Embedded Lower Tree Wavelet encoder

Otoniel M. López[1] , Miguel O. Martínez-Rach[2] , Pablo Piñol[3] , Jose S. Oliver[4] , Manuel P. Malumbres[5]

Resumen—**Traditional embedded coding systems involve higher complexity algorithms, requiring faster and more expensive processors. Recently, several authors have shown an interest on developing very fast and simple non-embedded wavelet encoders that are able to get reasonable good performance with reduced requirements of computing resources. In these encoders each coefficient is encoded as soon as it is visited and this results in the loss of SNR scalability and precise rate control capabilities. In this paper we present a new embedded encoder called Embedded-LTW which combine SNR scalability feature of the embedded encoders and the reduced requirements and computing resources inherent to non-embedded encoders like LTW.**

Palabras clave—**Image coding, Wavelets, Embedded bitstream, Sign coding.**

I. Introduction

WAVELET transforms have proven to be very powerful tools for image compression. Many state-of-the-art image codecs, including the JPEG2000 image coding standard, employ a wavelet transform in their algorithms ([1],[2],[3],[4],[5]). One advantage is the provision of both frequency and spatial localization of image energy. The image energy is compacted into a fraction of the transform coefficients and compression can be achieved by coding these coefficients.

Several ahuthors have improved coding efficiency of wavelet-based image encoders, achieving in this way a reduction in the bandwidth or amount of memory needed to transmit or store a compressed image. Unfortunately, many of these coding optimizations involve higher complexity, requiring faster and more expensive processors. For example, the JPEG 2000 [1] standard uses a large number of contexts and an iterative time-consuming optimization algorithm (called PCRD) to improve coding efficiency. Other encoders (like the one proposed in [6]) achieve very good coding efficiency with the introduction of high-order context modeling, being the model formation a really slow process. Even bit-plane coding employed in many encoders (like [2] and [7]) results in a slow coding process since an image is scanned several times, focusing on a different bit-plane in each pass, which in addition causes a high cache miss rate.

The above mentioned encoders are designed to obtain the maximum performance in rate-distortion terms and also a broader functionality, but sadly, other design parameters like complexity or memory resources are not considered as critical as the former ones. Recently, divers authors have shown an interest on developing very fast and simple wavelet encoders that are able to get reasonable good performance with reduced requirements of computing resources. The objective of these fast and efficient image encoders is mainly targeted to interactive real-time applications running under resource constrained devices. In that scenario, the data must be encoded as soon as possible to fit the application time restrictions using the scarce available resources in the system (memory and processing power).

Basically, these encoders do not present any type of iterative method, and each coefficient is encoded as soon as it is visited. This results in the loss of SNR scalability and precise rate control capabilities. They simply apply a constant quantization to all the wavelet coefficients, encoding the image at a constant and uniform quality, as it happened in the former JPEG standard, where only a quality parameter was available (and no rate control was performed).

In [8], a tree-based wavelet encoder (LTW) is presented. The LTW encoding process avoids bit-plane processing and predictive encoding techniques; instead it uses one-pass coefficient coding and a very reduced number of contexts for the final arithmetic coding stage. The LTW encoder requires a very short memory space for the coding process but it is not SNR scalable in a natural fashion (i.e. it is not SNR-embedded).

Another very fast non-embedded encoder has been proposed in [9]. This encoder is called PROGRES. It follows the same ideas of [8], avoiding bit-plane coding and using coefficient trees to encode wavelet coefficients in only one-pass. In this encoder, all the coefficients (and not only the zero coefficients) are arranged in trees. The number of bits needed to encode the highest coefficient in each tree is computed, and all the coefficients at the current subband level are binary encoded with that number of bits. Then, the following subband level is encoded (in decreasing order), simply by computing again the number of bits needed to represent each sub-tree at that level and using that number of bits again.

Recently, the BCWT encoder [10] was proposed. It offers high coding speed, low memory usage and good R/D performance. The key of BCWT encoder is its unique one-pass backward coding, which starts from the lowest level sub-bands and travels backwards. MQD map calculation and coefficient encoding are all carefully integrated inside this pass in such a way that there is as little redundancy as possible

[1]Dpto. de Física y Arquitectura de Computadores, Univ. Miguel Hernández, e-mail: `otoniel@umh.es`.

[2]Dpto. de Física y Arquitectura de Computadores, Univ. Miguel Hernández, e-mail: `mmrach@umh.es`.

[3]Dpto. de Física y Arquitectura de Computadores, Univ. Miguel Hernández, e-mail: `pablop@umh.es`.

[4]Dpto. de Informática de Sistemas y Computadores, Univ. Politécnica Valencia, e-mail: `joliver@disca.upv.es`.

[5]Dpto. de Física y Arquitectura de Computadores, Univ. Miguel Hernández, e-mail: `mels@umh.es`.

for computation and memory usage.

Our purpose is to obtain a new embedded encoder based on the LTW features but with the advantage of being SNR scalable. The main idea is to obtain the LTW symbol map and to store the significant bits from the significant coefficients in bit planes. First we encode the symbol map and the significant coefficient sign and then we send in bit plane order the significant bits from the significant coefficients obtaining in this manner an SNR and sptatial scalable bitstream.

The organization of the paper is the following one: in section II the LTW algorithm is described. In section III, we present the Embedded-LTW encoder version showing the bitstream structure diferences respect to original LTW and making special emphasys in sign encoding. Section IV introduces a sign coding variation over the Embedded-LTW encoder. Evaluation results are presented in section V and finally, in section VI some conclusions and future work are drawn.

II. Lower Tree Wavelet Encoding

In LTW, the quantization process is performed by two strategies: one coarser and another finer. The finer one consists in applying a scalar uniform quantization, Q, to wavelet coefficients. The coarser one is based on removing the least significant bit planes, *rplanes*, from wavelet coefficients.

A tree structure (similar to that of [2]) is used not only to reduce data redundancy among subbands, but also as a simple and fast way of grouping coefficients. As a consequence, the total number of symbols needed to encode the image is reduced, decreasing the overall execution time. This structure is called *lower tree*, and it is a coefficient tree in which all its coefficients are lower than $2^{rplanes}$.

Our algorithm consists of two stages. In the first one, the significance map is built after quantizing the wavelet coefficients (by means of both Q and *rplanes* parameters). The symbol set employed in our proposal is the following one: a $LOWER$ symbol represents a coefficient that is the root of a lower-tree, the rest of coefficients in a lower-tree are labeled as $LOWER\_COMPONENT$, but they are never encoded because they are already represented by the root coefficient. If a coefficient is insignificant but it does not belong to a lower-tree because it has at least one significant descendant, it is labeled as an $ISOLATED\_LOWER$ symbol. For a significant coefficient, we simply use a symbol indicating the number of bits needed to represent it.

Let us describe the coding algorithm. In the first stage (symbol computation), all wavelet subbands are scanned in 2×2 blocks of coefficients, from the first decomposition level to the Nth (to be able to build the lower-trees from leaves to root). In the first level subband, if the four coefficients in each 2×2 block are insignificant (i.e., lower than $2^{rplanes}$), they are considered to be part of the same lower-tree, labeled as $LOWER\_COMPONENT$. Then, when scanning upper level subbands, if a 2×2 block has four insignificant coefficients, and all their direct descendants are $LOWER\_COMPONENT$, the coefficients in that block are labeled as $LOWER\_COMPONENT$, increasing the lower-tree size.

However, when at least one coefficient in the block is significant, the lower-tree cannot continue growing. In that case, a symbol for each coefficient is computed one by one. Each insignificant coefficient in the block is assigned a $LOWER$ symbol if all its descendants are $LOWER\_COMPONENT$, otherwise it is assigned an $ISOLATED\_LOWER$ symbol. On the other hand, for each significant coefficient, a symbol indicating the number of bits needed to represent that coefficient is employed.

Finally, in the second stage, subbands are encoded from the LL_N subband to the first-level wavelet subbands. Observe that this is the order in which the decoder needs to know the symbols, so that lower-tree roots are decoded before its leaves. In addition, this order provides resolution scalability, because LL_N is a low-resolution scaled version of the original image, and as more subbands are being received, the low-resolution image can be doubled in size. In each subband, for each 2×2 block, the symbols computed in the first stage are entropy coded by means of an arithmetic encoder. Recall that no $LOWER\_COMPONENT$ is encoded. In addition, significant bits and sign are needed for each significant coefficient and therefore binary encoded.

III. Embedded LTW

In this section we present the Embedded-LTW encoder, based on the original LTW encoder but using an embedded bit stream. With an embedded bit stream, the reception of code bits can be stopped at any point and the image can be decompressed and reconstructed.

In this encoder, the quantization process is the same than in the original LTW, either the symbol computation step. Differences between original LTW encoder and Embedded-LTW encoder lies in the symbol coding step. In this second step (coding step), the Embedded-LTW encoder encodes the symbols map and the sign in such a manner that when a significant coefficient is found, then, their sign is encoded by means of an arithmetic encoder.

Whereas in the LTW encoder, significant bits were encoded as soon as a significant coefficient was visited, in the Embedded-LTW, these significant bits are stored in an structure ordered by subband and bit plane, so as to send them to the bitstream after the whole symbol map and signs are sent.

After the symbols map is encoded and sent to the bitstream, the significant coefficients bits are raw encoded in bitplane order (from the most significant bit (MSB) to the least significant bit (LSB)) until we reach the desired target bitrate. As we have stored the significant coefficients ordered in an structure by subband and bit plane, we do not have to go through the image more times. Insted of it, we only have to send the bits stored in this structure reducing the algorithm complexity.

A. Bitstream structure

Figures 1 and 2 shown the bitstream organization for both original LTW and Embedded-LTW encoders respectively. As can be seen, in the LTW encoder, first we send the symbol, then the sign(only for significant coefficients) and the significant bits for every significant coefficient from the low fequency wavelet subband to the first level wavelet subband. On the other hand, the Embedded-LTW encoder firstly sends the symbol and the sign (for significant coefficients) from the low frequency wavelet subband to the first level wavelet subband. Then, after all the symbols map has been sent, the bits coerresponding to significant coefficients are sent in bitplane order (from MSB to LSB) in the same subband order. Remark that this order provides both SNR and spatial resolution. We have performed several measures over the bitstream in both original LTW and the Embedded-LTW encoders like symbols map size, significant coefficients sign size and significant coefficients bits. As Table I shows, the symbols map size is an important amount from the total bitstream size as well as the significant coefficients sign size(up to 20% of the total bitstream for Lena). It is obvious taken into account these numbers that the arithmetic encoder used to compress the sign cannot compact the sign because the probability distribution of a significant coefficient being positive or negative is nearby the same, as Shapiro asses in [3].

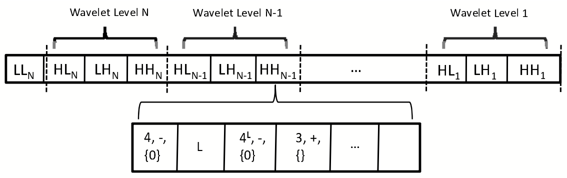

Fig. 1. LTW bitstream structure.

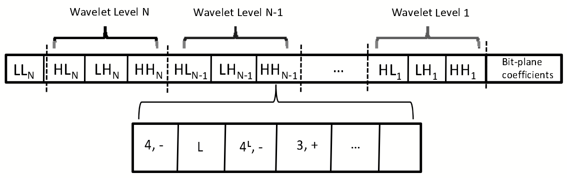

Fig. 2. Embedded-LTW bitstream structure.

TABLA I
BITSTREAM STRUCTURE SIZE(KB) FOR EMBEDDED-LTW FOR LENA.

rplanes; Q	bitrate (bpp)	Bits	Sign	Symbols Map	Total bitstream
2;0.96	1.0	3.24	6.25	22.51	32

TABLA II
BITSTREAM STRUCTURE SIZE(KB) FOR ORIGINAL LTW FOR LENA.

rplanes; Q	bitrate (bpp)	Bits	Sign	Symbols Map	Total bitstream
2;0.96	1.03	4.33	6.25	22.51	33.09

B. Sign coding

Typically, in an embedded wavelet image coder, the sign of a wavelet coefficient is coded into the bitstream immediately following the point at which that coefficient is coded as having significant magnitude. This sign information comprises a substantial portion of the available bitrate. For natural imagery, at rates of `.01` bpp and above, approximately 15-20% of the bitstream is dedicated to indicating the signs of wavelet coefficients. As shown in Table I for Lena image, using Embedded-LTW encoder, over 20% of the bit stream is used by the sign of significant wavelet coefficients. Thus significant improvements in the efficiency of sign coding will lead to corresponding significant overall improvements in compression performance.

In most wavelet image coding systems, the signs of transform coefficients are coded without compression. It is generally assumed that no correlation exists among the transform coefficient signs, and thus there is no gain from modeling or entropy coding the signs. Shapiro states that a quantized transform coefficient is equally likely positive as negative and thus one bit must be used to transmit the sign [3]. Said and Pearlman refer to the sign bit that is coded immediately after a coefficient is tagged as significant [2]. Li and Lei claim that "‘sign bits are at equilibrium between '0' and '1'"‘ [11]. Only recently have authors begun to explore the benefits of context modeling for sign coding. Possibly the first authors to consider context modeling for sign coding were Taubman and Zakhor, who used the sign of neighboring intraband pixels in their context modeling algorithm [12]. Other authors have employed similar sign coding context models, varying the number of intraband wavelet coefficients considered in the model and the number of derived contexts [13], [14], [4].

Both EBCOT and JPEG2000 standard consider local intraband neighbors as a context model [1], [15]. In this model, the contribution from the horizontal direction is formulated in Table III (identified by $\bar{h}$), where the relative positions of the neighboring nodes W and E are indicated in Figure 3. The sign information is available only for a significant neighbour. The vertical and diagonal contributions, $\bar{v}$, $\bar{d}_{45}$ and $\bar{d}_{135}$, are defined in a similar way. The sign bit $\hat{\chi}$ is then predicted by the rule outlined in Table IV, denoted by column $\hat{\chi}$. Instead of the sign bit itself, we encode the correctness of sign prediction ς, defined to be 1 if $\chi = \hat{\chi}$ and 0 otherwise.

EZBC [16] adopts the similar sign coding scheme

TABLA III

Contribution from the horizontal neighbours.

W	E	h
$significant,+$	$significant,+$	1
$significant,-$	$significant,+$	0
$insignificant$	$significant,+$	1
$significant,+$	$significant,-$	0
$significant,-$	$significant,-$	-1
$insignificant$	$significant,+$	-1
$significant,+$	$insignificant$	1
$significant,-$	$insignificant$	-1
$insignificant$	$insignificant$	0

TABLA IV

Look-Up table for sign coding.

h	$\bar{v}$	d_{45}	d_{135}	$\hat{\chi}$	Label
1	1	x	x	1	4
1	0	x	x	1	3
1	-1	x	x	1	2
0	1	x	x	-1	1
0	0	0	0	1	0
0	-1	x	x	1	1
-1	1	x	x	-1	2
-1	0	x	x	-1	3
-1	-1	x	x	-1	4

to EBCOT and JPEG2000 with an extension to the diagonal direction. This modification is found to improve coding efficiency slightly for some complex images such as Barbara.

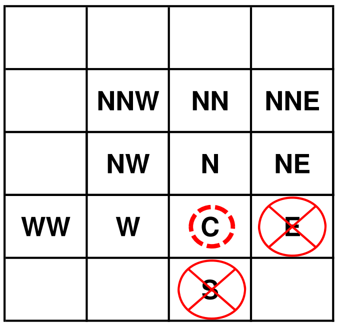

Fig. 3. Sign Intraband neighbourhood.

In [17] an indepth explanation of sign coding is presented. The main idea is finding correlations along and across edges. For example, in the HL subband we find high orrelacion along edges. The HL subbands of a multi-scale 2-D wavelet decomposition are formed from low-pass filtering vertically and high-pass filtering horizontally. The high-pass filtering detects vertical edges, and thus the HL subbands contain primarily vertical edge information. The LH subbands are oppositely defined, and contain mostly horizontal edge information. Given a vertical edge in an HL subband, it is reasonable to expect the transform coefficients along this edge to be positively correlated. This is because vertical correlation often remains very high along vertical edges in images. So, the pixels along these edges remain highly correlated in the vertical direction. Hence it is predicted that the associated strong edge in the HL subband will possess positive correlation in the vertical direction along the edge. Neighboring coefficients along the edge are considered valuable context information, and are expected to have the same sign as the coefficient being coded. This observation is independent of the type of wavelet filter employed. Regarding correlation across edges, in this case, the nature of the correlation is directly affected by the structure of the high pass filter. For Daubechies' 9/7 filters, wavelet coefficient signs are strongly negatively correlated across edges.

Having into account the LTW encoder symbol coding process, we found that we do not have information about some of the neighbours used in JPEG2000. Concretely, in the decoding process we do not have information about S, E, SW, SE neighbours (See Figure 3). So, using exclusively the four remaining nearest neighbours, we could not find the suited correlations to compact the sign. With the aim of finding correlations along and across edges, we have used a widely neighbourhood, adding NN,NNE,NNW,WW neigbours to the N,W,NE,NW ones. The main drawback we have found is the great amount of context using this eight neighbours. Remark that each neigbour have three posible states: insignificant, positive or negative (3^8 posibilities), and the actual coefficient being evaluated could be either positive or negative($2x(3^8)$ contexts). As this amount of context is not computationally efficient, we are evaluating several images so as to find their neighbourhood statistics trying to group behaviours reducing the context number to a maximun of five context as EBCOT and JPEG2000 does.

Althoug we still are analizing this statistics, in Table V we could see that both positive and negative correlations along edges exist taking only into account the North or West neighbour in HL and LH subbands respectively.

TABLA V

Six level probability distribution for Lena (HL subband).

HL Subband			
C	N	Prob.	$\Delta Prob.$
+	+	39.06	67.18
$-$	$-$	28.12	
LH Subband			
C	W	Prob.	$\Delta Prob.$
+	+	51.58	59.65
$-$	$-$	8.07	

IV. Embedded Run-length encoding

As a first aproximation to the coefficient sign compression, we have developed a new embedded version of LTW (RLE-Emb-LTW) which uses a Run-length encoding(RLE) technique to compress the sign. A RLE is a technique used to reduce the size of a repeating string of characters. This repeating string is called a run, typically RLE encodes a run of symbols into two bytes, a count and a symbol. RLE can compress any type of data regardless of its information content, but the content of data to be compressed affects the compression ratio.

In this encoder version, the quantization step is the same as in previous Embedded-LTW version. In order to exploit the symbol generation step, when we found a coefficient to be significant we store the count of previous coefficients with the same sign for every wavelet subband in the same order than the coding step needs to know the coefficient sign. Then, during the coding step, using a flipping technique, we only send the first coefficient sign (positive or negative) and a number showing the count of next coefficients with the same sign. In this way, while we are codign the symbol map, for every significant coefficient found we count down this run count. When this run count reach to 0, we encode a new run count but without coding the sign. Remark that we are using a flipping method, so the new run has the opposite sign to the previous one.

In the decoding process, we first decode the first sign symbol and the run length. Then we began to decode the LTW symbols map counting down the run count decoded when a significant coefficient is found. When this run count reach to 0 then we extract from the bitstream a new run count that corresponds to the opposite sign using a flipping technique.

The problem we have found using this method is that due to the way LTW encode the symbols map, the run sizes are very small, being the more frequently sizes 0, 1 or 2. So, there is no gain in using the RLE to encode the sign. Moreover, we need more bits to encode the run size than raw encoding the sign.

TABLA VI
Six level Run count entropy for Lena.

Run size	Positive	Negative	Prob.	Entropy
0	25	32	0.532	1.720
1	18	14	0.299	
2	7	3	0.093	
3	1	2	0.028	
4	1	2	0.028	
5	2	0	0.018	

To improve this behaviour, we decided to encode the run size as symbols for an arithmetic encoder. Before adopting this solution, we have obtained the distribution probability of run sizes for lena image. As Table VI shows, the entropy of this source is arround 1.7 bpp which is not worth enough to compress the sign.

V. Experimental results

Figures 4 and 5 show PSNR for Lena and Cafe images at several compression rates encoded with SPIHT, JPEG2000, Original LTW and Embedded-LTW encoders. At medium and high compression rates Embedded-LTW obtain similar results than SPIHT and JPEG2000. The slightly PSNR loss of Original LTW lies mainly in the lowest compression rate. As LTW is a non-embedded encoder, we use a rate control algorithm to obtain the target bitrate, but the real bitrate obtained uses to be under the target one.

At present, Embedded-LTW uses internal rate-control estimation to perform the quantization step. This lead to a R/D loss because the rate-control estimation uses to compress more than the desired Target bitrate [18].

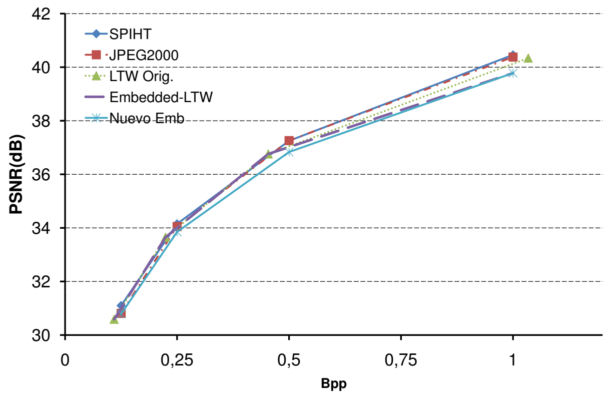

Fig. 4. PSNR (dB) for Lena(512x512).

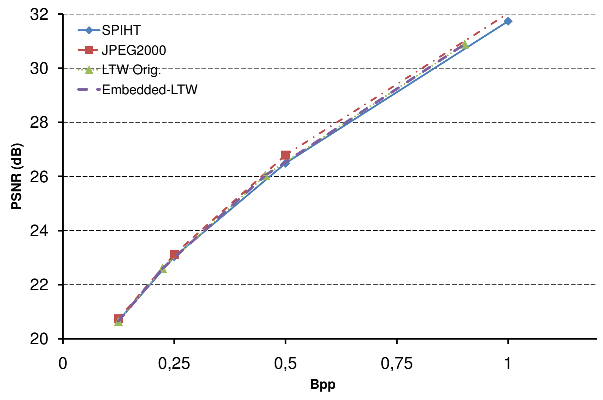

Fig. 5. PSNR (dB) for Cafe(2048x2560).

Although not shown here, RLE-Embedded-LTW encoder shows a similar behaviour than Embedded-LTW.

VI. Conclusions

Obtaining an embedded bit stream is a nice feature because you could obtain both SNR and spatial scalability. The coefficients sign coding is one of the more important steps when we try to obtain an embedded bit stream. In LTW case, the coefficients sign size is over a 20% of the total bit stream size. We have developed several proposals to encode the sign like using a two symbols arithmetic encoder (positive and negative), or using a RLE encoder. No one of these versions obtain coefficient sign compression enough to obtain better PSNR values than state of the art embedded encoders.

As future work, we pretend to develope an intraband context model sign coding method using the coefficient sign neighbourhood information. A first statistical results have been presented and they promise to obtain good compression results.

Grants

This work was funded by Spanish Ministry of education and Science under grant TIC2003-00339 and by Valencia Government under grant ARVIV-2007-045.

Referencias

[1] "Jpeg 2000 part i final int. std.," September 2000, ISO/IEC JTC 1/SC 29/WG 1 N1890.

[2] A. Said and A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits, Systems and Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.

[3] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445–3462, December 1993.

[4] X. Wu, "High-order context modeling and embedded conditional entropy coding of wavelet coefficients for image compression," in *31st Asilomar Conf. on Signals, Systems, and Computers*, 1997, pp. 1378–1382.

[5] K. Ramchandran Z. Xiong and M. Orchard, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing*, vol. 6, pp. 677–693, 1997.

[6] X. Wu, *The Transform and Data compression Handbook*, chapter Compression of wavelet transform coefficients, pp. 347–378, CRC Press, 2001.

[7] A. Drukarev A. Islam C. Chrysafis, A. Said and W.A. Pearlman, "Sbhp a low complexity wavelet coder," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2000.

[8] J. Oliver and M.P. Malumbres, "Fast and efficient spatial scalable image compression using wavelet lower trees," in *IEEE Data Compression Conference*, Snowbird, UT, 2003.

[9] A. Said Yushin Cho, W. A. Pearlman, "Low complexity resolution progressive image coding algorithm: Progres (progressive resolution decompression)," in *IEEE International Conference on Image Processing*, September 2005.

[10] Brian Nutter Tanja Karp Jiangling Guo, Sunanda Mitra, "A fast and low complexity image codec based on backward coding of wavelet trees," *Data Compression Conference*, 2006.

[11] J. Li and S. Lei, "An embedded still image coder with rate-distortion optimization," in *Porceedings SPIE*, 1997, pp. 36–47.

[12] D. S. Taubman and A. Zakhor, "Multirate 3-d subband coding of video," *IEEE Trans. Image Processing*, vol. 3, pp. 572–588, September 1994.

[13] P. Sementilli A. Bilgin and M. Marcellin, "Progressive image coding using trellis coded quantization," *IEEE Trans. Image Processing*, vol. 8, pp. 1638–1643, November 1999.

[14] A. Zandi E. Schwartz and M. Boliek, "Implementation of compression with reversible embedded wavelets," in *Proceedings SPIE Vol 2564*, 1995, pp. 32–43.

[15] D. S. Taubman and M. W. Marcellin, "Jpeg2000: Image compression fundamentals, standards and practice," *Norwell, MA: Kluwer*, 2002.

[16] Shih-Ta Hsiang, "Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling," in *Data Compression Conference*, 2001, pp. 83–92.

[17] Aaron Deever and Sheila S. Hemami, "What's your sign? efficient sign coding for embedded wavelet image coding," in *Data Compression Conference*, 2000, pp. 273–282.

[18] J. Oliver O. Lopez, M. Martinez-Rach and M.P. Malumbres, "Impact of rate control tools on very fast non-embedded wavelet image encoders," in *Proceedings SPIE IS&T Electronic Imaging Vol 6508*, 2007, pp. 1 – 8.