Motion-LTW: a fast and efficient intra video coding system with low memory consumption

Otoniel López, Miguel Martínez, Pablo Piñol, Manuel P. Malumbres
Dpto. de Física y Arquitectura de Computadores
Universidad Miguel Hernández
03202 Elche
{Otoniel, mmrach, pablop, mels}@umh.es

José Oliver
Dpto. de Informatica de Sistemas y Computadores
Univ. Pol. Valencia
46022 Valencia
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. So, a fast access to video content and the high quality reconstructed video are primary concerns. Most practical intra coding systems employ JPEG encoders due to their simplicity, low coding delay and low demanding memory resources. JPEG2000 is call to be the main candidate for 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 it is able to reach very good R/D performance results, as well as JPEG2000 or H.264 INTRA; it is very fast and requires low memory resources.

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 will rely on the ability to efficiently exploit the spatial redundancies of each video sequence frame avoiding complexity in the design of the encoding/decoding engines. There are several highly-optimized intra codecs that get very good R/D (Rate/Distortion) results. Unfortunately, many of these coding optimizations involve high complexity, requiring faster and more expensive processors. For example, the JPEG2000 [4] 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 [5] INTRA coding, where a powerful spatial prediction scheme with context modeling and rate-distortion optimization is employed in order to efficiently exploit spatial redundancies, as it does.

In this paper, we propose a new lightweight and efficient intra video coder, M-LTW (Motion Lower-Tree Wavelet), based on the LTW algorithm [9]. 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 using two symbols for pruning tree branches, and codes the significant coefficients also in one step.

In order to reduce, even more, the complexity of M-LTW we have performed some

modifications to the original LTW: (a) First, we have added a low complexity rate control tool [6] to encode the original video sequence to a user-defined target bitrate, since LTW is a non-embedded encoder. (b) We have changed the overall codec to work with fixed point arithmetic. So, we exchanged the original floating-point DWT for an equivalent DWT integer lifting version in order to speed up the DWT transform step. As a secondary benefit, the required memory space is halved (16-bit integer data types are used instead of 32-bit floats).

The organization of the paper is the following one: in section 2 the LTW algorithm is briefly described including the integer lifting version of the DWT transform step. In section 3, we describe the rate control tool employed in the M-LTW encoder. In section 4, we show some evaluation results using as performance metrics rate/distortion, complexity and memory requirements. Finally, in section 5 some conclusions and future work are drawn.

2. LTW: Lower tree wavelet coder

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 [10]) 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.

2.1. LTW_Int: Lower tree wavelet integer

To carry out a fast integer version of M-LTW, we have developed the DWT with an integer-to-

integer lifting scheme based on [1] and [3]. We have performed the expansion factor of DWT by an approximation to integer operations (multiply and shift). In this manner we avoid three extra lifting steps at the expense of making the DWT not reversible. Since we are interested in lossy compression, the proposed approximation does not introduce a meaningful error, being the difference respect to the regular lifting scheme negligible.

Concerning the LTW encoder engine, we have converted all float operations to integer ones.

Relating to the quantization process, this is similar to the one used by M-LTW described in previous section. The main difference lies in the scalar uniform quantization process, which it is performed using only fixed point arithmetic operations.

3. Fast rate control algorithm for non-embedded encoders

In [6], we proposed three lightweight rate control tools for non-embedded image encoders. These tools will predict the proper quantization values that lead to a final bit rate close to the target one. In particular, we proposed several bit rate prediction methods with increasing complexity and accuracy. We have chosen LTW [9] for evaluation purposes, although other encoders with scalar quantization could also be used. In [7] we evaluated the impact of these rate control tools in terms of R/D performance, coding delay and memory consumption. From this evaluation we have chosen the rate control tool algorithm based on a trivial coding model of the encoder, since it is the one that exhibits the overall best results.

3.1. Rate control based on a trivial coding model

The proposed rate control is founded on the definition of a simplified model of the 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) the estimation of the bit-rate produced by the arithmetic encoder 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). We reduce the error with a correction factor calculated from the Kodak image set [2].

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, we noticed that 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 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, based on the observed error, the appropriate values for Q.

In Figure 1, we can see the accuracy of the proposed rate control algorithm at different target bitrates.

4. 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, Intra mode, 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.

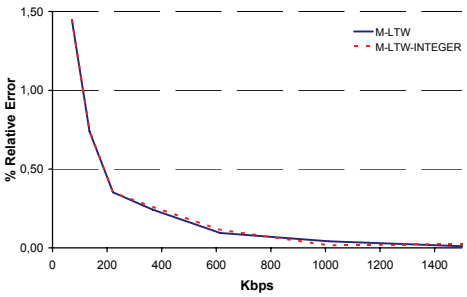

Figure 1. Rate control accuracy for Foreman (Qcif) sequence

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 coding 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.

Codec/ Bitrate (Kb/frame)	H.264	Motion JPEG 2000	M-LTW	M-LTW _Int
CODING Hall (QCIF, 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, 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 1. Execution time comparison of the coding process including DWT (time in seconds)

Table 2 shows the R/D evaluation of proposed encoders. The M-LTW is the one that bets results obtains (around 0.5 dB with respect to M-JPEG2000 in Foreman QCIF). The difference is higher at ITU and HD formats (up to 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 integer version is mainly due to the arithmetic precision loss, which is more outstanding at lower compression rates.

Codec/ Bitrate (Kb/frame)	H.264	M-JPEG 2000	M-LTW	M-LTW _Int
Foreman (QCIF, 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, 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, 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. PSNR (dB) with different bit-rate and coders

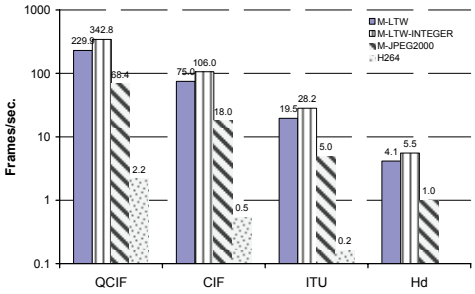

Figure 2. FrameRate for an average R/D of 30dB

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

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

(a) 39.29dB (b) 39.37dB (c) 39.94dB (d)38.26dB

(e) Original (Not compressed)

Figure 3. Foreman sequence (QCIF) compressed at 1012.16Kbps (Frame 14). a) H264 b) JPEG2000 c)M-LTW d) M-LTW_Int and e) Original (not compressed)

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) 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 memory needed.

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

Figure 3 we can observe a subjective comparison between all codecs for Foreman sequence (QCIF) at low compression rate. Although PSNR value shows differences greater than 0.5 dB between them, there are no subjective differences due to the saturation effect over 38dB where differences on PSNR are not perceptible as concluded in [8]. So, in figure 4 we show a subjective comparison of codecs under evalution using the Mobile sequence (ITU) at higher compression rates. As it can be appreciated, there are slight quality differences between M-LTW and H.264. Moreover, although M-LTW_Int gets lower PSNR, it has better subjective quality than JPEG2000 and slightly less quality than M-LTW.

5. Conclusions

In this paper we have presented two different implementations of intra video encoders based on the non-embedded LTW image encoder and we have compared their performance with H.264 and M-JPEG2000 encoders in R/D, execution time and memory consumption. To accomplish with the target bitrate, we have added one of the rate control algorithms presented in [6] to both M-LTW encoder versions. Here, we have presented the first preliminary results that show how the M-LTW proposal is the one that 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).

We have developed a very fast M-LTW integer version using an integer-to-integer DWT with lifting scheme that is able to encode in real time an ITU video signal (INTRA coding only) 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 for station2 HD sequence).

6. Acknowledgement

This work was funded by the Spanish Ministry of Education and Science under grant TIC2003-00339.

(a) 27.11dB

(b) 28.73dB

(c) 28.75dB

(d) 28.63

(e) Original (Not compressed)

Figure 4. Mobile Sequence (ITU) compressed at 1142Kbps (Frame 27). a) H264 b) JPEG2000 c)M-LTW d) M-LTW_Int and e) Original (not compressed)

References

[1] A. R. Calderbank, I. Daubechies, W. Sweldens, and B-L.Yeo, "Wavelet transforms that map integers to integers," Applied and Computational Harmonic Analysis, vol. 5, no. 3, pp. 332–369, July 1998.

[2] Center for Image Processing Research - Electrical, Computer, and Systems Engineering Department - Rensselaer Polytechnic Institute http://www.cipr.rpi.edu/resource/stills/kodak.html

[3] I. Daubechies, W. Sweldens, "Factoring wavelet transforms into lifting steps", Journal of Fourier analysis and applications, Vol. 4, Nº 3, pp 247-269, 1998.

[4] ISO/IEC 15444-1. Jpeg 2000 image coding system. Part 1:core coding system, 2000.

[5] ISO/IEC 14496-10:2003. Coding of audiovisual objects part 10:advanced videocoding. ITUT Recommendation H264 Advanced video coding for generic audiovisual services, 2003.

[6] O. López, M. Martinez-Rach, J. Oliver and M.P. Malumbres, "A heuristic bit rate control for non-embedded wavelet image encoders", 48th International Symposium ELMAR 2006, Jun 2006.

[7] O. López, M. Martinez-Rach, J. Oliver and M.P. Malumbres, "Impact of rate control tools on very fast non-embedded wavelet image encoders", Visual Communications and Image Processing 2007, Jan 2007.

[8] 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

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

[10] 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