Architecture Comparison in a High Bandwidth Application

Rade Kutil, Andreas Uhl

{rkutil,uhl}@cosy.sbg.ac.at
Salzburg University, Dept. of Scientific Computing
Jakob Haringer Str. 2, A-5020 Salzburg, Austria

Abstract

Todays most commonly used architectures in parallel computing are distributed memory multiprocessors, shared memory computers and computer clusters. In this work we compare the performance of the core of a video coding algorithm on all these architectures. This algorithm – the 3-D wavelet transform – is supposed to be representative for a wide range of applications because it includes signal filtering, border data problems and processing/distribution of large arrays of data.

1 Introduction

Image and video coding methods that use wavelet transforms have been successful in providing high rates of compression while maintaining good image quality and have generated much interest in the scientific community as competitors to DCT based compression schemes in the context of the MPEG-4 and JPEG2000 standardization process.

Rate-distortion efficient 3-D algorithms exist which are able to capture temporal redundancies in a more natural way (see e.g. [9, 5, 3, 13] for 3-D wavelet/subband coding). Recently, wavelet packet based compression methods[16, 10] have been developed which outperform the most advanced wavelet coders (e.g. SPIHT[11]) significantly for textured images in terms of rate-distortion performance. A similar development in the area of video compression may be observed for 3-D subband[8] or 3-D wavelet packet[4] video coding.

Unfortunately, these 3-D algorithms often show prohibitive computational and memory demands (especially for real-time applications). At least, prohibitive for a common microprocessor. Therefore, a MIMD architecture seems to be an interesting choice for such an algorithm. We choose three (quite contrary) architectures of MIMD computers to investigate the performance of the parallelisation of the 3-D wavelet packet decomposition:

- Distributed memory multiprocessors
- Shared memory multiprocessors
- Networks of common single or dual processor computers (clusters)

2 Sequential Algorithm

In the 3-D case the decomposition of a subband produces eight new subbands by convolving the data with a low pass and a high pass QMF filter along each of the three dimensions. In pyramidal wavelet transforms only one subband (the coarse scale approximation) is decomposed further whereas in complete wavelet packet decompositions each subband is decomposed, leading to a so called full decomposition tree which produces 8^n subbands at a decomposition depth of n.

As we want to use the transformed data for compression purposes, we have to estimate how suitable it is for this purpose by evaluating an information cost function (like e.g. entropy).

To determine the best basis one has to perform a complete decomposition and to compute for each subband on each depth its *best information cost* by comparing its *information cost* with the sum of its subbands *best information costs*. After that the subbands that minimize the *overall information cost* have to be selected by either accessing subband data kept in memory (high memory requirements) or performing a second decomposition run according to the best basis tree. Here we use a strategy that avoids the second run by backing up and restoring subbands and by walking through the decomposition tree in depth first order. Pseudo Code 1 implements this strategy as a recursive procedure.

```
SequDecompose (Subband S) return InformationCost :=
    C = InformationCost (S)
    – end of recursion:
    if DecompositionLevel (S) = MaxLevel
        return C
    Backup S
    Filter S to get Subbands S_1 ... S_8
    – decompose the subbands recursively:
    D = 0
    for i = 1 ... 8
        D = D + SequDecompose (S_i)
    – decide whether the best basis tree branches:
    if C < D
        Restore S; return C
    else
        return D
```

Pseudo code 1: Sequential Algorithm

3 Parallel Algorithm

We use message passing for the parallel algorithm. The main idea is to split the data into several parts along one or more axes and to distribute them among the PE. But at a certain depth subbands get too small to be processed efficiently in parallel. This stage is of course dependent on the number of processor elements (PE) employed and is denoted "redistribution level"[14, 1]. At this stage data has to be redistributed so that entire subbands are located on single PE which enables to perform the remaining computations without any additional computations. Therefore the concept of the sequential algorithm cannot be completely adapted to the parallel algorithm.

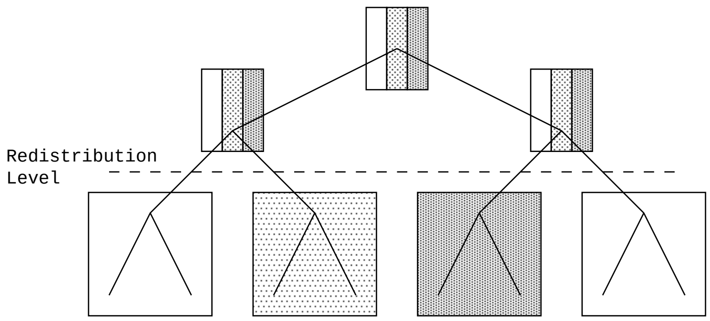


Figure 1: Example data/work distribution at several stages in the decomposition tree (simplified) – different colors symbolize different PE

We choose a mixed approach: Below the redistribution level data is completely decomposed to that level and recomposed if it is necessary to satisfy the best basis (which can be interpreted as partial reverse second run). Redistributed subbands are decomposed using the sequential algorithm from the previous section. Figure 1 shows a simplified (binary instead of octtree) example distribution scheme for three PE. Pseudo code 2 shows, how this can be implemented.

```
– Phase 1: split decomposition
   for t = 1 ... NumberOfNodes
      Send Part of data to Node t
   for l = 1 ... RedistributionLevel
      for all subbands S at level l
         Exchange border data
         Filter S into 8 new subbands
– Phase 2: subband based decomposition
   Redistribute subbands to responsible node
   for all subbands S at RedistributionLevel
      SequDecompose (S)
– Phase 3: partial reverse second run
   Determine part of best basis tree below RedistributionLevel
   Recompose subbands below RedistributionLevel if necessary
```

Pseudo code 2: Parallel Algorithm

The memory requirement of this algorithm for a node PE is therefore $(\frac{t}{\#PE} + (2^r - 1)(f - 2))xy$ floating point numbers for the first phase (t,x and y are the video data sizes along the corresponding axes, f is the filter length and r is the redistribution level) plus $txy8^{-r}(1+b)$ for the second phase (where b stands for the memory needed to backup the subbands before decomposition) which should be rather small for $r > 1$. In our case $b = 1 + \frac{1}{8} + \frac{1}{8^2} + \ldots \approx \frac{8}{7}$. If we performed a second run, backing up the subbands would not be necessary, therefore $b = 0$. The conventional algorithm (storing all subbands) would imply $b = d - r$, where d is the maximum decomposition depth.

Boundary problem A classical problem in parallel wavelet and wavelet packet algorithms is the question of how to handle the exchange of the necessary border data. In the literature two approaches[15, 12] for the boundary problems have been discussed and compared. During the *data swapping* method each PE calculates only its own data and exchanges these results with the appropriate neighbour PE in order to get the necessary border data for the next decomposition level. Employing *redundant data calculation* each PE computes also necessary redundant border data in order to avoid additional communication to obtain this data. It was shown that the decision, which method to apply, is very important for certain wavelet algorithms (e.g. the á trous algorithm [2] and 3-D pyramidal fast wavelet decomposition [6]). Therefore we investigate its influence on the wavelet packet decomposition.

4 Experimental Results

As environment for experimental results the following computer architectures were used:

Cray T3E (Distributed memory multiprocessor) Located at NIC Jülich. 512 compute nodes type 1200, cycle time 1.66 ns, 1200 MFLOPS/processor. Message passing library: Native MPI implementation.

Siemens Cluster (Cluster of dual processor computers) A hpcLine-Cluster at Paderborn Center for Parallel Computing. 96 Siemens Primergy compute nodes, each with 2 Intel Pentium II, 450 MHz, 512 MByte memory and 4 GByte local hard disk. 96 Dolphin PCI/SCI interfaces with 500 MByte/s SCI-Link bandwidth to form a 2D 12x8 torus with distributed switches. Message passing library: Native MPI implementation: ScaMPI release 1.9.1.

SGI Powerchallenge (Shared Memory multiprocessor) SGI Powerchallenge GR (at RIST++, Salzburg Univ.) with 20 MIPS R10000 processors and 2.5 GB shared memory. Message passing libraries: SGI MPI 3.1.1.0

4.1 Full Wavelet Packet Decomposition

Data sizes in use are 256 in each direction on the Cray T3E and the SGI Powerchallenge and 128 in each direction on the Siemens cluster.

Figures 2 (a,c,e) show the speedups for different redistribution levels. One can see clearly that a too low redistribution level limits the number of subbands that are redistributed, leaving some PE without work (load balancing problem). On the other hand, a too high redistribution level increases the communication complexity of the redistribution.

On the Cray T3E the speedup is nearly ideal up to 32 PE. After that it remains constant which indicates that a bandwidth limit is reached at this point. Although the Siemens cluster cannot reach as good speedups as the T3E its performance is competitive and the speedup does not show a bandwidth limit except for too high redistribution levels. Interestingly, the SGI Powerchallenge has its best performance at a high redistribution level. The reason might be that shared memory architectures can handle bigger numbers of smaller data packets more easily. (The performance breakdown 16 PE and above is probably an MPI implementation problem as this does not occur with PVM.)

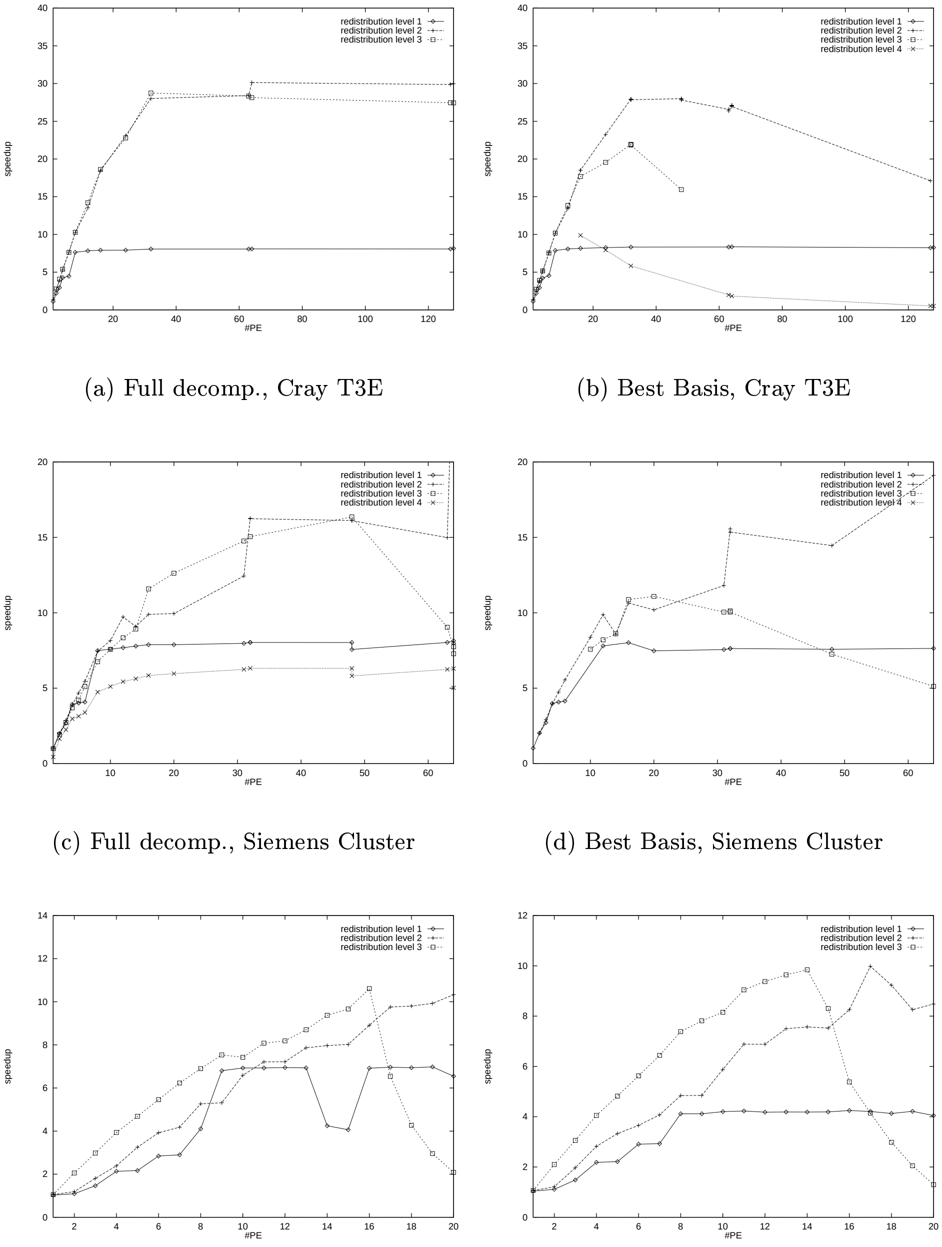
4.2 Best Basis

Figures 2 (b,d,f) show the speedups for the best basis decomposition.

The results are very similar to that with full wavelet packet decomposition. The reason is that the computational demand for each subband is increased in the same way over all decomposition levels, and the only additional communications are the exchange of subband statistics and subtree information which is negligible compared to the exchange of subband data itself.

4.3 Effect of Border Data Handling on Best Basis Decomposition

Figure 3 shows that the influence of border handling on the performance is much smaller than in the pyramidal case (compare [7]). The reason is that in the pyramidal case the entire processing time is dominated by the first decomposition level (where the choice of border handling strategy is crucial) whereas in the wavelet packet case all decomposition levels require roughly the same computational amount. The redundancy level in Figure 3



(a) Full decomp., Cray T3E

(b) Best Basis, Cray T3E

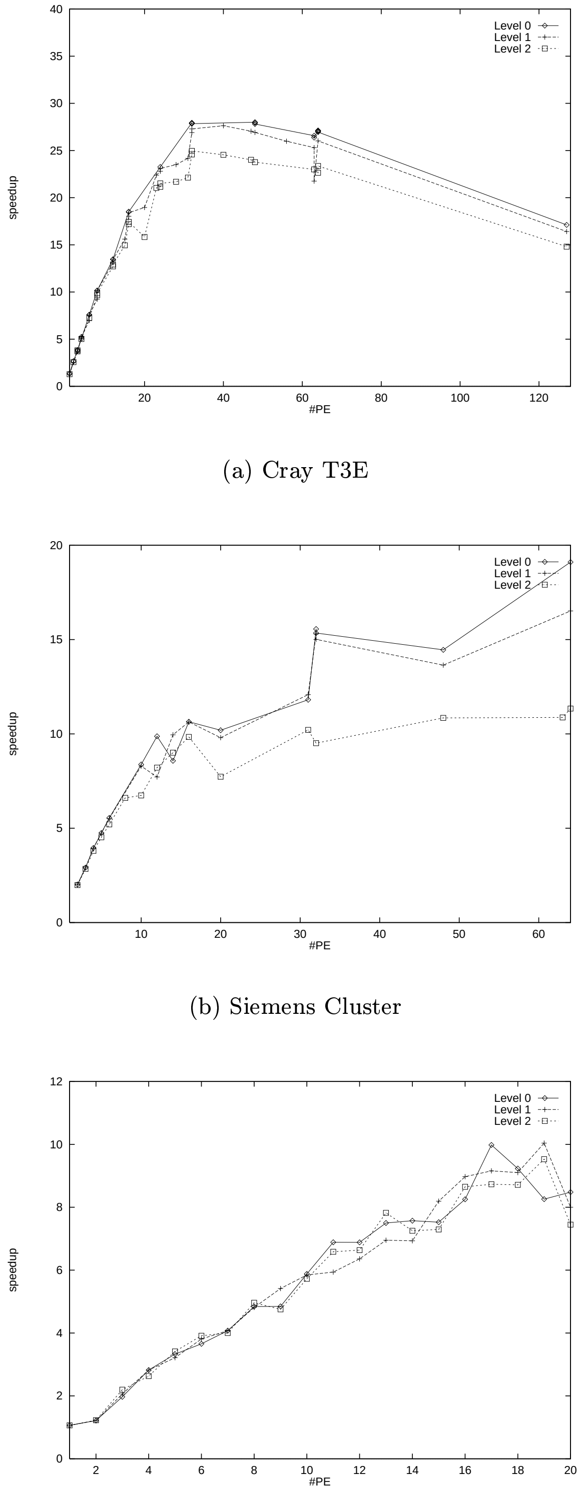
(c) Full decomp., Siemens Cluster

(d) Best Basis, Siemens Cluster

(e) Full decomp., SGI Powerchallenge

(f) Best Basis, SGI Powerchallenge

Figure 2: Speedups for full wavelet packet and best basis decomposition for different redistribution levels



(a) Cray T3E

(b) Siemens Cluster

(c) SGI Powerchallenge

Figure 3: Speedups of best basis decomposition for different levels of redundancy

means “data swapping only” if level is 0, “data swapping after the first decomposition step” if level is 1 and so on.

The Siemens cluster obviously is most sensitive to the border handling. The reasons might be that a high band-width is necessary in the initial data distribution phase and the interconnection structure (2-D mesh) is not optimal for the star-shaped communication structure.

4.4 Scalability

Having now sufficient information about the scalability of our algorithms applied to a fixed data size, we finally investigate the effect of varying the size of the video data in the spatial (just imagine the difference between QCIF and SHDTV) as well as in the temporal domain (it might be desirable to use (temporal) smaller blocks in order to keep the coding delay to a minimum in a real-time application).

Figures 4 (a,c,e) show execution times of the pyramidal decomposition. Data sizes are kept in cubic shape if possible. The straight line labeled “ideal scalability” is intended to give a comparison to a fictitious program that requires exactly an n-fold execution time to process n-fold data sizes. If the curve for our algorithm is steeper it means that the algorithm performs worse for bigger data sizes. If the curve is flat the algorithm performs worse for smaller data sizes.

One can see that for big enough data sizes the curves are nearly parallel to the ideal scalability. The flat parts for higher #PE and small data sizes can easily be explained because communication as well as computation times get very small, so, message latency and other communication overhead gets dominant. On the Siemens Cluster the execution times increase rapidly at data size of 128 MB. Obviously this is due to memory problems (although each node owns 512 MB of memory). The author could not find the actual source of this problem.

Figures 4 (b,d,f) show the equivalent plots for the best basis decomposition. Of course the execution times are higher than in the pyramidal case, but the overall behaviour is the same. Only the effects at small data sizes are more significant, because of more complex communication structures.

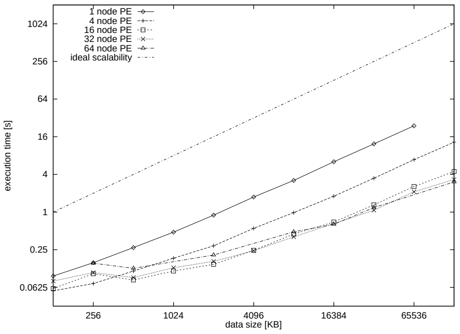
If equal #PE are compared the three architectures show approximately the same scalability.

References

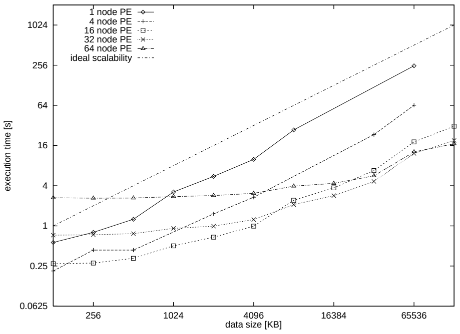
[1] M. Feil and A. Uhl. Algorithms and programming paradigms for 2-D wavelet packet decomposition on multicomputers and multiprocessors. In P. Zinterhof, M. Vajtersic, and A. Uhl, editors, *Parallel Computation. Proceedings of ACPC'99*, volume 1557 of *Lecture Notes on Computer Science*, pages 367–376. Springer-Verlag, 1999.

[2] M. Feil and A. Uhl. Real-time image analysis using wavelets: the “à trous” algorithm on MIMD architectures. In D. Sinha, editor, *Real-Time Imaging IV*, volume 3645 of *SPIE Proceedings*, pages 56–65, 1999.

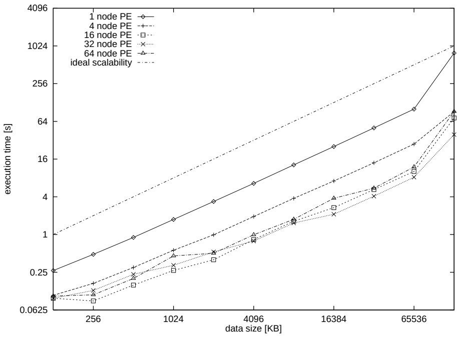
[3] K.H. Goh, J.J. Soraghan, and T.S. Durrani. New 3-D wavelet transform coding algorithm for image sequences. *Electronics Letters*, 29(4):401–402, 1993.



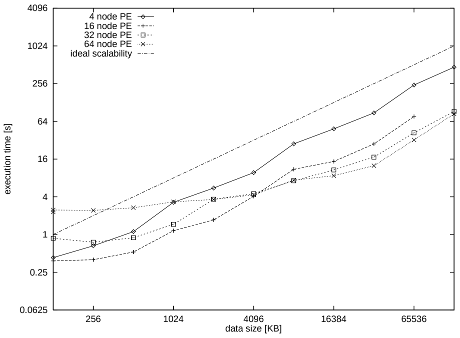
(a) Pyramidal, Cray T3E



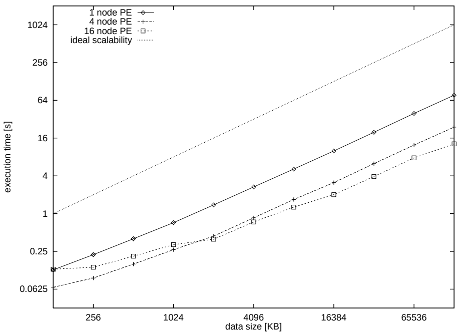
(b) Wavelet packet, Cray T3E



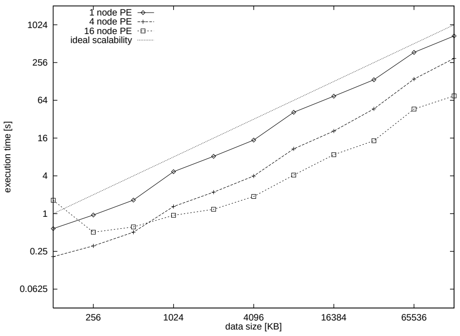
(c) Pyramidal, Siemens Cluster



(d) Wavelet packet, Siemens Cluster



(e) Pyramidal, SGI Powerchallenge



(f) Wavelet packet, SGI Powerchallenge

Figure 4: Scalability of pyramidal and best basis decomposition

[4] W.L. Hsu and H. Derin. Video compression using adaptive wavelet packets and DPCM. In H.H. Li, S. Sun, and H. Derin, editors, *Video Data Compression for Multimedia Computing*, pages 55–94. Kluwer Academic Publishers Group, 1997.

[5] B.J. Kim and W.A. Pearlman. An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT). In *Proceedings Data Compression Conference (DCC'97)*, pages 251–259. IEEE Computer Society Press, March 1997.

[6] R. Kutil and A. Uhl. Optimization of 3-d wavelet decomposition on multiprocessors. *Journal of Computing and Information Technology (Special Issue on Parallel Numerics and Parallel Computing in Image Processing, Video Processing, and Multimedia)*, 8(1):31–40, 2000.

[7] R. Kutil and A. Uhl. Parallel adaptive 3-d wavelet analysis for fast and efficient video coding. In E.H. D'Hollander, G.R. Joubert, F.J. Peters, and H.J. Sips, editors, *Parallel Computing: Fundamentals and Applications. Proceedings of the International Conference ParCo 1999*, pages 316–323. Imperial College Press, 2000.

[8] R.L. Lagendijk, F. Bosveld, and J. Biemond. Subband video coding. In A.N. Akansu and M.J.T. Smith, editors, *Subband and Wavelet Transforms*, pages 251–286. Kluwer Academic Publishers Group, 1996.

[9] A.S. Lewis and G. Knowles. Video compression using 3D wavelet transforms. *Electronics Letters*, 26(6):396–398, 1990.

[10] F.G. Meyer, A.Z. Averbuch, J.O. Strömberg, and R.R. Coifman. Fast wavelet packet image compression. In J.A. Storer and M. Cohn, editors, *Proceedings of the Data Compression Conference DCC'98*, page 563. IEEE Press, March 1998.

[11] A. Said and W.A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–249, 1996.

[12] S. Sullivan. Vector and parallel implementations of the wavelet transform. Technical report, Center for Supercomputing Research and Development, University of Illinois, Urbana, 1991.

[13] D. Taubman and A. Zakhor. Multirate 3-D subband coding of video. *IEEE Transactions on Image Processing*, 5(3):572–588, September 1993.

[14] A. Uhl. Wavelet packet best basis selection on moderate parallel MIMD architectures. *Parallel Computing*, 22(1):149–158, 1996.

[15] M-L. Woo. Parallel discrete wavelet transform on the Paragon MIMD machine. In R.S. Schreiber et al., editor, *Proceedings of the seventh SIAM conference on parallel processing for scientific computing*, pages 3–8, 1995.

[16] Z. Xiong, K. Ramchandran, and M.T. Orchard. Wavelet packet image coding using space-frequency quantization. *IEEE Transactions on Image Processing*, 7(6):892–898, June 1998.