IEEE/EG International Symposium on Volume Graphics (2010)
R. Westermann and G. Kindlmann (Editors)

A new sampling scheme for slice based volume rendering

J. Krüger[†]

Interactive Visualization and Data Analysis Group, DFKI, MMCI
SCI Institute, University Of Utah

Abstract

In this paper we present a novel approach to generate proxy geometry for slice based volume rendering. The basic idea is derived from the behavior of a ray-caster and is a simple extension of the well known 2D object-aligned texture stack based technique. From this our novel scheme inherits the advantage that it enables hardware-based volume rendering for devices that do not support 3D textures. On these devices previous object-aligned 2D texture based approaches suffered from disturbing view angle dependent stack-switching artifacts which are avoided by our novel method. Our approach also shows benefits compared to the widely used view aligned slicing algorithm as it avoids jagged boundary artifacts and increases performance.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques I.3.2 [Computing Methodologies]: Computer Graphics—Graphics Systems

1. Introduction and Related Work

For several decades, volume rendering has become an indispensable part of the toolbox of various scientific domains, most importantly in medical applications but also in natural sciences, engineering and arts.

Based on the observation that a single core CPU is not powerful enough to achieve interactive frame rates for reasonable data sets and screen resolutions, considerable effort has been spent on developing volume rendering methods for parallel architectures—in particular for graphics processing units (GPUs).

For graphics hardware capable of processing 3D textures Cullip and Neumann [CN94] presented the idea of using object- and image-space planes as proxy geometry to sample the volume. Their idea was extended by Cabral et al. [CCF94] with the particular focus on medical imaging applications. To allow for volume rendering on low-end hardware incapable of 3D texturing, Rezk-Salama et al. [RSEB*00] extended Cullip and Neumann's [CN94] as well as Lacroute et al.'s [LL94] ideas to develop techniques for 2D object aligned stacks. In their approach the volume is stored in three stacks slicing the volume in the X, Y, and Z directions. During runtime the stack most perpendicular to the viewing direction is chosen and rendered back to front. This means that during the rotation of the data at some point the system will switch from one stack to the other, causing noticeable changes in color and opacity of the volume rendered image.

As all of the slicing approaches exhibit a nonuniform sampling pattern for perspective projections, La Mar et al. [LMHJ99] proposed to use spherical shells to resemble the uniform step size sampling of a ray-caster. This approach, however, tends to create large amounts of geometry for the shells, severely limiting the performance of GPU volume rendering. To overcome this limitation Krüger and Westermann [KW03] as well as Röttger et al. [RGW*03] presented GPU based ray-casting systems when GPUs became sufficiently programmable. Their approach was later extended by Stegmaier et al. [SSKE05] to take full advantage of novel GPU features such as unlimited-length shaders to avoid multiple-passes.

With GPUs becoming ever more powerful, general research in the last couple of years has shifted from these basic sampling techniques to higher level problems, such as how to effectively guide and focus the user's attention to im-

[†] jens.krueger@dfki.de

DOI: 10.2312/VG/VG10/001-004

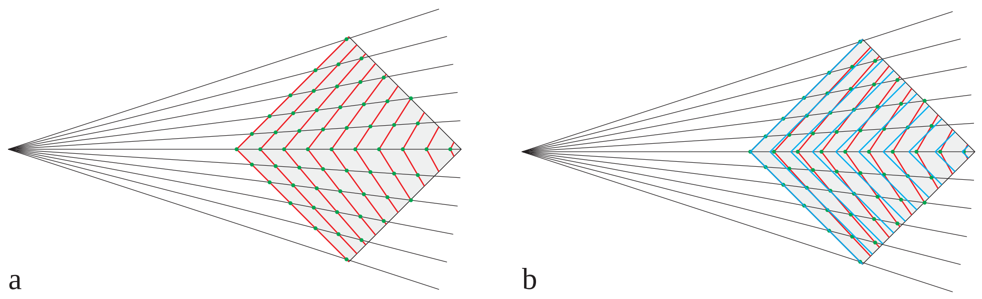

Figure 1: *Image a) shows the exact sampling pattern of a ray-caster as seen from atop (green dots) that starts marching at the bounding box entry points. If we connect the samples between those points by line segments (red) it can be seen the sampling pattern is more slice rather than shell like. Image b) shows overlayed in blue the clipped object aligned slices. In black/white print see color page for colored version of this image.*

portant features in the ever growing data sets. For a complete and detailed survey on the various volume rendering techniques we refer the reader to the text book by Engel et al. [EHK*06].

With this development in mind the question arises why nowadays one should care about hardware that does not support a 3D textures. Surprisingly, the vast majority of both mobile and desktop PCs do not support 3D textures [Int08] as a Mercury Research study estimates. Now one may argue that most of these systems with chipset integrated GPUs are only used as servers and are never used for volume rendering. Again, it is surprising to see that this is not the case. When we recently released the volume rendering system *anonymized* we implemented a telemetry feature that—with users consent—transfers data system failures back to our servers, interestingly the evaluation of this data in the last year has shown that 85% of the system failures are caused by missing 3D texture support. Thus, we decided to reinvestigate 2D texture based rendering as apparently a large amount of users do not have system with 3D texture support. If one considers even smaller devices such as netbooks, tablets or PDAs practically no 3D texture support is present.

2. Contribution

To overcome the stack switching artifacts of the 2D texture stack approach we present a simple yet powerful extension to Rezk-Salama et al.'s approach, eliminating the need to switch between the stacks that caused the noticeable changes in the color and opacity values. It is worth noting, that even on hardware that supports 3D textures our approach eliminates the boundary artifacts of view aligned slicing, and gives better performance.

3. Our Novel Method

In this section we will first give an intuitive explanation of our method, then describe a straight forward implementation and finally mention an optimized geometry generation method.

3.1. Key Idea

The key idea of our novel method is the sampling pattern created by most ray-casting implementations such as the one proposed by Krüger and Westermann [KW03]. In contrast to the spherical shells proposed by La Mar et al. [LMHJ99] these methods start at the entry point of the volume and progress with regular steps until they leave the volume (green dots in Figure 1a). As every ray in this implementation starts at different length the sampling pattern does not resemble shells but is dominated by the entry points into the bounding box (red lines in Figure 1a). Comparing this sampling pattern to object aligned slices clipped at their intersection points (blue lines in Figure 1b) shows that this is a reasonable approximation. Using the object aligned slices in this way in 3D is our novel strategy. The two topmost rows in Figure 2 show a comparison of the patterns in 3D.

3.2. Implementation

In the previous section we motivated our basic sampling strategy, which uses the same stack geometry as proposed by Rezk-Salama et al. but instead of selecting only a single stack, we use multiple stacks clipped in 2D at the line that starts from the front facing vertices and points in the viewing direction. In 3D, planes originating from the front facing edges of the bounding box are used to separate the stacks. To generate our improved proxy geometry we simply need to find the front facing edges, extend those to planes in the viewing direction, and clip the three object aligned stacks adjacent to these edges. Then we may draw the remaining geometry of the tree stacks in front to back or back to front order. Note that as the stacks do not overlap in depth we do not need to consider inter-stack sorting but can draw them in any order, which is advantageous for out-of-core data as we can first draw the data that is resident in the cache.

In order to achieve a similar spacing for all the stacks we scale the slice distance by the secants of the angle between the viewing direction and the stack normal (see Figure 3). Naturally, this angle varies over the slice but as only a sin-

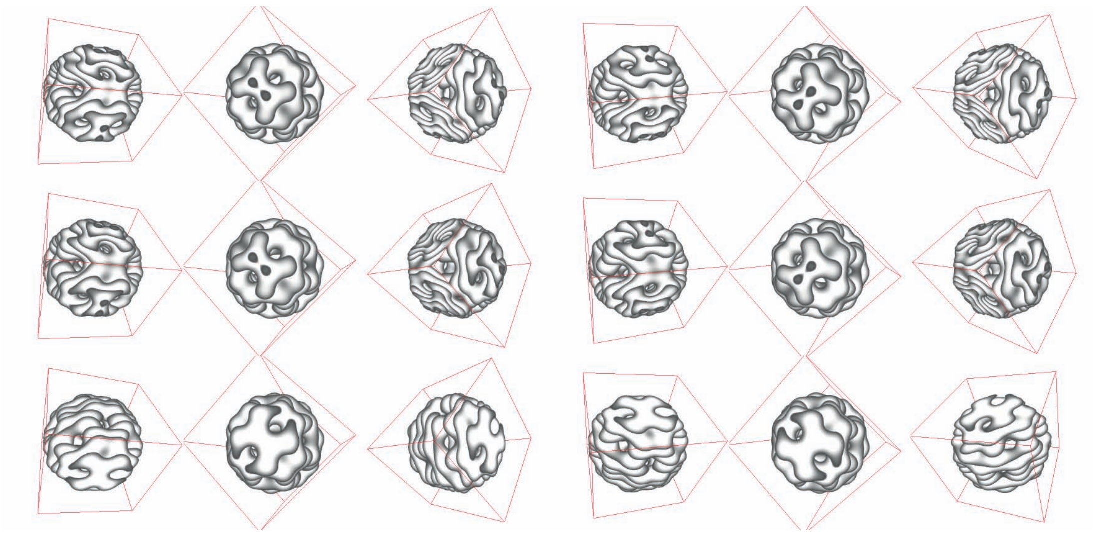

Figure 2: *A sequence of images of the C60 data set sampled at very low resolution to reveal the sampling pattern. Rendered with our novel method (top) a ray-caster (center) and traditional object aligned slices (bottom). As can be seen our method much more closely matches the ray-caster's sampling pattern.*

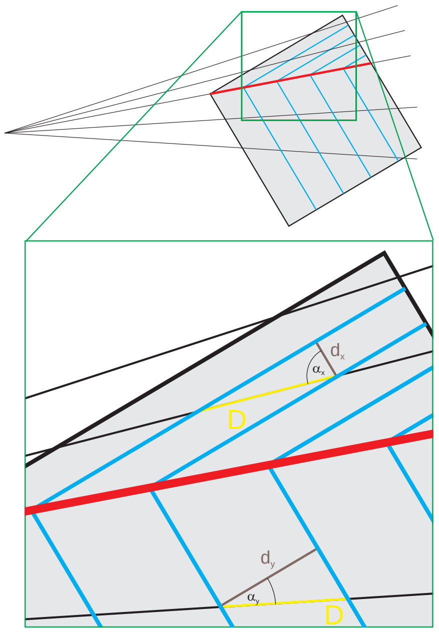

Figure 3: *The secant distance adjustment. The lower image shows a zoomed-in view of the sampling pattern above. To approximate a uniform distance D along the ray the slice distance d needs to be scaled by the secant of the angle between the view ray and the stack normal.*

gle slice distance can be chosen we approximate this scaling factor by the secants of the angle between stack normal and the ray to the slice center.

Since our approach adjusts the spacing between the slices we do not necessarily render one geometry slice per data set slice anymore. Therefore, we either interpolate between two 2D textures in a shader if possible or repeat the same 2D texture for multiple slices on older hardware. In the examples throughout this paper we used the former alternative.

3.3. Fast Geometry Generation Optimization

In the previous section we introduced the proxy geometry generation as first computing the quads in all three directions and then clipping these stacks appropriately. This approach while easy to understand, however, results in a significant overhead. If the clipping is performed on the GPU, large amounts of geometry are send to the GPU but are never rendered as most of the slices are clipped away. This approach also results in many unnecessary state changes as 2D textures are changed for the slices. This problem can be avoided by performing the polygon clipping on the CPU. In this case, however, the clipping routine quickly becomes a bottleneck.

This bottleneck can be avoided by simply clipping the bounding box before the stack generation instead of clipping the actual stack. Then we march along the bounding box with the secant corrected step size. As the clip planes extend from the edges of the bounding box they only clip away parts perpendicular to the stack generation direction and thus the generated geometry remains quadrilateral.

4. Conclusions and Future Work

In this paper we have presented a novel sampling scheme for slice based volume rendering, which is particularly useful for devices with a lack of 3D texture support. On these devices our method significantly improves the rendering quality, with only a marginal performance impact. On ultra mobile hardware we tested i.e. on a third generation iPod touch

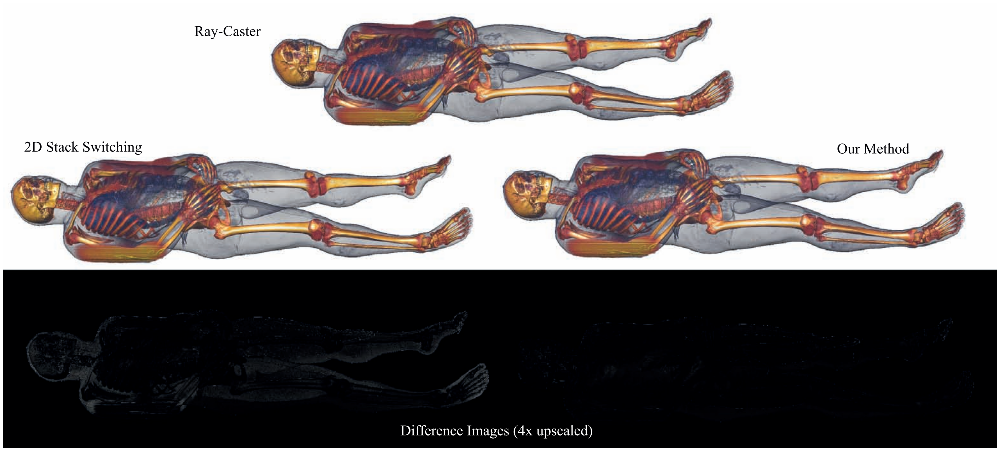

Figure 4: *The visible human frozen CT scan, an example of a larger data set (512x512x1900) rendered with our method (right) and the previous 2D texture method (left) using a step size equal to the distance d computed by our approach. On top the same view rendered with GPU based ray-caster is shown. Although the stack switching artifacts are hard to show in a single frame, the difference images on the bottom reveal that the novel method results in a closer approximation of the ray-cast image. In black/white print see color page for colored version of this image.*

and the iPhone 3Gs as well as on PC hardware with GPUs incapable of 3D textures (i.e. Intel integrated GMA graphics) the performance compared to an object-aligned implementation dropped by about 20%. We address this loss in performance to the additional slices drawn due to the secants correction step. While impacting the performance the quality in particular for smaller data sets increases significantly and even for larger data sets our method produces better quality than the previous approach (see Figure 4 for a comparison to a ray caster).

On devices that do support 3D hardware our algorithm is still of interest as an alternative to the view aligned slicing, since it eliminates the jagged artifacts at borders. In addition to this improvement in quality, we also observed increased performance of the novel method over the view aligned slicing. Even when using 3D textures, on an NVIDIA Quadro 5800 in core data sets (i.e. data that is already resident as 3D textures in GPU memory) were rendered about 30% faster using our method, while on an ATI/AMD Radeon 5970 series the performance increased by about 25% for out of core rendering. Tests were measured over a full 360 degree rotation. For out of core rendering the data transfer dominates the overall rendering time and the speedup decreases accordingly. We attribute this performance gain to a more GPU friendly traversal of the data.

For the future we intend to continue our investigation on the performance implications of our technique. We also intend to contact the GPU manufacturers to investigate the speedups further. Maybe it is feasible to give the graphics card a hint that we only access the 3D texture axis aligned, we assume that our access pattern should be much easier to optimize than the more arbitrary access of a ray-caster or view aligned slicing.

References

[CCF94] CABRAL B., CAM N., FORAN J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization* (New York, NY, USA, 1994), ACM, pp. 91–98. http://tinyurl.com/Cabral1994. 1

[CN94] CULLIP T. J., NEUMANN U.: *Accelerating Volume Reconstruction With 3D Texture Hardware*. Tech. rep., University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1994. http://tinyurl.com/Cullip1994. 1

[EHK*06] ENGEL K., HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D.: *Real-time Volume Graphics*. A. K. Peters, Ltd., 2006. http://tinyurl.com/Engel2006. 2

[Int08] INTEL/MERCURY RESEARCH: Common misconceptions of Intel integrated graphics. http://tinyurl.com/Intel2008, 2008. 2

[KW03] KRÜGER J., WESTERMANN R.: Acceleration techniques for GPU-based volume rendering. In *Proceedings IEEE Visualization* (2003). http://tinyurl.com/Krueger2003. 1, 2

[LL94] LACROUTE P., LEVOY M.: Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM, pp. 451–458. http://tinyurl.com/Lacroute1994. 1

[LMHJ99] LA MAR E. C., HAMANN B., JOY K. I.: Multiresolution techniques for interactive texture-based volume visualization. In *VISUALIZATION '99: Proceedings of the 10th IEEE Visualization 1999 Conference (VIS '99)* (Washington, DC, USA, 1999), IEEE Computer Society. http://tinyurl.com/LaMar1999. 1, 2

[RGW*03] RÖTTGER S., GUTHE S., WEISKOPF D., ERTL T., STRASSER W.: Smart hardware-accelerated volume rendering. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 231–238. http://tinyurl.com/Roettger2003. 1

[RSEB*00] REZK-SALAMA C., ENGEL K., BAUER M., GREINER G., ERTL T.: Interactive volume rendering on standard pc graphics hardware using multi-textures and multi-stage rasterization. In *Proc. ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2000). http://tinyurl.com/Rezk2000. 1

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A Simple and Flexible Volume Rendering Framework for Graphics-Hardware–based Raycasting. In *Proceedings of the International Workshop on Volume Graphics '05* (2005), pp. 187–195. http://tinyurl.com/Stegmaier2005. 1