A Smoothly Scalable and Fully JPEG2000-Compatible Video Coder

Marco Cagnazzo [1,2], Thomas André [1], Marc Antonini [1], Michel Barlaud [1]

[1]I3S Laboratory, UMR 6070 of CNRS, University of Nice-Sophia Antipolis
Bât. Algorithmes/Euclide, 2000 route des Lucioles - BP 121 - 06903 Sophia-Antipolis Cedex, France
Phone: 33(0)4.92.94.27.21 - Fax: 33(0)4.92.94.28.98 - {cagnazzo,andret,am,barlaud}@i3s.unice.fr

[2]Dipartimento di Ingegneria Elettronica e delle Telecomunicazioni,
Università Federico II di Napoli, via Claudio 21 - 80125 Napoli, Italy

***Abstract*— In this paper we analyze the scalability properties of the JPEG2000-compatible video encoder presented in [1], and we improve its performances by presenting a new technique for an efficient Motion Vectors (MVs) encoding, producing a motion bitstream also compatible with JPEG2000. Our study shows that, thanks to our encoding strategy and to our peculiar temporal filters, scalably encoded sequences have the same or almost the same quality than non-scalably encoded ones: this is what we call *smooth scalability*. We also compared our encoder performances with the recent H.264 standard, showing comparable or sometimes better performances.**

I. Introduction

Recent video coding standards are characterized by good performances but limited scalability [2], [3]. Moreover, for these techniques, scalability has a remarkable cost in terms of performances, that is, scalable encoding is far from optimal results achievable without scalability. Nevertheless, as long as networks keep on being as heterogeneous as they are, scalability, and above all smooth scalability (i.e. with little or no performances degradation), remains a priority among video encoder features. Another disadvantage of recent standards is that they do not provide any convergence with the new still images coding standard JPEG2000 [4], as they use mainly the DCT transform.

In this paper we analyze a fully scalable video codec with smooth scalability, whose basic encoding algorithm has been described in [1]. Its performances are comparable to state-of-the-art video encoders, and it has a full compatibility with JPEG2000, which allows to exploit both software and hardware solution currently developed for it. We propose a simple but efficient and scalable technique for Motion Vectors (MVs) coding. We study in details the smooth scalability offered by the codec, and we compare its performances with the H.264 standard video codec.

The paper has the following structure. In Section II, we recall the encoding algorithm, with the introduction of the proposed technique for Motion Vectors encoding. Section III details scalability features. Global performances and conclusions are given in Section IV.

Part of this work was supported by the IST programme of the EU in the project IST-2000-32795 SCHEMA.

II. Encoding algorithm

The proposed encoder is based on Motion Compensated Three-Dimensional Wavelet Transform (WT). Here we recall its basic features, in order to explain how we exploited them in achieving smooth scalability.

A. Motion Compensated Wavelet Video Coding

The input video sequence is first analyzed for Motion Estimation (ME), which employs both luminance and chrominance information [5]. Afterwards, it undergoes a Motion Compensated temporal transform, implemented via Lifting Scheme (LS) [6]–[8]. We used the $(N, 0)$ LS described in [5], which proved to be particularly suitable for the temporal analysis. In this filters family, the low-pass filter does not perform any operation, i.e., the low-pass subband is just a subsampled version of input signal. The output of the temporal filter is a set of M temporal subbands (SBs), which are then gathered in Groups of Picture (GOP), and encoded with JPEG2000. The scan-based implementation [9] of the temporal wavelet transform allows us to chose a GOP size independently from the number of temporal decomposition levels. For each GOP, the rate allocation among subbands must then be performed.

For a given total rate R_T, rate available for SBs is $R_{SB} = R_T - R_{MV}$ where R_{MV} is the rate required to encode MVs. The coding resources should be allocated among the different subbands. In other words, we have to find the rate R_i to use in encoding the i-th SB, with the constraint $\sum_{i=1}^{M} R_i = R_{SB}$. A Lagrangian algorithm computes the optimal allocation among subbands, i.e. the allocation which minimizes the output distortion for a given total rate. This algorithm takes as input the desired target rate and a parametric model of subbands rate-distortion (RD) curves, and outputs the optimal rate allocation vector. If $D_i(R_i)$ is the distortion of the i-th SB encoded at a rate R_i, it can be shown that the optimal allocation vector satisfies the equations:

$$\frac{\partial D_i}{\partial R_i}(R_i) = w_i \lambda \quad \forall i \in \{1, \ldots, M\} \tag{1}$$

where λ is the Lagrange multiplier, M is the number of SBs, and w_i is a suitable set of weights. The complexity of the rate allocation algorithm is negligible with respect to other parts

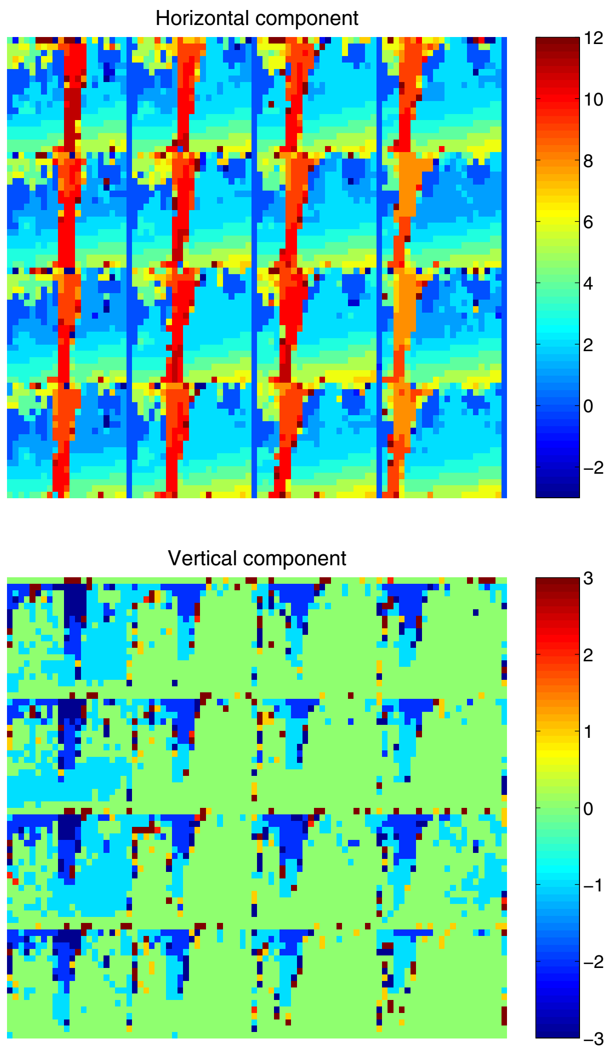

Fig. 1. Motion vectors images for the test sequence "Garden & Flowers"

of the encoder, such as the Wavelet Transform or the Motion Estimation.

B. *Motion Vector Encoding*

Motion vectors are encoded as follows. For each GOP and for each temporal decomposition level in a GOP, we gather all MVs, and we build two MVs images corresponding to the horizontal and vertical components respectively. In Fig. 1 we reported an example for the test sequence "Garden & Flowers", which is characterized by a dominant horizontal motion. These images can be encoded with JPEG2000, but some fine-tuning is needed in order to do this efficiently. As pointed out in our experiments, it is better to perform no wavelet transform on MVs (by specifying zero decomposition levels to the encoder): in other words, we encode MVs with EBCOT and we obtain lossless compression by including all bit-planes. We tested several vectors components arrangements

No. of Time Dec. Levels		1	2	3	4
"Flowers" sequence	Entropy	2.27	2.62	2.85	3.00
	Coding cost	1.71	2.21	2.55	2.83
"Bus" sequence	Entropy	3.54	4.01	4.34	4.54
	Coding cost	3.20	3.97	4.77	4.96

TABLE I

MVs ENTROPY AND CODING COST IN BIT PER VECTOR

in order to better exploit the temporal and spatial correlation between them, but the described configuration proved to be the most robust while being the simplest. This MV encoding technique also allows to separate MVs for each scalability level, as we will see in Section III.

The spatial correlation among neighboring vectors assures that this method achieves good coding rates. In Tab. I we report some results obtained on the test sequences "Garden & Flowers" and "Bus". We compared the MVs first order entropy (in bits per vector) to the encoding cost required by our technique, at different temporal decomposition levels. We note that the coding cost of the proposed technique is often inferior to the first order vectors entropy, even though we encode separately the two vectors components, and thus do not take advantage of their correlation.

III. Scalability

The proposed codec provides temporal, spatial, and quality scalability, ending up with a remarkable flexibility for an usage over heterogeneous networks. Temporal and spatial scalability is easily obtained by choosing suitably which subband to decode. Quality scalability is obtained thanks to embeddedness of JPEG2000 bitstream.

In a general way, a scalable bitstream has no better performance than what we can achieve by encoding directly the sequence at desired resolution, frame-rate and bitrate. So we call *scalability cost* the difference between the quality (expressed in terms of SNR) of the scalable bitstream decoded at a different resolution, frame-rate or bitrate from the original, and quality (SNR) that could have been achieved by directly encoding the original sequence with the desired parameters. A *smoothly scalable* encoder should have a null or very little scalability cost. In addition to this performance cost the complexity increase needed to perform a scalable encoding with respect to the non-scalable case should also be considered.

In order to define the notation, let $R^{(0)}$ be the bitrate available for the SBs. The non-scalable encoder must allocate these resources among the M SBs, finding the optimal rates vector $\mathbf{R}^{(0)} = \{R_i^{(0)}\}_{i=1}^{M}$, with the constraint $\sum_{i=1}^{M} R_i^{(0)} = R^{(0)}$.

A. *Bitrate scalability*

Bitrate scalability (or quality scalability) should allow to decode the bitstream at a set of predefined bitrates $R^{(j)}$ (with

Rate kbps	PSNR [dB] non-scalable	PSNR [dB] scalable	Scalability Cost [dB]
600	34.59	34.52	0.07
800	35.65	35.62	0.03
1000	36.53	36.47	0.06
1200	37.25	37.18	0.07
1400	37.87	37.81	0.06
1600	38.45	38.36	0.09
1800	38.95	38.88	0.07
2000	39.43	39.36	0.07

TABLE II

COST OF SNR SCALABILITY ("FOREMAN" SEQUENCE)

Rate kbps	PSNR [dB] non-scalable	PSNR [dB] scalable	Scalability Cost [dB]
375	35.22	35.05	0.17
500	36.40	36.27	0.13
625	37.35	37.22	0.13
750	38.19	38.07	0.12
1000	39.54	39.50	0.04
1250	40.70	40.67	0.03

TABLE III

COST OF TEMPORAL SCALABILITY ("FOREMAN" SEQUENCE)

$j = 1, \dots, N$) different from the encoding bitrate $R^{(0)}$. As each SB is already embeddedly encoded with JPEG2000, we could truncate its bitstream at an arbitrary rate $R_i^{(j)}$ (where the index i refers to the i-th SB), just provided that $\sum_{i=1}^{M} R_i^{(j)} = R^{(j)}$. However, with such a simple strategy, there is no optimal bitrate allocation among the SBs when decoding at the j-th bitrate. The solution is to compute in advance the bitrate allocation for each target bitrate $R^{(j)}$, finding the optimal vector $\mathbf{R}^{(j)} = \{R_i^{(j)}\}_{i=1}^{M}$. Then we encode the i-th subband with N quality layers that correspond to the bitrates $R_i^{(j)}$ for $j = 1, \dots, N$. At the decoder side, when we want the total bitrate $R^{(j)}$, it is sufficient to decode each SB at the quality level j. We note that the MV information is not affected by the bitrate scalability, as we still need the same vectors than for the non-scalable case.

Thus, the scalably decoded bitstream for each target rate is almost identical to the non-scalable bitstream, as SBs allocation is still optimal. The only difference is the additional headers required for the quality layers. As shown in Table II, it leads to a very little and practically negligible performance degradation. Here, we tested the SNR scalability by encoding non-scalably the "foreman" sequence at 10 different rates with our encoder. Then, we compared the resulting RD performances with that obtained by encoding scalably the input sequence and decoding it at different rates. The performance degradation is less than 0.1dB.

Another cost factor of the scalability is the complexity increase. In this case, we must run N times the allocation algorithm instead of only once. However, its complexity is much lower than the complexity of ME and WT, and thus can be safely neglected.

B. Temporal scalability

As our codec produces temporal SBs by WT, it is straightforward to obtain a temporal subsampled version of compressed sequence from the encoded bitstream: it suffices to simply decode the lower temporal SBs only. However, when a generic temporal filter is used, reconstructing only lower temporal SBs is equivalent to reconstructing a subsampled and *filtered* version of input sequence. This temporal filtering causes ghosting artifacts which can be very annoying to the final user. On the contrary, when $(N, 0)$ filters are employed, the temporal low pass filtering is indeed a pure subsampling. Thus, reversing the WT of a sequence by taking into account only lower temporal subbands is equivalent to reversing the WT of its temporal subsampled version. Moreover, the optimal rate allocation among SBs is preserved through the temporal subsampling: recalling the optimality condition (equation 1), we note that discarding a SB means only decreasing N, but the optimality condition still holds for surviving SBs.

The only problem to deal with is the following: if we simply discard higher temporal SBs, we loose control on the final total bitrate. The solution is once again to run the allocation algorithm only for the desired number of temporal SBs, with the suitable target rate. This will generate a new set of quality layers. A simple signaling convention can be established for the decoder to choose correctly the quality layers accordingly to the desired level of temporal (and possibly quality) scalability. We point out that MV can be easily organized in different streams for each temporal scalability layer, as they are encoded separately accordingly to the temporal decomposition level.

We remark that, in this case as well, the complexity increase is only due to the fact that now we need to run several more times the allocation algorithm. But, as mentioned before, its computational cost is negligible with respect to other part of encoder.

A second set of experiments was performed in order to assess the cost of the temporal scalability. We encoded the sequence at full frame-rate, and we decoded it at half the frame rate. Then we compared the results with those obtained by encoding the temporal subsampled sequence (Table III). In this case, we have a small scalability cost as well (less than 0.2 dB), as expected from theoretical consideration. This difference is ascribable to quality layers overhead. It is worth noting that if other filters than $(N, 0)$ had been used, a much greater performance cost would have been observed, due to the temporal filtering. This objective quality impairment would correspond to annoying subjective effects such as shadowing and ghosting artifacts.

C. Spatial scalability

Subband coding provides an easy way to obtain spatial scalability as well. Indeed, it is sufficient, once again, to discard high frequency SBs, in this case *spatial* frequencies. However, as far as the scalability cost is concerned, spatial scalability is more difficult to handle. The first problem is how to choose an "original" reduced-resolution set of data. We could act as we did for temporal scalability, and consider a spatially subsampled version of input data, but this choice would hardly account faithfully for subjective quality of reconstructed data. Indeed, this "original" sequence would be characterized by annoying spatial aliasing, and thus it would be quite an irrelevant reference for a performance test. We can use the corresponding QCIF sequence as a reference for a CIF input video, but a more general solution is needed. A filtered and subsampled version of input data can be considered, but, in this case, performances would become dependent from the spatial low-pass filter we choose. In our codec, the low-pass spatial analysis filter is the classical 9-taps Daubechies' filter [10], which produces a low resolution sequence whose visual aspect is pleasantly smooth.

Thus, fairly assessing the spatial scalability cost is not straightforward. Nevertheless, once a reference "original" data set is established, our algorithm allows theoretically to adapt to it, by allocating resources between spatio-temporal SBs in an optimal way. However, as we actually encode a *filtered* version of reduced resolution input sequence, we cannot obtain as good performances as if we would encode directly the subsampled version of input data.

We run similar experiments than those presented for temporal and quality scalability. We decoded the sequence at an inferior resolution, and compared the resulting performances with those obtained by directly encoding the reduced-resolution sequence. In this case, as we expected, objective scalability cost is quite large (up to 2dB), but the subjective quality is comparable.

Note that we used the same motion vectors for the half-resolution sequence than for the original one. We simply divided their values by two as well as the blocks size. This motion representation is not scalable.

IV. Experimental Results and Conclusions

The proposed algorithm was tested on the sequences "Garden & Flowers" and "Foreman" (figure 2). The proposed coder obtains performances near to or better than H.264 and, meanwhile, it has a very good scalability.

In conclusion, we presented in this paper a new, fully scalable and fully JPEG2000 compatible video coder, with a new and efficient technique for MV encoding. The scalability does not affect the RD performances or the computational complexity. Besides these valuable properties, our coder also has very promising performances, comparable to or sometimes better than a state-of-the-art coder as H.264.

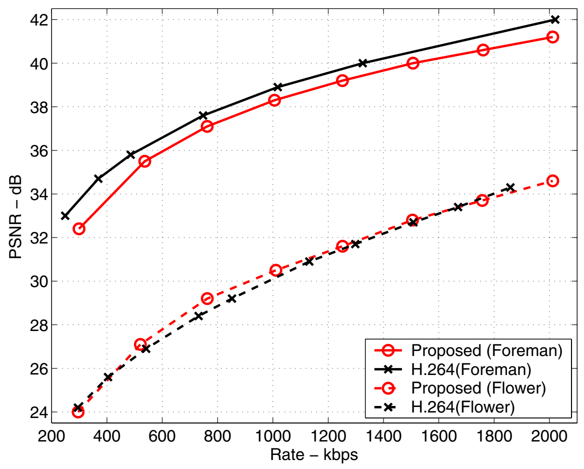

Fig. 2. Coding Performances on the first 64 images of the sequences "Garden & Flowers" and "Foreman"

Future work will deeper investigate trade-offs in lossy MVs coding and optimal bitrate allocation between motion information and subband coefficients. We also plan to address the problem of scalable motion representation.

References

[1] M. Cagnazzo, T. André, M. Antonini, and M. Barlaud, "A model-based motion compensated video coder with JPEG2000 compatibility," in *Proc. of International Conference on Image Processing*, Singapore, Oct. 2004.

[2] ISO/IEC JTC1, *ISO/IEC 14496-2: Coding of audio-visual objects.*, April 1999.

[3] Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, *Joint Committee Draft, JVT-C167*, May 2002.

[4] D. Taubman and M.W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, 2002.

[5] T. André, M. Cagnazzo, M. Antonini, M. Barlaud, N. Božinović, and J. Konrad, "(N,0) motion-compensated lifting-based wavelet transform," in *Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Processing*, Montreal, Canada, May 2004.

[6] A. Secker and D. Taubman, "Motion-compensated highly scalable video compression using an adaptive 3D wavelet transform based on lifting," in *Proc. of IEEE Intern. Conf. on Image Processing*, Greece, Oct. 2001, pp. 1029–1032.

[7] B. Pesquet-Popescu and V. Bottreau, "Three-dimensional lifting schemes for motion compensated video compression," in *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing*, 2001, pp. 1793–1796.

[8] J. Viéron, C. Guillemot, and S. Pateux, "Motion compensated 2D+t wavelet analysis for low rate fgs video compression," in *Proc. of Tyrrhenian Intern. Workshop on Digital Communications*, Capri, Italy, Sept. 2002.

[9] C. Parisot, M. Antonini, and M. Barlaud, "3D scan based wavelet transform and quality control for video coding," *EURASIP Journal on Applied Signal Processing*, Jan. 2003.

[10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transforms," *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992.