A Survey of Client-Server Volume Visualization Techniques

Lazaro Campoalegre and John Dingliana
Graphics, Vision and Visualisation Group, Trinity College Dublin
campoall@tcd.ie | john.dingliana@scss.tcd.ie

January 26, 2018

Abstract

This state-of-the-art report provides a comprehensive review of the research on client-server architectures for volume visualization. The design of such schemes capable of dealing with static and dynamic volume datasets has been an important challenge for researchers due to the need for the reduction of information transmitted. Thus, compression techniques designed to facilitate such systems are a particular focus of this survey. The ever increasing complexity and widespread use of volume data in interdisciplinary fields, as well as the opportunities afforded by continuing advances in computational power of mobile devices are strong motivations for this review. In particular, the client-server paradigm has particular significance to medical imaging due to the practical advantages and increased likelihood of use, of portable low-spec clients in lab and clinical settings.

1 Introduction

The term client-server refers to a computer applications paradigm comprising a highly-resourced server device, providing services typically to one or more lesser-powered clients that receive and request information over a network [1,2]. Client-server architectures have been popular for many years, since personal computers (PCs) became viable alternatives to mainframe computers. A large number of client-server techniques have been published in the scientific visualization literature. The motivations for these include facilitating remote exploration, distributed environments, collaborative multi-user systems and many others. In visualization, as in other fields, a high-powered graphical system provides visual computing services to low-performance clients, which might consist of mobile devices or desktop computers [3] [4]. The server device typically features more powerful central processors, graphical processing units (GPUs), more memory, or larger disk drives than the clients.

1.1 Client-server visualisation in medicine

With the continuing advancement of medical imaging techniques, it has become possible for specialists to obtain highly detailed and accurate information of the anatomical internal structures of the human organism. By leveraging different visualization techniques, experts can now obtain suitable images of bones, soft tissues, and the bloodstream, amongst other features. Computer visualization systems have become able to generate images with increasingly better resolution and information accuracy. Standards such as DICOM (Digital Imaging and Communication in Medicine) facilitate portability and manipulation of large volume sets, easing visualization, interaction and interpretation of models.

Recently, several important research areas in three-dimensional techniques for multimodal imaging have emerged. Applications include neurological imaging for brain surgery [5], tissue characterization, medical education, plastic surgery, surgical simulators [6] and others. At the same time, scientists are more familiarized with three-dimensional structures reconstructed from bi-dimensional images, and are able to use these to important practical benefits. Visualization of damaged tissues and tumors can help, for instance, in the treatment of patients with oncological pathologies. A key use in chemotherapy is to know whether a tumor is growing or shrinking. The application of current visualization algorithms can improve the the ability to highlight such pathologies during medical examination.

Furthermore, hospitals are becoming increasingly interested in tele-medicine and tele-diagnostic solutions. Tele-medicine [7] [8] is defined as the use of medical information exchanged from one site to another via electronic communications to improve the clinical health status of patients. This concept includes a growing variety of applications and services using two-way video, email, smart phones, wireless tools and other forms of telecommunications technology. Clinically oriented specialities can capture and remotely display physical findings, transmit specialized data from tests and carry out interactive examinations [9]. Tele-medicine, in turn, facilitates tele-diagnosis, the process whereby a disease diagnosis, or prognosis, is enabled by the electronic transmission of data between distant medical facilities. Some applications for remote visualization of medical images and 3D volumetric data, such as MRI or CT scans, could be categorized as forms of tele-medicine.

Remote visualization has become a topic of significant interest in recent years [10–12], however, for large volume datasets, interactive visualization techniques require last generation graphics boards, due to the intensive calculation and memory requirements of 3D rendering. Client-server approaches are a significant means of allowing such functionalities but the handling of three-dimensional information requires efficient systems to achieve fast data transmission and interactive visualization of high quality images.

In particular, there is still a scarcity of specific bibliography for volume visualization on mobile devices. Frequently, the use of mobile devices is necessary and desirable in practice due to their portability and ease of maintenance. However, transmission time for the volumetric information combined with low performance hardware properties make it quite challenging to design and implement efficient visualization systems on such devices. In order to address these issues and generally compensate for the limitations of low performance devices or to reduce costs, a large number of client-server schemes have been proposed.

1.2 Scope and Objectives

In this paper, we present a detailed survey of the *State of the Art* in client-server volume visualization techniques. We begin with an extensive review, in Section 2, of the main categories of client-server architectures that have been used in volume visualization. The throughput from high-powered server to remote client invariably requires reduction in the bandwidth of information transmitted, thus it is important to consider the compression techniques that facilitate most client-server communications. In Section 3 we present a detailed review of *Static Volume Compression Techniques*. A study of the specialized case of *Dynamic Volume Compression* schemes is presented in Section 4. Finally, we present a survey of recent *Volume Rendering Techniques for Mobile Devices* in section 5. We concluding with insights and observations resulting from our study in Section 6.

2 Client-Server architectures for volume visualization

An extensive survey of the published literature allowed us to identify four principal categories of client-server architecture for volume visualization. Essentially, we classify them according to the means by which they reduce volume information for transmission, and discuss each category in the subsections below.

A. Transmission of compressed volumes: In the first category (see Figure 1), the dataset is compressed on the server and transmitted to the client, where the data is decompressed, the transfer function applied and the reconstructed data is rendered. We also include, in this category, approaches that transmit the full volume to the client, without compression.

Callahan *et al.* [13] presented an isosurface-based method for hardware assisted progressive volume rendering. Their approach seeks to minimize the size of the data stored in the final client at each step of data transmission. The approach sends a compressed vertex array to the client during the reconstruction of the model. Moser and Weiskopf [14] proposed a 2D texture-based method which uses a compressed texture atlas to reduce interpolation costs. The approach proposes a hybrid high/low resolution rendering, to combine volume data and additional line geometries in an optimized way. By doing this, they achieve interactive frame rates. The technique runs on a mobile graphics device natively without remote rendering. Mobeen *et al.* [15] proposed a single-pass volume rendering algorithm for the WebGL platform. They built an application with a transfer function widget which enables feature enhancement of structures during rendering. To avoid 3D texture limitations of some devices, they mapped the volume into a single 2D

texture to develop an application able to run in any modern device with a basic graphics processor. A recent application developed by Rodriguez *et al.* [16] allows interaction with volume models using mobile device hardware. Their scheme does not in fact compress the volume data, but they are able to apply different transfer functions to volumes by selecting the most appropriate 2D, 3D, or ray-casting method best suited to available hardware capabilities.

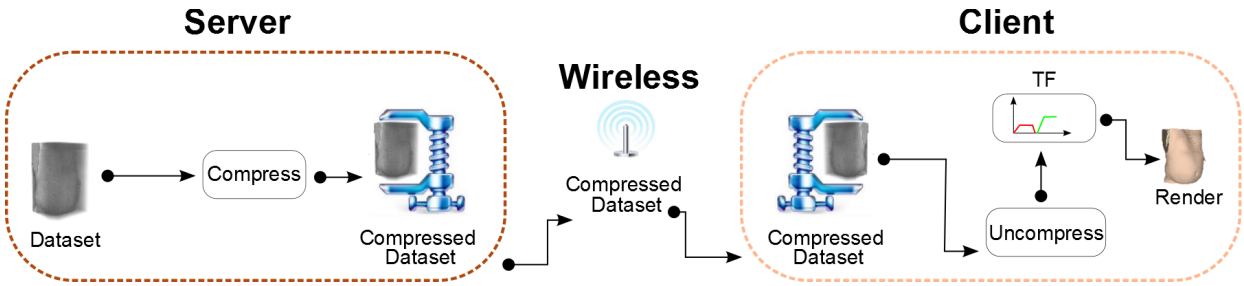

Figure 1: Client-Server Architecture, Case A: the dataset is sent to the client in a compressed or uncompressed way. The client applies the transfer function after decompression and before rendering.

B. Transmission of compressed 2D rendered images: In a number of schemes, where the transmitted data is a compressed image [17] [18] (see Figure 2), the transfer function is applied at the beginning of the pipeline, following which the volume is rendered to a 2D texture, all on the server side. A compressed image is then sent to the client, where decompression and image rendering take place. This approach is frequently referred to as ***Thin Client*** [19]. Other techniques included in this category are discussed below.

Engel *et al.* [20] developed an approach based on the Open Inventor toolkit, which provides a scene graph programming interface with a wide variety of 3D manipulation capabilities. Their application renders images off-screen, encodes images on-the-fly and transmits them to the client side. Once on the client, images are decoded and copied into a framebuffer. The client interface also provides a drawing area with mouse event handling capabilities to display images.

A new remote visualization framework is proposed in [21], where the dataset is loaded into a slicing tool on the client side. The tool allows axial, coronal and sagittal direction inspections of medical models. The application allows the selection of a sub-region by using object-aligned textures. Volume data is transferred to the server side to increase visualization quality. In a similar way to other techniques, the server first renders images off-screen, compresses the image and transmits the result to the client. Once on the client side, the image is decompressed and rendered. Mouse and GUI events are sent to the server for re-rendering operations. Qi *et al.* [22] designed a medical application to send images in a progressive way. Their approach creates a reference image of the entire data by applying transforms. The encoding scheme allows the gradual transmission of the encoded image, which is reconstructed on-the-fly during the rendering on the client side.

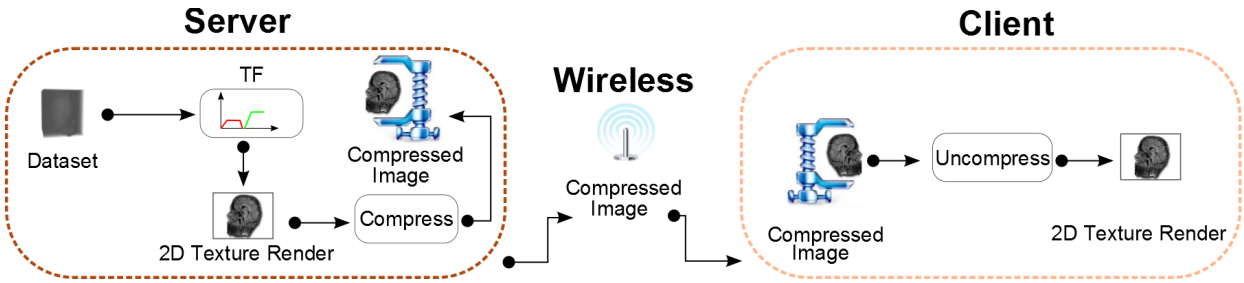

Figure 2: Client-Server Architecture, Case B: the transfer function is applied in the server, which also renders the volume data. The information sent to the client consists on compressed 2D images.

Constantinescu *et al.* [23] implemented an application that incorporates Positron Emission Tomography/Computer Tomography (PET/CT) data into Personal Health Records for remote use on internet-capable or handheld devices. Their client-server application is designed to display images in low-end devices such

as mobile phones. Users can control brightness and contrast, apply a color look-up table and view the images from different angles. The approach allows the transmission of images with a sufficient refresh rate to achieve interactive exploration of 2D images. Jeong and Kaufman [24] implemented a virtual colonoscopy application over a wireless network with a Personal Digital Assistant (PDA) as a client device. In their scheme, the server performs a GPU-based direct volume rendering to generate an endoscopy image during the navigation, for every render request from the client.

An explorable images technique was proposed by Tikhonova *et al.* [25]. The approach converts a small number of single-view volume rendered images of the same 3D data set into a compact representation. The mechanism of exploring data, consists of interacting with the compact representation in transfer function space without accessing the original data. The compact representation is built by automatically extracting layers depicted in composite images. Different opacity and color mappings are achieved by the different combination of layers.

C. Partitioning the volume data: Some approaches achieve a reduction of transmitted information by partitioning the rendered volume (see Figure 3). Partitions are transmitted to the client, where the composition of the entire volume takes place and the volume is rendered. For instance, Bethel. [26] proposed to subdivide and render volumes in parallel. The resulting set of 2D textures is sent to a viewer which uses 2D texture mapping to render the stack of textures, and provides the facility for interactive transformations.

Distributed volume rendering approaches are considered by scientists to be an important means of dealing with memory constraints and other hardware limitations of a standalone display system. Some authors have been able to exploit the advantages of distributed processing to improve interactive volume visualization by implementing complex client-server mechanisms. For instance, Frank and Kaufman [**?**], developed a technique to render massive volumes in a volume visualization cluster. By partitioning the volume to be rendered into synchronized clusters, they are able to reduce the memory requirements allowing them to deal with large datasets such as the full Visible Human dataset. Bethel et al. [**?**] proposed a distributed memory parallel visualization application that uses a sort-first rendering architecture. By combining an OpenRM Scene Graph and Chromium they achieve an efficient memory sorting algorithm that competently performs view-dependent parallel rendering in a first-order architecture.

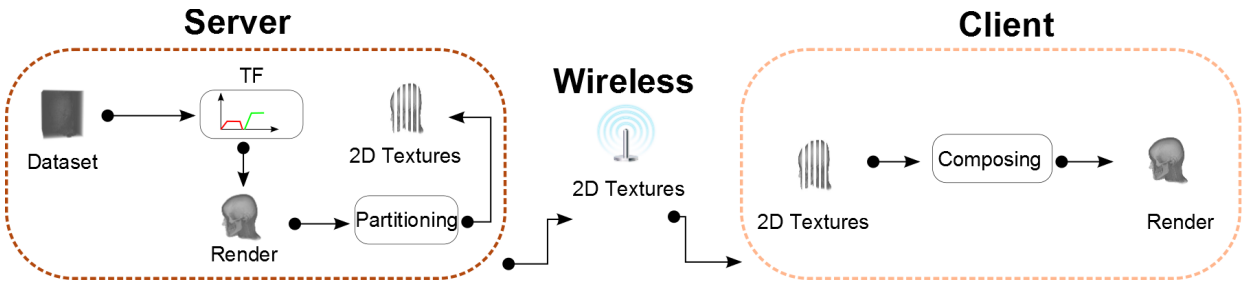

Figure 3: Client-Server Architecture, Case C: the server partitions the volume into a set of 2D slices, represented as 2D textures. The information sent to the client consists on a stack of 2D textures.

D. Sending compressed multiresolution volume information: In some approaches, data pre-processing ensures the reduction of the information, combined with different techniques for quantization, encoding and multiresolution representation.

In this category of approaches, a networking application is proposed by Lippert *et al.* [27]. Here, a local client with low computational power browses volume data through a remote database. The approach allows the treatment of intensity and RGB volumes. A Wavelet based encoding scheme produces a binary output stream to be stored locally or transmitted directly to the network. During rendering, the decoded Wavelet coefficients are copied into the accumulation buffer on the GPU. The bandwidth of the network and the frame rate control the transmission of the Wavelet coefficients in significance order to guarantee rendering quality.

Boada *et al.* [28] proposed an exploration technique where volume data is maintained in the server in a hierarchical data structure composed of nodes. The server receives the user parameters to select the correct list of nodes to be rendered in the client side according to its hardware capabilities. As a second rendering

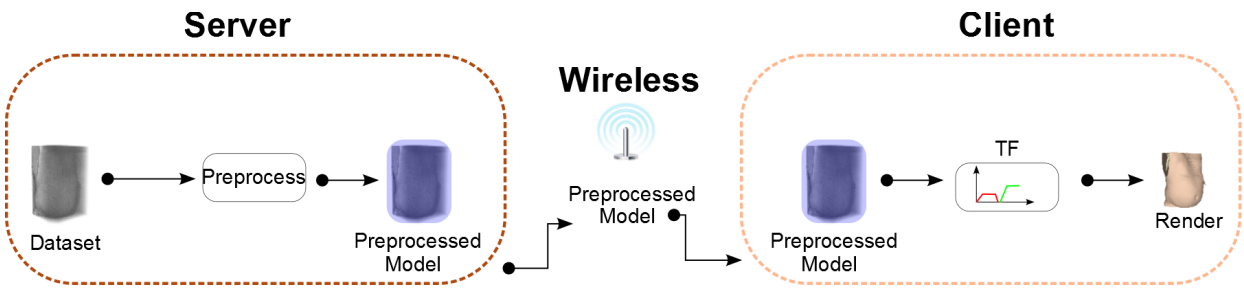

Figure 4: Client-Server Architecture, Case D: the server enriches the volume dataset by computing a multiresolution volume hierarchy, which is then compressed and sent to the client.

possibility, the user can select a region of interest (ROI) of the entire volume. To achieve this, the server transmits data in an incremental fashion. Recently Gobbetti *et al.* [29], proposed a progressive transmission scheme to send compact multiresolution data from the server to the client stage. The technique allows fast decompression and local access to data according to the user interaction in the client side.

Summary

Table 1 shows a comparison of several client-server oriented proposals for volume rendering. The columns show, for each approach, the category of solution (as discussed above), the rendering technique employed, and the form of data transmitted between server and client. We also indicate where a mobile device is used as client, and provide an estimation (Low, Medium or High) of latency and interactivity of each approach, based on its reported *fps* (frame per second).

A simple analysis of this table, shows that few techniques were designed to run on mobile client. Although some techniques successfully achieve visualization on mobile devices, the limited size of the models and the lack of advanced lighting and shading implementation leaves a gap for further research in this area.

Latency and interactivity are strongly associated concepts in client-server architectures for volume visualization. The ability of achieving interactive frame rates depends on the transmission procedure and the rendering algorithm implemented in both servers and clients. Some techniques achieve a good combination of these properties by applying progressive transmission schemes [13] and adaptive rendering algorithms [14, 29]. But unfortunately these techniques are still quite complex to run on low-end devices.

3 Compression Techniques for Static Volume Models

Dataset transmission from server to clients is considered a very important stage in client-server architectures for volume visualization. Efficient schemes require optimized algorithms to reduce the data and to send them through the network. The algorithms must achieve the maximum compression possible while allowing efficient decompression in the client side, where sometimes hardware and memory constraints decrease performance.

Compression algorithms can be classified into lossless and lossy [32, 33] techniques. With lossless compression, information that was originally in the file is fully recovered after the data has been uncompressed. On the other hand, lossy compression schemes reduce data by permanently eliminating certain information, especially redundant information.

Wavelets and Vector Quantization are popular techniques for approaches in which decompression takes place on the CPU. Wavelet transforms offer considerable compression ratios in homogeneous regions of an image while conserving the detail in non-uniform ones. The idea of using 3D Wavelets for volume compression was introduced by Muraki [34]. One of the limitations of this approach is the cost of accessing individual voxels. In [35] a lossy implementation of the $3D$ wavelet transform was applied to a real volume data generated from a series of 115 slices of magnetic resonance images (MRI). By applying a filtering operation three times, the approach obtains a multiresolution representation of a volume of 128^3 voxels. Using the coefficients from the Wavelet functions, they reconstructed a continuous approximation of the

Table 1: **Comparison of published client-Server architectures**. Columns show the category of client-server architectures (as in Figures 1...4), the rendering algorithm, the form of data transmitted clients, the compression scheme (Table 2) and whether the approach has been applied to mobile clients. The final columns provide an estimation of latency and interactivity for each technique.

					Latency			**Interactivity**		
Ref.	**Arch.**	**Rendering Algorithm**	**Data sent**	**Mobile**	**L**	**M**	**H**	**L**	**M**	**H**
[26]	C	2D Texturing	Images	×	✓				✓	
[28]	D	3D Texturing	Vertices, Indexes	×		✓		✓		
[13]	A	Iso-Surface	Vertices	×			✓			✓
[23]	B	2D Texturing	Images	✓		✓		✓		
[21]	B	2D Texturing	Images	×		✓			✓	
[20]	B	3D Texturing	Images	×		✓			✓	
[29]	D	RC	Octree Nodes	×	✓					✓
[24]	B	2D-Texturing	Images,Points	✓		✓			✓	
[27]	D	2D Texturing	Wavelet Coeff.	×	✓				✓	
[14]	A	2D Texturing	Intensities	✓	✓					✓
[15]	A	2D Texturing, RC	2D Texture Atlas	✓		✓			✓	
[30]	B	2D Texturing	Images	✓	✓				✓	
[16]	A	2D,3D , RC	Intensities	✓	✓				✓	
[25]	B	2D Texturing	Images	✓	✓				✓	
[22]	B	2D Texturing	Images	✓	✓			✓		
[31]	B	2D Texturing	Images	×	✓					✓

original volume at maximum resolution. The rendering technique prevents an interactive scheme, due to the cost of finding the intersection point of the ray with a complex 3D function, and consumes a considerable amount of time. Ihm and Park [36] proposed an effective 3D 16^3-block-based compression/decompression wavelet scheme for improving the access to random data values without decompressing the whole dataset.

Guthe *et al.* [37] proposed a novel algorithm that uses a hierarchical wavelet representation in an approach where decompression takes place on the GPU. The wavelet filter is locally applied and the resulting coefficients are used as the basic parameters for a threshold quantization based scheme. During rendering, the required level of the wavelet representation is decompressed on-the-fly and rendered using graphics hardware. The scheme allows the reconstruction of images without noticeable artifacts.

Current bottlenecks of wavelet based volume compression schemes are the lack of locality and the complexity of the decompression in low-end devices. Moreover, almost all present approaches are compress the whole volume, even in cases where the transfer function forces most of the medical structures to become invisible.

In the first group of approaches (see *Decomp. Stage: In CPU*, in Table 2) most of the implementations are lossy due to the application of quantization/encoding schemes (see also Table 3). Nguyen *et al.* [38] proposed a block based technique to compress very large volume data sets with scalar data on a rectilinear grid. By, working in the wavelet domain and using different quantization step sizes, the technique encodes data at several compression ratios. Although they ensure that compared to similar proposals their approach achieves better reconstruction quality, the resulting images show the existence of small blocking artifacts due to the block based coder. Furthermore, they can only perform two compression steps with limited multiresolution capabilities.

Many methods try to maintain genuine volumetric data during the quantization stage. In contrast Rodler [39] proposes, instead, to treat two dimensional slices in position or time and draw on results developed in the area of video coding. The first step of their encoder removes the correlation along the z-direction, assuming that two-dimensional slices are divided along this direction. A 3D Wavelet decomposition should be ideal to further remove correlation in the spacial and temporal directions. But in order to decrease

computational costs, they adopt, as a second step, a 2D Wavelet transform to handle the spacial redundancy. Finally, the quantization continues by removing insignificant coefficients to make the representation even sparser. The method is capable of providing high compression rates with fairly fast decoding of random voxels. They achieve high compression rates with notable cost in the decompression speed. The approach in [22] works on 3D medical image sets, but it is ultimately a 2D visualization scheme performed on the CPU. The approach has also been restricted to MRI datasets with relative large distance between slices. Although the authors describe the approach as a lossless compression scheme, the averaging and thresholding operations applied on the images result in a reduction in accuracy of the information. Although working with slices has the advantage that the memory format is identical to that of the final 3D texture used for rendering, this comes at the cost of losing spatial coherence.

Vector Quantization [40] is one of the most explored techniques for volume compression. Essentially, this involves decreasing the size of volumetric data by applying a specific encoding algorithm. The premise of this lossy compression method is to code values from a multidimensional vector space into values of a discrete subspace of lower dimension. Ning and Hesselink [41] were the first to apply vector quantification to volume models. In their scheme, the volume dataset is represented as indexes into a small codebook of representative blocks. The approach is suitable for a CPU-based ray-cast render. The proposed system compresses volumetric data and renders images directly from the new data format. A more efficient solution was proposed in [42], the scheme contains an structure that allows volume shading computations to be performed on the codebook, and image generation is accelerated by reusing precomputed block projections. Schneider and Westermann [43] implemented a Laplacian pyramid vector quantification approach that allows relatively fast volume decompression and rendering on the GPU. However this method does not allow using the linear filtering capabilities of the GPU and the render cost increases when using high zoom factors. Eric B. Lum *et al.* [44] propose a palette-based decoding technique and an adaptive bit allocation scheme. This technique fully utilizes the texturing capability of 3D a graphics card.

Bricking techniques subdivide large volumes into several blocks, referred to as bricks, in such a way that ensures each block fits into GPU memory. Bricks are stored in main memory, then they are sorted either in front-to-back or back-to-front order with respect to the camera position, depending on the rendering algorithm [45, 46].

The objective in *multiresolution modeling schemes* [47–49] is to render only a region of interest at high resolution and to use progressively low resolution when moving away from that region. Both bricking and multiresolution approaches need a high memory capacity on the CPU for storing the original volume dataset. Moreover, bricking requires a high amount of texture transfers as each brick is sent once per frame; multiresolution techniques have been built for CPU purposes and the translation to GPUs is not straightforward due to the required amount of texture fetching. As Table 2 shows, different techniques allow multiresolution. An old technique proposed by Ghavamnia *et al.* [50], involves the use of the Laplacian Pyramid compression technique, which is a simple hierarchical computational structure. By using this representation, a compressed volume dataset can be efficiently transmitted across the network and stored externally on disk.

Progressive transmission has become an important solution for client-server architectures, allowing transmission of large volume datasets to clients according to rendering capabilities as well as hardware and network constraints of both server and client [51]. Qi *et al.* [22] proposed an approach capable of progressive transmission. The approach essentially compresses data by reducing noise outside the diagnostic region in each image of the 3D data set. They also reduce the inter-image and intra-image redundancy adjusting pixel correlations between adjacent images and within a single image. By applying a wavelet decomposition feature vector, they select a representative image from the representative subset of the entire 3D medical image set. With this reference image, they achieve good representation of all data with good contrast and anatomical feature details. The encoding technique ensures the progressive transmission scheme. Codified versions, from coarse to fine, of the reference image are transmitted gradually. In medical applications, radiologists could determine during transmission whether the desired image should be fully reconstructed or stop the process before transmission of the entire image set. Menmann *et al.* [52] present a hybrid CPU/GPU scheme for lossless compression and data streaming. By exploiting CUDA they allow the visualization of big out-of-core data with near-interactive performance.

More recently, Suter *et al.* [53] have proposed a *multiscale volume representation* based on a Tensor

Approximation within a GPU-accelerated rendering framework. The approach can be better than wavelets at capturing non-axis aligned features at different scales. Gobbetti *et al.* [29] have proposed a different multi-resolution compression approach using a sparse representation of voxel blocks based on a learned dictionary. Both approaches allow progressive transmission and obtain good compression ratios, but they are lossy and require huge data structures and a heavy pre-process.

Summary

Some recent and relevant compression techniques for volume visualization have been discussed in the literature, and a number of them have been included in client-server solutions. In Table 2, we present a comparison of these techniques. The columns show the stage of the pipeline where decompression takes place, whether the compression is lossless or lossy, and the applied compression technique. We also indicate which of these techniques are designed for medical image applications [54] and whether progressive transmission is allowed. The final columns show an estimation of the compression ratio as well as a qualitative measure of the reconstruction quality.

Table 3 summarizes the compression pipelines of the techniques presented in Table 2. The approaches reviewed have not been designed to use in mobile devices as clients, and none of them are transfer-function aware. Some of these techniques are not even designed for client-server architectures, but for compressing data from disk or to decrease bandwidth limitations.

Compression quality is usually measured by computing rate distortion curves for representative datasets. A common measure in image compression is the Peak Signal-to-Noise Ratio (PSNR) [55]. Fout *et al.* [56] designed a hardware-accelerated volume rendering technique using the GPU. The approach consists of a block-based transform coding scheme designed specifically for real-time volume rendering applications. An efficient decompression is achieved using a data structure that contains a 3D index to the codes and codebook textures. Guitián *et al.* [57] implemented a complex and flexible multiresolution volume rendering system capable of interactively driving large-scale multiprojectors. The approach exploits view-dependent characteristics of the display to provide different contextual information in different viewing areas like field displays. The proposal achieves high quality rendered images in acceptable frame rates.

Table 2: **Compression Schemes**. Columns show the stage of the pipeline where decompression takes place, whether the compression is lossless or lossy, and the applied compression technique. The table also indicates which techniques are designed for medical image applications and whether progressive transmission is allowed. The last columns show an estimation of the compression ratio as well as a qualitative measure of the reconstruction quality.

			Compression							Comp. Ratio			Reconst. Quality		
Decomp. Stage		Ref.	Lossless	Lossy	Comp. Technique	Medical Images	Multiresolution	Prog. Trans	Mobile	L	M	H	L	M	H
In CPU		[35]		✓	Wavelets	✓	✓	✓	×		-		✓		
		[58]	✓		Wavelets	-	✓	✓	×	✓				✓	
		[38]		✓	Wavelets	-	✓	-	×			✓		✓	
		[39]		✓	Wavelets	-	✓	-	×	✓				✓	
		[37]		✓	Wavelets	✓	✓	✓	×	✓				✓	
		[59]		✓	Wavelets	-	✓	-	×	✓				✓	
		[27]	✓		Wavelets	✓	✓	✓	×		-				✓
		[60]		✓	Wavelets	✓	✓	-	×		✓				✓
		[41]		✓	VQ	✓	✓	-	×		✓				✓
		[42]		-	VQ	✓	-	-	×		-		✓		
		[61]	✓		Huffman	✓	-	-	×		-		✓		
		[22]	✓		Wavelets	✓	-	-	×	✓			✓		
		[62]		✓	Fourier Transf.	✓	✓	-	×			✓		✓	
		[63]		✓	Fourier Transf.	✓	✓	-	×		✓			✓	
		[64]		✓	Fourier Transf.	-	-	-	×			✓		✓	
In GPU		[44]		✓	VQ	-	✓	-	×		✓				✓
		[65]		✓	Wavelets	-	✓	✓	×		-			✓	
		[53]		✓	Tensor Approx.	-	✓	✓	×			✓			✓
	During Rendering	[43]		✓	VQ	✓	✓	✓	×	✓			✓		
		[52]		✓	LZO	-	✓	✓	×		✓			✓	
		[66]		✓	Block-Based Codf.	✓	-	✓	×		-			✓	
		[67]		✓	VTC-LZO	-	-	-	×	✓				✓	
		[57]		✓	Frame Encod.	-	✓	-	×			✓			✓
		[68]		✓	DCT	✓	✓	-	×			✓	✓		
		[50]		✓	Laplacian Pyramid	✓	✓	-	×		✓			-	
		[29]	✓		Linear Comb.	✓	✓	✓	×		✓				✓
		[56]		✓	texture Comp.	-	-	-	×			✓			✓

Table 3: **Compression Scheme Pipelines**. Columns show the kind of Input data used in each approach, the preprocessing and encoding techniques applied. The table also shows the data structure or function to represent data, and finally, the decoding and rendering algorithms implemented.

Ref.	Input Data	Preprocessing/Encoding	Data Representation	Decoding	Rendering
[35]	MRI	-	3D Orthonormal Wavelets	-	Ray Casting
[58]	RM Instability	RLE+Huffman Coding	3D Haar Wavelets	RLE+Huffman Decoding	Ray Casting
[38]	CT	Quantization of Wavelet Coefficients	3D Haar Wavelets	Dequant. Wavelet Coefficients	-
[39]	CT	Quantization of Wavelet Coefficients	3D Haar Wavelets	Dequant. Wavelet Coefficients	Ray Casting
[37]	-	Block Subdivision/Encoding Wavelet Coefficients	Wavelets	Decoding W. Coefficients	3D texturing
[59]	Scanning Model	Encoding Wavelet Coefficients	3D Orthogonal Wavelets	Decoding W. Coefficients	Ray casting
[27]	MRI,CT	Encoding Wavelet Coefficients	Wavelets(Haar/B-Splines)	Decoding W. Coefficients	3D texturing
[60]	MRI,CT	Encoding Wavelet Coefficients	3D Haar Wavelets	Decoding W. Coefficients	Ray Casting
[41]	-	VQ. Encoding	Code Books	VQ. Decoding	2D Texturing
[42]	CT	VQ. Encoding	Code Books	VQ. Decoding	2D Texturing
[61]	MRI,CT	DPCM+Huffman Coding	Encoding Stream	Decoding DCPM + Huffman	2D Texturing
[22]	MRI,CT	DPCM+Huffman Coding	Encoding Stream	Decoding DCPM + Huffman	2D Texturing
[62]	-	-	Packing Textures	Decod. of Compressed Blocks	2D Texturing
[63]	MRI	-	3D Fourier Transform	-	Ray Casting
[64]	-	Quantization + Entropy Encoding	Fourier Transform	Dequant. + Entropy Decoding	Ray Casting
[44]	MRI	Filtred and threshold/RLE+Arithmetic Coding	IWT	RLE+Arithmetic Decoding.	2D Texturing
[65]	Time Varying Models	DCT	Quant. DCT Coefficients	RLE+DCT Decod.	2D Texturing
[53]	MRI,CT	Sorting, normalization/RLE+Huffman Coding	Wavelets	RLE+Huffman Decoding	2D Texturing
[43]	CT	BLE Based on Tensor Quantization	Wavelets	Tensor Recost. and Decoding	Ray Casting
[52]	CT	VQ Encoding	Code Books	VQ Decoding	3D Texturing
[66]	Hydrodynamical Simulation	Brick Decomp./Variable Lenght Encoding	LZO Compression of Blocks	Variable Lenght Decod.	Ray Casting
[67]	CT	Block Subdivision/VQ Encoding	Code Books	VQ Decoding	3D Texturing
[57]	CT	-	PVTC-LZO	PVTC-LZO Decoding	Ray Casting
[68]	CT	Frame Ecoding	-	-	Ray Casting
[50]	-	-	DCT	-	2D-Textures
[29]	CT	-	Laplacian Pyramid	-	Ray Casting
[56]	CT	Rectilinear to Compact Multiresolution Volume	Octree	Decod. of Compressed Blocks	Ray Casting

4 Compression Techniques for Dynamic Volume Models

This section includes a study and comparison client-server visualization techniques for animated volumes, which present a specialized form of data with an added dimension of complexity. In the context of medical imaging, the main motivation for visualizing such data is the ability to study the dynamic process of the human organism. In particular, a lot of scientists are involved in the analysis of the time-varying behavior of the heart, lungs, blood flow, eye and others. However, dealing with real-time applications for dynamic medical image visualization is still a challenging problem and client-server based schemes play an important role in this area.

Several commercial applications have already been widely used in the remote visualization of dynamic volume data, [69], [70]. Silicon Graphics developed Vizserver [69], a system that allows OpenGL programs to run without any modification in a client-server environment. There are almost no restrictions on the visualization, however the number of users that are allowed to visualize and interact independently with the same dataset simultaneously under the same graphic pipeline is limited. Remote collaborative sessions involve a remote client starting a session on a newly created X server, and multiple clients subsequently joining the session. The system includes a lossless compression scheme that works by transmitting only differences from the previous frame. GLX is an OpenGL extension for the X server system [70]. It allows 3D graphics rendering applications to display on a remote X server. This scheme implements command streaming for rendering 3D data on the client side. All geometry models, texture maps, and operation commands are streamed to the client before rendering. Thus, for applications that perform complex graphics rendering, excessive network bandwidth may be required.

Temporal coherence based approaches are becoming useful tools for the visualization of animated volumes, [71], [72], [73], [74], [75]. Younesy et al. [71], exploit the temporal coherence concept by introducing a novel data structure called the Differential Time-Histogram table (DTHT) that stores voxels which are changing between time-steps or during transfer function updates. Fang et al. [72] developed a time activity curve (TAC) to identify temporal patterns while in [73], the detection of important regions is achieved by studying the local statistical complexity. Wang et al. [74], compute an importance curve for each data block after applying conditional entropy. Curves are then used to evaluate the temporal behaviour of

blocks. A more recent approach, [75], uses functional representation of time-varying datasets to develop an efficient encoding technique, taking into account the temporal similarity between timesteps. Akiba et al. [76] proposed a technique that uses time histograms for simultaneous classification of time-varying data by partitioning time histograms into temporally coherent equivalence classes.

Park et al. [77] presented a technique to effectively visualize time-varying oceanography data. The approach uses an offline parallel rendering algorithm to create high quality images. They also proposed a real-time rendering scheme with a hybrid technique that merges volume rendering and isocountorning. The proposal is a client/server oriented technique that allows generating high resolution videos of datasets of up to $78.6GB$ and dimensions of $2160 \times 960 \times 30$ with 121 timesteps.

Summary

Although there have been interesting practical solutions to time-varying volume data visualization, a lot of open problems still merit further investigation. Transmitting the whole spatiotemporal volume model ensures full interaction capabilities in the client without further communication, but limitations in the bandwidth and network latency during real-time visualization pose challenges for real-time visualization. Compressed volume rendering methods present a good option for rendering models whose size exceeds current hardware capabilities, which is easily the case with time-variant data that may consist of several frames of 3D volumetric information. However, performance during decompression poses alternative challenges. Thin-client solutions are popular means of simplifying the problem by transmitting only rendered images through the network, but the disadvantage is that there is no volume information on the client side, leading to decreased performance when new data is demanded.

5 Visualization on Low-End Devices

The objective of this brief section is to discuss the concerns of designing volume rendering algorithms for mobile devices specifically, including approaches related to both, static and dynamic dataset described in previous sections. The display resolution, the lack of power capacity, as well as memory and storage limitations, mean that mobile devices must be considered low-end devices in the context of client-server architecture implementations [78]. Also included in this category are computers that would be considered low-performance compared to sophisticated computers specifically equipped to run complex graphics applications.

As mentioned in the introduction, tele-medicine and tele-diagnostic are techniques widely demanded by modern hospitals. Several medical image applications require high performance computing and well designed network infrastructures. The advance of mobile telephony and mobile devices in general, have motivated novel development in addition to the refinement of existing visualization algorithms, many of which were originally designed for video games. In addition to libraries being updated, new software and platforms for embedded systems have also emerged. These factors as well as the relative ease of maintenance and portability make mobile devices the preferred alternative for scientists and developers. Because of this, their limitations are becoming a strong field of discussion for computer graphics researchers [78].

In parallel, volume models have grown significantly in complexity in recent years. The amount of memory of modern GPUs is also growing, but unfortunately this is surpassed by the rate of increase in the size of volumetric data sets. Most of the recent approaches for volume visualization in mobile devices have been implemented using OpenGL ES [14–16], which is an application programming interface for advanced 3D graphics targeted at handheld and embedded devices. This library addresses some device constraints, such as preprocessing capability and memory availability, low memory bandwidth, sensitivity to power consumption, and the lack of floating-point hardware. However, the limited amount of published work and effective solutions in the area, as well as the user interaction requirements for this application paradigm mean that volume visualization on mobile devices is a highly challenging and open area of research.

6 Conclusions and Discussion

Despite continual advances in computing power, the use of high performance graphic workstations is still considerably restricted due to the cost of the most modern hardware. Widespread deployment of high-speed wireless networks have increased the interest in remote rendering, which has some evident advantages. It provides rich rendering experiences to thin clients such as mobile devices with limited computing resources, shared by multiple clients and is a simple but effective cross-platform solution. However the minimization of interaction latency still remains a challenge.

The study of client-server architectures for volume visualization is at present a wide area of research for computer graphics scientists. We have made a comparative analysis of some relevant approaches in the area, but of course it is impossible to fully cover all published works in this paper. Although there have been interesting practical solutions in recent years, a number of open problems remain for further investigation.

Sending the whole volume model in a compressed way to the client device is still a challenging problem. This ensures full interaction facilities in the client without further server-client communication. Many proposals achieve good results, but the small allowed size of models and network latency during real time visualization, decrease interaction capabilities. As an alternative most techniques transmit images through the network, but such approaches are limited by the lack of volume information in the client side, decreasing performance when new data is demanded and affecting image quality when reconstruction takes place in clients.

Compressed volume rendering methods are a good alternative for rendering models whose size exceeds current hardware capabilities. However, performance during decompression is still an area much in need of improvements. Although some proposed techniques decompress data using the CPU, the current trend is to move all the decompression to the last part of the graphic pipeline. In this way, the data gets compressed to the GPU, which results in less memory consumption and better exploitation of available bandwidth.

Data transformation, quantization, encoding and progressive transmission, are strongly related concepts. Some approaches convert volume data into compact data structures ready to be gradually sent from servers to clients. In cases where no client-server architectures are implemented, compact data structures are used to transfer data between CPU and GPU or to efficiently perform out-of-core methods.

Medical visualization requires special treatment. Current solutions propose novel improvements without fully exploiting the fact that doctors usually adopt standard transfer functions. This factor can be leveraged to achieve even higher levels of data compression. Moreover, low-end devices such as mobiles are being preferred due to it easy maintenance and portability but their limitations demand further refinement of client-server algorithms to optimally increase interactivity while inspecting large volume models.

In many scientific simulations, exploiting spatial and temporal coherence is a means of avoiding increasing computational cost and reducing display time. Decreasing the time required to transfer a sequence of volumes to the rendering engine is still a considerable challenge. Any such improvements must be achieved without removing fine features from the handled dataset, which could be relevant to the task being supported by the visualisation.

References

[1] Keir Davis, John W Turner, and Nathan Yocom. Client-server architecture. In *The Definitive Guide to Linux Network Programming*, pages 99–135. 2004.

[2] M.M. Kekic, G.N. Lu, and E.H. Carlton. Client-server computer network management architecture, December 16 2003. US Patent 6,664,978.

[3] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. Aspects of networking in multiplayer computer games. *The Electronic Library*, 20(2):87–97, 2002.

[4] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D Kirchner, and James T Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. In *ACM Transactions on Graphics*, volume 21, pages 693–702, 2002.

[5] Goretti Echegaray, Imanol Herrera, Iker Aguinaga, Carlos Buchart, and Diego Borro. A brain surgery simulator. *IEEE Computer Graphics and Applications*, 34(3):12–18, 2014.

[6] Giuseppe Turini, Nico Pietroni, Giuseppe Megali, Fabio Ganovelli, Andrea Pietrabissa, and Franco Mosca. New techniques for computer-based simulation in surgical training. *International Journal of Biomedical Engineering and Technology (IJBET)*, 5(4):303–316, 2011.

[7] Odysseus Argy and Michael Caputo. Introduction to telemedicine. In *Information Technology for the Practicing Physician*, pages 227–233. 2001.

[8] Richard Wootton. Telemedicine: a cautious welcome. *Bmj*, 313(7069):1375–1377, 1996.

[9] Douglas A Perednia and Ace Allen. Telemedicine technology and clinical applications. *Jama*, 273(6):483–488, 1995.

[10] Yonggang Huang, ChunYang Hu, Yongwang Zhao, and Dianfu Ma. Web-based remote collaboration over medical image using web services. In *Global Information Infrastructure Symposium*, pages 1–8, 2009.

[11] M. Balpande and U. Shrawankar. Medical image fusion techniques for remote surgery. In *IEEE India Conference (INDICON)*, pages 1–6, Dec 2013.

[12] Wang Xiao-min, Wang Peng-cheng, and Xie Jin-dong. A new design of remote diagnosis system for medical images. In *International Conference on Management and Service Science*, pages 1–3, 2009.

[13] Steven P Callahan, Louis Bavoil, Valerio Pascucci, and Claudio T Silva. Progressive volume rendering of large unstructured grids. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1307–1314, 2006.

[14] Manuel Moser and Daniel Weiskopf. Interactive volume rendering on mobile devices. In *Vision, Modeling, and Visualization VMV*, volume 8, pages 217–226, 2008.

[15] Movania Muhammad Mobeen and Lin Feng. Ubiquitous medical volume rendering on mobile devices. In *International Conference on Information Society (i-Society)*, pages 93–98. IEEE, 2012.

[16] Marcos Balsa Rodríguez and Pere Pau Vázquez Alcocer. Practical volume rendering in mobile devices. In *Advances in Visual Computing*, pages 708–718. 2012.

[17] Carlos Antonio Andújar Gran, Jonás Martínez Bayona, et al. Locally-adaptive texture compression. *Proceedings of the Spanish Conference in Computer Graphics*, 2009.

[18] D. Taubman. High performance scalable image compression with ebcot. *IEEE Transactions on Image Processing*, 9(7):1158–1170, 2000.

[19] P. Simoens, P. Praet, B. Vankeirsbilck, J. De Wachter, L. Deboosere, F. De Turck, B. Dhoedt, and P. Demeester. Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices. In *Telecommunication Networks and Applications Conference*, pages 391–396, Dec 2008.

[20] Klaus Engel and Thomas Ertl. Texture-based volume visualization for multiple users on the world wide web. In *Virtual Environments*, pages 115–124. 1999.

[21] Klaus Engel, Thomas Ertl, Peter Hastreiter, Bernd Tomandl, and K Eberhardt. Combining local and remote visualization techniques for interactive volume rendering in medical applications. In *Proceedings of the conference on Visualization'00*, pages 449–452, 2000.

[22] Xiaojun Qi and John M Tyler. A progressive transmission capable diagnostically lossless compression scheme for 3d medical image sets. *Information Sciences*, 175(3):217–243, 2005.

[23] Liviu Constantinescu, Jinman Kim, and Dagan Feng. Integration of interactive biomedical image data visualisation to internet-based personal health records for handheld devices. In *International Conference on e-Health Networking, Applications and Services*, pages 79–83, 2009.

[24] Seok-Jae Jeong and Arie E Kaufman. Interactive wireless virtual colonoscopy. *The Visual Computer*, 23(8):545–557, 2007.

[25] Anna Tikhonova, Carlos D Correa, and Kwan-Liu Ma. Explorable images for visualizing volume data. In *IEEE Pacific Visualization Symposium*, pages 177–184, 2010.

[26] Wes Bethel. Visualization dot com. *IEEE Computer Graphics and Applications*, 20(3):17–20, 2000.

[27] Lars Lippert, Markus H Gross, and Christian Kurmann. Compression domain volume rendering for distributed environments. In *Computer Graphics Forum*, volume 16, pages C95–C107, 1997.

[28] Imma Boada and Isabel Navazo. Optimal exploitation of client texture hardware capabilities on a client-server remote visualization framework. In *Computational Science and Its Applications ICCSA*, pages 468–477. 2003.

[29] Enrico Gobbetti, José Antonio Iglesias Guitián, and Fabio Marton. Covra: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. In *Computer Graphics Forum*, volume 31, pages 1315–1324, 2012.

[30] Fabrizio Lamberti, Claudio Zunino, Andrea Sanna, Antonino Fiume, and Marco Maniezzo. An accelerated remote graphics architecture for pdas. In *Proceedings of the International Conference on 3D Web Technology*, pages 55–ff, 2003.

[31] Hariharan G Lalgudi, Michael W Marcellin, Ali Bilgin, Han Oh, and Mariappan S Nadar. View compensated compression of volume rendered images for remote visualization. *IEEE Transactions on Image Processing*, 18(7):1501–1511, 2009.

[32] Chuan-kai Yang. Integration of volume visualization and compression: A survey. *CiteSeer*, 2000.

[33] Marcos Balsa Rodriguez, Enrico Gobbetti, José Antonio Iglesias Guitián, Maxim Makhinya, Fabio Marton, Renato Pajarola, and Susanne Suter. A survey of compressed gpu-based direct volume rendering. In *Eurographics State of the Art Report*, 2013.

[34] Shigeru Muraki. Approximation and rendering of volume data using wavelet transforms. *Proceedings of the Conference on Visualization*, pages 21–28, 1992.

[35] Shigeru Muraki. Volume data and wavelet transforms. *IEEE Computer Graphics and Applications*, 13(4):50–56, 1993.

[36] Insung Ihm and Sanghun Park. Wavelet-based 3D compression scheme for interactive visualization of very large volume data. *Computer Graphics Forum*, 18:3–15, 1999.

[37] Stefan Guthe, Michael Wand, Julius Gonser, and Wolfgang Straßer. Interactive rendering of large volume data sets. In *IEE Visualization*, pages 53–60. IEEE, 2002.

[38] Ky Giang Nguyen and Dietmar Saupe. Rapid high quality compression of volume data for visualization. In *Computer Graphics Forum*, volume 20, pages 49–57, 2001.

[39] Flemming Friche Rodler. Wavelet based 3d compression with fast random access for very large volume data. In *Proceedings Of the Pacific Conference on Computer Graphics and Applications*, pages 108–117, 1999.

[40] R.M. Gray. Vector quantization. *ASSP Magazine, IEEE*, 1(2):4–29, April 1984.

[41] Paul Ning and Lambertus Hesselink. Vector quantization for volume rendering. In *Proceedings of the Workshop on Volume Visualization*, pages 69–74, 1992.

[42] Paul Ning and Lambertus Hesselink. Fast volume rendering of compressed data. In *Proceedings of the IEEE Conference on Visualization*, pages 11–18, 1993.

[43] Jens Schneider and Rüdiger Westermann. Compression domain volume rendering. In *IEEE Visualization*, pages 293–300, 2003.

[44] Eric B Lum, Kwan Liu Ma, and John Clyne. Texture hardware assisted rendering of time-varying volume data. In *Proceedings of the Conference on Visualization*, pages 263–270, 2001.

[45] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel. *Real-time Volume Graphics*. A. K. Peters, Ltd., 2006.

[46] Magnus Strengert, Marcelo Magallon, Daniel Weiskopf, Stefan Guthe, and Thomas Ertl. Hierarchical visualization and compression of large volume datasets using GPU clusters. *Parallel Graphics and Visualization*, D:41–48, 2004.

[47] Eric LaMar, Bernd Hamann, and Kenneth I. Joy. Multiresolution techniques for interactive texture-based volume visualization. In *IEEE Visualization*, pages 355–361, 1999.

[48] Imma Boada, Isabel Navazo, and Roberto Scopigno. Multiresolution volume visualization with a texture-based octree. *The Visual Computer*, 17(3):185–197, 2001.

[49] Manfred Weiler, Rüdiger Westermann, Charles D. Hansen, Kurt Zimmerman, and Thomas Ertl. Level of detail volume rendering via 3D textures. In *Volvis*, pages 7–13, 2000.

[50] Mohammad H Ghavamnia and Xue D Yang. Direct rendering of laplacian pyramid compressed volume data. In *Proceedings of the conference on Visualization*, page 192, 1995.

[51] F. J. Melero, P. Cano, and J. C. Torres. Bounding-planes octree: A new volume-based lod scheme. *Computer Graphics*, 32(4):385–392, 2008.

[52] Jörg Mensmann, Timo Ropinski, and Klaus Hinrichs. A gpu-supported lossless compression scheme for rendering time-varying volume data. In *Proceedings of the 8th IEEE/EG international conference on Volume Graphics*, pages 109–116, 2010.

[53] Susanne K Suter, Jose A Iglesias Guitian, Fabio Marton, Marco Agus, Andreas Elsener, Christoph PE Zollikofer, Meenakshisundaram Gopi, Enrico Gobbetti, and Renato Pajarola. Interactive multiscale tensor reconstruction for multiresolution volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2135–2143, 2011.

[54] Josien P.W Pluim, J.B Antoine Maintz, and Max A Viergever. Mutual-information-based registration of medical images: a survey. *IEEE Transactions on Medical Imaging*, 22(8):986–1004, 2003.

[55] Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.

[56] Nathaniel Fout, Hiroshi Akiba, Kwan-Liu Ma, Aaron E Lefohn, and Joe Kniss. High quality rendering of compressed volume data formats. In *Proceedings of the Eurographics/IEEE VGTC conference on Visualization*, pages 77–84, 2005.

[57] José Antonio Iglesias Guitián, Enrico Gobbetti, and Fabio Marton. View-dependent exploration of massive volumetric models on large-scale light field displays. *The Visual Computer*, 26(6-8):1037–1047, 2010.

[58] Han-Wei Shen. Visualization of large scale time-varying scientific data. In *Journal of Physics: Conference Series*, volume 46, page 535, 2006.

[59] Markus H Gross, Markus H Gross, and Markus H Gross. *Integrated volume rendering and data analysis in wavelet space*. Swiss Federal Institute of Technology, Computer Science Department, 1996.

[60] Insung Ihm and Sanghun Park. Wavelet-based 3d compression scheme for very large volume data. In *Graphics Interfaces*, volume 98, pages 107–116, 1998.

[61] James E Fowler and Roni Yagel. Lossless compression of volume data. In *Proceedings of the symposium on Volume visualization*, pages 43–50, 1994.

[62] Shane Dunne, Sandy Napel, and Brian Rutt. Fast reprojection of volume data. In *Proceedings of the First Conference on Visualization in Biomedical Computing*, pages 11–18, 1990.

[63] Takashi Totsuka and Marc Levoy. Frequency domain volume rendering. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pages 271–278, 1993.

[64] Tzi-cker Chiueh, Chuan-kai Yang, Taosong He, Hanspeter Pfister, and Arie Kaufman. Integrated volume compression and visualization. In *IEEE Proceedings on Visualization*, pages 329–336, 1997.

[65] Markus H Gross, Lars Lippert, and Oliver G Staadt. Compression methods for visualization. *Future Generation Computer Systems*, 15(1):11–29, 1999.

[66] Nathaniel Fout and Kwan-Liu Ma. Transform coding for hardware-accelerated volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1600–1607, 2007.

[67] Daisuke Nagayasu, Fumihiko Ino, and Kenichi Hagihara. Two-stage compression for fast volume rendering of time-varying scalar data. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, pages 275–284, 2006.

[68] Boon-Lock Yeo and Bede Liu. Volume rendering of dct-based compressed 3d scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):29–43, 1995.

[69] Chikai Ohazama. Opengl vizserver. *White Paper, Silicon Graphics Inc*, 1999.

[70] Mark J Kilgard. *OpenGL programming for the X Window System*. Addison Wesley Longman Publishing Co., Inc., 1996.

[71] J. Younesy, T. Moller, and H. Carr. Visualization of time-varying volumetric data using differential time-histogram table. In *Volume Graphics, 2005. Fourth International Workshop on*, pages 21–224, 2005.

[72] Zhe Fang, Torsten Möller, Ghassan Hamarneh, and Anna Celler. Visualization and exploration of time-varying medical image data sets. In *Proceedings of Graphics Interface 2007*, GI '07, pages 281–288. ACM, 2007.

[73] H Janicke, Alexander Wiebel, Gerik Scheuermann, and Wolfgang Kollmann. Multifield visualization using local statistical complexity. *IEEE Transactions onVisualization and Computer Graphics*, 13(6):1384–1391, 2007.

[74] Chaoli Wang, Hongfeng Yu, and Kwan-Liu Ma. Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1547–1554, 2008.

[75] Yun Jang, David S Ebert, and Kelly Gaither. Time-varying data visualization using functional representations. *Visualization and Computer Graphics, IEEE Transactions on*, 18(3):421–433, 2012.

[76] Hiroshi Akiba, Nathaniel Fout, and Kwan-Liu Ma. Simultaneous classification of time-varying volume data based on the time histogram. In *Proceedings of the Eighth Joint Eurographics / IEEE VGTC Conference on Visualization*, pages 171–178. Eurographics Association, 2006.

[77] Sanghun Park, Rajit Bajaj, and Insung Ihm. Effective visualization of very large oceanography time-varying volume dataset. In *in ICCS 2004, LNCS 3037*. Citeseer, 2001.

[78] T. Capin, K. Pulli, and T. Akenine-Moller. The state of the art in mobile graphics research. *IEEE Computer Graphics and Applications*, 28(4):74–84, 2008.