Edge Preservation in Volume Rendering Using Splatting

Jian Huang ***(huangj@cis.ohio-state.edu)***
Roger Crawfis ***(crawfis@cis.ohio-state.edu)***
Department of Computer and Information Science
The Ohio State University

Don Stredney ***(don@osc.edu)***
Ohio Supercomputing Center

Abstract:

This paper presents a method to preserve sharp edge details in splatting for volume rendering. Conventional splatting algorithms produce fuzzy images for views close to the volume model. The lack of details in such views greatly hinders study and manipulation of data sets using virtual navigation. Our method applies a non-linear warping to the footprints of conventional splat and builds a table of footprints for different possible edge positions and edge strengths. When rendering, we pick a footprint from the table for each splat, based on the relative position of the voxel to the closest edge. Encouraging results have been achieved both for synthetic data and medical data.

1. Introduction

Surface models offer a sharp representation of a particular iso-contour surface. Volume rendering can mimic these surfaces rather well using a ray-tracing scheme [Levoy88]. Polyhedral projection schemes for volume rendering can split the volume by a contour surface and render a volume interval with a sharp inter-voxel edge. However, these techniques are very expensive. Baining Guo [Guo 95] uses alpha shapes to calculate a polyhedra model of the interval between two iso-contour surfaces. Likewise, Issei Fujishiro [Fujishiro95], et al, generate tables similar to the marching cubes tables, but which output polyhedral elements bounded by two iso-contour surfaces. Both of these techniques are two-pass algorithms, generating a polyhedral model first, which must then be sorted and volume rendered. Max, Hanrahan and Crawfis [Max90] imbedded an iso-contouring method in their polyhedral volume renderer. Here, not only is the polyhedral model split by the iso-contour surface, but a semi-transparent glass surface can also be integrated into the visualization. All of these techniques are computationally expensive and not amenable for fast direct volume rendering.

The problem of enhancing edges or details is different from the problem addressed in [Swan 97] and [Mueller 98], where many splats projecting to the same pixels need to be blurred out to avoid aliasing. The appearance of the surface is dictated by a sharp transition from one material to another and also by the light reflections from the surface. To date, it is very difficult to maintain these sharp transitions with a volume renderer using a splatting approach. When the splats project to a very small area of the screen, sharp changes are certainly viewable, but a problem arises when one wishes to zoom in or through a voxel model. The reconstruction process inherent within splatting blurs out these (perhaps artificial) high frequencies. For many of the other volume rendering techniques, this effect may also be a problem when using just simple tri-linear interpolation.

Interpolation-based techniques have the advantage, however, in that, for a minor increase in cost, they can develop new (non-linear) interpolation techniques to sharpen or preserve these edges. Levoy [Levoy88] uses the gradient of the scalar field to ensure that a contour surface is not missed. This function blurs out the opacity transfer function in regions of high gradient. To date, no one has addressed the problem of preserving edges or surface boundaries within the volume in the context of direct volume rendering, and in particular, in the context of splatting.

2. Motivations and General Idea

For many medical applications, important features in structures of the human body, such as nasal passageways [Stredney98], colons [Kaufman97], and even blood vessels need to be studied or simulated by navigating through a virtual simulation. These features may comprise only a small number of voxels in their cross sections. With the conventional splatting renderer, these fine structures are blurred and navigation is hampered by the lack of contrast and detail. However, volume rendering is preferred over surface model for applications that need to deform or modify the underlying volume.

Likewise, for computational simulation, many users require the crisp, familiar representation of surfaces but with the ability to see the entire 3D volume. Mixed semi-transparent contour surfaces and volume rendering are still difficult. The techniques presented here allow for a more crisp surface representation while preserving the volume rendering.

Much research is needed into techniques that preserve edges in volume rendering. We examined two possible paths toward enhancing the edge details of a voxel model. The first technique was to simply add new voxels in the areas where we wished to induce high frequencies or edges, similar to supersampling. We explored various hierarchical schemes to add these new data points, either as a pre-process or as a view-dependent process. The second technique entails a direct manipulation of the underlying reconstruction kernels and resulting footprints used in the splatting process. This latter technique is the focus of this paper.

Traditional splatting uses the same footprint for all voxels. As we zoom in, the overlapping interpolation kernels ensure a smooth transition from one voxel to another, preventing a preservation of the edges. Our research has asked whether we can use different footprints for different voxels near an edge With the idea of manipulating the underlying reconstruction kernels, our goal was to imbed into the footprint tables a non-linear and anisotropic interpolation or reconstruction function. Crawfis and Max [Crawfis93] successfully constructed a large table of anisotropic splats to represent flow fields. Here, each voxel selected an appropriate footprint from the table, based on the voxel's projected vector field direction. To reconstruct or fabricate sharpened edge details from discrete data samples, we needed non-linear operations, and we constructed a similar table for our purposes.

3. Manipulations of Kernel Footprints

This section describes the kernel manipulation process we have developed. We first describe the basic goals and approaches in designing the new footprints and then include some implementation details. Finally, we present some results from both synthetic and real datasets.

3.1 Some considerations in designing new footprints

First, we wanted the choice of which footprint to use to be based solely on local information of the voxel data point only. Second, we wanted to support edges at different positions with different strengths and slopes. This criterion is essential for an accurate representation of an edge. Third, preservation of $\mathbf{C^1}$ continuity is desired. Discontinuity in the footprints introduces artifacts and aliasing in the images. Fourthly, the profile of the reconstructed edge should be reasonable, with a monotonoic shape and no additional notches. This criterion prevents sharpened edges from having unrealistic hill and valleys.

3.2 Compressing the edge effecting regions in footprints of kernels

For our C^1 constraint, an edge always strides over some finite region, and relatively speaking, the sharpness is dictated by the size of this region. Our principal idea was to contract the footprint around this region. This action increases the slope of the function in the area, and sharpens the edge. See Figure 6(b) for a profile of a compressed splat. For proper reconstruction on both sides of the edge, we need to adjust the other regions of the splat accordingly, as well as the footprints of neighboring voxels.

Our new method is based on the kernel Crawfis and Max [Crawfis93] optimized to achieve optimal reconstruction in areas of uniform intensity. This kernel is a low-pass filter that, when applied to discrete spatial data, around the edge, causes blurring of the image. This kernel decays rapidly to zero at an extent of about 1.6 voxel units. This small splat enables us to keep our stretching/compressing operations as local as possible. Using this kernel, for any point in 3D space, only 64 (4x4x4) space reconstruction basis functions come into play. Therefore, on the direction perpendicular to the edge, no more than 4 consecutive voxels, with 2 on each side of the edge, need to be taken care of.

As shown in Fig.1 below, only the shaded region of the splats in columns 0, 1, 2 and 3 interact with the edge. The four circles denote the extent of the four voxels on row i, and columns 0, 1, 2 and 3, respectively.

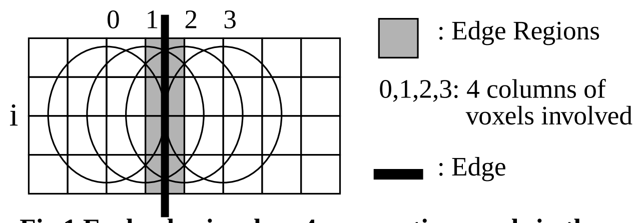


Fig.1 Each edge involves 4 consecutive voxels in the direction perpendicular to it.

Practically, edges don't often align strictly in the direction of voxel grids. When the orientation of the edge is rotated, it is problematic to still define the kernels, 0, 1, 2 and 3 along the direction perpendicular to the edge, as above. We thus introduced the concept of primary kernels and secondary kernels. Primary kernels are the ones with edges closer than 1.0 voxel units to the center. Secondary kernels have edges still within a splat's extent, but farther than 1.0 voxel units to the center. Figure 2, below, illustrates this idea. Since the splats are rotationally symmetric, any edge direction can be dealt with by rotating a canonical edge-enhanced splat. For the rest of this discussion, we will therefore use a vertical edge for illustration

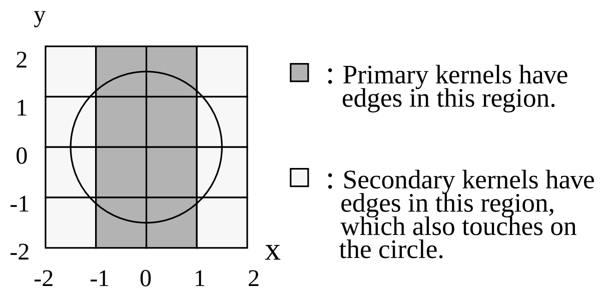

y
2
1
0
-1
-2
x
-2
-1
0
1
2
: Primary kernels have edges in this region.
: Secondary kernels have edges in this region, which also touches on the circle.

Fig. 2 Primary and Secondary Kernels. (Assume a vertical edge exist)

According to this definition, the kernels on columns 0 and 3 in Fig.1 are secondary kernels, and the kernels on columns 1 and 2 are primary kernels.

3.3 The stretching function

The stretching function we use for the shaded region in Fig. 1 applies both to primary and secondary kernels and is illustrated in the diagram below. The splats are stretched/compressed only on the direction perpendicular to the edge. Hence, we assume that the edge is locally linear.

We divide the shaded region in Figure 1 into three parts, which we label Interval 1, 2 and 3. The edge lies within Interval 2, and we wish to compress this region to a much smaller region surrounding the edge. To ensure smoothness and an accurate reconstruction, the footprint must be stretched in Intervals 1 and 3. We define two parameters t_1 and t_2, which separate Interval 1 and 2 and Interval

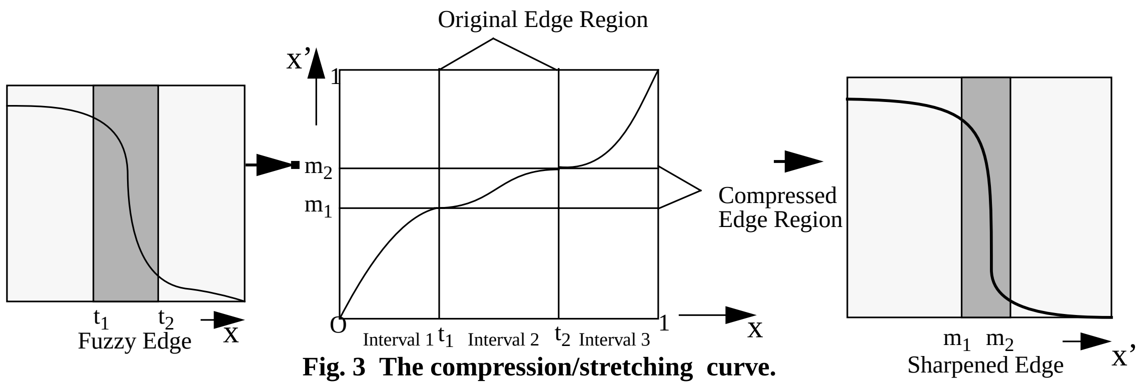


Fig. 3 The compression/stretching curve.

2 and 3, respectively. The smaller region to which we compress Interval 2 is specified using two additional parameters, m_1 and m_2. Hence, t_1 will be warped to m_1, and t_2 will be warped to m_2. Together, t_1, t_2, m_1 and m_2 also specify the strength of the edge being constructed.

To achieve this compression/stretching of the basic splat footprint, we define a change of variables mapping. Let the values in the isotropic splat be g(x,y) and the values in the stretched/compressed splat be h(x,y). We denote the compression/stretching function in Fig.3 as $x = map(x')$, which maps from [0,1] on the vertical axis to [0,1] on the horizontal axis. We need to manipulate only one-half of each splat for values of x between 0 and 1.6 for the splats 0 and 1 in Figure 1, and x between -1.6 and 0, for the splats 2 and 3 in Figure 1. Because of the anti-symmetric nature of the map(x), the stretching/compressions applied to the Splats 0,1,2 and 3 in Figure 1 are all different. We desire a simple relation between g(x,y) and h(x,y) and need to determine the mapping functions such that:

For Splat 0 in Figure 1 (secondary kernel with larger data value):

$$h(x, y) = \begin{cases} g(map(x-1)+1, y) & if(x \geq 1) \\ g(x, y) & else \end{cases}$$

For Splat 1 in Figure 1 (primary kernel with larger data value):

$$h(x, y) = \begin{cases} g(map(x), y) & if(0 \leq x \leq 1) \\ g(x, y) & else \end{cases}$$

For Splat 2 in Figure 1 (primary kernel with lower data value):

$$h(x, y) = \begin{cases} g(map(1+x)-1, y) & if(-1 \leq x \leq 0) \\ g(x, y) & else \end{cases}$$

For Splat 3 in Figure 1 (secondary kernel with lower data value):

$$h(x, y) = \begin{cases} g(map(2+x)-2, y) & if(x \leq -1) \\ g(x, y) & else \end{cases}$$

Note that the above "primary and secondary kernels with larger or lower data value" are specified for each splat. Because the compression/stretching function is not symmetric, we define the way of differentiating which side a splat is on by how large its data value is. This specification can certainly be switched, as long as the definition is consistent whenever it is applied.

Now, let's analytically define the function map(x). Since it's always more intuitive to study a function that maps from the horizontal axis to the vertical axis, let's define p as the inverse function of map(x), i.e. $map^{-1}(x)$.

To allow for flexibility in positioning the edges as well as specifying different strengths, we chose a cosine profile $p(x) = m_1 + \frac{m_2 - m_1}{2} \bullet \left(\cos\left(\left(\frac{x - t_1}{t_2 - t_1} \times (-\pi)\right) + 1\right)\right)$ as the compressing curve for Interval 2. For Intervals 1 and 3, we use a Cubic Spline of the form: $p(x) = ax^3 + bx^2 + cx + d$. The values of a,b,c and d can be obtained from the boundary conditions specifying the function value and the first derivative at the two end points of Interval 1 and Interval 3.

Interval 1:
$$\begin{bmatrix} p(0) \\ p'(0) \\ p(t_1) \\ p'(t_1) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ m_1 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \frac{-2(m_1 - t_1) - t_1}{t_1^3} \\ \frac{3(m_1 - t_1) + t_1}{t_1^2} \\ 0 \\ 1 \end{bmatrix}$$

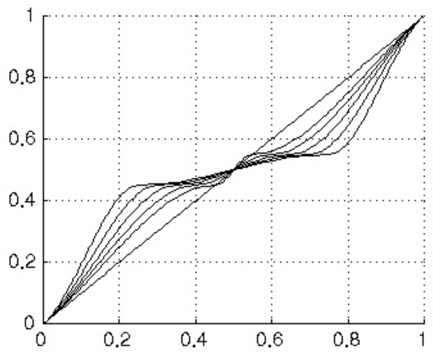
Interval 3:
$$\begin{bmatrix} p(t_2) \\ p'(t_2) \\ p(1) \\ p'(1) \end{bmatrix} = \begin{bmatrix} m_2 \\ 0 \\ 1 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{t_2(t_2 - 1)^2} \bullet \begin{bmatrix} m_2 - t_2 \\ -2t_2^2 + (3m_2 - 4)t_2 + 3m_2 \\ 4t_2^2 + (-6m_2 + 7)t_2 - 9m_2 \\ -2t_2^2 + (3m_2 - 6)t_2 + 5m_2 \end{bmatrix}$$

Since we impose the first derivative of p(x) be 1 at $x = 0$ and $x = 1$, C^1 is preserved across the splats. We can see this effect by examining the equation:

$$h_x'(rmap(x)) = g_x'(rmap)rmap'(x)$$

Where rmap(x) is any one of $map(x-1)$, $map(x)$, $map(x+1)-1$ and $map(x+2)-2$, having a first derivative of $rmap'(x) = map'(x)$, for each choice. When $x = 0$ or $x = 1$, $rmap'(x) = map'(x) = 1$, then $h_x'(rmap(x)) = g_x'(rmap) \cdot 1 = g_x'(rmap)$.

The general shape of the computed stretching and compressing profiles for Intervals 1, 2 and 3 are shown in Figure 4 for different edge strengths. The straight line y=x is drawn as a reference for comparison and indicates the case in which no compression/stretching is applied to the footprint. The compression/stretching curves for several different extents, corresponding to increasing the width of Interval 2 (the region between t_1 and t_2), are also shown. As the width between the two t values increases from 0.1 to 0.5, we increases the compression of Interval 2.



$t_{1,2} = 0.5 \pm (i \bullet 0.05)$, *with* m1:0.45, m2:0.55

Fig.4 Samples compression/stretching Curves

3.4 Reconstruction on synthetic data sets.

Here we show how to apply the stretching/compressing functions to reconstruct edges. Please note, by compressing/stretching the 2D Crawfis kernel, we get edge preservation kernels for 2D images. By compressing/stretching the reconstruction kernel from the 3D kernel, we get the reconstruction kernels for edge preservation. These kernels then need to be integrated for all possible views. We choose to focus on the manageable and efficient procedure of manipulating the footprint functions. Future research is needed to compare this approximation to a more accurate solution for volume rendering.

We use four different footprints for the four consecutive voxels on the direction most perpendicular to the edge. A comparison between conventional splatting and our new edge preservation

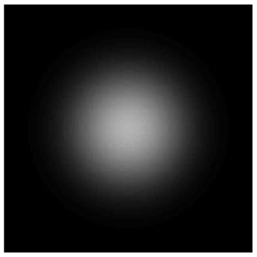


Fig.5 Crawfis's Splat

method is illustrated in Figure 6 below for a 1D cross section. To reconstruct an edge stepping from a value of 1 to a value of 0.5, (using t_1:0.1, t_2:0.8, m_1:0.2, m_2:0.4), we multiply the corresponding weights with the appropriate kernels at each voxel. The step-shaped curves shown at the top of the two diagrams represent the step function we are attempting to reproduce.

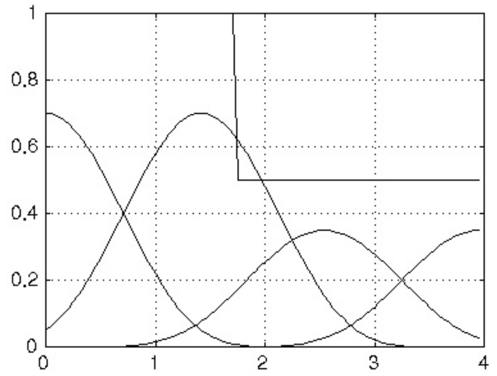
Note that in Figure 6(b), the splats with compression/stretching, have much sharper decreasing regions than the optimal splats in Figure 6(a). The four corresponding footprints for these kernels are shown in Figure 7 from left to right.

The reconstructed function for this sample edge is illustrated in Figure 8. The left side of Figure 8 shows the 1D cross section profile of the reconstructed 2D edge, along with the smooth curve generated using normal splats. The diagram on the right side is an image of the sharpened edge generated using several rows of the splats in Figure 6(b).

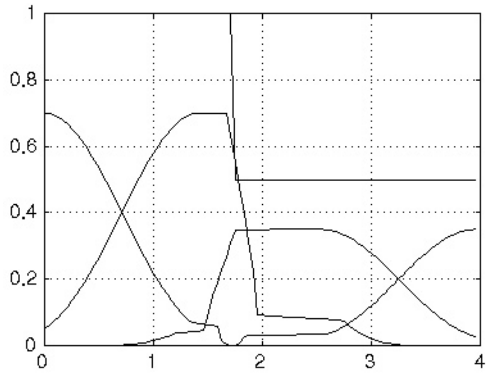
3.5 3D Implementation

For interactive rendering, we precomputed the splats at different edge positions and strength values. While rendering, we loaded the precomputed splats into texture memory and use textured polygons[Crawfis93].

So far, our analysis has dealt with edges in two dimensions only. For arbitrary 3D views, we can have edge directions pointing to any point on the 3D sphere. This section discusses our solution. Of course, if the edge direction is perpendicular to the viewing direction, we arrive back at our 2D edge reconstruction. When the viewing direction is parallel to the edge direction, the edge is imperceptible and no special treatment is needed for these voxels. For the cases between these two extremes, we project the edge



(a) Profile of Conventional Kernels



(b) Profile of Edge-Enhanced Kernels

Fig.6 The relative position of the 4 kernel involved in reconstruction.

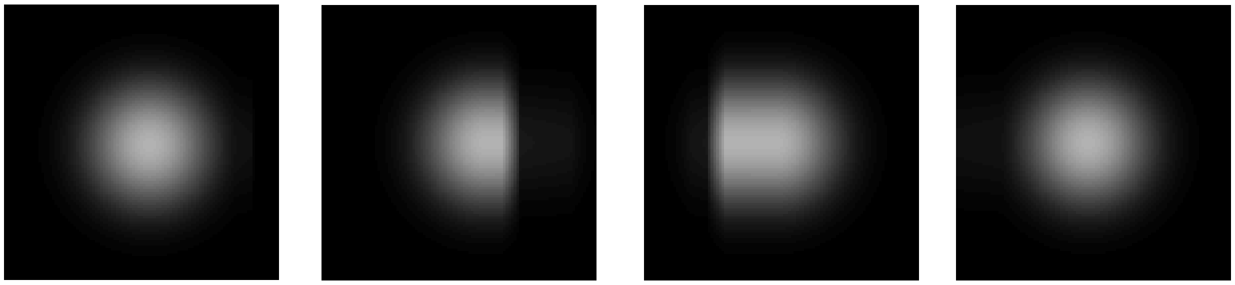


Fig.7 Corresponding footprints used for edge-enhanced reconstruction. From left to right, correspond to the 4 kernels in the right diagram in Fig.6(b)

direction onto the viewing plane. We then use this projected length to select a strength for the edge. We store a normalized edge direction separately from the edge distance or position for this purpose. For each voxel that needs an edge enhanced splat, we choose the footprint according to the projected edge position and strength, from the corresponding precomputed footprint table.

We have analyzed our method on a synthesized dataset with very low resolution, comprising a *12x12x12* grid. This dataset simulates small passageway, like a cube with a cylindrical hole in it. When viewed in a direction parallel to the hole with the conventional splatter, we obtain the image on the left in Figure 9. With our edge preservation method, we achieve the image on the right. The contrast here is obvious, and our method produces a nice sharp edge.

4. Challenges for Practical Applications and Current Results

This method of edge preservation requires an edge detection/estimation scheme with fairly good accuracy. Iso-contouring produce a sharp discontinuity in the volume, and we can obtain fairly good estimates of the edge direction and contour distance for each voxel. This distance is estimated using the gradient and the difference between the voxel value and the contour value [Levoy88], where the gradient corresponds to the edge direction. For experimental or acquired data sets, such as CT or MRI data, advanced techniques of edge detection, model extraction, data segmentation etc., are indispensable. Unlike most of the previous work in this area, we desire more information than a simple binary classification. Discussions about existing or needed research into these technologies are beyond the scope of this paper.

4.1 Sinus Cavity Dataset

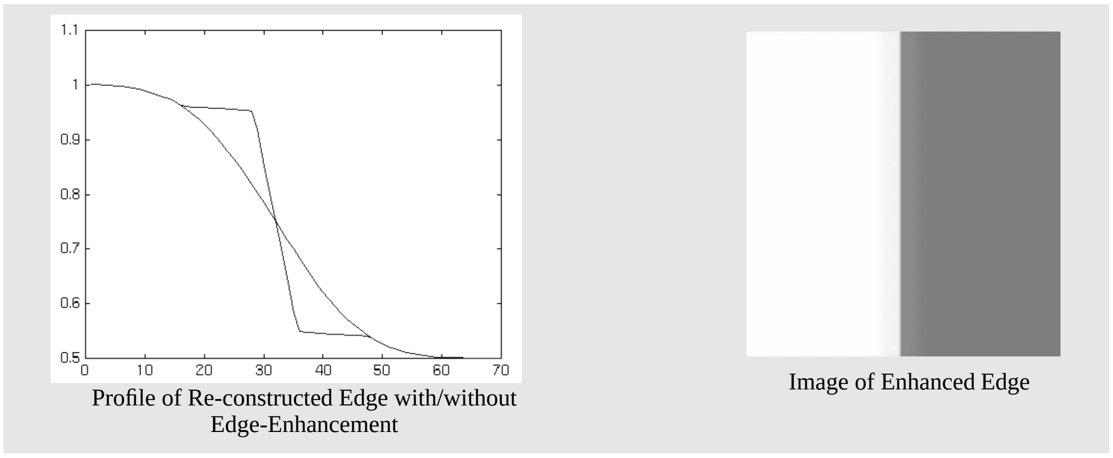
We have, however, applied our technique to a sinus cavity dataset derived from the Visible Human dataset. We applied a low pass filtering (3x3x3 averaging) and a binary threshold at 128. Estimates of gradients were based on central differences, and edge position estimation based on trilinear interpolation.

Having the gradient for each splat allows us to preform view-dependent light source shading. For the images shown, we used directional light located to the top left of the viewer. We apply this lighting only to those voxels labeled primary or secondary edge splats, regardless of whether an edge enhanced footprint is used.

Figures 10 and 10c (color) show various images from a virtual navigation through this dataset. As can be seen, in regions of close-up views, such as on the sidewalls, and regions of coarse resolution in the cross-section, such as the small hump to the right, edge preservation enhances the images. The details on the walls are also considerably enhanced. Because the inner surface of the data is not very smooth, we can see small extrusions comprising only several voxels protruding into the cavity. In images rendered with normal splatting algorithms, these details are not very visible.

4.2 Hipip Dataset

We have also applied our method to enhance details in the Hipip (HIgh Potential Iron Protein) data set, which is available from ‘ftp://omicron.cs.unc.edu/pub/projects/softlab/CHVRTD/volI’,



Profile of Re-constructed Edge with/without Edge-Enhancement

Image of Enhanced Edge

Fig.8 The profile and image of the reconstructed edge

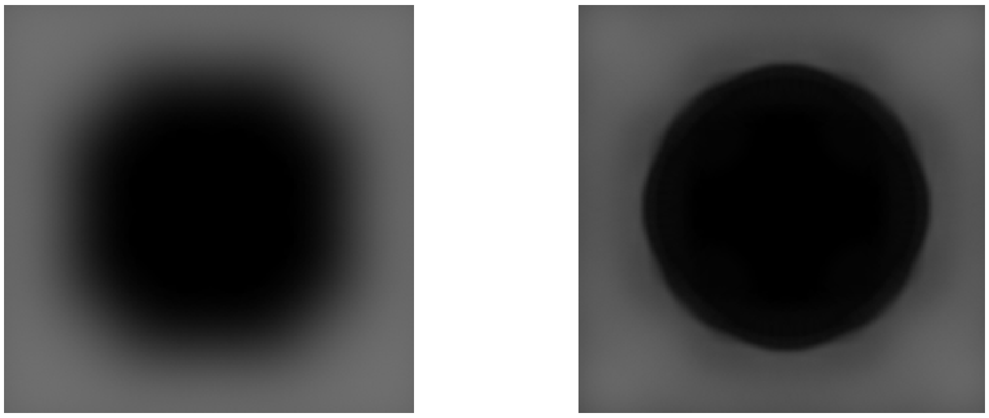


Image of 3-D Hole (Conventional)
Resolution: 12x12x12

Image of 3-D Hole (Enhanced)
Resolution:12x12x12

Fig.9 Comparison between cylindrical hole reconstructed with and without edge preservations.

credited to Louis Noodleman and David Case, Scripps Clinic, La Jolla, California. This *64x64x64* dataset is a description of quantum mechanics calculation of a one-electron orbital of a four-iron, eight-sulfur cluster found in many natural proteins.

We threshold the data set at 0.0017 and 0.5 to create iso-contour surfaces. Rendering the thresholded dataset, we get enhanced views into the data sets. In Figures 11, and 12, the images on the left are rendered with our conventional splatter, the ones on the right are edge enhanced with our method. Here, the necessary edge information is calculated directly from the gradient of the original scalar field and the desired iso-contour value, using a simple Taylor' series expansion. Color images from this data set are shown in Figure 11c.

5. Future Research

Several open problems still occur iwth our method.. Among them, the most prominent is how to stretch or compress the kernels in 3D and then integrate them into view-dependent 2D footprints. This method would be more accurate than manipulating the 2D footprint directly and would allow us to analyze possible errors and determine better compression and stretching functions. Our current stretching/compressing function can still be applied the 3D reconstruction kernels to produce a continuous 3D function with sharp C^1 transitions. Pre-integrating these for all views, edge positions, and edge strengths and building a table of different footprints would still enable efficient rendering on the fly.

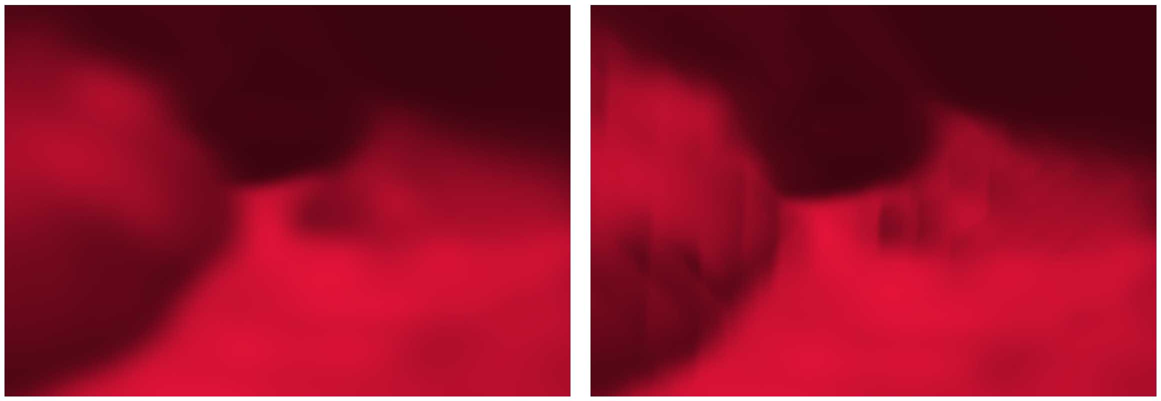
Finally, the small structures popping up into the sinus cavity are usually undersampled, an inherent problem with digitizing any data sets with high frequency components. Adaptive multivariate representations would be beneficial here. One practical case would be mix to different datasets obtained with 8-Tesla and 1.5-Tesla MRI's to resolve areas. The question is how to carry out edge preservation efficiently within this problem context.

Acknowledgements

Special thanks goes to the Ohio Supercomputing Center and Dennis Sessana for aiding in the integration of the sinus data. This data set is derived from the Visible Human data. Also, Dr. Gregory Weit deserves recognition for many valuable conversations and for providing the direction in virtual sinus surgery. Many thanks also to the reviews, whose comments helped to strengthen this paper. This work was supported in part by an Interdisciplinary Seed Grant at The Ohio State University.

References

[Levoy88] Levoy, M., Display of Surfaces from Volume


Fuzzy Renderer

With Edge Preservation

Fig. 10 Comparisons between fuzzy renderer and edge preserved renderer.

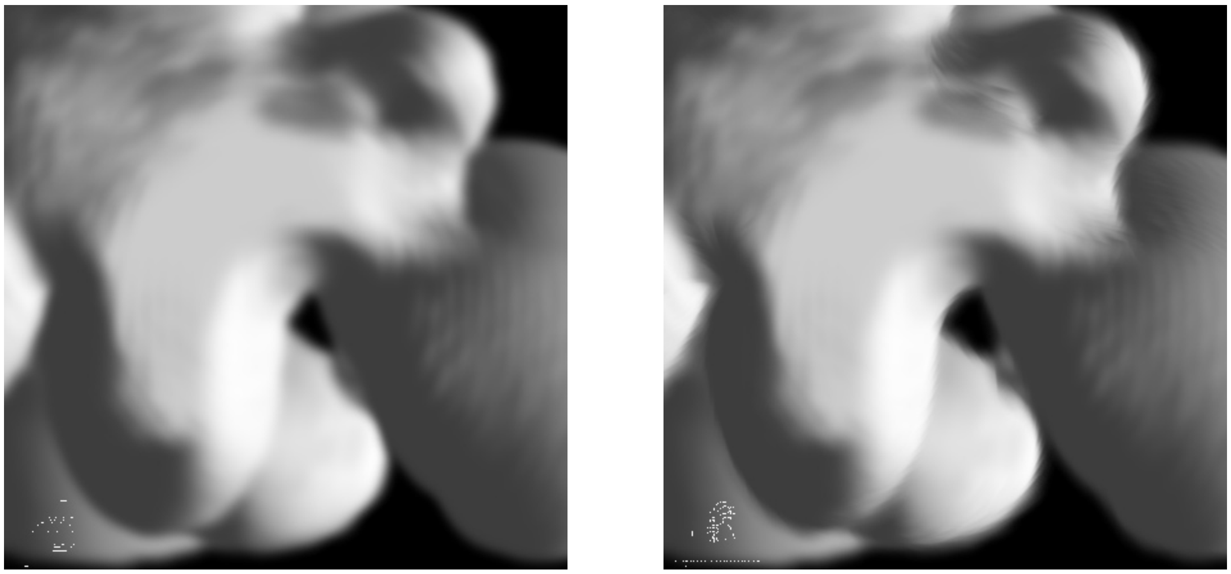


Figure 11. A view into the Hipip data set.

Data, IEEE Computer Graphics and Applications, 8(3):29-37, May 1988.

[Max90] Max, N., Hanrahan, P., Crawfis, R., Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions, Computer Graphics, vol.24, 27-33, November, 1990.

[Crawfis93] Crawfis, R., Max, N., Texture Splats for 3D Scalar and Vector Field Visualization, IEEE Visualization'93 Proceedings, October, 1993, 261-267.

[Guo95] Guo, B., Interval Set: A Volume Rendering Technique Generalizing Isosurface Extraction, IEEE Visualization'95 Proceedings, October, 1995, 3-10.

[Fujishiro95] Fujishiro, I., Maeda, Y., Sato, H., Interval Volume: A Solid Fitting Technique for Volumetric Data Display and Analysis, IEEE Visulization'95 Proceedings, October, 95,151-158.

[Kaufman97] Hong, L., Muraki, S., Kaufman, A., Bartz, D., He, T., Virtual Voyage: Interactive Navigation in the Human Colon, SIGGRAPH'97 Conference Proceedings, August, 1997, 27-34.

[Swan 97] J.E. Swan II, K. Mueller, T. Moller, N. Shareef, R.A. Crawfis, R. Yagel, "An Anti-Aliasing Technique for Splatting" Proceedings of IEEE Conference onVisualization 1997, pp. 197-204, October 1997.

[Mueller 98] K. Mueller, T. Moller, J.E. Swan II, R. Crawfis, N. Shareef, R. Yagel, "Splatting Errors and Antialiasing", to appear in IEEE Transactions on Visualization and Computer Graphics ITVCG 4(2), June 1998.

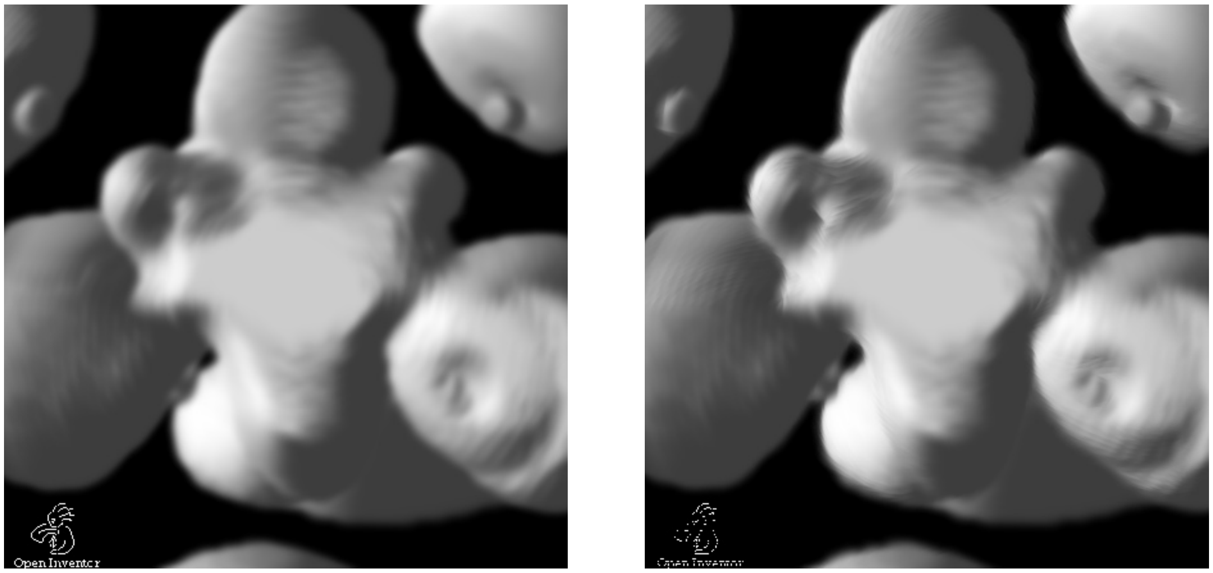


Figure 12. Another view into the Hipip data set.