Parallelisable Zero-Tree Image Coding with Significance Maps

Rade Kutil *

Salzburg University, Jakob Haringer-Str. 2, A-5020 Salzburg, Austria
rkutil@cosy.sbg.ac.at
http://www.cosy.sbg.ac.at/sc/staff/rade.kutil.html

Abstract. The wavelet transform is more and more widely used in image and video compression. As today the parallelisation of the wavelet transform is sufficiently investigated this work deals with the compression algorithm (SPIHT) itself as a next step. A derived algorithm with a simpler and spacially oriented coefficient scan order is presented, which is more suitable for parallelisation.

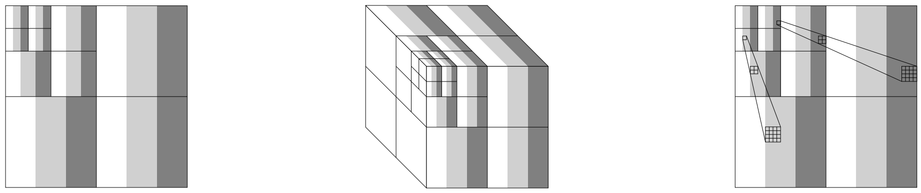
1 Introduction

Algorithms like SPIHT [6] and the JPEG-2000 standard [1] prove the superiority of wavelet methods in still image coding. Likewise, rate-distortion efficient 3-D algorithms for video coding exist (as e.g. [3]). A significant amount of work has already been done on parallel wavelet transform algorithms. This work concentrates on the parallelisation of the coding algorithm as the next step. A 3-D variant [3] of the SPIHT algorithm [6] was chosen for this purpose.

In [2] two approaches to parallelise the EZW algorithm (predecessor of SPIHT) are proposed: One is a straight-forward parallelisation with local algorithm execution on each processor element (PE) for distinct blocks. A similar approach was applied to SPIHT in [7]. This results in a loss of rate-distortion performance and bitstreams that are incompatible to the sequential algorithm. Therefore, the second approach reorders the encoded symbols after collection of the PE-local results. This approach was adopted to SPIHT in [4]. Unfortunately, it reveales some drawbacks as e.g. complicated bit-stream handling, additional communication and sequential code parts.

Here we will follow another approach, that is to modify the algorithm itself. The basic idea is to substitute the lists of coefficient positions involved in the algorithm by bitmaps to facilitate the parallelisation of the coefficient scan.

* The author wants to thank the EPCC (Edinburgh Parallel Computing Centre) and the TRACS programme (Training and Research on Advanced Computing Systems) for providing equipment, infrastructure and support for parallel computing (especially on Cray T3E). The author was also supported by the Austrian Science Fund FWF, project no. P13903.



(a) 2-D case (b) 3-D case (c) Zero-tree

Fig. 1. Distribution of coefficients. Different colours indicate different PEs

2 Wavelet Transform and Zero-Trees

In contrast to the parallelisation of the wavelet transform as presented in [5] the parallel wavelet transform used here dispenses with video data distribution as well as collection of transformed data. Parallelisation is data driven by distributing data in slices. After parallel decomposition data are distributed as shown in Fig. 1. Note that the speedups reported in this work do not include I/O operations as I/O is not viewed as a part of the algorithm.

Zero-tree based algorithms arrange the coefficients of a wavelet transform in a tree-like manner, i.e. each coefficient has a certain number of child coefficients in another sub-band (mostly 4 in the 2-D, 8 in the 3-D case). Here the following notations are used: $\mathrm{off}(p)$ is the direct offspring of a coefficient p, i.e. all coefficients whose parent coefficient is p. $\mathrm{desc}(p)$ are all descendants of a coefficient p. This includes $\mathrm{off}(p)$, $\mathrm{off}(\mathrm{off}(p))$ and so on. $\mathrm{parent}(p)$ is the parent coefficient of p. $p \in \mathrm{off}(\mathrm{parent}(p))$.

Furthermore, a zero-tree is a sub-tree which entirely consists of insignificant coefficients. The significance of a coefficient is relative to a threshold: $\mathrm{sig}(c) \Leftrightarrow |c| \geq$ threshold. The statistical properties of transformed image or video data (self-similarity) ensure the existence of many zero-trees. Zero-trees can be viewed as a collection of coefficients with approximately equal spacial position. While this fact implies that the coefficients significances are statistically related which is exploited by the SPIHT algorithm, this also means that zero-trees are local objects corresponding to the data distribution produced by the parallel wavelet transform (see Fig. 1(c)). This can be exploited by the parallelisation of the SPIHT algorithm (see Section 4).

3 The Modified Zero-Tree Compression Algorithm

To substitute the lists of coefficients used in SPIHT by bitmaps we have to rewrite the whole algorithm. In the following we will use three logical predicates $A(p)$, $B(p)$ and $C(p)$ as defined in Fig. 2. Corresponding to these predicates we will use the mappings a, b and c which essentially represent the same as A,

ProcessAll :=
  threshold $\leftarrow \max_{p \in \text{allcoefficients}} \mid p \mid$
  set a, b and c to all false
  for each refinement step
    threshold $\leftarrow$ threshold /2
    for p in approximation-subband
      ProcessCoeff(p, true)

ProcessCoeff$(p, \hat{c})$:=
  if a_p then Refine(p) else $a_p \leftarrow A(p)$
  if $\neg b_p \wedge \hat{c}$ then $b_p \leftarrow B(p)$
  if $b_p \wedge \neg c_p$ then $c_p \leftarrow C(p)$
  if b_p then
    for q in off(p)
      ProcessCoeff(q, c_p)

$$A(p) \Leftrightarrow \mid p \mid \geq \text{threshold}$$
$$B(p) \Leftrightarrow \bigvee_{q \in \text{desc}(p)} A(q)$$
$$C(p) \Leftrightarrow \bigvee_{q \in \text{off}(p)} B(q)$$

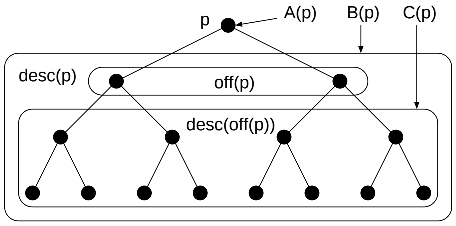


Fig. 2. Zero-tree coding algorithm

B and C. The difference is that A, B and C implicitly depend on threshold (implemented as a function) while a, b and c have to be updated explicitly (implemented as array of boolean values). We call a, b and c "significance maps".

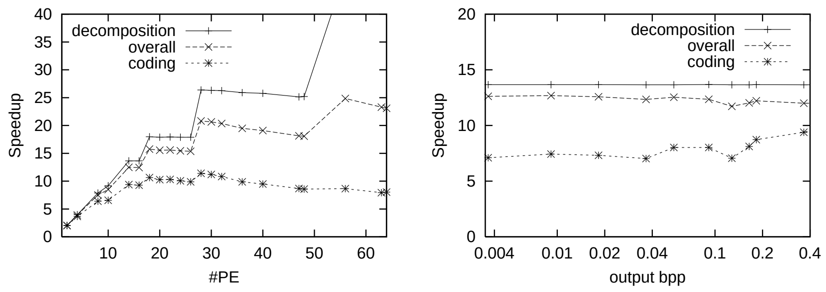
The algorithm is responsible for the equality of a, b, c and A, B, C respectively while the threshold is successively decreased by a factor of $\frac{1}{2}$. The evaluation of A, B and C should be avoided as far as possible because – following the idea of SPIHT – the result of each evaluation will be coded into the bit-stream as one bit to allow the decoder to reproduce the algorithms decisions. The algorithm that obeys these rules is shown in Fig. 2. It encodes the same information as SPIHT. Only, the order of the bits within a refinement step is different. Unfortunately, this causes a loss of rate-distortion efficiency in the middle of the refinement steps of up to 1 dB. There exists an approach to overcome this problem.

4 Parallelisation Results

In contrast to [4] the parallelisation of our modified algorithm is easy. All we have to do is to parallelise the inner loop in the procedure ProcessAll (which reads "for p in approximation-subband") according to the data distribution of the approximation sub-band (see Fig. 1). Each PE produces one continuous part of the bit-stream for each refinement step. At the end these parts have to be collected by a single PE and assembled properly (i.e. in an alternating way).

Experimental results were conducted on a Cray T3E-900/LC at the Edinburgh Parallel Computing Centre. Video data size is always 864 frames with 88 by 72 pixels. The video sequence used here is the U-part of "grandma". The wavelet transform is performed up to a level of 3.

Experiments show that the sequential algorithm outperforms SPIHT for higher bitrates by a factor of up to 1.6. Fig. 3 shows speedups of the modi-



(a) 0.14 bpp (b) 16 PEs

Fig. 3. Speedup of decomposition, zero-tree coding and overall speedup for varying #PE (a) and varying bitrate (b)

fied algorithm for both varying #PE and varying bitrate. For higher numbers of PEs the assembly of the bit-streams takes more execution time and thus limits the speedups. Note that although the speedup of the coding part increases with the bitrate the overall speedup remains almost constant because the share in execution time of the coding part increases with the bitrate.

References

[1] ISO/IEC JPEG committee. JPEG 2000 image coding system, March 2000. Final Committee Draft.

[2] C.D. Creusere. Image coding using parallel implementations of the embedded zerotree wavelet algorithm. In B. Vasudev, S. Frans, and S. Panchanathan, editors, *Digital Video Compression: Algorithms and Technologies 1996*, volume 2668 of *SPIE Proceedings*, pages 82–92, 1996.

[3] 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.

[4] R. Kutil. Zerotree based video coding on mimd architectures. In S. Panchanathan, V. Bove, and S.I. Sudharsanan, editors, *Media Processors 2001*, volume 4313 of *SPIE Proceedings*, pages 61–68, January 2001.

[5] R. Kutil and A. Uhl. Hardware and software aspects for 3-D wavelet decomposition on shared memory MIMD computers. In P. Zinterhof, M. Vajtersic, and A. Uhl, editors, *Parallel Computation. Proceedings of ACPC'99*, volume 1557 of *Lecture Notes on Computer Science*, pages 347–356. Springer-Verlag, 1999.

[6] 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.

[7] F.W. Wheeler and W.A. Pearlman. Low-memory packetized SPIHT image compression. In *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, October 1999.