Eurographics Symposium on Parallel Graphics and Visualization (2010), pp. 1–8
J. Ahrens, K. Debattista, and R. Pajarola (Editors)

Multi-Frame Rate Volume Rendering

Stefan Hauswiesner and Denis Kalkofen and Dieter Schmalstieg

Graz University of Technology, Institute for Computer Graphics and Vision

Abstract

This paper presents multi-frame rate volume rendering, an asynchronous approach to parallel volume rendering. The workload is distributed over multiple GPUs in such a way that the main display device can provide high frame rates and little latency to user input, while one or multiple backend GPUs asynchronously provide new views. The latency artifacts inherent to such a solution are minimized by forward image warping. Volume rendering, especially in medical applications, often involves the visualization of transparent objects. Former multi-frame rate rendering systems addressed this poorly, because an intermediate representation consisting of a single surface lacks the ability to preserve motion parallax. The combination of volume raycasting with feature peeling yields an image-based representation that is simultaneously suitable for high quality reconstruction and for fast rendering of transparent datasets. Moreover, novel methods for trading excess speed for visual quality are introduced, and strategies for balancing quality versus speed during runtime are described. A performance evaluation section provides details on possible application scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.1 [Computer Graphics]: Hardware Architecture—Parallel processing

1. Introduction

Modern workstations support multiple GPUs and lend themselves to parallel volume rendering approaches. Previous parallel volume rendering systems have focused on balanced workloads for maximum utilization of resources. Perfect load balancing can yield nearly linear speedups in the number of GPU nodes, but this may still be insufficient for interaction with very complex scenes. A recent trend in research therefore focuses on multi-frame rate rendering [SBW*07], which decouples display updates from image generation into a pipeline with asynchronous communication. The display update stage can guarantee fast frame rates and nearly latency-free response to interaction, while one or multiple GPUs in the backend stage can produce new high quality images at their own, slower pace.

However, since the slow nodes and the fast node operate at different frame rates, latency artifacts may arise: objects drawn by the slow nodes react delayed to user input. The problem can be reduced by image-based rendering (IBR) methods, which are able to reconstruct an image seen from a new camera position using one or more images from previous camera positions.

Multi-frame rate volume rendering is especially challenging because of the need for transparent volumetric objects. When a transparent object moves relative to the camera, its inner structure is revealed by features moving non-uniformly in screen space. Previous multi-frame rate systems reconstruct the whole scene as a single surface, which destroys such a motion parallax perception. Moreover, view-dependent lighting effects of transparent inner structures are not possible by shading only a single surface.

The contribution of this paper is a system that combines multi-frame rate volume rendering with a novel layered approach to image warping. This combination addresses the problem of high quality rendering of transparent volumes with multi-frame rate division of labor. We report on an efficient and effective layer separation relying on feature peeling and on corresponding shading computations.

Furthermore, we observed that image warping has become a fairly performant task on modern GPUs, thus allowing well-defined raycasting tasks to be performed on the fast-node GPU in addition to image warping. To make use of excess rendering time, advanced workload distribution schemes are introduced, which trade speed for visual quality. This process can even be driven automatically, which results

in a dynamically optimized quality vs. speed tradeoff during runtime. All techniques are evaluated quantitatively with respect to visual impact and rendering performance.

2. Related work

Volume rendering by raycasting was introduced by [Lev88] and has remained an active topic of research since, because of the high complexity of the problem. Recently interactive volume raycasting has become possible on the GPU, for example in [SSKE05] or [KGB*09], which is the CUDA-based raycaster used in this work.

Several approaches of parallel volume rendering by distributing computations to multiple rendering nodes emerged. Most of these methods can be categorized into three classes: sort-first, sort-middle and sort-last [MCEF94]. Sort-last systems with multiple GPUs such as [MMD08] make it easier to distribute and balance the workload over the render nodes, but require a lot of memory transfer in the composition stage. Our system sorts fragments in the compositing stage, and therefore can be classified as sort-last.

For coping with situations where the computation load exceeds the interactive rendering power of a parallel system, but some computations have to be performed with minimum latency, the concept of multi-frame rate rendering was introduced [SBW*07]. The speed advantage of multi-frame rate rendering comes at the price of visual errors: due to the discrepancy between the slow and fast nodes, visual artifacts may arise, until the scene comes to a rest, *i. e.*, none of the parts of the scene move relative to one another and the viewpoint. The error may become intolerable for fast camera movements.

Image warping [MB95] is a form of image-based rendering that allows to extrapolate new views from existing images with per-pixel depth information. This technique is capable of reducing the visual errors inherent to multi-frame rate rendering.

The work in [SvLBF09] and earlier publications of the group describe a multi-frame rate architecture, which accelerates image warping by using vertex shader programs for the required scatter operation. It covers the rendering and communication delay by producing intermediate frames, which are warped from previous frames. The group reports on problems with the reconstruction quality, and the tradeoff between quality and latency that such a system has to make. However, only non-transparent volume rendering was considered. [SLRF08] support multi-frame rate volume rendering with transparency, but assume a fixed view point. Moreover, no deferred shading step is applied to the volume rendering and workload distribution is less flexible.

Other image-based volume rendering approaches, such as pixel ray images [SLSM06] or light field representations [RSTK08] produce richer descriptions of the volume, but require considerable preprocessing and are less suitable for frequent bus transfer due to their size.

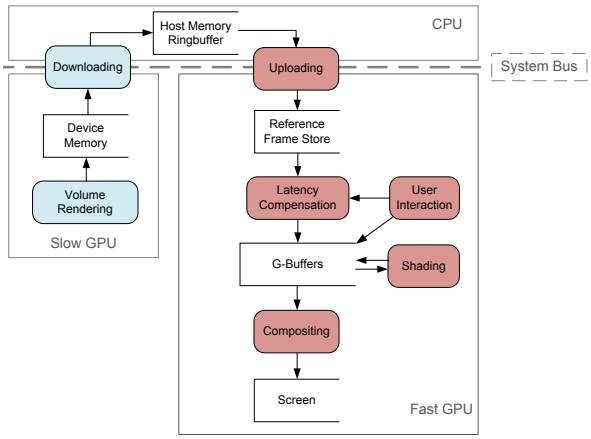

Figure 1: *The conceptual architecture of our system. The task distribution between a slow and a fast node is shown.*

3. System overview

From the user's point of view, our system looks like a simple model viewer. The camera is able to orbit the model and move towards the model or away from it, thus enabling zoom operations. The underlying rendering process is arranged in a pipeline starting at the slow node or nodes and continuing at the fast node. The GPU nodes of the system are not required to have equal hardware. Each GPU node executes CUDA code and is controlled by a separate CPU thread. Results are always buffered in local GPU memory to allow asynchronous operation of the pipeline. The slow node retrieves the instructions which image to produce as a result of user interaction (camera change). The fast node is mainly concerned with latency compensation and image compositing. However, it also processes user input and provides low-latency visual feedback. The main steps of the pipeline (figure 1) are:

Raycasting As the frame source of our system, we use a raycaster implemented in CUDA [KGB*09], which supports a large number of volumes, complex translucent and concave polyhedral objects as well as CSG intersections of volumes and geometry in any combination. It provides a color, depth and normal buffer for later stages of the pipeline.

Transfer to fast node Since no direct data link between GPUs exists, the image produced by the raycaster must first be downloaded to host memory from the slow node, and then uploaded to the fast node. The intermediate storage is process-local heap memory, arranged in a ring buffer, and therefore can be accessed by all rendering threads. This shared memory is also used for thread-safe communication

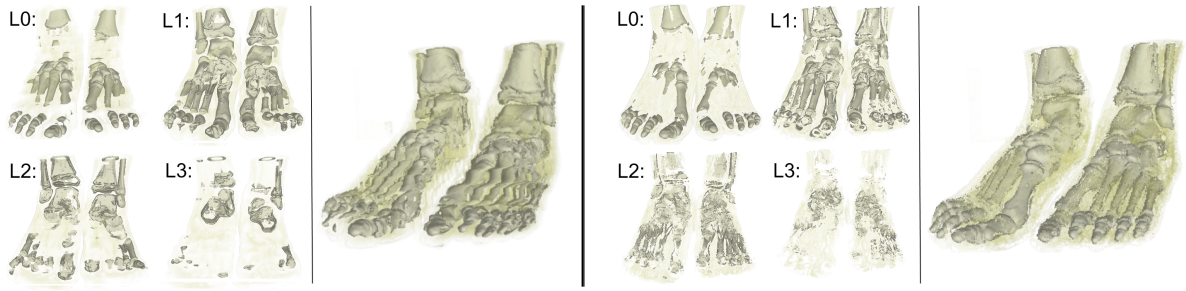

Figure 2: *Two different ways of layering a volumetric foot dataset (VIX [Osi09]), showing the first 4 layers and the resulting image when warped for a rotation of 21 degrees. The left layering approach slices the volume at regular intervals, whereas the right approach adapts to the features of the dataset.*

of job scheduling and flow control in the pipeline. Unfortunately, the data transfer through the host memory can be a major limitation of performance, especially when the frame rate of the slow node approaches the frame rate of the fast. Until a fast inter-GPU link becomes available, multi-frame rate rendering is therefore only economic for large rendering problems.

Latency compensation by image warping The reference frames uploaded to the fast node are stored on the GPU. From there they can be warped into G-Buffers. Each G-Buffer stores color, normal and depth. G-Buffers can be processed by a number of different operations, for example, morphological closing. Moreover, the fast node can produce its own renderings directly into a G-Buffer. Rendering on the fast node can encompass for example of magic lens effects or manipulation widgets, and use raycasting or another rendering technique. The only condition is that this additional rendering is lightweight and does not slow down the fast node too much.

Shading To correctly calculate view-dependent lighting in such a pipeline, the shading step has to be performed after image warping. The reason for this is that fragments are potentially viewed from a different direction than in the reference frame.

Compositing The processed buffers are input to the compositing step, which sorts all fragments per pixel, blends them together by accumulating the values from front to back and finally writes the result to the frame buffer. Compositing is entirely achieved by CUDA kernels.

4. Layered image warping

Conventional image warping often operates on opaque polygons and therefore only needs one warping operation per pixel. If the viewpoint is moving, the amount of projected screen space motion depends on the distance of a fragment. This effect is called motion parallax. With transparent objects, it is possible to perceive more than one fragment per pixel, and therefore a single image warp cannot provide sufficient cues about the inner structure of an object. Moreover, image warping requires deferred shading to correctly handle view-dependent lighting. A transparent volume cannot be shaded well by using a single map of normal vectors.

To overcome the most severe limitation of single image warping, a more detailed representation of the inner structure of the object is necessary. This structure can be represented by multiple partial images, each containing a rendering of a part of the volume, a layer. For a most efficient layered representation, the elements of the inner structure should be grouped by distance from the camera, but also by similar opacity, because opacity is relevant for consistent shading. For the creation of a suitable representation that addresses these requirements, we adopted the idea of feature peeling [MMG07]. Local minima and maxima of smoothed opacity along the ray are detected and used to separate layers. To further suppress noise, all layers are required to have at least 10 percent opacity. Separation of connected, highly opaque regions is avoided, as they usually belong to the same feature.

Each pixel in a layer image produced by the raycasting of a volume must be created together with a depth value, so that it can be used for warping. The depth value should represent the interval along the ray that was considered for creating the pixel. The entrance or exit depth along the ray interval would be simple choices, but frequently fail to capture the main effect of light attenuation, which may happen somewhere in the middle of the ray interval. Figure 2 illustrates this observation. We evaluated two ways of finding this main effect: it is either located at the point with the maximum opacity after classification, or at the point with the maximum impact on the color, which is calculated by the voxel's opacity multiplied with the remaining opacity along the ray segment. We found both approaches viable, with slight advantages for the maximum color impact method. See figure 3 for such a layering with depth selection. Local maxima in the discrete

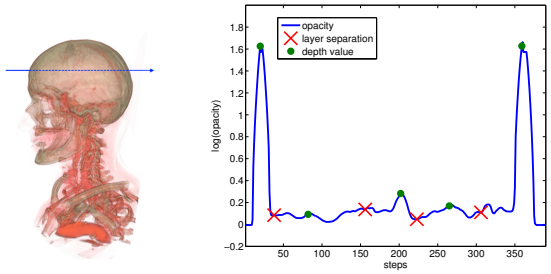

Figure 3: *Ray profile of a ray cast through the head CT dataset MANIX [Osi09]. On the left side the location of the ray is shown. The right side visualizes the smoothed opacity values encountered along the ray, and possible layer separation points. The two peaks with high opacity represent the skull bone.*

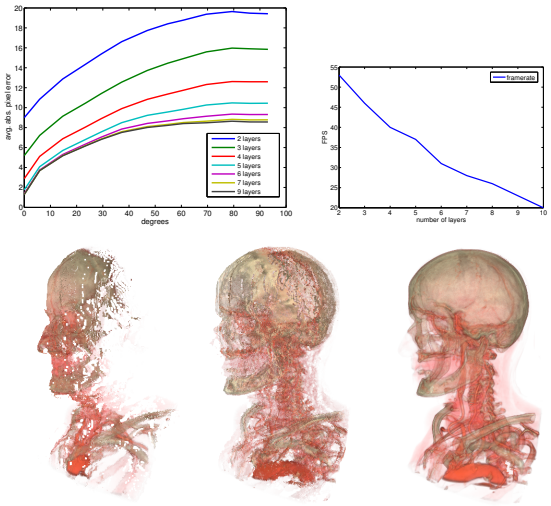

Figure 4: *The left plot shows the average pixel error of the MANIX dataset over an increasing reconstruction angle for different layer numbers. The right plot shows the frame rate for these layer numbers at a resolution of 512x512. It becomes apparent that the benefit of warping more than 6 layers vanishes, whereas the performance degrades linearly. The bottom row shows the dataset at a rotation of 55 degrees, with non-layered, 10-times layered and reference images.*

domain may span several consecutive sample points. In such cases we select the median of these sample points.

The visual approximation with this approach is better than with a single layered image warp, because depth cues are preserved. Figure 4 shows data from a reconstruction quality measurement. For the *MANIX* head dataset with a fairly transparent transfer function, the layered approach outperforms the single surface approach easily. The *VIX* foot dataset (see figure 2) uses an opaque transfer function revealing the bones: the difference in quality is not as significant, as there is less inner structure.

Note that layer images can be transferred separately to the fast node as soon as they are completed, while raycasting commences. Consequently, more layers reduce the latency of the slow node as the higher priority layers become available earlier to the fast node.

However, the feature peeling used to separate layers is a heuristic and can fail to produce satisfactory results. Memory for layers must be allocated in advance, which means that only a fixed number of layers is available. If these layers are not distributed well inside the volume, the visual quality is not optimal. Such artifacts can be caused by an insufficient number of layers, or by an overly sensitive layering heuristic, which "wastes" layers on an over-representation of features close to the camera, while distant features are underrepresented as a consequence. See figure 4 for an evaluation.

To correctly calculate view-dependent lighting, the shading coefficients must not be multiplied with the resulting color of a fragment before image warping. Instead, a normal vector is stored for each fragment, and the shading calculation is performed in a deferred shading step before compositing the final image. However, choosing a representative normal for a ray segment is not trivial, and can only be an approximation to a conventionally shaded raycasting.

During ray traversal, a volume gradient can be computed at each sampling step, but for deferred shading we can only use one normal per segment. Possible strategies include selecting the longest gradient, or using transfer function information to select the gradient with the highest impact (=opacity) on the ray segment. Following the notion of maximum visual impact, the gradient vector can be accumulated very much like the color RGB-vector: by multiplying the current alpha and the remaining opacity with the normalized gradient, and adding to the result. See figure 5 for evaluation results that rule out the longest gradient and the gradient at the opacity maximum.

CT or MRT datasets often suffer from noise in the measured data, which consequently leads to very noisy gradients. Choosing the wrong gradient for representing a ray segment causes strong visual differences to a per-sample shaded raycasting, as can be seen in the example images of our evaluation. The adopted strategy after empirical investigation is therefore a modified accumulation algorithm, which weights closer samples along a ray segment even stronger than the normal accumulation. This behavior can be achieved by taking the remaining opacity to the third power before accumulation. Since opacity values are defined in the range from 0 to 1, the impact on the resulting gradient falls off very quickly along the segment. The pixel-error evaluation reveals that this can indeed be a successful strategy.

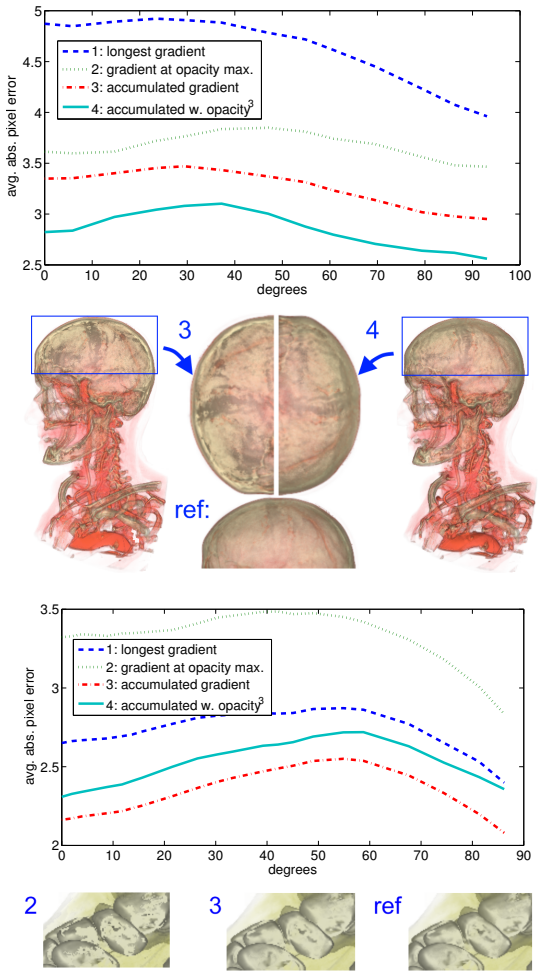

Figure 5: *The figure shows the reconstruction error over an increasing angle to the reference position with three different calculation methods for the fragment normal used by deferred shading for two datasets MANIX (top) and VIX (bottom).*

5. Advanced workload distribution schemes

A typical workload distribution in a multi-frame rate system assigns the raycasting to the slow node, while image warping, which is used for latency compensation, is performed on the fast node. The performance of raycasting scales with viewport resolution and sampling rate, and if either or both of these factors is small enough, raycasting on modern GPUs can be acceptably fast. In such a case, the strict assignment to slow and fast node can be relieved, and visual quality can be improved by replacing some of the image warping by raycasting. This of course reduces the frame rate of the fast node, so special attention has to be paid to the amount of workload that is shifted. The latency compensation step of the system (see figure 1) is extended by raycasting.

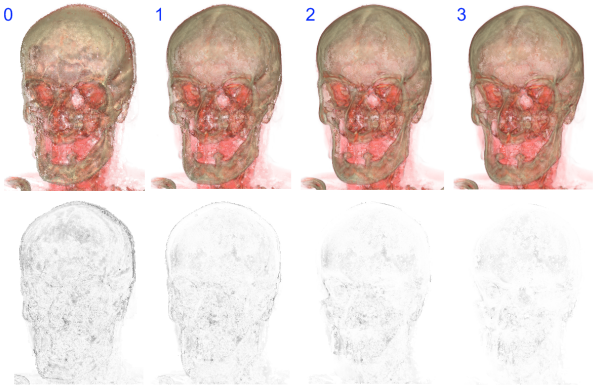

Figure 6: *Layered separation of raycasting and warping with number of raycasted layers printed above and resulting pixel difference to a fully raycasted reference image.*

Typical parallel raycasting algorithms divide the workload by separation planes either in screen space or object space (sort-first or sort-last, respectively). When using such a separation plane or region to divide forward from backward sampled areas, this approach often leads to a perceivable boundary. This is mainly due to the fact that the deferred shading from warped images is only an approximation to a per-sample phong shading. Moreover, these regions do not adapt well to what is really interesting in a scene: often there are focus regions, segmentations or magic lenses, which better describe important scene features. Even if no special region is present, one can assume that features close to the current view point are more important and better perceivable than distant features. To account for these facts, a more flexible workload distribution has to be found, and sampling algorithms have to be assigned accordingly.

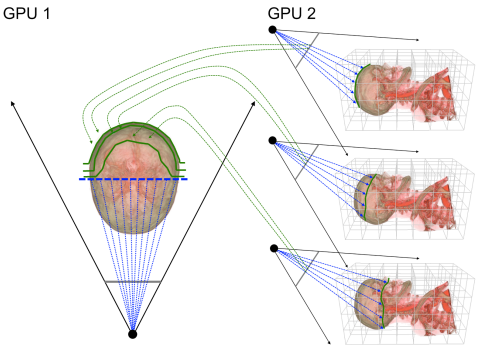

Figure 7: *Combination of raycasting (backward sampling, shown as blue arrows) and forward image warping (shown as green arrows) on two different GPUs.*

5.1. Distribution by depth layer

The distribution by depth layer exploits the observation that close structures inside a volume dataset typically contribute more to the final image than distant structures. It is therefore beneficial to use the best sampling quality (which in our case is achieved with raycasting) for the first n depth layers, and multi-frame rate image warping for the remaining layers. This implies that raycasting of close layers is performed by the fast GPU, and raycasting of back layers by the slow GPU. The number of layers which are raycasted controls the tradeoff between quality and performance. This process can even be driven automatically, depending on the current frame rate and a desired minimum frame rate, and therefore gives the best quality for a given maximum latency. Our system decreases the number of raycasted layers immediately when the current frame rate falls below the minimum. Likewise, it periodically renders the scene with $n+1$ raycasted layers to check if improved quality is possible. See figure 7 for a schematic view of the process. Figure 8 contains evaluation results, which were obtained from the scene shown in figure 6.

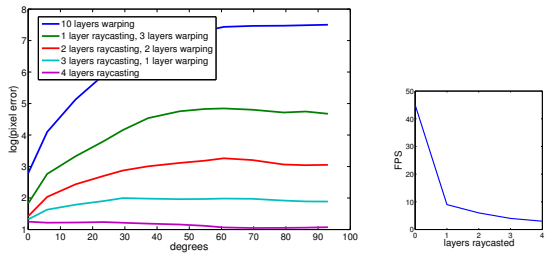

Figure 8: *Difference error to a reference image during a rotation of 90 degrees (left). Layered warping is compared with hybrid raycasted and warped reconstruction. With more layers raycasted, the reconstruction converges into the reference image. The right plot shows the frame rate for these setups.*

5.2. Distribution by region of interest

Magic lens applications follow the notion of focus and context, with focus regions being defined by a 2D or 3D shape. The user indicates the region of interest, so the sampling quality can be adapted easily: the interior of the magic lens is raycasted with best quality on the fast GPU, whereas context is established by image warping of frames, which are produced on a slow GPU.

Segmentations which are common in medical imaging, behave like fixed 3D magic lenses and therefore also mark regions of interest. The fact that a segmentation is defined in object space can be exploited: like in typical sort-last parallel systems, each rendering node only needs to keep its sub-scene graph in memory. However, this is only feasible if the regions are static in object space.

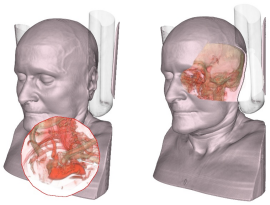

Figure 9: *Focus and context rendered with raycasting and image warping respectively in a magic lens setup (left), and in a segmentation setup (right).*

Figure 9 shows these two types in a focus and context setup. The advantage of this approach is the ability to let the user decide about the quality/speed tradeoff, which happens naturally as the focus region is created or moved. However, this introduces also the disadvantage of not being able to predict frame rates during a user session.

5.3. Distribution by quality maps

Another way of driving the sampling scheme is based on actual image quality. For example, [YNXC03] use the disocclusion of a reprojected iso-surface to detect holes in the output image, and apply raycasting to these pixels. Similarly, we can use the proxy geometry of the volume dataset along with information on the content of the layers of the volume to produce a map with relative pixel quality. Badly reconstructed pixels, which mostly come from disocclusions or undersampling during image warping, can be discarded and replaced by a high-quality raycasting. Figure 10 shows a scene augmented by its quality map for the current view and the result after raycasting the disocclusions.

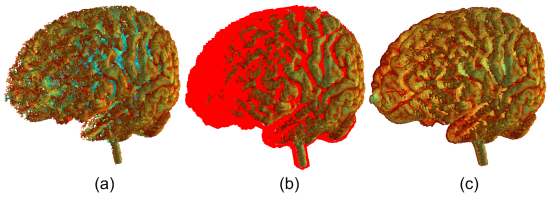

Figure 10: *Disocclusions of the image warping process (a) are detected to form a quality map (b). Low quality regions are filled by raycasting (c). The quality map parameter m is set to* 1 *in this example.*

To create such a quality map, the fast node executes the following tasks: first, use image warping to reconstruct m layers. Then project the volume's bounding geometry to the screen. Iterate through the closest m layers for each pixel that is covered by the projection. If layers are empty then raycast this pixel.

In this way, all pixels which are disoccluded to more than a certain extent are replaced by correct, raycasted pixels. The parameter m defines how many empty layers are required per pixel to classify the pixel as disoccluded. Note that in a layered image warping system, completely empty pixels are rare, but holes in the foremost layers commonly happen and disocclude content from the back layers, which results in unsatisfying reconstructions.

The performance of such an approach scales with the amount of frame-to-frame coherence: large viewpoint motion induces more raycasting and thus lower performance, whereas slow and steady motion allows for using more of the warped fragments. The system is therefore able to prevent strong disocclusions, but performance is not easily predictable: the user's input influences the sampling behavior of the system. Nevertheless, the m parameter can be used to roughly steer the quality vs. performance tradeoff. Datasets, which are relatively fast to render via raycasting, should receive a lower m in order to switch to raycasting more frequently.

6. Performance evaluation

When the fast node is entirely based on image-based rendering, its performance directly scales with the screen resolution. Each pixel has to be transferred, warped and shaded several times, depending on the number of layers.

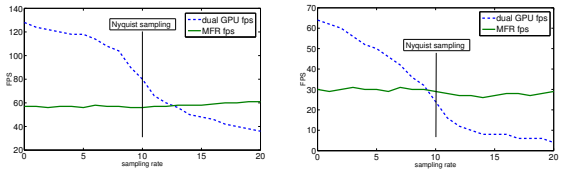

Figure 11: *Performance of the two datasets dice (left) and MANIX (right) at different sampling rates, ranging from* 6 *times undersampling to* 6 *times oversampling. The plot compares a multi-frame rate system versus an assumed parallel approach with a* 512x512 *viewport resolution.*

Figure 11 shows the results of a performance test with two datasets. We compare a single-frame rate dual GPU approach, which we assume to be twice as fast as the single GPU raycaster in our system, and our multi-frame rate system. For reconstruction we use 3 layers for the dice (64^3 voxels), and 7 layers for the *MANIX* dataset (256x256x230 voxels), both with deferred shading. The frame rate is measured for different sampling rates, from six times undersampling to six times oversampling compared to Nyquist rate sampling, thereby covering a wide range of workloads.

The results show that if the complexity of the volume rendering exceeds a break even point, it is beneficial to use multi-frame rate rendering. As expected, the fast node's frame rate stays stable, independent of the sampling rate. In fact, due to the frequency of bus transfer, the fast node's performance increases slightly when the slow node's performance degrades. At the Nyquist sampling rate, the *MANIX* dataset is better handled with multi-frame rate rendering, whereas the simple dice dataset is not. The opaque surface plot in figure 12 shows the *MANIX* dataset under a varying pixel resolution and sampling rate. For most of the input range, multi-frame rate again proves to be superior. Moreover, the frame rate is more stable than with sort-first.

7. Dynamic single and multi-frame rate rendering

When the raycasting task becomes simple enough, however, multi-frame rate rendering may become obsolete or even unsatisfying, and the rendering system should fully converge into single-frame rate multi-GPU raycasting, as conventional sort-first or sort-last parallel volume raycasting avoids image warping artifacts entirely.

With decreasing raycasting complexity, the dynamic layer distribution approach mentioned in section 5.1 would eventually migrate all layers onto the fast node to be raycasted there, which follows the notion of converging into a single-frame rate rendering. Similarly, the quality map approach of section 5.3 converges into full raycasting with a m setting of 0. But this also renders additional GPUs in the system useless, as the raycasting is then entirely performed by the fast node.

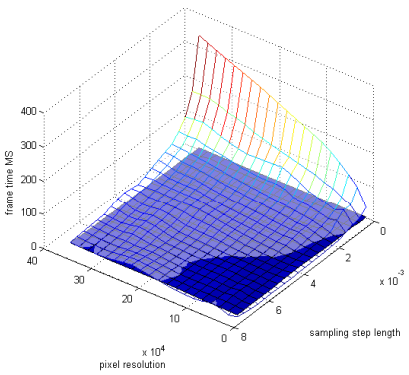

Figure 12: *Performance of the MANIX dataset for different resolutions and sampling rates with multi-frame rate rendering (opaque mesh) and conventional sort-first parallel raycasting (wireframe mesh).*

To counter this problem, the system has to switch to conventional sort-first or sort-last parallel raycasting when appropriate, thus utilizing all available resources. Raycasting performance depends on a variety of parameters, for example the sampling rate, zoom level, occlusion, dataset size and structure or transfer function. Depending on the application,

some of them can change during runtime. For these parameters, performance data can be collected for their value range to form a decision function, which yields the best rendering method for the current frame. This data has to be recomputed when one or more of the performance relevant parameters, which are not input to the function, change.

Figure 12 shows such a decision function for the two parameters sampling rate and resolution. The resolution is measured by computing the pixel area of the bounding rectangle, which makes it a measurement of the model's *zoom* level. The evaluation setup is equivalent to the *MANIX* setup also shown in figure 11. The measurements for this plot took roughly 2 minutes, but a good approximation can be obtained within 10 seconds. Evaluating the function has virtually no performance impact. The blue, opaque region at the bottom shows a range in which sort-first parallel raycasting performs superior to multi-frame rate rendering. Our system exploits such a situation by evaluating the decision function periodically and switching the rendering method when appropriate.

8. Conclusions

We have described a system that is suitable for displaying transparent volumetric scenes at high frame rates and little latency to user input using a multi-frame rate approach in combination with image-based rendering techniques. We implemented an evaluation system using GPUs as rendering nodes with unbalanced workloads. The fast node achieves the required low latency, whereas the slow nodes act as the frame source. The fast node is not synchronized with any of the slow nodes to uncouple their rendering cycles and thus increase speed.

This work focuses on applications of direct volume rendering by raycasting, which requires extensions to existing approaches: reference images are produced in layers defined by feature peeling. We discussed methods for selecting representative depth values and normal vectors. Latency compensation is performed by image warping from multiple layers. This approach preserves motion parallax, in particular for complex interior structures of transparent volumes.

Moreover, we presented novel ways of workload distribution for multi-frame rate volume rendering systems: by depth layer, by region of interest and by quality maps. Excess speed can be traded for reconstruction quality, even dynamically during runtime. Our implementation was evaluated and limitations are described. The process of recording performance data, which allows for a dynamic decision between multi- and single frame rate, is presented.

Future work should focus on improving the quality of the image warping process, which may include postprocessing and image-caching. Also, revisiting the idea of frameless rendering [BFMZ94] for reducing the bus load may be of interest. Direct communication between GPUs would remove the bandwidth limitation altogether and allow for improved collaboration.

Acknowledgements This work was supported by the Austrian Research Promotion Agency (FFG) under the BRIDGE program, project #822702 (NARKISSOS).

References

[BFMZ94] BISHOP G., FUCHS H., MCMILLAN L., ZAGIER E. J. S.: Frameless rendering: double buffering considered harmful. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM, pp. 175–176. 8

[KGB*09] KAINZ B., GRABNER M., BORNIK A., HAUSWIESNER S., MUEHL J., SCHMALSTIEG D.: Ray casting of multiple volumetric datasets with polyhedral boundaries on manycore gpus. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–9. 2

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications* (1988). 2

[MB95] MCMILLAN L., BISHOP G.: Plenoptic modeling: an image-based rendering system. In *SIGGRAPH '95: Proceedings of the 22nd annual conf. on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM, pp. 39–46. 2

[MCEF94] MOLNAR S., COX M., ELLSWORTH D., FUCHS H.: A sorting classification of parallel rendering. *IEEE Comput. Graph. Appl. 14*, 4 (1994), 23–32. 2

[MMD08] MARCHESIN S., MONGENET C., DISCHLER J.-M.: Multi-gpu sort last volume visualization. In *EG Symposium on Parallel Graphics and Visualization (EGPGV'08), Eurographics* (April 2008). 2

[MMG07] MALIK M. M., MÖLLER T., GRÖLLER M. E.: Feature peeling. In *GI '07: Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 273–280. 3

[Osi09] OSIRIX: Dicom sample image sets. OsiriX Imaging Software, http://pubimage.hcuge.ch:8080/, 2009. 3, 4

[RSTK08] REZK-SALAMA C., TODT S., KOLB A.: Raycasting of light field galleries from volumetric data. *Computer Graphics Forum 27*, 3 (May 2008), 839–846. 2

[SBW*07] SPRINGER J. P., BECK S., WEISZIG F., REINERS D., FROEHLICH B.: Multi-frame rate rendering and display. In *VR* (2007), Sherman W. R., Lin M., Steed A., (Eds.), IEEE Computer Society, pp. 195–202. 1, 2

[SLRF08] SPRINGER J. P., LUX C., REINERS D., FROEHLICH B.: Advanced multi-frame rate rendering techniques. In *VR* (2008), IEEE, pp. 177–184. 2

[SLSM06] SHAREEF N., LEE T.-Y., SHEN H.-W., MUELLER K.: An image-based modelling approach to gpu-based rendering of unstructured grids. In *Volume Graphics* (2006), pp. 31–38. 2

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A simple and flexible volume rendering framework for graphics-hardware-based raycasting. *International Workshop on Volume Graphics 0* (2005), 187–241. 2

[SvLBF09] SMIT F. A., VAN LIERE R., BECK S., FRÖHLICH B.: An image-warping architecture for vr: Low latency versus image quality. In *VR* (2009), IEEE, pp. 27–34. 2

[YNXC03] YUAN X., NGUYEN M. X., XU H., CHEN B.: Hybrid forward resampling and volume rendering. In *VG '03: Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics* (New York, NY, USA, 2003), ACM, pp. 119–127. 6