Octree–based Volume Sculpting

J. Andreas Bærentzen *

Technical University of Denmark

Abstract

A volume sculpting system is presented. The system provides tools for interactive editing of a voxel raster that is stored in an octree data structure.

Two different modes of sculpting are supported: Sculpting by adding and subtracting solids, and sculpting with tools that are based on a spray can metaphor.

The possibility of extending the method to support multiresolution sculpting is discussed.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages I.4.10 [Image Processing and Computer Vision]: Image Representation—Hierachical; Volumetric

Keywords: volume rendering, isosurfaces, applications of visualization, multiresolution modeling

1 Introduction

By volume sculpting we will understand the interactive editing of a 3D object represented in a voxel raster. The purpose may be either shape modeling [3, 7] or the modification of an existing data set [1].

Voxel–based sculpting was introduced by Galyean and Hughes [3] in 1991 as a new method of creating free form 3D shapes by interactively editing a model represented in a voxel raster.

The user sculpts by moving a 3D locator around inside the voxel raster. Every time the locator moves, material is added by adding to the voxels in the raster in a small area around the locator. Similar tools exist for removing and smoothing material.

Kaufman and Wang have created another sculpting system where the tools are based on the notions of carving and sawing [7].

It seems that the tools introduced by Kaufman and Wang create more regular shapes, and that they are best suited for arts & crafts–like objects such as chairs, tables or lamps, while the tools introduced by Galyean and Hughes allow for freer and more organic shapes, although the method is not precise enough for objects of the kind that Kaufman and Wang aimed at.

The different modalities of the tools found in the previous projects have motivated the present work where the first aim is to create tools that allow the user to work in both modes.

The second aim concerns the data structure used to contain the voxel raster: In the previous systems the voxel raster has been stored in a 3–dimensional array.

The memory requirements of a realistically sized voxel raster are not prohibitive for a modern workstation but, since a voxel raster usually contains large homogeneously empty regions, a sparse representation would certainly be advantageous.

*Graphical Communication, IFP, Technical University of Denmark, Building 116, DK-2800 Lyngby, Denmark. **phone** +45 4525 1659 **email** jab@gk.dtu.dk

2 System overview

The implemented volume sculpting system runs on a UNIX workstation and uses a mouse for input.

On system start–up the user is presented with a *graphics* window where the visualized voxel model is displayed and the actual sculpting takes place. Auxiliary windows contain control panels for adjusting visualization parameters and parameters of the sculpting tools.

The system provides tools belonging to two different categories

- **CSG tools** The CSG tools are solids that may be subtracted from (set difference) or added to (set union) the object being sculpted. These tools are described in section 5.
- **Spray tools** The spray tools are for making small incremental changes on the surface of the model. With a spray tool the user may for instance add material to the model by spraying it onto the surface. Spray tools are described in section 6.

Every time the model is changed, the graphics window is updated, but only the part of the image where the change is visible is redrawn.

3 Voxel representation

The object being sculpted is represented in a $256 \times 256 \times 256$ voxel raster. Each voxel contains a value in the real interval $[0, 1]$, and the value (denoted the voxel *density*) is represented by 256 discrete levels.

Each voxel is represented by four bytes. One byte for the density and three that are reserved for uses related to color and material properties.

4 Octree data structure and rendering

A typical voxel raster contains large homogeneously empty regions, and by not representing these regions, it is possible to reduce the memory requirements substantially.

A traditional pointer based octree data structure [5] has been chosen to represent the voxel raster. An octree may not always be more compact than an array, but in my experience *almost any* realistic raster is more efficiently represented in an octree than in an array.

The volume is recursively subdivided until either the subdivided volume is empty, or the subdivision has reached the leaf level. In the first case the subdivided volume is an empty leaf node that is represented by a NULL pointer. If the leaf level has been reached a voxel is inserted. Hence, for volumes of size $256 \times 256 \times 256$ the leaf level is eight, since $2^8 \times 2^8 \times 2^8 = 256 \times 256 \times 256$

Visualization is performed with a ray casting technique that renders isosurfaces in the dataset. The choice of an image order technique is motivated by the fact that no individual sculpting operation requires the entire image to be updated, and with an image order technique, the problem of finding the voxels that contribute to the part of the image that needs updating is trivially solved [7].

Furthermore, ray casting can be significantly accelerated by the octree data structure, whereas other methods for fast volume rendering such as volume rendering using hardware supported 3D texturing [9] would be very difficult to combine with an octree data strucure.

The technique employed is similar to the one used by Levoy [4] except that the classification is binary, i.e. voxels with densities greater than the isovalue are wholly opaque, and for the sake of efficiency, nearest neighbour interpolation is used instead of trilinear interpolation.

The ray casting is accelerated by skipping over areas of the voxel raster that are represented by empty octree nodes. When an empty octree node is encountered, smapling continues from the intersection point where the ray leaves the node.

5 CSG tools

The CSG tools are function represented solids that can be added to, or subtracted from the voxel model. The actual operation is performed when the user presses the mouse button, and a simplified wireframe model of the tool is used for positioning.

CSG tools are similar to the carving tools in [7], but whereas the solid is represented by a voxel raster in Kaufman and Wang's system, here it is represented by a function $f : R^3 \rightarrow [0, 1]$. A point $\mathbf{x}$ where $f(\mathbf{x}) = 1$ is inside the solid, a point where $f(\mathbf{x}) = 0$ is outside and the area where $0 < f < 1$ is called the border region.

The important requirements of f are that it must be continuous, and the border region should be of a certain thickness.

If the thickness of the border region is very small compared to the distance between two voxel centroids, we will almost only obtain values of 0 or 1, when the function is volume sampled, and we know from [8] that binary sampling leads to *object space aliasing*.

The remedy is a thick border region. A border region of about three times the distance between two voxel centroids is appropriate.

As an example, a very simple function that represents a sphere can be defined as

$$f(\mathbf{x}) = \begin{cases} 1 & r < R \\ \frac{R+\Delta b-r}{\Delta b} & R < r < R + \Delta b \\ 0 & R + \Delta b < r \end{cases} \quad (1)$$

where $r = |\mathbf{x} - \mathbf{x_0}|$ is the distance from $\mathbf{x}$ to the centre of the sphere $\mathbf{x_0}$.

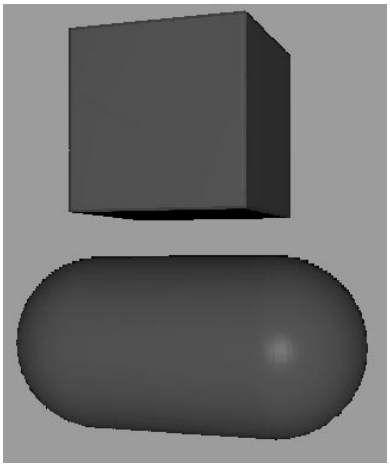

Figure 1: Two CSG tools

In figure 1 two of the more commonly used CSG tools are shown.

Adding the solid is performed by creating a new voxel model that is the union of the old voxel model and the solid. Like in [7] this is done on a "per voxel basis". For each voxel within the bounding box of the solid defined by f, we perform

$$newvox = vox \cup f(\mathbf{x}) = max(vox, f(\mathbf{x})) \quad (2)$$

where the point $\mathbf{x}$ is the centroid of the voxel and vox is the density of the voxel. $newvox$ is the density of the voxel after the operation. Similarly, it is possible to subtract the solid with the operation defined by

$$newvox = vox \setminus f(\mathbf{x}) = min(vox, 1 - f(\mathbf{x})) \quad (3)$$

6 Spray tools

With the mouse, the user positions a 2D locator somewhere on the image of the model being sculpted. When the tool is activated, a *viewing ray* is cast from the position of the locator in the graphics window and into the raster.

The first point on the surface that is hit, becomes the tools *centre of influence*, and an operation is performed in a small (e.g. $9 \times 9 \times 9$ voxels) area around the centre of influence.

There are two spray tools, the *metamatter* tool and the *smoothing* tool.

The metamatter tool is for adding or subtracting material. For each voxel in the affected (spherical) area, the following computation is performed

$$newvox = min(0, max(1, vox + i\frac{R - r}{R})) \quad (4)$$

where r is the distance from voxel vox to the centre of influence, i is the intensity, and R is the radius of the affected area. If $i > 0$ material is added, otherwise subtracted. This operation is similar to the one used in [1].

The smoothing tool works by filtering the voxels in the area around the centre of influence with a $3 \times 3 \times 3$ averaging filter.

An important aspect of the spray tools is that modifications are usually performed in a series. For instance, (with the metamatter tool) if the user holds down the mouse button and drags the mouse, a trail of material will be drawn on the surface of the model.

7 Implementation

The system has been written in C++ and tested on SGI/Irix and Intel/Linux platforms. All cited measurements have been performed on the Linux platform which is a Pentium Pro 200 MHz with 64 MB RAM.

8 Sculpted models

Figure 2 shows a crude model of a troll–like creature. The earring was created with a toroidal CSG tool.

The tongue was created by adding material with the metamatter tool, and the ear was created mainly by removing material with the same tool.

Figure 3 shows the most ambitious model yet. The work process is illustrated in the video accompanying this paper.

The initial shape was a rectangular "slab" of voxels, and in the beginning of the sculpting process, the overall shape was worked by adding and removing large amounts of material with the CSG tools.

Later the features were added, mostly with the metamatter tool, and the features were smoothed with the smoothing tool. It is interesting to note that the combination of the metamatter tool and the

Figure 2: Troll–like creature

Model	Octree size	Raster utilization
head	7.65 MB	8.5 %
troll–like creature	6.68 MB	5.9 %
chair	3.73 MB	2.5 %
sphere (unit diameter)	45.59 MB	54 %

Table 1: Sizes and raster utilization

smoothing tool is very well suited for creating smooth, spline–like, surfaces.

As an example of a furniture model, a chair has been sculpted (figure 4.) It is made almost exclusively with CSG tools, although the spray tools have also been employed (especially for the seat).

All the models are sculpted in a raster with a resolution of $256 \times 256 \times 256$ voxels. The sizes of the cited models and a sphere that exactly fits the volume, are shown in table 1.

There is a great difference between the time it took to sculpt the three models. Sculpting the troll (figure 2) took roughly half a day, while the head (figure 3) took several days. The chair (figure 4) was sculpted in less than an hour.

The figures 3 and 4 are ray traced images. The sculpted vol umes were polygonized, using an implicit surface polygonizer by Jules Bloomenthal [2], and the resultant surfaces were rendered with POVRay. The other figures were volume-rendered with the presented system.

9 Performance measurements

Since spray tool operations are performed in a series, it is important to insure that each individual operation is so fast that a series of operations feels like one continuous operation.

For most operations with the metamatter tool, between roughly 200 and 600 pixels must be updated for each modification.

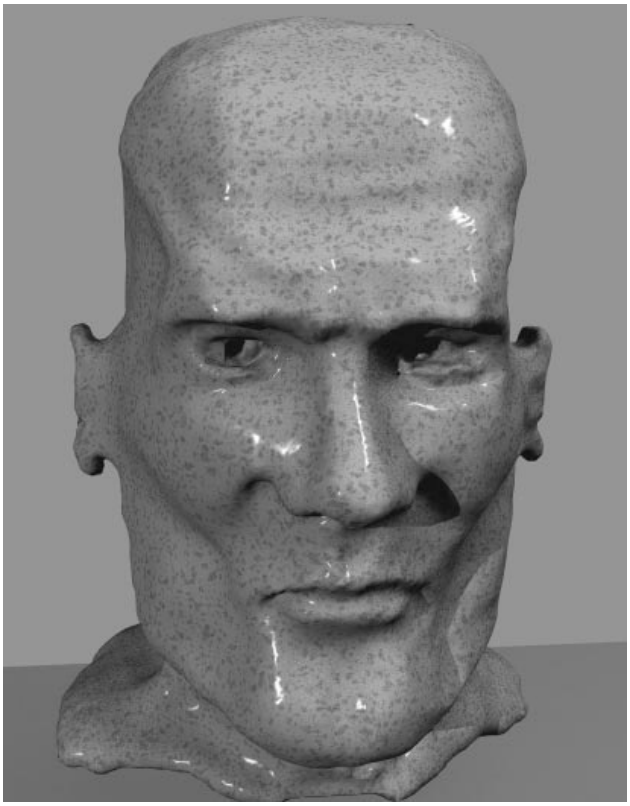

Figure 3: Head

During a sessions with a metamatter tool that modifies a $7 \times 7 \times 7$ area, 999 modifications and image updates were performed. An average of 205 pixels had to be updated each time, and the average time for one data modification and image update cycle was 0.031 seconds.

An otherwise identical experiment with an $11 \times 11 \times 11$ tool resulted in an average number of 577 pixels modified per update, and the average update time was 0.083 seconds.

These numbers yield update rates of between 32 Hz (clearly real–time) and 12 Hz (near real–time). Measurements were also carried out for a smoothing tool with a size of $9 \times 9 \times 9$ voxels. The result was roughly 7 Hz which is less "near real time", but in practice acceptable, and unsurprising since this tool performs a filtering operation. The timings are summarized in table 2.

Tool type	size	update rate
metamatter	$7 \times 7 \times 7$	32 Hz
metamatter	$9 \times 9 \times 9$	17 Hz
metamatter	$11 \times 11 \times 11$	12 Hz
smooth	$9 \times 9 \times 9$	7 Hz

Table 2: Spray tool update rates

10 Extending the method to multiresolution sculpting

The octree data structure has proved to be a very efficient and flexible way of storing the volume, but it has one more important possibility that has not been explored yet. By allowing voxels to be in-

Figure 4: Chair

serted at different levels in the octree, one would obtain a sparsely represented voxel raster with *dynamic resolution*, and this is the main future goal for this project.

The initial data structure for multiresolution volume sculpting has been designed, and it is quite similar to the one presented in section 4, except that there is no leaf level since leaf nodes may be inserted at any level.

This has two important implications:

- Non–empty homogenous regions may be grouped together and represented by a voxel at a lower level of subdivision, thereby storing the volume more efficiently.

- Fine details may be added at a high level of subdivision. This is especially important, since it enables us to have high resolution only where it is needed.

The basic mode of operation in the multiresolution system will be much like it is in the system presented, except that when the user chooses a tool, he not only chooses the type (CSG tool or spray tool) and size, but also the level of subdivision where voxels are to be inserted into the octree.

This makes little difference when voxels are added to empty parts of the octree, and in the case where a tool with a high level of subdivision is used on an area that is subdivided to a lower level, it is obviously necessary to subdivide the original area to the same level as the tool before applying (2), (3) or (4).

When a coarse tool is used on an area with higher level of subdivision, one approach would be to first coarsen (e.g. by averaging) the target area and then apply the tool.

This approach could easily lead to aliasing, however, and it is probably a better approach to always let the highest level of subdivision prevail, and then supply the user with a separate “coarsening” tool.

11 Conclusions and future work

A system has been created that combines two different modes of sculpting: Sculpting with CSG tools and sculpting with spray tools.

The first kind of sculpting is very useful for creating large and regular shapes while the second kind of sculpting allows for organic and more complex shapes.

The examples all show models that are sculpted from voxelized primitives, but this need not be the case. Other applications such as bone or dental reconstruction on the basis of acquired medical data–sets, are easily envisaged.

While the main scope of my future research will be the exten sion of the method to multiresolution sculpting, other areas are also worth studying.

Material properties would make the system more interesting as a tool for editing acquired data sets. Especially if the effects of the various tools were made dependent on the kind of material they operated on.

Many ideas from 2D image manipulation, should be equally use ful in a volume sculpting context. A deformation tool to twist, bend or squeeze parts of model would make an important addition, as would a facility for cutting, copying and pasting volume elements.

Finally, some shapes could be approximated more quickly if the CSG operations were made ”softer”, and this could probably be accomplished by adding blending [6] to the set union (2) and set difference (3) operations.

References

[1] Ricardo S. Avila and Lisa M. Sobierajski. A haptic interaction method for volume visualization. In Roni Yagel and Gregory M. Nielson, editors, *Visualization ‘96*. IEEE, 1996.

[2] Jules Bloomenthal. An implicit surface polygonizer. In Paul S. Heckbert, editor, *Graphics Gems IV*. Academic Press, 1994.

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

[4] Marc Levoy. Display of surfaces from volume rendering. *IEEE Computer Graphics & Applications*, 8(3), 1988.

[5] Donald J. Meagher. Efficient synthetic image generation of arbitrary 3-d objects. *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, June 1982.

[6] Pasko, Adzhiev, Sourin, and Savchenko. Function representation in geometric modeling: Concepts, implementation and applications. *The Visual Computer*, 11(8), 1995.

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

[8] Sidney Wang and Arie E. Kaufman. Volume sampled voxelization of geometric primitives. In Gregory M. Nielson and Dan Bergeron, editors, *Proceedings, Visualization 93*. IEEE, 1996.

[9] Rüdiger Westermann and Thomas Ertl. Efficiently using graphics hardware in volume rendering applications. In *SIGGRAPH 98 Conference Proceedings*. ACM SIGGRAPH, 1998.