

Anti-Aliased Volume Extraction

Sarang Lakare and Arie Kaufman[†]

Center for Visual Computing (CVC)
and Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11790-4400

Abstract

We present a technique to extract regions from a volumetric dataset without introducing any aliasing so that the extracted volume can be explored using direct volume rendering techniques. Extracting regions using binary masks generated by contemporary segmentation approaches typically introduces aliasing at the boundary of the extracted regions. This aliasing is especially visible when the dataset is visualized using direct volume rendering. Our algorithm uses the binary mask only to locate the boundary. The main idea of the algorithm is to retain the natural fuzziness at the boundary of a region even after it is extracted. To achieve that, intensities of the boundary voxels are flipped so that they are now representing a fuzzy boundary with the empty region surrounding it, while preserving the boundary position.

Categories and Subject Descriptors (according to ACM CCS): I.4.3 [Image Processing And Computer Vision]: Grayscale Manipulation I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

The techniques that are used to isolate a region of interest in a 3D volumetric dataset are commonly referred to as *segmentation techniques*. Some of these techniques were initially developed for 2D images and later extended to 3D¹⁰, while some were developed specifically for 3D^{13,17}. Some of the simplest and most commonly used segmentation techniques are thresholding and edge-detection¹. Other advanced techniques include classification¹⁸, deformable models^{13,23} and level-sets¹⁴, and many other techniques^{6,15,17}. The abundance of these techniques is due to the fact that there is no one universal technique that works for all problems. Although these techniques are different, they all share the same goal: determine a boundary which separates the region of interest from the rest of the volume. In almost all of the segmentation techniques this boundary is sharp. That is, each voxel in the volume belongs to exactly one region. As a result, the voxels in the volume are divided into two groups: the voxels which belong to the region of interest, and others which belong to the “the rest” of the volume.

Visualization of the segmented region of interest in 3D is traditionally performed using surface-based rendering techniques, which require a polygonal mesh of the region’s surface. This mesh is typically obtained using surface-fitting segmentation techniques^{13,22,23} or by using surface-construction algorithms such as the marching cubes¹¹. We refer to all these as surface extraction techniques. In Figure 1 we show this surface-based approach for the analysis of volumetric data.

Direct volume rendering is a more attractive option for visualization of volumetric data than surface-based rendering. Unlike surface-based rendering, where only the polygon mesh of the region’s surface is rendered, direct volume rendering techniques render the entire volume. This allows the user to examine not only the surface, but also the interior of the region. Peeking inside the region becomes possible by making the region’s surface transparent or translucent.

Visualizing a segmented region of interest using direct volume rendering involves different considerations than surface rendering. Direct volume rendering, by default, renders all the voxels in the volume. When a region is segmented, we have the knowledge of which voxels belong to the region and which do not. This knowledge can be represented by a

[†] {lsarang,ari}@cs.sunysb.edu

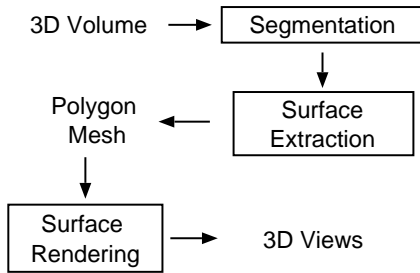


Figure 1: Traditional approach to segmentation and analysis of volumetric data.

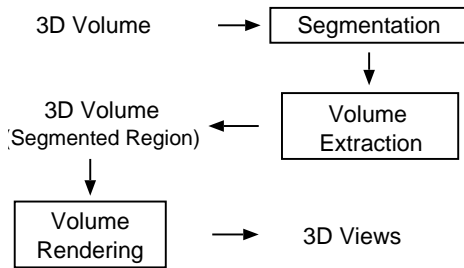
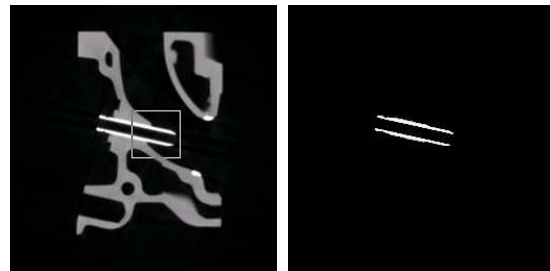


Figure 2: Direct volume rendering approach to segmentation and analysis of volumetric data.

mask volume in which the mask is set only for voxels which belong to the region of interest. Some volume rendering algorithms were designed to include such mask information during rendering⁵. However, many direct volume rendering algorithms and implementations do not take any mask into consideration. We are thus left with two choices: modify the volume rendering algorithm to incorporate mask information, or modify the input volume such that it only contains the voxels which belong to the region. It is easy to see that the second choice is better for hardware¹⁶ implementations and existing off the shelf software implementations. Also, when rendering segmented and unsegmented data together, it is better to change the data and not the volume rendering algorithm.

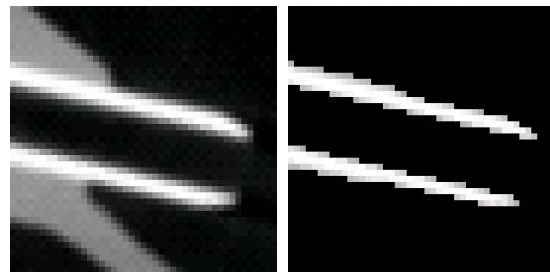
We refer to the process of extracting the voxels which belong to a region of interest as *volume extraction*. This term has been used previously in the literature^{3,22}, but it always referred to the extraction of 3D surfaces from a volume. In this paper, we actually extract voxels from a volume into another 3D volume. Once volume extraction has been performed, visualizing the region using direct volume rendering is straightforward. Any direct volume rendering algorithm or implementation can be used to render this newly extracted volume. In Figure 2 we show our proposed approach to volume exploration using direct volume rendering.

One of our main goals is high quality direct volume ren-



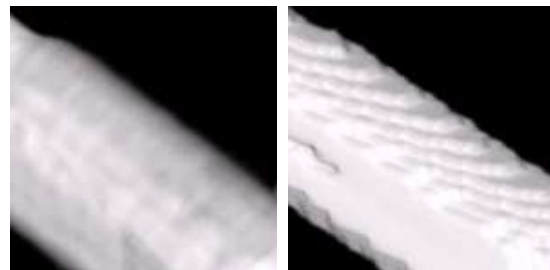
(a) Original CT cross-section

(b) Valve segmented by simple thresholding



(c) Original zoomed

(d) Segmented zoomed



(e) Original rendered

(f) Segmented rendered

Figure 3: Loss of fuzzy boundary can adversely affect direct volume rendering results of an engine CT dataset.

dering of the segmented region. To achieve this, we need to avoid artifacts caused by aliasing. Any aliasing in the data results in aliasing effects in the volume rendered image. One of the locations where aliasing often occurs in extracted volumes is at the boundary of the segmented region. We illustrate this with an example dataset. Figure 3a shows a cross-sectional image of an engine CT dataset. Figure 3b is the result of segmentation by thresholding followed by simple extraction of the valve in which the intensities of voxels that do not belong to the valve are set to the air intensity. Figure 3c and Figure 3d are the zoomed-in images of the se-

lected region in Figure 3a. It can be seen that in the original image (3a and 3c) the object boundaries are fuzzy. That is, there is a smooth intensity transition as you move from one region to another due to the partial volume effect. In the segmented image (3b and 3d) however, the fuzziness is not present. There is a sharp contrast between the intensities of the segmented valve and the surrounding air. This sharp contrast causes the aliasing effects in the rendered images (Figure 3f) as compared with the image in Figure 3e.

In this paper we present our algorithm for volume extraction which retains the fuzziness that is present at the region boundaries. This produces extracted volumes that are free of aliasing. We first review some of the related previous work.

2. Related Work

2.1. Anti-aliased Voxelization

Voxelization is a technique that represents continuous geometric objects by means of 3D voxel volumes. Work in this field has been aimed at reducing the aliasing artifacts that arise when geometric models are converted to voxel volumes²¹. This approach can be used to generate anti-aliased extracted volumes from the polygonal mesh that results from the surface extraction step (Figure 1). However, the result would be a volume in which the internal information of the region is not preserved. Thus, the internals of the segmented region cannot be explored - negating the advantages of direct volume rendering.

2.2. Transfer Functions

Transfer functions are an integral part of direct volume rendering algorithms. Their main purpose is to assign optical properties, such as color and opacity to the voxel data. These assigned properties are in effect for subsequently rendered images. In some cases, transfer functions can be carefully manipulated to make the region of interest visible or emphasized while other regions invisible or de-emphasized. This can be done by assigning high opacity to the voxel intensities that make up the region of interest and low opacity or transparency to other voxel intensities. Thus, it is possible to show only the region of interest from a volume without extracting the region as we propose.

Transfer functions however suffer from at least two major drawbacks. First, the transfer function assignment is position independent and typically only depends on the intensity value of the voxel (and possibly that of a small neighborhood of the voxel). Thus, any property (opacity, color, etc.) assignment on a given intensity value affects all voxels which have that intensity value. This makes it impossible to differentiate voxels which have the same intensity value, but which belong to two separate regions in space. For example, it is impossible to differentiate between two bones in different parts of the body using only a transfer function.

With volume extraction, it is possible to extract just one bone while ignoring the others. In practical terms, transfer functions perform intensity-based segmentation or thresholding. Any region which can be segmented by thresholding alone, is a possible candidate for exploration using transfer functions alone. However, thresholding by itself is not an appropriate segmentation technique in most cases. And, even in the cases where transfer functions are all that are needed, there remain distinct advantages to volume extraction:

- With only the region of interest being rendered, the user is free to adjust the transfer function to interactively explore the region without worrying about interference with other regions, as they are not present in the volume.
- Extraction can reduce the number of significant voxels considerably. Cropping, compression, or a combination of the two can be used to further reduce the size of the volume. This will have two benefits - faster volume rendering and lower storage requirement.

The second drawback of transfer functions is the difficulty in selecting a good transfer function. It can be extremely tedious to come up with the right transfer function for a given region of interest. A poorly chosen transfer function can fool the user by showing features that do not exist or by hiding data that should have been seen. In spite of some recent work on semi-automatic generation of transfer functions^{7,8}, and multi-dimensional transfer functions⁹, working with transfer functions is still a difficult task.

With our approach, it is possible to use specialized segmentation algorithms to mark the region of interest. In cases where thresholding alone is not enough, these segmentation algorithms are bound to give a much better segmentation of the region of interest. Our algorithm can take the output of any such segmentation algorithm and extract the segmented region which can then be subsequently explored using volume rendering. Transfer functions provide a very powerful tool to explore a volume, however we believe that their potential is truly utilized when they are applied only to the voxels that belong to the region of interest.

2.3. Soft Segmentation

Another approach to our goal might be gleaned from the *soft-segmentation* technique²⁰. Soft-segmentation tries to overcome the fundamental problem in image segmentation. For many images there is no way to uniquely and correctly determine the object boundaries. Instead of the usual crisp segmentation where a fixed boundary is derived for a region, soft-segmentation proposes an approach where a voxel can belong to more than one region. This results in voxels around the region boundary being assigned partially to each of the neighboring regions. Although the fundamental ideas are similar to ours to a certain degree, there are two differences. First, this algorithm is a segmentation technique by itself and cannot be used with other segmentation techniques.

Second, the algorithm does not produce an anti-aliased extracted volume. In the absence of an extracted volume, existing direct volume rendering algorithms and implementations will have to be modified to work with the masks produced by soft segmentation. This method is thus not suitable for use with existing direct volume rendering implementations.

3. Intensity Flipping

We now introduce the main technique used in our algorithm, called *intensity flipping*. We use this term because we alter or *flip* the intensity of the voxels as part of the process. Suppose a voxel v has an original intensity δ_v , then the intensity after intensity flipping, δ'_v , is given by,

$$\delta'_v = f(\delta_v) \quad (1)$$

where $f()$ is an intensity flipping function.

Before we describe our algorithms, we clarify some terminology that we use in the following sections. We use the term *region*⁴ to refer to a connected group of voxels with similar characteristics (e.g., air, a bone, a muscle). We use the term *region of interest* (ROI) to describe the group of regions (or a single region) that the user is interested in, and hence needs to be extracted. For example, a bone by itself, or a bone along with a bone-marrow (which is inside the bone and has lower intensity and so comprises yet another *region*) could be a ROI.

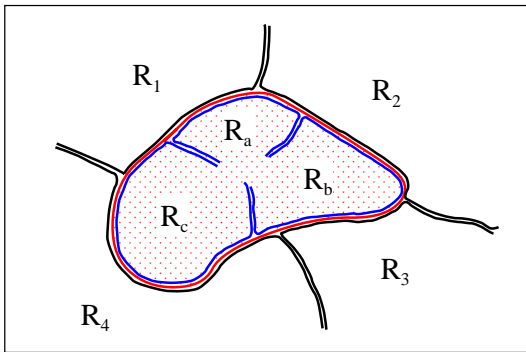


Figure 4: The region of interest to be extracted (dotted) is itself made up of many regions (R_a, R_b, \dots), and surrounded by other regions (R_1, R_2, \dots).

The input to our algorithm is a mask volume marking the voxels which belong to the ROI. This mask volume can be the output of any segmentation algorithm. In Figure 4 we show a typical input dataset. For simplicity, we only show a cross section of a 3D dataset. The dataset has many regions $R_1, R_2, \dots, R_a, R_b, \dots$, with their boundaries (i.e., voxels on the boundary) marked in black or blue. The ROI is shown with a red boundary and is dotted. For the sake of clarity we give alphabetic subscripts to regions which belong to the ROI (R_a, R_b, \dots), and numerical subscripts (R_1, R_2, \dots) to

the unwanted regions. Note that the regions are shown for the sake of explanation only and prior knowledge of these regions is not required by our algorithm. Only the boundary of the ROI (shown in gray) is needed.

The goal of our algorithm is to extract the segmented region from this volume. The segmented region in this case is the dotted region enclosed by the red boundary (regions R_a, R_b, \dots). Along with the extraction of the segmented region, the goal is to maintain the fuzziness at the boundary (area around the gray boundary). Our algorithm can be divided into four steps as follows:

Step 1. Divide the voxels in the dataset into three categories:

1. Voxels that should be subject to intensity flipping.
2. Voxels that should not be altered. That is, intensity flipping is not applied.
3. Voxels whose intensity should be set to air so that they become transparent in subsequent volume rendering.

Voxels in category 1 are the voxels that form the boundary between the ROI and its surrounding regions. These are the voxels whose intensity values lie somewhere between those of the regions that surround them which result in the fuzziness at the boundaries (Figure 3c). To find these voxels, we erode the region of interest once and mark the eroded voxels. We then dilate the original region of interest once and mark the dilated voxels. In Figure 5, we show the dilated and eroded region by pink color. These marked voxels, along with the voxels which formed the original boundary (red) now belong to category 1. This is the region between and including the pink lines in Figure 5. For both erosion and dilation, we use a 18-neighbor (in 3D) versions of the 8-neighbor (in 2D) erosion and dilation algorithms¹⁹, respectively.

Voxels in category 2 are all voxels that belong to the ROI minus the voxels which now belong to category 1. These are the voxels in the region marked by dots. All the remaining voxels of the volume belong to category 3.

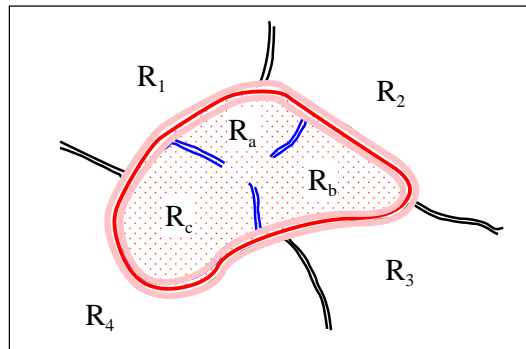


Figure 5: At the end of step 1, the boundary voxels are marked.

Our algorithm next calculates the new intensity for each of the voxels that belong to category 1. The intensity of these voxels has to be changed because some of their neighboring regions (the ones that do not belong to the ROI) will eventually be replaced by voxels with the intensity of air.

The next two steps of the algorithm are repeated for each voxel p which belongs to category 1.

Step 2. With p as the center, a region-grow is performed such that the region includes voxels which are nearest to the central voxel and belong to category 2 or category 3. The region-grow continues until it includes at least λ voxels from category 2 and at least λ voxels which belong to category 3. The λ voxels are used to calculate the average intensity of the respective regions. For practical purposes, λ can be a small number. For our experiments, we chose a value of 5. Let us assume that at any instance of the region-grow process, λ_1 voxels belong to category 2 and λ_2 voxels belong to category 3. We stop the region grow when the following condition is satisfied:

$$\lambda_1 \geq \lambda \quad \& \quad \lambda_2 \geq \lambda \quad (2)$$

The idea behind the region-grow is to find the voxels that are nearest to voxel p . Some of these belong to category 2 and some to category 3. These neighboring voxels represent the regions that surround p , and are responsible for influencing the intensity value at p . We therefore calculate the average intensity of each of these sets of neighboring voxels. We denote the average intensity of the λ_1 voxels by δ_1 , and that of the λ_2 voxels by δ_2 :

$$\delta_1 = \frac{\sum_{k=1}^{\lambda_1} \delta_k}{\lambda_1}, \quad (3)$$

$$\delta_2 = \frac{\sum_{k=1}^{\lambda_2} \delta_k}{\lambda_2} \quad (4)$$

Step 3. We now have the approximate average intensities of the two regions surrounding voxel p . Since p lies between the two regions, its voxel intensity lies between the average intensity of the two regions. Our goal now is to find the new intensity at voxel p , when one of the regions around it is removed.

In Figure 6 we show an intensity profile along a hypothetical ray that would pass through voxel p , moving along the approximate direction of the gradient, but traversing through voxel centers. Before intensity flipping, the ray profile would be as shown by the long-dashed curve with blue points which are the voxels that the rays traverses. As we go across voxel p (dashed vertical line), we see that the intensity gradually changes from δ_1 to δ_2 . This intensity profile basically depicts how the intensity changes at the boundary of the region (solid red vertical line), as we move across the boundary.

Our goal is to maintain the fuzziness at the boundary. For

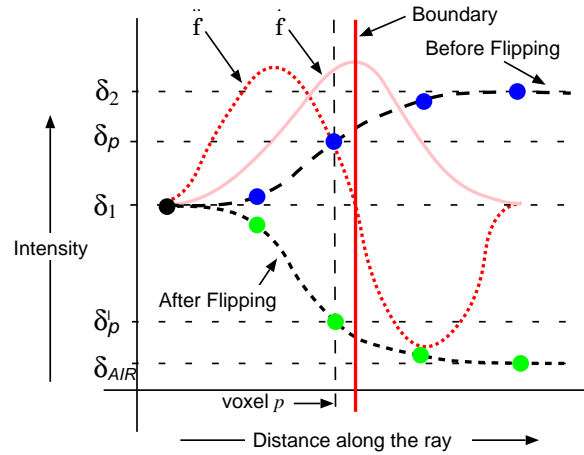


Figure 6: Intensity flipping for voxel p .

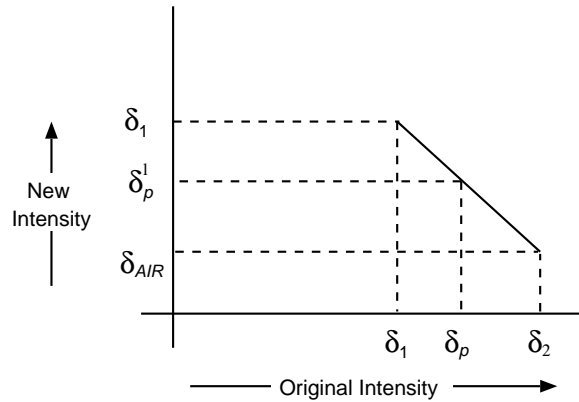


Figure 7: Intensity flipping graph for voxel p .

this, we try to retain the boundary location as we remove one of the regions attached to the boundary. Because of band-limiting, the boundary position is spread over a range of positions. To find the exact location of the boundary, we take clues from two edge detectors commonly used in image processing, the Canny ² and the Marr-Hildreth ¹². These edge detectors define the boundary by the maximum value of the first derivative (gradient) and by the zero-crossing of the second derivative, respectively. We show the first derivative \hat{f} by the light pink line, and the second derivative $\hat{\hat{f}}$ by the red-dotted line. The exact boundary location itself is shown by the red vertical line.

With this information on the boundary position, we find new intensity values for the voxels in category 1 so that the boundary position remains unchanged. To do this, we use the following intensity flipping equation:

$$\delta'_p = \delta_{AIR} + \frac{\delta_2 - \delta_p}{\delta_2 - \delta_1} (\delta_1 - \delta_{AIR}) \quad (5)$$

Where, δ'_p is the new intensity for the voxel p .

The intensity flipping curve corresponding to Equation 5 is as shown in Figure 7. When this intensity flipping is applied to all the voxels along the ray, the intensity profile of our hypothetical ray will change. The new intensity profile is shown by the short-dashed curve with green points in Figure 6. The green points are the intensities of the same voxels along the ray but now with different values than before. As a result of this flipping, the first derivative changes, but the maximum of the first derivative (the boundary) remains at the same position. Similarly, the second derivative zero-crossing remains unchanged. This nice property gives us an unchanged boundary position, even though one of the regions around the boundary has changed.

As mentioned earlier, steps 2 and 3 are repeated for all voxels which belong to category 1.

Step 4. In the final step, all the voxels that belong to category 3 are converted to air. That is, the intensity of these voxels is set to the air intensity. Figure 8 shows the state of our dataset after this step. It can be seen that all the unwanted regions are now air regions.

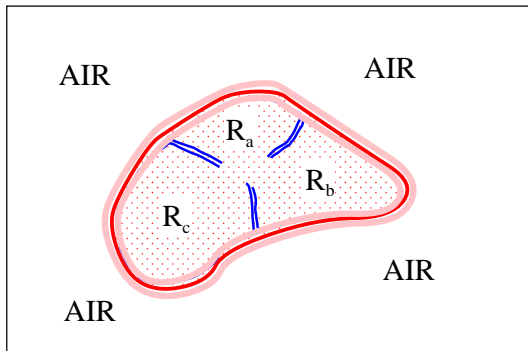


Figure 8: At the end of step 4, all category 3 voxels are converted to air.

4. Implementation

Although our algorithm is almost straight forward to implement, we devised a technique to accelerate it. The region-grow for each voxel in step 2 is implemented as an 18-connected region-grow⁴. Since the region-grow is to be repeated for each category 1 voxel, it is computationally very expensive. To reduce the complexity of this step, we perform a small amount of pre-processing. We pre-compute the voxel offsets (with respect to the start voxel) for all the voxels that are traversed in a typical 18-connected region grow. To do

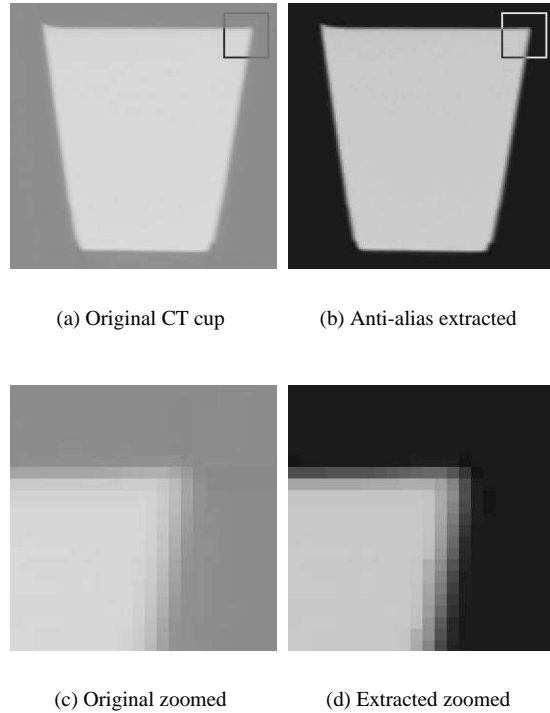


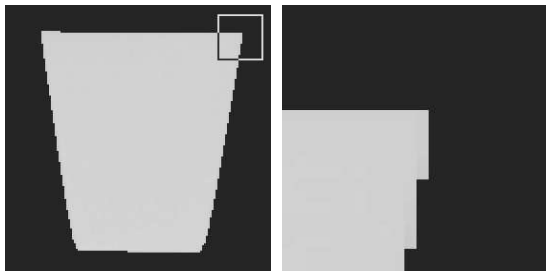
Figure 9: A cup is extracted using our algorithm.

this pre-computation, we perform a dummy region grow and save the offsets. Then, for each region grow in step 2, we simply access the voxels using the offsets we already know. Since we use only offsets, there is no need for a mask volume during the region grow. This accelerates the step and has no memory overhead.

5. Experimental Results

Figure 9a shows an image of a cup dataset which is a CT scan of a high-density material surrounded by a relatively low-density material. We first segment the cup using simple thresholding. We then provide the result of this segmentation as input to our intensity flipping algorithm. The results are shown in Figure 9. Figure 9b shows the extracted cup from the dataset using our method. Figures 9c and 9d show a zoomed view of the area marked in Figures 9a and 9b, respectively. From Figure 9c it can be seen that the boundary between the cup and its surrounding is fuzzy. Figure 9d shows that the extracted cup using our method retains this fuzziness at its boundary.

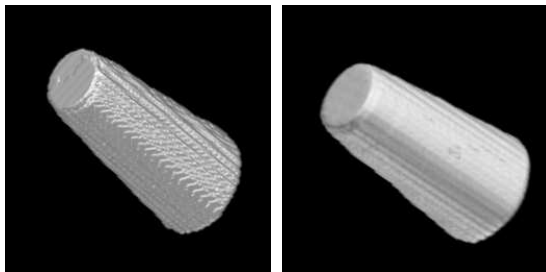
To show the effectiveness of our algorithm, we compare it with common segmentation. We first apply simple intensity-threshold based segmentation to the cup dataset and extract the cup (Figure 10). On comparing Figure 9d and Figure 10b



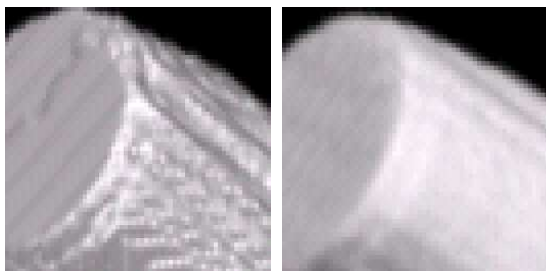
(a) Simple thresholding extraction (b) Zoomed in view

Figure 10: Extraction of cup using simple thresholding.

it can be observed that the results of common segmentation are sharp. The fuzziness at the boundary is lost resulting in aliasing effects. To observe the effects on direct volume rendering, we volume rendered the two extracted cup data (Figures 9b and 10b) using the same parameters and the

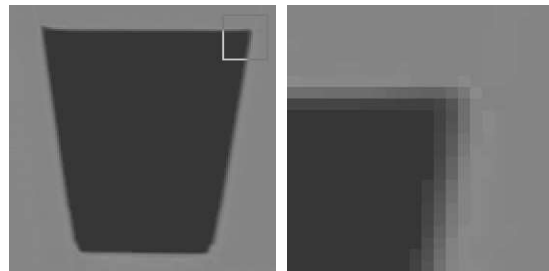


(a) Aliasing effects of simple thresholding extraction (b) No aliasing after our anti-aliased extraction



(c) Simple thresholding extraction - zoomed (d) Anti-aliased extraction - zoomed

Figure 11: Direct volume rendering of the extracted cup.

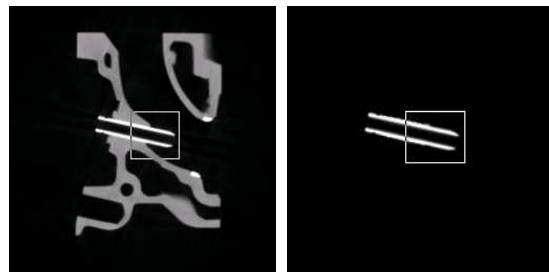


(a) Reverse extraction (b) Zoomed-in view

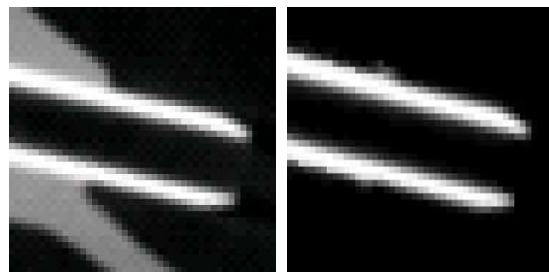
Figure 12: Using global intensity flipping to extract the cup.

same algorithm (Figure 11). The aliasing artifacts of common segmentation are clearly visible in Figure 11a, whereas Figure 11b shows a smooth surface devoid of aliasing effects.

Just as we extracted the cup from the dataset, we can extract its surroundings and get rid of the cup. The results of

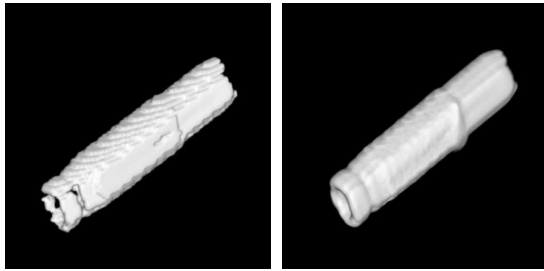


(a) Original CT cross-section (b) Anti-alias extracted valve



(c) Original zoomed (d) Extracted zoomed

Figure 13: Extraction of the valve from the CT engine dataset using our anti-aliased extraction.



(a) Simple thresholding extraction (b) Our anti-aliased extraction

Figure 14: Direct volume rendering of the extracted valve.

our algorithm are shown in Figure 12. Figure 12b shows a zoomed image of the area marked in Figure 12a. It can be seen that the fuzziness at the boundary is still retained and now it is between air and the low-density surrounding material.

Figure 13a presents some results for the engine CT dataset. Our goal is to extract the valve from the dataset.



Figure 15: Direct volume rendering of an anti-aliased extracted bone of the right foot of the Visible Human CT dataset.

The segmentation algorithm we use prior to extraction is a simple intensity-based thresholding. Figure 13b shows the valve extracted from the engine using our method. The part of the valve marked in Figure 13b is zoomed in and shown in Figure 13d. The corresponding original data is shown in Figure 13c. It can be seen that the fuzziness along the entire boundary of the valve is preserved. After the remaining engine is removed, the resultant valve is as if it was CT scanned separately.

We compare results of our algorithm with those from simple thresholding-based extraction. Figure 3d shows the result after applying simple extraction. It can be seen that the fuzziness on the boundaries is lost and the entire boundary shows aliasing effects. These aliasing artifacts are clearly visible on volume rendering of the valve extracted using simple extraction (Figure 14a). The same valve, when extracted using our algorithm (Figure 13d) shows no aliasing on volume rendering (Figure 14b).

We now apply our algorithm to a very complex dataset. We use the right foot of the Visible Human CT dataset and extract the entire bone along with the interior bone-marrow. Before applying our algorithm, we perform segmentation to mark the outer boundary of the bone. This becomes our region of interest. Then, we apply our algorithm and show the direct volume rendering of the extracted bone in Figure 15. We zoom in to show that there is no aliasing even when we view the bone very closely (Figure 16).

To see the bone marrow hidden inside the bone, we simply alter the transfer function to make the bone material transparent, the bone marrow opaque, and give it a red color. The



Figure 16: Zoomed in portion of the extracted bone shown in Figure 15



Figure 17: Bone marrow clearly visible. (Also see Color Plate Figure 1.)



Figure 18: Bone marrow hidden behind muscle tissues. (Also see Color Plate Figure 2.)

results can be seen in Figure 17. If we try to see the bone marrow in the original Visible Human dataset by altering the transfer function, we get the image shown in Figure 18 with poor results, because the bone marrow has the same intensity distribution as the surrounding soft tissue/muscles. Thus, it is impossible to make the surrounding tissues transparent by simply using a transfer function.

All of the above results were generated on an AMD Athlon 1GHz Linux PC with 1GB of RAM. Table 1 summarizes the results for the three example datasets. We found the time for volume extraction is largely dependent on the boundary area (category 1 voxels) which are listed in the last column of Table 1.

Table 1: Experimental results with three datasets.

Dataset	Size	Extraction time	Category 1 voxels
Cup	$512 \times 512 \times 346$	14.6 sec	428K
Engine	$256 \times 256 \times 110$	1.8 sec	22K
Foot	$140 \times 270 \times 190$	14.2 sec	489K

6. Conclusions

In this paper we presented an algorithm to extract regions from volumetric datasets without introducing any aliasing artifacts. The resultant volumes are ready for high quality direct volume rendering and can be visualized using any direct volume rendering software or hardware. We believe that this algorithm should be an integral part of a volume exploration system using direct volume rendering (Figure 2).

We have shown the importance of retaining the fuzziness at region boundaries. Volumes obtained from scanning techniques such as CT and MRI, show fuzziness or partial volume effect at region boundaries. Using our algorithm, this natural fuzziness can be maintained even when one of the regions at the boundary is removed. A unique aspect of our algorithm is that the fuzziness is not added artificially. Rather, the fuzziness that already exists in the input volume is retained. The intensity of the voxels that make up the fuzziness in the volume is flipped such that the new intensities blend properly with the new intensities of neighboring regions. This also has a nice feature that the boundary position, which can be defined by the zero-crossing of the second derivative, remains the same after intensity flipping. As a result, we are able to maintain the boundaries even when the regions around it change.

Another feature of our algorithm is that it can be used in conjunction with any segmentation algorithm. Since the input to our algorithm is the volume which is to be segmented and a binary mask volume with the results of the segmentation, any segmentation algorithm can be used to generate the mask.

From the experimental results we observe that transfer functions are more useful for volume exploration when they are applied to an extracted region rather than the entire volume. Figure 17 shows that it was possible to see the bone marrow inside the extracted bones using simple manipulation of the transfer function. The same is impossible when applied to the entire volume (Figure 18). This shows that volume exploration using direct volume rendering is much more effective in conjunction with our volume extraction algorithm.

As future work, we would like to make our algorithm more robust. At present, a successful anti-aliasing at the

boundary depends on the accuracy of the segmented region boundary. If the segmented region boundary is not in the partial volume region where the natural fuzziness occurs, then the resultant boundary will not be fuzzy either. We would also plan to address issues related to noise.

Acknowledgements

This work has been supported by grants from NIH #CA82402, ONR #N000110034, and Viatronix Inc. The engine dataset is courtesy of GE, the cups dataset was provided by the University Hospital of the State University of New York at Stony Brook, and the Visible Human dataset is courtesy of the Visible Human project. The authors wish to thank Manjushree Lakare, Klaus Mueller, Jerome Liang and Suzanne Yoakum-Stover for their help, encouragement and discussions.

References

1. D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, N.J., 1982.
2. J. F. Canny. A Computational Approach to Edge Detection. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.
3. A. Doi, S. Fujiwara, K. Matsuda, and M. Kameda. 3D Volume Extraction and Mesh Generation Using Energy Minimization Techniques. In *1st International Symposium on 3D Data Processing Visualization and Transmission*, 2002.
4. E. R. Dougherty and C. R. Giardina. *Matrix Structured Image Processing*, chapter Topological Operations, pages 140–148. Prentice-Hall, 1987.
5. R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *Proc. of SIGGRAPH '88*, pages 65–74, 1988.
6. R. M. Haralick and L. G. Shapiro. Image Segmentation Techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, January 1985.
7. T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In *Proc. of Visualization '96*, pages 227–234. IEEE Computer Society Press, 1996.
8. G. Kindlmann and J. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In *Symposium on Volume Visualization*, pages 79–86, 1998.
9. J. Kniss, G. Kindlmann, and C. Hansen. Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets. In *Proc. of Visualization '01*, pages 255–262, 2001.
10. H. K. Liu. Two and Three Dimensional Boundary Detection. *Computer Graphics and Image Processing*, 6:123–134, April 1977.
11. W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM Computer Graphics*, 21(24):163–169, July 1987.
12. D. Marr and E. Hildreth. Theory of Edge Detection. In *Proc. of Royal Soc. London*, volume B. 207, pages 187–217, 1980.
13. J. V. Miller, D. E. Breen, W. E. Lorensen, R. M. O'Bara, and M. J. Wozny. Geometrically Deformed Models: A Method for Extracting Closed Geometric Models from Volume Data. In *Proc. of SIGGRAPH '91*, pages 217–226, 1991.
14. S. Osher and J. Sethian. Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.
15. N. R. Pal and S. K. Pal. A Review on Image Segmentation Techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.
16. H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro Real-Time Ray-Casting System. In *Proc. of SIGGRAPH '99*, pages 251–260, Aug 1999.
17. D. L. Pham, C. Xu, and J. L. Prince. A Survey of Current Methods in Medical Image Segmentation. Technical report, Johns Hopkins University, Baltimore, January 1998.
18. D. L. Pham, C. Xu, and J. L. Prince. Current Methods in Medical Image Segmentation. *Annual Review of Biomedical Engineering*, 2, 2000.
19. William K. Pratt. *Digital Image Processing*, chapter Morphological Image Processing, pages 449–490. A Wiley-Interscience, second edition, 1991.
20. D. Prewer and L. J. Kitchen. Soft Image Segmentation by Weighted Linked Pyramid. *Pattern Recognition Letters*, 22(2):123–132, 2001.
21. M. Sramek and A. E. Kaufman. Alias-Free Voxelization of Geometric Objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):251–267, July 1999.
22. I. Takanashi, S. Muraki, A. Doi, and A. Kaufman. 3D Active Net - 3D Volume Extraction. *Journal of the Institute of Image Information and Television Engineers*, 51(12):2097–2106, 1997.
23. D. Terzopoulos and K. Fleischer. Deformable Models. *The Visual Computer*, 4(6):306–331, December 1988.