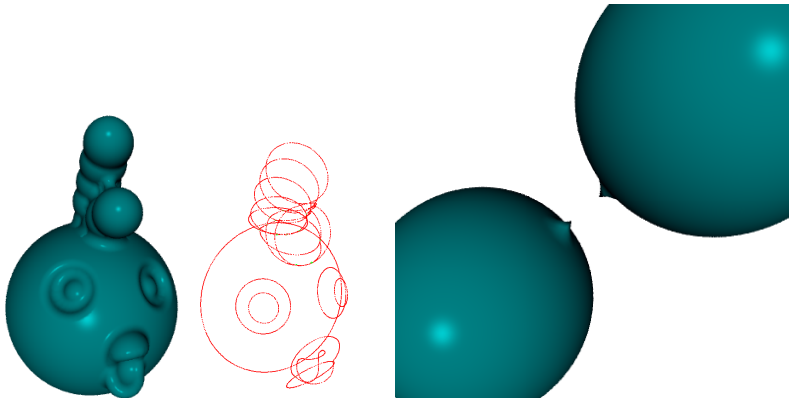

Fig. 5. Sculpted models. Model (left a) and the I curves (left b). An exaple of a model produced by $\bigcup_v$ that does not have openness property.

The technique assumes that the clamped signed distance V–model is used and that the input solids both have the openness and closedness properties. If these conditions are met, the openness and closedness properties are also preserved in the combined solid.

The only exception is when the combination of two solids yields a new solid that cannot at once be open and closed. This problem pertains to the V–model (and the volumetric representation in general), rather than to this technique for volumetric CSG.

For reasonable tool sizes and choices of r, the CSG operation is easily fast enough for interactive changes to the volume, and an interactive sculpting tool based on the CSG operations has been developed.

Future Work

As mentioned, the tool operand in a volumetric CSG operation is a V–model and not another volume. In general, this is advantageous, because voxelizing the

tool before the operation would only complicate matters and add overhead. It would, however, be nice to be able to do cut and paste operations and in that case, the tool must be a volume.

In some cases, the implicit close operation performed by the algorithm results in a solid that is not open. Furthermore, the result of a CSG operation on such a solid is not well–defined. The most important extension of this method is to change the algorithm so that it demonstrably handles these cases gracefully. How to do this requires further investigation.

References

1. Christoph M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, 1989.
2. Andreas Bærentzen, Miloš Šrámek, and Niels Jørgen Christensen. A Morphological Approach to the Voxelization of Solids. In Vaclav Skala, editor, *Proceedings of WSCG 2000*, volume I, February 2000.
3. Miloš Šrámek and Arie Kaufman. Alias–free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3), July/September 1999.
4. Sidney Wang and Arie E. Kaufman. Volume–Sampled 3D Modeling. *IEEE Computer Graphics & Applications*, 1994.
5. Min Chen, John V. Tucker, and Adrian Leu. CROVE – A Rendering System for Constructive Representations of Volumetric Environments. In *International Workshop on Volume Graphics, Swansea 1999*, 1999.
6. Shiaofen Fang and Rajagopalan Srinivasan. Volumetric-CSG – A Model Based Volume Visualization Approach. In *WSCG'98. The Sixth International Conference in Central Europe on Computer Graphics and Visualization'98*, 1998.
7. Andreas Bærentzen. Octree–based Volume Sculpting. In Craig M. Wittenbrink and Amitabh Varshney, editors, *LBHT Proceedings of IEEE Visualization '98*, October 1998.
8. Arie Kaufman, Daniel Cohen, and Roni Yagel. Volume Graphics. *IEEE Computer*, 26(7), July 1993.
9. David E. Breen. Constructive Cubes. In F.H. Post and W. Barth, editors, *Eurographics '91*, 1991.
10. Bradley A. Payne and Arthur W. Toga. Distance Field Manipulation of Surface Models. *IEEE Computer Graphics & Applications*, 12(1), 1992.
11. David E. Breen, Sean Mauch, and Ross T. Whitaker. 3D Scan Conversion of CSG Models into Distance Volumes. In Stephen Spencer, editor, *Proceedings of IEEE Symposium on Volume Visualization*, October 1998.
12. Sarah F.F. Gibson. Using Distance Maps for Accurate Surface Representation in Sampled Volumes. In Stephen Spencer, editor, *Proceedings of IEEE Symposium on Volume Visualization*, October 1998.
13. Jean Serra. *Image Analysis and Mathematical Morphology*, volume 1. Academic Press, 1982.
14. J. Andreas Bærentzen. *Volumetric Manipulations with Applications to Sculpting.* PhD thesis, IMM, Technical University of Denmark, 2001.
15. Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9), 1975.