Memory Conscious 3D Wavelet Transform

Gregorio Bernabé, José González, José M. García
Dpto. Ing. y Tecnología de Computadores
Universidad de Murcia
30071 Murcia (Spain)
{gbernabe,joseg,jmgarcia}@ditec.um.es

José Duato
Dpto. Inf. de Sistemas y Computadores
Universidad Politécnica de Valencia
46071 Valencia (Spain)
jduato@gap.upv.es

Abstract

The video compression algorithms based on the 3D wavelet transform obtain excellent compression rates at the expense of huge memory requirements, which drastically affect the execution time of such applications. The goal of this work is to mitigate the memory problem by exploiting the memory hierarchy of the processor through blocking. In particular, we present two blocking approaches: cube and rectangular that differ in the way that the original working set is divided. We also propose the reuse of previous computations in order to decrease the number of memory accesses and floating point operations. Results show that the rectangular overlapped approach with computation reuse obtains the best results in terms of execution time, a speedup of 2.42 over the non-blocking non-overlapped wavelet transform, maintaining the compression ratio and the video quality (PSNR) of the original encoder based on the 3D wavelet transform.

1 Introduction

The increase in the volume of medical video generated in hospitals, as well as its strict regulations and quality constraints makes the research in compression techniques especially oriented to this video an interesting area.

In the last few years, the application of the wavelet transform [9] has become an important development. The wavelet transform has been mainly applied to image compression. Several coders have been developed using 2D wavelet transform [3][15][17][20]. The 2D wavelet transform has been used for compressing video [12] as well. However, three dimensional (3D) compression techniques seem to offer better results than two dimensional (2D) compression techniques which operate in each frame independently. Muraki introduced the idea of using 3D wavelet transform to efficiently approximate 3D volumetric data [18][19]. Since one of the three spatial dimensions can be considered similar to time, a 3D subband coding using the zerotree method (EZW) was presented to code video sequences [8] and posteriorly improved with an embedded wavelet video coder using 3D set partitioning in hierarchical trees (SPIHT) [13]. Nowadays, the standard MPEG-4 [4][5] supports an ad-hoc tool for encoding textures and still images based on a wavelet algorithm.

In previous works [6][7], we presented an implementation of a lossy encoder for medical video based on the 3D Fast Wavelet Transform (FWT). This encoder achieves high compression ratios with excellent quality, so that medical doctors cannot find differences between the original and the reconstructed video. However, one of the main drawbacks of using the 3D wavelet transform is its excessive execution time. Since three dimensions are exploited in order to obtain high compression rates, the working set becomes huge and the algorithm becomes limited by memory (memory bound).

In this work, we propose a memory conscious 3D wavelet transform that attempts to exploit the memory hierarchy by means of blocking algorithms, thus reducing the final execution time. Blocking is a well-known optimization technique for improving the effectiveness of memory hierarchies [1][14][16]. Instead of operating on entire rows, columns or frames of the working set, blocked algorithms operate on working subsets or blocks, so that data loaded into the faster levels of the memory hierarchy can be reused. Blocking has been shown to be useful for many algorithms in linear algebra like BLAS [10], LAPACK [2] or more recently ATLAS [21]. In particular, we propose and evaluate several blocking approaches that differ in the way that the original working set is divided. We also propose the reuse of some computations to save floating point operations as well as memory accesses.

Results show that the rectangular partition provides the best execution times, maintaining the compression ratio and the video quality.

The rest of this paper is organized as follows. Section 2 describes our proposed blocking approaches. We present the main details of each method. Experimental Results with some test medical video are analyzed in Section 3. Finally, Section 4 summarizes the work and concludes the paper.

2 Blocking approaches

Previous Wavelet-based encoders obtain excellent results in compression rate and quality (PSNR), as it can be observed in [6][7]. These results were obtained with the 3D-FWT working on video sequences of 64 frames of $512x512$ pixels (16 MBytes of working set). This huge working set limits the performance of such algorithm, making it unfeasible for real-time video transmission. Initial results showed that this algorithm is completely memory bound, therefore, blocking techniques become an interesting approach to reduce its memory requirements and thus the execution time. The goal of blocking algorithms is to exploit the locality exhibited by memory references by means of partitioning the initial working set in limited chunks that fit in the different levels of the memory hierarchy. In this way two positive effects appear: in the one hand, memory accesses are accelerated since data are actually at the higher levels of the memory hierarchy (closer to the processor core). On the other hand, traffic between main memory and the processor chip is drastically reduced, obtaining a better use of the bandwidth provided by the baseline computer system.

However, applying blocking algorithms to video coders is a challenge: not only the memory hierarchy must be exploited by means of an optimum data partitioning, but also quality must be preserved. Note that partitioning the working set into independent blocks may lead to unexpected degradations on the quality of the resulting video due to artifacts in the block bounds.

In this section we present two different approaches to the blocking version of the 3D-FWT transform: cube and rectangular, that differ in the way that the original working set is divided.

2.1 Cube approach

In this first approach, we propose to divide the original sequence, for example a video sequence of 64 frames of $512x512$ pixels, into several subcubes as we can see in figure 1, and the wavelet transform is independently applied to each of these subcubes. Regarding the size

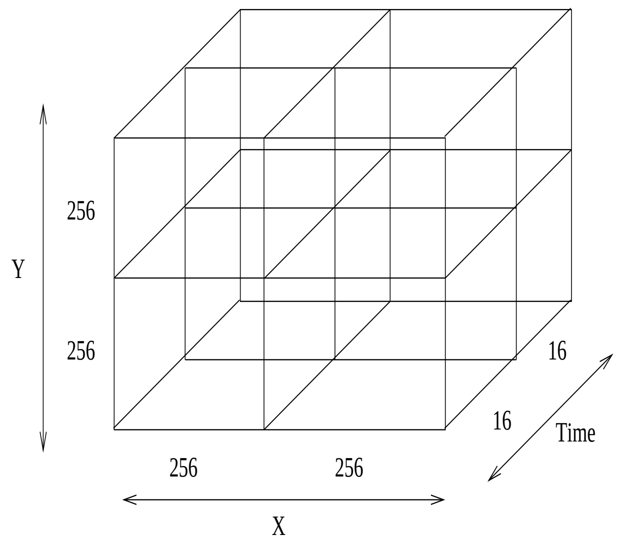

Figure 1. Cube approach

of these subcubes, X and Y axis have the same size (different block sizes are evaluated), whereas the number of frames in the time dimension is fixed to 16, which is the minimum number of frames needed to apply the transform 2 times.

However, this approach presents two disadvantages. First, as the compression ratio increases, the Peak Signal to Noise Ratio (PSNR) drops significantly and, second, it is detected an increasing degree of visibility of the discontinuity in the reconstruction at adjacent subcubes boundaries because artifacts effects appear. This is due to the way that computation is performed in the FWT, where, for a particular pixel, the value of its coefficient after the transform is correlated with the original values of its neighboring pixels.

To illustrate this problem, Figure 2 shows how the wavelet transform is applied for an unidimensional signal of 8 pixels using the Daubechie's of four coefficients as mother function (Daub-4). This signal is divided into two blocks of 4 pixels where the FWT is computed independently. The resulting coefficient for the first pixel depends on the second, third, fourth and itself, all of them belonging to the same block. However, the second pixel depends on the third, fourth, fifth and the sixth pixel (the last two pixels belong to a different block and would not be available in this original partitioning). The same happens for the rest of the pixels. Since additional pixels are needed to compute the transform in any dimension, two different alternatives can be considered to provide this information. $Non-Overlapped$ approaches utilize pixels from the same block (for instance replicating last pixels, or using first pixels). *Overlapped* approaches use pixels from

```
/* c0, c1, c2, c3: Daub-4 coefficients */
/* pixels 1..8 = p[0..7] */
/* temporal vector: low-pass */
float low[8], high[8];
n = 8;
for(i = 0, j = 0; j < (n/2) - 1; i+ = 2, j + +) {

   low[j]=c0*p[i]+c1*p[i+1]+c2*p[i+2]+c3*p[i+3];
  high[j+n/2]=c3*p[i]-c2*p[i+1]+c1*p[i+2]-c0*p[i+3];

}
low[j]=c0*p[n-2]+c1*p[n-1]+c2*p[0]+c3*p[1];
high[j+n/2]=c3*p[n-2]-c2*p[n-1]+c1*p[0]-c0*p[1];
```

Figure 2. Algorithm of 1D FWT with Daub-4

the following block. Although the latter does not seem to exploit memory locality, it provides better compression and quality results as we will later show.

Furthermore, the 3D-FWT implies the computation of the 1D-FWT in the time dimension. Following the aforementioned approach, information from additional frames is needed, which can be obtained from the block itself or from the following blocks. The amount of frames depends on the number of steps of wavelet transform. For example, with the W_4 mother wavelet, applying the wavelet transform just once needs two more frames, six frames are necessary for two wavelet transforms, and, with three wavelet transforms, fourteen frames are needed.

Thus, choosing between the *overlapped* and $non-overlapped$ approaches for the 3D-wavelet transform is one of the main decisions it must be taken to achieve a good trade-off between execution time and quality. Whereas the $non-overlapped$ approach seems more memory efficient, since computations are carried out using the working set of the block, quality of the reconstructed video is clearly affected by the artifacts that appear in the block bounds due to the fact that the coefficients of the block bounds are computed without taking into account their neighbors.

Therefore, in order to avoid the artifacts caused by discontinuities in reconstructions between adjacent coding subcubes, X, Y and time axis are overlapped. We refer to this cube modified approach as *cube overlapped*. Since the FWT is applied twice, six rows, six columns and six frames must be overlapped (e.g. for subcubes of 256 rows-columns of 16 frames, now we will need subcubes of 262 rows-columns of 22 frames).

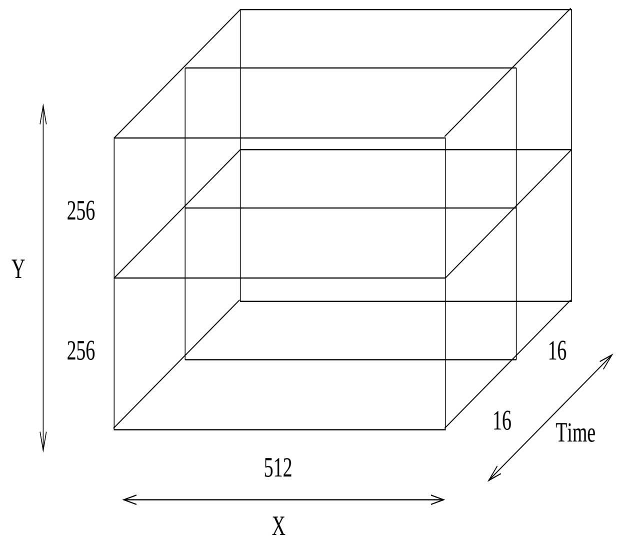

Figure 3. Rectangular approach

2.2 Rectangular approach

The 3D-FWT algorithm is programmed in C and thus frames are stored in memory following a row order. For the space locality of memory references to be better exploited, it may be interesting to analyze a different data distribution. In this section we present the rectangular partitioning, where the original cube is divided into several rectangles, as we can observe in figure 3.

We also apply the $overlapped\ wavelet\ transform$ as in the cube approach, in order to avoid the artifacts and the decrease of PSNR, but only Y and time dimensions are overlapped. For example, a video sequence of 64 frames of $512x512$ pixels can be divided into 8 rectangles of 16 frames of $512x256$ or 32 rectangles of 16 frames of $512x128$ pixels. After overlapping, rectangles of 22 frames of $512x262$ pixels or 22 frames of $512x134$ pixels are obtained.

In this approach we present another contribution of this work: the reuse of some computations in order to reduce the number of floating point operations and memory accesses. As we use the overlapped wavelet transform, operations are repeated across different blocks. For example, for the previous video sequence, divided into 8 rectangles of 16 frames of $512x256$ pixels, in the first rectangle 6 rows and 6 frames must be overlapped. When the first wavelet transform is applied to the Y dimension, 130 low and 130 high rows are obtained. Last two low and high rows are the first ones of the next rectangle, so they should not be computed again in the following block. As it can be seen in figure 4, some computations car-

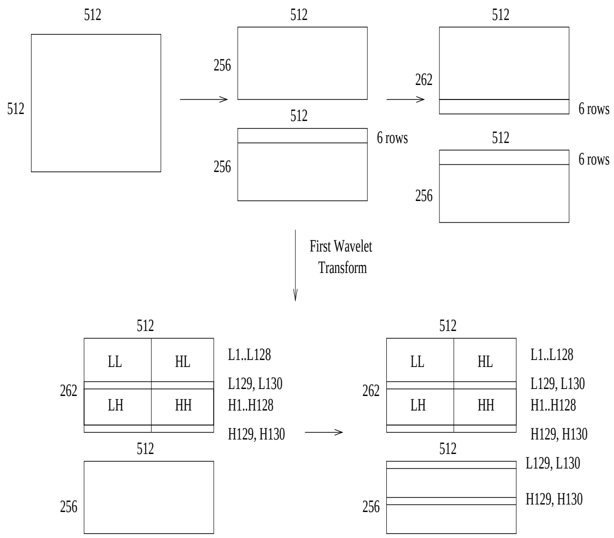

Figure 4. Reuse in Rectangular approach

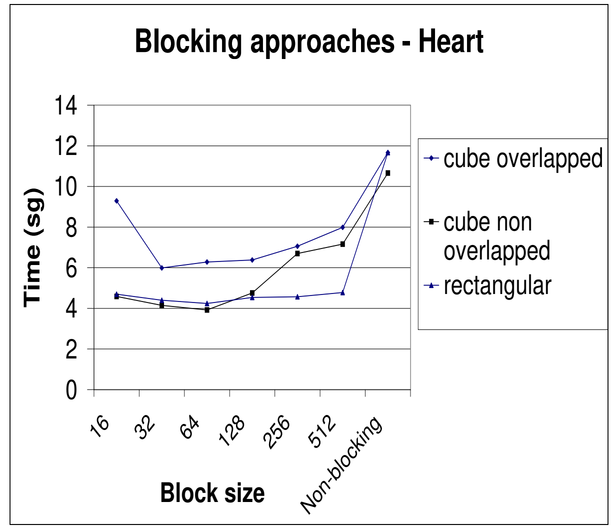

Figure 5. Execution Time of Blocking approaches for Heart video sequence

Level 0	L1 inst TLB, 4K page, 4-way, 32 entries L1 data TLB, 4K page, 4-way, 64 entries
Level 1	L1 inst cache, 16 KB, 4-way, 32 byte line L1 data cache, 16 KB, 4-way, 32 byte line
Level 2	L2 cache, 256 KB, 8-way, 32 byte line
Level 3	512 Mbytes DRAM

Table 1. Description of the memory hierarchy

ried out for the first block are reused for the second block. For instance, if we divide into several rectangles of 16 frames of $512x32$ or $512x16$ pixels, 12% and 25% of the operations will be reused respectively in the Y dimension.

3 Experimental Results

The evaluation has been carried out on a 1GHz-Intel Pentium-III processor with 512 Mbytes of RAM. The main properties of the memory hierarchy are summarized in table 1. The operating system was Linux 2.2.14. The programs have been written in the C programming language.

Our measurements have been made using the performance monitoring counters available in the P6 processor family. The Intel Pentium-series processors include a 64-bit cycle counter, and two 40-bit event counters, with a list of events and additional semantics that depend on the particular processor. We have used a library, Rabbit (v.2.0.1) [11], to read and manipulate Intel processor hardware event counters in C under the Linux operating system.

We have compared execution time consumed by the 3D-wavelet transform for the different blocking approaches and with the original 3D-FWT lossy compression method [6], on a *heart* video medical sequence of 64 frames of $512x512$ pixels coded in gray scale (8 bits per pixel).

Figure 5 shows the execution time obtained with the fast wavelet transform to compute 64 frames of $512x512$ pixels and for the different blocking approaches: cube non-overlapped, cube overlapped and rectangular. Results are presented for different block sizes, from $16x16x16$ to $512x512x16$ in the cube approaches and from $512x16x16$ to $512x512x16$ in the rectangular approach. Also, we have included the execution time without blocking, using the non-overlapped and the overlapped wavelet transform.

First of all, we can observe that blocking approaches clearly reduce the execution time of the original algorithm for all configurations. The optimal block size in the cube non-overlapped approach ($64x64x16$) obtains a speedup of 2.71 over the original non-overlapped wavelet transform, whereas overlapped blocking approaches, cube (optimal block size $32x32x16$) and rectangular (optimal block size $512x64x16$), provide a speedup of 1.77 and 2.42 respectively, compared to the non-overlapped wavelet transform.

As we can see, among the different blocking approaches, the rectangular approach obtains the best results, as we expected. This behavior is due to the better exploitation of locality of its memory accesses

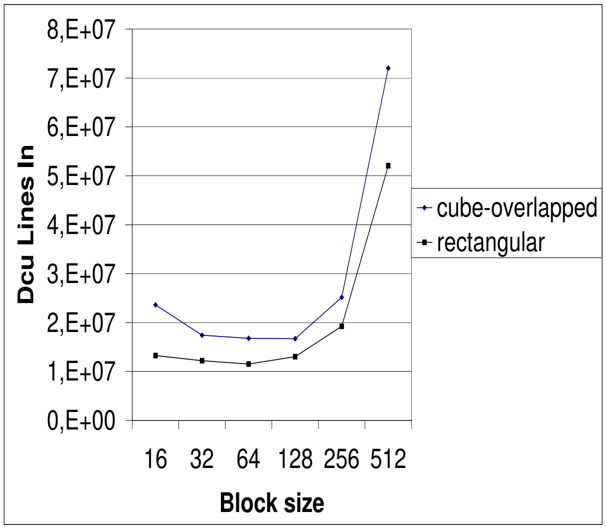

Figure 6. DCU Lines In of Blocking approaches for Heart video sequence

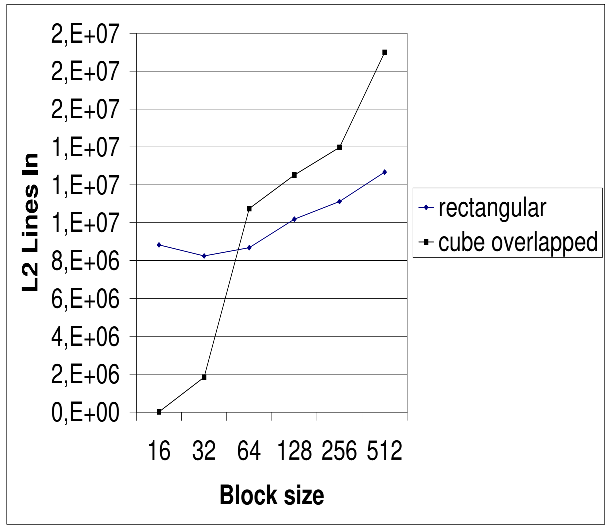

Figure 7. L2 Lines In of Blocking approaches for Heart video sequence

and the reuse of floating point operations. For instance, in the rectangular approach, the optimal configuration is $512x64x16$ which obtain a speedup of 1.48 over the $64x64x16$ in the cube overlapped approach. In some configurations ($16x16x16$, $32x32x16$ and $64x64x16$) the cube non-overlapped approach obtains faster times than the rectangular approach, however this approach presents artifacts and a decrease of the PSNR (around 4 points) in the reconstructed video, that discards it for high quality compression of medical video. On the other hand, overlapped approaches maintain the same compression rate and quality than the non-blocking approaches, which confirm the potential of these methods. Higher execution times on overlapped blocking approaches compared to the non-overlapped ones are due to the increase of the working set of blocks since data from the following blocks must be incorporated. However, these overlapped approaches obtain the best trade-off between performance and quality.

In order to gain some insight about the speedups obtained by blocking approaches, Figures 6 and 7 present the memory cache behavior for the heart video sequence. We measure this behavior using Data Cache Unit (DCU) Lines In and L2 Lines In events of the performance counters, which represent the number of lines allocated in the L1 Data Cache and the L2 cache respectively (i.e. the number of accesses that miss in both caches). It can be observed that the rectangular approach allocates less number of L1 and L2 lines than the cube overlapped approach, justifying the decrease in the execution time. Recall that data are stored by rows, since the rectangular approach keeps more coefficients in a row than the cube approach, spatial locality is better exploited and the number of compulsory misses is drastically reduced.

Summarizing, overlapped approaches maintain the compression rate and the quality of the video whereas the non-overlapped approach produces an unacceptable degree of visibility in the reconstructed video.

Regarding the blocking overlapped approaches, the rectangular one exploits better the memory hierarchy than the cube, and thus the execution time is significantly reduced. The effect of rectangular or cube blocking achieves execution times 12% ($512x64x16$) and 33% ($32x32x16$) faster than blocks of $512x512x16$.

4 Conclusions

In this work, we have focused on reducing the execution time of the 3D-Fast Wavelet Transform when it is applied to code medical video. We have presented two proposals. First, we have developed and evaluated several blocking algorithms in order to exploit the memory hierarchy. Second, we propose the reuse of computations in order to decrease the number of floating point operations and memory accesses. Results show that the rectangular approach obtains the best results, achieving for optimal block size ($512x64x16$) speedups of 2.42 over the non-blocking non-overlapped wavelet transform and 1.48 over the optimal ($64x64x16$) cube overlapped approach. Furthermore, the rectangular overlapped approach maintains the same video quality and the compression ratio of the original encoder.

5 Acknowledgments

This work has been partially supported by the Spanish CICYT under grant TIC2000-1151-C07-03. The video sequences have been donated by the Hospital Recoletas (Albacete, Spain). We are grateful to the reviewers for their valuable comments.

References

[1] N. Ahmed, N. Mateev, and K. Pingali. Tiling imperfectly-nested loop nests. *In Proceedings of Supercomputing*, November 2000.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. M. Kenney, and D. Sorensen. Lapack: A portable linear algebra library for high-performance computers. *Tech. Report CS-90-105, (LAPACK Working Note #20), Univ. of Tennessee, Knoxville*, 1990.

[3] M. Antonini and M. Barlaud. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[4] S. Battista, F. Casalino, and C. Lande. Mpeg-4: A multimedia standard for the third millenium, part 1. *IEEE Multimedia*, 6(4):74–83, October 1999.

[5] S. Battista, F. Casalino, and C. Lande. Mpeg-4: A multimedia standard for the third millenium, part 2. *IEEE Multimedia*, 7(1):76–84, January 2000.

[6] G. Bernabé, J. González, J. M. García, and J. Duato. A new lossy 3-d wavelet transform for high-quality compression of medical video. *Proc. of IEEE EMBS International Conference on Information Technology Applications in Biomedicine*, pages 226–231, November 2000.

[7] G. Bernabé, J. González, J. M. García, and J. Duato. Enhancing the entropy encoder of a 3d-fwt for high-quality compression of medical video. *Proc. of IEEE International Symposium for Intelligent Signal Processing and Communication Systems*, November 2001.

[8] Y. Chen and W. A. Pearlman. Three-dimensional subband coding of video using the zero-tree method. *Proc. of SPIE-Visual Communications and Image Processing*, pages 1302–1310, March 1996.

[9] I. Daubechies. *Ten Lectures on Wavelets.* Society for Industrial and Applied Mathematics, 1992.

[10] J. Dongarra, J. D. Croz, I. S. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprogram. *ACM Trans. Math. Soft*, 14:1–17, 1988.

[11] D. Heller. Rabbit: A perfomance counters library for intel/amd processors and linux. *Available at http://www.scl.ameslab.gov/Projects/Rabbit/.*

[12] M. L. Hilton, B. D. Jawerth, and A. Sengupta. Compressing still and moving images with wavelets. *Multimedia Systems*, 2(3), 1994.

[13] B.-J. Kim and W. A. Pearlman. An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (spiht). *Proceedings of Data Compression Conference*, 1997.

[14] M. S. Lam, E. E. Rothberg, and M. E. Wolf. The cache perfomance and optimizations of blocked algorithms. *Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IV)*, April 1991.

[15] A. S. Lewis and G. Knowles. Image compression using the 2-d wavelet transform. *IEEE Transactions on Image Processing*, 1(2):244–256, April 1992.

[16] A. W. Lim, S.-W. Liao, and M. S. Lam. Blocking and array contraction across arbitrarily nested loops using affine patitioning. *In Proceedings of the 8th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, June 2001.

[17] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of jpeg-2000. *Proceedings of Data Compression Conference*, March 2000.

[18] S. Muraki. Approximation and rendering of volume data using wavelet transforms. *Proceedings of Visualization*, pages 21–28, October 1992.

[19] S. Muraki. Multiscale volume representation by a dog wavelet. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):109–116, June 1995.

[20] J. M. Shapiro. Embedded image coding using zerotrees of wavelets coefficients. *IEEE Transcations on Signal Processing*, 41(12):3445–3462, December 1993.

[21] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimizations of software and the atlas project. *Parallel Computing*, 27(1-2):3–35, 2001.