Invited Paper

Introducing differential motion estimation into hybrid video coders

M. Cagnazzo, B. Pesquet-Popescu

Institut Telecom, Telecom Paris-Tech, Paris, FRANCE

ABSTRACT

Differential motion estimation produces dense motion vector fields which are far too demanding in terms of coding rate in order to be used in video coding. However, a pel-recursive technique like that introduced by Cafforio and Rocca can be modified in order to work using only the information available at the decoder side. This allows to improve the motion vectors produced in the classical predictive modes of H.264.

In this paper we describe the modification needed in order to introduce a differential motion estimation method into the H.264 codec. Experimental results will validate a coding mode, opening new perspectives in using differential-based motion estimation techniques into classical hybrid codecs.

Keywords: Motion estimation, pel-recursive, H.264

1. INTRODUCTION

The most common algorithms for motion estimation (ME) in the context of video compression are based on the matching of blocks of pixels (block matching algorithms, BMA). However, this is not the only solution: gradient and pel-recursive (PR) techniques, have been developed for video analysis and they solve the optical flow problem using a differential approach.[1] These methods produce a dense motion vector field (MVF), which does not fit into the classical video coding paradigm, since it would demand an extremely high coding rate. On the contrary, it is quite well suited to the distributed video coding (DVC) paradigm, where the dense MVF is estimated only at the decoder side, and it has been proved that the pel-recursive method for motion estimation introduced by Cafforio and Rocca[2–4] improve the estimation quality of missing frames (Wyner-Ziv frames in the context of DVC).[5,6]

Starting from this observation, we have developed a way to introduce the Cafforio-Rocca algorithm (CRA) within the H.264 encoder, in order to provide an alternative coding mode for Inter macroblocks (MB). In particular we used the JM V.11 KTA1.9 implementation. The new mode is coded exactly as a classical Inter-mode MB, but the decoder is able to use a modified version of the CRA and then to compute a motion vector per each pixel of the MB. The motion-compensated prediction is then more accurate, and there is room for RD-performance improvement.

This paper is organized as follows. In section 2 we recall briefly the original method proposed by Cafforio and Rocca. In section 3 we expose our proposal for the introduction of the CRA into an hybrid coder such as H.264. Moreover we have introduced some relevant changes to the algorithm, in order to better adapt it to the hybrid coder paradigm. We give details about all the problems and the design choice we have been faced with. Experimental results are reported in section 4, and finally section 5 draws conclusions and highlights possible future developments of this work.

Further author information: (Send correspondence to M.C.)
M.C.: E-mail: cagnazzo@telecom-paristech.fr, Telephone: +33 1 45 81 79 63

Visual Communications and Image Processing 2010, edited by Pascal Frossard, Houqiang Li, Feng Wu, Bernd Girod, Shipeng Li, Guo Wei, Proc. of SPIE Vol. 7744, 774415 · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.863825

2. CAFFORIO-ROCCA MOTION ESTIMATION

The CR algorithm is a dense pel-recursive motion estimation algorithm: this means that it produces a motion vector (MV) for each pixel, and that previously computed vectors can be used for the initialization of the next pixel to be processed. When applying the original CRA, we suppose that we have the current image, indicated as I_k, and a reference one, indicated as I_h. This reference can be the previous ($h = k - 1$) or any other image in the sequence.

More precisely, once a proper scanning order has been defined, the CRA consists in applying for each pixel $\mathbf{p} = (n, m)$ three steps, producing the output vector $\widehat{\mathbf{v}}(\mathbf{p})$.

1. **A priori estimation.** The motion vector is initialized with a function of the vectors which have been computed for the previous pixels. For example, one can use the average of neighboring vectors. However, a different initialization is needed for the first pixel: it can be for example a motion vector computed with a block matching algorithm (BMA). The result of this step is called *a priori* vector, and it is indicated as $\mathbf{v}^0$.

2. **Validation.** The *a priori* vector is compared to the null vector. In particular, we compute

$$A = \left| I_k(\mathbf{p}) - I_h(\mathbf{p} + \mathbf{v}^0) \right|$$
$$B = |I_k(\mathbf{p}) - I_h(\mathbf{p})| + \gamma$$

   If the prediction error for the current pixel is less than the one for the null vector (possibly incremented by a a positive quantity γ), – that is, if $A < B$ – the a priori is retained as validated vector: $\mathbf{v}^1 = \mathbf{v}^0$; otherwise, the null vector is retained, that is $\mathbf{v}^1 = \mathbf{0}$.

3. **Refinement.** The vector retained from the validation step, $\mathbf{v}^1$, is refined by adding to it a correction $\delta\mathbf{v}$. This correction is obtained by minimizing the energy of first-order approximate prediction error, under a constraint on the norm of the correction. A few calculations show that this correction is given by:

$$\delta\mathbf{v}(n, m) = \frac{-e_{n,m}}{\lambda + \|\phi_{n,m}\|^2} \phi_{n,m} \tag{1}$$

   where λ is the Lagrangian parameter of the constrained problem; $e_{n,m}$ is the prediction error associated to the MV $\mathbf{v}^1$, and ϕ is the spatial gradient of the reference image motion-compensated with $\mathbf{v}^1$.

In conclusion, for each pixel $\mathbf{p}$, the output vector is $\widehat{\mathbf{v}}(\mathbf{p}) = \mathbf{v}^1 + \delta\mathbf{v}$.

3. USING THE CAFFORIO-ROCCA ALGORITHM IN H.264

The basic idea is to use the CRA to refine the MV produced by the classical BMA into an H.264 coder. This should be done by using only data available at the decoder as well, so that no new information has to be sent, apart from some signalling bits to indicate that this new coding mode is used. In other words, the new mode (denoted as CR mode) is encoded exactly like a standard Inter mode, but with a flag telling the decoder that the CRA should be used to decode the data. The operation of the new mode is the following.

At the encoder side, first a classical Inter coding for the given partition is performed, be it for example a 16×16 partition. The encoded information (motion vector and quantized transform coefficient of the residual) is decoded as a classical Inter16×16 MB, and the corresponding cost function $J_{\text{Inter}} = D_{\text{Inter}} + \lambda_{\text{Inter}} R$ is evaluated. Then, *the same encoded information* (namely the same residual) is decoded using the CR mode. The details about the CR mode decoding are provided later on; for the moment we remark that we need to compute the cost function J_{CR} associated to the mode, and that when all the allowed coding modes have been tested, the encoder chooses the one with the smallest cost function. In particular, if the CR mode is chosen, the sent information is the same it would transmit for the Inter mode, but with a flag signalling the decoder to used the CRA for decoding.

Now we explain how the information in a Inter MB can be decoded using the CRA. When we receive the Inter MB, we can decode the motion vector and the residual, and we can compute a (quantized) version of the current MB. Moreover, in the codec frame buffer, we have a quantized version of the reference image. We use this information (the Inter MV, the decoded current MB and the decoded reference image) to perform the modified CRA. This is done as follows*:

If the current pixel is the first one in the scan order, we use the Inter motion vector, $\mathbf{v}^0(\mathbf{p}) = \mathbf{v}_{\text{Inter}}(\mathbf{p})$. Otherwise we can initialize the vector using a function of the already computed neighboring vectors, that is $\mathbf{v}^0(\mathbf{p}) = f\left(\{\widehat{\mathbf{v}}(\mathbf{q})\}_{\mathbf{q}\in N(\mathbf{p})}\right)$. We could also initialize all the pixels of the block with the Inter vector, but this results less efficient.

We compare three prediction errors and we choose the vector associated to the best one. This is another change with respect to the original CRA, where only 2 vectors are compared. First, we dispose of the quantized version of the motion compensated error obtained by first step. Second, we compute the error associated to a prediction with the null vector. This prediction can be computed since we dispose of the (quantized) reference frame, and of the (quantized) current block, decoded using the Inter16 mode. This quantity is possibly incremented by a positive quantity γ, in order to avoid unnecessary reset of the motion vector. Finally, we can use again the Inter vector. This is useless only for the first pixel, but can turn out important if two objects are present in the block. In conclusion, the validated vector is one among $\mathbf{v}^0(\mathbf{p})$, 0 and $\mathbf{v}_{\text{Inter}}(\mathbf{p})$; we keep as validated vector the one associated to the least error.

The refinement formula in Eq. (1) can be used with the following modification. The error $e_{n,m}$ is the quantized MCed error; the gradient ϕ is computed on the motion-compensated reference image.

Now we discuss the modification made on the CRA, and how the impact on the algorithm's effectiveness. First, we dispose only of the quantized version of the motion-compensation error, not of the actual one. This affects both the refinement and the validation steps. Moreover, this makes impossible to use the algorithm for the Skip mode, which is almost equivalent to code the MC error with zero bits. The second problem is that we can compute the gradient only on the decoded reference image. This affects the refinement step. These remarks suggest that the CRA should be used carefully when the quantization is heavy. Finally, we observe that the residual decoded in the CR mode, is the one computed with the single motion vector computed by the BMA. On the other hand, the CRA provides other, possible improved vectors, which are not perfectly fit with the sent residual. However, the improvement in vector accurateness leaves room for possible performance gain, as shown in the next section.

Because of the complexity of the H.264 encoder, we have take into account some other specific problems, such as the management of CBP bits, the MB partition, the multiple frame reference, the mode competition and the bitstream format. Most of the solutions adopted are quite straightforward, so we do not described them here for the sake of brevity. However, some other issues were not considered yet in our first implementation, namely the management of B-frames and of color information.

4. EXPERIMENTAL RESULTS

We considered several video sequences with different motion content, (from low motion content to irregular, high motion sequences) and we performed two kind of experiments on them. In a first set of experiments, described in section 4.1, we only considered the effectiveness of CR motion vector improvement in the framework of the H.264 codec. In the second set of experiments, described in section 4.2, we considered the introduction of the CRA within the H.264 codec, and we evaluated the RD performances with respect to the variations of all relevant parameters.

4.1 Tests on Motion Estimation

In this section we consider tests made with the goal of validating the CR ME. We will not consider the impact over the global RD performances of the encoder.

*We should first define a scanning order of pixels; we have considered a raster scan order, a forward-backward scan, and a spiral scan.

Method		MSE
H.264 Vectors		54.64
CR MSE	$\lambda = 10$	95.48
CR MSE	$\lambda = 100$	60.23
CR MSE	$\lambda = 1000$	53.22
CR MSE	$\lambda = 10000$	53.06
CR MSE	$\lambda = 100000$	53.17
CR MSE	$\lambda = \infty$	53.17
H.264 Coding MSE		6.08

Table 1. Prediction error energy, "foreman" sequence.

4.1.1 Comparison with a Block Matching Algorithm

In a first set of experiments we compared the CR MVs with a MVF obtained by classical block-matching algorithms. We considered several input video sequences, and for each of them several pairs of current-reference frames (not necessarily consecutive images). For each pair of current-reference images, we computed the prediction error energy with respect to the full-search BMA MVF, and the prediction error energy in the case of CR ME. In this case we use the original (*i.e.* not quantized images), and the computation has been repeated for several values of the Lagrangian parameter λ.

The results are shown in Fig. 1. For all the test sequences we find a similar behavior: but for very small values of λ, the CR vectors guarantee a better prediction with respect of the BMA (green dashed line). For increasing values of λ the MSE decreases quickly, reaches a minimum and then increases very slowly towards a limit value, corresponding to $\lambda = \infty$. The latter case corresponds to a null refinement: all the improvement is due to the validation test, and corresponds with difference between the green and blue dashed lines in Fig. 1. The minimum value of the MSE is obtained when besides the validation test, the vectors are modified by the refinement step. We also remark that the CRA is quite robust wrt the choice of λ; the difference between the minimum and the asymptotical value of the black curve is the contribution of the refinement step.

4.1.2 Motion estimation with quantized images

The first experiment shows us that the CRA has the potential to improve the BMA motion vectors. However comparing only the prediction MSE is not fair since the cost of encoding the vectors is not taken into account. Of course, in this case, it would be extremely costly to encode the CR vectors, since there is a vector per pixel. The RD comparison would be definitely favorable to the classical ME algorithm. This is however why the CRA is not used in hybrid video coders at least in its classical form. On the contrary, the in the modified CRA the cost of MV coding is zero, but this is obtained by sacrificing the accuracy of validation and refinement, which must be performed using quantized data.

We designed the a second kind of tests in order to evaluate the potential gains of the CRA in the framework of a H.264-like coder. Unlike the previous case, we did not used the original images to perform the CRA, but we used those available to a H.264 decoder. More precisely, we first used H.264 to produce: the MVF between two images in a video sequence (indicated by $\mathbf{v}$, and the decoded current and reference image, indicated as $\hat{I}_k$ and $\hat{I}_h$ (*e.g.* $h = k - 1$).

Then the following quantities were computed:

H.264 Vectors MSE : The prediction error mean squared value for the H.264 vectors.

$\mathbf{v}_{\mathrm{CR}}$: The CR motion vectors, obtained by using $\hat{I}_k$ and $\hat{I}_h$ and $\mathbf{v}$.

CR MSE : The prediction error mean squared value for the CR vectors.

Some results are summarized in table 1 and 2. We performed the same test over many other sequences, and we obtained similar results.

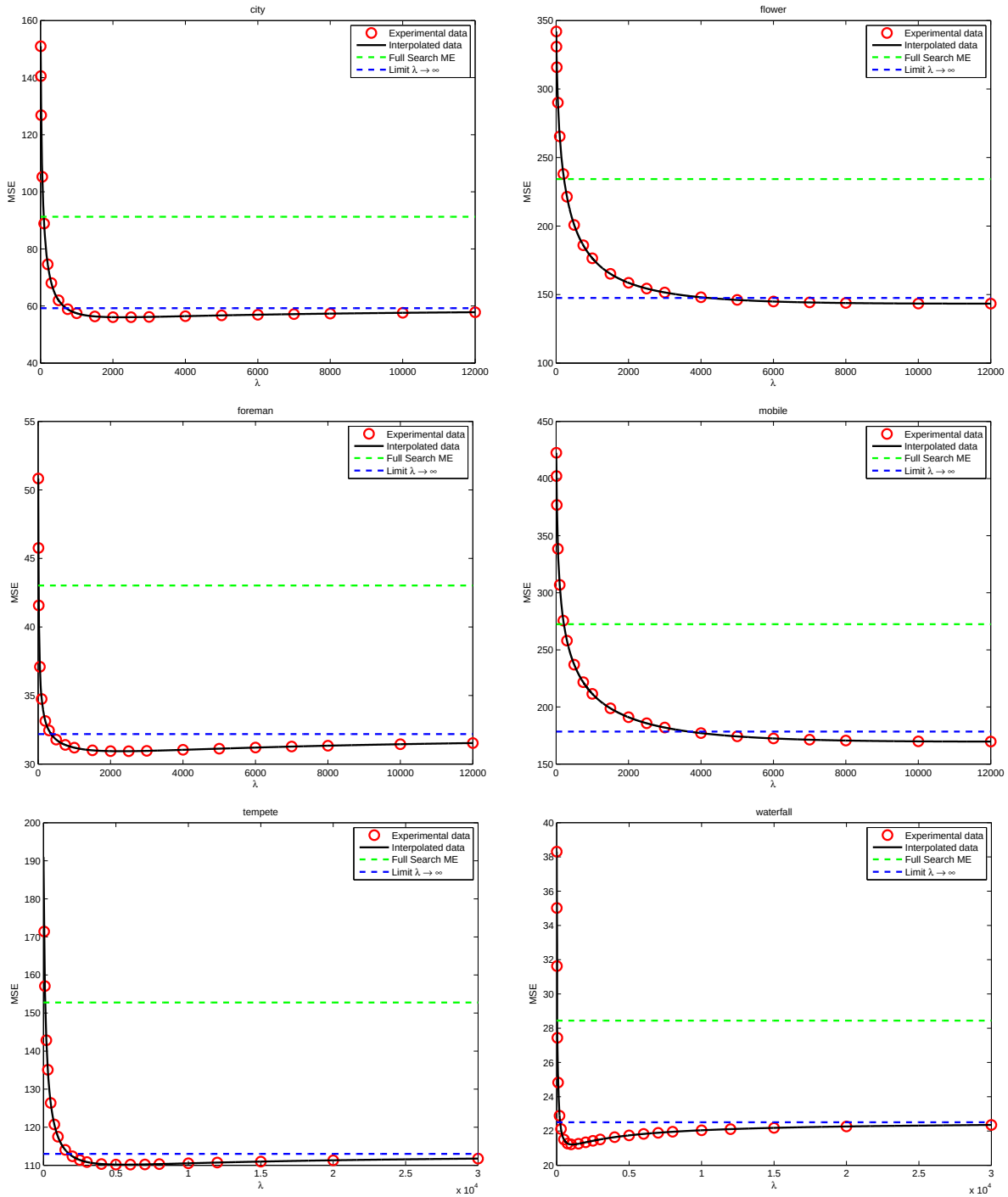

Figure 1. Motion Estimation performances of the CRA algorithm, different sequences

We observe that in this case as well, the CR method is able to potentially improve the performance: the prediction produced with the CR vectors has (but for small λ values) a smaller error than the original H.264 vectors, even if the CRA is run over quantized images. Moreover in this case the comparison is fair in the sense that the CR vectors do not cost any bit more than the original ones.

The last line of the table reports the MSE of the decoded macroblock in H.264. Of course it is much smaller

Method	MSE
H.264 vectors	90.79
CR $\lambda = 10$	571.02
CR $\lambda = 100$	237.34
CR $\lambda = 1000$	92.09
CR $\lambda = 10000$	72.86
CR $\lambda = 100000$	71.68
CR $\lambda = \infty$	71.69
H.264 MSE	3.49

Table 2. Prediction error energy, "mobile" sequence.

	Δ PSNR	Rate reduction
bus	0.01	-0.02%
city	0.01	-0.11%
coastguard	0.01	-0.07%
flower	0.04	-0.76%
football	0.02	-0.35%
mobile	0.02	-0.42%
paris	0.02	-0.55%
tempete	0.01	-0.13%

Table 3. Rate distortion performances improvement when introducing the new coding mode into H.264.

of the prediction error energy, since it benefits from the encoded residual information. As a conclusion, it is critical to keep an efficient residual coding, since it is responsible for a large amount of the distortion reduction.

4.2 RD performances

In this section we comment about the performance of the modified H.264 coder. In order to assess the effectiveness of the proposed method, we have implemented it within the JM H.264 codec. We considered several design choice: the effects of the Lagrangian parameter, of the threshold and of the initialization method. Finally we compare the RD comparison to the original H.264 coder.

The proposed method seems not to be too affected by the value of the threshold γ provided that it is not too small (usually $\gamma > 10$ works well). For the sake of brevity we do not report here all the experimental results. Likewise, we found that setting $\lambda = 10^4$ works fairly well in all the test sequences. In the following, these values of the parameters are kept.

A larger impact on the performance is due to the validation step. Introducing a third candidate vector for validation allows a fast motion vector recover when passing from one object to another within the same block.

Finally we compared an H.264 coder where the new coding mode was implemented with the original one. The results are shown in table 3, where we report, for each test sequence, the improvement in PSNR and the bit-rate reduction computed with the Bjöntegard metric[7] over four QP values. We observe that the improvement are quite small, in part because the CR mode is rarely selected rarely (usually only for 10% of the blocks).

It is worth noting that the coding time with the modified coder are very close to those of the original one: we usually observed an increase less than 2%.

5. CONCLUSIONS AND FUTURE WORK

Differential motion estimation is very effective but the resulting motion vector fields are too demanding (in terms of coding rate) to be profitably used in video coding. However we show that a modified version of the Caffario-Rocca algorithm has the potential to improve the performance of the H.264 coder, since we have designed a method that resets to zero the additional coding rate needed for the dense motion vector field.

The experimental results are encouraging as far as the motion estimation part is concerned. The integration into the H.264 encoder shows for the moment smaller gains, but studies are under way in order to improve this

part. In particular, we intend to implement a few methods that proved to be effective in the context of distributed video coding. Among them, introducing some constraint on the regularity of the motion vector field (such that discontinuity are only allowed across object borders) seems quite promising. This has been implemented in DVC[6] using the Nagel-Enkelmann constraint.[8] Another possible improvement lies in further coding the actual residual associated to the CR mode, instead of simply using the one of the Inter mode.

REFERENCES

1. B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.
2. C. Cafforio and F. Rocca, "The differential method for motion estimation," in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang, Ed., 1983, pp. 104–124.
3. ——, "Methods for measuring small displacements of television images," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 5, pp. 573–579, Sep. 1976.
4. C. Cafforio, F. Rocca, and S. Tubaro, "Motion compensated image interpolation," *IEEE Trans. Commun.*, vol. 38, no. 2, pp. 215–222, Feb. 1990.
5. M. Cagnazzo, T. Maugey, and B. Pesquet-Popescu, "A differential motion estimation method for image interpolation in distributed video coding," in *Proceed. of IEEE Intern. Conf. Acoust., Speech and Sign. Proc.*, vol. 1, Taiwan, 2009, pp. 1861–1864.
6. M. Cagnazzo, W. Miled, T. Maugey, and B. Pesquet-Popescu, "Image interpolation with edge-preserving differential motion refinement," in *Proceed. of IEEE Intern. Conf. Image Proc.*, Cairo, Egypt, 2009.
7. G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in *VCEG Meeting*, Austin, USA, Apr. 2001.
8. H. Nagel and W. Enkelmann, "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 5, pp. 565–593, Sep. 1986.