Demand-Driven Volume Rendering of Terascale EM Data

Johanna Beyer, Markus Hadwiger
KAUST

Won-Ki Jeong, Hanspeter Pfister, Jeff Lichtman
Harvard University

1 Introduction

In neuroscience, a very promising bottom-up approach to understanding how the brain works is built on acquiring and analyzing electron microscopy (EM) scans of brain tissue, an area known as *Connectomics*. This results in volume data of extremely high resolution of 3-5nm per pixel and 25-50nm slice thickness, overall leading to data sizes of many terabytes [Jeong et al. 2010]. To support the work of neurobiologists, interactive exploration and analysis of such volumes requires novel visual computing systems because the requirements differ from those of current systems in several key aspects. In this talk, we describe the system that we are working on to enable neuroscientists to interactively roam terascale EM volumes and support their analysis. A major design principle was to avoid the standard approach of pre-computing a 3D multi-resolution hierarchy such as an octree. Data acquisition proceeds from 2D image tile to 2D image tile, where not only the slices along the z axis are scanned independently, but each slice is itself acquired as many smaller image tiles. These images tiles need to be aligned and stitched, and neurobiologists also want to be able to combine different resolutions used for scanning different regions, without re-sampling everything to a single global resolution. Therefore, we focus on working directly with a stream of individual 2D image tiles, instead of a 3D volume that usually is assumed to exist in its entirety for visualization. We perform interactive volume rendering of a "virtual" volume, where the corresponding physical storage is only represented and populated in a sparse manner with 2D instead of 3D image data on the fly during rendering. Furthermore, these 2D image tiles can be of different resolution, scale, and orientation.

2 Demand-Driven Volume Rendering

In order to enable 3D volume rendering in the described scenario, we use a GPU ray-caster that renders a "virtual octree". The physical data contained in this octree are only computed on demand, driven by the actual visibility and the current screen-projection size of the volume data. These parameters are detected on the fly during ray-casting and drive the actual data reconstruction from a stream of 2D image tiles, which is performed by a flexible *demand-driven data back-end* that runs decoupled from the ray-caster in a separate background thread. The back-end can also accommodate data produced by continuously scanning microscopes in a progressive fashion while rendering. The ray-caster drives data reconstruction in a *display-aware* fashion, by detecting *cache misses* of virtual octree nodes that are required for front-to-back traversal but are not yet resident in the GPU cache. This is illustrated in Figure 1. If a cache miss cannot be satisfied from the larger cache in CPU memory, the physical data of matching resolution are reconstructed on the fly in the background, while rendering proceeds at interactive rates. Determining the octree nodes for reconstruction in this manner significantly bounds the actual cache working set required for volume rendering because even high display resolutions are much smaller than our data sets. In this way, the required resolution for reconstruction in the data back-end is determined directly by the output display and is independent of the volume resolution itself.

To achieve interactive frame rates, we use a multi-level paged virtual memory management system for reconstructed octree nodes as well as image tiles, both in GPU memory as well as in CPU memory. In between these cache levels, we page data on demand. In addition to rendering a virtual octree that is populated only on demand, another major difference to other octree volume rendering

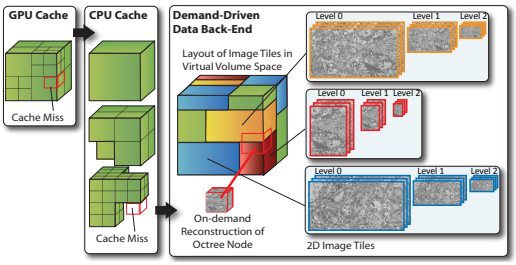

Figure 1: *Demand-driven reconstruction for sparsely populating the virtual octree from individual 2D image tiles during rendering.*

systems [Gobbetti et al. 2008; Crassin et al. 2009] is that we use a simpler and faster scheme for volume traversal, which builds on arbitrary direct access to the full virtual octree in virtual memory. During ray-casting, all sample positions are computed in virtual volume space, which are translated on the fly through a two-level page table hierarchy. Only a very small top-level page directory is fully resident in GPU memory. The current working set of the 2nd-level page table resides in a small page table cache, whose entries further refer to physical octree nodes, only a small subset of which is resident in the physical cache texture. Performing look-ups in this page table hierarchy is extremely fast when spatial coherence along rays is exploited. Whenever a required octree node is not physically present, the ray-caster generates a cache miss. We use a small size of 32^3 physical voxels for each octree node, which achieves very good culling and update granularity. However, our system still scales to very large volumes because of the page table hierarchy. For example, we can render a 1TB volume with a top-level page table of 48 KB, a 2nd-level page table cache of 32 MB, and a physical texture cache with sizes from 1 GB on a single GPU to a total combined size of several gigabytes on multiple GPUs.

We currently render three large EM volumes acquired by the Harvard Center for Brain Science: mouse cortex ($21,494 \times 25,790 \times 1,850 = 955$ GB), and two mouse hippocampus volumes (43 GB and 92 GB, respectively). Rendering is fully interactive and hides the latency of data reconstruction, because the ray-caster can substitute missing data with data of lower resolution and reconstruction proceeds simultaneously in the background. In the background, getting a 3D block of size $512 \times 512 \times 32$ from the demand-driven data back-end can be as fast as 30ms when the required 2D image tiles are already in memory, but may take up to a few seconds if all images still have to be read from disk. However, interactive roaming is always possible, which is crucial for neurobiologists. We are currently focusing on reducing the latency in the back-end itself.

References

CRASSIN, C., NEYRET, F., LEFEBVRE, S., AND EISEMANN, E. 2009. GigaVoxels : Ray-Guided Streaming for Efficient and Detailed Voxel Rendering. In *Proceedings of 2009 Symposium on Interactive 3D Graphics and Games*, 15–22.

GOBBETTI, E., MARTON, F., AND GUITAN, J. 2008. A Single-pass GPU Ray Casting Framework for Interactive Out-of-core Rendering of Massive Volumetric Datasets. *The Visual Computer 24*, 7, 797–806.

JEONG, W.-K., BEYER, J., HADWIGER, M., BLUE, R., LAW, C., VASQUEZ, A., REID, C., LICHTMAN, J., AND PFISTER, H. 2010. SSECRETT and NeuroTrace: Interactive Visualization and Analysis Tools for Large-Scale Neuroscience Datasets. *IEEE Computer Graphics and Applications 30*, 3, 58–70.

SIGGRAPH 2011, Vancouver, British Columbia, Canada, August 7 – 11, 2011.
ISBN 978-1-4503-0921-9/11/0008