M-LTW: A Fast and Efficient Non-Embedded Intra Video Codec[1]

O.López[a], M.Martínez-Rach[a] P. Piñol[a], J.Oliver[b] and M.P.Malumbres[a]

[a]Miguel Hernández University, Avda. Universidad s/n, Elche, Spain 03202,
{otoniel,mmrach,pablop,mels@umh.es};
[b]Technical University of Valencia, Camino de Vera s/n, Valencia, Spain 46022,
{joliver@disca.upv.es}

Abstract. Intra video coding is a common way to process video material for applications like professional video editing systems, digital cinema, video surveillance applications, multispectral satellite imaging, HQ video delivery, etc. Most practical intra coding systems employ JPEG encoders due to their simplicity, low coding delay and low memory requirements. JPEG2000 is the main candidate to replace JPEG in this kind of applications due to the excellent R/D performance and high coding flexibility. However, its complexity and computational resources required for proper operation could be a limitation for certain applications. In this work, we propose an intra video codec, M-LTW, which is able to reach very good R/D performance results, as well as JPEG2000 or H.264 INTRA, with faster processing and lower memory usage.

Keywords: image and video coding, tree-based wavelet coding, integer lifting transform, low complexity coding.

1. Introduction

A wide variety of video compression schemes have been reported in the literature. Most of them are based on the DCT transform and motion estimation/compensation techniques. However, a lot of research interest was focused on developing still image and video wavelet coders due to the great properties of wavelet transform. Most wavelet-based video encoding proposals are strongly based on inter-coding approaches, which require high-complexity encoder designs as counterpart to the excellent R/D performance benefits. However, some applications like professional video editing, digital cinema, video surveillance applications, multispectral satellite imaging, HQ video delivery, etc. would rather use an intra coding system that is able to reconstruct a specific frame of a video sequence as fast as possible and with high visual quality.

So, the strength of an intra video coding system relies on the ability to efficiently exploit the spatial redundancies of each video sequence frame avoiding complexity in

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

the design of the encoding/decoding engines. There are several still image codecs that get very good R/D (Rate/Distortion) results. Unfortunately, most of them propose complex algorithms to achieve the pursued R/D performance. As a consequence of the higher computational complexity demanded by these coders, their software (even hardware) implementations would require powerful processors with enough computational resources to cope with the algorithm requirements. For example, the JPEG2000 [1] standard uses a large number of contexts and an iterative time-consuming optimization algorithm (called PCRD) to improve coding efficiency, increasing the complexity of the encoding engine. Something similar happens with H.264/AVC [2] INTRA coding, where a powerful spatial prediction scheme with context modeling and rate-distortion optimization is employed in order to efficiently exploit spatial redundancies.

In this paper, we propose a new lightweight and efficient intra video coder, M-LTW (Motion Lower-Tree Wavelet), based on the LTW algorithm [3]. The main contribution of LTW is the way that it builds the significance map when coding each video frame. As other tree-based wavelet coders, it is based on the construction and efficient coding of wavelet coefficient trees. However, it does not use an iterative loop in order to determine the significant coefficients and to assign them bits. It builds the significant map in only one step by using two symbols for pruning tree branches, and codes the significant coefficients also in one step.

Several rate control schemes have been reported in the literature. Most of them are applied to DCT transform like Test Model Near-term version 5 (TMN5) [4] used in H.263 standard or the MPEG Test Model 5 (TM5) [5]. In [6] the authors propose a new rate control scheme based on Game Theory and a deep introduction to rate control is made.

Since the LTW encoding engine is non-embedded and it is based on DWT transform, we have proposed a low complexity rate control tool to encode the original video sequence to a user-defined target bitrate, in order to increase the flexibility of M-LTW video encoder and allowing LTW to work with rate-adaptive applications. Also, we have changed the overall codec to work with fixed point arithmetic. So, the DWT transform may use the original lifting floating-point approach or an equivalent DWT integer lifting version which will speed up the DWT transform step. As a secondary benefit, the required memory space is halved (16-bit integer data types instead of 32-bit floats).

The organization of the paper is the following one: in section 2 the M-LTW algorithm is described, making special emphasis in the LTW spatial coding and the proposed rate control tool. In section 3, we show some evaluation results using as performance metrics rate/distortion, complexity and memory requirements. Finally, in section 4 some conclusions and future work are drawn.

2. M-LTW Coder Description

As shown in figure 1, the proposed M-LTW intra video encoder is composed of (a) DWT module, which computes the Discrete Wavelet Transform, (b) the proposed rate control tool, which adjusts quantization parameters to fit a user-specified target

bitrate, (c) the coding engine, which is based on the LTW still image encoder, (d) an arithmetic entropy encoder, which is fully integrated with the LTW encoder, and (e) a format bitstream module to multiplex the info delivered by rate control (quantization parameters) and the two data sets delivered by LTW (entropy encoded significant map symbols and the raw coefficient values as explained later).

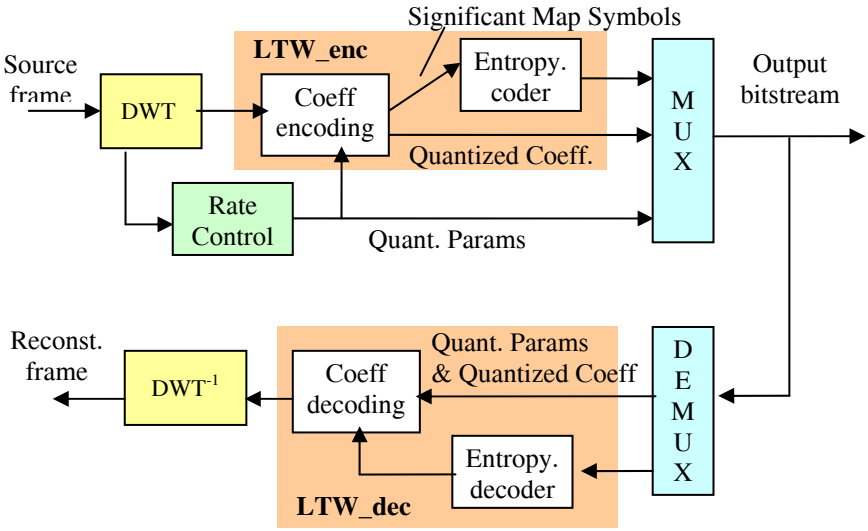

Fig. 1. M-LTW block diagram

We have also developed two versions of the DWT transform module: (a) a standard lifting version based on 7/9 biorthogonal filter (as many other wavelet encoders use) and (b) a version of the former one using an integer-to-integer lifting scheme based on [8] and [9]. We have performed the expansion factor of DWT by an approximation to integer operations (multiplication and shift). In this manner we avoid three extra lifting steps at the expense of making the DWT not reversible. The proposed approximation does not introduce a meaningful error, being the difference respect to the regular lifting scheme negligible.

The M-LTW is designed to work with fixed point arithmetic, with the exception of the standard lifting version of DWT transform module, that uses float data types for its computations.

2.1. LTW: The intra coding engine

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 [7]) 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 N^{th} (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 and they are 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.

2.2. M-LTW Rate Control tool

The proposed rate control is founded on the definition of a simplified model of LTW coding engine. Applying this idea to the LTW encoder, the simplified coding model will lead us to get an initial and fast bitrate estimation for different values of the coarser quantizer *rplanes* (from 2 to 7). This estimation is computed as follows: for

each specific value of *rplanes*, the probability distribution of significant and insignificant symbols is calculated. Then, the bit rate estimation (E_{bpp}) for each *rplanes* value is calculated, taking into account: (a) an estimation of the bit-rate that the arithmetic encoder will produce, and (b) the number of bits required to store the sign and significant bits (which are binary coded). The resulting estimation gives a biased measure of the real bit rate for all operative bit-rate range (from 0.0625 to 1 bpp), so we will reduce the error by means of a correction factor calculated from the Kodak image set [10].

After that, the target bit-rate, T_{bpp} will establish the proper value of the quantization parameter *rplanes* ($E_{bpp}(rplanes) > T_{bpp} > E_{bpp}(rplanes+1)$). In order to determine the proper value of the quantization parameter Q, the bit rate progression from the current *rplane* to the next one follows a second order polynomial curve with a common minimum. So, with the estimated values ($E_{bpp}(rplanes)$ and $E_{bpp}(rplanes+1)$), we can build the corresponding expression that will supply the estimated value of Q for a given target bitrate.

To perform the rate control in the overall video sequence, we have extended the rate control explained above by using a very simple approach. Firstly, we apply the proposed rate control algorithm to the first frame, in order to estimate the values of *rplanes* and Q quantization parameters that produce the frame bitrate budget. After coding the first frame, we compute the estimation error, so we will try to compensate it when coding the following frames. We will do that keeping the same value of *rplanes* and estimating the appropriate values for Q based on the observed error. When the estimated error reaches a threshold, the algorithm detects a scene change and runs the initial estimation algorithm in order to re-estimate more suitable *rplanes* and Q parameters. Then, the accumulated error will be corrected gradually in order to avoid great R/D alterations. After several experiments, the accuracy of the proposed error control was always better than 98.5% (worst case at very low target bitrates).

3. NUMERICAL RESULTS

In addition to R/D performance we will also employ other performance metrics like coding delay and memory consumption. All the evaluated encoders have been tested on an Intel PentiumM Dual Core 3.0 GHz with 1Gbyte RAM Memory. We have selected H.264 (Baseline, JM10.2), M-JPEG2000 (Jasper 1.701.0), M-LTW and M-LTW_Int (the integer version of M-LTW), since their source code is available for testing. The correspondent binaries were obtained by means of Visual C++ (version 2005) compiler with the same project options and under the above mentioned machine. The test video sequences used in the evaluation are: Foreman (QCIF and CIF), Hall (QCIF and CIF), Container (QCIF and CIF), News (QCIF and CIF), Mobile (ITU→576p30) and Station2 (HD→1024p25).

Table 1 shows the R/D evaluation of the proposed encoders. In general, the M-LTW obtains the best results (about 0.5 dB with respect to M-JPEG2000 in Foreman QCIF). The difference is higher with ITU and HD formats (around 2 dB with respect to H.264). At these sizes, optimal DWT decompositions can be exploited. The M-LTW_Int encoder has slightly lower PSNR results than H.264. The lower

performance of the integer version is mainly due to the arithmetic precision loss, which is more noticeable at lower compression rates.

Table 2 shows the encoding delay for all encoders under evaluation. As expected, H.264 is the slowest encoder and M-LTW is one of the fastest encoders. All M-LTW versions are faster than M-JPEG2000, specially the integer version that performs the encoding process six times faster on average than M-JPEG2000.

In Table 3, the memory requirements of different encoders under test are shown. The M-LTW needs only the amount of memory to store the source image (in-line processing is another feature of LTW encoder) and an extra of 1.2 KB basically used to store the histogram of significant symbols, required by the rate control algorithm. M-JPEG2000 requires two times the memory of M-LTW, and H.264 needs six times the memory of M-LTW for QCIF size and eight times for CIF size. Note that M-LTW_Int could be implemented using 16-bit integer, reducing to the half the amount of needed memory.

Table 1. PSNR (dB) with different bit-rate and coders

Codec/Bitrate (Kb/frame)	H.264	M-JPEG 2000	M-LTW	M-LTW _Int
Foreman (QCIF 176x144, 30Hz)				
2.36	22.86	19.99	**23.03**	23.01
7.40	**28.72**	28.00	28.69	28.58
20.49	**35.36**	34.53	34.99	34.40
33.73	39.24	38.78	**39.37**	37.62
Mobile (ITU 640x512, 30Hz)				
38.08	27.04	28.42	**28.59**	28.48
119.93	32.29	32.39	**32.57**	32.26
213.36	35.29	35.07	**35.40**	34.75
386.23	38.59	38.41	**38.87**	37.21
Station2 (HD 1920x1024, 25Hz)				
93.92	30.49	32.35	**32.45**	32.19
180.00	32.58	34.36	**34.49**	34.06
604.64	37.55	38.66	**39.02**	37.73
1117.53	40.37	40.76	**41.38**	39.08

Table 2. Execution time comparison of the coding process including DWT (time in seconds)

Codec/Bitrate (Kb/frame)	H.264	M-JPEG 2000	M-LTW	M-LTW _Int
CODING Hall (QCIF 176x144, 30Hz)				
2.70	121.92	4.04	0.86	0.51
7.77	137.18	4.39	1.07	0.71
19.54	165.67	4.55	1.57	1.10
29.50	184.67	4.87	1.97	1.41
CODING News (CIF 352x288, 30Hz)				
14.91	531.40	15.63	3.96	2.62
23.62	559.45	15.20	4.26	2.81
57.73	650.47	15.54	5.98	3.94
89.91	720.44	16.43	7.17	4.95

Table 3. Memory requirements for evaluated encoders (KB) (Results obtained from Windows XP task manager, peak memory usage column)

Codec/ Format	H.264	M-JPEG 2000	M-LTW	M-LTW _Int
QCIF	6508	2264	1104	1104
CIF	13016	3920	1540	1540

Figure 2 shows the maximum frame rate for all evaluated encoders at different sizes for an average PSNR video quality of 30 dB. The integer version of M-LTW is the fastest of all encoders and it can encode an ITU size sequence in real time.

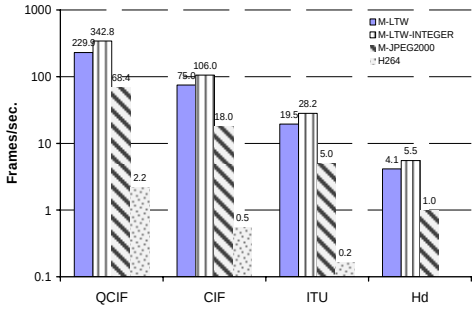

Fig. 2. Maximum frame rate for an average R/D of 30dB

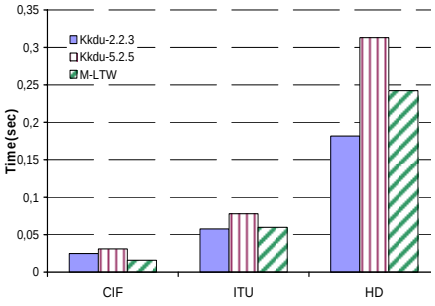

Fig. 3. Execution time comparison (end-to-end) of the coding process

The M-LTW implementation was developed finding the optimizations for maximizing R/D performance, so its software code is not optimized, just like H.264 and JPEG2000 reference software. However, we have compared its performance with respect to a fully optimized implementation of JPEG2000: Kakadu [12], in order to evaluate if a full optimization of M-LTW will be worth the effort. For that purpose, we have used two versions of Kakadu software: (a) version 2.2.3, compiled without optimization options, and (b) the last version 5.2.5 which is fully optimized including multi-thread multi-core hardware capabilities.

As shown in figure 3, M-LTW is a very fast encoder even though not being fully optimized. The speed of M-LTW lies on the simple engine coding model. M-LTW is approximately 2 times faster than Kkdu-5.2.5 for News CIF sequence for a PSNR of 32dB. For HD images, M-LTW is slower than Kkdu-2.2.3, due to the cache page miss fail of the lifting DWT implementation. In terms of R/D, there are slightly differences between all codecs as shown at table 4. For small and medium size images M-LTW outperforms KKDU at medium and high compression rates. For larger images, M-LTW has slightly lower PSNR than both versions of Kakadu, but these differences are not perceptible when PSNR is over 38dB as concluded in [11].

Regarding to memory requirements, M-LTW needs only the amount of memory required to store the source image, while Kakadu memory requirements are independent of the image size due to its DWT block-based implementation.

Table 4. PSNR (dB) comparison between Kakadu and M-LTW

Codec/ (Kb/frame)	KKDU 2.2.3	KKDU 5.2.5	M-LTW
News (CIF, 30Hz)			
14,91	27.63	27.44	**27.74**
23,62	30.27	29.96	**30.42**
36.75	33.33	33.31	**33.36**
57,73	**37.26**	37.10	36.89
Mobile (ITU, 30Hz)			
38.08	28.59	28.39	**28.61**
119.93	32.56	32.52	**32.62**
213.36	35.34	35.34	**35.47**
386.23	38.85	38.89	**38.90**
Station2 (HD, 25Hz)			
93.92	**33.79**	33.70	33.62
180.00	**36.16**	36.15	36.08
604.64	**41.11**	41.11	40.96
1117.53	**43.18**	43.18	42.94

5. Conclusions

In this paper we have presented a fast an efficient intra video coder, M-LTW, which is based on the non-embedded LTW image coder. We have proposed a fast rate control algorithm to both M-LTW encoder versions. After evaluating M-LTW performance in terms of R/D, execution time and memory consumption, it exhibits the best trade-off between R/D performance, coding delay (3 times faster than M-JPEG2000 and 108 times faster than H.264) and overall memory usage (half the memory of M-JPEG2000 and 6 times less than H.264). Also, the M-LTW coder is able to encode in real time an ITU video signal with very low memory demands and good R/D performance at moderate to high compression rates (up to 2 dB with respect to H.264 in the HD sequence).

For further evaluation, we have compared M-LTW coder with a highly optimized version of JPEG2000 (Kakadu), being also competitive in terms of coding delay (up to 2 times faster than Kkdu for small and medium size images) and R/D performance (0.4 dB for CIF, and 0.1 dB for ITU at medium and high compression rates). So, a fully optimization process will make M-LTW even faster and with lower memory requirements (with line-based or block-base DWT implementations).

References

1. ISO/IEC 15444-1. Jpeg 2000 image coding system. Part 1: core coding system, 2000.
2. ISO/IEC 14496-10:2003. Coding of audiovisual objects part 10: advanced video coding for generic audiovisual services, 2003.
3. J. Oliver, M. P. Malumbres, “Fast and efficient spatial scalable image compression using wavelet lower trees,” in Proc. IEEE Data Compression Conf., Snowbird, UT, March 2003.

4. Video Codec Test Model for the Near-Term 5 (TMN5), ITU-T SG 15 Experts Group for Very Low Bitrate Visual Telephony, 1995.
5. "Test Model 5 (TM5)," ISO/IEC JTC1/SC29/WG11, Document N400, 1993.
6. Ishfaq Ahmad and Jiancong Luo, "On using Game Theory for Perceptually Tuned Rate Control Algorithm for Video Coding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 2, Feb. 2006, pp. 202-208.
7. A. Said, A. Pearlman. "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," IEEE CSVT, vol. 6, nº 3, pp.243-250, 1996
8. A.R. Calderbank, I. Daubechies, W. Sweldens, and B-L.Yeo,"Wavelet transforms that map integers to integers", Applied & Computational Harmonic Analysis, 5(3) pp.332–369, 1998.
9. I. Daubechies, W. Sweldens, "Factoring wavelet transforms into lifting steps", Journal of Fourier analysis and applications, Vol. 4, Nº 3, pp 247-269, 1998.
10. Center for Image Processing Research - Electrical, Computer, and Systems Engineering Dept - Rensselaer Polytechnic Institute http://www.cipr.rpi.edu/resource/stills/ kodak.html
11. M. Martinez-Rach, O. López, P. Piñol, J. Oliver, M.P. Malumbres, "A Study of Objective Quality Assessment Metrics for Video Codec Design and Evaluation", in Proc. IEEE Computer Society, pp. 517-524, 2006
12. Kakadu Software, http://www.kakadusoftware.com/