ADAPTIVE MULTI-PATTERN FAST BLOCK-MATCHING ALGORITHM BASED ON MOTION CLASSIFICATION TECHNIQUES

Iván González-Díaz, Manuel de-Frutos-López, Sergio Sanz-Rodríguez, Fernando Díaz-de-María

Department of Signal Theory and Communications
Universidad Carlos III, Leganés (Madrid), Spain

ABSTRACT

Motion estimation is the most time-consuming subsystem in a video codec. Thus, more efficient methods of motion estimation should be investigated. Real video sequences usually exhibit a wide-range of motion content as well as different degrees of detail, which become particularly difficult to manage by typical block-matching algorithms. Recent developments in the area of motion estimation have focused on the adaptation to video contents. Adaptive thresholds and multi-pattern search algorithms have shown to achieve good performance when they success to adjust to motion characteristics. This paper proposes an adaptive algorithm, called MCS, that makes use of an especially tailored classifier that detects some motion cues and chooses the search pattern that best fits to them. Specifically, a hierarchical structure of binary linear classifiers is proposed. Our experimental results show that MCS notably reduces the computational cost with respect to an state-of-the-art method while maintaining the quality.

Index Terms— Block-matching, motion estimation, motion classification, binary linear classifier

1. INTRODUCTION

Motion estimation is the most time-consuming subsystem in a video codec. Thus, more efficient methods of motion estimation should be investigated. In this paper we deal with this problem focusing on the emerging H.264 standard, that has shown an excellent behaviour in a broad range of coding applications. Furthermore, H.264 incorporates new features such as the variable-size block estimation or multi-reference coding that make the motion estimation more effective but also more time-consuming.

Most recent developments in the area of motion estimation have focused on the adaptation to video contents. Real video sequences contain a wide-range of motion contents, distributed along spatial regions whose shape and position change over time. Recognizing these features can help with the selection of appropriate search patterns that minimize both the number of search points and the distortion.

Initial research on block-matching adaptation made special emphasis on the generation of adaptive thresholds: in [1] accumulated costs of previous blocks on the frame are used to generate adaptive thresholds that lead the search process; in [2] additional spatial regionalization is applied by computing costs of four spatial neighbours; however, the usefulness of these thresholds is still limited to the detection of early stops during the search process. Other developments, as in [3], employ various search patterns and select the most appropriate based on motion classification techniques; the main drawback of these techniques is that motion classification strongly depends on previous search results; thus, local minima may affect the next pattern selections, potentially inducing new errors on next blocks. In [4] this dependence is weaker, since the classification method uses a dispersion measure over the local vector field. Even when a local minimum occurs, dispersion values can be large enough to select more exhaustive patterns in next blocks. Anyway, this measure by itself may not be robust enough to successfully classify every case.

Our proposal uses a complete classification scheme that selects the most appropriate search pattern based on estimated motion characteristics. The objective of this scheme is to reduce computational complexity on those cases where faster patterns can reach the optimal solution, while maintaining more exhaustive search patterns when needed.

The remainder of this paper is organized as follows. The details of our motion classification method are given in Section 2. A description of the *average cost map* used in the algorithm is provided in Section 3. Section 4 contains a performance comparison between our algorithm and another solution that has demonstrated good efficiency at any bitrate. Finally, Section 5 summarizes our conclusions and outlines further work.

This work has been partially supported by PRODYS S.L. and the regional grant UC3M-TEC-05-059. The authors would also like to thank David Muñoz-Mejías and José Carlos Pujol-Alcolado for their inestimable help.

2. MOTION CLASSIFICATION-BASED SEARCH (MCS)

The Motion Classification-based Search (MCS) chooses among several heterogeneous search patterns one that best fits particular motion contents. In particular, we pose this search pattern selection as a multi-class recognition problem that relies on some information available at coding time in order to characterise the motion and to select the search pattern that seems to be more robust and efficient on each case. As in any classification problem, we need to carefully define the inputs (parameters), the outputs (classes) and the classifier structure.

2.1. Local motion vector predictions

Motion vector (MV) predictions are used as potential starting points of posterior refinements of the local search. The H.264 standard defines a median-based prediction from the MVs of spatially adjacent blocks. However, this prediction can be insufficient and usually other correlated vectors are included to minimize later unnecessary searches around the initial search point. Our particular prediction set also includes:

1. Compensated MV of the co-located block.
2. Compensated MVs of four neighbouring blocks (left, up, up-left, up-right) on their best reference frame.
3. Median of compensated neighbouring vectors.
4. Uplayer vector (that resulting from the motion estimation performed for the next larger block size in the same reference).
5. Vector (0,0).

Vector compensation scales vector magnitudes when they have been coded using different time distances (in visualization order) between the reference frame and the current frame. This process is explained in detail in [5].

2.2. Problem parameterization: selected input features

The problem parameterization is a critical issue in any classifier design. Based on correlation estimates between potential parameters and desired outputs, we have chosen nine inputs to be considered by our classifier, namely:

1. **Block size:** expressed in terms of the number of 4x4 blocks (1 for 4x4 to 16 for 16x16).
2. **Binary input for homogeneous neighbourhood:** when the four spatial neighbouring blocks (left, up, up-left, up-right) have the same compensated MVs this input is set to 1, otherwise is set to 0.
3. **Time distance:** when the time distance (presentation order) between a frame and its reference is larger, the expected time correlation between them decreases and more exhaustive searches are needed.
4. **Number of neighbouring INTRA blocks:** except in borders, INTRA coded blocks may indicate non-uniform motion that has been impossible to detect. Detecting this situation can prevent from continuously falling in local minima.
5. **Absolute cost ($\mathbf{C_{abs}}$):** The cost is measured as a sum of a distortion term (SAD or Sum of Absolute Differences) and a term that refers to the differential coding of the MV (compund of a Lagrangian Term λ and a difference in bits between the prediction and the final MV). The absolute cost of the prediction is linearly adapted to 16x16 block-size and, then, made independent from the λ value by means of:

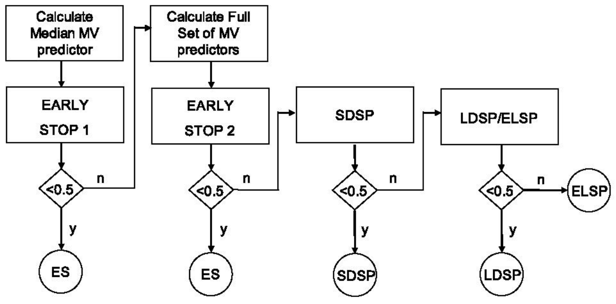
$$C_{abs} = C_{pred} - 45\lambda \tag{1}$$

   The value of 45λ has been empirically obtained by observing the evolution of the costs with respect to λ.
6. **Cost ratio ($\mathbf{C_{ratio}}$):** a ratio between predicted MV costs and the Average Cost Map that will be described in section 3. The prediction cost is linearly adapted to 16x16 block-size and, then, the ratio is calculated as:

$$C^{i}_{ratio} = \frac{C^{i}_{pred}}{T^{n,i}_{avg}} \tag{2}$$

7. **Prediction MV magnitude:** large prediction vectors are less reliable and usually require more exhaustive search patterns. The magnitude is calculated as:

$$Mag = |MV_{pred}(x)| + |MV_{pred}(y)| \tag{3}$$



Calculate Median MV predictor
Calculate Full Set of MV predictors
EARLY STOP 1
EARLY STOP 2
SDSP
LDSP/ELSP
<0.5
n
y
ES
SDSP
LDSP
ELSP

Fig. 1. Classification scheme of MCS. *Early Stop 1* uses inputs $\{1,3,5,6,7,9\}$ while the rest of binary classifiers employ the complete input set

8. **Motion Vector Dispersion (MVD):** dispersion of temporal (co-located) and spatial (adjacent) correlated MVs can be useful to characterize the uniformity of the MV field of a block. Our dispersion measure obeys:

$$D = \frac{1}{N}\sum_{i=1}^{N}(|MV_i(x) - MV_{pred}(x)| + |MV_i(y) - MV_{pred}(y)|) \tag{4}$$

9. **Inverse of λ (λ^{-1}):** In our experiments, λ^{-1} has turned out to be more suitable to our purposes than λ.

2.3. Classification outputs: search patterns

In the MCS, the outputs of the classification are the search patterns described below:

1. **Early Stop (ES):** the algorithm considers a prediction as the optimal solution and avoid any refinement.
2. **Small Diamond Search Pattern (SDSP):** a diamond pattern with a 1 pixel step-size is used until a center point of one iteration is better than the other points.
3. **Large Diamond Search Pattern (LDSP):** a diamond pattern with a 2 pixels step-size is used until a center point of one iteration is better than the other points; afterwards, a refinement is carried out by means of an SDSP pattern.
4. **Exhaustive Logarithmic Search Pattern (ELSP):** this pattern extends the well-known TSS [6] with a new stage. Specifically, it comprises 4 stages in which the step size decreases logarithmically starting from an initial step size of 8 pixels (sizes of 8, 4, 2, 1). This search pattern tries to emulate the functionality of the *search range dependent predictor set* presented in [5] and overcome difficult situations such as those when predictions are far from the optimal solution or when the cost functions are not monotonic towards the optimal solution.

2.4. Classification scheme

Although the block-matching process is most time-consuming subsystem of the video codec, it is actually due to the high number of block-matching operations rather than to their individual cost. Therefore, the block-matching algorithm itself cannot incorporate complex and expensive classifiers that may take as much time as the posterior search patterns. Based on this assumption, we have chosen *binary linear classifiers* as an example of learning machines that can get accurate results with low computational cost. A binary linear classifier obeys:

$$y = \mathbf{w}^T \mathbf{x} + b \tag{5}$$

where $\mathbf{w}$ represents the vector of weights, $\mathbf{x}$ is the input vector with the parameter values (scaled between 0 and 1), y is the output of the classifier and b is the bias term. This classifier makes soft decisions that afterwards must be compared with a threshold (0.5 in our case), providing a hard decision (0 or 1). We use several binary classifiers to solve the multi-class (multi-pattern) problem. In particular, we propose a hierarchical structure, illustrated in Fig. 1, that sequentially performs four binary classifications. The output of each binary classifier is compared with a fix threshold, so that smaller values imply a final classification decision and larger values lead to subsequent stages. Input values for *Early Stop 1* classifier are related to H.264 standard median predictor. If an early stop is not detected, the full set of predictors is computed and the best initial vector is chosen, thus subsequent classifiers use updated information.

2.5. Training and test sets generation

In order to design every binary classifier, a training set and a test set (to evaluate the solution in terms of classification accuracy) have been generated by encoding typical video sequences for a thorough grid of QP values. These sets have been designed taking into account two main requirements: 1) they should be representative of the input space; and 2) they should make special emphasis on those samples in which selecting erroneous patterns produces unacceptable bitrate increases. Ignoring the second requirement leads to classifiers that always select classes with higher prior probability (early stops in our case). To meet both requirements, we assign to every sample a probability of being included on the training/test set according to the cost obtained by every potential search pattern for this particular sample (in order to compute this probabilities every pattern is tried for every sample):

$$P = K \left(R + \frac{1}{C_{min}} \sum_{i=1}^{N} (C_i - C_{min}) \right) \tag{6}$$

where C_i is the cost achieved by the search pattern i, N is the number of search patterns, C_{min} is the minimum cost (that obtained by the optimal search pattern), R is a regularization term which ensures that any sample has a non-zero probability of being included on the set (even in early stops when $C_i = C_{min}$), and K is a function that manages the size of the training/test sets and guarantees that $0 \leq P \leq 1$.

2.6. Classifiers training

The training phase, i.e., the optimization of weights, is based on the minimization of the Mean Square Error (MSE) and LMS (Least Mean Squares) algorithm is used. However, it has not been possible to apply this procedure to every parameter. In particular, if we introduce λ^{-1} on the optimization process, its corresponding weight turns out to be positive. This implies that faster patterns are strengthened when working at lower qualities, what generates substantial bitrate increases. Therefore, we need to be more conservative (using more exhaustive patterns) at high QP values (high λ values). To overcome this issue we have trained classifiers that use all input features except λ^{-1} and, subsequently, got the optimal value of λ^{-1} weight by means of a cross-validation process.

2.7. Switching among search patterns

Since our algorithm makes some classification errors, switches among search patterns should be considered for potential error detections. Although many potential switches have been tested, we have finally included just one change between ELSP and SDSP, that is taken when at the end of the second step of the ELSP, the center is the best point and

$$C^i < \left(\alpha + \frac{\beta}{\lambda} \right) T_{avg}^{n,i} \tag{7}$$

where α and β have been empirically obtained. Again, when λ decreases, switches to faster patterns become more likely.

3. AVERAGE COST MAP

In order to achieve a good level of adaptation to several video features such as motion nature or background detail, we have developed an Average Cost Map (ACM). The ACM allows us to adapt the costs to the region contents (for example: high detailed regions usually imply high distortion values even when the block-matching process has reached the optimal solution). The averaging process is made in two separate domains: space and time. In space, an spatial average S_{avg}^i is made for each one macroblock-size region i as a weighted arithmetic mean of the costs of the macroblock i and its eight spatial neighbours:

$$S_{avg}^i = \alpha C_i + \frac{1-\alpha}{N} \sum_{j=1}^{N} C_j \tag{8}$$

where C_i is the accumulated cost of the MB i, $N = 8$ represents the eight adjacent blocks, and α is a weighting factor ($0 \leq \alpha \leq 1$) that establishes the relative importance of a block with respect to its neighbours. Then, a time average is performed as follows:

$$T_{avg}^{n,i} = \beta S_{avg}^i + (1-\beta) T_{avg}^{n-1,i} \tag{9}$$

where n is the time instant, β is a weighting factor ($0 \leq \beta \leq 1$) that establishes the relative importance of new data with respect to previous samples.

4. EXPERIMENTAL RESULTS

The proposed algorithm has been embedded in the H.264/AVC Reference Software Encoder (v10.2). Simulations have been made using IBBPBBP pattern, non-optimized RD (for real-time coding applications), search area of 33x33, 5 references and exhaustive subpel refinement. A thorough grid of QP values (1-51 with two step size) has been used in order to obtain results at bitrates very closed to those ones included on the Tables. The results at these particular bitrates have been obtained by linear interpolation.
Tables 1 and 2 show, respectively, *PSNR vs bitrate* and *computational complexity* (in terms of total search points per block and total coding time) comparison among Full Search (FS), full EPZS

Table 1. Performance comparison (PSNR) among Full Search (FS), EPZS and MCS

PSNR and PSNR variation Δ (dBs) vs Bitrate (Kbps)								
			128	**256**	**512**	**1024**	**1536**	**2048**
Mobile	**FS**	PSNR	20.20	23.32	26.50	30.26	32.81	34.83
	EPZS	Δ	0.12	-0.06	-0.03	-0.01	0.01	0.01
	MCS	Δ	0.10	-0.07	-0.03	-0.02	-0.02	-0.01
Coastguard	**FS**	PSNR	26.89	28.98	31.28	34.07	36.05	37.74
	EPZS	Δ	0.02	0.01	0.00	0.01	0.04	0.05
	MCS	Δ	0.01	-0.01	-0.02	0.00	0.05	0.07
Container	**FS**	PSNR	34.34	37.22	40.17	43.12	44.95	46.28
	EPZS	Δ	0.13	0.06	0.04	0.04	0.02	0.00
	MCS	Δ	0.12	0.07	0.04	0.04	0.04	0.02
Tempete	**FS**	PSNR	26.11	28.98	31.83	35.07	37.26	39.03
	EPZS	Δ	0.10	0.02	-0.01	-0.01	0.01	0.02
	MCS	Δ	0.10	0.01	-0.02	-0.01	0.02	0.02

Table 2. Performance comparison (Search Points/Mean Coding Time per frame in seconds) among Full Search (FS), EPZS and MCS

Sequence	FS	EPZS	MCS
Mobile	1089/ 4.63	10.99/ 0.80	7.96/ 0.73
Coastguard	1089/ 6.44	13.74/ 0.97	8.14/ 0.85
Container	1089/ 4.83	5.48/ 0.74	1.40/ 0.64
Tempete	1089/ 5.08	10.99/ 0.88	4.85/ 0.76

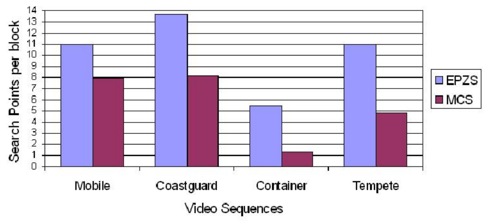
[5] (with extended diamond pattern and fixed, temporal and spatial memory predictors), and the proposed MCS algorithm. On the one hand, the FS is included just to provide reference PSNR vs bitrate results. On the other hand, the EPZS algorithm has shown good performance for a wide-range of motion contents and coding qualities, while it is computationally much more affordable than FS. However, it is still computationally expensive since it uses many predictors and search points with independence of the coding situation.

MCS results in PSNR (see Table 1) are really close to those achieved by EPZS and FS, with a mean loss of $0.01 dBs$ and a benefit of $0.05 dBs$, respectively. Furthermore, PSNR losses do not increase at lower qualities, issue that becomes a classical problem for many block-matching algorithms. Poor FS results on low bitrates are due to the cost approximation used with the non-optimized RD coding.

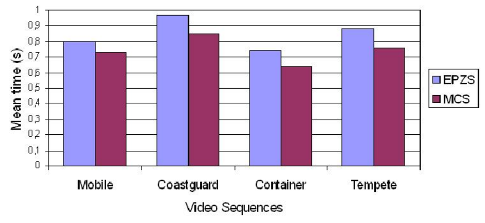
On the other hand, MCS algorithm achieves a mean computational complexity reduction (see Table 2) of 99.5% versus FS and 49.73% versus EPZS in mean search points per block. This reduction has an impact on the total coding time, with decreases of 86.6% versus FS and 12.3% vs EPZS. Complexity comparisons between EPZS and MCS are also provided in Figure 2.

5. CONCLUSIONS AND FURTHER WORK

The wide range of real-time video coding applications demands algorithms able to work at a wide range of qualities. MCS is intended to work at any coding quality as well as to follow different types of motion content. Motion classification techniques, which are the core of the MCS, can become the basis for the development of adap-



(a) Mean Search Points per block



(b) Mean total coding time per frame

Fig. 2. Computational complexity comparison between EPZS and MCS in mean search points per block (a) and total encoding time (b)

tive algorithms that fulfil these requirements. The reported results show that MCS obtains similar quality values that known robust algorithms while notably reducing the computational complexity.

Further work mainly focuses on the development of different costs functions with two aims: the first one is to include λ^{-1} in the training phase in the same way that the rest of the input features. The second is to make special emphasis on difficult cases without modifying the prior probabilities of the classes.

6. REFERENCES

[1] Xiaoquan Yi and Nam Ling, "Improved partial distortion search algorithm for rapid block motion estimation via dual-halfway-stop," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05).*, 2005, vol. 2, pp. ii/917–ii/920 Vol. 2.

[2] Jang-Jer Tsai and Hsin-Chia Chen, "Predictive block-matching discrepancy based rhombus pattern search for block motion estimation," in *IEEE International Conference on Image Processing. ICIP 2005.*, 2005, vol. 1, pp. I–1073–6.

[3] Shih-Yu Huang, Chuan-Yu Cho, and Jia-Shung Wang, "Adaptive fast block-matching algorithm by switching search patterns for sequences with wide-range motion content," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 11, pp. 1373–1384, 2005.

[4] I. Ahmad, Weiguo Zheng, Jiancong Luo, and Ming Liou, "A fast adaptive motion estimation algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 420–438, 2006.

[5] A. M. Tourapis, "Fast me in the jm reference software," Joint Video Team (JVT) of ISO/IEC MPEG ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6) 16th Meeting: 24-29 July 2005, Poznań. JVT-P026.

[6] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing.," in *NTC*, 1981, pp. vol. 4, pp G5.3.1–G5.3.5.