A Significance Map Based Adaptive Wavelet Zerotree Codec (SMAWZ)

Rade Kutil

Dept. of Scientific Computing, University of Salzburg, Austria

ABSTRACT

This paper presents a wavelet based codec (SMAWZ) that uses the most important features of the SPIHT algorithm (zero-trees) while eliminating all concepts that are incompatible with a reduced hardware environment. This is done by replacing dynamic list structures with static maps (significance maps) which leads to a simpler and spacially oriented coefficient scan order. Additionally, extensions such as wavelet packet support and object based coding are included.

Keywords: Wavelets, Zero-tree, Compression, Image coding

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[1] and JPEG2000[2] standardisation processes.

SPIHT[3] is a wavelet based image compression algorithm that uses so-called zero-trees[4] (sets of insignificant, small coefficients). SPIHT is simple, fast and competitive in terms of rate-distortion performance. Therefore, SPIHT is a good candidate for hardware-near applications. Unfortunately, SPIHT makes use of lists of coefficients to manage the significance information. This method has two disadvantages: Firstly, the lists can grow almost arbitrarily large, which makes it hard to estimate the memory demands in hardware. Secondly, parallelisation[5] is difficult because of a chaotic coefficient scan order.

This paper presents a modified algorithm that uses bit-maps (significance maps) instead (similarly to Wheeler et al.[6] and Xiong et al.[7]). As a result, coefficient scans by list iteration are turned into spacially oriented scans. Therefore, parallelisation is easier.[8] Moreover, there is no need for dynamic memory management.

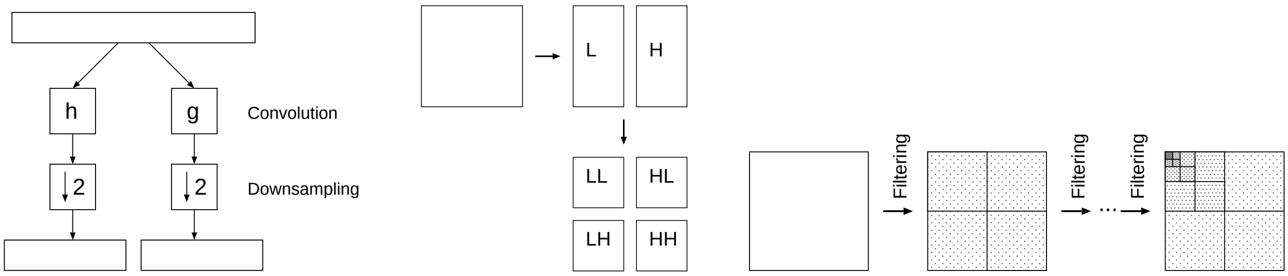
Image compression schemes that use wavelet packet decomposition[9] have proven to be able to outperform[7,10] other wavelet based codecs for certain types of images. There are several ways[11–13] to search for the wavelet packet basis that best suits to the compression scheme and the image to compress. We choose to use the best basis algorithm[11] in the SMAWZ codec. The zero-tree paradigm has to be adapted to wavelet packets.

Object based coding as introduced for SPIHT[14] is also included in the SMAWZ codec. First, the wavelet transform has to be adapted to the object shapes.[15,16] Then, the zero-trees are restricted to wavelet coefficients that are located within the object.

2. WAVELET TRANSFORM

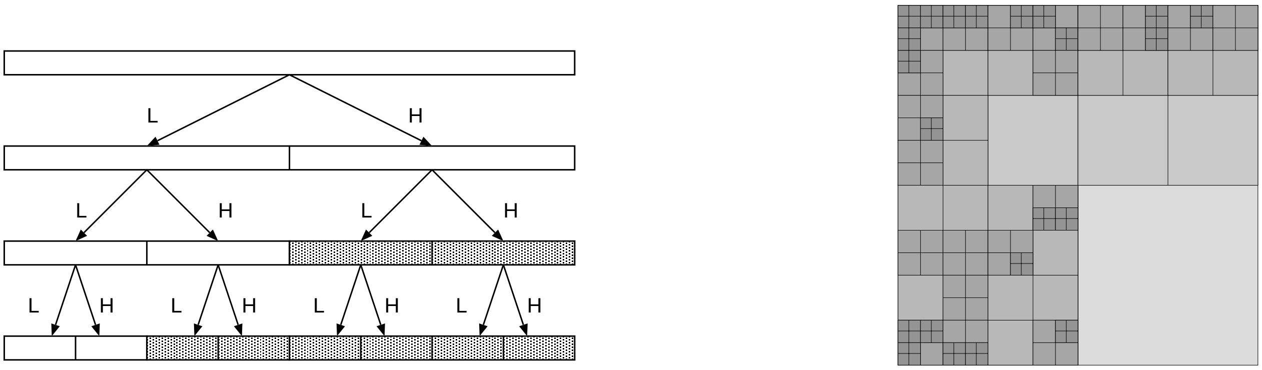
The fast wavelet transform (as shown in Figure 1) can be efficiently implemented by a pair of appropriately designed Quadrature Mirror Filters (QMF) consisting of a low-pass and a high-pass filter which decompose the original data set into two frequency-bands. These sub-bands are down-sampled by 2, and the same procedure is recursively applied to the coarse scale (low-pass filtered) sub-band. In the 2-D case, consecutive filtering of rows and columns produces four sub-bands. Only the one which is filtered with the low-pass filter in each dimension is decomposed further. This is called the pyramidal wavelet transform.

Author contact information: University of Salzburg, Dept. of Scientific Computing, Jakob Haringer Str. 2, A-5020 Salzburg, Austria; E-mail: rkutil@cosy.sbg.ac.at



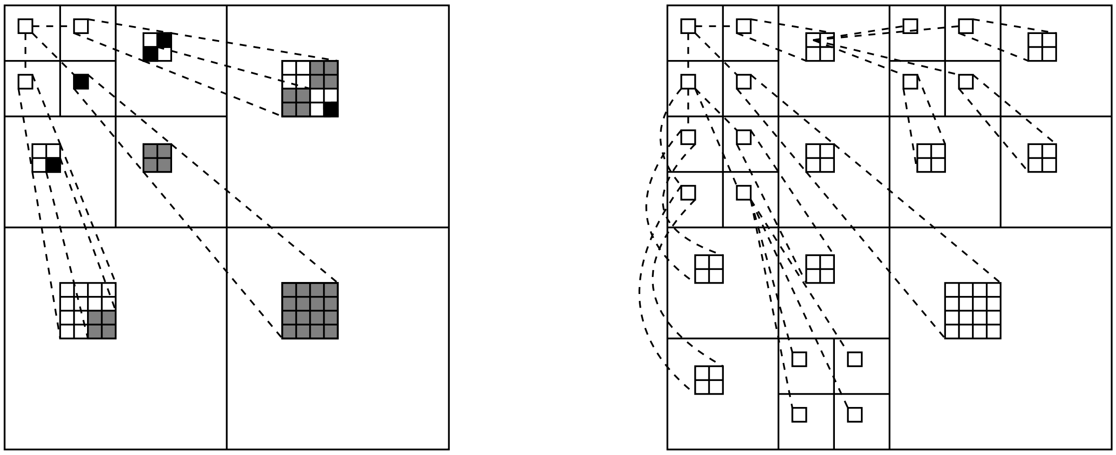
(a) 1-D filtering (b) 2-D filtering (c) 2-D decomposition

Figure 1: Pyramidal wavelet transform. The transform consists of several stages of recursive filtering



(a) 1-D wavelet packets tree (b) example 2-D decomposition

Figure 2. Wavelet packet decomposition. The filtering is recursively applied to all sub-bands. If the decomposition is stopped at optimal stages (b) the result is called best basis



(a) Pyramidal decomposition (b) Wavelet packet decomposition

Figure 3. Coefficient trees in zero-tree coding. The dark areas are insignificant coefficients. Black coefficients are zero-tree roots

Wavelet packets[9] (as shown in Figure 2) represent a generalisation of the method of multi-resolution decomposition and comprise the entire family of sub-band decompositions. Whereas in the pyramidal case the decomposition is applied recursively to the coarse scale approximation only, in the full wavelet packet decomposition the recursive procedure is applied to all the coarse and detail sub-bands, which leads to a complete wavelet packet tree (i.e. binary tree and quad-tree in the 1D and 2D case, respectively) and more flexibility in frequency resolution.

The wavelet packet "best basis algorithm"[11] performs an adaptive optimisation of the frequency resolution of a wavelet packet decomposition tree by selecting the most suitable frequency sub-bands for signal compression. This *suitability* is approximated by an information cost function. The cost function should be additive: The cost of a union of wavelet coefficient sets should be equal to the sum of the cost of separate sets. Examples are $-\sum_j |x_j|^2 \log |x_j|^2$, an additive modification of the entropy, or $\sum_j log|x_j|^2$, the logarithmic energy.

To determine the best basis, one has to perform a complete decomposition and to compute for each sub-band on each level its *best information cost* by comparing its *information cost* with the sum of its sub-bands *best information costs*. After that, the sub-bands that minimise the *overall information cost* have to be selected. This can be done by either accessing sub-band data kept in memory (high memory requirements) or by performing a second decomposition run according to the best basis tree (see e.g. Kutil et al.[17]).

3. COEFFICIENT TREES

3.1. Zero-trees

Zero-tree based algorithms arrange the coefficients of a wavelet transform in a tree-like manner (see Figure 3 (a)). All coefficients are connected by trees that are rooted in the approximation sub-band's coefficients (the sub-band that is low-pass filtered only). The following notations are used:

off(p) The direct offspring of a coefficient p, i.e. all coefficients whose parent coefficient is p.

desc(p) All descendants of a coefficient p. This includes off(p), off(off(p)) and so on.

parent(p) The parent coefficient of p. $p \in$ off(parent(p)).

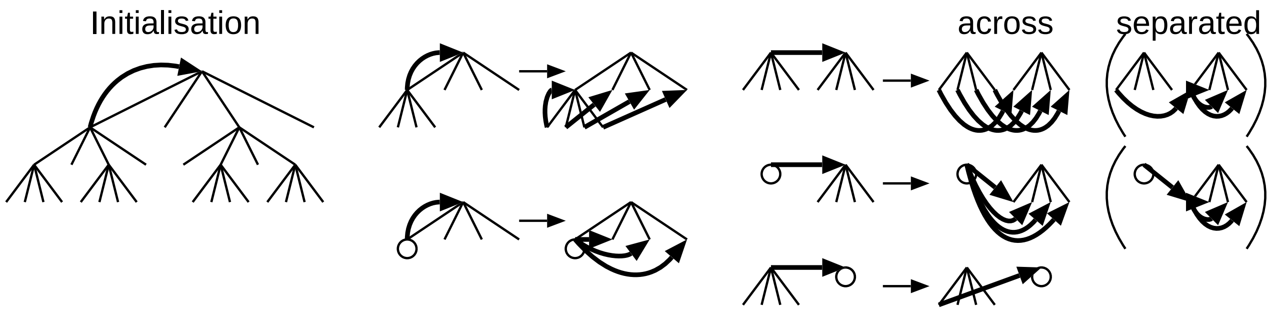


Figure 4. Construction of similarity trees. Thin lines indicate the wavelet packet decomposition tree; thick lines indicate the similarity tree. Starting with an arbitrary packet decomposition tree (left), the shown tree transform operations are applied until only leaf sub-bands are connected. Small circles indicate leaf sub-bands

Furthermore, a zero-tree is a sub-tree which entirely consists of insignificant coefficients. The significance of a coefficient is relative to a threshold which plays an imported role in zero-tree coding algorithms:

$$\mathrm{sig}(p) \Leftrightarrow |p| \geq \mathrm{threshold}$$

We will see that sometimes the root coefficient of the subtree (or even its direct offspring) does not have to be insignificant. Zero-trees can be viewed as a collection of coefficients with approximately equal spacial position.

3.2. Similarity Trees

To build trees of coefficients is more complicated when we face arbitrary wavelet packet decompositions (see Figure 3 (b)). So let us start at the beginning: Coefficient trees should connect coefficients of approximately equal spacial positions in a way so that bigger coefficients (usually the low-pass filtered ones) are next to the root of the trees and the significance (absolute value) of parent and child coefficients is statistically related (similarity condition).

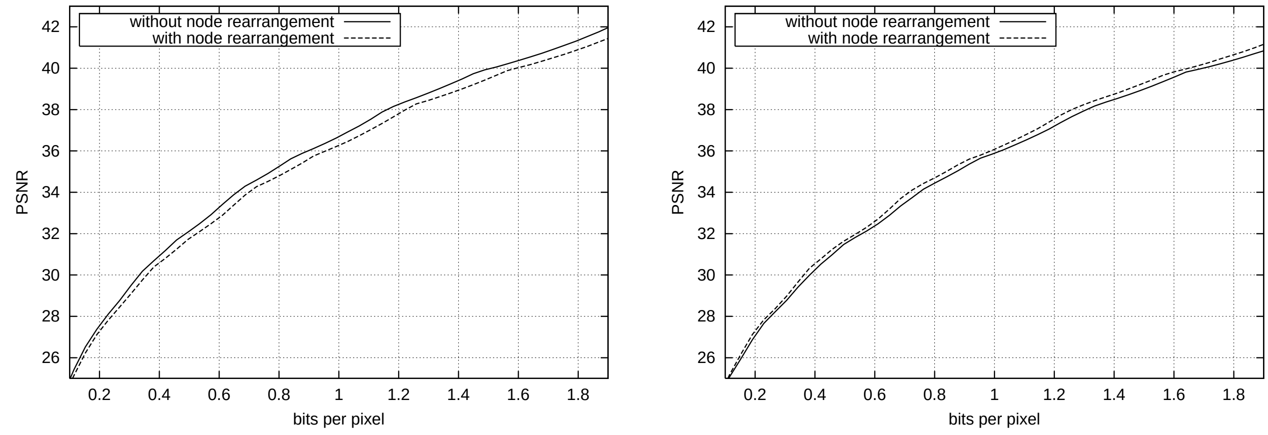
To achieve this, we have to find similar sub-bands to define parent-child relationships between their coefficients. We connect similar sub-bands by trees we will call similarity trees.

Note that there are three kinds of trees now: (a) the wavelet packet decomposition tree, (b) the similarity tree that connects the leaves of the packet tree and (c) the coefficient tree ("carrying" the zero-trees) that connects coefficients as forced by the similarity tree.

See Figure 4 for a set of construction operations for similarity trees. The leftmost child-node always is the double-low-pass-filtered sub-band (LL). The construction starts with the basic similarity between the first approximation sub-band and the original image. The operations transform coarse similarity relations into detail relations until only leaf-sub-bands (non-decomposed) are part of the similarity tree.

There are two options: *across* and *separated*. The reader might perform a simple construction by hand to see the difference. (Figure 3 (b) is of type across.) Each one has its advantages. Figure 5 shows a comparison of SMAWZ coding performance in relation to the two types of similarity trees. The results (across is superior) are similar for all images tested by the author.

A method was proposed by Rajpoot et al.[18] to rearrange the nodes (sub-bands) within the similarity tree to avoid the case of a parent sub-band being larger than its child sub-band. The problem with this is that in the corresponding coefficient trees a coefficient could have more than one parent coefficient. We choose to resolve this by terminating the coefficient trees at all but one of the multiple parent coefficients. I.e. only one parent coefficient is connected to its child coefficients. Nevertheless, the rearrangement method was implemented additionally for comparison purposes. See Figure 5 for PSNR results. Unfortunately, the rearrangement does not increase the rate-distortion performance significantly. The difference is neglectable for most images. If not, this method usually increases performance slightly for similarity trees of separated type and decreases performance for the across type.



(a) across (b) separated

Figure 5. Comparison of codec performance for distinct types of similarity trees. The results relate to the "Barbara" image at a maximum decomposition depth of 6 using the biorthogonal 7/9-tap filter, best-basis algorithm and the L1 norm (sum of absolute values) as cost function

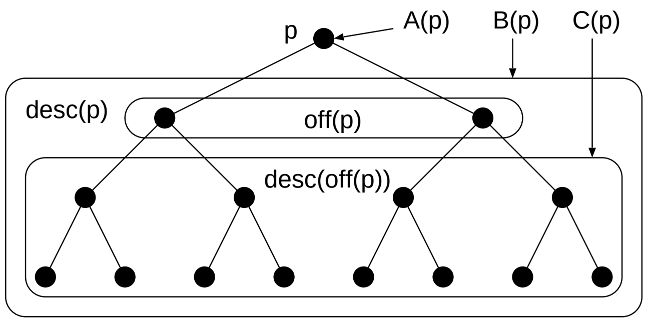


Figure 6: Predicates used in the algorithm

4. CODING ALGORITHM

4.1. Simple Approach

As mentioned above, we want to take the SPIHT algorithm and substitute the lists of coefficients by bitmaps. Nevertheless, this is not as easy as it sounds. Thus, we will rewrite the whole algorithm.

In the following we will use three logical predicates $A(p)$, $B(p)$ and $C(p)$ which are defined as follows:

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

$A(p)$ simply denotes the significance of the coefficient p. $B(p)$ is true if and only if at least one of p's descendants is significant, while $C(p)$ denotes the same but does not include the direct offspring of p. This is visualised in Figure 6. By these predicates the state of significance of a given set of coefficients can be described in terms of zero-trees. Corresponding to these predicates we will use the mappings a, b and c which essentially represent

```
ProcessAll :=
    threshold ← max_{p∈allcoefficients} | p |
    set a, b and c to all false
    for each refinement step
        threshold ← threshold /2
        for p in approximation-subband
            ProcessCoeff(p, true)
```

```
ProcessCoeff(p, ĉ) :=
    if a_p then Refine(p) else a_p ← A(p)
    if ¬b_p ∧ ĉ then b_p ← B(p)
    if b_p ∧ ¬c_p then c_p ← C(p)
    if b_p then
        for q in off(p)
            ProcessCoeff(q, c_p)
```

Figure 7: Zero-tree coding algorithm

the same as A, B and C. The difference is that A, B and C immediately change their values if the threshold is changed and are therefore implemented as a function in the used programming language. a, b and c have to be updated explicitly and are therefore implemented as array of Boolean values. a, b and c are the "significance maps" that appear in the title and substitute the lists of coefficients used in SPIHT.

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 $\frac{1}{2}$. This should be done by avoiding the evaluation of A, B and C as far as possible because – following the idea of the SPIHT algorithm – the result of each evaluation will be encoded into the bit-stream as one bit to allow the decoder to reproduce the decisions the encoder has made.

The algorithm that obeys these rules is shown in Figure 7. The outer loop is the refinement loop which divides the threshold by 2 in each iteration. This is exactly the same as in the original SPIHT algorithm. Within this loop the algorithm navigates through the set of coefficients along trees of coefficients in a depth-first manner starting at the set of coefficients contained in the approximation sub-band. This is accomplished by the recursive procedure "ProcessCoeff".

For each coefficient the state of a, b and c is checked one after another. $a_p \leftarrow A(p)$ has to be evaluated only if a_p is false because the transition true $\mapsto$ false is not possible for a_p. If $A(p)$ is true then the sign of p has to be encoded as well (not shown). If a_p is already true the procedure "Refine" is called which encodes another bit of the coefficients value to refine the decoded approximation. $b_p \leftarrow B(p)$ has to be evaluated only if (again) b_p is false *and* $\hat{c} = C(\text{parent}(p))$ is true because $B(p) \Rightarrow C(\text{parent}(p))$. $c_p \leftarrow C(p)$ has to be evaluated only if c_p is false *and* b_p is true because $C(p) \Rightarrow B(p)$. At last the recursion to the child coefficients only has to be performed if b_p is true (for an obvious reason).

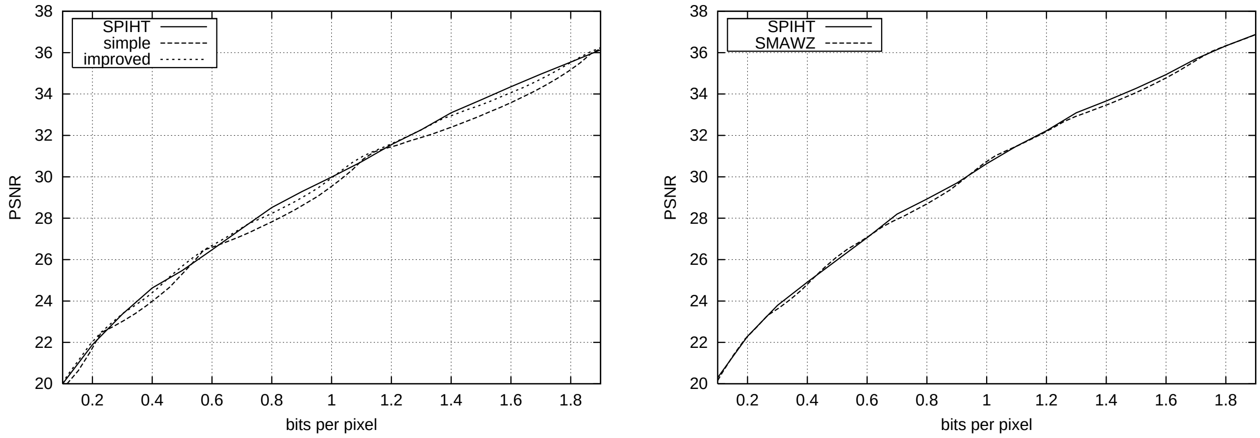
The decoding algorithm looks exactly the same. The only difference is that instead of encoding the results of the evaluation of A, B and C this information is read from the bit-stream. Together with the sign- and refinement-bits this is enough information to enable the decoder to perform the same steps as the encoder and approximate the coefficients with an error below the threshold.

Note that this algorithm encodes the same information (in fact the same bits) as the SPIHT algorithm. Only, the order of the bits is different. This means that at the end of each refinement step the compression performance is equal to that of SPIHT. In between the order of the bits that are written to the bit-stream is crucial because the bits can have different effect on the decoded image. Thus, it is important to encode bits with bigger effect first. Figure 8 (a) shows the comparison of the PSNR-performance for the "houses"-image and pyramidal wavelet decomposition. The simple algorithm shows major drawbacks with respect to SPIHT.

4.2. Improved Approach

The fact that the simple approach is significantly inferior to SPIHT forces us to do something against it. The reason for the loss of rate-distortion performance must have something to do with the order in which bits are written to the bit-stream during a refinement step – since the bits are essentially the same as in SPIHT. The order of bits is determined by the order in which the coefficients are processed. So this is what has to be changed – without losing the advantage of significance maps and the depth-first scan order.

First of all, we have to distinguish between important and unimportant bits: Important bits are those that have big impact on the reconstructed image, i.e. A-bits plus sign-bits and to some extent refinement bits. Most



(a) without arithmetic coding

(b) with arithmetic coding

Figure 8. PSNR of SMAWZ compared to SPIHT. The results relate to the “houses” image at a decomposition depth of 6 using the biorthogonal 7/9-tap filter and pyramidal wavelet decomposition

ProcessAll :=
  threshold $\leftarrow \max_{p \in \text{allcoefficients}} \mid p \mid$
  for each refinement step
    threshold $\leftarrow$ threshold /2
    for p in approximation-subband
      ProcessCoeffPass1(p)
    for p in approximation-subband
      ProcessCoeffPass2(p, true)
    do
      foundnewcoeffs $\leftarrow$ false
      for p in approximation-subband
        ProcessCoeffPass3(p, true)
    while foundnewcoeffs

CheckDescSig($p, \hat{c}$) :=
  if $\hat{c}$ then $\bar{b}_p \leftarrow B(p)$
  if $\bar{b}_p$
    for q in off(p) do $a_q \leftarrow A(q)$
    foundnewcoeffs $\leftarrow$ true

ProcessCoeffPass1(p) :=
  if $\neg a_p$ then $\bar{a}_p \leftarrow A(p)$
  if b_p then
    for q in off(p) do ProcessCoeffPass1(q)

ProcessCoeffPass2($p, \hat{c}$) :=
  if $\bar{a}_p$ then $\bar{a}_p \leftarrow$ false, $a_p \leftarrow$ true else Refine(p)
  if b_p then
    if $\neg c_p$ then $c_p \leftarrow C(p)$
    for q in off(p) do ProcessCoeffPass2(q, c_p)
  else CheckDescSig($p, \hat{c}$)

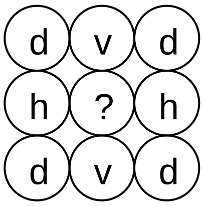
ProcessCoeffPass3($p, \hat{c}$) :=
  if b_p then
    for q in off(p) do ProcessCoeffPass3(q, c_p)
  elsif $\bar{b}_p$ then
    $\bar{b}_p \leftarrow$ false, $b_p \leftarrow$ true
    if $\neg c_p$ then $c_p \leftarrow C(p)$
    for q in off(p) do CheckDescSig(q, c_p)

Figure 9: Improved zero-tree coding algorithm

important are coefficients that are very likely to turn significant ($A(p) = \text{true} \wedge a_p = \text{false}$). Unimportant bits are those that have no direct influence on image content, i.e. B- and C-bits. Thus, important bits have to be encoded as early as possible – without loss of correctness.

The algorithm in Figure 9 takes these requirements into account. Each refinement step is split into at least three passes. The first pass only processes A-bits of insignificant coefficients the are not covered by zero-trees. No evaluations of B and C are performed. Pass 2 and 3 evaluate B- and C-bits as well as A-bits of newly discovered potentially significant coefficients (immediately after they are discovered). B- and C-bits of these newly discovered coefficients are evaluated in subsequent passes. This is to evaluate unimportant bits as late as possible. Pass 3 has to be repeated until no new coefficients are found.

In this way, the rate-distortion performance of our improved algorithm matches the performance of SPIHT,



Significance coding:

LL,LH,HL	h	0	0	0	0	0	1	1	1	2
	v	0	0	0	1	2	0	0	1+	x
	d	0	1	2+	x	x	0	1+	x	x
	context	0	1	2	3	4	5	6	7	8
HH	d	0	0	0	1	1	1	2	2	3+
	h + v	0	1	2+	0	1	2+	0	1+	x
	context	0	1	2	3	4	5	6	7	8

Sign coding:

h	1	1	1	0	0	0	-1	-1	-1
v	1	0	-1	1	0	-1	1	0	-1
predicted	1	1	1	-1	1	1	-1	-1	-1
context	4	3	2	1	0	1	2	3	4

Figure 10. Contexts for arithmetic coding taken from EBCOT. Neighbouring coefficients are used to predict the significance of a coefficient

as is shown in Figure 8 (a).

4.3. Arithmetic Coding

SPIHT uses arithmetic coding to improve rate-distortion performance by about 0.5 dB. Four bits of four neighbouring coefficients (the direct offspring of a certain parent coefficient) are grouped together and fed into the arithmetic coder as a single symbol.

In the case of wavelet packet decompositions (and similarity trees), the direct offspring does not necessarily consist of exactly four coefficients. This makes the above approach not applicable. Looking for another context-based arithmetic coding approach for coefficient significance, we find EBCOT, which JPEG2000 is based upon.

It basically sums the significance bits of neighbouring coefficients (horizontally, vertically and diagonally) and builds several contexts out of this information (see Figure 10). Each context yields a different probability for 1- and 0-bits. The contexts also depend on the sub-band type (LL, HL, LH and HH).

A similar approach is used for the sign bits. 1 and -1 denote positive and negative signs respectively and 0 denotes a so far insignificant coefficient. If both neighbouring coefficients are significant and their signs are not equal, 0 is used as well.

Using this approach, the rate-distortion performance is improved by about 0.5 dB (as in SPIHT). See Figure 8 (b) for a comparison of SMAWZ including arithmetic coding and SPIHT.

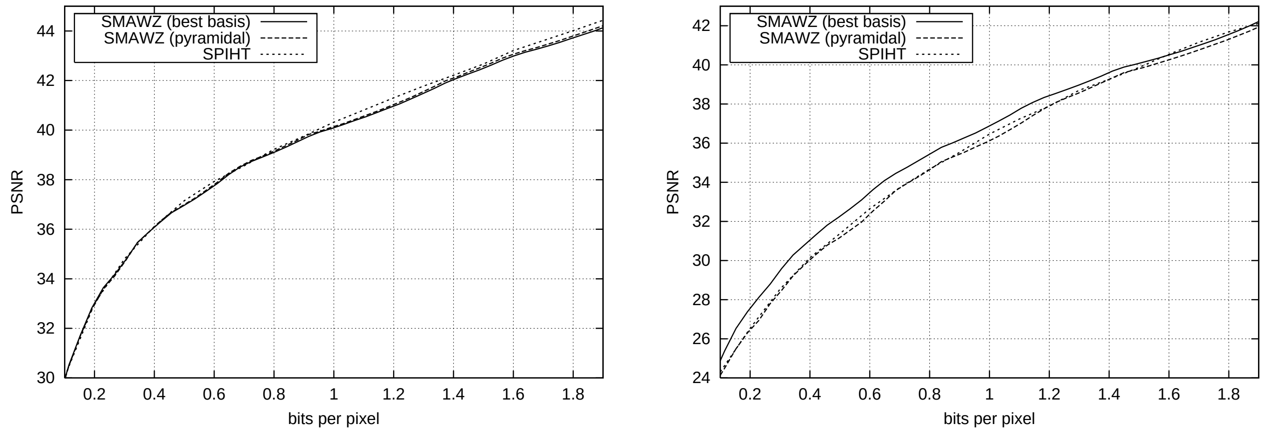
4.4. SMAWZ and Wavelet Packets

Rate-distortion results of SMAWZ using all its features (wavelet packet decomposition, similarity trees, and the improved algorithm plus arithmetic coding) are shown in Figure 11.

4.5. Memory and Timing Performance

A very important argument for the use of the algorithms presented in this paper is the static memory demand. So, what is the actual memory demand? Let n be the number of pixels in the image ($n = \text{width} \cdot \text{height}$). Of course, we need n floating point numbers (or fixed point – depending on the implementation) to store the image in memory. The wavelet decomposition can be implemented without appreciable additional buffer memory. However, the best-basis algorithm needs extra memory if not implemented using the second-run approach which is too time consuming. As stated in previous work[17] this extra memory can be reduced to $\frac{n}{4} + \frac{n}{4^2} + \frac{n}{4^3} + \ldots \approx \frac{n}{3}$ if we assume that the decomposition depth is at least 1.

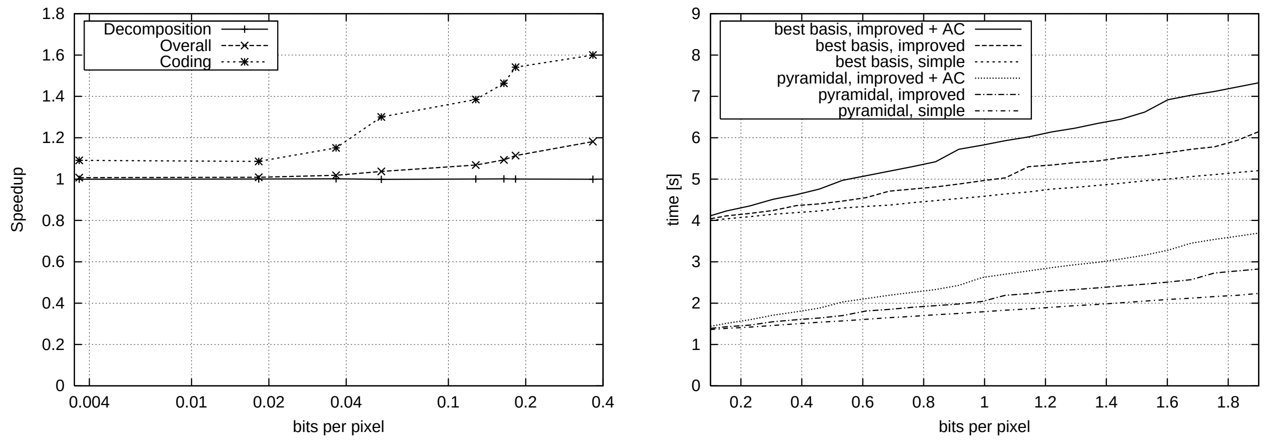
The simple coding algorithm needs extra memory for a, b and c, i.e. 3 bits per pixel. The improved algorithm uses two more bit-maps $\bar{a}, \bar{b}$ and, thus, needs 5 bits per pixel. There is a method to improve performance by



(a) Lena

(b) Barbara

Figure 11. Performance results for well-known images (512 × 512 pixel). For images without regular textures (a) the performance of SMAWZ equals that of SPIHT. For images including such textures (b) SMAWZ outperforms SPIHT due to the wavelet packet decomposition. The results relate to a maximum decomposition depth of 6 using the biorthogonal 7/9-tap filter, best-basis algorithm and log-energy cost function



(a) Speedup compared to SPIHT

(b) Execution time of SMAWZ

Figure 12. Execution time of SMAWZ. Figure (a) relates to a 3-D implementation of the simple approach and a comparably optimised implementation of SPIHT. Figure (b) contains execution times of SMAWZ on a 400 MHz Pentium II

storing the information about where newly found coefficients actually are, so that subsequent passes do not need to navigate through the whole coefficient tree. This method requires another bit per pixel and is able to reduce the execution time of the coding part by about 20%.

Figure 12 (a) shows that the use of significance maps instead of lists (SPIHT) can improve the timing performance – at least for the simple approach. Figure 12 (b) shows execution times of SMAWZ encoding a 512×512 pixel image. The results include the wavelet transform: 0.97 sec. for pyramidal and 3.35 sec. for best basis decomposition. Two parts of SMAWZ do not depend on the bit-stream size and depend linearly on the image size: the wavelet transform (linear only for pyramidal decomposition) and the calculation of $\max_{q\in\mathrm{desc}(p)} |q|$ for all coefficients p. The coding algorithm itself does not depend on the image size but on the bit-stream size only (apparently almost linearly).

5. OBJECT BASED CODING

Traditional image coding is based on rectangular images, while in object based coding, the image boundaries are arbitrary. To support object based coding in SMAWZ, firstly, the wavelet decomposition has to be modified.

5.1. Object Based Wavelet Transform

Objects are embedded into rectangular images and associated with a mask (1 if inside, 0 if outside the object). As sub-bands are down-sampled, there has to be an appropriate down-sampled mask for each scale. Therefore, masks are extended to even boundaries before filtering. Filtering is performed on contiguous lines and rows with appropriate border handling at even boundaries.

Note: SMAWZ does not use the ability of uneven sized biorthogonal filters to filter uneven sized data sets because all kinds of filters should be supported.

5.2. Object Based Zero-Trees

When each sub-band is associated with a mask, the offspring of a coefficient that is inside the the object is possibly partly outside the object. Therefore, we have to change the definition of **off** and **desc** (Section 3.1) so that a coefficient q is only element of $\mathrm{off}(p)$ if $\mathrm{mask}(q) = 1$. In this way, the algorithms in this paper work with objects without modification.

5.3. Object Based Coding

Object based coding in SMAWZ is especially useful because of the possibility to decompose distinct objects into distinct wavelet bases. Note that distinct objects can have distinct best bases.

As mentioned above, a SMAWZ bit-stream is an embedded bit-stream i.e. bit-stream cut-off yields optimal rate-distortion performance for the so reduced bit-rate. But, what if two of more objects have to be encoded into a single bit-stream? The brute force approach would be to encode the first object at a certain bit-rate followed by the second object. Clearly, the resulting bit-stream would not be embedded. The next approach would be to interleave bit-stream partitions in an optimal way. This is complicated.

There is a better and very simple approach: Consider the union of the approximation sub-bands of all involved objects. This set can serve as a set of coefficient tree roots that can be used by the presented algorithms in this paper. In this way, the errors introduced by the codec are kept below the threshold for all coefficients of each object.

6. CONCLUSION

A wavelet based image codec was presented that is, despite its dynamic and complex nature, highly suitable for hardware implementations. This is a result of its coding core which is both simple and highly competitive in terms of rate-distortion performance and speed and has static memory requirements due to the use of significance maps.

The capabilities of the compact algorithm include efficient support for wavelet packets, arithmetic coding and object based coding.

ACKNOWLEDGEMENTS

The author was partly supported by the Austrian Science Fund FWF, project no. P13903. The author wants to thank Andreas Pommer, Manfred Feil, Peter Meerwald, and Thomas Schell for their cooperation in SMAWZ programming.

REFERENCES

1. R. Koenen, "Overview of the MPEG-4 standard." ISO/IEC JTC1/SC29/WG11, July 1996.
2. I. J. committee, "JPEG 2000 image coding system," Mar. 2000. Final Committee Draft.
3. 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**, pp. 243–249, June 1996.
4. J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Process.* **41**, pp. 3445–3462, Dec. 1993.
5. R. Kutil, "Zerotree based video coding on MIMD architectures," in *Media Processors 2001*, S. Panchanathan, V. Bove, and S. Sudharsanan, eds., *SPIE Proceedings* **4313**, pp. 61–68, Jan. 2001.
6. F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists," in *Proceedings of the IEEE ICASSP 2000*, (Istanbul, Turkey), June 2000.
7. Z. Xiong, K. Ramchandran, and M. T. Orchard, "Wavelet packet image coding using space-frequency quantization," *IEEE Transactions on Image Processing* **7**, pp. 892–898, June 1998.
8. R. Kutil, "Parallelisable zero-tree image coding with significance maps," in *Parallel Processing. Proceedings of EuroPar'01*, R. S. et al., ed., *Lecture Notes on Computer Science* **2150**, pp. 674–677, Springer-Verlag, 2001.
9. M. Wickerhauser, *Adapted wavelet analysis from theory to software*, A.K. Peters, Wellesley, Mass., 1994.
10. F. G. Meyer, A. Z. Averbuch, and J.-O. Strömberg, "Fast wavelet packet image compression," in *Proceedings Data Compression Conference (DCC'98)*, J. Storer and M. Cohn, eds., p. 563, IEEE Press, (Snowbird, UT, USA), Mar. 1998.
11. R. Coifman and M. Wickerhauser, "Entropy based methods for best basis selection," *IEEE Transactions on Information Theory* **38**(2), pp. 719–746, 1992.
12. K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *IEEE Trans. on Image Process.* **2**(2), pp. 160–175, 1993.
13. N. M. Rajpoot, R. G. Wilson, F. G. Meyer, and R. R. Coifman, "A new basis selection paradigm for wavelet packet image coding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP'01)*, pp. 816–819, (Thessaloniki, Greece), Oct. 2001.
14. H. Flatscher and A. Uhl, "oSPIHT - embedded object-based SPIHT image coding," in *Proceedings of the 2nd IEEE Region 8 - EURASIP Symposium on Image and Signal Processing and Analysis, ISPA '01*, S. Loncaric and H. Babic, eds., pp. 593–598, IEEE Signal Processing Society, (Pula, Croatia), June 2001.
15. H. Katata, N. Ito, T. Aono, and H. Kusao, "Object wavelet transform for coding of arbitrarily shaped image segments," *IEEE Transactions on Circuits and Systems for Video Technology* **7**(1), pp. 234–237, 1997.
16. S. Li and W. Li, "Shape-adaptive discrete wavelet transform for arbitrarily shaped visual object coding," *IEEE Transactions on Circuits and Systems for Video Technology* **10**, pp. 725–743, Aug. 2000.
17. R. Kutil and A. Uhl, "Parallel adaptive wavelet analysis (Invited Paper)," *Future Generation Computer Systems (Special Issue on High Performance Numerical Methods and Applications)* **1**(18), pp. 97–106, 2001.
18. N. M. Rajpoot, F. G. Meyer, R. G. Wilson, and R. R. Coifman, "On zerotree quantization for embedded wavelet packet image coding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, (Kobe, Japan), Oct. 1999.