Volume Rendering using Seed Filling Acceleration: Supporting Cut Planes by Fast Re-seeding

Jarkko Oikarinen Lasse Jyrkinen Rami Hietala

Abstract—

Seed filling in view lattice is a method to accelerate volume rendering. Our previously published results on using seed filling had the limitation that they do not support cut planes. Cut planes are one of the main advantages of using volume rendering over surface rendering. In this paper we present a fast re-seeding process that may be performed for each frame. The re-seeding process does a connected component search on the cut plane within the volume data set. The amount of processing done by a connected component search algorithm is proportional to N^2, assuming the data set size is N^3. Our measurements were performed with orthogonal cut planes on two medical MRI data sets. Our research is work-in-progress and more thorough evaluation is still required.

Index Terms: volume visualization, three-dimensional (3D) imaging, ray casting, volume rendering, medical applications, seed filling, connected component search.

I. Introduction

Interactive volumetric rendering requires immense amounts of computing power. New algorithms, aiming to optimize the processing required for volumetric rendering, have been developed simultanously with faster computer hardware. There are two main paths that have been taken to visualize volumetric data.

The first path is to create a surface, usually a triangulated patch surface, and then just render the polygons using graphics accelerator boards. The best known of these algorithms is the Marching Cubes algorithm proposed by Lorensen and Cline in 1987 [1]. The main advantage of this approach is that a multitude of hardware manufacturers provide powerful graphics accelerators that can render even millions of polygons per second.

The second path is to render the volumetric data directly; hence it is often called direct volume rendering. The main advantages of this approach include the possibility to do some modifications to the displayed objects without recreating the triangles as is the case with Marching Cubes. These modifications include displaying isosurfaces and having cut planes to cut the volumetric object to be displayed.

One of acceleration techniques which renders the volumetric data directly uses seed filling in view lattice to calculate the rendered image [2] [3]. Our previous implementations and descriptions of this acceleration technique had the drawback of not providing a way to support interactively cut volumes in the volume rendered view. Overcoming this problem will broaden the field of possible applications for the seed filling acceleration technique.

In this paper we propose a solution to this problem. The principles of our solution apply to any cuts performed on the volume data set, but in this paper we mainly concentrate in cutting with a plane that is perpendicular to one of the main axes of the volume data set. Our solution is to add a re-seeding algorithm that is executed every time the cut plane is moved in relationship with the volume data. This re-seeding algorithm adds a new seed point for each 4-connected component on the cut plane. This will guarantee that there will be at least one seed for each 6-connected component in the data set.

Connected component labeling is a well-known technique in computer vision and image processing. In this paper we apply a preprocessing technique, simplified from a 2-dimensional connected component labeling algorithm [4], that may be performed to find the new seed points.

J. Oikarinen is with the Departments of Diagnostic Radiology and Neurosurgery, Oulu University Hospital, PO Box 22, 90221 OULU, FINLAND. http://www.iki.fi/jto

L. Jyrkinen is with the Departments of Electrical Engineering and Diagnostic Radiology, University of Oulu, PO Box 22, 90221 OULU, FINLAND

R. Hietala is with the Department of Diagnostic Radiology, Oulu University Hospital, PO Box 22, 90221 OULU, FINLAND

II. Seed Filling

Seed filling in view lattice accelerates volumetric rendering by avoiding the processing of a portion of those sample points that do not contribute to the final image. In this paper those sample points that we try to avoid are referred as *empty sample points*, which refers to the fact that they are fully transparent. Partially transparent or fully opaque sample points are referred as *non-empty sample points*.

An example of a data set consisting of empty and non-empty voxels is presented in Figure 1. Initially the algorithm requires one seed voxel for each 6-connected component.

If the view lattice is dense enough and samples the volume lattice at such a high resolution that the connectivity of the 6-connected components is maintained, the volume rendering may be performed by a 3- or $2\frac{1}{2}$-dimensional seed filling algorithm operating in view lattice. This is illustrated in Figures 2 and 3 and has been proposed in [2] and [6] for 3- and $2\frac{1}{2}$-dimensional seed fillings, respectively.

The seed filling algorithm requires a set of seeds; at least one seed for each 6-connected component in the data set. These seeds are fed to the seed filling algorithm, which is then guaranteed to access all those sample points that contribute to the final image.

The 3-dimensional seed filling algorithm presented in [2] processes only those sample points that are presented as hollow spheres in Figure 2. The accesses to the other sample points are completely avoided.

Maximum intensity projection requires that all non-empty sample points are accessed, and hence the 3-

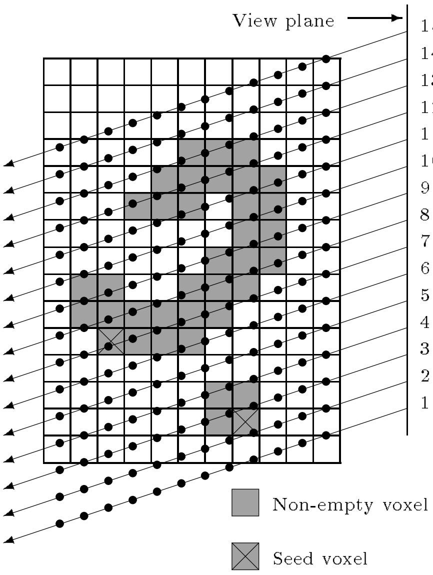

Fig. 1. The non-empty voxels in the volume lattice can be divided into 6-connected components, each indicated by a seed voxel

dimensional seed fill algorithm used in that case will need to perform more memory references [3].

When the non-empty sample points are completely opaque (binary opacity), it is sufficient to seed fill only the front surface of the object; this is what we refer as $2\frac{1}{2}$-dimensional seed filling [6]. This is illustrated in Figure 3.

If the volume lattice is cut with cut planes, some seed points may fall outside the data set and some connected components may lose their connectivity. Hence the requirement of one seed for each 6-connected component is not met after the volume is cut with a cut plane. This is illustrated in Figure 4.

A re-seeding process is needed to solve this problem. The re-seeding process adds new seed points so that the requirement of having at least one seed point for each connected component can be met even after the volume is cut by the half-space. The next section presents a technique, based on well-known two-dimensional connected component labeling algorithm, that may be used to perform this re-seeding process for cut planes perpendicular to any of the axes x, y or z.

III. Supporting Cut Planes by a Re-Seeding Process

In this section we draw our attention to those voxels in the data set that have a neighbor cut out by the cut plane. If the cut plane is perpendicular to any of the main axes (x, y or z), these voxels are all the voxels in the first image slice that is not removed by the cut plane (Figure 5). Performing a new connected component search in this plane and adding a new seed for each 4-connected component found in this plane will guarantee that there is at least one seed voxel for each 6-connected component in the whole data set.

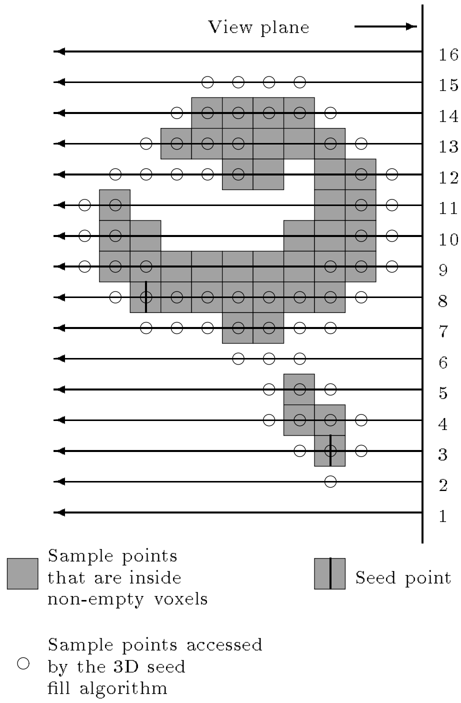

Fig. 2. An illustration of the accesses to the sample points made by the 3D seed fill acceleration technique

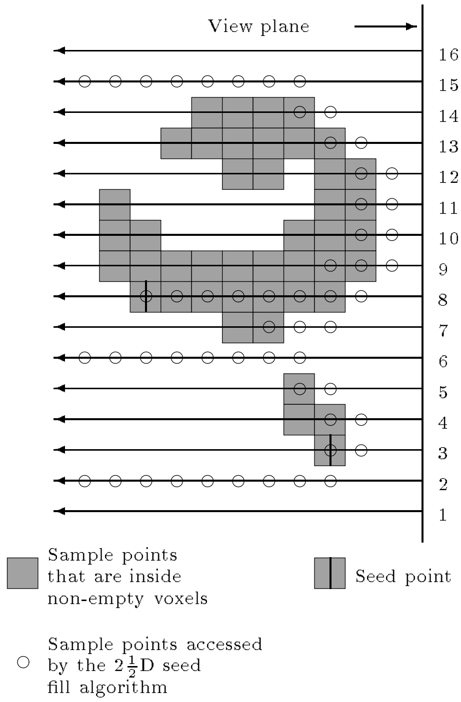

Fig. 3. An illustration of the accesses to the sample points made by the $2\frac{1}{2}$D seed fill acceleration technique

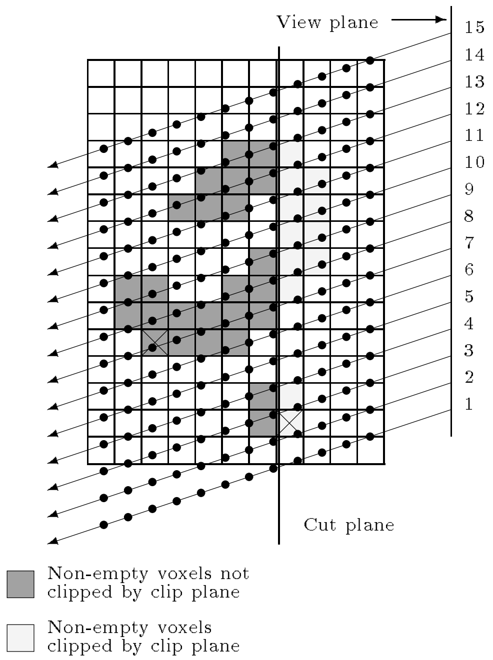

Fig. 4. Connected components lose their connectivity and seed points fall outside the data set if the view lattice is cut with cut planes

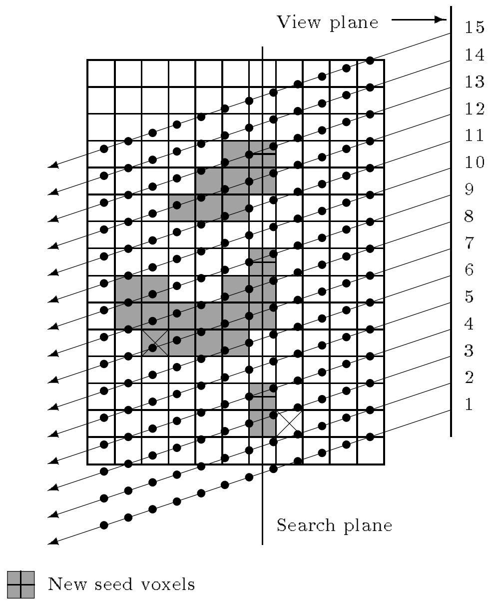

Fig. 5. A connected component search on the search plane will guarantee a seed voxel for each 6-connected component in the data set

Algorithms for fast connected component search have been proposed since 1966 [5]. The main improvements over the original algorithm have mostly been in reducing the amount of memory and memory references taken by the link tables. A connected component search algorithm goes through a binary image and assigns a separate label for each connected component. The connected component search algorithms found in the literature perform two passes over the data set; first pass is used to find out the components and the second pass is used to assign the labels to non-empty voxels. A link table is built simultaneously during the first pass. The purpose of the link table is to store the information on how components with different labels may be linked to each other. The second pass goes through the voxels and labels the voxels using the information in the link table. In our case we only need to find out a seed for each connected component and hence the second pass is unnecessary. Our algorithm adds new seeds to the seed table during the first pass; every time the last reference to a label is freed, and the label has no other links that are still active, a new seed is added. Our implementation is based on the algorithm presented in [4].

When the size of the data set is N^3 and the cut plane is perpendicular to one of the main axes, the re-seeding process requires N^2 accesses to the data set.

In a more general case the cut planes may be oblique and expressed in the view coordinate system or in the volume coordinate system. If the cut plane is fixed in the volume coordinate system, the re-seeding process is needed only when the cut plane is moved. If, on the other hand, the cut plane if fixed in the view coordinate system, the re-seeding process has to be performed every time the viewing direction is changed, because each change in the view orientation changes the relationship between the cut plane and the volume coordinate system.

IV. Experiments

In our preliminary experiments we used two MR data sets of human head to evaluate the effectiveness of the connected component search that is done for each cut plane perpendicular to x, y or z axis. The dimensions of both data sets were 256x256x120. All the measurements were done on a 200 MHz Pentium Pro workstation with 64MB of main memory, running Linux 2.0.30 operating system. The times taken by the algorithm were obtained with the GNU profiler.

V. Results

Two images with different cut planes are shown in figures 6 and 7. The times required to perform the re-seeding for these two images were 6.7 ms and 3.6 ms for the figures 6 and 7, respectively. The number of voxels accessed during the connected component search are 256 x 256 and 120 x 256, for the two cases, respectively.

VI. Conclusion

Our preliminary research indicates that the time taken by the seed calculation will slow down the algorithm only slightly on our workstation. Assuming the rendering speed of 9 to 10 frames per second that we have previously measured in [6] for the $2\frac{1}{2}$-dimensional seed filling algorithm for a 256x256x128 MR data set, the effect of re-seeding would be clearly less than 1 frame per second, even if the cut plane is changed for every frame. The number of seeds, as long as it stays within hundreds, does not have a significant effect on rendering, as shown in [3]. There may also

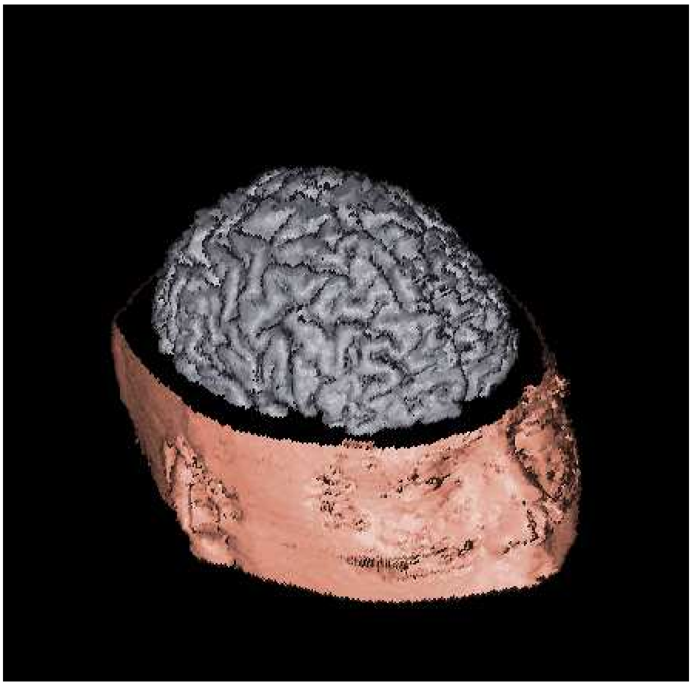

Fig. 6. The first MR data set cut by an axial cut plane which reveals the brain surface

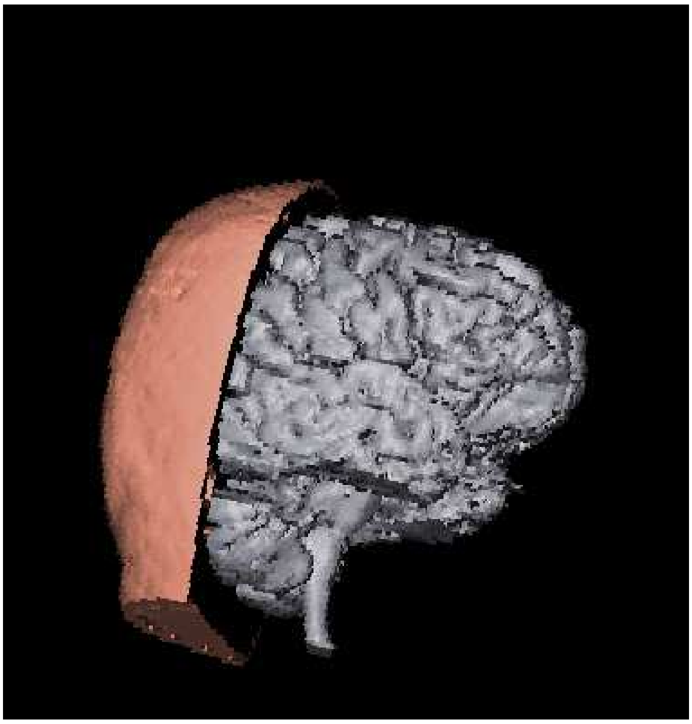

Fig. 7. The second MR data set cut by a coronal cut plane which reveals the brain surface

be several cut planes, each applied to different object.

The results that we present in this paper show that the seed filling acceleration may be made to support cut planes, because allowing cut planes does not significantly sacrifice the rendering speed of the seed filling algorithm. However, our current implementation of the cutting algorithm is not efficient and requires scanning through the complete data set. This slows down the rendering unacceptably. We are working on improving this portion of the algorithm and hence the presented work is work-in-progress.

One drawback of the seed filling acceleration that we have not yet addressed is the ability to support only parallel projections. Perspective provides much better depth perception, but is computationally more expensive than parallel projection with almost all known volume rendering accelerations.

Our future plans include the use of the seed filling acceleration technique as one of the visualization techniques for the interventional MRI system that is being developed in the Oulu University Hospital and University of Oulu. The project we plan to use the results in is a project partly funded by the European Union; "Intraoperative Real-time Visualisation and Instrument Tracking in MRI", IRVIT-ESPRIT P27230.

References

[1] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *SIGGRAPH 87 Conference Proceedings*. ACM SIGGRAPH, Aug. 1987, vol. 21 of *Annual Conference Series*, pp. 163–169.

[2] Jarkko Oikarinen, "Using 3-dimensional seed filling in view space to accelerate volume rendering," in *IEEE Visualization'97 Late Breaking Hot Topics*, Amitabh Varshney and David S. Ebert, Eds. IEEE Computer Society and ACM/SIGGRAPH, Oct. 1997, pp. 17–20.

[3] Jarkko Oikarinen and Lasse Jyrkinen, "Maximum intensity projection by 3-dimensional seed filling in view lattice," in *EDU- and COMPUGRAPHICS'97 Proceedings*, Harold P. Santo, Ed. GRASP, Dec. 1997, pp. 173–183.

[4] Michael B. Dillencourt, Hannan Samet, and Markku Tamminen, "A general approach to connected-component labeling for arbitrary image representations," *Journal of the Association for Computing Machinery*, vol. 39, no. 2, pp. 253–280, 1992, ISSN 0004-5411.

[5] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital signal processing," *Journal of the Association for Computing Machinery*, vol. 13, no. 4, pp. 471–494, 1966.

[6] J. Oikarinen, "Using 2- and $2\frac{1}{2}$-dimensional seed filling in view lattice to accelerate volumetric rendering," *IEEE Transactions on Visualization and Computer Graphics*, Submitted.