GPU-BASED FRAMEWORK FOR DISTRIBUTED INTERACTIVE 3D VISUALIZATION OF MULTIMODAL REMOTE SENSING DATA

Martin Lambers and Andreas Kolb

Institute for Vision and Graphics, University of Siegen, Germany
{lambers,kolb}@fb12.uni-siegen.de

1. INTRODUCTION

Interactive visualization of remote sensing data requires both interactive navigation in very large data sets and interactive adjustment of visualization parameters. For example, interactive adjustment of dynamic range reduction methods for SAR amplitude images can reveal or accentuate details that might otherwise be overlooked [1, 2]. Furthermore, interactive combination of multiple data sets from different sensors benefits tasks such as comparison, interpretation, and quality assessment.

The programmable graphics processing units (GPUs) of todays commodity graphics hardware provide the computational power that is necessary to interactively process, combine, and render remote sensing data [3].

In this paper, we present techniques for a GPU-based interactive 3D visualization framework for remote sensing data that allows interactive navigation, interactive adjustment of visualization parameters, and interactive combination of multimodal data sets. Height information for terrain visualization is extracted from digital elevation models (DEMs). The terrain is textured using an interactively defined combination of data sets from imaging sensors, e.g. aerial photographs or SAR amplitude images. The framework can be used on a wide variety of systems, from laptops with modern graphics cards, desktops, and workstations, to high resolution display walls and virtual environment systems driven by render clusters.

2. METHODS

2.1. Data Management

Remote sensing data sets are often representable as 2D rectangles that span a subset of the WGS84 earth map. Examples are DEMs, aerial photographs, or SAR images. To handle the large amounts of data generated by high-resolution remote sensing techniques, a hierarchical data structure is necessary. Variants of quadtrees are commonly used for this task [4]. We use a tile tree, which is a quadtree variant that is a generalization of the tiling pyramid used in [3]. See Fig. 1.

Each level of a tile tree represents a WGS84 map of the whole earth. The top level always consists of a single tile. Each higher level quadruples the number of tiles. Since all tiles have the same number of pixels, this corresponds to a duplication of the resolution. With a tile size of 512×256 pixels, level 16 roughly corresponds to a ground coverage of 1m^2 per pixel, and level 26 to 1mm^2 per pixel. The resolution of a given data set will lie between to tile tree levels, as shown in Fig. 1. To build the tile tree, the original data set is first resampled to the next higher tile tree level, using a magnification factor between 1 and 2. The required tiles of a lower level in the tile tree can be computed by combining tiles from the next higher level. Areas of a tile which are not covered by the given data set are marked with a special value.

Each tile is extended with a border containing additional data set information. This border is not part of the core area of the tile as represented by the tile tree and is not displayed, but allows data processing methods to work on local neighborhoods without the need to access neighboring tiles [3].

The tile tree is stored in a directory tree that allows retrieval of individual tiles via standard network protocols.

2.2. Process Management

The visualization framework encapsulates the information that is necessary to render the current scene in render states, and the OpenGL context and data management in render contexts. The main application process manages the master render state. Each render context manages its local copy of the render state and can render a different view of the current scene.

In a common desktop environment, there is only one render context (provided by the graphical user interface) which renders a single view of the scene. For high resolution display walls or virtual environment systems, multiple render contexts can exist on multiple graphics processing units (e.g. on visualization workstations) and/or on multiple hosts (e.g. the nodes of a render cluster).

The distribution of render contexts and the synchronization of the render state is handled by the Equalizer middleware for parallel rendering [5].

Each render context uses multiple tile fetcher threads to retrieve and cache tiles from the active data sets. This allows asynchronous and progressive retrieval of tiles for continuous rendering and navigation.

2.3. Processing and Rendering

Tiles are prepared for rendering by a data type dependant GPU-based processing chain that applies the interactively selected visualization methods and parameters. An example processing chain for SAR amplitude images, including despeckling and dynamic range reduction, is described in [3]. Similar processing chains can be built for other data types.

To render the 3D scene, a triangle mesh for the currently visible section of the earth is recursively refined on the GPU using the available DEM data set tile trees. This is achieved using geometry shaders, which are computational units introduced by the latest generation of GPUs.

The GPU's fragment shaders compute color information for each output pixel by combining results of the processing chains of chosen data set tile trees (e.g. an aerial photograph and a SAR amplitude image).

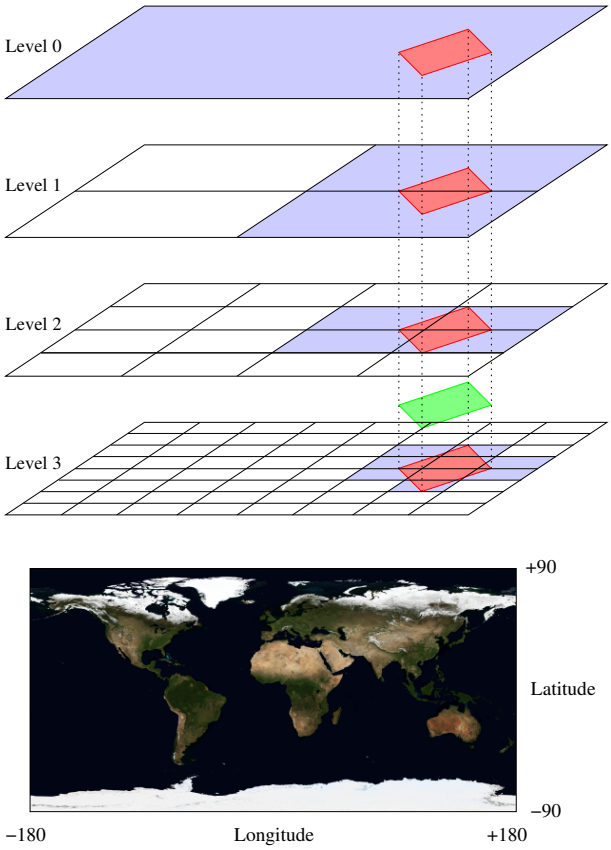

Fig. 1. Tile tree (top) for a data set that spans a subset of the WGS84 earth map (bottom). The original data set is marked green. Required tiles are marked blue.

3. RESULTS

We are currently integrating implementations of the techniques described above to build a GPU-based interactive 3D visualization framework for remote sensing data that allows interactive navigation, interactive adjustment of visualization parameters, and interactive combination of multimodal data sets. Utilizing the full power of current GPUs is key to achieve this goal. The framework is designed to work both on single-display computers and on visualization systems driven by multiple render nodes and/or graphics cards.

4. REFERENCES

[1] M. Lambers, H. Nies, and A. Kolb, "Interactive Dynamic Range Reduction for SAR Images," *Geoscience and Remote Sensing Letters*, vol. 5, no. 3, pp. 507–511, 2008.

[2] M. Lambers and A. Kolb, "Adaptive Dynamic Range Reduction for SAR Images," in *Proc. 7th European Conference on Synthetic Aperture Radar (EUSAR)*, 2008, vol. 3, pp. 371–374.

[3] M. Lambers, A. Kolb, H. Nies, and M. Kalkuhl, "GPU-based framework for interactive visualization of SAR data," in *Proc. Int. IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, July 2007.

[4] R. Pajarola and E. Gobbetti, "Survey of semi-regular multiresolution models for interactive terrain rendering," *Vis. Comput.*, vol. 23, no. 8, pp. 583–605, 2007.

[5] S. Eilemann, "The Equalizer parallel rendering framework," `http://www.equalizergraphics.com/`.