Volume Rendering with advanced GPU scheduling strategies

Institute for Computer Graphics and Vision

Graz University of Technology

Philip Voglreiter[1], Markus Steinberger[1], Rostislav Khlebnikov[1], Bernhard Kainz[2], Dieter Schmalstieg[1]

[1] Institute for Computer Graphics and Vision, Graz University of Technology
[2] Department of Computing, Imperial College London

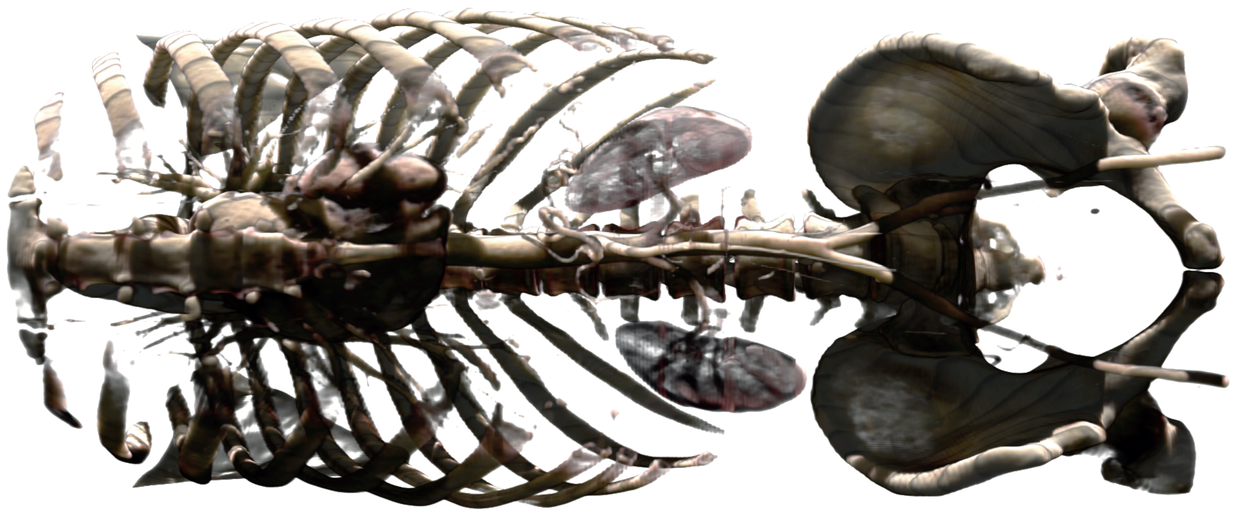

Figure 1: MECANIX rendered with scheduled IPSVI

Introduction

With advanced scheduling strategies on GPUs, we circumvent certain bottlenecks occuring in volume rendering. Low resource utilization, multi-pass approaches and thread divergence were analyzed for three different algorithms. By introducing adaptions and new techniques, we were able to achieve speedups of several magnitudes.

GPU Scheduling

Figure 2: Bonsai with IPSVI

We use Softshell [1], a generic framework enabling scheduling mechanisms on GPUs. Its capabilities, such as regrouping threads in more coherent blocks and spawning work from within GPU execution, are ideal for many volume rendering tasks.
However, since there is no hardware support for these mechanisms, it introduces a considerable computational overhead. Therefore, related parameters require to be fine-tuned for a given task.
For evaluation, we use a CUDA implementation of the generic design and measure the performance of volume rendering algorithms by directly comparing standard CUDA vs Softshell implementations.

Scheduled Ray Casting

GPU ray casting [2] with early ray termination suffers from thread divergence in the sampling loop. We check every **L** iterations whether the number of coherent threads drops below a percentage **P** and trigger ray re-convergence. We use two settings: one with **L**=20 and **P**=0.7 (RCVG) and a more aggressive one with **L**=1 and **P**=0.95 (A-RCVG). Both increase hardware utilization considerably, but the overhead influences gained performance. Figure 3 shows the increase in utilization. Generally, the divergence in ray casting is not high enough to drastically increase render speed. Figure 4 depicts the re-convergence process as observed during ray-casting. Active threads are aggregated into full blocks, while finished threads and corresponding empty blocks vanish.

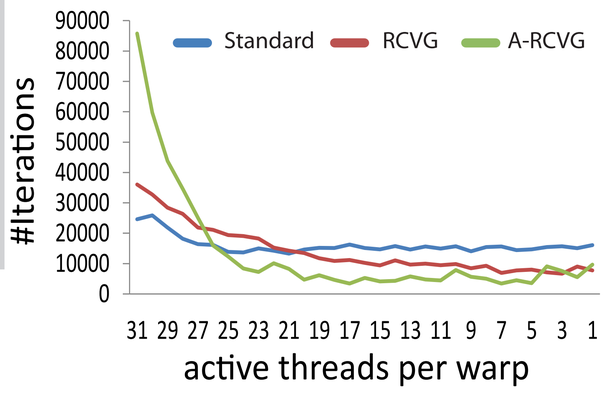

Figure 3: thread coherence for ray casting: Bonsai

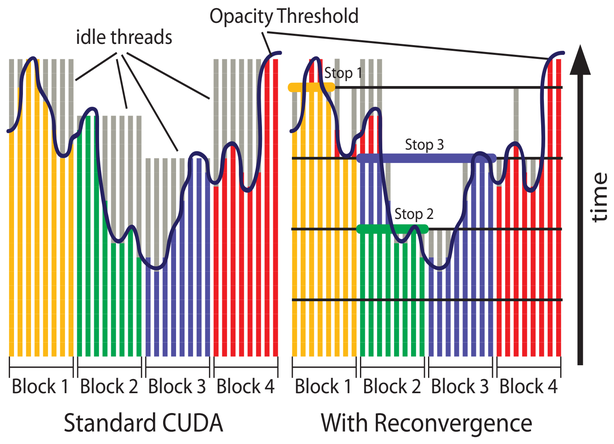

Figure 4: idle threads (left, gray) block hardware resources we free by re-converging blocks containing only active threads. The four colours denote current assignment to a working block.

Scheduled Image Plane Sweep Volume Illumination

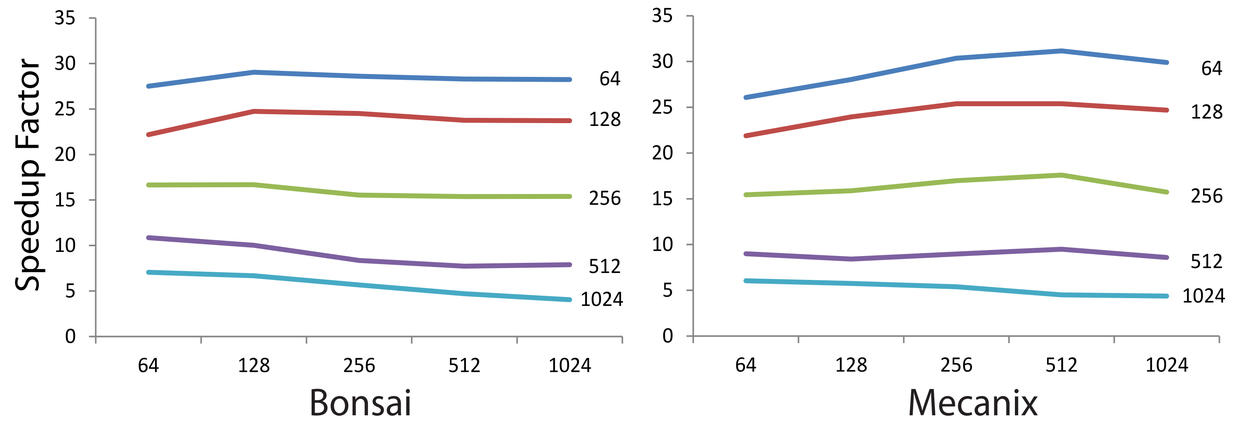

Figure 5: Speedups observed of scheduled IPSVI over plain CUDA. The highest speedups were achieved with low resolution scanlines, while we could maintain a speedup of >4 even for higher resolutions. The x-axis shows the screen resolution (closest to) orthogonal to the light direction, while lines denote resolutions parallel to it.

Image plane sweep volume illumination (IPSVI) [3] performs single scattering via depth shadow maps in an iterative approach. A scan line sweeps over the image, introducing ping-pong computation between CPU and GPU. With Softshells capabilities of launching work from within GPU execution, we efficiently utilize hardware resources without interruption. On top, we parallelize the sampling process of single rays to increase the general resource utilization. Figure 4 shows the speedup measured for the Bonsai dataset while Figure 5 depicts the same measurement for Mecanix. We observe a tremendous speedup which is, for low resolutions, mostly accounted to the increased parallelism. For high resolutions, where generally higher utilization is expected, our persistent rendering approach still maintains a speedup of more than 4 as compared to the non-scheduled version.

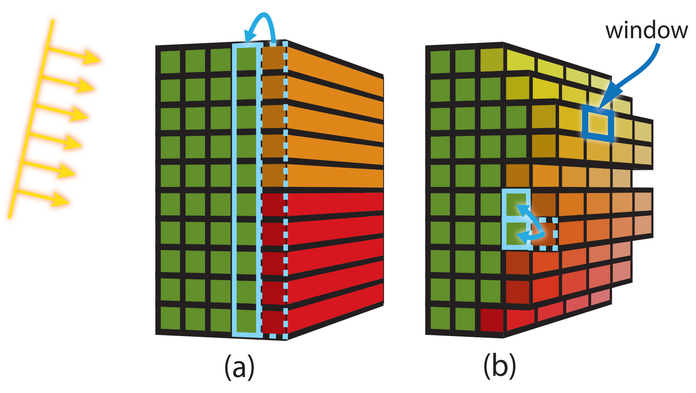

Figure 6: Standard IPSVI in 6(a) with synchronized ray front and dependencies. 6(b) shows our version with an arbitrarily shaped scan `line' and sample windowing.

Scheduled Particle Based Volume Rendering

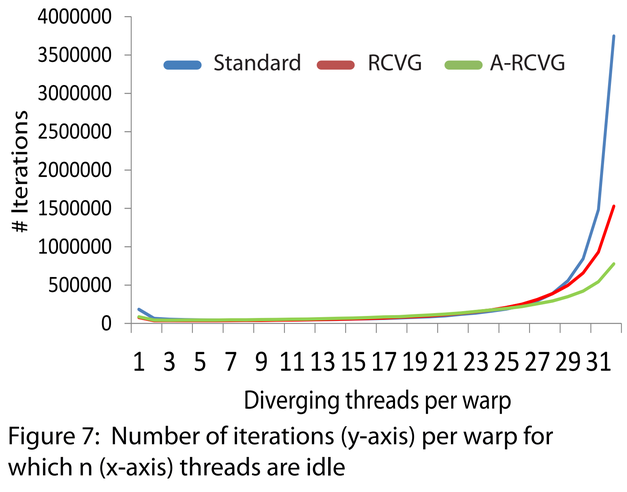

Figure 7: Number of iterations (y-axis) per warp for which n (x-axis) threads are idle

For irregular grids, object order approaches, such as PBVR [4][5], often are the method of choice. This particular approach generates a particle field whose local density depends on both size and optical properties of tetrahedra. Due to vastly differing particle counts in single cells, tremendous thread divergence is inevitable. Again, re-convergence strategies help alleviating this issue. We examine two test cases: a tetrahedralized version of the regular Stanford Dragon (worst case) as well as an inherently irregular simulation of a radio frequency ablation (intended case). We again observe a drastic increase in utilization, but as opposed to ray casting, we achieve speedups of up to 13 despite Softshell's overhead.

Conclusion

We have shown that the currently rigid GPU computing model does not allow for optimal resource utilization in several categories of volume rendering algorithms. However, scheduling strategies advancing over the state of the art can tremendously increase performance of algorithms. Increased hardware support for functionalities such as thread re-convergence would decrease the impact of the induced overhead and even further increase the speedup as compared to the currently necessary software implementation of GPU scheduling techniques.

Contact:
Philip Voglreiter: voglreiter@icg.tugraz.at
Markus Steinberger: steinberger@icg.tugraz.at
Rostislav Khlebnikov: khlebnikov@icg.tugraz.at
Bernhard Kainz: b.kainz@imperial.ac.uk
Dieter Schmalstieg: schmalstieg@icg.tugraz.at

Acknowledgements:
This work was supported by the Austrian Science Fund (P23329) and the European Union (ICT-2011.5.2 600641).
Bernhard Kainz was supported by a Marie Curie Intra-European Fellowship within the 7th. European Community Framework Programme (FP7-PEOPLE-2012-IEF F.A.U.S.T. 325661).

References:
[1] M. Steinberger, B. Kainz, B. Kerbl, S. Hauswiesner, M. Kenzel, and D. Schmalstieg. Softshell: dynamic scheduling on gpus. ACM Trans. Graph., 31(6):161:1–161:11, 2012.
[2] M. Levoy. Efficient ray tracing of volume data. ACM Trans. Graph., 9(3):245–261, 1990.
[3] E. Sundén, A. Ynnerman, and T. Ropinski. Image Plane Sweep Volume Illumination. IEEE TVCG(Vis Proceedings), 17(12):2125–2134, 2011.
[4] B. Csébfalvi and L. Szirmay-Kalos. Monte carlo volume rendering. In Proc. of IEEE Visualization, pages 449–456, 2003.
[5] N. Sakamoto, J. Nonaka, K. Koyamada, and S. Tanaka. Particle-based volume rendering. In Visualization, 2007. APVIS, pages 129 –132, 2007.