EG UK Theory and Practice of Computer Graphics (2011)
Hamish Carr, Ian Grimstead (Editors)

Enabling Visualization of Massive Datasets Through MPP Database Architecture

A. Al-Naser[†1] , M. Rasheed[‡1,2] , J. Brooke[§1] and D. Irving[¶1,2]

[1]The University of Manchester, UK [2]Teradata

Abstract

We are developing a novel visualization architecture which is specifically designed to render very large (terabyte scale) datasets. Our method differs from the classic visualization pipeline of Harber and McNabb. In particular we eliminate the need to create geometric objects, for example surfaces composed of polygons, as a stage before rendering. Such objects require specialist HPC servers for their creation and manipulation; our solution eliminates the need for such servers. We replace the geometric objects by structures stored and tagged in a database next to the original dataset; we call these Spatially Registered Data Structures (SRDS). Such structures are linked to a single rendering pipeline through the on-the-fly creation of a Feature Embedded Spatial Volume (FESVo). This solution exploits recently developed capabilities of in-database Massive Parallel Processing (MPP) and parallel data streaming, together with the rapidly developing capabilities of GPUs. We describe an early prototype of an architecture applied to seismic data from the oil and gas industry.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture—Parallel processing I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types, Standards I.3.8 [Computer Graphics]: Applications—

1. Introduction

In this work we describe a novel method for visualizing spatial features in very large (O(10TB)) datasets. The classical visualization pipeline described by Haber and McNabb in 1990 [HM90] involves the creation of geometric objects as an intermediate in the transformation from data to rendered visual image. We illustrate problems that this causes in the visualization of very large datasets by considering the case of isosurfaces, since these are important objects in the visualization of features in geospatial data. In our study case of seismic data, *isosurfacing* is used for visualizing as well as extracting and picking seismic related surfaces, mainly horizons and faults [MR04]. The marching cube technique is commonly used for generating triangle mesh of isosurface objects. However, when dealing with a massive volume, issues with polygonal representation are encountered [DGY07]: mainly the generation of a large number of polygons is computationally expensive, often requiring a parallel computing cluster; this geometric object is then rendered in the final stage of the pipeline.

The size of the geometric object makes interactive visualization impossible without techniques for reducing the fraction that has to be rendered. Thus complexity reduction techniques are applied. Two essential techniques are visibility culling and level-of-details (LOD) [DGY07]. LOD reduction technique provides the least required resolution based on field of view; e.g. the closer field of view is the higher resolution is fetched. Thus a paradox emerges: we create a very large object which we then have to reduce in order to view it interactively. In this paper we describe an architecture for visualization that removes the need for the creation and storage of such intermediate geometric objects. This also eliminates the need for the deployment of large parallel servers and enables visual exploration of very large datasets on desktop resources.

As a use case to illustrate our method we focus on

† E-mail: aqeel.al-naser@cs.manchester.ac.uk
‡ E-mail: masroor.rasheed@postgrad.manchester.ac.uk
§ E-mail: john.brooke@manchester.ac.uk
¶ E-mail: duncan.irving@teradata.com

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG11/109-112

terabyte-scale seismic datasets, serving the the oil and gas industry. Our approach locates the feature detection algorithms and attribute computations directly on a Massively Parallel Processing (MPP) database where the data is stored. The dataset in the database is about 40% more than in its original format (SEG Y) due to binary-ASCII metadata conversion. We maintain depth data as binary array. Derived attributes are computed on-the-fly instead of traditional approaches were they are stored exclusively. Thus, no extra data storage in the database.

We visualize data using minimal and parallel point-to-point call queries to retrieve data that is tagged in the database as belonging to a particular feature[†]. Tagging can be linked to physically stored data or on-the-fly derived data. In the case of seismic data, tagging is occurring at an array level (traces). We describe how the architecture delivers such features to programmable GPU cards where we utilise direct volume rendering techniques to display the features. Despite the focus on a case study of seismic data, our work is generic for any spatial data type.

† A feature is a subset of derived attributes or user marked dataset.

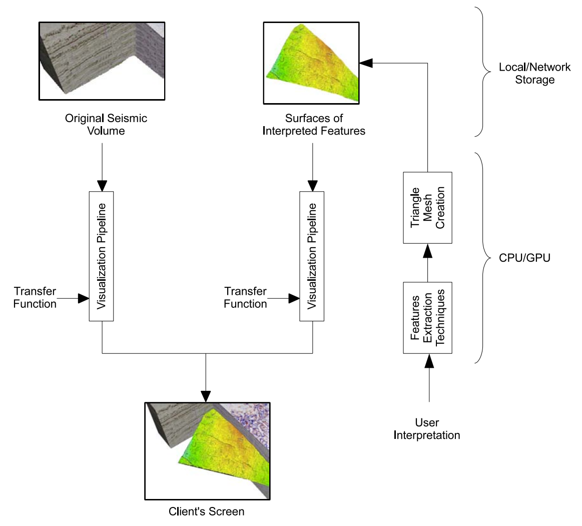

Figure 1: *An abstract view on the current seismic visualization solution. Features are stored and processed independent from the original volume.*

2. Seismic Visualization in the Oil and Gas Industry

Exploration for oil and gas starts with analysing and interpreting seismic data. To acquire seismic data, acoustic waves are artificially generated on the earth's surface and reflects from subsurface geological layers and structures. Due to the variation of material properties, these waves are reflected back to the surface and their (1) amplitudes and (2) travel time are recorded via receivers [MR04]. This data is then processed to generate a 2D or 3D image illustrating the subsurface layers. A 2D seismic profile consists of multiple vertical *traces*. Each *trace* holds the recorded amplitudes sampled at, typically, every four milliseconds. Seismic imaging is then interpreted by a geo-scientist to extract geological features such as horizons and faults. This interpretation potential identifies hydrocarbon traps: oil or gas. Due to the continuous demand on hydrocarbon resources, geo-scientists are seeking efficient visualization of large seismic datasets.

The SEG Y [The] format has been used by the industry to store seismic data since mid 1970s. Data in a SEG Y file is stored sequentially and therefore retrieval of seismic data for a 3D visualization could negatively affect interactivity. For this reason, seismic visualization and interpretation applications, such as Petrel [Sch], offer an internal format which stores seismic data in multi-resolution bricks for fast access. However, interpreted surfaces such as horizons and faults are represented, stored and rendered in separate objects; see Figure 1. These objects are rendered often as polygons which require displayed points to be kept in memory. Thus handling a large volume with several interpreted objects still requires a high end machine for the end user.

3. Feature Aware Visualization

As described in the Introduction, the classic visualization pipeline presents the need to create geometric objects which can be comparable to, or larger than, the size of original data. Such objects cannot fit in the memory of desktop or mobile devices. Two popular solutions to this problem are remote geometry delivering and remote rendering (as described by J. Ge [Ge07]). In both solutions resources (visualization servers) are scaled up in proportion to the size of a dataset since it is visualized classically. Moreover, remote rendering solutions require a connection of at least 100 Mbps[‡] [SFNC09] between client and server in order to achieve a good frame rate that is higher than 20 fps[§]. In addition, because the application is running on the server and only images are being streamed, a continuous connection is required all the time as users can perform no action offline. Solutions to this have been proposed [GMC*02], however they do not address the problems of tera-sized data.

Our alternative approach replaces geometric objects with what we call "features". A feature is a subset of the original data, appropriately tagged to indicate that it belongs to a particular spatial feature. Thus, instead of a surface being considered as being composed of polygons (geometric approach) it is now a subset of the data, appropriately tagged.

‡ Mbps = megabit per second
§ fps = frames per second

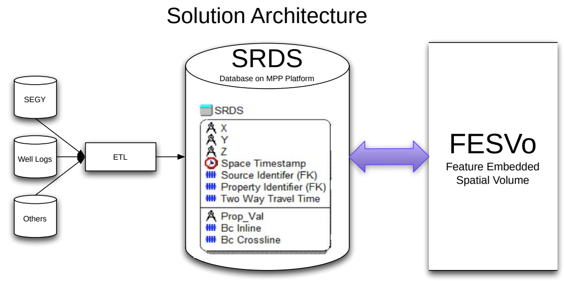

Figure 2: *An abstract illustration of the proposed solution architecture. (**ETL**: Extract, Transform and Load, **SRDS**: Spatially Registered Data Structures)*

We then associate visualization methods with a feature. This approach was first described in the context of data from oceanographic simulations [BMPS07], where novel visualization techniques exploiting the (then) recently available programmable GPUs.

With the rapid advances in the capabilities of GPUs, direct volume rendering techniques such as *3D texture slicing* and *GPU-based ray casting* have become more efficient for interactive visualization on a large uniform grid. Recent advancements of utilising multi-core compute units under Open Computing Language (OpenCL)[¶] and direct modification of voxel through shading languages such as OpenGL Shading Language (GLSL)[‖] allow us to render on desktop the objects we call Feature Embedded Spatial Volumes (FESVo). In the next section we describe our proposed visualization architecture to enable this for a wide range of applications.

4. Proposed Solution Architecture

In this section we propose a complete architecture from data representation to the visualization of massive spatial datasets as illustrated in Figure 2. The architecture is an alternative to the classic visualization pipeline; it dispenses with the need to create geometric objects. Instead, we organize and tag the data in the database to create structures called Spatially Registered Data Structures (SRDS). This allows the possibility of creating and editing SRDSs using parallel capabilities on the database itself. Such parallel capabilities are created to process very large datasets from the commercial sector (e.g. CRM data), we utilise this capability to eliminate the need for a parallel HPC cluster.

¶ `http://www.khronos.org/opencl/`

‖ `http://www.opengl.org/documentation/glsl/`

4.1. Data Preparation and Parallelism

We illustrate the first stage of creating an SRDS via our use case, which involves both seismic (large volume and lower resolution data) and oil well logs (low volume and higher resolution) data. All data is converted into a single coordinate system (WGS84) and data quality checks are performed. The data is then broken into the smallest voxel, gridded to appropriate finite resolution and tagged with a spatial property. The data is distributed on a *primary accessor key*[**] by using a hashing algorithm. This newly formatted data is referred to as SRDS which is later utilised for run-time feature extraction and analysis, and parallel streaming query for visualization.

SRDS primary storage is derived from a Cartesian coordinate system [MS61] and adds dimensions of time, property, projection and metadata integrator. SRDS extensions [CH05] allows for subtype polymorphism which enables data to be re-constructed. For example, `Value.Horizons()` would extracts only those voxels that comply with static definition of Horizon. An extension can be designed based on parametrized calls. SRDS allows data structures to be derived from voxel as extensions. Each SRDS tuple is a work unit which enables data processing to be parallel and distributed on massively parallel processing (MPP) database.

4.2. Feature-Embedded Spatial Volume (FESVo)

Classically, features are stored and rendered independently from the original seismic data, as illustrated in Figure 1. By associating feature-aware parallel queries (further explained in the next subsection) to SRDS, we can create in real-time a Feature-Embedded Spatial Volume (FESVo) being ready for direct volume rendering.

We are currently developing methods to directly render raw seismic data (seismic amplitude) with its interpreted features in a single pipeline. The FESVo comprises a volume dataset that contains the feature identified by the SRDS and it is this volume that is visualized. To render the SRDS feature, we propose to directly modify voxel colour property using GLSL. The shared spatial location of raw data and its features would allow us to perform such modification on the fly. As more data arrives from the SRDS, the FESVo is updated and rendered continuously; the more data arrives the better quality appears. This is a true streaming visualization as there is no pre-defined volume size.

4.3. Querying and Feature Awareness

An important gain of our architecture is that we can use database information extraction queries to assist the user of

** The primary accessor key is composed of X and Y which identifies a geographical 2D location.

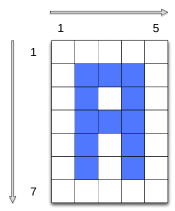

Figure 3: *A simple example illustrating 35 parallel SQL point-to-point calls to the RDBMS but only 12 will return data.*

the system to identify features. Querying can be performed on any of the attributes presented at SRDS. To exploit parallelism as the data flows through the components of the architecture, querying is performed in structure of general steps. We illustrate this process by the simple example shown in Figure 3.

1. Discovery—reads metadata information to obtain information of bounding area: The bounding dimensions are determined from metadata in SRDS. A feature in 2D with bounding box (1,1) to (5,7) and incremental of 1 in both directions.
2. Submit a *parallel point-to-point call* query based on field of view—This will allow 7*5 = 35 parallel SQL calls to the RDBMS[††]. In this case, only 12 will return data. *Point-to-point calls* are used instead of *range elements* to avoid full table scans (look for this feature from the entire datasets) and creation of index to improve performance. *Hashing algorithm* on primary accessor keys provides information of the exact location of the data element. Hence without index creation, high through-put for streaming is achievable in theory.
3. Apply data processing before returning data to Visualization—Virtual Processing elements (providing the extension for SRDS) are attached to each data piece or a collection of tuples with or without range.
4. Create FESVo on-the-fly and continuously render as data arrives—FESVo's field of view dictates what data is requested in parallel while it dynamically renders as information is constantly arriving.

5. Current Status and Future Work

We reviewed the evolution of the class visualization pipeline and state-of-the art rendering techniques and used this as a basis for a new approach. We highlight the following:

- We conclude that to achieve high performance and efficient visualization of massive spatial datasets we need an architecture which starts from the data representation level.
- We propose a complete solution architecture of (1) a Spatially-Registered Data Structure (SRDS) and (2) a single rendering pipeline relying on creating a Feature-Embedded Spatial Volume (FESVo) on the fly.
- SRDS is partially deployed. We are currently testing a *geological horizon* feature extraction method; more feature extractions are to be implemented and tested. At the same time, we have started the implementation of FESVo concept; it is currently in its early stage. We have tested querying and rendering seismic slices; further enhancements are to be implemented. Later, we will be working on querying the features in SRDS along with neighbouring original seismic data to create FESVo on the fly, and render it in a single pipeline.
- The architecture is to be evaluated in different ways. This would include: (1) evaluating the interactivity rate which could be measured by calculating rendered frames per second and (2) measuring potential of reduction on the data being transferred over the network to indicate the mobility of the architecture.

[††] RDBMS: relational database management system

References

[BMPS07] BROOKE J. M., MARSH J., PETTIFER S., SASTRY L. S.: The importance of locality in the visualization of large datasets. *Concurrency and Computation: Practice and Experience 19*, 2 (Feb. 2007), 195–205. 3

[CH05] CAROMEL D., HENRIO L.: *A Theory of Distributed Objects*. Spring-Links, 2005. 3

[DGY07] DIETRICH A., GOBBETTI E., YOON S.-E.: Massive-Model Rendering Techniques: A Tutorial. *IEEE Computer Graphics and Applications 27*, 6 (2007), 20–34. 1

[Ge07] GE J.: *A Point-Based Remote Visualization Pipeline for Large-Scale Virtual Reality*. PhD thesis, University of Illinois at Chicago, 2007. 2

[GMC*02] GLENCROSS M., MARSH J., COOK J., DAUBRENET S., PETTIFER S., HUBBOLD R.: Distributed interactive virtual prototyping. In *Sketches and Applications Programme* (San Antonio, Texas, 2002), SIGGRAPH. 2

[HM90] HABER R., MCNABB D.: Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. *Visualization in Scientific Computing* (1990), 74–93. 1

[MR04] MA C., ROKNE J.: *3D Seismic Volume Visualization*, vol. VI. Springer Netherlands, Norwell, MA, USA, 2004, ch. 13, pp. 241–262. 1, 2

[MS61] MOON P. H., SPENCER D. E.: *Field theory handbook : including coordinate systems, differential equations, and their solutions*. Springer-Verlag, 1961. 3

[Sch] SCHLUMBERGER: Petrel Seismic to Simulation Software. http://www.slb.com/services/software/geo/petrel.aspx. 2

[SFNC09] SASTRY L., FOWLER R., NAGELLA S., CHURCHILL J.: Supporting Distributed Visualization Services for High Performance Science and Engineering Applications A Service Provider Perspective. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (2009), IEEE Computer Society, pp. 586–590. 2

[The] THE SOCIETY OF EXPLORATION GEOPHYSICISTS:. http://www.seg.org/. 2