Scalable Low Complexity Image Coder For Remote Volume visualization

Hariharan G. Lalgudi[1], Michael W. Marcellin[1], Ali Bilgin[1,2] and Mariappan S. Nadar[3]

[1]Signal Processing and Coding Lab, Department of Electrical and Computing Engineering, The University of Arizona, Tucson, AZ, USA;
[2]Department of Radiology, The University of Arizona, Tucson, AZ, USA;
[3]Siemens Corporate Research, Imaging and Visualization Department, Princeton, NJ, USA

ABSTRACT

Remote visualization of volumetric data has gained importance over the past few years in order to realize the full potential of tele-radiology. Volume rendering is a computationally intensive process, often requiring hardware acceleration to achieve real time visualization. Hence a remote visualization model that is well-suited for high speed networks would be to transmit rendered images from the server (with dedicated hardware) based on view point requests from clients. In this regard, a compression scheme for the rendered images is vital for efficient utilization of the server-client bandwidth. Also, the complexity of the decompressor should be considered so that a low end client workstation can decode images at the desired frame rate. We present a scalable low complexity image coder that has good compression efficiency and high throughput.

Keywords: remote visualization, image compression.

1. INTRODUCTION

With tremendous advances in medical imaging modalities, efficient storage and transmission of digital imagery is critical. In this regard, it is necessary to consider remote visualization of volumetric datasets to realize the full potential of telemedicine. One way to achieve this would be to transmit volumetric data to the client where it is rendered for visualization. Volume rendering[1] is a computationally intensive process often requiring hardware acceleration for high quality, real time rendering. Strategies have been developed that trade accuracy for speed-up to achieve interactivity using software implementations.[2–4] Thus it is likely that the quality and frame rate would be limited in a low cost client side rendering system. Another remote visualization model would be to do the rendering at a PACS server with dedicated hardware. This method has been investigated,[5, 6] and it has been shown that real time viewing experience is achievable with a high speed server to client connection. Generic compression schemes such as LZO,[7] ZLIB[8] and color cell compression[9] compression have been examined.[6] In this work, we present a scalable, fast and efficient image codec designed for interactive transmission of 2D rendered images to a client.

The paper is organized as follows. Section 2 describes the remote visualization model. In Section 3, we present our Scalable Low Complexity Coder (SLCC). In Section 4, we present compression and throughput performance.

2. REMOTE VOLUME RENDERING

Fig. 1 shows the client-server communication system. All the steps in the volume rendering pipeline are executed at the server using dedicated hardware. Based on view point requests from a client, the server transmits the sequence of 2D rendered images interactively. With unconstrained bandwidth, raw images can be sent to get lossless real time viewing of the rendered images. In the case of limited bandwidth, an efficient compression scheme is vital for transmitting high quality rendered images. In addition to efficiency, the complexity of the decompressor (at the client) should be low so that the transmitted data can be decompressed at a desired frame rate (25-30 frames per second).

This work was supported in part by Siemens Corporate Research.

JPEG2000, the current international standard for 2D image compression,[10, 11] offers superior compression performance and highly scalable codestream. However, JPEG2000 present decoding speed may be a bottleneck for achieving high quality real time (25-30 frames/sec) visualization. Hence, we present a 'JPEG2000-like' system that sacrifices compression efficiency to give increased throughput of encoder and decoder.[*] SLCC has 2 quality layers. The server may transmit only the first layer during an interactive session. Once the interaction stops, the second layer can be sent to give lossless representation of the image at that particular view point.

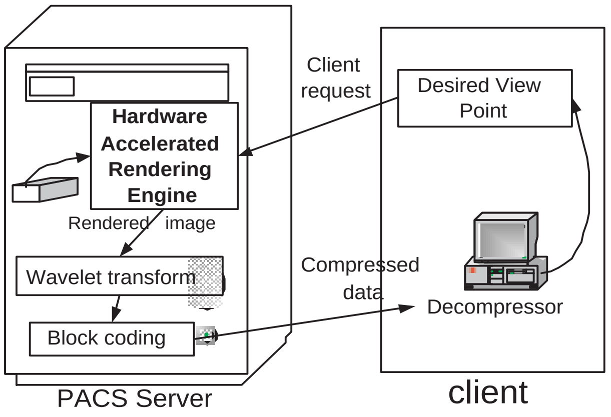


Figure 1. Client-Server communication.

3. SCALABLE LOW COMPLEXITY CODER

Fig. 2 gives the encoding architecture of SLCC. The input image samples are subjected to 2-D dyadic Discrete Wavelet Transform (DWT). Specifically, the image is first decomposed into 4 subbands: LL, HL, LH and HH. The process is repeated on the LL band successively to create subbands at different levels. Subbands resulting from a 3 level wavelet decomposition are illustrated in Fig. 3. The subscript in the subband notation denotes the number of levels. Each subband is divided into codeblocks which are compressed indepedently by a block coder.

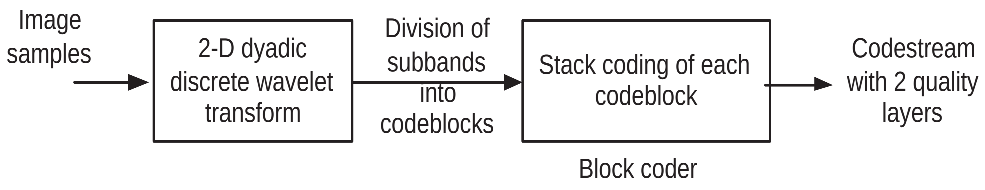


Image samples
2-D dyadic discrete wavelet transform
Division of subbands into codeblocks
Stack coding of each codeblock
Block coder
Codestream with 2 quality layers

Figure 2. Schematic of the scalable low complexity coder.

Fig. 4 illustrates the data structure of codeblocks in different subbands. Each codeblock is divided into two parts (layers). For codeblocks in the LL2 subband, all bit-planes above the 4^{th} bit plane are stacked and sent as the first quality layer. The other subbands have fewer bit planes included in the first layer due to their lesser importance. The importance of each subband is measured from a MSE point of view, by taking into account the synthesis filter energy weights associated with the inverse wavelet transform. These energy weights are rounded to the nearest power of two and used to adjust the stack lengths. In the example of Fig. 4, codeblocks belonging to the HL2 and LH2 subbands will have one less bit plane in the first layer while the HH2 and level-1 subbands have two less. All-zero bit planes in a codeblock are termed as missing MSBs and are indicated in the header information. The stack of bit-planes (discounting missing MSBs) from each codeblock contributing to a layer

[*]At the time of submission for this manuscript, we have just learned of a software only JPEG2000 implementation capable of real-time HD compression and decompression, on modest PC platforms. Such implementation may eliminate the need for systems such as SLCC

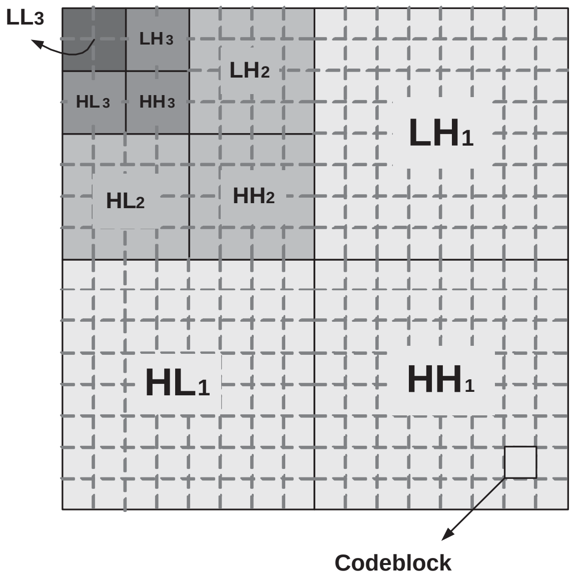


Figure 3. Image samples subjected to three levels of wavelet transform. Each subband is divided into codeblocks.

is coded in one single pass. The coding scheme employed depends on the stack length of the codeblock and is shown in Fig. 5a. Entropy coding is restricted to the first three (or less) bit-planes in the first layer. When there is one bit-plane, the position indices of 'ones' in that bit-plane are coded. Run-value and Quad-Comma coding are used for stack lenghts of 2 and 3 respectively. These two techniques are described below. For codeblocks with more than 3 bit planes, Quad-Comma coding is used for the three MSBs and raw bits are coded for the remaining bitplanes.

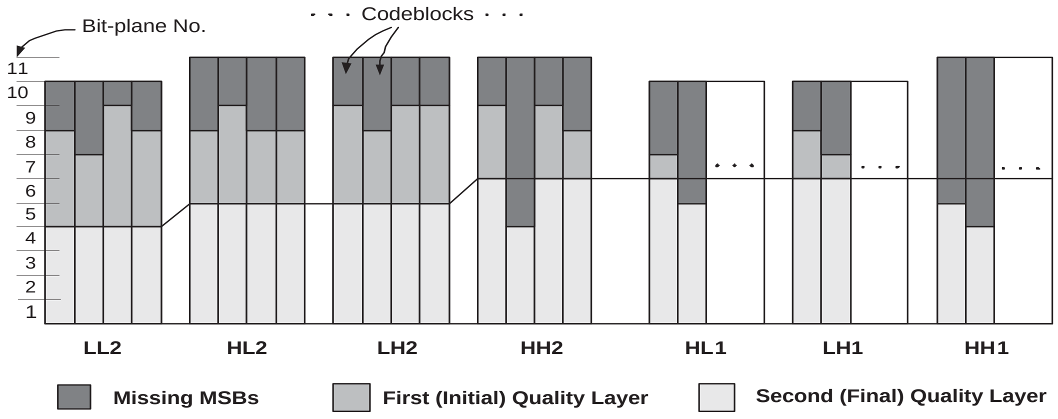
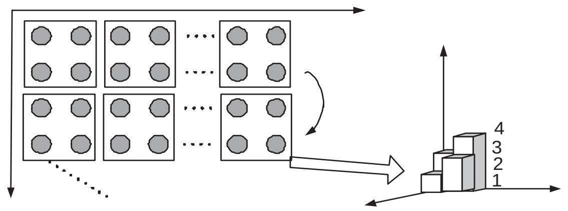


Figure 4. Wavelet coefficient structure.

Run-value coding is chosen here on the assumption of a sparse distribution of non-zero values. The run length of zeros and value of the significant coefficient terminating the run are coded. Additional gain is obtained by coding values of three consecutive coefficients after the zero run. This is because neighbors of a significant coefficient have higher probability of being significant. With Quad-Comma coding (Fig. 5b), one bit is first spent to indicate the significance of a quad. Based on statistical experiments, coefficient values in a significant quad were found to follow a distribution that is close to geometric with parameter $\rho \leq 0.5$. Comma codes[12] are optimum for such geometric distributions and hence are used to code each of the four coefficients in a significant quad. For both the coding schemes described above, the sign bit is appended to the value of the significant coefficient.

Stack length of a codeblock in the first layer	Method
= 1	Position index of significant coefficients
= 2	Run value coding
= 3	Quad-Comma codes
> 3	Quad-comma codes for 3 MSBs and raw bits for remaining bit-planes

(a) Coding scheme decisions



(b) Quad coding with comma codes

Figure 5. Low complexity entropy coding schemes

Using the two layer scheme of SLCC described above, the server may transmit only the first layer during an interactive session. Once the interaction stops, the second layer can be sent to give lossless representation of the image at that particular view point. The number of bit-planes of the first layer can be adjusted based on a desired bit rate, which may be computed from available bandwidth and desired frame rate.

4. RESULTS

Volume rendering typically produces RGB color images. In this work, the RGB color components are compressed without the use of a decorrelating transform. The YCbCr transform that is generally used for natural RGB images did not yield any improvement for our test images. Experimental results are presented with 2 test sequences. These 2 sequences were obtained using different rendering parameters. Each sequence has 30 frames and the frame size is 600×600. Fig. 10(a) shows the 1^{st} frame in test sequence 1. The throughput performance of SLCC is compared with kakadu V5.0 - an efficient JPEG2000 software[13] implementation (see footnote on pg 2). Timing experiments were carried out on a PC with 'Intel Core2 Duo T5470 1.6GHz' processor and 2GB RAM.

The end-to-end decompression time consists of reading the compressed data from memory, block decoding, inverse 2-D DWT and writing the decompressed image to display. The end-to-end decompression time of SLCC is compared to kakadu V5.0 in Fig. 6 and Fig. 7 for test sequence 1 and test sequence 2, respectively. The fastest mode of JPEG2000, referred as the 'bypass' mode is used.[11] As seen from the figures, 2-4 times speed-up is obtained with SLCC at moderate to high rates.

Table 1 and Table 2 show the compression performance of SLCC and kakadu V5.0 for test sequence 1 and test sequence 2 respectively. As noted from the tables, SLCC has 1.1 to 3.1 dB PSNR loss for sequence 1 and 1.1 to 2.3 dB PSNR loss for sequence 2, compared to kakadu V5.0.

Table 1. Compression performance of test sequence 1.

Rate (in bits/pixel)	0.19	0.39	0.87	1.75	3.61
JPEG2000	30.46	31.50	33.49	36.54	41.69
SLCC	29.38	30.52	32.01	34.58	38.59

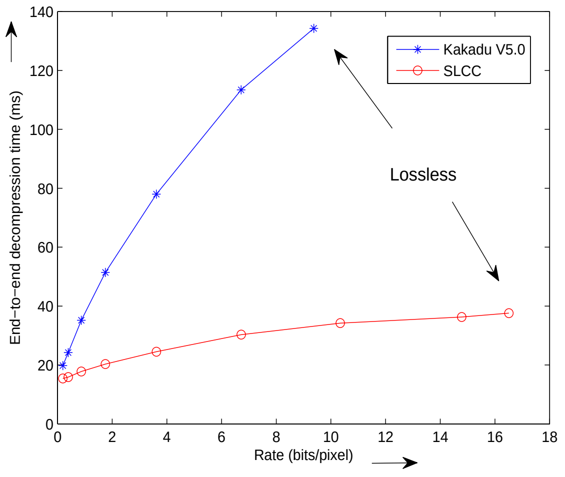


Figure 6. Comparison of end-to-end decompression time for test sequence 1.

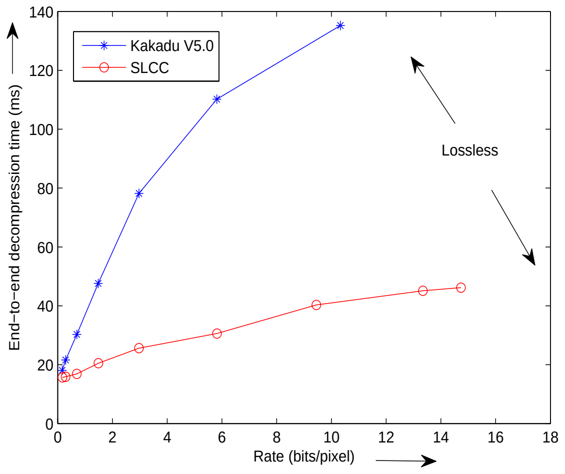


Figure 7. Comparison of end-to-end decompression time for test sequence 2.

Table 2. Compression performance of test sequence 2.

Rate (in bits/pixel)	0.17	0.29	0.70	1.48	2.97
JPEG2000	30.72	31.54	33.29	36.21	40.43
SLCC	29.63	30.71	32.12	34.39	38.14

Fig. 9 compares the quality of decompressed images at the client, under a decoding time constraint, when the frame rate is 30 fps. At this frame rate, the decompressor will have to work with a time constraint of 33ms/frame. In the case of kakadu V5.0, the decompression time reaches this limit at 0.76 bpp (7.8Mbps). Thus the image quality with kakadu V5.0 will be limited to 33.03 dB (PSNR at 0.76 bpp) at all bandwidths greater than 7.8 Mbps. If the available bandwidth exceeds approximately 15 Mbps, SLCC can use the available data to yield higher PSNR than kakadu V5.0. For faster JPEG2000 implementations, this crossover will occur at higher bandwidths. Fig. 10b and Fig. 10c show the decompressed images from kakadu V5.0 and SLCC respectively, when the availble bandwidth is 70 Mbps. As seen from the figure, SLCC can display very high quality images at this bandwidth.

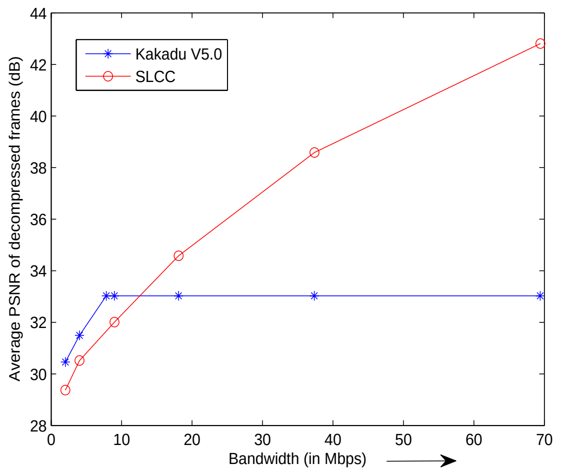


Figure 8. Average PSNR of decompressed images at the client for test sequence 1, under decoding time constraint corresponding to 30 fps.

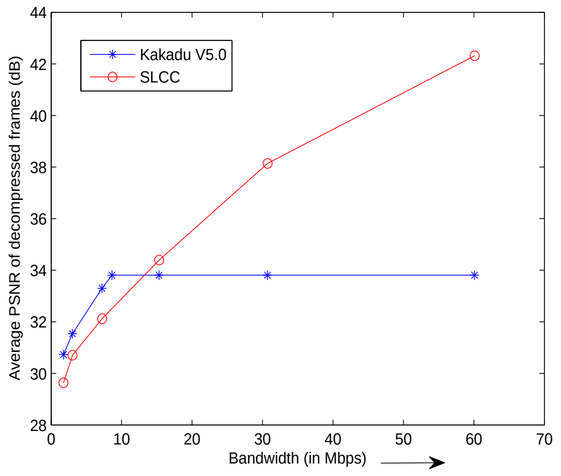
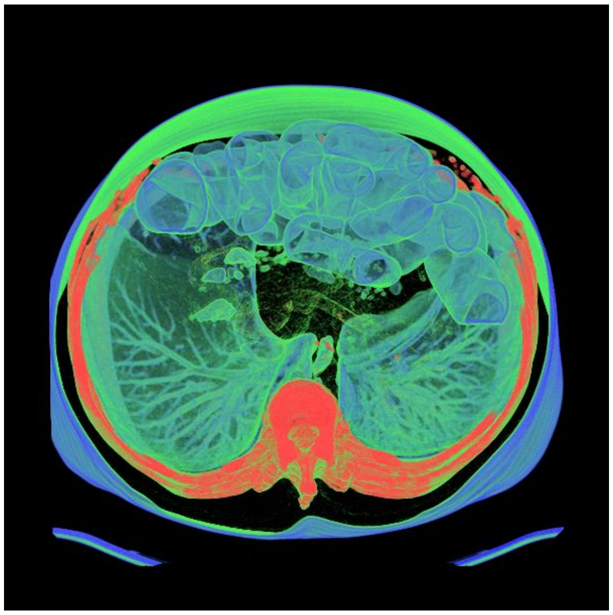


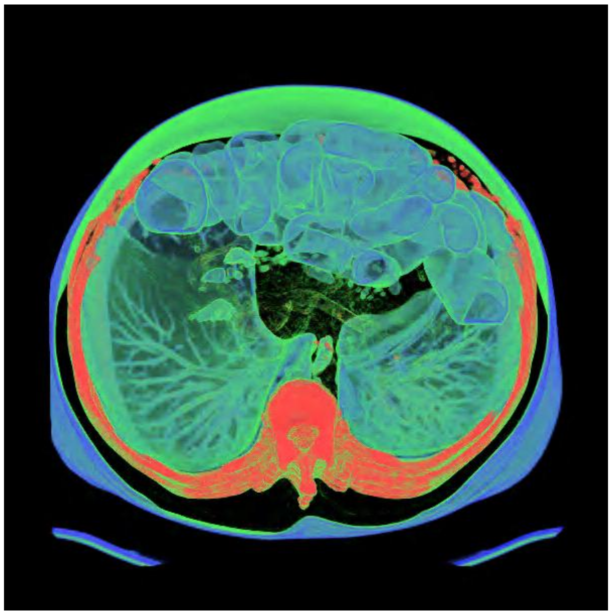
Figure 9. Average PSNR of decompressed images at the client for test sequence 2, under decoding time constraint corresponding to 30 fps.

REFERENCES

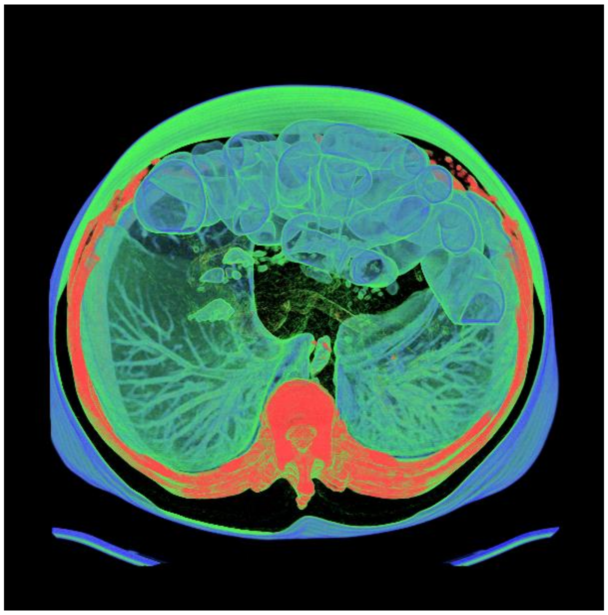
1. M. Levoy, "Efficient ray tracing of volume data," *ACM Transactions on Graphics* **9**(3), July 1990.
2. D. Laur and P. Hanrahan, "Hierarchical splatting: a progressive refinement algorithm for volume rendering," *SIGGRAPH* **25**(4), pp. 285–288, July 1991.
3. J. Schneider and R. Westermann, "Compression domain volume rendering," *IEEE International Conference on Visualization* , pp. 293–300, Oct 2003.
4. I. Ihm and S. Park, "Wavelet-based 3d compression scheme for interactive visualization of very large volume data," *Computer Graphics Forum* **18**(1), pp. 3–15, Mar 1999.



(a) Original image



(b) Decompressed image -- JPEG2000



(c) Decompressed image -- SLCC

Figure 10. Frame 1 of test sequence 1.

5. S. Stegmaier, M. Magalln, and T. Ertl, "Visualization techniques i: A generic solution for hardware-accelerated remote visualization," *Proceedings of the symposium on Data Visualisation* , pp. 87–94, May 2002.
6. S. Stegmaier, J. Diepstraten, M. Weiler, and T. Ertl, "Widening the remote visualization bottleneck," *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis* , pp. 174–179, Sept 2003.

7. M. oberhumer, *http://www.oberhumer.com/opensource/lzo/*.
8. L. Gailly and M. Adler, *http://www.gzip.org/zlib*.
9. G. Campbell, T. A. DeFanti, I. Frederiksen, S. A. Joyce, and L. A. Leske, "Two bidpixel full color encoding," *Proceedings of the 13th annual conference on Computer graphics and inreractive techniques* , pp. 215–223, 1986.
10. *Information technology - JPEG2000 image coding system - Part 1: Core coding system, ISO/IEC 15444-1:2000 Std.*, Jul 2002.
11. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Boston: Kluwer Academic Publishers, 2002.
12. R. G. Gallager and D. C. V. Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Trans. on Information Theory* **21**(2), pp. 228–230, Mar 1975.
13. D. Taubman, *http://www.kakadusoftware.com/*.