PARALLEL ADAPTIVE 3-D WAVELET ANALYSIS FOR FAST AND EFFICIENT VIDEO CODING

R. KUTIL*AND A. UHL

RIST++ & Dept. of Scientific Computing, University of Salzburg, Austria

E-mail: {rkutil,uhl}@cosy.sbg.ac.at

Adapted wavelet analysis in the sense of wavelet packet algorithms is a highly relevant procedure in different types of applications, like e.g. data compression, classification problems, data analysis etc. Given a 3-D (video) data set the computational demand is too high for interactive or "nearly-interactive" processing. Therefore, parallel processing is one of the possibilities to accelerate the processing speed. In the 3-D case special attention has to be paid towards handling of the large amount of data. We investigate different data decomposition approaches, border handling techniques and programming paradigms. The memory consuming decomposition into a given arbitrary basis after adaptive basis choice is resolved by a localized decomposition strategy.

1 Introduction

Wavelet packets[1] represent a generalization of the method of multiresolution decomposition and comprise the entire family of subband coded (tree) decompositions. Whereas in the wavelet case the decomposition is applied recursively to the coarse scale approximations (leading to the well known (pyramidal) wavelet decomposition tree), in the wavelet packet decomposition the recursive procedure is applied to all the coarse scale approximations and detail signals, which leads to a complete wavelet packet tree (i.e. binary tree and quadtree in the 1D and 2D case, respectively) and more flexibility in frequency resolution.

Whereas a significant amount of work has been already done concerning parallel algorithms for the fast wavelet transform, only few papers have been devoted to parallel wavelet packet decomposition and its specific features and demands (e.g. approaches for performing the best basis algorithm and the irregular decomposition into such a basis on parallel MIMD architectures[2,3,4,5], application of parallel wavelet packet decomposition in numerics[6]).

Most video compression algorithms rely on 2-D based schemes employing motion compensation techniques. On the other hand, rate-distortion efficient 3-D algorithms exist which are able to capture temporal redundancies in a

*PARTIALLY SUPPORTED BY THE AUSTRIAN SCIENCE FUND FWF, PROJECT NO. P11045-ÖMA.

more natural way (e.g. 3-D wavelet/subband coding and hybrid 3-D fractal/subband coding).

Recently, wavelet packet based compression methods[7,8] have been developed which outperform the most advanced wavelet coders (e.g. SPIHT[9]) 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[10] or 3-D wavelet packet[11] video coding. Therefore, wavelet packet decomposition currently attracts much attention and a thorough examination of parallelization possibilities for the 3-D case is desirable.

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. An example is the entropy $H(X) = -\sum_j p_j \log p_j$ where $p_j = \frac{|x_j|^2}{||X||^2}$ and $X = (x_0, x_1, x_2, \ldots)$ are the coefficients in the subband. An additive analogue is $\lambda(X) = -\sum_j |x_j|^2 \log |x_j|^2$ by the use of which $\sum_i \lambda(X_i)$ can be compared to $\lambda(X)$ if X is decomposed into $X_1 \ldots X_8$.

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. We present a strategy how to avoid 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 then 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
    then Restore S; return C
    else return D
```

Pseudo code 1: Sequential Algorithm

3 Parallel Algorithm

3.1 Message Passing

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"[4,5]. 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.

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.

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

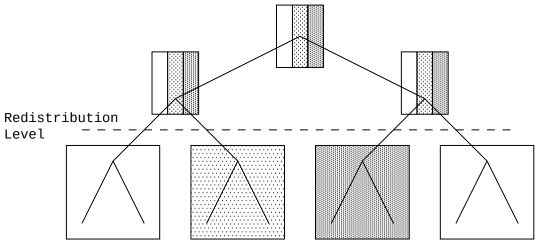


Figure 1. Example data/work distribution at several stages in the decomposition tree (simplified) – different colors symbolize different PE

```
-- 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

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[12,13] 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[14] and Figure 3 (a) for 3-D pyramidal fast wavelet decomposition). Therefore we investigate its influence on the wavelet packet decomposition.

Multidimensional data splitting There are several possibilities to distribute the data in the beginning, because it can be split along several axes. It can be shown, that by splitting the data not only in time domain but also in spatial domain it is possible to reduce the amount of border data that has to be exchanged, but it increases the communication complexity.

3.2 Data Parallel Approach

Data parallel programming on a shared memory architecture is easily achieved by transforming a sequential algorithm into a parallel one by simply identifying areas which are suitable to be run in parallel i.e. in which no data dependencies exist and different iterations access different data.

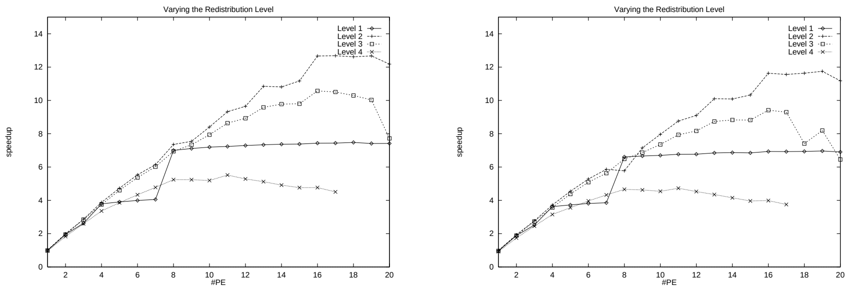
Here we choose parallelization of a decomposition step along the time axis of the video data. Because one has to take care, that parallel regions are not too short with regard to execution time, we have to change this strategy at a certain decomposition depth. This leads to a simulation of the message passing algorithm (data redistribution).

4 Experimental Results

We conduct experiments on an SGI POWERChallenge GR (at RIST++, Salzburg Univ.) with 20 MIPS R10000 processors. The size of the video data is 128×128 pixels in the spatial domain, combined to a 3-D data block consisting of 512 frames. QMF filters with 8 coefficients are used. The PVM version employed is a special shared memory variant for SGI systems.

Figure 2 (a) shows clearly, that a too low redistribution level limits the subbands that are redistributed, leaving some PE without work (load balancing problem). A too high redistribution level increases the communication complexity of the redistribution. This shows, that redistribution based on subbands is necessary.

Comparing this Figure with Figure 2 (b) one can see, that although execution time is about 25% shorter if best basis computations are omitted, the speedup behaviour is the same. The reason is, that the computational demand for each subband is increased in the same way over all decomposition levels

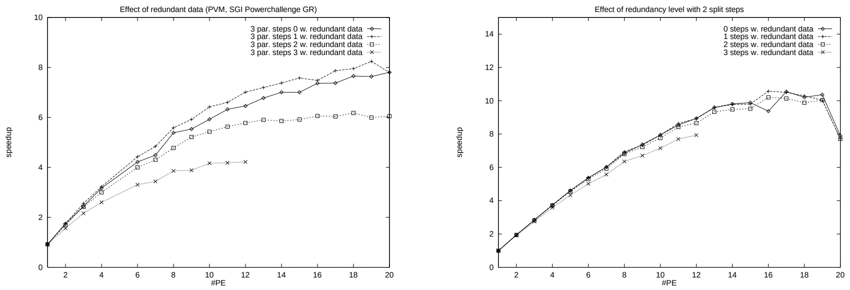


(a) With best basis selection

(b) Without best basis belection

Figure 2. Effect of subband based redistribution

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.



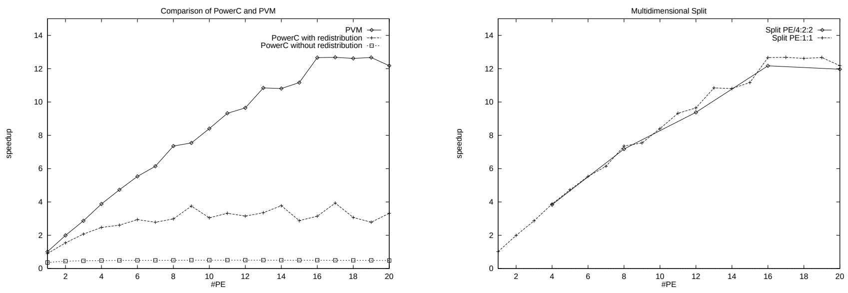
(a) Pyramidal wavelet decomposition

(b) Wavelet packet decomposition

Figure 3. Effect of different data decomposition techniques

Figure 3 shows, that whereas for pyramidal wavelet decomposition (a) the choice of the border handling strategy does affect the performance, it turns out to be of almost no relevance for wavelet packet decomposition (b). ("n steps with redundant data" means, that the first n steps of the decomposition do not have to exchange border data.) The reason is that in the pyramidal case the entire processing time is in contrast to the wavelet packet case dominated by the first decomposition level (where the choice of border handling strategy is crucial). The computations above the redistribution level are not affected by any border problem at all which explains the different behaviour.

For the same reason multidimensional data splitting shows little influence on the speedup, as can be seen in Figure 4 (b). (The notation "p:q:r" means, that the video data is split p times in time domain and q and r times along the two spatial axes. PE is the number of processor elements.)



(a) Message passing vs. data parallel

(b) Multidimensional split

Figure 4. Message passing vs. data parallel and multidimensional split

Finally we compare the message passing approach with a version of data parallel programming (see Figure 4). A straightforward approach (denoted "PowerC without redistribution" in the plot) does not even show any speedup across the entire range of processors. Only by simulating the message passing approach (which is very expensive in terms of implementation effort) we reach some speedup but still significantly below the message passing implementation.

5 Conclusion

The experiments show, that in contrast to the pyramidal wavelet transform the border data problem can be neglected. A subband based redistribution is necessary to gain reasonable speedups. The data parallel approach is not able to compete against the message passing paradigm. A localized decomposition strategy can avoid the need for a second run and load balancing problems.

References

1. M.V. Wickerhauser. *Adapted wavelet analysis from theory to software.* A.K. Peters, Wellesley, Mass., 1994.
2. E. Goirand, M.V. Wickerhauser, and M. Farge. A parallel two-dimensional wavelet packet transform and some applications in computing and compression analysis. In R. Motard and B. Joseph, editors, *Applications of Wavelet Transforms in Chemical Engineering*, pages 275–319. Kluwer Academic Publishers Group, 1995.
3. A. Uhl. Adapted wavelet analysis an moderate parallel distributed memory MIMD architectures. In A. Ferreira and J. Rolim, editors, *Parallel Algorithms for Irregulary Structured Problems*, volume 980 of *Lecture Notes in Computer Science*, pages 275–284. Springer, 1995.
4. A. Uhl. Wavelet packet best basis selection on moderate parallel MIMD architectures. *Parallel Computing*, 22(1):149–158, 1996.
5. 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.
6. L. Bacchelli Montefusco. Parallel numerical algorithms with orthonormal wavelet packet bases. In C.K. Chui, L. Montefusco, and L. Puccio, editors, *Wavelets: Theory, Algorithms and Applications*, pages 459–494. Academic Press, San Diego, 1994.
7. 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.
8. F.G. Meyer, A.Z. Averbuch, J.O. Strömberg, and R.R. Coifman. Fast wavelet packet image compression. In *Proceedings of the Data Compression Conference DCC'98*, page 563. IEEE Press, March 1998.
9. 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.
10. 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.
11. 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.
12. 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.
13. S. Sullivan. Vector and parallel implementations of the wavelet transform. Technical report, Center for Supercomputing Research and Development, University of Illinois, Urbana, 1991.
14. 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.