Volume Sculpting: Intuitive, Interactive 3D Shape Modelling

Andreas Bærentzen

May 15, 2001

Abstract

A system for interactive modelling of 3D shapes on a computer is presented. The system is intuitive and has a flat learning curve. It is especially well suited to the creation of organic shapes and shapes of complex topology. The interaction is simple; the user can either add new shape features or smooth and deform existing features.

1 Introduction

Interactive modelling of 3D shapes on a computer should be as simple and intuitive as doodling 2D shapes using pencil and paper. Simpler, in fact, since on a computer changes can always be undone, and the user is more free to explore and experiment.

The fact that simple and intuitive 3D shape modelling is not quite "here" yet, we attribute to the fact that the present systems for modelling (or sculpting) 3D objects are based on shape representations which impose undesirable constraints on both the user interface and on the range of shapes that are possible.

In this paper, we present an intuitive sculpting system that is being developed at IMM. The system is based on the volumetric representation rather than the traditional surface oriented representations. The volumetric representation and volume visualization are normally associated with medical data and visualization but the volumetric representation is also very well suited to solid modelling. There are several reasons for this:

First of all, in volume graphics, operations which change the genus (i.e. change the number of holes) of the solid are trivially supported.

Likewise it is easy to support manipulations that may divide a model into separate parts. This is noteworthy, because in surface based representations, keeping track of topology increases the complexity of the system. That is why some systems, like for instance, the recent, gesture based and very intuitive system, Teddy [1], supports only shapes that can be deformed to a sphere.

Another salient point about volume graphics is that it seems to lend itself quite well to the sculpting of complex and organic shapes.

Finally, two technologies have proven to be very effective in the context of volume graphics:

- Constructive Solid Geometry (CSG).
- Level Set techniques [2]

CSG is a paradigm that has been used for many years in the context of solid modelling. Intuitively, CSG is about adding and subtracting shapes from other shapes. For instance the user may add or subtract a sphere from the solid he is working on. CSG is easily implemented in the context of volume graphics.

Level set techniques are very useful for both global and local deformations of shapes. In the sculpting system discussed in this paper, level set techniques have been used to implement local smoothing, global smoothing, tools for adding and removing material &c. Level set techniques also mesh well with the volume representation.

1.1 Outline

The next section is an introduction to the volume representation. Section 3 is about volumetric CSG and Section 4 is about deformative tools using the level set techniques. Section 5 is about the actual user interaction.

In the three final sections, we discuss some models created with the system, the software and hardware platforms, and finally some conclusions are drawn together with suggestions for further reading.

2 The Volume Representation

Perhaps the simplest way to understand the volume representation is that a volume is a stack of images as shown in Figure 1. Each image corresponds to a single slice of the represented object. In the volume representation, the pixels are called voxels. Just like a pixel is a small square a voxel can be seen as a small box. Thus, the stack of voxel images forms a 3D array of boxes that is called a volume.

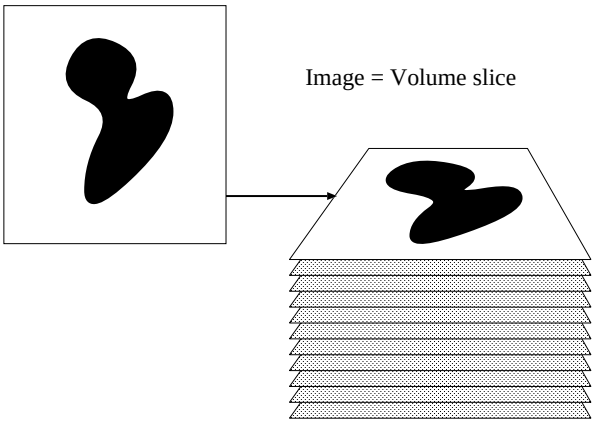

Figure 1: A volume seen as a stack of images.

According to a slightly different outlook, we can see a voxel as a point in 3D space (just like we can see a pixel as a point in a 2D space). Now, the voxels are the vertices of a 3D lattice (as shown in Figure 2.

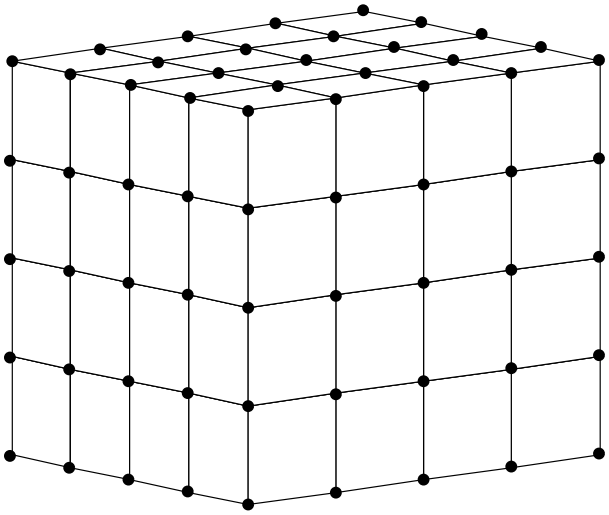

Figure 2: A volume seen as a 3D lattice of voxels

Voxels are characterized by their *position* and an associated *value*. When the volume representation is used for solid modelling it seems reasonable that the voxel value should be a binary variable indicating the presence or absence of matter in the immediate vicinity of the voxel.

However, this representation suffers from a problem known as *aliasing*. Aliasing turns up many places in computer graphics. Probably the simplest example is that in two–colour computer graphics a straight line becomes a jagged sequence of little black squares (pixels). To ameliorate the problem, we use antialiasing which usually means making the line a little fuzzy. The line is approximated using pixels of various shades of gray depending (for instance) on how close the line is to their centres.

A similar solution is used in volume graphics. Instead of storing only binary values, voxels contain scalar values that are sampled from a function which is somtimes called a *characteristic* function $f : R^3 \rightarrow R$.

Now, the obvious question is: How can the characteristic function, f, represent a solid? The answer is that the surface (boundary) of the solid should be embedded as an *iso–surface*, say the iso–surface corresponding to the value 0. In that case, and if we use the convention that the sign is negative on the interior:

$$f(\mathbf{x}) < 0$$

implies that $\mathbf{x}$ is on the interior side of the boundary.

$$f(\mathbf{x}) = 0$$

that $\mathbf{x}$ is on the boundary, and, finally,

$$f(\mathbf{x}) > 0$$

that $\mathbf{x}$ is outside the boundary.

Normally, if $|f| > \tau$ where τ is some constant, we record only whether we are inside or outside (basically, the sign of f) and not the value of f. This makes most of the algorithms faster, and enable a more compact representation of the solid.

The region where we store the actual value of f, i.e. $\{\mathbf{x}| \ -\tau < f(\mathbf{x}) < \tau\}$ is called the transition region.

The scheme is illustrated in 2D in Figure 3. The black region is interior, the white region exterior, and the fuzzy gray region is the transition region.

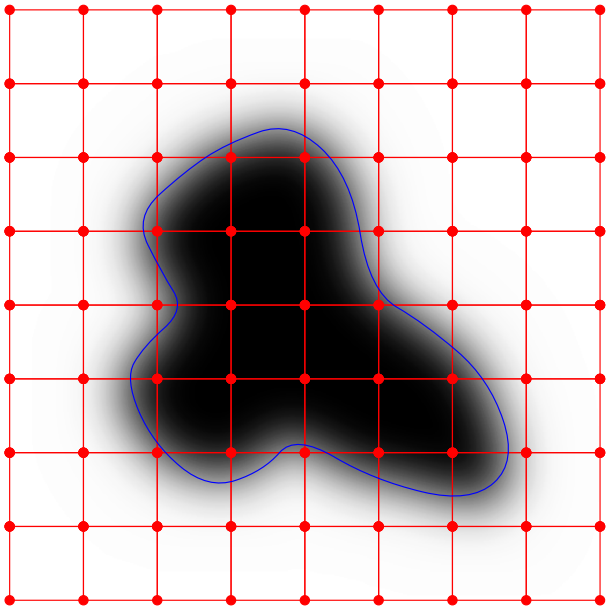

Figure 3: A slice of a volume. The grid is the voxel lattice. The iso–surface (iso–curve in 2D) is shown as a curve.

2.1 Voxelization and reconstruction

The process of generating volume data from a different representation is known as voxelization. Voxelization of solids amounts to the sampling of their characteristic function.

A typical example of a solid is the sphere. The characteristic function of a sphere with centre $\mathbf{p_0}$ and radius r might be and the distance function is

$$f(\mathbf{p}) = ||\mathbf{p} - \mathbf{p_0}|| - r \tag{1}$$

It is clear that $f(\mathbf{p}) = 0$ corresponds to the periphery of the sphere.

If we voxelize (i.e. sample) a characteristic function f at a reasonable resolution, it is possible to reconstruct f by interpolating in the volume. Most often simple linear interpolation is used. Reconstruction of the value of the characteristic function at an arbitrary point is frequently necessary – especially during rendering.

2.2 Rendering

The art of generating images from volumes is know as volume visualization. The two most important techniques for volume visualization are *iso–surface polygonization* and *volume rendering* by ray casting. Perhaps the best known example of the former technique is the Marching Cubes algorithm [3].

In the interactive program discussed below, a different technique known as point rendering is used. A dense cloud of points on the iso–surface of the represented solid are estimated on the basis of the transition voxels, and these points are rendered using OpenGL. The points are rendered so large that no holes appear. This very simple method is faster than the traditional volume rendering techniques, and yields an acceptable result for interactive purposes.

The higher quality images in this paper were rendered using ray casting.

3 CSG Operations

The simplest manipulation of a volume consists of changing the value of a single voxel. In principle, it is possible to manipulate volumes in that way, but it rarely makes sense. In general, we change many voxels at a time.

Assume that we have voxelized a solid A. A simple manipulation operation would be to voxelize a new solid B and combine the voxels of the new solid with the old. This could be done by superimposing the two volumes and for each voxel location combine the voxels v_A and v_B using some simple operation.

Let us call the resulting volume C. It turns out that we can use

$$v_C = \min(v_A, v_B) \tag{2}$$

on all voxels to generate the union of the solids. This operation can be explained by the observation that the shortest distance to the union of two solids is the minimum of the shortest distance to either.

In fact there is often no reason why the tool solid B should be voxelized, since the sampling of the characteristic function can be integrated into the CSG operation. Pseudocode for the volumetric CSG union of a volume and a solid represented by a characteristic function is shown below in Figure 4. Intersection and difference are also possible. For instance $\max$ is often used to compute the intersection. Unfortunately, CSG operations using min and max introduces sharp edges in the resulting solid. Sharp edges are problematic in volume graphics, since in general the boundary surfaces of geometric solids should be curvature limited to ensure a

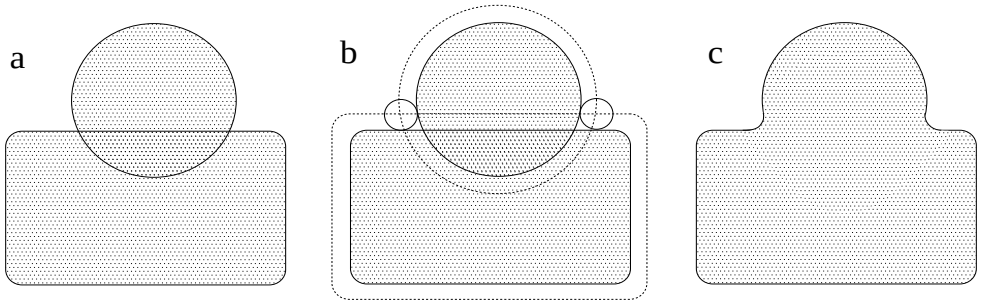

Figure 5: CSG operation

```
CSG_union(Volume vol, CharacteristicFun f)
{
  for_all( i,j,k in Vol)
    Vol[i,j,k] = min(Vol[i,j,k], f(i,j,k));
}
```

Figure 4: CSG union of volume and characteristic function

good reconstruction. To overcome this problem, blending is added to the CSG operations.

The result of a CSG union operation is shown in Figure 6 and a 2D diagram of the operation is shown in Figure 5.

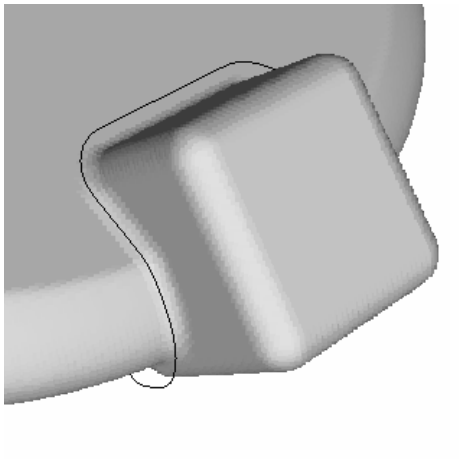

Figure 6: Locus of blending sphere

4 Deformative Operations

The shapes that can be created using only volumetric CSG are rather limited. The resulting shapes look to regular and do not exploit the fact that volume graphics is very suitable for organic shapes.

The remedy is to implement deformative methods, and a good way to do this is through Level Set methods. Basically, a level set is the same thing as an iso–surface (discussed previously in Section 2). The central idea is to deform the iso–surface by changing its embedding, i.e. the voxels in the volume.

Much has been written about the level set approach, and it would carry to far to repeat even the basic algorithm here[1]. Instead a slightly simplified explanation of the deformation technique will be given in the following.

First of all, the deformation is surface oriented. The boundary surface of the solid is pushed in the normal direction. To determine how much the surface is pushed we need a force function. Typically, the force function, say F, is defined in 3D, i.e. $F : R^3 \rightarrow R$ but it is only evaluated on the boundary surface. For a given point on the boundary, the value of the force function indicates how much the surface should be pushed in the normal direction.

For all voxels in the transition region, the closest boundary point is found, and the value of the force function is evaluated. The voxel value is then changed depending on the force. Say the voxel is v_{ijk}, and the closest boundary point is $\mathbf{p}_{v_{ijk}}$. The new value is sim-

[1] However, level set techniques are a very interesting field, and the interested reader is referred to [2] or `http://www.math.berkeley.edu/ sethian/`

ply the old value minus the value of the force function:

$$v_{ijk} = v_{ijk} - F(\mathbf{p}_{v_{ijk}}) \quad (3)$$

In our examples, values < 0 correspond to inside and values > 0 to outside. Hence, if the force function is positive, this operation pushes the boundary outward. Similarly if the force is negative, the object shrinks. As the object expands or shrinks, new voxels are added to the transition region as needed.

The value of the force function should not be too large. In general the value of F should not be larger than the distance between two adjacent voxels.

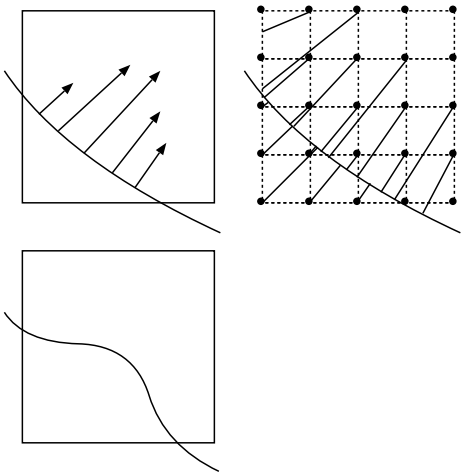

Figure 7: Illustration of deformation scheme: Normals scaled by the magnitude of the force function (top left), Lines indicate closest boundary points from voxels where the force function is sampled (top right). The iso–surface is changed (bottom).

It is very easy to design a force function for creating small bumps, but more sophisticated deformations are also possible. For instance, if the curvature of the surface is used, geometric smoothing of the surface is possible.

It is important to note that the deformations may resut in topological changes. For instance, a negative force function can be used to make a dent. Repeated application can be used to create a hole through the model. Since the surface is not explicitly represented but exists only as an iso–surface of the volume, this is not problem and such situations are trivially handled.

The implemented tools and their respective force functions are summarized below:

- Add material (positive force function)
- Remove material (negative force function)
- Smooth (Force depending on curvature)
- Un–smooth (Force depending on curvature. Sign reversed.)
- Dilate. (Constant force, inflates object.)

5 Interaction

Manipulation should be both simple and powerful. By powerful, we mean that the user should be able to control the placement and orientation of the tool completely and precisely. To retain simplicity, we have made the system modal. In one mode, the navigation mode, the user mainly controls the object. In the other mode which is intended for CSG, the user has more precise control over the CSG tool.

Initially, in navigation mode, the user is able to pan, rotate and zoom in on the object he or she is working on. When the user has found an appropriate angle to work from, a number of deformation tools and CSG tools may be applied.

The CSG tools cube, cylinder, tetrahedron and sphere may be selected directly by the user, but ellipsoids and convex polyhedra in general are also supported by the software framework. Deformation tools include a local smoothing tool and a matter tool that can be used to create small bumps or dents. There is also a global smoothing tool, and a tool for dilating the shape.

In navigation mode, the user places the tool on the surface of the object. This is done simply by moving the mouse over the desired location on the object surface. The initial position of the tool is found by unprojecting the screen space x,y,z position of the locator. The x,y position is simply the mouse coordinates, and the corresponding z value is obtained from the z–buffer. The initial orientation of the tool is either set to the surface orientation at that point or to the direction toward the eye point.

Once the tool is placed, the user simply presses a button to activate. Hence a point and a click is all it takes to do a CSG operation, to smooth a bit of the volume or to add a small bump. If the tool is a CSG tool, the user can lock the view (switching to manipulation mode) and rotate or translate the tool to a more precise location if desired.

When switching to manipulation mode, the initial position and orientation of the tool are used to define a coordinate system, the tool coordinate system – TCS, for further manipulation of the tool. The user can now perform the following operations.

- Rotation. The tool is rotated using a separate trackball. It is always rotated about its own center.
- XY Translation. The tool can be moved around in the XY plane of the tool coordinate system.
- Z Translation. Again in the TCS. This motion is useful for boring holes or to extend cylindrical shapes.

The described operations are all performed using the mouse in the graphics window. A separate control panel allows the user to set the size and type of the tool and to choose whether to add or subtract the tool shape.

6 Models

Some models created using the system are shown in Figure 8. A mask model is shown together with three head models. The simplest of these, head 1, is, in fact, the starting point for head 2 and head 3.

Head 1 is sculpted using only CSG operations. Head 2 is created from head1 through extensive use of the deformative tools for adding, subtracting and smoothing material. The fat appearance of head 3 was created by dilating head 2 and sculpting on from there. The beard on the mask model was created using negative smoothing.

7 Implementation

The system has been implemented in C++ using OpenGL for graphics and FLTK for the user interface. We have tested the system on PCs equipped with Geforce2 GTS graphics cards running either Linux or Windows.

In addition, a slightly modified version runs on a 16 processor SGI Onyx 2 with 3 InfiniteReality2E graphics boards in the UNI-C Virtual Reality Center. The Onyx is connected to a 6.5 by 2.5 m^2 stereo display, and it is possible to sculpt in stereo. Because each processor (MIPS R10000) runs at only 195 MHz, the computationally intensive manipulation operations are quite slow on this architecture. Hence, the CSG manipulation operations have been parallelized using the pthreads library.

8 Conclusions

A volume sculpting system has been presented. The system allows the user to edit a volumetric solid by constructive and deformative operations. These operations are provided through a simple and intuitive user interface.

The system runs on an ordinary PC running either Linux or Windows. This may be surprising since, until recently, volume visualization and volume graphics required hardware outside the range of pc–buyers; but it seems that the swift advances in PC technology and graphics cards has leveraged volume graphics in the past few years. This may also explain why the interest in interactive volume sculpting seems to be increasing currently.

8.1 Further Reading

This paper has only grazed the surface of volume graphics. An in–depth publication about the presented sculpting system has not yet been written, but there are a number of interesting publications:

- Recent sculpting systems: [4, 5, 6].
- Not–so recent sculpting systems: [7, 8, 9].

There is also a recent book entitled ‘Volume Graphics’ [10] which is a collection from the 1999 workshop on volume graphics. Finally, there is a single commercial (extremely expensive) sculpting system:

`http://www.sensable.com/freeform/`

References

[1] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proceedings of SIGGRAPH 1999*.

[2] James A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, second edition, 1999.

Figure 8: Mask model (top left), head 1 (top right), head 2 (bottom left) and head 3 (bottom right)

[3] W. E. Lorensen & H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Computer Graphics*, July 1987.

[4] Andreas Bærentzen. Octree–based volume sculpting. In Craig M. Wittenbrink and Amitabh Varshney, editors, *LBHT Proceedings of IEEE Visualization '98*, October 1998.

[5] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Practical volumetric sculpting. *the Visual Computer*, 16(8):211–221, December 2000.

[6] Alon Raviv and Gershon Elber. Three–dimensional freeform sculpting via zero sets of scalar trivariate functions. *Computer–Aided Desing*, 32:513–526, August 2000.

[7] Sidney Wang and Arie E. Kaufman. Volume sculpting. In *1995 Symposium on Interactive Graphics*. ACM SIGGRAPH, 1995.

[8] Tinsley A. Galyean and John F. Hughes. Sculpting: An interactive volumetric modeling technique. *ACM Computer Graphics*, 25(4), July 1991.

[9] Alan E. Richardson, Robert P. Burton, and William A. Barrett. A volumetric system for interactive three-dimensional design. *Journal of Imaging Technology*, 17(4):188–194, August/September 1991.

[10] Min Chen, Arie E. Kaufman, and Roni Yagel, editors. *Volume Graphics*. Springer, 2000.