UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

LOW COMPLEXITY CONTEXT-BASED MOTION COMPENSATION FOR VLBR VIDEO ENCODING

F. G. B. De Natale and F. Granelli

January 2002

Technical Report # DIT-02-0003

Low-Complexity Context-Based Motion Compensation for VLBR Video Encoding

Francesco G.B. De Natale, *Member, IEEE*, and Fabrizio Granelli, *Member, IEEE*

Abstract--***A significant improvement of block-based motion estimation strategies is presented, which provides fast computation and very low bitrate coding. For each block, a spatio-temporal context is defined based on nearest neighbors in the current and previous frames, and a prediction list is built. Then, the best matching vector within the list is chosen as an estimation of the block motion. Since coder and decoder are synchronous, only the index of the selected vector is needed at the decoder to reconstruct the motion field. To avoid the propagation of the error, an additional correction vector can be sent when prediction error exceeds a threshold. Furthermore, bitrate saving is achieved through an adaptive sorting of the prediction list of each block, which allows to reduce the entropy of the motion indexes. Tests demonstrate that the proposed method ensures a speed up over 1:200 as compared to full search, and a coding gain above 2, with a negligible loss of accuracy. This allows real-time implementation of VLBR software video coders on conventional PC platforms.***

Index Terms—Video coding, Motion estimation

I. INTRODUCTION

Block-based motion estimation (BME) is widely used in video coders due to the good trade-off between reconstruction quality and bitrate reduction. It defines a sparse array of motion vectors (*motion field*) associated to a regular block partitioning of the frame. Each vector is estimated by minimizing a local distance measure called Displaced Frame Difference (DFD), which is usually computed as the pixel-by-pixel difference (absolute, square, …) between the target block and the displaced blocks in a reference frame. In full-search (FS) approaches, all possible displacement vectors are considered within a predefined search window: this procedure provides the best possible match, but implies a huge computational load, inappropriate for real-time implementations.

Several methods have been proposed in the literature to achieve an estimation performance close to full-search BME with a lower computation. A first class of techniques limits the number of DFD measures by choosing a reduced subset of positions within the search window: these methods are referred to as *fast search methods* [1, 2], and are usually based on descent algorithms applied to the DFD function. A second class of approaches is based on the reduction of the operations to be performed for each DFD: in [3, 4] the use of decimation strategies is proposed, combined with either full- or fast-search methods; in [5, 6] the use of less expensive error measures is suggested.

Finally, adaptive methods represent a third class. In hierarchical methods, an initial estimation is performed at a coarse scale and progressively refined towards the higher resolution: at each stage only small corrections are introduced [7]. Hierarchical approaches are also used to achieve variable-size BBME [8], where higher computation/bitrate gains can be obtained by computing a non-uniform motion field. Another interesting approach is presented in [9], where motion estimation is performed through a two steps algorithm: first, a rough estimation of the motion vector is provided by considering a reduced subset of the spatial and temporal neighbors, then the search is refined by using a pixel-recursive approach. In [10], a fast motion estimation algorithm employs adaptive search patterns, taking into account matching criteria as well as statistical properties of object displacement. The selection of the appropriate search pattern is performed by exploiting the relationship between the motion vector and the DFD of each block. A reduction of the search time can also be achieved by estimating, for each block, the best starting point for motion estimation through a compensation of the search area based on the temporal and spatial correlation of the motion field [11].

A new class of techniques that is gaining interest is based on the exploitation of motion field redundancy to achieve effortless vector prediction. Two interesting proposals in this sense are provided in [12], where local correlation is introduced in a multiresolution BME, and in [13], where the spatio-temporal redundancy among motion vectors is used to achieve a more compact description of the field for VLBR video coding.

In the present work we extend and improve the concept of vector prediction in order to attain a real-time BME, which efficiently exploits the local spatio-temporal correlation of the motion field. For each block of the frame to be compensated, the algorithm generates a prediction list containing the candidate motion vectors. A properly defined spatio-temporal context (the list of candidate vectors) is used for this purpose. A reduced number of DFD measures is then required to select the vector within the prediction list that provides the better compensation (target vector). To avoid error propagation, it is possible to refine the estimated motion vector when the compensation error exceeds a pre-defined value. In this case, a reduced search window is used to ensure a low computational effort. Furthermore, in order to decrease the number of bits required to address the prediction list, a sorting strategy was developed that ensures a sharp reduction of the code entropy.

The structure of the paper is as follows: in Section 2 the proposed method is explained in more detail, while in Section 3 the experimental results are described and in Section 4 the conclusions are drawn.

II. CONTEXT-BASED BLOCK MOTION ESTIMATION AND CODING

In Fig. 1, a complete scheme of the proposed technique (CB-BME) is shown.

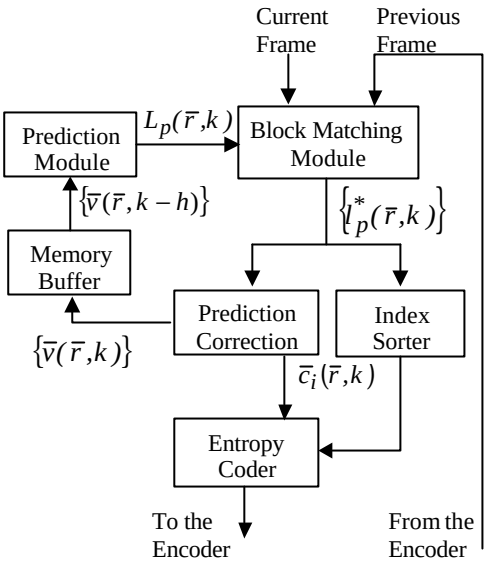


Figure 1. Complete scheme of CB-BME technique

The core of the system is the prediction module. It applies spatio-temporal prediction rules to generate a prediction list, which contains a set of candidate vectors for the estimation of the motion field at time k (starting from a reference field at time k-h). The prediction list for each frame block is then sent to the block matching engine, which selects the displacement vector that minimizes the DFD. The selected vector is sent to the prediction correction module in order to correct errors above a given threshold.

The information to be transmitted includes two items: the index associated to the selected vector within the prediction list, and the correction vector (if necessary). In order to efficiently encode the vector indexes, a further module is introduced in the scheme (called Index Sorter) that enhances the performance of the following entropy coder.

The following paragraphs describe the different modules in detail, introducing also a brief analysis of the computational complexity of the technique.

A. Spatio-temporal prediction

The idea of using temporal correlation among successive frames to predict the motion field was proposed in the past in the framework of *autocompensation* techniques [14, 15]. These methods hypothesize a motion invariance (a sort of inertia) between consecutive frames, which allows to extrapolate the motion field at time k from a previous reference motion field estimated at time $(k$-$h)$. Each vector is estimated by summing the adjacent vectors of the previous frame with appropriate weights. This prediction can be considered reliable if the sequence is characterized by slow and continuous motion. On the contrary, in the presence of strong temporal discontinuities it turns out to be very noisy and subject to error propagation. In [14], a generalization of the autocompensation technique is proposed, aimed at improving the prediction by adding some spatio-temporal information (called *autocompensation-with-parameter*). The BME algorithm presented in this paper (Context-Based BME, or CB-BME) is a significant evolution of the above technique, targeted to the efficient and low-cost spatio-temporal prediction of the motion field.

Figure 2 presents a conceptual scheme of the prediction scheme.

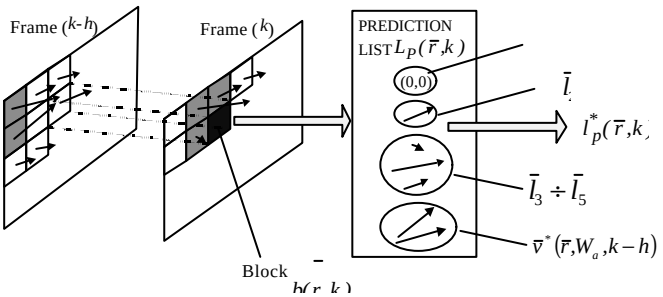


Figure 2. Conceptual scheme of CB-BME

For each $B \times B$ block $b(\bar{r},k)$ of the frame to be compensated ($\bar{r}$ = block spatial position, (k) = discrete temporal instant), a prediction list $L_p(\bar{r},k)$ of dimension $n_p(\bar{r},k)$ is built, containing a set of motion vectors belonging to the

spatio-temporal context of $b(\bar{r},k)$. A prediction vector $l_p^*(\bar{r},k)$ is then selected among the candidate vectors contained in the list: to this purpose, the compensation error $e_p^i(\bar{r},k)$ is computed for each displaced block according to the prediction list, and the displacement that provides the minimum error $e_p^{min}(\bar{r},k)$ is chosen. A particular attention is paid to the construction of the prediction list $L_p(\bar{r},k)$; the following elements are always included:

- the null element $\bar{l}_1 = (0,0)$
- the motion vector $\bar{l}_2 = \bar{v}(\bar{r},k-h)$ of the corresponding block in the reference frame;
- the motion vectors of the blocks adjacent to $b(\bar{r},k)$ in the current frame, already calculated in the raster scan processing of the frame, i.e., $\bar{l}_3 = \bar{v}(\bar{r}-(1,1),k)$, $\bar{l}_4 = \bar{v}(\bar{r}-(0,1),k)$ and $\bar{l}_5 = \bar{v}(\bar{r}-(1,0),k)$;
- the motion vectors $\bar{v}^*(\bar{r},W_a,k-h)$, $with\ \xi = 1,\cdots,\Xi(\bar{r},k)$, belonging to the reference frame and falling, after autocompensation, within a window of dimension $W_a \times W_a$, $W_a = 2B+1$, centered on the current block position.

The terms $\bar{l}_3$, $\bar{l}_4$ and $\bar{l}_5$ take into account the spatial correlation of the motion field, while the terms $\bar{l}_1$, $\bar{l}_2$ and those generated by autocompensation exploit the temporal redundancy, thus achieving a quite accurate prediction model.

To ensure that the best among candidate motion vectors is selected for $b(\bar{r},k)$, an exhaustive computation is performed on $L_p(\bar{r},k)$. This process requires $n_p(\bar{r},k)$ DFD measures, where the number of vectors in the list is generally not fixed, as it strongly depends on the local motion activity in the sequence. Only average values can be estimated for a given class of video data. Furthermore, before performing blockmatching, $L_p(\bar{r},k)$ is scanned in order to delete repeated vectors and reduce in this way the length of the list.

The spatio-temporal prediction employed in CB-BME is significantly more accurate than that presented by Kauff et al. in [9], where for each block only three candidate vectors are considered: the two vectors of the spatial neighbors and the vector of the block in the previous frame. An evidence of the advantages of CB-BME is provided in figure 3, where the average peak signal-to-noise ratio of the motion compensated frames is compared between the two prediction approaches (without encoding the compensation error).

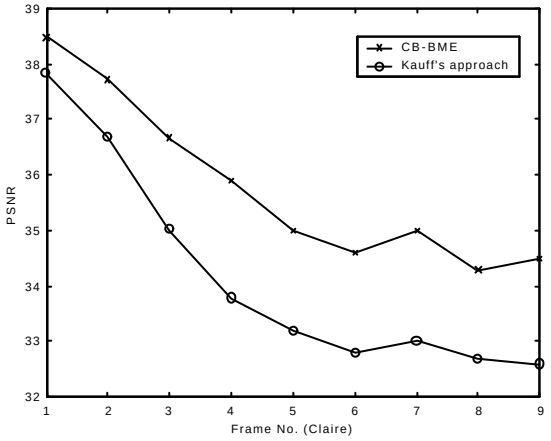


Figure 3. Comparison between CB-BME and Kauff approach in terms of PSNR of the reconstructed frames (Claire video sequence, QCIF format).

Clearly, the spatio-temporal prediction needs to be initialized with a starting motion field, which can be estimated using FS or one of the low-complexity methods presented in the introduction.

In addition, figure 3 shows that the reconstruction quality drops down when the prediction is propagated for a large number of successive frames without error correction. For this reason, the next paragraph introduces a simple approach to avoid error propagation among successive motion compensated frames.

B. Correction of spatio-temporal prediction

The proposed spatio-temporal prediction provides an estimation of the motion field that is usually quite close to the one produced by the full-search approach (in terms of DFD and visual perception). Nevertheless, some corrections are sometimes required, especially in the presence of fast or chaotic motion, to improve the compensation and limit possible error propagation problems.

Furthermore, the statistical distribution of the difference vectors between CB-BME and FS-BME shows a peak on the null vector, and approximately display a central symmetry with respect to that point, with exponential-like behaviour (see fig. 4). This suggested to implement the correction of the spatio-temporal prediction using a BME module using a reduced search window.

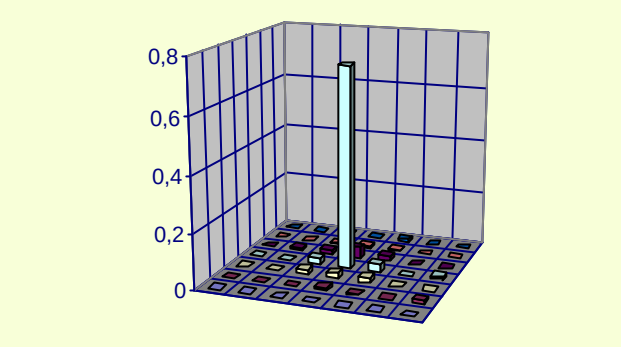


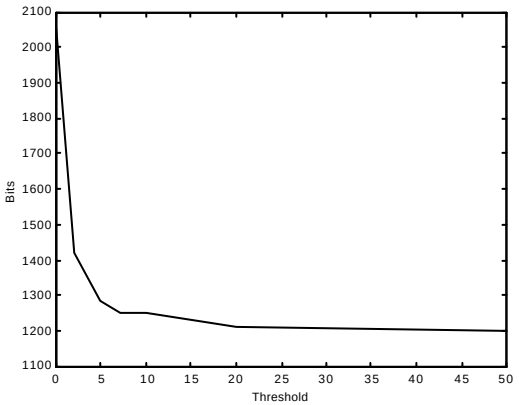
Figure 4. Average distribution of the difference vectors between motion field estimated with FS-BME and spatio-temporal prediction (Flower Garden video sequence, CIF format).

The refinement is activated on the basis of a threshold T_H applied to the DFD measure of the compensated block. For each estimated motion vector $l_p^*(\bar{r},k)$ that produces an error $e_p^{min}(\bar{r},k) > T_H$, a correction vector $\bar{c}_i(\bar{r},k)$ is calculated by inspecting a search window of dimensions 3x3 pixels centered on the prediction vector.
The threshold T_H represents the maximum tolerated error for a block and it is set through empirical evaluations. In the paper, the employed DFD measure is the *Mean-Absolute-Error* (MAE) per pixel and it is compared with the threshold T_H. Typical values of T_H are between 5 and 15.
This parameter allows also to trade off between reconstruction quality and compression ratio, since low values of T_H imply high quality and low compression ratio, while higher values bring lower quality but higher compression ratios. The average behaviour of the system is represented in figure 5.

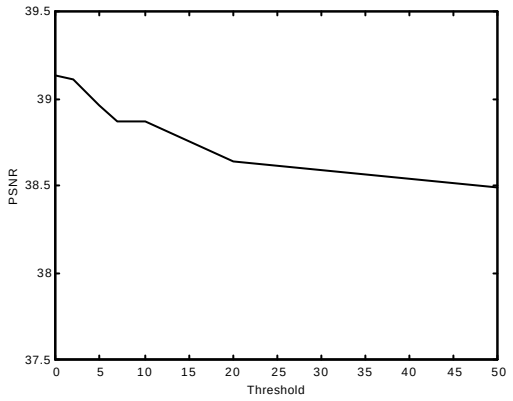
C. Entropy minimization of the motion prediction

In this paragraph, a lossless technique is introduced that allows to reduce the bitrate for the transmission of the motion field using the CB-BME scheme.
Figure 6-a shows the histogram of the indexes transmitted by the motion prediction module, applied to a frame of the *Flower Garden* sequence. Such indexes are used at the decoder to select the correct motion vector within the prediction list. As it can be guessed from the observation of the chart, the statistical distribution of indexes shows a relatively high entropy value (2,64 bits), thus leading to a suboptimal code.
In order to reduce the impact of the prediction on the overall bitrate, for each block a sorting procedure is applied to the vectors belonging to the prediction list. The underlying concept is quite simple: depending on the context, some vectors contained in the prediction list are more likely to be used than others. By associating the most likely vectors to the lower indexes, it is possible to minimize the entropy of the sequence of indexes to be transmitted. This sorting procedure should be of course reproducible at the decoder without requiring any extra information, in order to achieve an effective gain. Different approaches can be set up to this purpose. In our tests we considered four different classes of methods, and we compared them in terms of effectiveness in entropy reduction and complexity:

- the “most-frequently used” principle;
- the “most-recently used” principle;
- filtering of the contextual neighbors;
- advanced predictors.



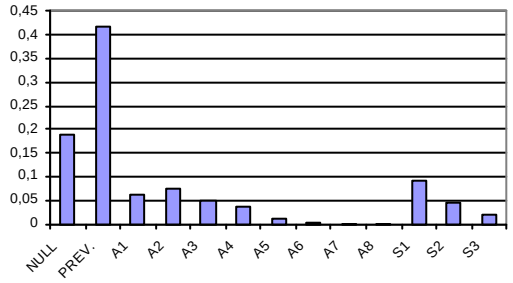
(a)



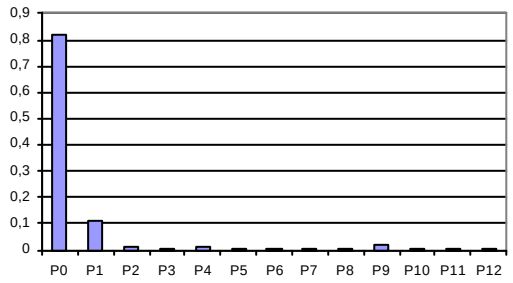
(b)

Figure 5. An example of the behaviour of CB-BME in terms of average number of bits per frame (a) and PSNR (b) versus the threshold T_H (Claire video sequence, QCIF format).

The first two approaches mimic the behaviour of caching systems, where the probability of using a given object is considered in some way proportional to the previous use of the same object in a local context. The most-frequently-used method considers more probable the vector that shows a higher frequency of occurrence over a given area.
The most-recently-used approach is similar to the former one, but includes proportionality to the topological distance of the used vector from the current one. Most recently used motion vectors can be simply computed by using the motion vectors selected for the previous block or macro-block or by estimating the probabilities of occurrence over a rectangular area, properly weighted on the basis of the distance from the considered block. Fig. 7-b shows sample weighting coefficients using city-block distance.



(a)



(b)

Figure 6. Statistical dis tribution of the indexes representing the chosen prediction vectors before (a) and after (b) sorting (Claire video sequence, QCIF format). 'NULL' represents the null vector, 'PREV' is the motion vector of the same block in the previous frame, 'An' are the terms deriving from autocompensation, and 'Sn' the spatial neighbors.

The last two methods are far more complex. The filtering approach consists in applying a (linear or non-linear) filter to the vectors contained in the context region, and sorting the indexes of the prediction list on the basis of the Euclidean distance from the resulting vector. Examples of possible filters are mean, median, low pass, etc. [16]

Real prediction techniques need a dedicated module for the estimation of the best candidate vector to drive the sorting procedure. Inside this class, we can mention neural network predictors, LMSE estimators, etc. Even if such kinds of prediction techniques are far more accurate than the other ones, they are not implemented in this paper because of the computational complexity as well as the relative slow speed of such approaches. However, in cases where real-time and low complexity are not mandatory, neural network or LMSE estimators could probably be very useful in providing a good and faithful prediction of the motion vectors.

In the experimental testing presented in this paper, we preferred the most-frequently-used strategy, for it is characterized by a low computational complexity and a sufficiently good performance. An intuitive way to implement it, is the following:

1. consider a contextual region of a maximum of 5x5 blocks (the shaded blocks in Fig. 7), and compute the usage statistics of the motion vectors associated to the blocks enclosed in such a region;
2. sort the prediction list by ordering the vectors on the basis of the probability of occurrence in the contextual area;
3. perform motion estimation for current block using the sorted prediction list, as described in paragraph 2.1
4. transmit the index of the selected vector in the ordered list, entropy coded.

(a) (b)

Figure 7. The region shaded in grey is the contextual area considered for statistics evaluation (a) and the corresponding city-block distance modifier (b).

Fig. 6-b shows the results of the above algorithm. It can be observed that this approach exploits the notable amount of local correlation between vectors belonging to the same motion field. This is due to the fact that, in a given region of a frame, motion vectors associated to neighboring blocks have similar orientation, since "moving objects" in the scene usually span over several adjacent blocks.

The statistical distribution of the indexes deriving from the use of this technique is characterized by an effective reduction of the entropy (in the example 1.05 bits instead of 2.64), and consequently by a more compact code. Moreover, it does not require any information overhead, since the decoder is able to build the statistics directly from the decoded stream. The only cost of the operation is represented by the extra computation required by the sorting procedure (to be applied at both encoder and decoder), which is almost negligible as explained in the following.

A further problem is how to notify the decoder about the need of sending prediction correction vectors. An efficient solution lies in adding a new index to the prediction list $L_p(\bar{r},k)$ in position $n_p(\bar{r},k)+1$, having the meaning of a *prediction fault*. When the compensation error is higher than the threshold T_H, the prediction fault index is sent, followed by the actual prediction vector $l_p^*(\bar{r},k)$ and the correction vector, in sequence. The correction vector is usually estimated over a 3x3 search window centered on $l_p^*(\bar{r},k)$. No entropy coding is used for the correction vector, since the average statistical distribution of symbols is nearly uniform. An alternative solution after a prediction fault, is to transmit the complete motion vector computed with a fast method. This approach ensures the maximum protection against error propagation, but increases the computational complexity, due to the need of performing a block-based motion estimation over a large window.

D. Computational complexity

The computational load of the motion estimation procedure is not fixed, and mainly depends onto two factors: the level of temporal activity of the sequence and the accuracy of the initial motion prediction. The first parameter influences the dimension of the prediction list $n_p(\bar{r},k)$, while the second affects the number of refinements required, and is proportional to the probability $P\left(e_p^{min}(\bar{r},k) > T_H\right)$ that a given prediction provides an unsatisfactory compensation. Calling this last probability P_{fault}, we can write the mean number of DFD measures per block (Γ) as follows:

$$\Gamma = \bar{n}_p + P_{fault} \cdot N \quad (1)$$

where $\bar{n}_p$ is the mean length of the prediction lists and N is the number of the allowed correction vectors (8 in the case of a refining window of 3x3 pixels).

In Eq. 1, the operations required to build the prediction list are not taken into account, since it can be easily observed that the prediction vectors can be achieved by just addressing the motion field of the reference frame. More complex prediction strategies require of course additional computation.

In addition, for each block of the current frame the sorting procedure of the prediction list requires a number of comparisons Λ which can be evaluated by the well-known formula [17]:

$$\Lambda = \frac{3}{8}\left(\bar{n}_p{}^2 - 1\right) \quad (2)$$

where $\bar{n}_p$ represents again the mean length of the prediction lists.

The computational load deriving from Λ comparisons is in general negligible as compared to the computation of the DFD measures indicated by the parameter Γ.

III. Experimental Results

The CB-BME algorithm was tested on several CIF- and QCIF-format video sequences. In this section, the results are presented for three widely available video clips, namely: *Flower Garden*, *Silent*, and *Mobile and Calendar.* These sequences show different type of activity (camera motion, several objects with different motion) and a wide range of textures and details.

The algorithm demonstrated a good robustness with respect to the only tunable parameter, which is the threshold T_H on the prediction error. As a matter of fact, the results presented in the following charts are achieved with a unique threshold T_H = 5, independently of the characteristics of the video sequence. This corresponded to an average percentage of error-corrected blocks around 25 %. The threshold T_H = 5 was selected as a good trade-off between bitrate reduction and quality preservation: looking back at the charts in Fig. 5, it is evident that for very low values of T_H the bitrate grows exponentially, while for high values of T_H the additional bitrate gain is not sufficient to justify a further quality loss. The block size and the search window were fixed to 8 and ±15 pixels, respectively, for compatibility with the current standards.

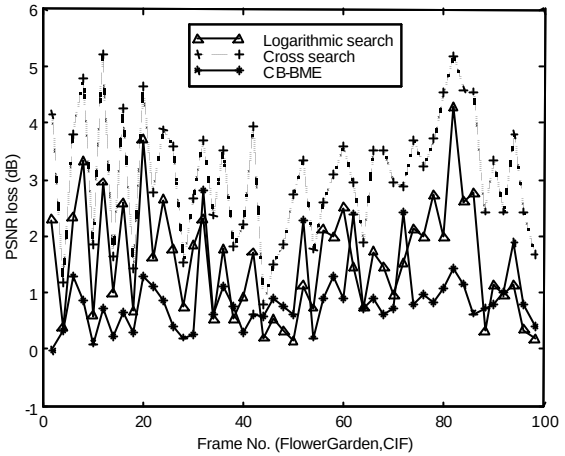
The charts in Fig. 8 compare the PSNR performance achieved by CB-BME with the full-search (FS) algorithm (that provides the best match) and two well-known fast search methods: the two-dimensional logarithmic search (TDLS) and the cross-search (CS). In detail, the graph shows the quality loss, expressed in dB, attained by CB-BME, TDLS and CS with respect to full search, for a set of consecutive frames. It can be observed that CB-BME outperforms the other fast approaches, but also provides a more uniform behavior (i.e., a quite constant PSNR loss, around 0.7 dB).

In Fig. 9 the performance of CB-BME is analyzed in terms of computational efficiency. Again, the complexity of the technique is compared to TDLS and CS in terms of percentage reduction of the computation with respect to FS, over a set of consecutive frames. It is possible to observe that CB-BME requires an almost constant processing, and is around three times faster than CS and five times faster than TDLS. As compared to full-search it provides a speed-up around 140.

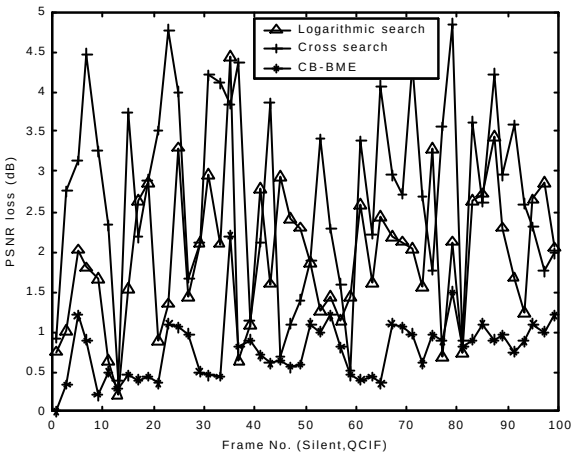
Furthermore, the entropy of the motion information was analyzed, in order to verify the possibility of an efficient encoding of the data stream. The charts in Fig. 10 present the relevant results: in this case, CB-BME was tested with and without the error correction and compared with FS, TDLS and CS. The best performance was obtained by CB-BME without error correction, thus demonstrating the effectiveness of the prediction strategy and the usability of this method for low-bitrate applications. Also CB-BME with correction provided a good result, almost always superior to CS and greatly better than TDLS and FS.

For the sake of completeness, CB-BME was also implemented and tested within a H.263 codec [18]. Table 1 presents the average results in terms of PSNR for different bitrates. It is to be pointed out that, besides the reduction in complexity of the block matching procedure which provides a significant speed-up of the encoder, the proposed method offers a non negligible benefit from the viewpoint of the overall reconstruction quality, thanks to the allocation of the bits saved in the transmission of the motion information to the DFD encoder.

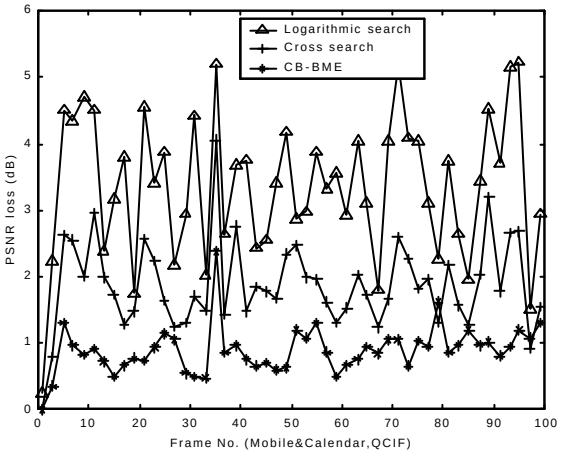
Finally, in Fig. 11 a graph is proposed that summarizes the comparison among the three motion estimation techniques (CB-BME, TDLS, CS). Here the complexity (percentage reduction as compared to FS) and distortion (PSNR loss in dB compared to FS) parameters are averaged and plotted in a single chart for the three methods and for the three test sequences, thus providing a very intuitive overview of the relative performance.



(a)

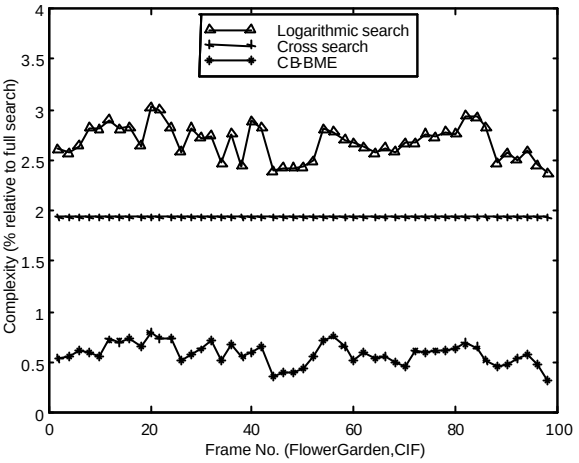


(b)

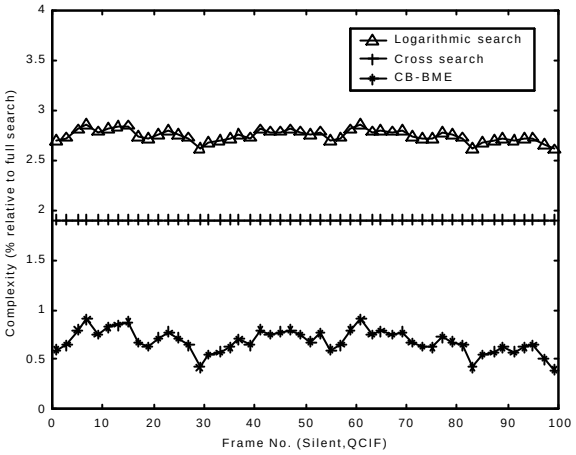


(c)

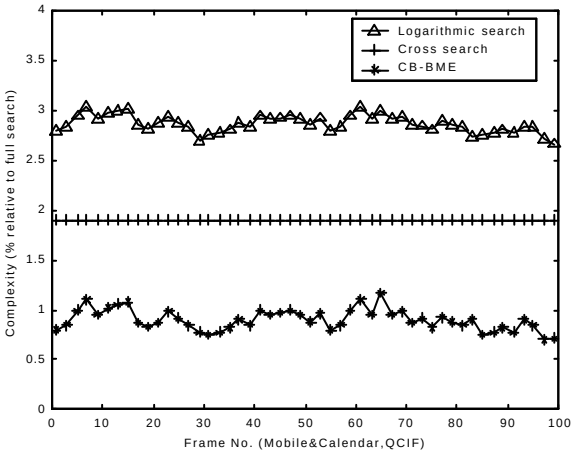
Figure 8. Distortion expressed in loss versus full-search for three test sequences : (a) Flower Garden, (b) Silent, and (c) Mobile&Calendar



(a)

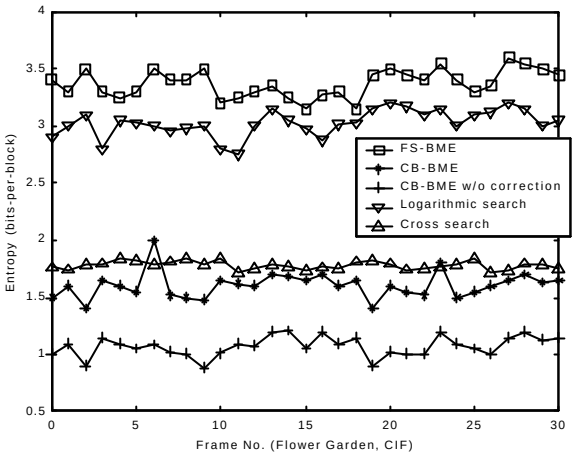


(b)

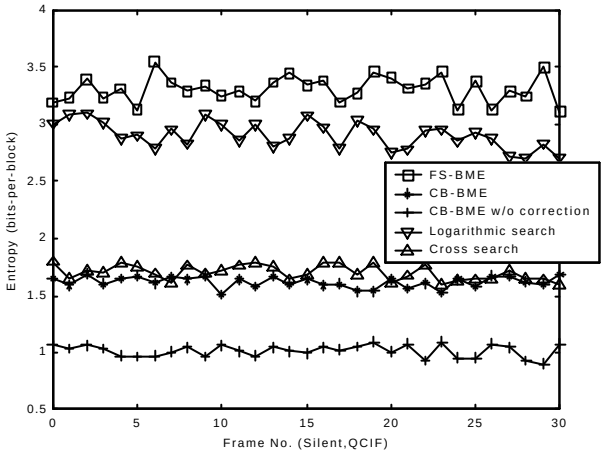


(c)

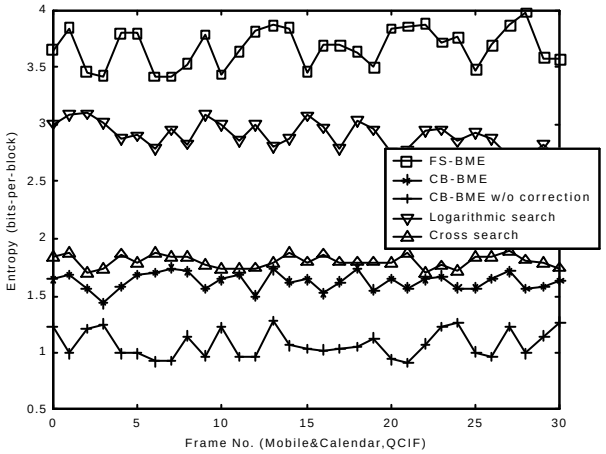
Figure 9. Complexity expressed in percentage versus full search for three test sequences :
(a) Flower Garden, (b) Silent, and
(c) Mobile&Calendar



(a)



(b)



(c)

Figure 10. Comparison between FS-BME, logarithmic search, cross search, CB-BME with and without correction in terms of entropy of the resulting bitstream. 30 consecutive frames for each test sequence are considered: (a) Flower Garden, (b) Silent, and (c) Mobile&Calendar

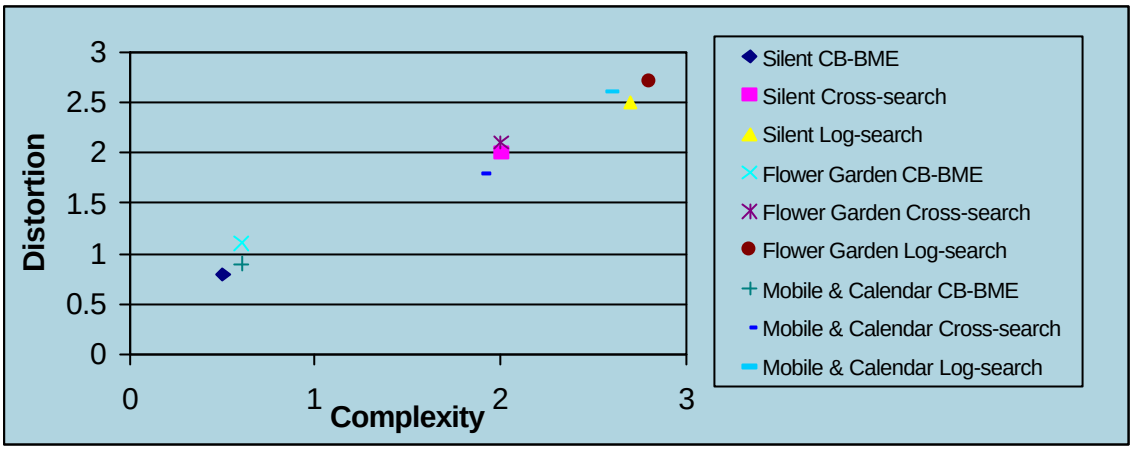


Figure 11. Graphical comparison among different ME schemes, in terms of average complexity and distort ion, using different test sequences.

Sequence	Average PSNR (H.263 with CB-BME)	Average PSNR (H.263)
FlowerGarden (QCIF) @ 64 kbps	22.34	21.82
FlowerGarden (QCIF) @ 128 kbps	24.72	23.80
Silent (QCIF) @ 64 kbps	33.90	33.76
Silent (QCIF) @ 128 kbps	35.89	35.39
Mobile&Calendar (CIF) @ 450 kbps	24.09	23.15
Mobile&Calendar (CIF) @ 640 kbps	24.48	24.29

Table 1. Comparison in terms of average PSNR values between H.263 (TMN-2.0) and H.263 (same version) implementing CB-BME at different

IV. CONCLUSIONS

An algorithm for efficient motion estimation and coding in video sequences was presented, based on the concepts of spatio-temporal context and predictive coding. The technique provides a sharp bitrate and complexity reduction over other existing approaches. Experimental results showed that the proposed method is very advantageous also when compared to other fast search methods, for it is able to lower bitrate and complexity at the same time. The trade-off between distortion and complexity can be modulated by a single parameter, which was experimentally proven to be non critical for the performance of the technique. Furthermore, a very convenient representation the motion field is proposed that allows to achieve a very compact encoding.

V. REFERENCES

[1] J.R. Jain, and A.K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Transactions on Communications*, Vol.29, No.12, pp.1799-1808, Dec. 1981.

[2] M. Ghambari, "The Cross-Search Algorithm for Motion Estimation," *IEEE Transactions on Communications*, Vol.38, No.7, pp.950-953, July 1990.

[3] B. Liu, and A. Zaccarin "New Fast Algorithms for the Estimation of Block Motion Vectors," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.3, No.2, pp.148-157, April 1993.

[4] L.K. Liu, and E. Feig "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.6, No.4, pp.419-422, August 1996.

[5] X. Lee, and Y.Q. Zhang "A Fast Hierarchical Motion-Compensation Scheme for Video Coding Using Block Feature Matching," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.6, No.6, pp.627-635, December 1996.

[6] K. Sauer, and B. Schwartz, "Efficient Block Motion Estimation Using Integral Projections," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.6, No.5, pp.513-518, Oct. '96.

[7] M. Bierling, "Displacement Estimation by Hierarchical Block Matching," *Visual Communication and Image Processing, Proc. SPIE*, Vol. 1001, pp. 942-951, 1988.

[8] M. Accame, F.G.B. De Natale, and D.D. Giusto, "Hierarchical Motion Estimator (HME) for Block-Based Video Coders," *IEEE Transactions on Consumer Electronics*, Vol. 43, No. 4, pp.1320-1330, November 1997.

[9] P. Kauff, O. Schreer, J.-R. Ohm, "An Universal Algorithm for Real-Time Estimation of Dense Displacement Vector Fields," *Proc. of Int. Conference on Media Futures*, Florence, Italy, May 2001.

[10] D.-K. Lim, and Y.-S. Ho, "A Fast Block Matching Motion Estimation Algorithm Using Optimal Search Patterns," *Proceedings of the SPIE*, vol. 4310, pp. 767-775, 2001.

[11] J.-Y. Nam, J.-S. Seo, J.-S. Kwak, M.-H. Lee, and Y. H. Ha, "New Fast-Search Algorithm for Block Matching Motion Estimation Using Temporal and Spatial Correlation of Motion Vector," IEEE Trans. On Consumer Electronics, vol. 46, No. 4, pp. 934-942, Nov. 2000.

[12] J. Chalidabhongse, and C.-C. J. Kuo, "Fast Motion Vector Estimation Using Multiresolution-Spatio-Temporal Correlations," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.7, No.3, pp.477-488, June 1997.

[13] T. Ebrahimi, E. Reusens, and W. Li, "New Trends in Very Low Bitrate Video Coding," *Proceedings of the IEEE*, Vol.83, No.6, pp. 877-891, June 1995.

[14] M. Accame, and F. Granelli, "Autocompensation-with-parameter for Motion Field Prediction," *Electronics Letters*, Vol.34, No.1, pp.51-52, 8th January 1998.

[15] J. Vetterli, and M. Khansari, "Motion Compensation of Motion Vectors," *Proc. International Conference on Image Processing* (ICIP-95), Washington D.C., USA, October 23-26, 1995.

[16] J. Zan, M.O. Ahmad, and M.N.S. Swamy, "Median Filtering-Based Pyramidal Motion Vector Estimation", *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-2001)*, Vol. 3, pp. 1605-1608.

[17] A.K. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall, 1989.

[18] ITU-T Recommendation H.263. Video Coding for Low Bitrate Communication, 11/95.