Fast Interactive Region of Interest Selection for Volume Visualization*

Dominik Sibbing and Leif Kobbelt

Lehrstuhl für Informatik 8, RWTH Aachen, 52056 Aachen
Email: {sibbing,kobbelt}@informatik.rwth-aachen.de

Abstract. We describe a new method to support the segmentation of a volumetric MRI- or CT-dataset such that only the components selected by the user are displayed by a volume renderer for visual inspection. The goal is to combine the advantages of direct volume rendering (high efficiency and semi-transparent display of internal structures) and indirect volume rendering (well defined surface geometry and topology). Our approach is based on a re-labeling of the input volume's set of isosurfaces which allows the user to peel off the outer layers and to distinguish unconnected voxel components which happen to have the same voxel values. For memory and time efficiency, isosurfaces are never generated explicitly. Instead a second voxel grid is computed which stores a discretization of the new isosurface labels. Hence the masking of unwanted regions as well as the direct volume rendering of the desired regions of interest (ROI) can be implemented on the GPU which enables interactive frame rates even while the user changes the selection of the ROI.

1 Introduction

Generating 2D images from volumetric datasets like MRI or CT scans is an important visualization task in medicine. These images help to understand anatomical structures, to make a diagnoses or to observe the healing process. They can be used for planning an operation or for surgical training. With modern graphics cards one can generate a 3D view of this data at interactive frame rates [1]. Unfortunately volumetric images often contain a lot of information, which one would not like to see all at once. Using only a transfer function which maps intensity values of the 3D image to colors and opacity values is not sufficient, because in most of the scans different tissues show similar intensity values. This often leads to the occlusion of the interesting components, for example the occlusion of the human brain by the skull. Therefore one has to separate somehow the interesting data for a certain application from the unimportant data [2]. We present a method which segments the volumetric image in an automated way using a set of isosurfaces. An isosurface is always closed and no isosurface can penetrate another isosurface. Applying a transfer function to the volumetric dataset does not change the set of isosurfaces (only their individual labels) and therefore one can consider this set of isosurfaces as an invariant geometric representation of the dataset.

* This work is supported by the DFG (IRTG 1328).

The isosurfaces have a hierarchical structure which is induced by their inclusion relation. With this information the user can easily select regions like the human brain by simply peeling of the layers outside the ROI.

2 State of the art and advances by the presented contribution

The common techniques for volume rendering are divided in direct and indirect methods. Direct methods render the voxels of the volumetric data and try to mask out those voxels that do not belong to the ROI. This method is preferable because it shows the original data and can therefore provide all the information of the original 3D image. One direct rendering technique is the registration of the input volume to a standard Talairach space and the usage of a template to mask the uninteresting voxels, see e.g. [3]. Unfortunately this is limited to healthy human brains and to a fixed set of templates. Another direct rendering technique uses a transfer function which maps intensity values to colors and opacities [1]. As said before this can lead to occlusions so the ROI is not visible anymore.
A marching cubes algorithm [4] which can be used for indirect volume rendering computes a polygonal representation of an isosurface, that can be rendered. Unfortunately the extraction of an isosurface surrounding the ROI often is a trial and error method. A more advanced technique is it to use active contours [5] which evolve until they approximate the boundary of the ROI. However, to control the stopping criterion is a hard task, see also [6], so active contours will not always produce a good approximation. Similar to that is the usage of deformable models for generating a certain ROI. But they are also limited to a fixed set of templates and cannot be used for arbitrary volumetric datasets.
Our method is a direct volume rendering method in the sense that we render the *original volumetric data* in interactive framerates. But for the segmentation and visual improvement of the volumetric dataset we use techniques from indirect volume rendering, based on the extraction of isosurfaces. By combining both techniques we obtain the best of both worlds, i.e. the structure and geometry control from indirect methods and the flexibility and performance of direct methods.

3 Methods

The idea of our method is to first extract a large set of isosurfaces with the well known marching cubes algorithm [4], which provide the geometric information contained in the *original volumetric data* independent from a transfer function. After separating these isosurfaces in connected components we relabel these components based on their inclusion relation, and store them in a tree data structure. In this tree component A is parent of B iff A encloses B. This will later allow the user to easily mask out the irrelevant components, by selecting nodes from the tree (Fig. 1) similar to the navigation in a hierarchical file system. In our experiments we extracted 64 isosurfaces for various gray levels between the minimum and maximum voxel value. As we will show, these isosurfaces can be stored in a very memory efficient way.
The new labels are assigned by a depth first traversal in the tree and are used to generate an additional volumetric dataset where voxels are labeled by the smallest isosurface in

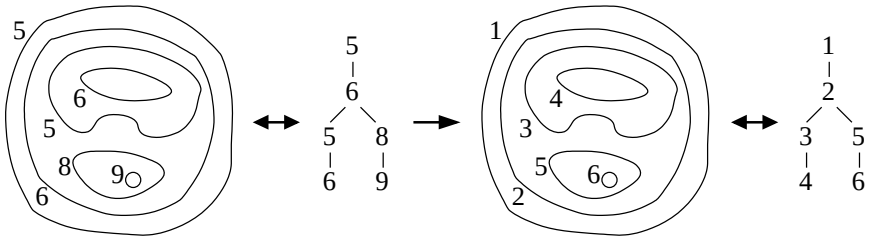

Fig. 1. Calculation of new labels, by traversing an thereby uniquely relabeling the nodes of a tree in a depth first order.

which they are contained. Both the *original volumetric data* and the new *labeled volumetric data* are stored in the memory of the graphics card. For displaying the tree we designed an interface similar to a file browsing system, which is capable to visualize large structures in an easy way. One Click on one of the nodes toggles the visibility of the whole subtree. During interactive display, we only send the information which labels are visible to the graphics card, and decide for each fragment if it is visible or not by looking up its label in the *labeled volumetric data*. After that we can render the fragment according to the value stored in the *original volumetric data* using a simple transfer function. This allows for interactive frame rates.
The setup for the algorithm is as follows. We call an edge of the voxel grid x-edge if the adjacent grid point differ by one in the x-coordinate.

For building the tree and calculating the *labeled volumetric data* we need two important procedures. The *'IsIn'*-Test which decides whether a surface is inside another surface and a scanline algorithms which line by line assigns the new labels for the voxels.

3.1 *IsIn*-Test and Scanline Algorithm

To test whether a surface A lies inside a surface B we first locate the point $p \in A$ with the highest x-coordinate. Note that p always lies on an x-edge. From p we shoot a ray in x-direction and count the number N of intersections with surface B. We observe:

$$A \text{ inside } B \Leftrightarrow N \mathit{mod}\, 2 = 1 \tag{1}$$

Due to the linear interpolation between the voxel samples, one (and only one) intersection with B happens for each x-edge which connects two voxel with values $V^- \leq B < V^+$.

For setting the values in the *labeled volumetric data* we use a scanline algorithm, which traverses rays along the positive x-direction. Everytime we enter a surface s, we activate s. If we leave s we deactivate it. Then a voxel will get the label of the last surface we marked so far.

To put it more precisely, we process one of the $n_y \cdot n_z$ scanlines by first initializing a stack with label 0 on top. For every voxel $p = (x, y, z)$ we encounter, we first set the label to the value stored on top of the stack. We update the stack by looking at all the intersections on the next x-edge right of p. Intersecting a surface which has the same label as stored on top of the stack will remove the element on top of the stack. In the other case we push the label of the intersected surface on the stack. Note that each x-edge can intersect only once with a specific surface, because the marching cube algorithm linearly and hence monotonically interpolates between adjacent grid points to generate a vertex of the surface.

3.2 Efficient storage of the isosurfaces

As said before isosurfaces provide a lot of information, like vertex positions in 3D and the connectivity of the mesh. For our purpose we don't need all of the information. All we have to ensure is that the *'IsIn'*-Test and the scanline algorithm work. We observe that in both procedures we shoot a ray along x-edges of the grid. Because the marching cube algorithm only generates vertices on the edges of the grid we only can enter or leave a surface by passing a vertex on a x-edge. So we only store those vertices in form of an offset in x-direction w.r.t the left grid point, which extremely reduce the memory consumption. Both procedures can be executed very efficient if we store for every grid point a list of vertices which lie on the right x-edge and a list of their corresponding surfaces.

4 Results

We generated all results with an AMD64 with 2.2GHz, 2GB RAM and a NVidia GeForce 7800 GTX graphics card. The images 2a-d were generated from a MRI scan with a resolution of 256x256x170. The extraction of 64 isosurfaces and the calculation of the tree and the new labels took less than 15 Min. Fig. 2c and d show our method in comparison to a direct (Fig. 2a) and an indirect (Fig. 2b) method.

5 Discussion

The advantage of this method is that it works on differend kind of volume images. Nevertheless these images should not contain too much noise, because then it is difficult to extract smooth isosurfaces. In our case we use a bilateral filter [7] to denoise the intensity distribution of the *original volumetric data*. The selection of the interesting regions is very intuitive, because the result of each selection can be seen immediately. If the user wants to see a deeper layer, he simply traverses the tree downwards. But for a large set of isosurfaces the tree can get rather large. Therefore we allow the user to decimate this tree, by discarding surfaces which have approximately the same volume as the parent surface, i.e. we can combine the nodes $\{1,2\}$, $\{3,4\}$ and $\{5,6\}$ in the example of Fig. 1.

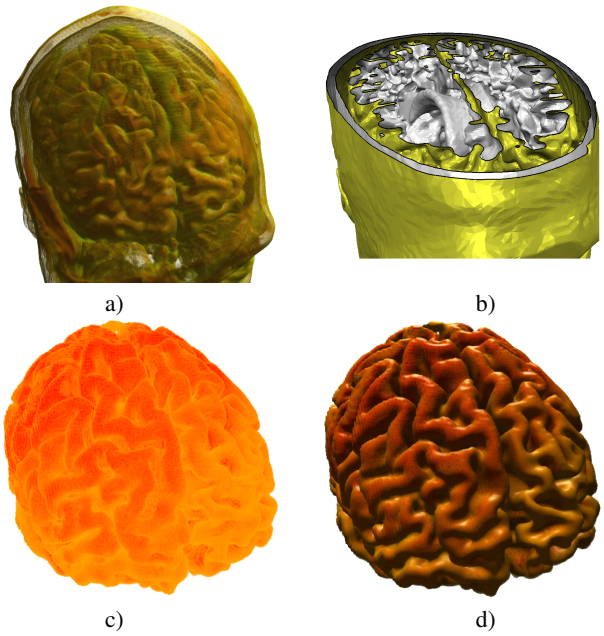

Fig. 2. a) Direct Volume Rendering: On the one hand direct volume rendering does not distinguish between voxels with identical labels, so occlusions may occur. But on the other hand this method achieves high performance. b) Indirect Volume Rendering: Isosurfaces which further can be separated into connected components (otherwise occlusion may occur) are a good representation of the geometric information contained in the *original volumetric data*. Unfortunately finding the right labels is a trial and error method with low interactivity. c) Combining the best of both methods lead to a flexible and efficient direct volume renderer which is capable to distinguish every connected voxel component. d) Integrating lighting and shadows further improve the 3D impression of the image.

References

1. Engel K, Hadwiger M, Kniss JM, et al. Real-time volume graphics. In: ACM SIGGRAPH 2004 Course Notes; 2004. p. 29.
2. Wang L, Zhao Y, Mueller K, et al. The Magic Volume Lens: An Interactive Focus+Context Technique for Volume Rendering. In: IEEE Visualization; 2005. p. 47.
3. Collins DL, Neelin P, Peters TM, et al. Automatic 3D intersubject registration of MR volumetric data in standardized Talairach space. J Comput Assist Tomogr 1994;18(2):192–205.
4. Lorenson W, Cline H. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics 1987;21(4):163–169.
5. Davatzikos C, Prince J. An active contour model for mapping the cortex. IEEE Trans Med Imaging 1995;14:65–80.
6. Bischoff S, Kobbelt L. Sub-voxel topology control for level set surfaces. In: Eurographics 2003 Proceedings; 2003. p. 273–280.
7. Tomasi C, Manduchi R. Bilateral Filtering for Gray and Color Images. In: ICCV; 1998. p. 839–846.