Eurographics Conference on Visualization (EuroVis) (2015)
E. Bertini, J. Kennedy and E. Puppo (Editors)

Short Papers

Exploratory Performance Analysis and Tuning of Parallel Interactive Volume Visualization on Large Displays

A. Panagiotidis, S. Frey, and T. Ertl

Visualization Research Center, University of Stuttgart, Germany

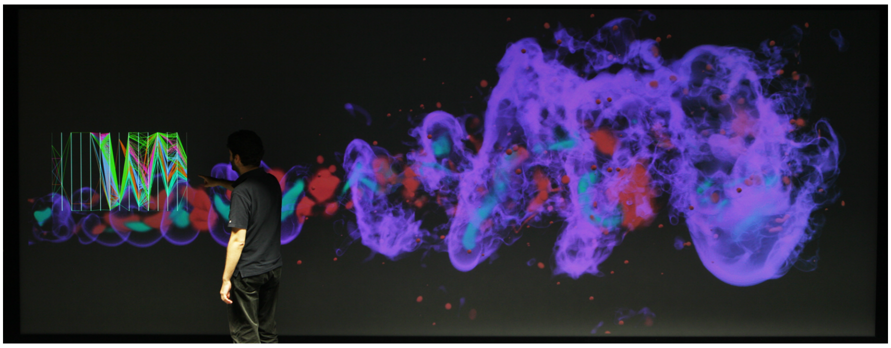


Figure 1: *Real-time performance visualization of parallel interactive volume rendering on a 44-megapixel powerwall.*

Abstract

We present an exploratory approach to performance analysis and tuning of interactive parallel volume visualization for large displays. While traditional approaches target non-interactive applications and focus on separate specialized views for post-mortem performance analysis, we show metrics from the GPU and volume ray casting together with the volume visualization and allow users to interact with both of them simultaneously. With this, users can explore the data set together with the corresponding metrics to investigate both the visual and the performance impact of different parameter settings jointly, like camera position, sampling density, or acceleration technique. In particular, this supports parameter tuning by providing the user not only with timings and quality measures, but also internal metrics from the GPU and the ray caster that help to understand the connection between parameter settings and their induced outcome. We demonstrate the usage and utility of our approach for performance analysis and tuning at the example of distributed volume ray casting for a high-resolution powerwall with the goal to achieve interactive frame rates with the best possible image quality.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.2]: Graphics Systems—Distributed/network graphics;

1. Introduction

Visualization methods are often highly customizable by means of numerous parameters, like transfer functions and sampling rates. Choosing the right parameters for an aesthetic and correct image while maintaining a fluid user experience contrasts with growing system sizes and increasing complexity of hardware, software, and data, as well as the demand for higher resolutions. By analyzing the performance characteristics of such a system, it is possible to find the best trade-off between quality and performance by tuning the respective parameters accordingly. Previous works in the field of performance analysis and visualization establish methods and frameworks but often concentrate on non-interactive applications (e.g., simulations) or infrastructure (e.g., networks) and focus on the collection and analysis of

© The Eurographics Association 2015.

DOI: 10.2312/eurovisshort.20151118

delivered by
EUROGRAPHICS DIGITAL LIBRARY
www.eg.org diglib.eg.org

the performance data itself [IGJ*14]. The context, i.e., the physical domain of simulations or the visualization, is either unavailable, static, or pre-recorded. As such, it is challenging to judge how user interaction and the combination of parameters affect the resulting image and user experience.

In this paper, we discuss an exploratory approach to tuning parallel interactive volume ray casting on a high-resolution display (see Figure 1). Based on metrics collected in real-time from the GPU and the ray caster, we analyze the necessary trade-offs to achieve interactive frame rates while maintaining the best possible image quality. These metrics are presented in real-time together with the volume, so users can fine-tune the parameters of the ray caster to specific data sets for demonstration purposes. We choose an exploratory approach due to the many combinations of the available parameters and because error metrics might differ from perception and aesthetics, especially when considering customized powerwall setups. We discuss our approach by exploring the performance characteristics of different parameter settings and tuning these parameters for a showcase scenario.

2. Related Work

Isaacs et al. [IGJ*14] surveyed the state of the art of performance visualization. They conclude that there is an increasing need for highly scalable visualizations and improved integration of multiple views of performance data. Heath and Etheridge [HE91] employ user-defined annotations in the performance data to correlate it to application code. Wylie and Geimer [WG11] show traced performance metrics in the simulation domain separately from the visualization of the simulation data. Schulz et al. [SLB*11] project performance metrics of a hydrodynamics code onto the respective visualization. In contrast, our approach shows the visualization together with generic and application-specific metrics in real-time. During user interaction with the visualization application, a separate view of the metrics is composited onto the rendering to allow users to correlate their actions and parameter settings to the perceived performance.

We discuss our approach in the context of distributed volume rendering. Beyer et al. [BHP14] recently presented an overview on the current state of the art of GPU-based volume visualization. We employ sort-first rendering [MCEF94] to generate images for the tiles of the powerwall (see Figure 2a), i.e., different sections of the screen are rendered in parallel by different nodes (e.g., [SZF*99, MWMS07]). Early ray termination (ERT) stops the integration for a ray when the opacity exceeds a certain threshold [Lev90]. Empty space skipping uses larger sampling distances along rays if only fully transparent regions are traversed (e.g., [CS94, KSSE05]). Frey et al. [FSME14] explicitly control the render time balancing spatial and temporal errors. Similarly, we manually tune parameters to achieve interactive frame rates while keeping the spatial error low.

Table 1: *Metrics used in our example ray casting scenario, their source (NVML (NVIDIA management library) or volume ray casting (VR)), and a brief description.*

Label	Source	Description
Util	NVML	GPU processor utilization
Samples	VR	Total number of samples
NonEmpty	VR	Samples with non-zero opacity
ERTL/BB	VR	Rays terminated by ERT in %
TD	VR	Iterations wasted by lockstep
Time	VR	Kernel execution time
PSNR	Quality	Peak signal-to-noise ratio
MSSIM	Quality	Multi-scale structural similarity [WSB03]

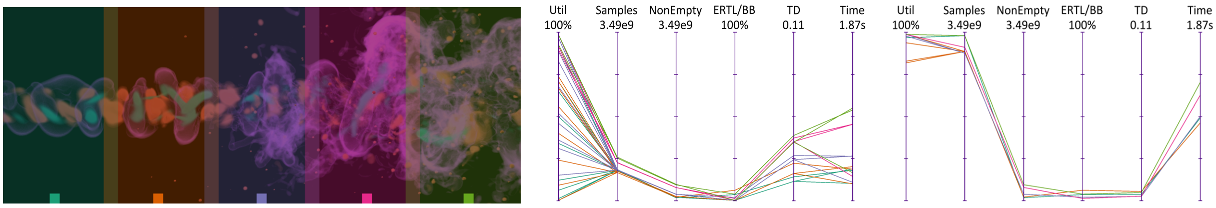
3. Exploratory Performance Tuning

Our motivation is to fine-tune a parallel volume ray caster that runs on the display nodes of our powerwall (see Figure 1) [MRE13]. We utilize sort-first rendering since it is a natural fit for our setup in which each node drives one of five projectors. We use simple empty space skipping by using a multiple of the normal step size to advance along a ray when the latest sample was fully transparent (we used $n = 6$ times the normal step size in our evaluation). For adaptive sampling, when a non-empty sample is obtained along a ray, we go back $n-1$ steps and sample this segment with the normal step size. Rays are terminated early (ERT) when they reach an opacity saturation of 99% or above. We use Blinn-Phong shading with central differences for gradient estimation (e.g., [HLSR09]). We use the parameter *sampleDist* to adjust step size along a ray, and *imageFactor* to scale the image resolution in both directions. A frame lock synchronizes the rendering between the display nodes.

During the volume rendering, we collect metrics from several sources (see Table 1 for the metrics discussed in this paper). GPU performance metrics are queried from NVIDIA's GPU management library. Quality differences between different parameter settings are determined using MSSIM [WSB03] and PSNR. The ray casting kernel was manually instrumented to provide metrics about the volume rendering. Metrics can be aggregated per node or globally.

Presenting the metrics to users is challenging since each node provides many metrics for each frame (the metrics collected for each frame on a node can be seen as a multi-dimensional data point). Furthermore, it is unclear beforehand what phenomena and correlations are interesting. Additionally, a compact visual representation is desirable to minimize occluding the application. Techniques like scatter plots, bar charts, or line charts are thus unfavorable. We chose to use parallel coordinates since they can present many metrics simultaneously. They allow for quick comparison of the metrics between the nodes and outlier spotting. The distinct patterns between dimension axes also help to identify the relations between metrics. Each poly-line in the parallel coordinates plot represents the metrics for one frame. The

© The Eurographics Association 2015.



(a) *Screen-space partitioning.* **(b)** *Adaptive sampling enabled.* **(c)** *Adaptive sampling disabled.*

Figure 2: *Each of the five display nodes of our powerwall covers a certain area that overlaps with its neighbors for smooth blending between projectors. The parallel coordinates plots in Figures (b) and (c) show the metrics of each display node for the view in (a) with and without adaptive sampling, respectively. The lines are colored according to the color overlays in (a).*

metrics can be shown individually on each display node or collectively on one.

Users can interact with the volume rendering as usual, for example by zooming, panning, and rotating. The parallel coordinates plot supports well-known actions like rearranging and scaling the axes and brushing the poly-lines for selection. Users can also move and scale the parallel coordinates plot to minimize occlusion of the volume, or hiding it if not needed. The number of poly-lines per node can be freely chosen by users, but to reduce clutter and overdraw a lower number is advisable; we found a history of 20 frames to be useful. The collected metrics can also be cleared, for example before starting an analysis series.

Post-mortem analysis approaches can offload their metrics without interfering with the application, for example using dedicated threads or external monitoring. While our approach also supports persistent storage of metrics for later analysis, it relies on a swift collection of metrics for real-time analysis. Consequently, the metric collection and visualization should not impose a significant performance overhead. Collecting the metrics and rendering the parallel coordinates plot on one node is in the order of 10 ms for our sort-first approach on five display nodes. With sort-last volume rendering (e.g., object-space partitioning of the volume), often a much larger number of nodes is used. In this case, collecting the metrics is negligible (4 bytes for each metric per frame and node) with respect to the image data that needs to be gathered on the display nodes. Yet, the resulting parallel coordinates plot would be unreadable due to the increased number of nodes providing metrics. Advanced techniques are then required, such as density-based approaches, clustering, or bundling, as well as sophisticated interaction, for example using fisheye lenses. We believe that our approach would be helpful for such larger and more complex setups, but this requires further effort and remains for future work.

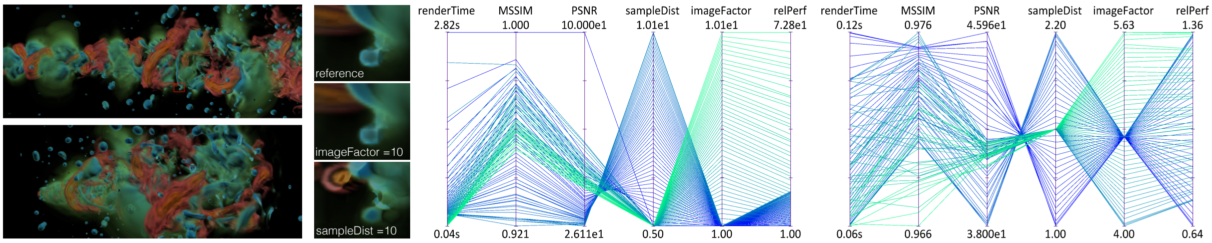
Although our main goal is to tune a visualization system, understanding its performance characteristics is helpful as it allows to better understand and anticipate the impact of certain parameter changes. To that end, we collect numerous metrics (only a subset of which are discussed in this paper). This includes metrics depicting user-perceived experience (e.g., image quality, frames per second) as well as metrics helping users to understand the reasons behind these (e.g., GPU/CPU utilization). These are then analyzed in relation to the directly user-changeable parameter settings. More sophisticated performance analysis (e.g., for finding and eliminating bottlenecks) is beyond the scope of this paper, and would require the consideration of even more metrics, such as network or disc utilization. We believe our approach would be applicable to other problem domains in scientific visualization (e.g., flow visualization) that employ similar parallelization strategies.

4. Results

Our distributed visualization is run on five display nodes (with NVIDIA Quadro 6000 GPUs), each connected to a 4K projector with a resolution of 2400×4096 pixels. The resulting image of 10800×4096 pixels (including blending areas, see Figure 2a) is shown on a powerwall with the size of 6×2.2 meters (see Figure 1) [MRE13]. Users interact with the application and performance visualization on the powerwall through a head node that broadcasts events to the display nodes. For our analysis, we used the Jet data set (720×320×320 voxels) that shows the pressure output from a simulation. We utilize parallel coordinates with a resolution of 1760×1000 pixels on the leftmost display node (see Figure 1) during the evaluation, showing the performance metrics listed in Table 1 in real-time together with the volume rendering. The parallel coordinates plot is cleared at the beginning of every analysis series. We discuss only a subset of all available performance metrics for clarity and brevity.

At the example of adaptive sampling, we first assess the impact of a modification a parameter and the reasons behind a resulting change in performance (see Figure 2). Disabling adaptive sampling (compare (c) to (b)) leads to a uniform GPU utilization (*Util*) and render time distribution (*Time*). While the number of non-empty samples stays the same (*NonEmpty*), the total amount of samples increases significantly (*Samples*). However, as the non-empty and empty space are passed with the same sampling distance along a

© The Eurographics Association 2015.



(a) *Volume from side (w/ closeups) and front.* **(b)** *Performance results for (a) top.* **(c)** *Performance results for (a) bottom.*

Figure 3: *(a) Views of interest for an interactive visualization session. (b)(c) Respective performance results with lines representing different parameter settings (colored chronologically from blue to cyan). While MSSIM and PSNR provide the image quality across the whole powerwall, render time is given for the slowest node only, as this determines the overall performance.*

ray, divergence (*TD*) is significantly reduced. This limits the speedup of GPUs for this optimization technique, as threads with terminated rays idle as long as one thread of that warp still traces a ray (see [NHD10, FRE12] for discussion and approaches to tackle this issue).

We now adjust the sampling rate in image and object space to achieve fluent interaction for views of interest of the volume, while at the same time compromising the resulting image quality as little as possible. We employ two image metrics, *MSSIM* and *PSNR*, to allow the user to estimate the quality of a whole range of parameters at one glance (with *MSSIM* $\in [0,1]$ and *PSNR* $\in [0,\infty)$, where low values indicate high deviation from the reference). Note that such metrics can merely supply indications of perceived quality [dFZS05], in particular for a special setup like a powerwall, thus in the end a human has to judge the quality. *MSSIM* and *PSNR* are full reference metrics, i.e., before adjusting the respective parameters and evaluating their quality impact, we need to take a reference image with high-quality settings (one ray per pixel with step size 0.5 along a ray). With our setup, this took approximately three seconds.

We start with a typical side view of the data set (see Figure 3a *(top)*) and individually test a range of settings for the parameters *imageFactor* and *sampleDist*, the image space and object space sampling distance. The corresponding parallel coordinates plot (see Figure 3b) clearly shows the trade-off between render time and quality. We aim to render the volume at 20 FPS (i.e., the slowest node may take 50 ms to render). To achieve this, going from top to bottom in the parallel coordinates plot (high to low quality), we look for values of *sampleDist* and *imageFactor* that yield similar image quality, and inspect the speedup that can be achieved with regard to the reference time of 2.82 seconds (*relPerf*) (a total speedup ≈ 56 is required here to yield our target render time). Since the impact of these two parameters on the speedup is largely independent, their speedup can be assumed roughly multiplicative. For the selected view, this is achieved with *sampleDist*=1.6 and *imageFactor*=4.75 resulting in *MSSIM*=0.9653 and *PSNR*=42.4. The image factor can be chosen comparably large as the resolution of the powerwall significantly exceeds the (projected) resolution of this volume data set (note that the sampling distance is already given relative to the size of a voxel).

We now achieve 20 FPS for this camera configuration, and continue our exploration to other views of interest. In the view of Figure 3a *(bottom)*, the interaction performance significantly drops to render times of around 80 ms. Thus, we further refine our parameter settings based on this view and evaluate variations of our current parameter settings (see Figure 3c). Based on this plot and using the same quality metric-oriented approach as before, we now adjust our parameter settings to *sampleDist*=2 and *imageFactor*=5.5, to yield 50 ms again. Our approach allows this speedup to be achieved with only a minor decrease in rendering quality (*MSSIM*=0.9739, *PSNR*=41.5 to *MSSIM*=0.9697, *PSNR*=38.9). Checking back with the first view we tested, it is now rendered in 30 ms with only minor decreases in quality with respect to our previously determined parameter settings (*MSSIM*=0.9625 and *PSNR*=41.3).

5. Summary and Future Work

We presented an exploratory approach to analyzing and tuning an interactive parallel volume visualization on a large display. We collect metrics from the GPU and the ray caster and determine the image quality during user interaction with the volume rendering. The metrics are visualized as parallel coordinates plot in real-time to facilitate the joint exploration of the data set as well as the visual and performance impact of different parameter settings. We discussed this approach in the context of parameter tuning and performance analysis. For future work, we plan to extend our approach to other problem domains and to conduct an expert study. We also intend to explore how our approach applies to comparison of ensembles and classification of performance phenomena.

Acknowledgements

This work was partially funded by the Federal Ministry of Education and Research of Germany (BMBF) as part of the FeToL project.

References

[BHP14] BEYER J., HADWIGER M., PFISTER H.: A Survey of GPU-Based Large-Scale Volume Visualization. In *Eurographics/IEEE Conference on Visualization State-of-the-Art Reports* (2014). 2

[CS94] COHEN D., SHEFFER Z.: Proximity clouds — an acceleration technique for 3d grid traversal. *The Visual Computer 11*, 1 (1994), 27–38. 2

[dFZS05] DE FREITAS ZAMPOLO R., SEARA R.: A comparison of image quality metric performances under practical conditions. In *IEEE International Conference on Image Processing* (2005), vol. 3, pp. III–1192–5. 4

[FRE12] FREY S., REINA G., ERTL T.: SIMT Microscheduling: Reducing Thread Stalling in Divergent Iterative Algorithms. In *Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing* (2012), pp. 399–406. 4

[FSME14] FREY S., SADLO F., MA K.-L., ERTL T.: Interactive Progressive Visualization with Space-Time Error Control. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2397–2406. 2

[HE91] HEATH M., ETHERIDGE J.: Visualizing the performance of parallel programs. *IEEE Software 8*, 5 (1991), 29–39. 2

[HLSR09] HADWIGER M., LJUNG P., SALAMA C. R., ROPINSKI T.: Advanced Illumination Techniques for GPU-based Volume Raycasting. In *ACM SIGGRAPH 2009 Courses* (2009), pp. 2:1–2:166. 2

[IGJ*14] ISAACS K. E., GIMÉNEZ A., JUSUFI I., GAMBLIN T., BHATELE A., SCHULZ M., HAMANN B., BREMER P.-T.: State of the Art of Performance Visualization. In *Eurographics/IEEE Conference on Visualization State-of-the-Art Reports* (2014). 2

[KSSE05] KLEIN T., STRENGERT M., STEGMAIER S., ERTL T.: Exploiting frame-to-frame coherence for accelerating high-quality volume raycasting on graphics hardware. In *IEEE Visualization* (2005), pp. 223–230. 2

[Lev90] LEVOY M.: Efficient Ray Tracing of Volume Data. *ACM Transactions on Graphics 9*, 3 (1990), 245–261. 2

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

[MRE13] MÜLLER C., REINA G., ERTL T.: The VVand: A Two-Tier System Design for High-Resolution Stereo Rendering. In *CHI POWERWALL 2013 Workshop* (2013). 2, 3

[MWMS07] MOLONEY B., WEISKOPF D., MÖLLER T., STRENGERT M.: Scalable Sort-First Parallel Direct Volume Rendering with Dynamic Load Balancing. In *Eurographics Symposium on Parallel Graphics and Visualization* (2007), pp. 45–52. 2

[NHD10] NOVÁK J., HAVRAN V., DACHSBACHER C.: Path Regeneration for Interactive Path Tracing. In *Eurographics 2010 Short Papers* (2010), pp. 61–64. 4

[SLB*11] SCHULZ M., LEVINE J., BREMER P.-T., GAMBLIN T., PASCUCCI V.: Interpreting Performance Data across Intuitive Domains. In *2011 International Conference on Parallel Processing (ICPP)* (2011), pp. 206–215. 2

[SZF*99] SAMANTA R., ZHENG J., FUNKHOUSER T., LI K., SINGH J. P.: Load balancing for multi-projector rendering systems. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (1999), pp. 107–116. 2

[WG11] WYLIE B. J. N., GEIMER M.: Large-scale performance analysis of PFLOTRAN with Scalasca. In *Proceedings of the 53rd Cray User Group meeting, Fairbanks, AK, USA* (2011). 2

[WSB03] WANG Z., SIMONCELLI E., BOVIK A.: Multiscale structural similarity for image quality assessment. In *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers* (2003), vol. 2, pp. 1398–1402 Vol.2. 2

© The Eurographics Association 2015.