Least Squares Motion Estimation Algorithm in the Compressed DCT Domain for H.26x / MPEG-x Video Sequences

Nuno Roma
Nuno.Roma@inesc-id.pt

Leonel Sousa
las@inesc-id.pt

Instituto Superior Técnico / INESC-ID
Rua Alves Redol No.9, 1000-029 Lisboa - PORTUGAL

Abstract

A new compressed domain motion estimation algorithm that makes use of the DCT coefficients directly obtained from the H.26x or MPEG-x video stream is presented. The proposed algorithm is based on an iterative scheme that computes the new motion vectors by applying a least squares estimation technique. To reduce its computational effort, the algorithm may consider only an arbitrary subset of non-null DCT coefficients. The performance of the algorithm was assessed in a DCT domain H.263 video transcoder, where the obtained motion vectors provided the means to significantly enhance the quality of the temporal prediction scheme with a consequent reduction of the required bit-rate.

1. Introduction

Video transcoding is an increasingly popular technique to adapt or change the characteristics of a given precoded video stream, such as its bit-rate, its spatial or temporal resolution or even the coding standard, to better suit the constraints and requirements of different transmission systems or terminal devices. To minimize the computational complexity of these systems, several techniques have been developed and proposed to directly operate in the Discrete Cosine Transform (DCT) domain [1]. With such algorithms, data can be processed in the same domain as it is received from the encoding system, thus offering significant advantages both in terms of computational complexity and distortion reduction [1].

One of these techniques that has been given more attention is Motion Estimation (ME) in the transform domain. In fact, many video processing systems require the computation of new Motion Vectors (MV)s that will be used to encode the new video stream. Most ME techniques that have been proposed up to now reuse the MVs decoded from the received video stream to estimate these new MVs by using several possible schemes, such as the simple mean, the weighted average, the median, etc. [2, 3, 4]. However, it has been observed that most of these schemes often require a post-processing re-estimation stage to refine the estimated MVs. Moreover, few algorithms have been presented to estimate entirely new MVs in the absence of any precoded MV (e.g. INTRA to INTER frame conversion) or when those MVs do not have any correlation with the motion activity in the new coded frame (e.g. logo insertion [5]).

One of these ME algorithms was proposed by Koc *et. al.* [6] and is based on the application of the DCT pseudo-phases techniques, by employing the sinusoidal orthogonal principle, to extract the shift information from the pseudo-phases hidden in the 2D DCT coefficients of the second kind (DCCT-II), decoded from the received stream. However, this algorithm presents a significant computational complexity and it makes use of other kinds of the discrete sine and cosine transforms of the processed blocks that cannot be directly obtained from the H.26x or MPEG-x video stream. In fact, not only does it require the computation of the discrete cosine transform of the first kind (DCCT-I) of the blocks under processing, but it also needs to compute the four variations of the discrete sine/cosine trigonometric transforms of the second kind: DCCT-II, DCST-II, DSCT-II and DSST-II. Moreover, all these 2D transformations must be computed for an image area corresponding to the macroblock under processing, which implies a further computational effort in order to calculate the coefficients obtained from the concatenation of the four decoded blocks.

Other approaches were also applied in photogrammetric techniques for image registration of JPEG compressed aerial photos [7] and tried to apply matching techniques using convolution in the DCT domain. Unfortunately, the experimental results demonstrated that not only does this algorithm present a poor precision in the obtained values, but it also suffers from a sign ambiguity in the disparity result.

Consequently, a different approach is presented in this paper to perform the estimation of new MVs using the DCT coefficients directly obtained from the H.26x or MPEG-x video stream. The proposed algorithm is based on an iterative scheme that estimates the new MVs by applying a Least Squares Estimation (LSE) technique and following an approach somewhat similar to the one frequently adopted

in image registration techniques [7]. Moreover, to reduce its computational effort, the algorithm may consider only an arbitrary subset of non-null DCT coefficients. To assess its performance, the proposed algorithm was incorporated in a H.263 video transcoder [5] that performs a simple down-scale operation in the DCT domain. The experimental results demonstrated that the obtained MVs provide the means to significantly enhance the quality of the prediction scheme with a consequent reduction of the required bit-rate.

2. Linear Least Squares Estimation

The LSE method is frequently used to fit a set of observed points to a given theoretical model with a set of adjustable parameters. One of the most popular applications of this method is found in linear regression, to fit a set of M data points (x_i, y_i) to a straight-line model:

$$y(x) = y(x; a, b) = a + bx + n(x), \tag{1}$$

where $n(x)$ is the Gaussian noise introduced by the sampling procedure.

The solution of this optimization process is often found by minimizing a merit function that estimates the error between the theoretical model and the observed samples:

$$\varepsilon^2(a, b) = \sum_{i=1}^{M} (y_i - a - bx_i)^2 \tag{2}$$

At this minimum, the derivatives of $\varepsilon^2(a, b)$ with respect to a and b will vanish:

$$\frac{\partial \varepsilon^2(a, b)}{\partial a} = -2 \sum_{i=1}^{M} (y_i - a - bx_i) = 0 \tag{3}$$

$$\frac{\partial \varepsilon^2(a, b)}{\partial b} = -2 \sum_{i=1}^{M} x_i (y_i - a - bx_i) = 0 \tag{4}$$

and the optimal values for the parameters a and b can be found by solving a linear system of equations.

Models like this, where there is a linear function in the estimated parameters, are denoted by *linear models*. On the other hand, when the models are not linear in the parameters, the solution can not be found by simply solving a linear system of equations. In such cases, the model is often denoted by:

$$y = f(x; \theta) + n(x), \tag{5}$$

where $f(x; \theta)$ is a known function, that is non-linear in the parameter θ. The least squares criterion for these cases becomes:

$$\varepsilon^2(\theta) = \sum_{i=1}^{M} (y_i - f(x_i, \theta))^2 . \tag{6}$$

By minimizing the derivatives of ε^2 with respect to the parameters θ_j, one obtain:

$$\frac{\partial \varepsilon^2(\theta)}{\partial \theta_j} = 0 \tag{7}$$

$$\Leftrightarrow -2 \sum_{i=1}^{M} (y_i - f(x_i, \theta)) \left. \frac{\partial f(x_i, \theta)}{\partial \theta_j} \right|_{\theta=\theta^0} = 0 \tag{8}$$

Since $f(x, \theta)$ is non-linear in the parameter θ_j, these equations are usually quite difficult to solve and this procedure is rarely used for non-linear cases. In such *non-linear models*, a Taylor series expansion is often used to approximate the non-linear regression model to a simpler model defined with linear terms and then employ the LSE method to estimate the parameters.

3. Least Squares Motion Estimation

The ME approach used in H.26x and MPEG-x video coding standards has a lot in common with the non-linear LSE described above. The main objective of such procedure is to find the best predicting macroblock in a search region defined in the previous image $s(x_1, x_2)$ that best matches the macroblock of the current image $r(x_1, x_2)$ under processing. Since most video coding standards restrict this displacement to a simple translational model, the output of this algorithm will be a motion vector $v = (v_1, v_2)$ that defines the distance between the macroblock in the current image $r(x_1, x_2)$ and the best predicting macroblock in the previous image $s(x_1 + v_1, x_2 + v_2)$.

However, although this translational model can be described with a simple linear equation:

$$\mathbf{y} = \mathbf{x} + \mathbf{v} \tag{9}$$

the procedure to find the best motion vector associated to all pixels of a given macroblock is highly non-linear. Hence, the non-linear LSE model, as described in eq. 5, proves to be quite suitable to be adopted and applied to the current and previous images macroblocks:

$$r(x_1, x_2) \simeq s(x_1 + v_1, x_2 + v_2). \tag{10}$$

By decomposing this expression in a first-order Taylor series expansion:

$$r(x_1, x_2) \simeq s(x_1 + v_1^0, x_2 + v_2^0) + + \sum_{n=1}^{2} \left. \frac{\partial s}{\partial v_n} \right|_{v=v^0} \left(v_n - v_n^0\right) \tag{11}$$

$$\simeq s(x_1 + v_1^0, x_2 + v_2^0) + + \left. \frac{\partial s}{\partial v_1} \right|_{v=v^0} \left(v_1 - v_1^0\right) + \left. \frac{\partial s}{\partial v_2} \right|_{v=v^0} \left(v_2 - v_2^0\right), \tag{12}$$

the desired displacement parameter v can be estimated by following an iterative procedure and by computing the several increments $dv^i = v^i - v^{i-1}$, where $v^0 = (0,0)$. The best matching macroblock is obtained by minimizing the prediction error $e(x_1, x_2)$:

$$e(x_1,x_2) = r(x_1,x_2) - s(x_1+v_1^i, x_2+v_2^i) \simeq \\ \simeq \left.\frac{\partial s}{\partial v_1}\right|_{v=v^{i-1}} dv_1^i + \left.\frac{\partial s}{\partial v_2}\right|_{v=v^{i-1}} dv_2^i \ . \quad (13)$$

The equation above can be represented as a linear system with M equations and 2 unknowns (dv_1^i and dv_2^i), where M denotes the number of considered samples:

$$\mathbf{e} = \mathbf{j_s} \cdot \mathbf{dv}^i \ . \quad (14)$$

In this equation, $\mathbf{e}$ denotes a $M \times 1$ vector obtained by rearranging the columns of the difference matrix $\mathbf{e} = \mathbf{r} - \mathbf{s}$:

$$\mathbf{e} = \left[\begin{array}{c} \vdots \\ r(x_l,x_c) - s(x_l+v_1^i, x_c+v_2^i) \\ \vdots \end{array} \right]_{M\times 1} \quad (15)$$

$\mathbf{j_s}$ is a $M \times 2$ matrix whose columns contain the partial derivatives of s disposed in major column order:

$$\mathbf{j_s} = \left[\begin{array}{cc} \vdots & \vdots \\ \frac{\partial s}{\partial v_1} & \frac{\partial s}{\partial v_2} \\ \vdots & \vdots \end{array} \right]_{M\times 2} \quad (16)$$

and $\mathbf{dv}$ is the desired displacement matrix:

$$\mathbf{dv^i} = \left[\begin{array}{c} dv_1^i \\ dv_2^i \end{array} \right]_{2\times 1} . \quad (17)$$

Eq. 14 can be solved by simple algebraic manipulation:

$$\mathbf{e} = \mathbf{j_s} \cdot \mathbf{dv^i} \quad (18)$$

$$\Leftrightarrow \mathbf{j_s}^T \cdot \mathbf{e} = \mathbf{j_s}^T \cdot \mathbf{j_s} \cdot \mathbf{dv^i} \quad (19)$$

$$\Leftrightarrow \mathbf{dv^i} = \left(\mathbf{j_s}^T \cdot \mathbf{j_s}\right)^{-1} \cdot \mathbf{j_s}^T \cdot \mathbf{e} \quad (20)$$

At the end of each iteration the partial displacement v is updated: $v^i = v^{i-1} + dv^i$ and the displaced image s is obtained by motion compensation and interpolation: $s^i(x_1 + v_1^i, x_2 + v_2^i)$. The algorithm is then repeated by computing the value of $\left.\frac{\partial s}{\partial v}\right|_{v=v^i;\ s=s^i}$.

This iterative process will continue until an arbitrary stop condition is met, e.g.:

$$\|v^i - v^{i-1}\| < \delta \quad (21)$$

4. Least Squares Motion Estimation in the DCT domain

By denoting the DCT of a given signal $A = DCT(a) = \mathbf{C} \cdot a \cdot \mathbf{C}^T$, where:

$$[\mathbf{C}]_{\mathbf{m,i}} \triangleq C(m,i) = \sqrt{\frac{2}{N}}\, \xi(m) \cos\left(\frac{m\left(i+\frac{1}{2}\right)\pi}{N}\right), \quad (22)$$

and $\xi(0) = \sqrt{\frac{1}{2}}$ and $\xi(m) = 1$ for $m > 0$, the described algorithm can be easily applied in the compressed DCT-domain if one takes into account the orthonormal properties of the DCT:

$$\mathbf{C} = \mathbf{C}^*;\ \mathbf{C}.\mathbf{C}^T = \mathbf{C}.\mathbf{C}^{-1} = \mathbf{I} \ \Rightarrow \ \mathbf{C}^{-1} = \mathbf{C}^T \quad (23)$$

In this case, eq. 13 will become:

$$E = R - S \simeq \frac{\partial S}{\partial v_1} dv_1^i + \frac{\partial S}{\partial v_2} dv_2^i \quad (24)$$

$$\Leftrightarrow \mathbf{E} = \mathbf{J_s} \cdot \mathbf{dv^i} \quad (25)$$

$$\Leftrightarrow \mathbf{dv^i} = \left(\mathbf{J_s}^T \cdot \mathbf{J_s}\right)^{-1} \cdot \mathbf{J_s}^T \cdot \mathbf{E} \quad (26)$$

where $\mathbf{E} = DCT(e)$ is a $M \times 1$ vector obtained by rearranging the columns of the difference matrix in the transform domain:

$$\mathbf{E} = \left[\begin{array}{c} \vdots \\ R(k_1,k_2) - S(k_1,k_2) \\ \vdots \end{array} \right]_{M\times 1} . \quad (27)$$

$\mathbf{J_s}$ is a $M \times 2$ matrix whose columns contain the partial derivatives of S disposed in column form:

$$\mathbf{J_s} = \left[\begin{array}{cc} \vdots & \vdots \\ \frac{\partial S}{\partial v_1} & \frac{\partial S}{\partial v_2} \\ \vdots & \vdots \end{array} \right]_{M\times 2} \quad (28)$$

and dv is the desired displacement matrix:

$$dv^i = \left[\begin{array}{c} dv_1^i \\ dv_2^i \end{array} \right]_{2\times 1} . \quad (29)$$

4.1. Computing the image derivative in the DCT domain

Despite the apparent simplicity of the described procedure, the practical interest of the proposed algorithm will greatly depend on the possibility and the easiness of computing the derivatives of S from the input coded image.

Let us consider the displacement of a given image by a small motion vector $v = (v_1, v_2)$. According to the definition of the inverse DCT, $s(y_1, y_2) = s(x_1 + v_1, x_2 + v_2)$ will be given by:

$$s(y_1, y_2) = \sum_{m_1=0}^{N-1} C(m_1, y_1) \sum_{m_2=0}^{N-1} C(m_2, y_2) \cdot S(m_1, m_2) \tag{30}$$

By taking into account the chain rule of the derivatives of $s\,(y_1, y_2)$:

$$\frac{\partial s\,(y_1, y_2)}{\partial v_1} = \frac{\partial s}{\partial y_1} \cdot \frac{\partial y_1}{\partial v_1} = \frac{\partial s}{\partial y_1} \cdot 1 = \frac{\partial s\,(y_1, y_2)}{\partial y_1} \tag{31}$$

one obtain:

$$\frac{\partial s(y_1, y_2)}{\partial y_1} = \sum_{m_1=0}^{N-1} \frac{\partial C(m_1, y_1)}{\partial y_1} \sum_{m_2=0}^{N-1} C(m_2, y_2) \cdot S(m_1, m_2) \tag{32}$$

$$= \sum_{m_1=0}^{N-1} P(m_1, y_1) \sum_{m_2=0}^{N-1} C(m_2, y_2) \cdot S(m_1, m_2). \tag{33}$$

In matrix form:

$$\left[\frac{\partial s(y_1, y_2)}{\partial y_1}\right] = \mathbf{P} \cdot \mathbf{S} \cdot \mathbf{C}^T \tag{34}$$

where

$$\mathbf{P} \triangleq P(m_1, y_1) = \frac{\partial C(m_1, y_1)}{\partial y_1} = \sqrt{\frac{2}{N}}\, \xi\,(y_1) \left(-\frac{m_1 \pi}{N}\right) \sin\left(\frac{m_1 \left(y_1 + \frac{1}{2}\right) \pi}{N}\right). \tag{35}$$

Likewise, it can be shown that

$$\left[\frac{\partial s(y_1, y_2)}{\partial y_2}\right] = \mathbf{C} \cdot \mathbf{S} \cdot \mathbf{P}^T\,. \tag{36}$$

Consequently, in the transform domain, one have:

$$\left[\frac{\partial S}{\partial v_1}\right] = \mathbf{C} \left[\frac{\partial s}{\partial v_1}\right] \mathbf{C}^T = \mathbf{C}\,\mathbf{P}\,\mathbf{S}\,\mathbf{C}^T\,\mathbf{C}^T = \mathbf{D_1}\,\mathbf{S}\,\mathbf{D_2}^T \tag{37}$$

$$\left[\frac{\partial S}{\partial v_2}\right] = \mathbf{C} \left[\frac{\partial s}{\partial v_2}\right] \mathbf{C}^T = \mathbf{C}\,\mathbf{C}\,\mathbf{S}\,\mathbf{P}^T\,\mathbf{C}^T = \mathbf{D_2}\,\mathbf{S}\,\mathbf{D_1}^T \tag{38}$$

where $\mathbf{D_1} = \mathbf{CP}$ and $\mathbf{D_2} = \mathbf{CC}$ are constant $N \times N$ matrices that can be pre-computed and stored in memory.

Step 0: Read the macroblocks of the current (R) and previous (S) images from the coded video sequence:
$R = DCT\,(r\,(x_1, x_2))$
$S = DCT\left(s\left(x_1 + v_1^0, x_2 + v_2^0\right)\right)\big|_{\left(v_1^0, v_2^0\right)=(0,0)}$

Step 1: Compute the prediction error in the DCT domain:
$E = [R - S]_{M\times 1}$, by considering v^i

Step 2: Compute the partial derivatives of S:
$\mathbf{J_s} = \left[\begin{array}{c:c} \mathbf{D_1} \cdot \mathbf{S} \cdot \mathbf{D_2}^T & \mathbf{D_2} \cdot \mathbf{S} \cdot \mathbf{D_1}^T \end{array}\right]_{M\times 2}$

Step 3: Compute the displacement increment $\mathbf{dv^i}$:
$\mathbf{dv}^i = \left[\begin{array}{c} dv_1^i \\ dv_2^i \end{array}\right]_{2\times 1} = \left(\mathbf{J_s}^T \cdot \mathbf{J_s}\right)^{-1} \cdot \mathbf{J_s}^T \cdot \mathbf{E}$

Step 4: Update the motion vector $\mathbf{v}^i$:
$\mathbf{v}^i = \mathbf{v}^{i-1} + \mathbf{dv}^i$

Step 5: Evaluate the stop condition:
If $\|v^i - v^{i-1}\| < \delta$, stops the algorithm and sets $\mathbf{v} = \mathbf{v}^i$;
otherwise, re-compute $S|_{v=v^i}$ and return to **Step 1**.

Figure 1: Iterative LSE algorithm of the motion vectors in the DCT domain.

4.2. Algorithm

As a consequence, the iterative least squares motion estimation algorithm in the DCT domain can be described by the procedure presented in figure 1.

4.3. Scalable Computational Complexity

In most video coding standards, temporal redundancies are exploited by applying a motion compensation algorithm to the disjoint set of macroblocks that compose each frame. In turn, each macroblock is composed by four $N \times N$ pixels luminance blocks, with $N = 8$, thus corresponding to an image area with $256 = (16 \times 16)$ pixels. To exploit the spatial redundancies present in each image, the DCT is applied to each of these (8×8) blocks and the obtained coefficients are quantized and transmitted to the decoding system.

The main advantage of using the DCT is that it concentrates most of the pixels energy in the lower frequency coefficients of the coded block. As a consequence, most high frequency coefficients are set to zero after the quantization step. This fact conducts to significant advantages when the proposed motion estimation algorithm is performed in the DCT domain.

Most of the computational effort of the proposed algorithm is spent in the computation of the partial derivatives matrix j_s. As it was described in section 3, when this algorithm is applied in the pixel domain, all $N \times N$ pixels of the four luminance blocks that compose each macroblock should be taken into account in the iterative estimation procedure. Consequently, the number of considered samples will be $M = 4 \times (8 \times 8) = 256$ and most of the computa-

tional effort is spent in the calculation of the (256×2) sized j_s matrix, which implies the computation of 16 $(N \times N)$ matrix multiplications.

In contrast, when the proposed estimation algorithm is applied in the DCT domain, only the non-null coefficients need to be taken into account, thus leading to a significant smaller number of considered samples M. This fact can even be used to scale the computation effort that should be spent by the algorithm. As it was suggested by Reeves [7], a subset of the $(N \times N)$ DCT coefficients of each block can be selected using a zig-zag pattern similar to the one that is adopted by most coding standards, thus discarding the remaining high frequency coefficients but still keeping the information that is more relevant for the image structure. Moreover, since higher frequencies will contain more noise, discarding those high frequency DCT coefficients may even improve the convergence of the method [7]. Furthermore, to avoid the interference of any eventual difference of the global luminance level of the two considered frames, the DC coefficient of the blocks should not be considered.

5. Application: Motion Re-Estimation in DCT Domain Scaled Video

To evaluate the proposed algorithm, a Least Squares Motion Estimation (LSME) module was incorporated in a H.263 DCT domain video transcoder that performs a $\frac{1}{2} \times \frac{1}{2}$ down scaling operation, as described by Chang [8]. As a consequence, the set of four luminance blocks $b_{i,j}$ $i,j \in \{1,2\}$ that compose each macroblock of the original frame will be represented by a single block $\hat{b}$ in the scaled frame:

$$\hat{b} = h_1 b_{11} w_1 + h_2 b_{21} w_1 + h_1 b_{12} w_2 + h_2 b_{22} w_2 \qquad (39)$$

with

$$h_1 = w_1^T = \begin{bmatrix} \mathbf{f} \\ \emptyset \end{bmatrix} \quad \text{and} \quad h_2 = w_2^T = \begin{bmatrix} \emptyset \\ \mathbf{f} \end{bmatrix}, \qquad (40)$$

where $\emptyset$ is a $(N/2 \times N)$ null matrix and $\mathbf{f}$ is given by:

$$\mathbf{f} = \frac{1}{2} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}_{N/2 \times N} \qquad (41)$$

However, although h_i and w_i matrices are sparce, $H_i = DCT(h_i)$ and $W_i = DCT(w_i)$ are not. Consequently, a different formalization based on the decomposition proposed by Shibata [9] was adopted, which minimizes the number of operations without any loss in the image quality. By denoting $F = f \cdot C^T$, it can be shown [9] that the scaled macroblock may be obtained by concatenating the several partial matrices represented in eq. 42.

$$\hat{B} = C \cdot \begin{bmatrix} F \cdot \left[B_{11} \cdot F^T \vdots B_{12} \cdot F^T \right]_{N,N} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ F \cdot \left[B_{21} \cdot F^T \vdots B_{22} \cdot F^T \right]_{N,N} \end{bmatrix} \cdot C^T \qquad (42)$$

The same scale factor was then applied to the decoded motion vectors v of the original video sequence, in order to compute an initial estimate of the motion vector of each scaled macroblock. Several different approaches have been proposed in the last few years [3] [4]. In particular, it was adopted the Simple Motion Estimation Scaling (SMES) algorithm proposed by Liang [3], which takes into account the spatial activity of the original macroblocks in the computation of the scaled motion vector $\hat{v}$:

$$\hat{v} = \frac{1}{\rho} \cdot \frac{\sum_{i=1}^{I} \sum_{j=1}^{J} v_{i,j} \cdot \alpha_{i,j} \cdot \beta_{i,j}}{\sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_{i,j} \cdot \beta_{i,j}} \qquad (43)$$

where $\alpha_{i,j}$ is the spatial activity of macroblock (i,j), measured by the number of non-null AC coefficients, $\beta_{i,j}$ is the area (in terms of number of pixels) of the original macroblock (i,j) involved in the composition of the transcoded scaled macroblock and ρ is the scale factor. For the considered case, it was assumed that $\rho = 2$, $\beta_k = N^2$ and $I = J = 2$.

The motion vector $\hat{v}$, obtained from eq. 43, was then used as a coarse estimate of the desired motion vector of the scaled macroblock. This estimate, as well as the scaled versions of the current and previous frames resulting from the application of eq. 42, were then applied to the proposed LSME algorithm (fig. 1) to obtain an accurate refinement of the desired motion vector. This algorithm was implemented so that a maximum of 3 iterations are performed to converge to the final motion vector, unless the constraints corresponding to the adopted stopping condition are met: $\|v^i - v^{i-1}\| < \delta$, where $\delta = 0.1$.

6. Experimental Results

To assess the performance of the proposed LSME algorithm, a simpler version of the H.263 video transcoding system for DCT domain video scaling described above was also implemented [5], without the developed least squares motion re-estimation module incorporated in the coding loop. In such simpler transcoder, the scaled motion vectors obtained with eq. 43 were directly considered in the coding algorithm, without any further refinement stage.

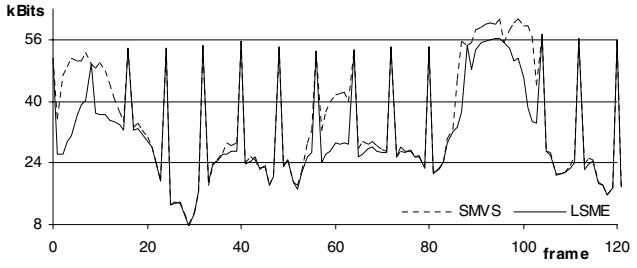
The obtained experimental results are presented in table 1, considering a set of 300 frames of several CIF video sequences characterized by a considerable amount of movement and two distinct quantization steps: Q=4 and Q=8. In

Table 1: Bit-rate and PSNR measures obtained with the considered transcoding systems using the proposed LSME and SMES algorithms.

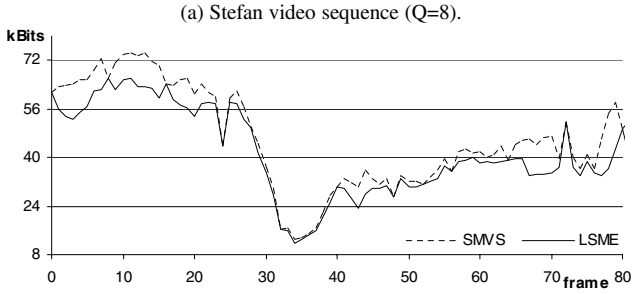
Video Sequence	Q=4		
	PSNR [dB]	ΔPSNR [dB]	ΔBits
Football	37.45	+0.05	-10.52%
Stefan	37.25	+0.00	-6.62%
Carphone	38.58	+0.06	-4.93%
Table-Tennis	37.44	-0.01	-3.08%
Mobile	36.21	+0.04	-0.50%

Video Sequence	Q=8		
	PSNR [dB]	ΔPSNR [dB]	ΔBits
Football	33.22	+0.10	-12.09%
Stefan	31.74	+0.03	-7.37%
Carphone	34.23	+0.07	-5.24%
Table-Tennis	33.29	+0.02	-3.70%
Mobile	30.30	+0.06	-1.31%

this table, it is presented the PSNR quality measures obtained with the simpler transcoding system using the SMES algorithm and the corresponding gains, both in terms of the PSNR and bit-rate, that were obtained with the introduction of the proposed LSME refinement module. To emphasize the direct influence of the proposed LSME module on the output bit-rate, the output buffer controller of the encoding system was also disabled, so that the obtained differences of the amount of bits required to encode each frame can be directly attributed to the influence of the developed motion re-estimation module. As a consequence, the encoded video sequences presented quite similar PSNR measures, since all



(a) Stefan video sequence (Q=8).



(b) Football video sequence (Q=4).

Figure 2: Number of bits required to encode each frame using the proposed LSME and the SMES algorithms.

prediction differences could be encoded with the maximum resolution for the considered quantization steps. The charts presented in figure 2 illustrate the difference between the required number of bits to encode each frame using the two considered motion estimation approaches.

As it can be seen from table 1 and figure 2, a further reduction of the number of bits required to encode each video sequence can be obtained with the proposed LSME algorithm. For the considered test sequences, this reduction was more significant in video sequences with more amount of movement and achieved values up to 12%.

7. Conclusion

A new compressed domain ME algorithm was proposed. This algorithm is based on an iterative scheme that estimates the new MVs by applying a LSE technique. To do so, it only uses an arbitrary subset of non-null DCT coefficients, directly obtained from the H.26x or MPEG-x video stream. The proposed algorithm was then incorporated in a H.263 video transcoder that performs a simple down-scale operation in the DCT domain. The experimental results demonstrated that the obtained MVs provide the means to significantly enhance the quality of the temporal prediction scheme with reductions up to 12% in the obtained bit-rate.

References

[1] H. Sun, T. Chiang and X. Chen, *Digital Video Transcoding for Transmission and Storage*, CRC Press, 2004.

[2] Jie Chen, Ut-Va Koc and K. J. Ray Liu, *Design of Digital Video Coding Systems - A Complete Compressed Domain Approach*, Marcel Dekker, 2002.

[3] Y. Q. Liang, L. P. Chau and Y. P. Tan, "Arbitrary downsizing video transcoding using fast motion vector re-estimation," *IEEE Signal Processing Letters*, vol. 9, no. 11, pp. 352–355, Nov. 2002.

[4] Y.-P. Tan, Y. Liang and H. Sun, "On the methods and performances of rational downsizing video transcoding," *Signal Processing: Image Communication*, vol. 19, pp. 47–65, 2004.

[5] N. Roma and L. Sousa, "Fast transcoding architectures for insertion of non-regular shaped objects in the compressed dct-domain," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 659–683, Sep. 2003.

[6] U.V. Koc and K. Ray Liu, "DCT-based motion estimation," *IEEE Transactions on Image Processing*, vol. 7, no. 7, pp. 948–965, Jul. 1998.

[7] R. Reeves, *Image Matching in the Compressed Domain*, Ph.D. thesis, Queensland Univers. of Techn., Australia, 1999.

[8] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal Selected Areas in Comm.*, vol. 13, no. 1, pp. 1–11, Jan. 1995.

[9] Y. Shibata, Z. Chen and R. H. Campbell, "A fast degradation-free algorithm for DCT block extraction in the compressed domain," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'99)*, Mar. 1999, vol. 6, pp. 3185–3188.