Compression / Decompression Strategies for Large Volume Medical Imagery

Karthik Krishnan[a], Michael W. Marcellin[a], Ali Bilgin[a], Mariappan Nadar[b]

[a]Signal Processing and Coding Laboratory, Department of Electrical and Computer Engineering, 1230 E. Speedway Blvd, University of Arizona, Tucson, AZ, USA 85721;
[b]Siemens Corporate Research Inc. Princeton, NJ, USA 08540

ABSTRACT

We present a scheme for compressed domain interactive rendering of large volume data sets over distributed environments. The scheme exploits the distortion scalability and multi-resolution properties offered by JPEG2000 to provide a unified framework for interactive rendering over low bandwidth networks. The interactive client is provided breadth in terms of scalability in resolution, position and progressive improvement by quality. The server exploits the spatial locality offered by the DWT and packet indexing information to transmit, in so far as possible, compressed volume data relevant to the clients query. Once the client identifies its volume of interest (VOI), the volume is refined progressively within the VOI. Contextual background information can also be made available having quality fading away from the VOI. The scheme is ideally suited for client-server setups with low bandwidth constraints, with the server maintaining the compressed volume data, to be browsed by a client with low processing power and/or memory. Rendering can be performed at a stage when the client feels that the desired quality threshold has been attained. We investigate the effects of code-block size on compression ratio, PSNR, decoding times and data transmission to arrive at an optimal code-block size for typical VOI decoding scenarios.

Keywords: JPEG2000, embedded coding, scalable compression, medical volume compression, volume of interest, interactive visualization, DWT, code-block

1. INTRODUCTION

With the widespread use of the Internet, online medical volume databases, such as those maintained by the National Library of Medicine (NLM) have gained popularity. With recent advances in picture archiving communication systems and telemedicine, improved techniques for interactive visualization across distributed environments are being explored. Typically, data sets are stored and maintained by a database server, so that one or more remote clients can browse the datasets interactively and render them as in Fig. 1. The client, in many cases, is a low-end workstation with limited memory and processing power. An interactive user may be willing to initially sacrifice some rendering quality or field of view in exchange for real-time performance. Hence one of the fundamental needs of a client is breadth in terms of interactivity (such as reduced resolution viewing, ability to view a select subsection of the volume, pan, zoom, view select slices, etc.) and a pleasant and real-time viewing experience (immediate and progressive refinement of the view volume, etc.). Such a setup enforces a rigorous set of constraints.

- *Compression*: In general volumetric data sets are massive, for example the Visible Male data set consists of about 15 gigabytes of volume data. If such data sets need to transmitted over low-bandwidth networks with varying loads and latency constraints, efficient compression schemes must be employed.

Further author information: (Send correspondence to Karthik Krishnan)
Karthik Krishnan: E-mail: karthikk@ece.arizona.edu, Telephone: 1 520 621 9769

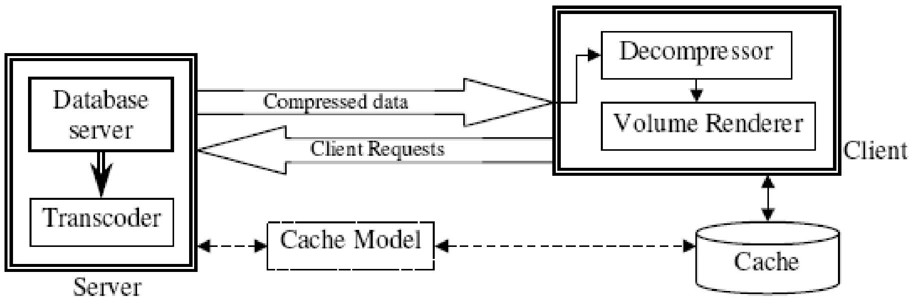


Figure 1. Client-Server model

- *Scalability*: The benefit of compression would be somewhat diminished if the entire data had to be decompressed prior to visualization. Hence it is desirable for the compressed bit-stream to be scalable. Considering that clients are typically limited in memory, it is imperative that the data transmitted by the server be scalable by resolution. This enables a client to browse a low-resolution version of the volume and appropriately choose a Volume of Interest (VOI). Distortion scalability is also of interest, so that the VOI of the client is progressively refined by quality. Additionally, scalability by position, or spatial location is also sought after in interactive applications, where interactive users may wish to view a particular sub-section of the volume. Finally, since rendering time is linear in the size of the data set, the compression technology must be based on a multi-resolution framework, with reduced resolution viewing making it possible to save on compressed data transmitted through the network as well as rendering time.

- *Communication Protocol*: The client and the server must employ a generic protocol that is easily deployed on a variety of channels. Firstly, the client-server protocol must be transport neutral. Secondly, each packet returned by the server must be a self-contained unit. The need for this stems from the fact that packets may be received out of order and the client should not have to wait to improve its cache. The scheme becomes mandatory when transport protocols, with very few error recovery services and/or high packet erasure channels are employed. In such cases absence of a 'self-contained' scheme would mean that the client must wait till the server resends lost packets.

- *Cache Model*: Volumetric data sets are huge, so memory and disk caching schemes will improve performance. The client must maintain a cache of data transmitted by the server and must be able to delete elements from the cache, once they are outside the VOI of an interactive user. The server must also maintain a model of the client's cache, so as to avoid resending any elements that the client already has in its cache. The need for the server to be aware of the clients cache contents is particularly necessary in wavelet based compression schemes, since even two VOI's can have significant overlap of compressed data. Hence the communication protocol must allow the client to communicate its cache state to the server.

1.1. Prior Work

Volume rendering of compressed 3D scalar data using DCT has been addressed in [1] . The drawback with the DCT based JPEG scheme is its lack of richness in terms of scalability. Wavelet based 3D coders have been studied in [2] . Popular wavelet based 3D coders have been surveyed in [3] . 3D Set Partitioning in Hierarchical Trees (SPIHT) has been employed in [4] to perform lossless compression of volumetric medical datasets. Motion Adaptive wavelet transforms have been applied to volumetric datasets in [5] . A wavelet splatting based scheme has bee proposed in[6] for interactive visualization across networks.

JPEG2000[7] is the new image compression technology designed to support a variety of applications, including the compression and transmission of medical images. In Nov 2001, JPEG2000 was selected for inclusion in the DICOM standard for medical image transfer. The DICOM protocol now includes JPEG2000 transfer syntaxes. The research presented here attempts to use the underlying features of JPEG2000 and the JPEG2000 Interactive Protocol (JPIP)[8] to provide a framework for compressed domain interactive visualization across networks.

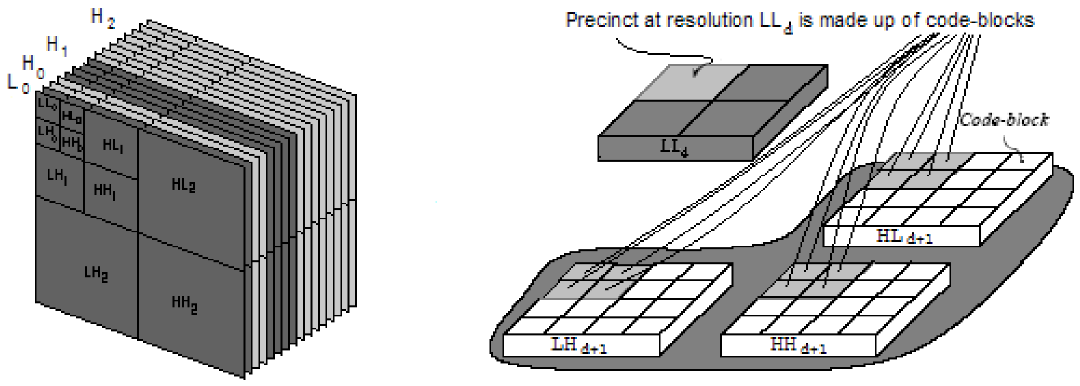


Figure 2. (a) DWT of the volume into subbands. (b) The precinct and code-block partitions

2. JPEG2000 CODE-STREAM ELEMENTS

JPEG2000 is a highly scalable image compression standard which supports multiple progression orders. It employs the Discrete Wavelet Transform (DWT), followed by Embedded Block Coding with Optimized Truncation [9] . Each level of DWT decomposes its input into four spatial frequency subbands, LL_d, LH_d, HL_d and HH_d. A D_{xy} DWT decomposes an image into 3 D_{xy}+1 subbands. We are dealing with medical volumes, which are typically acquired as a sequence of slices. We exploit the correlation across slices by applying a DWT along the slice direction prior to the X-Y DWT (as specified in JPEG2000 Part 2), forming subbands as illustrated in Fig. 2(a). Since JPEG2000 supports up to 2^{32}-1 components, the resulting sequence may be compressed into a JPEG2000 code-stream with the subband slices treated as components.

2.1. Code-Blocks, Precincts and Packets

Each subband of each slice is partitioned into rectangular blocks, known as *code-blocks*. Entropy coding is performed independently on each code-block. The code-block may be multi-layered to provide for distortion scalability. A *precinct* is a collection of code-blocks representing some finite spatial extent at some resolution. (See Fig. 2(b)). Unlike the code-block or subband partition, the precinct partition does not affect the transformation or coding of sample data. Instead, precincts and packets play a key role in organizing compressed data in the code-stream. The precinct grouping structure facilitates extraction of a VOI. The fundamental unit of code-stream organization is the *packet*. A packet can be thought of as one quality increment (layer) of one spatial location (precinct) of one resolution of one tile-component.

2.2. Scalability

The ordering of packets within the JPEG2000 code-stream is the key to the rich set of progression orders that it supports. Resolution scalability is a direct consequence of the multi-resolution property of the DWT. Distortion scalability is provided by the multi-layered organization of the code-stream into packets. Spatial random access is possible due to the presence of precincts, since a precinct is a collection of spatially adjacent code-blocks and each code-block is associated with a limited spatial region.

3. CLIENT-SERVER COMMUNICATION

Considering that clients are typically low-end workstations, the task of figuring out the compressed elements of interest and ordering them for transmission is delegated to the server. The communication protocol we employ here is JPIP compliant. The interactive user makes a request to view a particular region of the volume, typically at a lower resolution, which may be different along the slice and the xy directions. Since we treat slices as components, with a D_{slice} level of transform applied across the components, the client must deduce the image components of interest. The client then makes a request for the ROI_{xy} across the relevant image components. The server deduces the precincts of interest corresponding to the client's request and delivers them incrementally to enhance the client's VOI. It must be noted that the trivial task of evaluating the image components of interest could have been delegated to the server. However we left the task to the client, so as to have a JPIP compliant syntax, since JPIP is designed for images and supports no syntax for volumes.

3.1. Transcoder

The JPEG2000 code-stream of a volume may be ordered in a variety of progression orders and the code-stream itself might have been compressed with arbitrarily chosen parameters. Precinct sizes could have been arbitrarily large so as to fill the entire subband. Hence the server may transcode the input volume, on the fly, into one with smaller precinct sizes [8] , nominally chosen to be twice the size of the code-block. Since transcoding simply repackages existing code-blocks to conform to smaller precinct sizes, without actually decoding them, it does not consume significant resources. Thus the transcoder provides granularity suitable for interactive visualization. However this granularity is still limited by the code-block size, which can have a significant impact on interactivity, as we shall see below.

4. RESULTS

We conducted experiments on a 512x512x256 segment of a data set obtained from a CT scan of the abdomen, obtained from Siemens Medical Solutions. The volume was sampled at 0.74 mm along the x and y directions and the slice thickness was 1 mm. Two bytes are used for each voxel, with 12 bits being the actual bit depth. The tests were conducted on a PC workstation with 1.8 GHz processing power and 1GB of RAM.

4.1. Compression Performance

It is natural to expect higher compression ratios by exploiting the correlation across slices. We tested the PSNR for volumes compressed with compression ratios varying from 10 to 160 for code-block sizes of 4, 8, 16 and 32. The use of the 9,7 wavelet along the z direction provided an average PSNR improvement of 3.5 dB for each given compression ratio. (See Fig. 3). It should be noted that apart from providing higher compression ratios, the slice-DWT enables reduced resolution visualization along the slice direction, which is crucial to an interactive client.

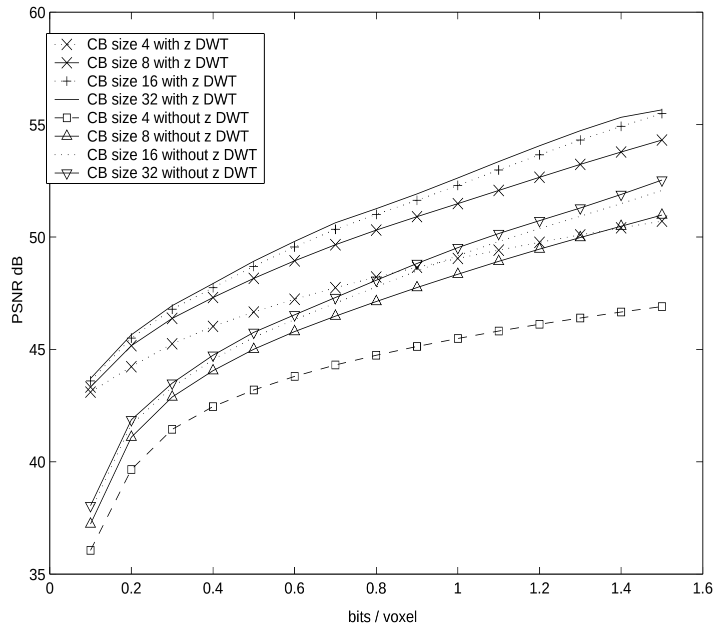


Figure 3. PSNR gain by applying the 9,7 wavelet across the slices (z direction).

4.2. Code-Block Sizes

Although code-block sizes of 32 and 64 provide the highest PSNR to compression ratio performance, they are not ideally suited for an interactive application. Each code-block is associated with a limited spatial region due to the finite footprint of the wavelet kernel. Hence smaller code-block sizes give finer granularity, and the need for less data transmission. However conforming precinct sizes to the smaller code-block dimensions can result in increased packet signaling overhead and increased disk thrashing. Besides, as seen from Fig. 3, smaller code-block sizes give reduced compression performance. Hence we expect a tradeoff in terms of compression performance, decoding times and packet signaling overhead against granularity when the client requests a certain VOI.

We first compared the decoding times taken to decode the entire volume at 0.5 bits/voxel, full resolution. To discount the effects of network latency, we made the compressed volume data available to the client on the hard disk. As expected, smaller code-block sizes result in increased decoding times due to higher disk fetches and the increased overhead of decoding smaller code-blocks. We then compared the decoding times to decode a 20%x20%x20% subsection of the compressed volume data at 0.5 bits/voxel, full resolution. To reconstruct a volume of interest the decoder evaluates the subband coefficient locations on the grid that are involved in the reconstruction of the VOI and then decodes the code-blocks that have a non-zero intersection with the relevant subband coefficients. It must be noted that although a mere fraction of the subband samples of a code-block may be relevant to the reconstruction of the VOI, the context modeling employed by the arithmetic coder necessitates decoding of the entire code-block. Hence, smaller code-block partitions give finer granularity and reduced VOI decoding times, but at the expense of increased overhead due to increased code-block fetches.

From Table 1, we see that code-block sizes of 16 and 8 consume significantly reduced decoding times with code-block sizes of 16 giving nearly 25% reduction in decoding times as compared to code-block sizes of 32. Even when the entire volume was decoded, decoding times resulting from code-block sizes of 16 were nearly the same as those of 32 and 64. However, in this case, code-block sizes of 4 performed poorly. Fig. 4 compares decoding times vs. bit rate for the test volume, a 60%x60%x20% subsection and a 20%x20%x20% subsection of the test volume.

	Code-block size				
	4	8	16	32	64
Decoding times of entire volume (s)	14.00	12.45	12.02	12.07	12.02
VOI decoding times (s)	1.59	1.28	1.21	1.61	1.63

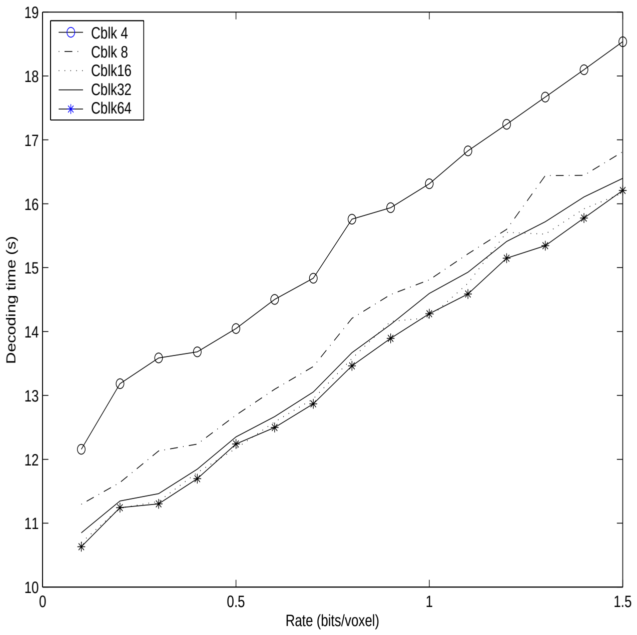
Table 1. Decoding times in seconds of the entire volume and of the VOI at 0.5 bits/voxel, full resolution, VOI represents 20%x20%x20% subsection of the volume.

4.3. Packet Data

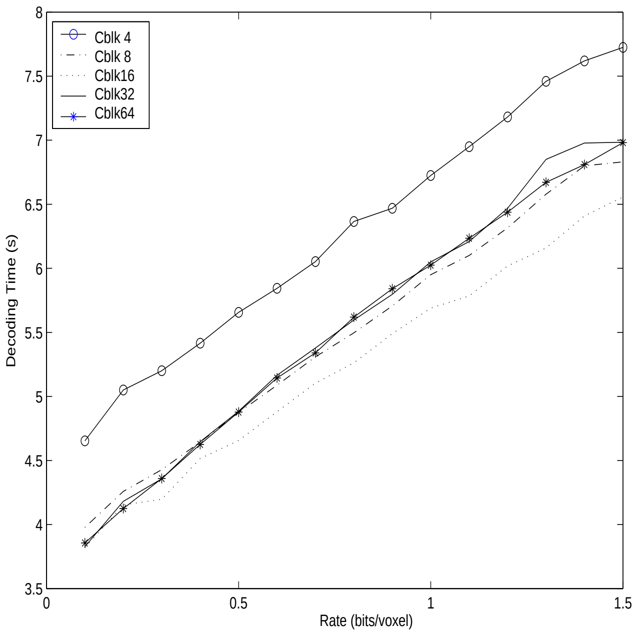
As mentioned in Section 1, the server responds to the client requests with a chunk of packets. The size of each chunk is appropriately chosen so as to maintain responsiveness. Some packaging overhead is inevitably involved with each server response. The chunks contain an encoded header, as defined by JPIP's data-bin identifier, which completely describes its contents in such a way that the client can use almost any server reply to improve its view volume. The size of each transmitted sequence is appropriately chosen based on the network bandwidth and user defined parameters, so as to maintain responsiveness.

4.4. Progression Order

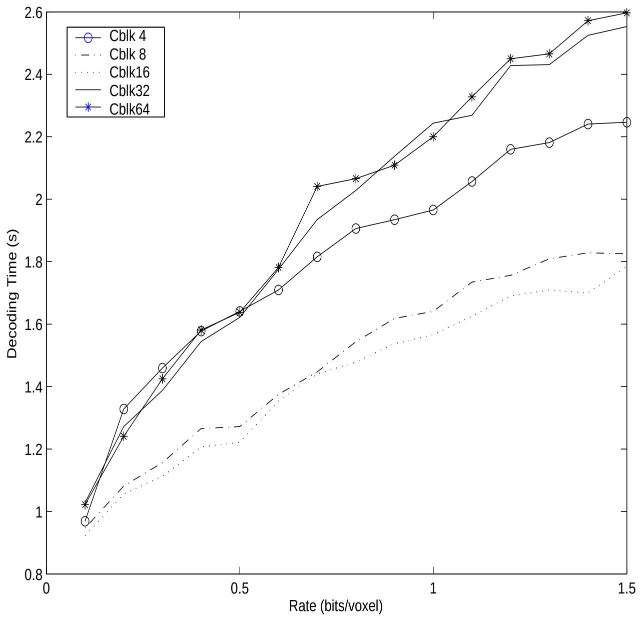
As mentioned earlier, progression enables increasing quality, resolution, spatial extent and/or color components (in our case subband slices) as more code-stream bytes are decoded sequentially. The type of progression present in a JPEG2000 code-stream is governed by the order in which packets appear within tile-streams. Five progression orders are supported by the standard namely, LRCP (Layer - Resolution - Component - Position), RLCP (Resolution - Layer - Component - Position), RPCL (Resolution - Position - Component - Layer), PCRL (Position - Component - Resolution - Layer) and CPRL (Component - Position - Resolution - Layer). The ordering of packets, in each of these progressions, can be understood as a nested loop with the first term of the



(a) Full volume.

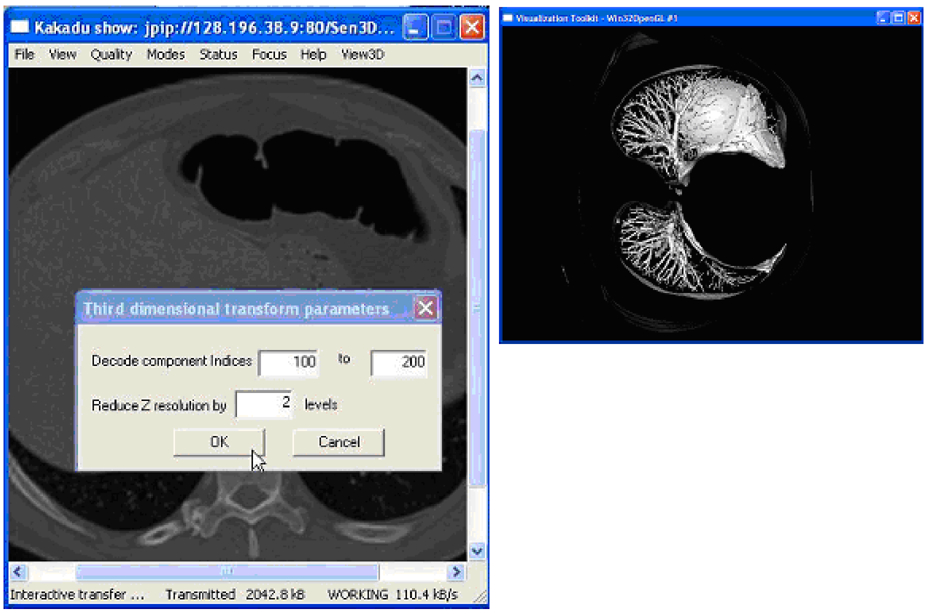


(b) 60%x60%x20% subsection of the volume.



(c) 20%x20%x20% subsection of the volume.

Figure 4. Decoding times in seconds vs. bit rate for code-block sizes of 4, 8, 16, 32 and 64.


Kakadu show: jpip://128.196.38.9:80/Sen3D...
File View Quality Modes Status Focus Help View3D
OK
Cancel
Visualization Toolkit - Win32OpenGL #1

Third dimensional transform parameters

Decode component Indices 100 to 200

Reduce Z resolution by 2 levels

Interactive transfer ... Transmitted 2042.8 kB WORKING 110.4 kB/s

Figure 5. Sample output from the application, developed using Kakadu JPEG 2000 source code and Visualization toolkit.

progression representing the outermost loop and the last representing the innermost loop. It is typical for an interactive user viewing massive volume data to request for a low resolution version of the volume to identify his VOI and then request a more precise sub-section of the volume, this time at a higher resolution. A progression order, such as RPCL, which is primarily progression by resolution, followed by position, makes it easier for the server to extract the proper data to serve such a user's needs.

5. CONCLUSIONS

To our knowledge this is the first scheme enabling interactive compressed domain visualization of volume data using the underlying capabilities of JPEG2000, which are ideal for a resolution scalable and distortion scalable framework. We believe that the method would facilitate visualization of massive volume data on low-end workstations, which would otherwise be impossible to load in main memory. The entire volume can easily be rendered at reduced resolutions. In our current implementation, the user can interactively select any VOI and view it at any desired resolution and then render the voxel data (See Fig. 5).

REFERENCES

1. B. Yeo, B. Liu,"Volume rendering of DCT-based compressed 3D scalar data," *IEEE Trans. Visualization and Computer Graphics*, vol. 1, no. 1, pp. 2943, Mar. 1995.
2. I. Ihm, S. Park, "Wavelet-based 3D compression scheme for interactive visualization of very large volume data," *Computer Graphics Forum*, vol. 18, No. 1, pp. 3-15, Mar. 1999.
3. P. Schelkens, A. Munteanu, J. Barbarien, M. Galca, X. Giro-Nieto, J. Cornelis, "Wavelet coding of volumetric medical datasets," *IEEE Trans. Medical Imaging*, vol. 22, no. 3, pp. 441-458, Mar. 2003.
4. Y. Kim, W. Pearlman, "Lossless volumetric medical image compression," *Proc. of SPIE, Applications of Digital Image Processing*, vol. 3808, pp. 305-312, Oct. 1999
5. D.Taubman, R. Leung, A. Secker, "Scalable compression of volumetric images," *Proc. IEEE Intl. Conf. Image Processing*, vol. 2, pp. 22-25, Sept. 2002.
6. L. Lippert, M. H. Gross, C. Kurmann, "Compression domain volume rendering for distributed environments," *Proc. Eurographics '97, COMPUTER GRAPHICS forum*, vol. 14, no. 3, 1997.

7. D. Taubman, M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice.* Boston: Kluwer Academic Publishers, 2002.
8. D. Taubman, and R. Prandolini, “Architecture, Philosophy and Performance of JPIP: Internet Protocol Standard for JPEG2000,” *International Symposium on Visual Communications and Image Processing*, 2003
9. D. Taubman, “High performance scalable image compression with EBCOT,” *Proc. IEEE Int. Conf. Image Proc.*, vol. 3, pp. 344-348, Sept 2002.