Interactive Rendering and Efficient Querying for Large Multivariate Seismic Volumes on Consumer Level PCs

Liang Zhou*  
SCI Institute and the School of Computing,  
University of Utah

Charles Hansen†  
SCI Institute and the School of Computing,  
University of Utah

* e-mail: zhoul@cs.utah.edu  
† e-mail: hansen@cs.utah.edu

Figure 1: Our proposed method allows interactive visualization and efficient query of large multivariate seismic datasets on consumer level PCs. Shown here is a test seismic data of six attributes with size: 1278 × 1653 × 1704.

ABSTRACT

We present a volume visualization method that allows interactive rendering and efficient querying of large multivariate seismic volume data on consumer level PCs. The volume rendering pipeline utilizes a virtual memory structure that supports out-of-core multivariate multi-resolution data and a GPU-based ray caster that allows interactive multivariate transfer function design. A Gaussian mixture model representation is precomputed and nearly interactive querying is achieved by testing the Gaussian functions against user defined transfer functions on the GPU in the runtime. Finally, the method has been tested on a multivariate 3D seismic dataset which is larger than the size of the main memory of the testing machine.

Keywords: Multivariate volume, Out-of-core Methods, Transfer Functions

1 INTRODUCTION

Due to the advance in 3D seismic imaging techniques, resolution of 3D seismic datasets used in petroleum industry are usually of giga-bytes. Multiple attributes derived from the original seismic amplitude dataset have been used for aiding the interpretation of the seismic survey. With these additional attributes however, the size of the entire dataset may become far larger than the capacity of the GPU memory and even the main memory of a typical workstation.

Recently, GPU-based multi-resolution out-of-core volume rendering systems have been proposed. The Gigavoxel approach by Crassin et al. [1] and CERA-TVR by Engel [2] divide the dataset into multi-resolution bricks and utilize an octree structure and ray guided paging system to efficiently render large sparse volume datasets. Hadwiger et al. [3] propose a rendering system which uses a virtual memory system and 2D mip-mapping to support dense and noisy peta-scale microscopy scans. However, none of these methods are able to handle large multivariate volume datasets.

In this work, we extend the approach by Hadwiger et al. to support interactive rendering of large multivariate seismic datasets on a consumer level PC. On the other hand, data value querying raises another challenging issue for multivariate datasets especially when the data is very dense as hierarchical acceleration techniques, e.g. octrees may not be beneficial. As such, we propose an efficient data querying technique based on precomputed per-block Gaussian mixture models and run-time ellipse-polygon intersection detection. An interactive exploration system has been built to allow the user to visualize the multivariate volumes as well as to edit multivariate transfer functions with the query feedback on parallel coordinate plots.

2 MULTIVARIATE OUT-OF-CORE VOLUME RENDERING

Since the seismic datasets are dense and noisy, a virtual memory based volume rendering framework is suggested more efficient as shown in [3]. As such, we adopt the framework proposed by Hadwiger et al. and extend it to support interactive multivariate volume rendering. The virtual memory structure is extended to store multivariate data blocks, and the ray caster is modified to support multivariate transfer functions (TFs).

2.1 Virtual Memory Structure for Multivariate Volumes

We share the same virtual memory hierarchy as in the work of Hadwiger et al., namely, in a top-down manner: page table directory, page table and block caches. The difference is that instead of storing a single scalar volume in the block cache, we store data of all attributes at a given block location contiguously in the block cache. The page table entries are set to point to the beginning of the first attribute of each block. When the volume renderer makes paging
We empirically choose the number of Gaussians to be three as it describes the distribution using GMM which is very compact in computational space. Assuming the datasets follow Gaussian distribution, we are able to conduct the query using ellipse-polygon intersection detection: i.e. if any part of the ellipse intersects with the TF polygon in any 2D sub-space of the \( n \times 1 \) 2D TF space, all values in the distribution are selected. Ellipse-polygon intersection is hard in the original space, and as such we compute a circle-triangle intersection in a transformed space. It is known that the ellipse can be transformed from a circle using matrix \( \Sigma^{1/2} \) which is the square root matrix of matrix \( \Sigma \) which holds the eigen vectors of the ellipse. Therefore, the ellipse can be transformed back to a circle using the inverse matrix \( \Sigma^{-1/2} \). The 2D polygon can be triangulated, and the triangles that form the polygon can also be transformed using \( \Sigma^{1/2} \) into the circle’s space, and then a much easier circle-triangle intersection test can be performed. Consequently, the query result is rendered using PCP by transforming the data values inside the Gaussian blobs to lines in the PCP.

## 3 Efficient Multivariate Query

To allow efficient data query on the noisy seismic datasets, we propose a two-stage approach that utilizes Gaussian Mixture Model (GMM) to compactly approximate the multidimensional distribution of data. The method first computes GMM for each block in a pre-computation stage and then tests ellipse-polygon intersection in runtime to query data values for voxels selected by user defined TFs. The per-block GMM is required to compute only once for a dataset and the runtime querying achieves near interactive performance.

### 3.1 Per-block Gaussian Mixture Model Computation

Assuming the datasets follow Gaussian distribution, we are able to describe the distribution using GMM which is very compact in terms of storage. GMMs are computed using the well-known expectation maximization algorithm, and we pre-compute GMMs for each block at its finest resolution for only once. In the same fashion of our TF space as described in Section 2.2, we compute \( n \times 1 \) 2D GMMs. The computation is performed using CUDA thrust library and the result is written to a file which records the mean value and covariance matrix for each Gaussian distribution for each block. We empirically choose the number of Gaussians to be three as it strikes a balance between the closeness of GMM approximation of the original distribution and the compactness of storage.

## 3.2 Runtime Ellipse-Polygon Intersection Test

During visualization, the system queries data values for user defined TFs on the GPU with the GMM information stored as a texture. For any given pair of attributes, each Gaussian distribution is a 2D ellipse while each user defined TF is a 2D polygon. As such, we are able to conduct the query using ellipse-polygon intersection detection: i.e. if any part of the ellipse intersects with the TF polygon in any 2D sub-space of the \( n \times 1 \) 2D TF space, all values in the distribution are selected. Ellipse-polygon intersection is hard in the original space, and as such we compute a circle-triangle intersection in a transformed space. It is known that the ellipse can be transformed from a circle using matrix \( \Sigma^{1/2} \) which is the square root matrix of matrix \( \Sigma \) which holds the eigen vectors of the ellipse. Therefore, the ellipse can be transformed back to a circle using the inverse matrix \( \Sigma^{-1/2} \). The 2D polygon can be triangulated, and the triangles that form the polygon can also be transformed using \( \Sigma^{1/2} \) into the circle’s space, and then a much easier circle-triangle intersection test can be performed. Consequently, the query result is rendered using PCP by transforming the data values inside the Gaussian blobs to lines in the PCP.

## References