Real Time Isosurface Browsing

Caleb Lyness Edwin Blake

Collaborative Visual Computing Laboratory
University of Cape Town

Abstract

As volumetric datasets get larger, exploring the data sets becomes more difficult and tedious. Two approaches have previously been used to solve this problem: presentation of an abstraction of the data and acceleration of extraction and rendering of the data. We present a user centered approach which decouples the volume visualisation into two modes. Selection of the mode is done based on the user's actions. The first mode uses traditional isosurface rendering and extraction techniques and is applied when the user knows the isovalue of interest. The second mode uses a novel view dependent, sampling based isosurface rendering and extraction technique, which allows interactive browsing of the isosurfaces.

CR Categories: I.3.3 [Computer Graphics]: Pictures/Image Generation; I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism

Keywords: User centered approach, decoupling, isosurface browsing, volumetric data exploration

1 Introduction

Volumetric data sets, which are common in many fields, are often visualised using isosurfaces. Isosurfaces are typically generated using a technique such as Marching Cubes [11]. The triangle mesh output from these techniques can be displayed using standard polygonal rendering. The mesh is an intermediate format, which can be manipulated and re-rendered from a different view with little cost.

While Marching Cubes is very effective, it does, have a few problems. Some of the problems have been resolved, for example the ambiguities in choosing triangles to represent voxel intersections [12, 18]. Other problems, such as the selection of isosurfaces, have yet to be solved satisfactorily. If the desired isosurface's isovalue is known in advance, the algorithm can be run once with an acceptable delay between value selection and visible output. However, when the isovalue is unknown and the viewer wishes to explore the data, this delay makes interaction difficult, especially as the volume grows in size. Several techniques have been proposed and reported in the literature to speed up isosurface extraction with varying degrees of success.

This sketch presents a user centered approach to isosurface viewing. The isosurface visualisation is separated into two stages based on the viewer's actions. One stage deals with the viewing of a selected isosurface, while the other the exploration of all possible isosurfaces from the current viewpoint. A novel view dependant, sampling based preprocess and isosurface extraction technique is presented, which allows interactive volume exploration when used in this setting.

Section 2 gives a review of the relevant literature, Section 3 discusses the decoupling scheme, Section 4 covers the real time isosurface implementation. Section 5 gives some timings and results. And Section 6 ties up with a conclusion and possible future work.

2 Background

The need to support the user in exploring volumetric data has been recognised by [4]. They proposed an abstraction based on Hyper Reeb graphs. Their system uses the topological changes of the isosurface within the volume to select isovalues of interest. The abstraction seems very effective. However, the authors do not report results for volumes consisting of complex topologies, which may cause the abstractions usefulness to break down.

Prior to the above abstraction, a more direct approach of accelerating the extraction and rendering of isosurfaces was attempted. Rendering acceleration was achieved by reorganising the triangle mesh to form strips [7]. Further, decimation techniques where used to reduce the number of rendered triangles, while remaining within an acceptable error bound [15, 14].

The extraction process is accelerated by reducing visits to voxels not containing the rendered isosurface. As there are usually far more empty voxels than voxels which contribute to the isosurface, the speed increase is often significant. The accelerated extraction algorithms may be catagorised as belonging to one of two groups: seed based and range based.

Seed based algorithms make use of a list of isovalues and a set of starting voxels. From these starting points the isosurface is constructed via propagation to adjacent cells. Seed based techniques typically have a preprocess of $O(n)$, where n is the number of voxels [16, 1, 8].

Range based methods can be considered part of the space partition group of algorithms. The first reported technique was based on octrees [19]. Later techniques more suited to unstructured grids where developed [5]. These techniques group the voxel data into buckets. A similar approach [6] sorted the buckets on the maximum and minimum isovalues and took advantage of isosurface coherence to accelerate the rendering of isosurfaces whose isovalues where close. These algorithms where reported to have a run time of $O(n)$ and require a preprocess of similar order. More recently [10, 2] kd-trees were used to reduce the complexity to $O(\sqrt{n} + k)$. k is the number of contributing voxels.

Adaptive reconstruction of isosurfaces using a modified octree structure (average pyramids) was used in [17] to provide real time exploration. The typical cracking and sampling problems associated with adaptive reconstruction were resolved, providing a view independent solution. The algorithm described provides real time exploration by trading off quality for speed and requires the user to specify a point of interest, around which higher detail rendering is done.

Recent developments in high end consumer hardware, specifically Nvidia's GeForce family of graphics cards, has renewed interest in accelerating isosurface rendering. Using clever techniques and the new features on these graphics cards has allowed interactive rendering of isosurface and direct volume rendering [13, 3]. These techniques are firmly tied to the underlying hardware and architecture.

Rather than relying heavily on hardware, we have adopted a decoupling approach. The idea of decoupling a problem can be found in volume rendering literature [9]. In his thesis work Lacroute suggested the use of two different data structures for volume rendering. Each data structure being optimal for the different task — classification and rendering of the voxels.

3 The decoupling scheme

The proposed decoupling follows a natural separation of user actions when exploring volumetric data. Two distinct modes of operation become clear when observing a subject using standard isosurface rendering techniques. Generally a subject will start with an arbitrary isovalue. If the generated isosurface is of interest they will examine it from several different views, otherwise another isovalue is selected based on the observed isosurface. This process continues until an interesting isosurface is found.

The decoupling allows two separate algorithms to be used. Figure 1 illustrates the decoupling and how user actions change the modes and associated algorithms. The mode switching occurs as transparently as possible.The switching occurs based on what actions the user applies to the interface — dragging the trackbar versus trying to rotate the object.

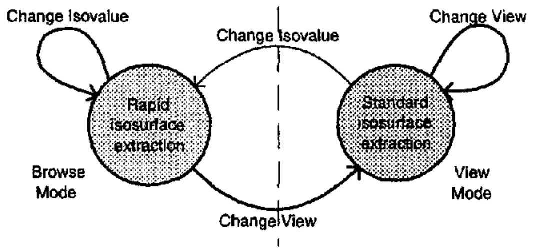


Figure 1: Isosurface rendering is decoupled into two states. User actions control which state system is in and thus which algorithm is run.

The current system uses an implementation of the marching cubes algorithm for displaying a selected isosurface from an arbitrary view point (View Mode in Figure 1). Note that any of the acceleration techniques mentioned in the Background section can be applied. For isosurface browsing (Browse Mode in Figure 1), a modified ray casting algorithm is used. This technique is presented in the next section.

4 Isosurface browsing

Rapid isosurface extraction allows the viewer to find isosurfaces of interest interactively. The rapid extraction is achieved by reducing the amount of data and reordering it with respect to the isovalues.

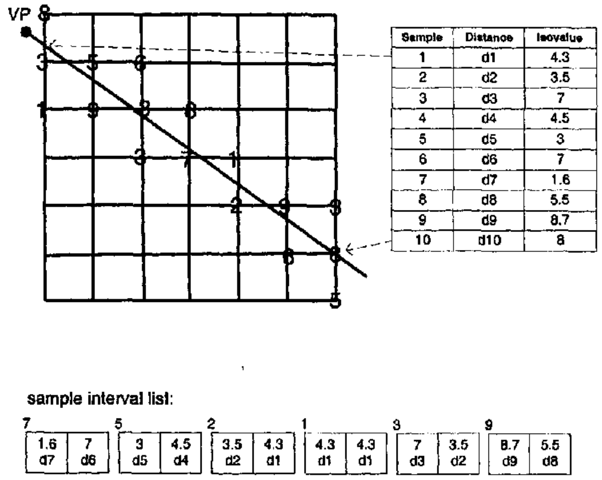
The algorithm is based on ray casting and runs in two stages. The first stage runs as a preprocess for a selected view. It involves calculating and storing the first occurrence of an isovalue and its distance along a ray from the viewpoint. The second stage is run every time the user selects a new isovalue and is responsible for generating the mesh which approximates the isosurface. A more detailed description of the stages follows.

4.1 Stage 1: Preprocessing

Isosurface construction techniques assume volumetric data is a sampled representation of a continuous field. Calculating the first occurrence of an isovalue along a ray can be described more accurately as finding the first interval in which the isovalue lies. Each interval is described as a pair of samples in the data.

The preprocess samples the volume by firing a set of rays from the viewpoint. As a ray is fired into the volume it is intersected with the volume slices in the x,y and z planes. A sample is extracted for each intersection. Every pair of samples in the volume form an interval. As the ray moves away from the eye, there is greater and greater chance that a new interval contains no new iosvalues and can thus be discarded. The discarding of the redundant information reduces the amount of data considerably.

As new intervals are constructed they are added into a list by insert sorting, based on the first unique isovalue they contain — see Figure 2. As the list reorders the samples, the distance from the viewpoint is calculated and stored. However to avoid the expense of calculating the full distance, only the squared distance is calculated. Once the final, reduced set of intervals has been found, the true distance is calculated along with an approximation of the normal at the sample points.



Sample	Distance	Isovalue
1	d1	4.3
2	d2	3.5
3	d3	7
4	d4	4.5
5	d5	3
6	d6	7
7	d7	1.6
8	d8	5.5
9	d9	8.7
10	d10	8

7		5		2		1		3		9	
1.6 d7	7 d6	3 d5	4.5 d4	3.5 d2	4.3 d1	4.3 d1	4.3 d1	7 d3	3.5 d2	8.7 d9	5.5 d8

Figure 2: During the preprocess stage, rays are fired into the volume. The first occurrence of the isovalues and their distances are calculated. The table to the right indicates the sample at each grid-ray intersection, along with the associated distance (d_i) and isovalues, the final sample interval set is shown beneath. Each block represents the sample interval, with the top, left value representing the unique, sorted isovalue and the the right value the sample partner.

4.2 Stage 2: Extracting and rendering isosurfaces

With the set of sorted sample intervals associated with each ray fired into the volume, extracting an isosurface becomes a problem of searching the set of intervals. Once the interval which contains the user specified isovalue is found, the distance and normal are calculated using linear interpolation.

Triangle meshes are constructed from the information extracted from each ray. The approximated, triangle mesh isosurface is then rendered using OpenGL. The left image in Figure 3 show the extracted points used to construct a triangle mesh.

5 Results

Some initial timings for the algorithms running on a Pentium II, 400Mhz with 256Mb of memory and a Riva TNT2 video card are given in Figure 4.

Volume size	128x128x93
Marching Cubes	3,515 ms
Mesh Rendering	4,907 ms
Preprocess	22,111 ms
Extraction	290 ms
Mesh Rendering	350 ms

Figure 4: The average number of clock ticks spent running marching cubes, the preprocess and the extraction stages and the time to render the constructed mesh.

The short extraction time indicates that real time browsing of the volume is possible. For the 128x128x93 volume 3.4 extractions per second are possible and we can construct the mesh at rate of 2.8 per second. This means that on average 1.5 new isosurface can be displayed per seconds. If one tried to use marching cubes for real time isosurface browsing, you would be able to display 0.11 isosurfaces per second, in other words it takes 9 seconds to generate 1 isosurface.

The reduced rendering times associated with the isosurface browsing technique can be attributed to the reduced number of triangles and vertices rendered, a side effect of the hidden face removal.

6 Conclusions and possible future work

We have presented an alternative technique to support a user exploring volumetric data. The visualisation process is split into two modes, based on observed user actions — a browsing mode and a viewing mode. In the viewing mode traditional isosurface extraction and rendering techniques are used, while in browsing mode a new techniques, which was presented, offers interactive isosurface extraction and display rates of 1.5 isosurfaces per second. These rates make it possible for the user to browse the data interactively.

While the initial results indicate that rapid browsing is possible, the time spent in the current preprocess is significant. Future work will be to reduce the run time of the preprocess. Two avenues of work exist:

1. Implementation of an alternate ray casting engine, which makes greater use of occlusion information to prevent expensive calculations, and
2. A reduction in the number of rays fired. Currently the system samples the volume with rays at half the size of the projected voxel size. However in large volumes this is often unnecessary as the volume is either too large to be viewed entirely or is too distant to discern the samples. In such situations screen space sampling would be very effective.

Further, to improve user interaction, modifications to support a progressive ray casting algorithm and extraction step would allow the user to interact with coarsely sampled data, while the in between samples are being completed.

The isosurface browsing algorithm is limited to the current viewpoint, and the preprocess needs to be rerun when a new viewpoint is selected. When used in the decoupled scheme presented, the effects of this limitation are reduced.

7 Acknowledgments

We would like to thank African Medical Imaging (AMI) Inc. for their generous support, as well as the MRC/UCT Medical Imaging Research Unit under which this project falls.. Further we would like to thank Shaun Nirenstein for providing much food for thought.

References

[1] C. L. Bajaj, V. Pascucci, and D. Schikore. Fast Isocontouring for Improved Interactivity. In *ACM Symp. Volume Visualization '96*, 1996.

[2] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno. Speeding Up Isosurface Extraction Using Interval Trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):158–170, 1997.

[3] K. Engel, M. Kraus, and T. Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. Accepted for Siggraph/Eurographics Workshop on Graphics Hardware 2001, 2001.

[4] I. Fujishiro, Y. Takeshima, and et al. Volumetric Data Mining Using 3D Field Topology Analysis. *IEEE Computer Graphics and Applications: Visualization*, 20(5):46–51, September/October 2000.

[5] R. Gallagher. Span filter: an optimization scheme for volume visualization of large finite element models. In *Proceedings of Visualization 1991*, pages 68–75, San Diego, Los Alamitos, CA, October 22-25 1991. IEEE Computer Society Press.

[6] M. Giles and R. Haimes. Advanced interactive visualization for CFD. *Computing Systems in Engineering*, 1(10):51–62, 1990.

[7] C. T. Howie and E. H. Blake. The Mesh Propagation Algorithm for Isosurface Construction. *Computer Graphics Forum*, 13(3):65–74, 1994.

[8] T. Itoh and K. Koyamada. Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):319–327, 1995.

[9] P. Lacroute and M. Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. In *Proceedings of SIGGRAPH '94*, pages 451–458, July 1994.

[10] Y. Livnat, H. Shen, and C. Johnson. A near optimal isosurface extraction algorithm for structure and unstructured grids. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84.

[11] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, 1987.

[12] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993.

[13] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In *SIGGRAPH/EUROGRAPHICS Workshop On Graphics Hardware*, pages 109–118, August 21-22 2000.

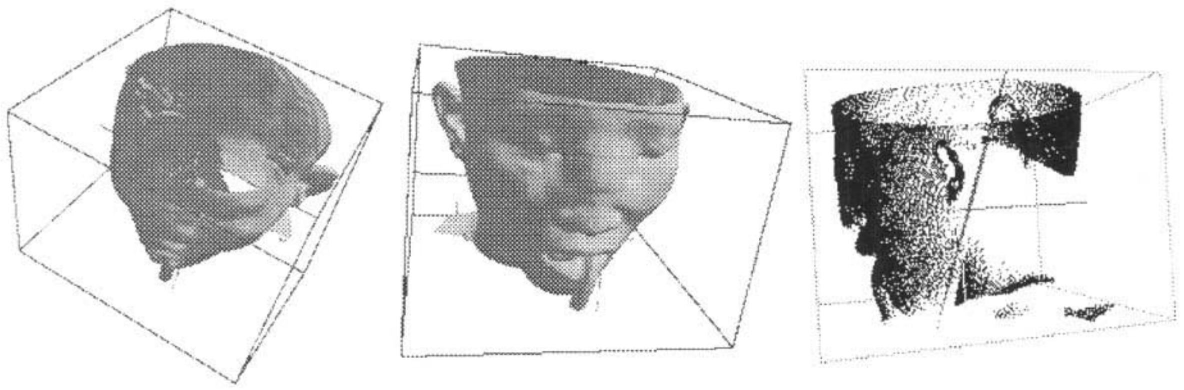


Figure 3: Left: The output from marching cubes for a given isovalue. Middle: Another view of the same isosurface, in which a large portion of the mesh has been hidden by the front part. Right: The sampled point cloud extracted for construction of the triangle mesh during Stage 2 using the same isovalue. The image has been rotated to illustrate how the hidden surfaces have been occluded.

[14] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics.* Prentice Hall, 1996.

[15] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, 1992.

[16] M. van Kreveld, R. van Oostrum, and C. Bajaj. Contour trees and small seed sets for isosurface traversal. In *SIGGRAPH 85*, 13th Annual ACM Symposium on Computational Geometry, pages 212–219, 1997.

[17] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15(2):100–111, 1999. ISSN 0178-2789.

[18] J. Wilhelms and A. V. Gelder. Topological considerations in isosurface generation, extended abstract. *Computer Graphics, Special Issue on San Diego Workshop on Volume Visualization*, 24(5):79–86, 1990.

[19] J. Wilhelms and A. V. Gelder. Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.