IMPROVED MOTION CLASSIFICATION TECHNIQUES FOR ADAPTIVE MULTI-PATTERN FAST BLOCK-MATCHING ALGORITHM

Iván González-Díaz, Fernando Díaz-de-María

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

ABSTRACT

In several video coding standards, such as H.264, motion estimation becomes the most-time consuming subsystem. Therefore, recently research on video coding focuses on the development of novel algorithms able to save computations with minimal effects over the coding distortion. Due to the fact that real video sequences usually exhibit a wide-range of motion content, from uniform to random, and to the vast amount of coding applications demanding different degrees of coding quality, adaptive algorithms have revealed as the most robust general purpose solutions. In particular, multi-pattern algorithms can adapt to video contents as well as to required coding quality by means of the use of a set of heterogeneous search patterns, each one adapting better to particular motion and quality requirements. This paper applies some improvements to the Motion Classification based Search, an adaptive multi-pattern algorithm based on motion classification techniques. Our experimental results show that MCS notably reduces the computational cost with respect to some well-known algorithms while maintaining the quality.

***Index Terms*—** Block-matching, motion estimation, multi-pattern algorithms, motion classification, binary linear classifier

1. INTRODUCTION

Motion Estimation (ME) constitutes an essential process in video coding that consumes a notable portion of the total processing time. On this classically time-consuming task, the new standard H.264 incorporates some novel features, such as variable-size motion compensation or multi-reference coding, that provide great flexibility at the expense of notably increasing the complexity of the ME. Consequently, fast block-matching algorithms are even more necessary in order to minimize the number of search points evaluated to find the optimal solution while maintaining low distorsion values.
Several features, such as zonal searches (centered on predictions obtained from spatial and temporal correlated information) or early stop conditions, have shown to be very efficient in achieving time savings in ME. However, current block-matching algorithms still present some limitations and have difficulties adapting to video contents and motion characteristics.
Real video sequences contain a wide-range of motion contents, distributed along spatial regions whose shape and position change over time. Most recent developments in the area of motion estimation have focused on the adaptation to video contents, in many cases, by means of the utilization of multi-pattern search algorithms. These algorithms make use of various search patterns, selecting the most appropriate to video contents and motion nature. However, multi-pattern solutions are only reliable if it is possible to find a robust pattern selection algorithm.
Many of these algorithms select the search pattern based on an intial motion estimation. In [1] two intial search patterns are employed for this prupose and, later, other algorithms are used depending on the first results. EPZS algorithm [2] uses a large set of predictions, specially for the *search range dependent prediction set.* case, and also evaluates some initial patterns (increasing the number of search points). Other approaches employ a pattern selection method that uses a motion classification measure. In particular, in [3], a dispersion measure over the local vector field is used to characterize motion and select an initial pattern which, afterwards, can switch to other patterns depending on the block-matching performance. In [4] the same measure is used to choose the initial pattern but, additionally, a MB-level computation allocation system manages the number search points and uses other patterns when possible. Anyway, this intial measure may not be robust enough to successfully classify every case and the algorithm can fall in local minima when using non-robust patterns at the initial step.
The Motion Classification based Search (MCS) proposed in [5] introduces the utilization of well-known general purpose binary linear classifiers to provide a hierarchical classification scheme that selects the most appropriate search pattern. In order to achieve a robust enough decision, the classification relies on 9 input features, while the simplicity of the classifiers makes the process computationally affordable. In this paper we present some improvements of the MCS algorithm.
The remainder of this paper is organized as follows. A review of the MCS algorithm is provided in Section 2. The proposed extensions of the MCS are discussed in Section 3. Section 4 contains a performance comparison between our algorithm and another solution that has shown to be efficient at any bitrate. Finally, Section 5 summarizes our conclusions and outlines further work.

2. A REVIEW OF THE MOTION CLASSIFICATION-BASED SEARCH (MCS)

The MCS algorithm [5] chooses among several heterogeneous search patterns that one which best fits the current motion characteristics. This novel approach considers the search pattern selection as a multi-class classification problem and makes use of low computational cost classifiers to achieve this purpose.

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. MCS also includes an extended prediction set, with compensated vectors of co-located and four neighbouring blocks, median of the aforementioned compensated vectors, uplayer vector (that resulting from the motion estimation performed for the next larger block size in the same reference) and static 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 [2].

2.2. Problem parameterization: selected input features

The problem parameterization involves the selection of the input parameters which adjust better to the classification model. Based on a previous study about the correlation coefficients (linear relationship) between potential parameters and desired outputs, we have chosen nine inputs, 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 have the same compensated MVs this input is set to 1, otherwise is set to 0.
3. **Time distance:** visualization time distance between a frame and its reference.
4. **Number of neighbouring blocks that been coded as INTRA**
5. **Absolute cost ($\mathbf{J_{abs}}$):** 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:

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

The value of 45λ has been empirically obtained by observing the evolution of the costs with respect to λ.

6. **Cost ratio ($\mathbf{J_{ratio}}$):** a ratio between predicted MV costs and the Average Cost Map that will be described in section 2.3. The prediction cost is linearly adapted to 16x16 block-size and, then, the ratio is calculated as:

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

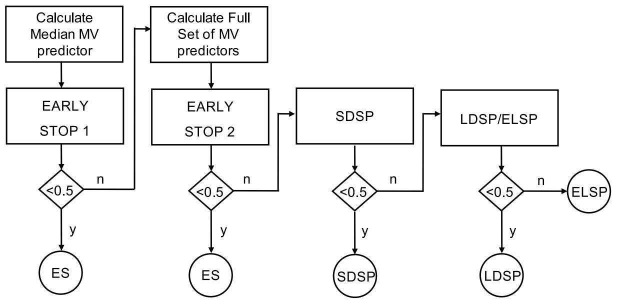
7. **Prediction MV magnitude:** large prediction vectors are less reliable and usually require more exhaustive search patterns. The magnitude is calculated as:

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

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.



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

2.3. Average Cost Map

In order to achieve a good level of adaptation to several video features such as motion nature or background detail, MCS includes an Average Cost Map (ACM). The ACM allows us to adapt the distortion 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 C^i_{avg} is made for each one macroblock-size region i as a weighted arithmetic mean of the distortions of the macroblock i and its eight spatial neighbours:

$$S^i_{avg} = \alpha J_i + \frac{1-\alpha}{N}\sum_{j=1}^{N} J_j \tag{5}$$

where J_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:

$$T^{n,i}_{avg} = \beta S^i_{avg} + (1-\beta)T^{n-1,i}_{avg} \tag{6}$$

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.

2.4. 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 is described in [5]. Its use is intended to overcome difficult situations such as those when predictions are far from the optimal solution or when the cost curves are not monotonic towards the optimal solution.

2.5. Classification scheme

The MCS bases its functionality on a serial classification scheme which is illustrated on figure 1. This scheme goes from faster to slower search patterns and its utilization, rather than the use of parallel structures, allows us to collect misclassified samples corresponding to faster patterns and classify them as the next pattern (thus minimizing the computational increase due to an incorrect classification). With this approach we can strengthen simple patterns to reduce computational complexity.
Each pattern detector is performed by a binary linear classifier, an example of low computational cost learning machines that can get accurate results in an efficient manner. A binary linear classifier obeys:

$$\tilde{y} = \mathbf{w}^T\mathbf{x} + b \tag{7}$$

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).
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.
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) 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, the achieved weight does not lead to a good solution (refer to [5] for details). 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.

3. PROPOSED EXTENSIONS OF THE BASIC MCS

3.1. Use of distortion measures instead of cost measures

The first improvement concerns cost measures, absolute or relative, used in (1),(2) and (5). We have observed that cost values are problematic at high QP values due to the term associated to the motion vector coding. This term becomes very large for small partitions, since the linear adaptation of the costs entalis multiplying by the number of trasmitted motion vectors (i.e., the number of partitions). For example, the use 4x4 blocks implies 16 motion vectors per macroblock. This fact leads to the selection of exahustive patterns when coding small blocks (nearly independently of the distance between the prediction and the optimal solution), with the consequent increase of computational complexity. For this reason, we suggest to use the SAD as a more appropriate measure for this purpose.
To be more precise, we absolute SAD instead of absolute Cost in eq. (1):

$$SAD_{abs} = SAD_{pred} - 25\lambda \tag{8}$$

The value of 25λ has been empirically obtained by observing the evolution of the SAD with respect to λ. Furthermore, the spatial average of the new Average Distortion Map (ADM) obeys:

$$S^i_{avg} = \alpha SAD_i + \frac{1-\alpha}{N}\sum_{j=1}^{N} SAD_j \tag{9}$$

where SAD_i is the accumulated SAD 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. The time average $T^{n,i}_{avg}$ remains as in (6). Finally, SAD ratio is used in (2) instead of the Cost ratio:

$$SAD^i_{ratio} = \frac{SAD^i_{pred}}{T^{n,i}_{avg}} \tag{10}$$

3.2. A new cost function for the binary linear classifiers

As an extension of the basic MCS, we have designed a novel cost function to train binary classifiers. This new cost function obeys to two main objectives:

1. To avoid a suboptimal selection of the weight associated to λ^{-1} (in the previous version, the best weight for λ^{-1} is estimated once the other weights are fixed).
2. To avoid the design of a modified training set in which the prior probability of the samples is modified in order to put more emphasis on those samples more sensitive to classification errors, i.e., those samples for which an error in the search pattern selection leads to a high increment of the bit rate.

Therefore, the new cost function allows us to directly obtain the weight associated to λ^{-1} and, besides, incorporates the cost due to classification errors on the pattern selection process, making more emphasis on those critical samples without altering their prior probabilities.
Specifically, the proposed cost function obeys:

$$C(\alpha, \beta, \gamma) = e^2[1 + \alpha(J_i - J_{min})^\beta \lambda^\gamma_M] \tag{11}$$

where:

- $e = y - \tilde{y}$ is the output error, being y the desired output (0 or 1) and $\tilde{y}$ the obtained output as defined in (7).
- J_i is the cost achieved by the search pattern i. J_{min} is the minimum cost (that obtained by the optimal search pattern). The cost is measured as a sum of a distortion term (SAD) and a term that refers to the differential coding of the MV.
- λ is the Lagrangian parameter that weigths the two terms of the cost
- α, β, γ are the parameters which must be optimized to adapt the cost function to the coding model. β manages how the variation of classification errors affects the training, γ controls the influence of the coding quality, and α assigns a relative influence of these two effects over the displacement of the classification boundary.

Furthermore, our training algorithm only updates the weights when a classification error is made. Only in those cases, the training algorithm moves the classification boundary in the appropriate direction (given by the derivative of the error).
Such a parametric function needs a cross-validation process in order to select those parameters which best fit our model. Thus, each combination of parameters is firstly trained on the training set, which is made of 40000 samples (block-matching operations) randomly obtained from the coding of typical video sequences for a thorough grid of QP values. Then, the weights obtained for the combination of α, β and γ that achieves the best results on a small validation set of video sequences are chosen. This validation set is composed of 25 frames of three heterogeneous sequences (Paris, Football and Coastguard) at a thorough grid of QP values.

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

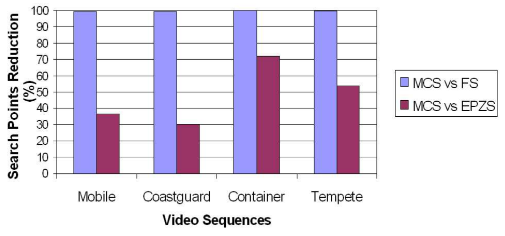
PSNR and PSNR variation Δ (dBs) vs Bitrate (Kbps)								
			128	**256**	**512**	**1024**	**1536**	**2048**
Mobile	**FS**	PSNR	19.61	22.24	25.26	29.23	31.79	33.85
	EPZS	Δ	0.03	0.01	-0.00	-0.01	0.01	0.01
	MCS	Δ	0.00	-0.01	-0.02	-0.04	-0.02	-0.02
Coastguard	**FS**	PSNR	26.30	28.32	30.56	33.37	35.27	36.89
	EPZS	Δ	0.05	0.03	0.03	0.02	0.05	0.10
	MCS	Δ	0.03	-0.01	0.01	0.01	0.05	0.12
Container	**FS**	PSNR	33.80	36.54	39.45	42.67	44.53	45.92
	EPZS	Δ	0.07	0.06	0.05	0.03	0.03	0.02
	MCS	Δ	0.07	0.06	0.04	0.01	0.01	0.00
Tempete	**FS**	PSNR	25.30	27.69	30.36	33.56	35.73	37.53
	EPZS	Δ	0.10	0.05	0.02	0.01	0.02	0.03
	MCS	Δ	0.05	0.02	-0.04	-0.01	0.00	0.02

4. EXPERIMENTAL RESULTS

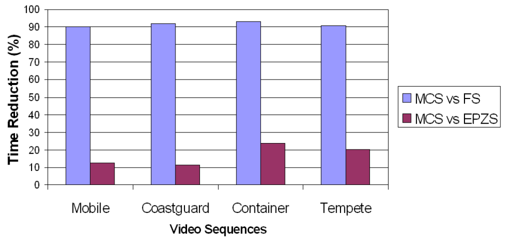
The proposed algorithm has been embedded in the H.264/AVC Reference Software Encoder (v10.2). Simulations have been made using IBBPBBP pattern, non-optimized RD (for real-time coding applications), search area of 33x33, 5 references and disabled subpel refinement (in order to get results that are independent of other modules in the coder). The test set involves a coding process over 100 planes of many video sequences (only results of some of them are included on this paper). A thorough grid of QP values (from QP 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 and Figures. The results at these particular bitrates have been obtained by linear interpolation. Table 1 and Figure 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 [2] (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 $0.02 dBs$ and a benefit of $0.01 dBs$, 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 in the local search.
On the other hand, MCS algorithm achieves a mean computational complexity reduction (see Figure 2) of 99.55% versus FS and 48.11% versus EPZS in mean search points per block. This reduction has an impact on the total coding time, with decreases of 91.45% versus FS and 17.04% vs EPZS.

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



(a) Search points reduction of the MCS



(b) Total coding time reduction of the MCS

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

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 adaptive 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 inclusion of other learning machines for the binary classifiers as well as the design of the classification scheme using other structures, such as multi-class schemes.

6. REFERENCES

[1] C.-S. Yu and S.-C. Tai, "Adaptive double-layered initial search pattern for fast motion estimation," *Multimedia, IEEE Transactions on*, vol. 8, no. 6, pp. 1109–1116, 2006.

[2] 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, Poznaǹ. JVT-P026.

[3] 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.

[4] Ching-Yeh Chen, Yu-Wen Huang, Chia-Lin Lee, and Liang-Gee Chen, "One-pass computation-aware motion estimation with adaptive search strategy," *Multimedia, IEEE Transactions on*, vol. 8, no. 4, pp. 698–706, 2006.

[5] Iván González Díaz, Manuel de Frutos López, Sergio Sanz Rodríguez, and Fernando Díaz de María, "Adaptive multi-pattern fast block-matching algorithm based on motion classification techniques," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2007. (ICASSP '07).*, 2007.