Available online at www.sciencedirect.com
SCIENCE DIRECT®
Signal Processing: *Image Communication* 18 (2003) 659–683
SIGNAL PROCESSING: IMAGE COMMUNICATION
www.elsevier.com/locate/image

Fast transcoding architectures for insertion of non-regular shaped objects in the compressed DCT-domain

Nuno Roma*, Leonel Sousa

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

Abstract

This paper addresses the problem concerned with the insertion of non-regular shaped objects, such as logos or subtitles, in compressed video signals. To achieve this objective, a different approach from the usual pixel-domain compositing operation is adopted, in order to avoid the presence of undesired semi-transparent rectangular regions around the inserted objects. The transposition of this technique to the compressed DCT-domain is presented, as well as the integration of the logo insertion module in several architectures of compressed domain video transcoders. A detailed study of the main characteristics of the proposed DCT-domain transcoders, with a special emphasis on the required computational load and on the obtained coding quality (PSNR) of the processed video sequences shows that DCT-domain insertion architectures can offer significant advantages, both in terms of subjective quality and computational cost. Moreover, the distinctive features presented by the proposed architectures provide the means to adapt the insertion scheme to the particular requirements of the target application.

Keywords: Video Transcoding; Logo Insertion; Subtitling; DCT manipulation

1. Introduction

Recently, there has been a general proliferation of advanced video services and multimedia applications, where video compression standards, such as MPEG-x or H.26x, have been developed to store and broadcast video information in digital form. However, once video signals have been compressed, delivery systems and operators frequently face the need for further manipulation and processing on such compressed bit-streams. Consequently, *video transcoding* has recently emerged as a new research area concerning a set of manipulation and adaptation techniques such as space scaling, frame skipping, bit-rate adjustment, etc. One other important issue that has also emerged is concerned with intellectual property management and protection. Actually, the insertion of visible objects, such as logos and open subtitles (henceforward simply designated by "*logos*") in the compressed domain has faced a growing interest by broadcasting television networks to provide the possibility of inserting their own logos and subtitles in pre-encoded video streams [8]. To take account for the several video standards currently in use, not only should this mechanism provide the insertion capability in MPEG block-based coding standards [10,13]

*Corresponding author. Fax: +351-21-3145843.
E-mail addresses: nuno.roma@inesc-id.pt (N. Roma), las@inesc-id.pt (L. Sousa).

0923-5965/03/$ - see front matter
doi:10.1016/S0923-5965(03)00058-4

(including the simplest MPEG-1 and MPEG-2), but it should also be compatible with other real-time services based on H.261 and H.263 video standards.

Recent developments on video transcoders have shown that significant advantages concerning the computational complexity of the algorithms can be achieved by fully operating in the frequency domain [1]. In fact, not only does it avoid the implementation of both the forward discrete cosine transform (DCT) and its inverse (IDCT), but it also takes advantage of the presence of a large number of null quantized DCT coefficients to heavily reduce the data manipulation rate [6,18].

Up until now, most insertion algorithms were based on the compositing operation proposed by Porter [2,14]. However, despite its simplicity, when a non-regular shaped object (NRSO) (such as a letter or a logo) is inserted, it gives rise to an undesired semi-transparent rectangular region around the object, corresponding to the area that is actually processed by the insertion algorithm. In this paper, a different insertion technique is proposed, by restricting the Porter's algorithm [14] to the logo area. However, the application of this technique in the compressed DCT-domain will require the usage of the multiplication-convolution relationship for the DCT, since the simple linear combination of Porter's algorithm will not be applicable any more [2].

Having proposed the NRSO insertion algorithms in both the pixel-domain and in the compressed DCT-domain, several transcoding architectures will be derived, offering different characteristics in what concerns the obtained video quality (PSNR) and the computational load required to perform the insertion algorithm.

This paper is organized as follows: in Section 2 the insertion algorithm in the pixel-domain will be described. The transposition of this algorithm to the compressed DCT-domain will be described in Section 3. In Section 4 several architectures of both pixel-domain transcoders and DCT-domain transcoders will be presented. The experimental results obtained with the proposed architectures will be presented in Sections 5 and 6 will conclude the presentation.

2. Insertion of irregular shaped objects in the pixel domain

Object insertion in the pixel domain can be performed by combining the pixels of the background image $b(n_1, n_2)$ with the object $\ell(n_1, n_2)$ to obtain the output image $f(n_1, n_2)$. This operation is usually expressed as a linear combination of the form [2,14]:

$$f(n_1, n_2) = \alpha \cdot \ell(n_1, n_2) + (1 - \alpha) \cdot (n_1, n_2), \tag{1}$$

where the α factor determines the transparency level of the object. In particular, when $\alpha = 1$ all pixels of the background image are replaced by the object, giving rise to an opaque overlapping of the object over the input image.

The main disadvantage of this method is emphasized when NRSOs are inserted. In fact, since most video coding algorithms perform their processing on blocks with $N \times N$ pixels (usually $N = 8$), the insertion of objects whose shape and position do not coincide with the defined block geometry and grid implies the extension of the original object area to an integer multiple of $N \times N$ blocks. This extension is usually performed by defining a transparency color (T), whose value is assigned to all pixels of this set of blocks that do not belong to the original NRSO. Three different regions in the output image usually arise from this extension:

- the pixels corresponding to the original object, where:
  $f_1(n_1, n_2) = \alpha \cdot \ell(n_1, n_2) + (1 - \alpha) \cdot b(n_1, n_2)$;
- the transparent pixels of the extended blocks that do not belong to the original object, where:
  $f_2(n_1, n_2) = \alpha \cdot T + (1 - \alpha) \cdot b(n_1, n_2)$;
- the blocks that do not contain any pixel of the object to be inserted, where:
  $f_3(n_1, n_2) = b(n_1, n_2)$.

Independently of the considered value for T, this scheme gives rise to an undesired semi-transparent rectangular region around the object, where $f(n_1, n_2) = f_2(n_1, n_2) \neq b(n_1, n_2)$. An example of this phenomenon is illustrated in Fig. 1(a),

where Eq. (1) has been applied to insert the object using $\alpha = 0.5$.

This undesired semi-transparent region can be avoided if Eq. (1) is restricted to the pixels of the original object [15,16]. However, this will imply the usage of segmentation techniques to isolate the pixels corresponding to the original object from the rest of the pixels of the block. Even so, the previously described formalism for the linear combination can still be applied if the concept of transparency mask is introduced and defined as:

$$m(n_1,n_2) = \begin{cases} 1, & (n_1,n_2) \in \text{NRSO}, \\ 0, & (n_1,n_2) \notin \text{NRSO}. \end{cases} \quad (2)$$

Hence, Eq. (1) becomes

$$f(n_1,n_2) = [\alpha \cdot m(n_1,n_2)] \odot \ell(n_1,n_2) + [1 - \alpha \cdot m(n_1,n_2)] \odot b(n_1,n_2) \quad (3)$$

$$= \phi(n_1,n_2) + \psi(n_1,n_2) \odot b(n_1,n_2), \quad (4)$$

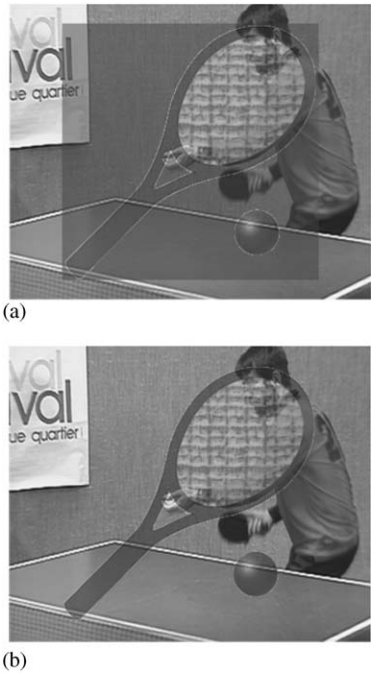

Fig. 1. Pixel domain insertion of an NRSO ($\alpha = 0.5$): (a) Porter's technique; (b) proposal technique.

where $\odot$ denotes pixel-wise multiplication. In this equation, $\phi(n_1,n_2)$ and $\psi(n_1,n_2)$ represent constant matrices that are solely dependent on the considered object. Consequently, they can be precomputed and stored in memory. In Fig. 1(b), it is illustrated the result of this technique using $\alpha = 0.5$. As it can be seen, the undesired semi-transparent region is not present around the NRSO any more.

3. Insertion of objects in the compressed DCT domain

The pixel domain insertion algorithm presented in Eq. (1) can be directly applied in the compressed domain by using the orthogonality properties of the DCT. Since α and $(1-\alpha)$ are scalars and the DCT is linear:

$$F(k_1,k_2) = \alpha \cdot L(k_1,k_2) + (1-\alpha) \cdot B(k_1,k_2), \quad (5)$$

where $X(k_1,k_2) = DCT[x(n_1,n_2)]$.

However, since most video coding algorithms perform the DCT computation on $N \times N$ pixel blocks, this relation cannot be used if the undesired semi-transparent region around the NRSO is intended to be avoided [15,16]. In fact, since $[\alpha \cdot m(n_1,n_2)]$ and $[1 - \alpha \cdot m(n_1,n_2)]$ of Eq. (3) are not scalars, the pixel-wise multiplications will have to be replaced by convolutions in the DCT domain. Consequently, the application of Eq. (3) in the compressed DCT-domain is stated as follows:

$$F(k_1,k_2) = \Phi(k_1,k_2) + \Psi(k_1,k_2) \circledast B(k_1,k_2), \quad (6)$$

where $F(k_1,k_2) = \text{DCT}[f(n_1,n_2)]$, $\Phi(k_1,k_2) = \text{DCT}[\phi(n_1,n_2)]$, $\Psi(k_1,k_2) = \text{DCT}[\psi(n_1,n_2)]$ and $B(k_1,k_2) = \text{DCT}[b(n_1,n_2)]$. However, although the DCT is closely related to the discrete Fourier transform (DFT), a complete and consistent formalization of the multiplication-convolution property for the DCT was only presented relatively recently [2,7]. This property is based on a symmetric convolution operation over $(2N \times 2N)$ extended sequences. For the particular case of the DCT-II transform, these $(2N \times 2N)$ extended sequences should have a *whole-sample anti-symmetry* (WSWA) (as described in [7]) and are given

by Eq. (7). Hence, the above 2D convolution should be computed as follows:

$$\tilde{X}(k_1,k_2)=\begin{cases}0, & k_1=0 \text{ or } k_2=0,\\ \hat{X}(N-k_1,N-k_2), & k_1=1\ldots(N-1),\ k_2=1\ldots(N-1),\\ \hat{X}(k_1-N,N-k_2), & k_1=N\ldots(2N-1),\ k_2=1\ldots(N-1),\\ \hat{X}(N-k_1,k_2-N), & k_1=1\ldots(N-1),\ k_2=N\ldots(2N-1),\\ \hat{X}(k_1-N,k_2-N), & k_1,\ k_2=N\ldots(2N-1).\end{cases} \tag{7}$$

$$\Psi(k_1,k_2)\circledast B(k_1,k_2) = W_{N\times N}(\tilde{\Psi}(k_1,k_2)\,\textcircled{S}\,\tilde{B}(k_1,k_2)), \tag{8}$$

where $\tilde{\Psi}(k_1,k_2)$ and $\tilde{B}(k_1,k_2)$ are WSWA ($2N\times 2N$) symmetric extensions of $\Psi(k_1,k_2)$ and $B(k_1,k_2)$ as described in Eq. (7), with $\hat{X}(k_1,k_2)=X(k_1,k_2)/(C(k_1)C(k_2))$ and

$$C(k)=\begin{cases}1/\sqrt{2}, & k=0,\\ 1, & k=1\ldots(N-1).\end{cases} \tag{9}$$

$$\Psi(k_1,k_2)\,\textcircled{S}\,B(k_1,k_2)=\frac{1}{2N}C(k_1)C(k_2)\left[\sum_{m_1=0}^{2N-1}\sum_{m_2=0}^{2N-1}\tilde{\Psi}(m_1,m_2)\tilde{B}[\mathrm{mod}_{2N}(k_1-m_1),\mathrm{mod}_{2N}(k_2-m_2)]\cdot S(k_1-m_1)\cdot S(k_2-m_2)\right]. \tag{10}$$

The symbol Ⓢ denotes the skew-circular convolution and is defined by Eq. (10), where

$$S(k)=\begin{cases}1, & k\in[0,(2N-1)],\\ -1, & \text{otherwise},\end{cases} \tag{11}$$

and $W_{N\times N}(k_1,k_2)$ is a $N\times N$ rectangular window used to extract the representative samples out of the base period of the result of the convolution.

This operation requires a significant amount of multiplications and sums ($\mathcal{O}(N^4)$). Recently, Shen et al. [19] proposed a different approach to compute the DCT-domain convolution by exploiting the symmetry and the orthogonality properties of the DCT to reduce the overall complexity of this operation. With such algorithm, it is possible to reduce the total amount of operations to only about 25% [19].

In the following pictures, it is illustrated the application of the proposed insertion method in the compressed DCT-domain to insert two NRSOs, corresponding to the logo presented in Fig. 2(a) and the subtitle shown in Fig. 2(b) in the CIF *carphone* video sequence. The frame presented in Fig. 3(a) was obtained using a transparency factor $\alpha=0.5$. As it can be seen, it is still possible to perceive the presence of some details of the background image in the area corresponding to the inserted object. Fig. 3(b) presents the same frame using a transparency factor $\alpha=1.0$. Contrasting with the previous setup, in this case the insertion is 100% opaque. The corresponding transparency mask $m(n_1,n_2)$, which was applied to the whole image, is presented in Fig. 3(c).

4. Transcoding architectures for insertion of non-regular shaped objects

In this section, a set of several different architectures of both pixel-domain and compressed-domain

(a) (b)

Fig. 2. Considered set of NRSOs ($T=0$); (a) Logo; (b) subtitle.

transcoders for insertion of NRSOs in compressed video sequences will be presented. In the following, the considered primordial objective was the optimization of the tradeoff between the obtained *image quality* (PSNR) and the involved *computational cost* (number of sums and multiplications).

Fig. 3. Object insertion in the compressed DCT-domain using the CIF *carphone* video sequence and the transparency mask of (c): (a) $\alpha = 0.5$; (b) $\alpha = 1$ and (c) transparency mask, $m(n_1, n_2)$.

4.1. Pixel-domain transcoder with re-estimation of motion vectors

The most straightforward architecture of a pixel-domain transcoder for object insertion is illustrated in Fig. 4. This decoder–encoder cascaded pair is usually regarded as the most trivial architecture, since it comprises one full decoder followed by one full encoder. The input bit stream is first fully decoded and then re-encoded after the application of the pixel-domain insertion algorithm, based on the compositing scheme described in Eq. (3).

Despite its simplicity, the main disadvantage of this architecture is concerned with its considerably high computational cost. Since the insertion algorithm is performed in the pixel-domain and there is no re-usage of the original coding parameters, a significant amount of operations is required to perform both the direct and inverse DCT transforms, as well as the full-search block-matching motion estimation algorithm at the encoder side of the system.

4.2. Pixel-domain transcoder without re-estimation of motion vectors

Considering that motion estimation is undoubtedly the most computational demanding block of the video coding algorithm, an usual approach to significantly reduce the complexity level of transcoders is to remove the motion estimation function from the encoder side, so that the MVs of the

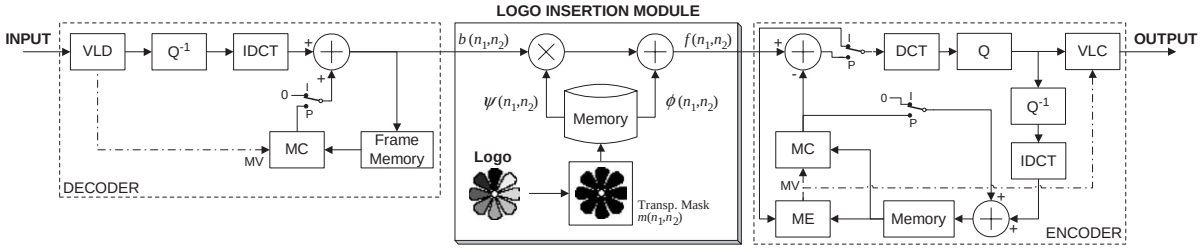

Fig. 4. Block diagram of the pixel-domain transcoder with re-estimation of motion vectors.

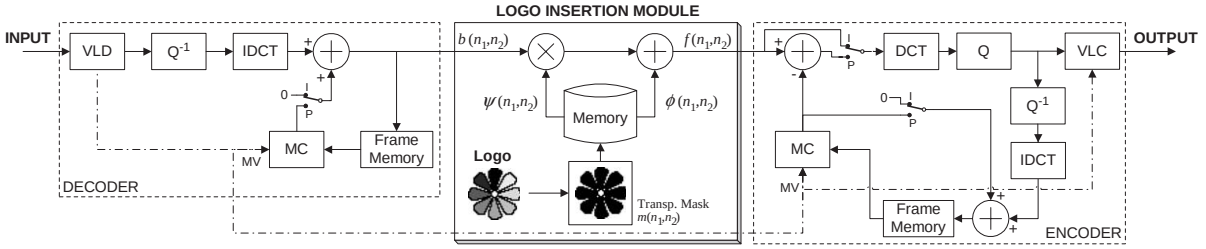

Fig. 5. Block diagram of the pixel-domain transcoder without the motion estimation function.

incoming bit stream are re-used, instead of calculating new ones. In this case, the architecture of the pixel-domain transcoder is the one illustrated in Fig. 5.

Even so, this architecture still demands a significant amount of operations, since the computation of both the forward and the inverse DCTs are still required by this scheme. This is, in fact, the main motivation for the recent proposals of frequency domain video transcoders [1,5,17,21].

4.3. Compressed DCT-domain transcoder

The most straightforward approach to implement the proposed transcoder in the compressed domain is to perform the compositing operation of Eq. (3) in the DCT-domain, as it was described in Eq. (6). With such a scheme, it is possible to fully operate in the frequency domain, thus avoiding the implementation of both the forward DCT and its inverse [1].

However, this will imply that certain operations, such as motion compensation (MC), will also have to be performed in the frequency domain. As it was shown by Chang and Messerschmitt [2] and by Merhav and Bhaskaran [9], the DCT transform of the prediction block $\hat{X}$ can be obtained from the DCT coefficients of the corresponding adjacent blocks $\hat{X}_i$ by multiplying each of these blocks by appropriate matrices (H_{i1} and H_{i2}), that perform the separation and the displacement of the required regions according to the following equation (see Fig. 6) [2,9]:

$$\hat{X} = \sum_{i=1}^{4} H_{i1}\hat{X}_i H_{i2}, \quad H_{ij} = \mathrm{DCT}(h_{ij}). \tag{12}$$

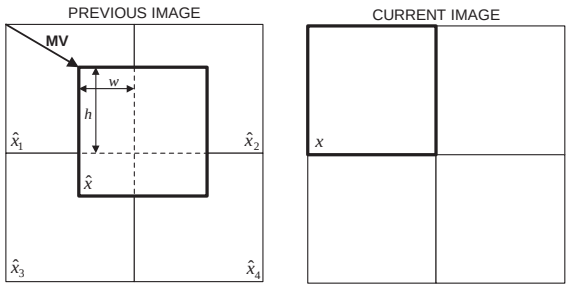

Fig. 6. Motion compensation procedure.

The set of matrices h_{i1} and h_{i2} are defined by the number of lines and columns of the area intersected by each block $\hat{x}_i$ with the prediction block $\hat{x}$ (see Fig. 6). Their structure is similar to the upper and lower triangular matrices u_h and l_w, where I_h and I_w are $h \times h$ and $w \times w$ identity matrixes, respectively:

$$h_{11} = h_{21} = u_h \triangleq \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix}, \tag{13}$$

$$h_{12} = h_{32} = \ell_w \triangleq \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix}. \tag{14}$$

Similarly,

$$h_{31} = h_{41} = \ell_{N-h}, \quad h_{22} = h_{42} = u_{N-w}. \tag{15}$$

This operation presents a computational complexity proportional to $\mathcal{O}(N^3)$, which is significantly higher than its pixel-domain counterpart, since, in the general case, four neighbor blocks will have to be processed in order to obtain the prediction of the current block. Even so, several efficient algorithms have recently been proposed to

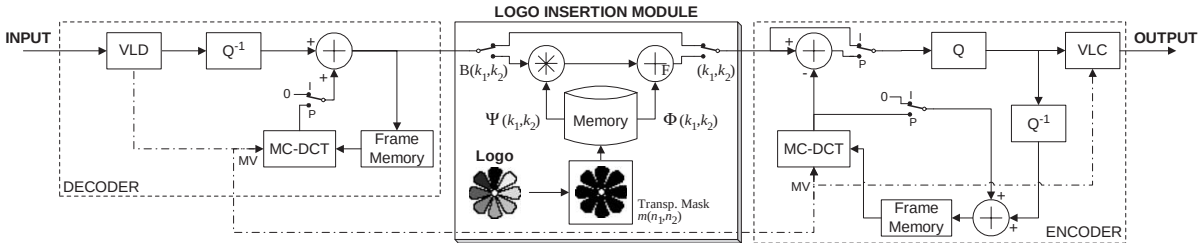

Fig. 7. Block diagram of the compressed DCT-domain transcoder.

reduce the computational load of this DCT-domain motion compensation operation [4,6]. Their main advantages are obtained by exploiting the sparseness property of the DCT representation of the several blocks as well as the spatial continuity between adjacent blocks. Song and Yeo also proposed a very efficient scheme where a significant speedup is achieved by carefully rearranging the computation steps across correlated target blocks and by taking advantage of the shared information in the prediction of multiple neighboring blocks [20].

The block diagram of the implemented DCT-domain transcoder is presented in Fig. 7. As in the previous pixel-domain transcoder, the motion vectors (MVs) decoded from the incoming bit stream were re-used. The insertion algorithm is performed in a block-by-block basis: for each block of the input image, it convolves its DCT transform with $\Psi(k_1,k_2)$, corresponding to the transparency mask, using the 2D-symmetric convolution in the DCT domain and sums the result with the data corresponding to the NRSO $\Phi(k_1,k_2)$ (see Eq. (6)). Since $\Psi(k_1,k_2)$ and $\Phi(k_1,k_2)$ only depend on the object that is being inserted, they can be pre-computed and stored in memory, thus leading to a significant reduction of the required number of operations (see Fig. 7).

4.4. Computational-reduced compressed DCT-domain transcoder

From the analysis of the previously described compressed DCT-domain transcoder, one can easily realize that the proposed insertion algorithm in the transform domain still involves a significant amount of operations (even when re-usage of motion vectors is considered). The functional blocks that are the most responsible for this computational cost are:

- *DCT domain motion compensation*, with a complexity level proportional to $\mathcal{O}(N^3)$;
- *2-D symmetric convolution*, with a complexity level proportional to $\mathcal{O}(N^4)$.

This contrasts with the computational load required by the traditional pixel-domain transcoder (without re-estimation of the MVs), whose most computationally demanding functional blocks (the direct and the inverse DCTs) present a computational complexity proportional to $\mathcal{O}(N^3)$. This substantial extra amount of computations incited the research for more efficient architectures.

Meanwhile, to clarify the presentation of this and other architectures proposed in this paper, the following nomenclature will be adopted:

I_n – INTRA type image;
P_n – INTER type image;
$\bar{I}_n$ – INTRA type image after the insertion of the NRSO;
$\bar{P}_n$ – INTER type image after the insertion of the NRSO.

The sequencial number of each image was appended to its representation as a subscript index (e.g.: $I_nP_{n+1}P_{n+2}\ldots P_{n+M-1}I_{n+M}P_{n+M+1}\ldots$), where M is the adopted number of frames in each Group of Pictures (GOP).

The architecture presented in this section tries to achieve a reduction of the computational load by eliminating the DCT motion-compensation operation from the processing of certain macroblocks (MBs) in INTER type images. This can be

achieved by applying the insertion algorithm directly on the transformed-differences signal (E_t), available in the input bit-stream, instead of using the transformed-pixels (P_t), obtained from the application of the MC operation. Considering the general expression used to decode INTER type images at the decoder end of the video coding system:

$$p_t = \mathrm{MC}(p_{t-1}) + e_t, \tag{16}$$

which is stated in the DCT-domain space as:

$$P_t = \mathrm{MC}(P_{t-1}) + E_t, \tag{17}$$

the objective of this architecture is to process INTER type images using only the information corresponding to the current frame that is received from the communication channel: the differences signal, the motion vectors, the quantization levels, etc., with a special emphasis on the received differences signal (E_t). Consequently, the desired expression for the decoding operation of INTER type images, to be performed by the ultimate decoder of the whole video communication system, should have the form:

$$\bar{P}_t = \mathrm{MC}(\bar{P}_{t-1}) + E_t^*, \tag{18}$$

where $\mathrm{MC}(\bar{P}_{t-1})$ is obtained by applying the motion compensation algorithm to the previous decoded image (with the NRSO already inserted on it). The operand E_t^* represents the differences signal after the application of the insertion algorithm to the original differences frame: $E_t^* = f(E_t)$.

Considering a fraction of a given video sequence composed by only two frames ($\ldots i_{t-1} p_t \ldots$) and an NRSO $\ell(n_1, n_2)$, the insertion algorithm described in Eq. (1) for a given α is as follows:

$$\bar{i}_{t-1} = \alpha\ell + (1-\alpha) i_{t-1} \tag{19}$$

$$= i_{t-1} + \alpha(\ell - i_{t-1}) \tag{20}$$

$$= i_{t-1} + r_{t-1}, \tag{21}$$

$$\bar{p}_t = \alpha\ell + (1-\alpha) p_t \tag{22}$$

$$= p_t + \alpha(\ell - p_t) \tag{23}$$

$$= p_t + r_t, \tag{24}$$

where $r_{t-1} = \alpha(\ell - i_{t-1})$ and $r_t = \alpha(\ell - p_t)$. According to Eqs. (16) and (24), it follows that:

$$\bar{p}_t = \mathrm{MC}(p_{t-1}) + e_t + r_t \tag{25}$$

$$= \mathrm{MC}(\bar{p}_{t-1} - r_{t-1}) + e_t + r_t \tag{26}$$

$$= \mathrm{MC}(\bar{p}_{t-1}) + e_t - \mathrm{MC}(r_{t-1}) + r_t \tag{27}$$

$$= \mathrm{MC}(\bar{p}_{t-1}) + e_t^*, \tag{28}$$

where

$$e_t^* = e_t - \mathrm{MC}(r_{t-1}) + r_t. \tag{29}$$

In the compressed DCT-domain, Eqs. (21) and (28) are stated as

$$\bar{I}_{t-1} = I_{t-1} + R_{t-1}, \tag{30}$$

$$\bar{P}_t = \mathrm{MC}(\bar{P}_{t-1}) + E_t^*, \tag{31}$$

where

$$E_t^* = E_t - \mathrm{MC}(R_{t-1}) + R_t \tag{32}$$

and

$$R_t = \alpha(k_1, k_2) \circledast [L(k_1, k_2) - P_t(k_1, k_2)]. \tag{33}$$

From the previous description, one realizes that Eq. (31) corresponds to the desired expression for the DCT-domain insertion algorithm, presented in Eq. (18). In fact, it can be shown, from Eq. (32), that it can be decomposed into two distinct operations:

1. Insertion of the data corresponding to the fraction of the NRSO that should be placed in the current position: $+R_t$;
2. Removal of the data corresponding to the fraction of the NRSO that was inserted in the previously processed image. According to the coding algorithm, it is necessary to take into account the possible displacement between the current MB and its prediction MB: $-\mathrm{MC}(R_{t-1})$.

The execution of each of these operations will depend on the particular MB under processing (i.e. if the NRSO should be inserted or not) and on the corresponding prediction MB (i.e. if the previous inserted NRSO should be removed or not).

The block diagram of the frequency-domain transcoder using this insertion algorithm is presented in Fig. 8. As it can be seen, in INTER type images the NRSO data is inserted in the

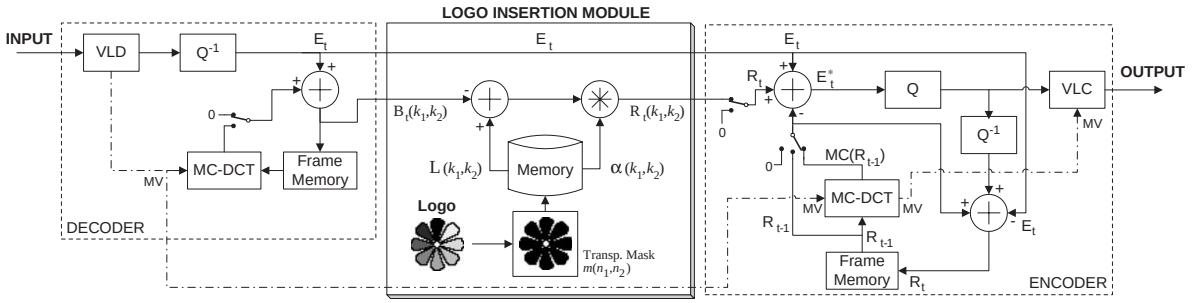

Fig. 8. Block diagram of the computational-reduced compressed DCT-domain transcoder with an object insertion module.

differences signal (E_t). This contrasts with the previous architecture, where the NRSO was inserted in the decoded image (P_t). Consequently, a significant reduction of the computational load can now be achieved in the encoder part of the transcoder, since while in the previous architecture the MC operation was applied to the entire frame, in this architecture it is only performed in the area covered by the NRSO. Therefore, the degree of this reduction will depend on the particular characteristics of the NRSO that is being inserted, as well as on the video sequence that is being processed.

Despite the described computational saving strategy, one realizes that this architecture still requires a significant amount of computations. Namely, in the DCT-domain motion compensation operation that is performed over the entire image at the decoder part, in the 2D symmetric convolution that is performed over the area covered by the NRSO at the insertion module and in the DCT-domain motion compensation operation that is performed over the area covered by the NRSO in the encoder side of the transcoding system.

4.5. Open-loop compressed DCT-domain transcoder

From the description of the previously presented compressed-domain insertion algorithms, one realizes that the corresponding architectures still require a significant amount of computations. This high computational load led to the development of more ingenious schemes that take into account some information concerning to the GOP structure of the coded video under processing and are somewhat more permissive to the introduction of a certain distortion level in the area of the image corresponding to the insertion. In fact, if one takes into account that NRSOs (such as logos or subtitles) usually do not change with time (or only change between a significant amount of frames), one can naturally consider the NRSOs as being constant entities. Consequently, one can easily raise the question about the need for inserting them in all coded frames. As an example, INTER type images are coded with temporal prediction schemes, where the DCT is applied to the difference between the current image and the image obtained from the application of the MC mechanism to the previously decoded frame. Considering that logos (or other types of NRSOs) do not change with time, it would not be difficult to accept that no information concerning them would be present in the differences signal corresponding to the coding of INTER type images. Consequently, only INTRA type images would have to be processed by the insertion algorithm.

Unfortunately, this assumption is not completely true, due to the MC mechanism and to the fact that the considered algorithm deals with semi-transparent type NRSOs. As an example, to obtain the prediction of a given MB in a INTER type image, it is necessary to conveniently displace the previous image MBs according to the corresponding MVs. If one considers the simplest case where one of the MBs corresponding to NRSO has a non-null motion vector, its prediction may not contain the NRSO, or even if it does, it will be

probably not placed in the correct position. Therefore, some important issues will have to be considered in order to solve these problems.

From the previous sections, one realizes that the implementation of the insertion algorithm in the compressed DCT-domain requires one 2D-symmetric convolution ($\mathcal{O}(N^4)$) and one or more DCT-domain motion-compensation operations ($\mathcal{O}(N^3)$). The proposed approach considers the possibility of tolerating a certain amount of distortion in order to decrease this required computational load. To achieve such objective, it assumes a rather simplified architecture for the insertion module, which is based on the two following principles:

1. NRSOs are solely inserted in INTRA type images;
2. Only the differences signal (E_t) is processed in INTER type images.

As it was already stated, to comply the insertion procedure with the existing video coding standards, the general expression for the decoding operation of INTER type images should have the form:

$$\hat{\tilde{P}}_t = \mathrm{MC}(\hat{\tilde{P}}_{t-1}) + E_t^*. \tag{34}$$

In this and in the following expressions, it was adopted the $\hat{X}$ nomenclature to represent signals where a certain level of degradation or distortion has been tolerated as a result of the application of the simplified insertion algorithm.

According to the first principle previously enunciated, the information that is considered in the processing of all $(M-1)$ INTER type images of a given GOP refers only to the previously processed INTRA frame:

$$\bar{i}_t = \alpha\ell + (1-\alpha)i_t = i_t + \alpha(\ell - i_t) \tag{35}$$

$$= i_t + r_t, \tag{36}$$

where $r_t = \alpha(\ell - i_t)$. Consequently, this r_t factor should be made constant within a given GOP. As an example, in the processing of image p_u, the same r_t factor should also be used, where $u - t \leqslant M$. Hence, instead of having

$$\bar{p}_u = \alpha\ell + (1-\alpha)p_u \tag{37}$$

$$= p_u + \alpha(\ell - p_u) \tag{38}$$

one will have:

$$\hat{\bar{p}}_u = p_u + \alpha(\ell - i_t) \tag{39}$$

$$= p_u + r_t. \tag{40}$$

Eqs. (37) and (39) can be used to estimate the amount of distortion (ε_t) introduced by this simplified scheme. In fact, considering that $r_t = \alpha(\ell - i_t)$:

$$\hat{\bar{p}}_u = p_u + \alpha(\ell - i_t) \tag{41}$$

$$= \alpha\ell + p_u - \alpha i_t \tag{42}$$

$$= \alpha\ell + (1-\alpha)p_u + \alpha(p_u - i_t) \tag{43}$$

$$= \alpha\ell + (1-\alpha)p_u + \varepsilon_t, \tag{44}$$

where $\varepsilon_t = \alpha(p_u - i_t)$. From Eq. (40), the processing of $\hat{\bar{p}}_u$ will be as follows:

$$\hat{\bar{p}}_u = p_u + r_t \tag{45}$$

$$= \mathrm{MC}(p_{u-1}) + e_u + r_t \tag{46}$$

$$= \mathrm{MC}(\hat{\bar{p}}_{u-1} - r_t) + e_u + r_t \tag{47}$$

$$= \mathrm{MC}(\hat{\bar{p}}_{u-1}) + e_u - \mathrm{MC}(r_t) + r_t \tag{48}$$

$$= \mathrm{MC}(\hat{\bar{p}}_{u-1}) + e_u^*. \tag{49}$$

Hence, in the compressed DCT-domain, Eqs. (36) and (49) are stated as:

$$\bar{I}_t = I_t + R_t, \tag{50}$$

$$\hat{\tilde{P}}_u = \mathrm{MC}(\hat{\tilde{P}}_{u-1}) + E_u^*. \tag{51}$$

Once more, Eq. (51) is entirely similar to Eq. (34), representing the decoding operation of INTER type images at the decoder end of the video coding system. E_u^* represents the processed differences signal and is obtained from the application of the insertion algorithm to the original differences frame:

$$E_u^* = E_u - \mathrm{MC}(R_t) + R_t. \tag{52}$$

In Fig. 9, it is illustrated the block diagram corresponding to this processing scheme. The main advantages of this method are concerned with the reduced computational load that is required to insert the NRSOs in the input sequence. Not only are the required operations performed solely in the area that is affected by the NRSO insertion

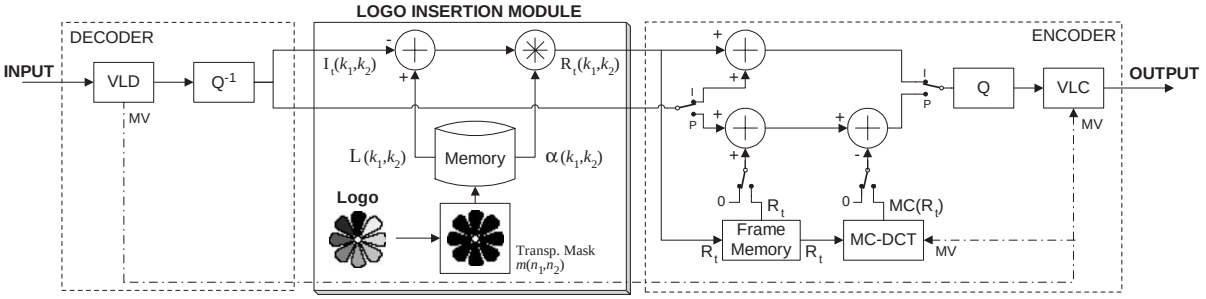

Fig. 9. Block diagram of the open-loop compressed DCT-domain transcoder.

(which is usually much smaller than the whole image area) but the amount of operations required to process each image block is significant lower: while the processed blocks in INTRA type images only require a 2D-symmetric convolution, the blocks that need any processing in INTER type images require one addition and, when the current $MV \neq 0$, a compressed DCT-domain motion-compensation operation to remove the fraction of the NRSO that was inserted during the processing of the previously coded image.

The required amount of memory was also significantly reduced: while in the presented scheme it is only necessary to store the data corresponding to R_t, in the closed-loop compressed DCT-domain transcoders, presented in Sections 4.3 and 4.4, two image memories were required by the insertion algorithm. It is still worth noting that, in this algorithm, this data is only updated when a new INTRA type image is processed.

Contrasting with the transcoder architectures presented in the previous sections, the transparency factor (α) plays, in this architecture, a somewhat different role. More than the transparency degree of the inserted NRSO, it is also of great influence on the possibility of performing the inverse operation (removal of the NRSO that was inserted in the previous processed image), required in some blocks of INTER type images when $MV_n \neq 0$. In fact, with the proposed insertion scheme, the output pixels in the processed area are obtained by a linear combination of the NRSO data (weighted by the α factor) and of the background data (weighted by the $(1-\alpha)$ factor). In the limit situation, when $\alpha = 1$, the result of this linear combination is only composed by the pixels of the NRSO, making it impossible to perform the inverse operation and recover the background image. Moreover, one has already noted that the introduced distortion $\varepsilon_t = \alpha(p_u - i_t)$ increases significantly with α.

5. Experimental results

Several experiments were carried out to evaluate the performance of the proposed NRSO insertion transcoders, both in the pixel-domain and in the DCT-domain. These experiments were performed using four standard video sequences (QCIF format), belonging to classes A, B and C of the MPEG-4 Video Verification Model [11], characterized by different spatial detail and amount of movement (see Fig. 10):

- *Akiyo* (Class A), with low spatial detail and low amount of movement;
- *Silent Voice* (Class B), with medium spatial detail and medium amount of movement;
- *Carphone* (Class C), with high spatial detail and moderate amount of movement, consisting in local displacements of the head and lips of the person in the scene and regular translational movements of the background (car window);
- *Table-tennis* (Class C), with moderate spatial detail and with a significant amount of movement, both translational type (ball) and zoom-out type (video camera).

The first 130 frames of each video sequence were coded using a GOP length $M = 15$ frames and

Fig. 10. Considered video sequences: (a) Akiyo; (b) Silent-Voice (c) Carphone and (d) Table-tennis.

considering two different quantization setups: full-precision ($Q = 0$) and a real quantizer with $Q = 15$ (step size = 30). The logo and subtitle illustrated in Figs. 2(a) and (b) were inserted in each of these video sequences using a transparency factor $\alpha = 0.5$, at positions (5,5) and (120,26), respectively. With this setup, 60 out of the total amount of blocks in each QCIF frame (396 blocks) require some processing, which corresponds to 15.15% of the total amount of $N \times N$ blocks ($N = 8$).

The following characteristics of the considered transcoders were evaluated:

- The overall efficiency of each transcoder and the computational load involved in the insertion of the NRSOs in each video sequence, namely, the number of additions and multiplications;
- The quality of the video sequences after the insertion of the NRSOs;
- The main degradation effects introduced by the several transcoders.

To ease the representation of the results in the several charts presented in the following subsections, it was adopted the following nomenclature to identify the considered transcoding architectures:

`PX_MV`	pixel-domain transcoder with re-estimation of motion vectors;
`PX_nMV`	pixel-domain transcoder without re-estimation of motion vectors;
`DCT_CL`	closed-loop compressed DCT-domain transcoder
`DCT_CL_fast`	computational-reduced compressed DCT-domain transcoder
`DCT_OL`	open-loop compressed DCT-domain transcoder.

5.1. Computational efficiency of the NRSO insertion transcoders

The computational load required by the proposed architectures to insert the considered logo and subtitle in the video sequences is illustrated in Figs. 11–18, both for $Q = 0$ and $Q = 15$. To accommodate the high dynamic range of the presented values, these charts were represented using a semi-logarithmic set of axis, with a scaling factor of 10^6 in the *yy* axis.

As it can be seen, these charts present a pseudo-periodic characteristic, corresponding to the processing of the several GOPs, with distinctive maximums/minimums corresponding to each INTRA type frame. This fact is easily justified by the different processing requirements of INTRA and INTER type images (e.g.: motion estimation/compensation, convolution, etc.).

The pixel-domain transcoder with re-estimation of the motion vectors (`PX_MV`) is the most computationally demanding scheme. Since this architecture does not perform any simplification of the coding, decoding and insertion algorithms, the required number of operations to process each block is always the same, depending only on the type (INTRA/INTER) of the considered frame. Consequently, if one compares the charts corresponding to the two considered pixel-domain transcoding architectures (`PX_MV` and `PX_nMV`), it is possible to conclude that the difference in the number of required additions (about 62.8×10^6) is mainly due to the huge amount of additions/subtractions performed by the full-search block-matching motion estimation algorithm to compute the sum of absolute differences matching criteria for every candidate macroblock of INTER type images. All other processing blocks of these two

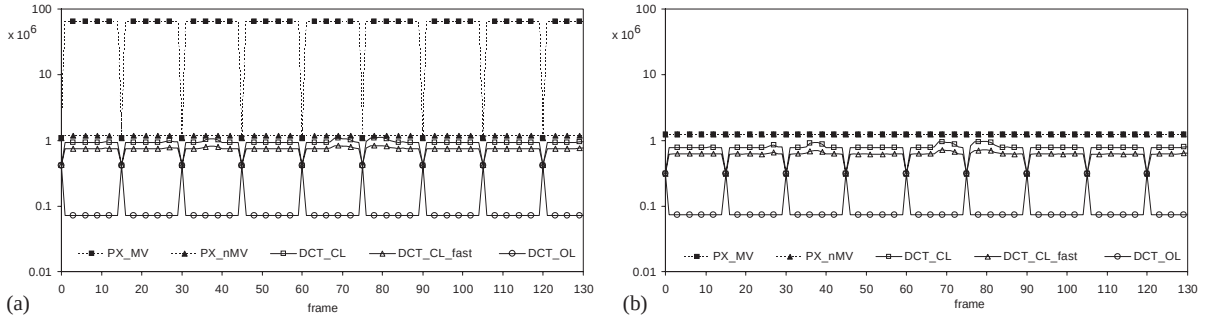

Fig. 11. Number of operations required to process the *Akiyo* video sequence using $Q = 0$: (a) Additions and (b) multiplications.

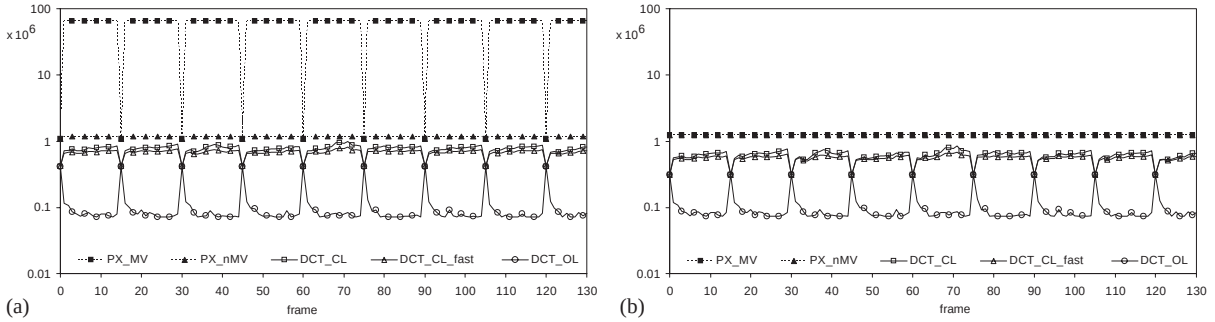

Fig. 12. Number of operations required to process the *Akiyo* video sequence using $Q = 15$: (a) Additions and (b) multiplications.

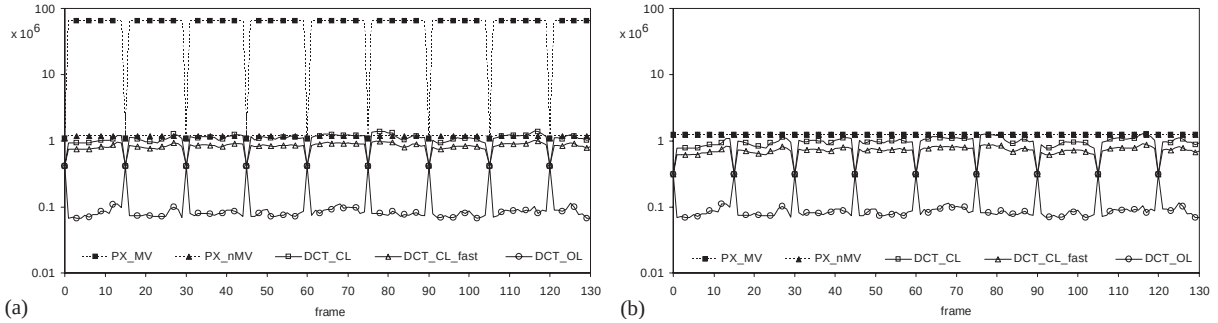

Fig. 13. Number of operations required to process the *Silent-Voice* video sequence using $Q = 0$: (a) Additions and (b) multiplications.

architectures are exactly the same, leading to the same amount of multiplications required by the two insertion algorithms.

Tables 1–3 present some figures of merit concerning the considered DCT-domain architectures, namely, the minimum, the average and the maximum number of operations required to process the considered video sequences using the two quantizer setups. For comparative purposes, the values presented in these tables were normalized in relation to the corresponding computational load of the pixel-domain architecture

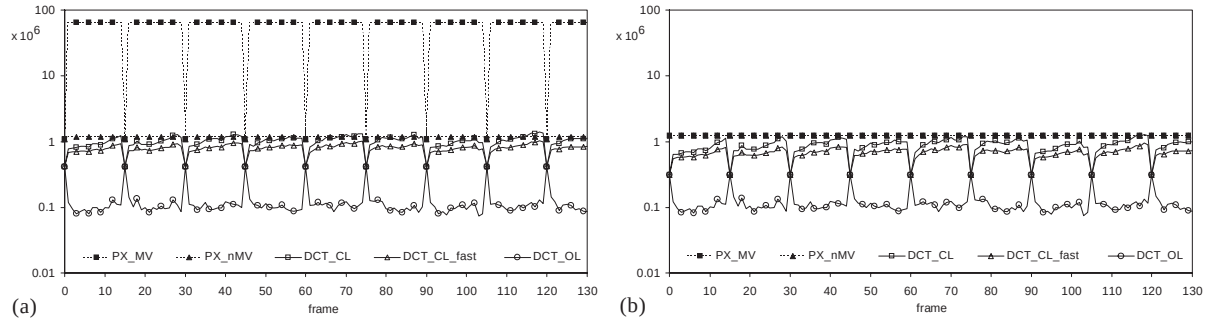

Fig. 14. Number of operations required to process the *Silent-Voice* video sequence using $Q = 15$: (a) Additions and (b) multiplications.

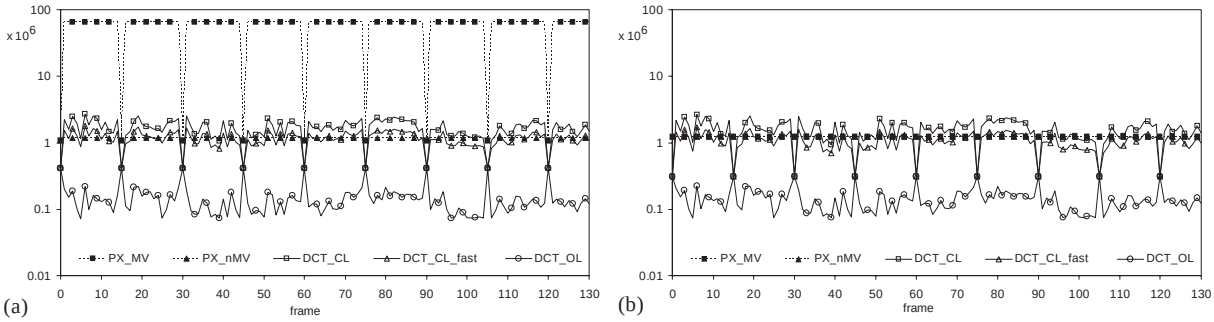

Fig. 15. Number of operations required to process the *Carphone* video sequence using $Q = 0$: (a) Additions and (b) multiplications.

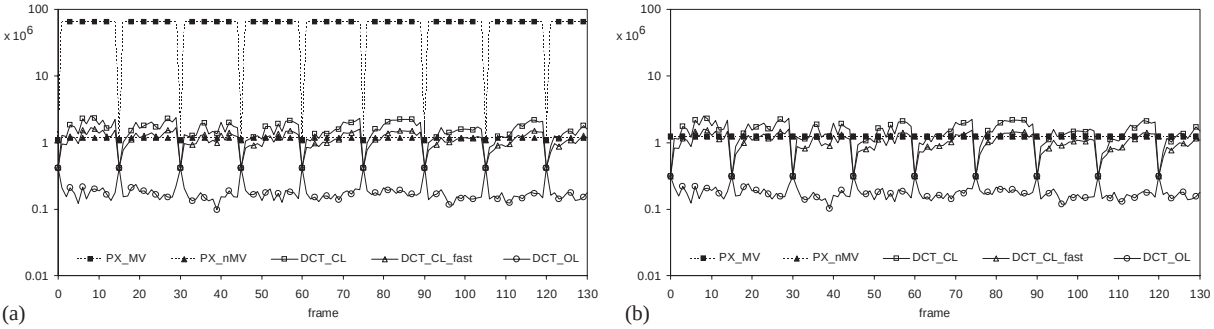

Fig. 16. Number of operations required to process the *Carphone* video sequence using $Q = 15$: (a) Additions and (b) multiplications.

without re-estimation of the motions vectors (`PX_nMV`).

Contrasting with the pixel-domain architectures, the amount of computations required to process INTER type images by the DCT-domain transcoders is not constant within a certain GOP (see Figs. 11–18). In fact, the number of macroblocks that require the insertion or removal of the NRSOs is not always the same, thus leading to the presented variation. Moreover, the computational cost of the two proposed closed-loop DCT-domain architectures (`DCT_CL` and `DCT_CL_fast`)

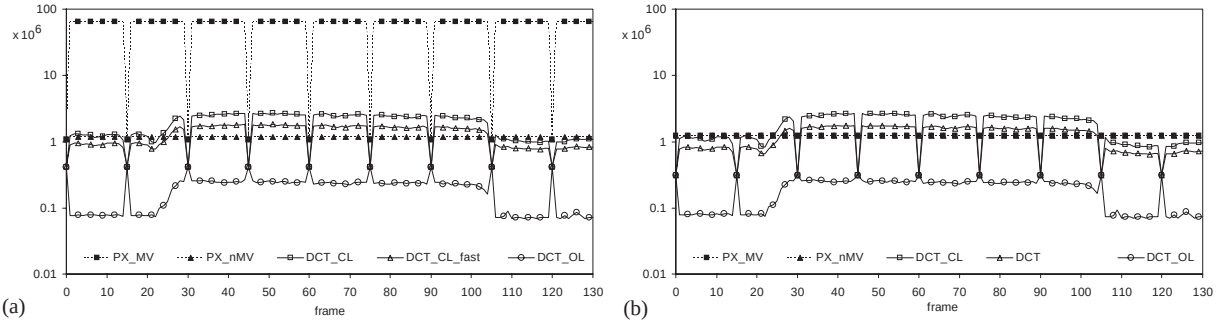

Fig. 17. Number of operations required to process the *Table-tennis* video sequence using $Q = 0$: (a) Additions and (b) multiplications.

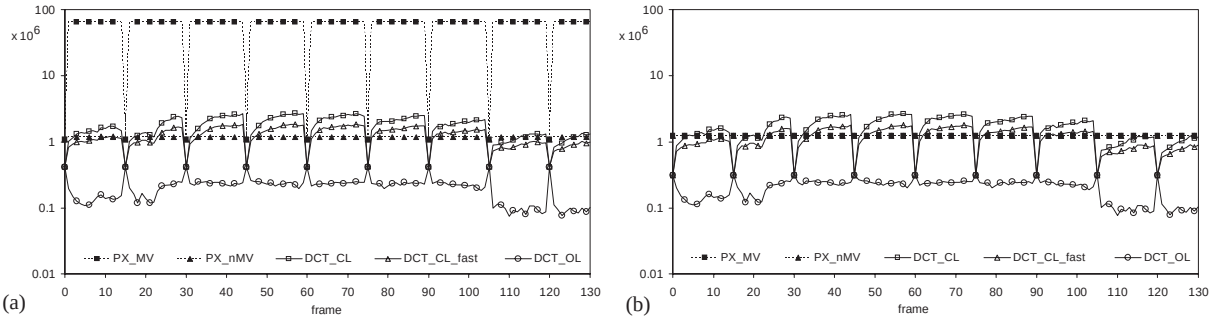

Fig. 18. Number of operations required to process the *Table-tennis* video sequence using $Q = 15$: (a) Additions and (b) multiplications.

Table 1
Relation between the minimum, average and maximum number of operations required by the closed-loop DCT-domain transcoder (`DCT_CL`) and those required by the pixel-domain architecture without re-estimation of the motion vectors (`PX_nMV`).

Video sequence	$Q = 0$						$Q = 15$					
	Additions			Multiplications			Additions			Multiplications		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Akiyo	0.38	0.78	0.95	0.25	0.62	0.78	0.38	0.66	0.84	0.25	0.49	0.68
Silent-Voice	0.38	0.93	1.21	0.25	0.76	1.04	0.38	0.87	1.20	0.25	0.70	1.03
Carphone	0.38	1.42	2.31	0.25	1.26	2.15	0.38	1.34	2.16	0.25	1.18	1.99
Table-tennis	0.38	1.59	2.31	0.25	1.42	2.15	0.38	1.48	2.31	0.25	1.31	2.15

has its absolute minimum values (corresponding to 38% of the number of additions and 25% of the number of multiplications of the `PX_nMV` architecture) at the instants corresponding to INTRA type frames. This fact can be easily justified by observing the structure of these two transcoders described in Sections 4.3 and 4.4: for INTRA type images it is not necessary to perform the motion compensation mechanism neither it is necessary to perform the removal of previously inserted objects in certain macroblocks of the image being processed.

Consequently, these two DCT-domain architectures may present computational loads that vary

Table 2
Relation between the minimum, average and maximum number of operations required by the computational-reduced closed-loop DCT-domain transcoder (DCT_CL_fast) and those required by the pixel-domain architecture without re-estimation of the motion vectors (PX_nMV)

Video sequence	$Q = 0$						$Q = 15$					
	Additions			Multiplications			Additions			Multiplications		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Akiyo	0.38	0.63	0.72	0.25	0.49	0.58	0.38	0.58	0.69	0.25	0.44	0.55
Silent-Voice	0.38	0.71	0.87	0.25	0.57	0.73	0.38	0.68	0.90	0.25	0.54	0.76
Carphone	0.38	1.00	1.52	0.25	0.85	1.38	0.38	0.97	1.43	0.25	0.83	1.29
Table-tennis	0.38	1.12	1.56	0.25	0.97	1.42	0.38	1.08	1.57	0.25	0.94	1.43

Table 3
Relation between the minimum, average and maximum number of operations required by the open-loop DCT-domain transcoder (DCT_OL) and those required by the pixel-domain architecture without re-estimation of the motion vectors (PX_nMV).

Video sequence	$Q = 0$						$Q = 15$					
	Additions			Multiplications			Additions			Multiplications		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Akiyo	0.07	0.08	0.36	0.06	0.07	0.25	0.07	0.09	0.36	0.06	0.08	0.25
Silent-Voice	0.06	0.09	0.36	0.06	0.08	0.25	0.07	0.11	0.36	0.06	0.10	0.25
Carphone	0.07	0.13	0.36	0.06	0.12	0.25	0.09	0.16	0.36	0.08	0.14	0.25
Table-tennis	0.06	0.16	0.36	0.06	0.15	0.25	0.07	0.18	0.36	0.06	0.16	0.25

significantly along the video sequence under processing. This phenomenon is emphasized in the charts representing the amount of computations required to process the *Table-tennis* sequence (Figs. 17 and 18). In fact, three distinct fractions can be distinguished in this sequence: the fraction corresponding to frames 1 through 25, the fraction corresponding to frames 25 through 105 and the fraction corresponding to frames 105 through 130. While in the first and in the third fractions the movements presented in the video sequence are local and restricted to certain regions of the image (e.g. ping-pong ball), during the second fraction there is a global movement of the whole scene caused by a zooming-out mechanism performed by the video camera. This zoom-out mechanism is responsible for a significant increment of the number of MBs that require some processing, thus justifying the abrupt increase presented in the charts of Figs. 17 and 18. Consequently, this variable computational load should be taken in consideration. It is greatly dependent on the motion activity in the scene and on the amount of MBs that are involved in the insertion of the considered NRSOs.

Thus, while the DCT_CL architecture requires a computational load that varies, on average, between 0.49 and 1.59 of the PX_nMV, the DCT_CL_fast structure requires an amount of computations that ranges between 0.44 and 1.12 of the load required by the reference architecture, still achieving a video quality very close to the DCT_CL architecture (as it will be shown in Section 5.2). Consequently, from the results presented in Tables 1 and 2 one should conclude that, depending on the video sequence under processing, DCT-domain insertion algorithms may offer significant advantages in what concerns the computational load, requiring, in certain cases, an average number of operations that is one half of the

number of operations required by the traditional pixel-domain insertion algorithms.

To conclude, a special attention should be given to the number of operations required by the open-loop DCT-domain architecture (DCT_OL). From the values in Table 3 and from the charts presented in Figs. 11–18, and by taking into account the description presented in Section 4.5, one can conclude that this architecture provides a significant saving in the number of required operations to perform the insertion algorithm. As in the other DCT-domain architectures, this saving depends greatly on the motion activity in the scene and corresponds to a tremendous low average computational load, that ranges between 6% and 18% of the number of operations required by the pixel-domain architecture (PX_nMV). As it was referred before, this significant computational saving is mainly due to the absence of the 2D-symmetric convolution block in the processing of INTER type images. Consequently, and contrasting with the previously described DCT-domain insertion architectures, the overall computational cost of this scheme has its maximum instants corresponding to the processing of INTRA type images, as it can be seen in the charts presented in Figs. 11–18. Therefore, this architecture may offer computational advantages over the other two schemes in the processing of video sequences with long GOP structures.

5.2. Quality and degradation effects in the coded video sequences

The quality of the video sequences that are obtained after the application of the proposed insertion algorithms is greatly dependent on several degradation effects that are introduced by the transcoding algorithms. As it will be seen, the reason for this degradation is not only the usual set of quantization operations, but it is also a deficient usage of the prediction scheme in the

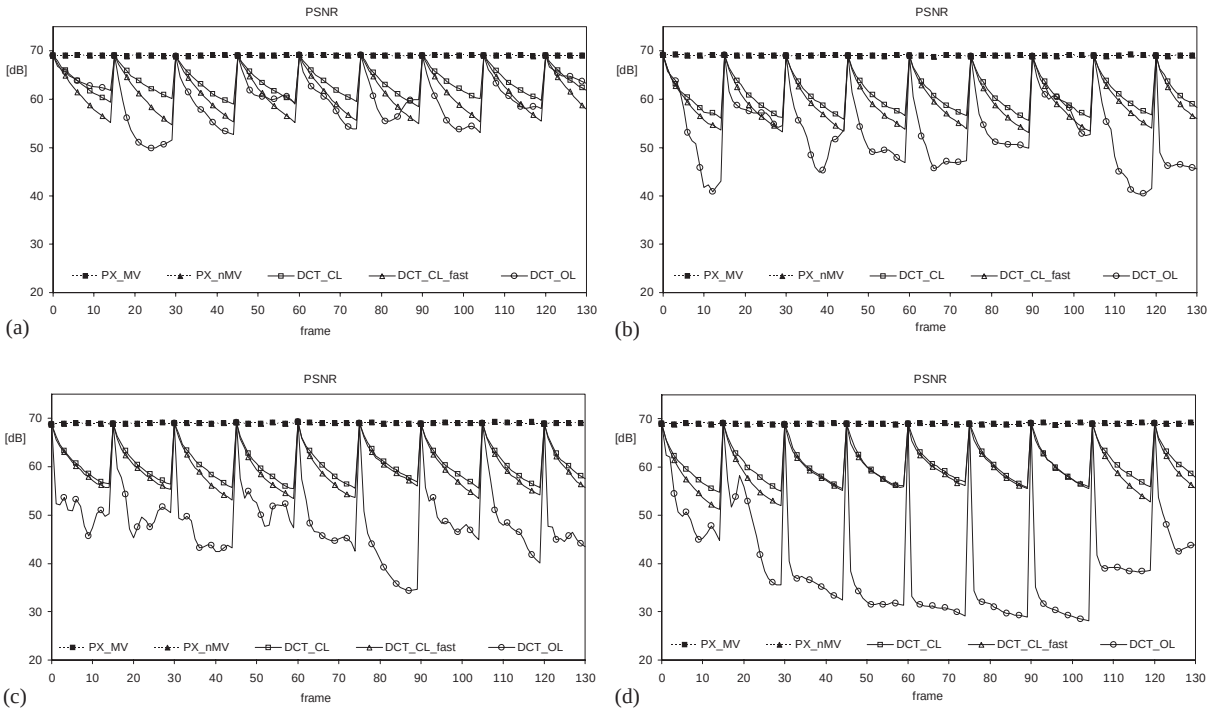

Fig. 19. Obtained PSNR for video sequences (a) *Akiyo*; (b) *Silent-Voice*; (c) *Carphone* and (d) *Table-tennis* using $Q = 0$.

coding algorithm. The main degradation effects that are present in the proposed transcoder architectures are:

(A) – Precision errors in the computations;
(B) – Usage of invalid motion vectors;
(C) – Motion estimation based on already distorted images;
(D) – Two-fold dequantization-quantization;
(E) – Drift introduced in INTER type images.

A detailed discussion about these degradations will be presented in the following subsections, using the widely adopted PSNR measure and an optimal pixel-domain compositing scheme to obtain the reference (maximum quality) video sequence. Such presentation will be illustrated with the variation of the PSNR for the four considered sequences using a full-precision quantizer ($Q = 0$) (see Fig. 19) and a non-null quantizer with $Q = 15$ (see Fig. 20).

In Fig. 21, it is presented the variation of the average PSNR (considering the whole set of frames of each video sequence) with the transparency factor α. These charts provide us the ability to realize that the video quality decreases when the opacity of the inserted NRSO is increased. This fact was already expected, since the α factor can be regarded as a measure of the overall disturbance introduced in the video sequence. Moreover, this behavior complies with the estimation of the amount of distortion introduced by the open-loop insertion architecture, which was previously presented in Eq. (44).

5.2.1. *Precision errors in the computations*

By analyzing the several charts of the PSNR obtained for a full-precision quantizer ($Q = 0$) (see Fig. 19) one can easily conclude that the overall quality of the video sequences processed by the compressed DCT-domain transcoders is somewhat lower than the quality obtained with pixel-domain transcoders. In the absence of any source of degradation due to quantization, the reason for this fact is associated with the introduction of

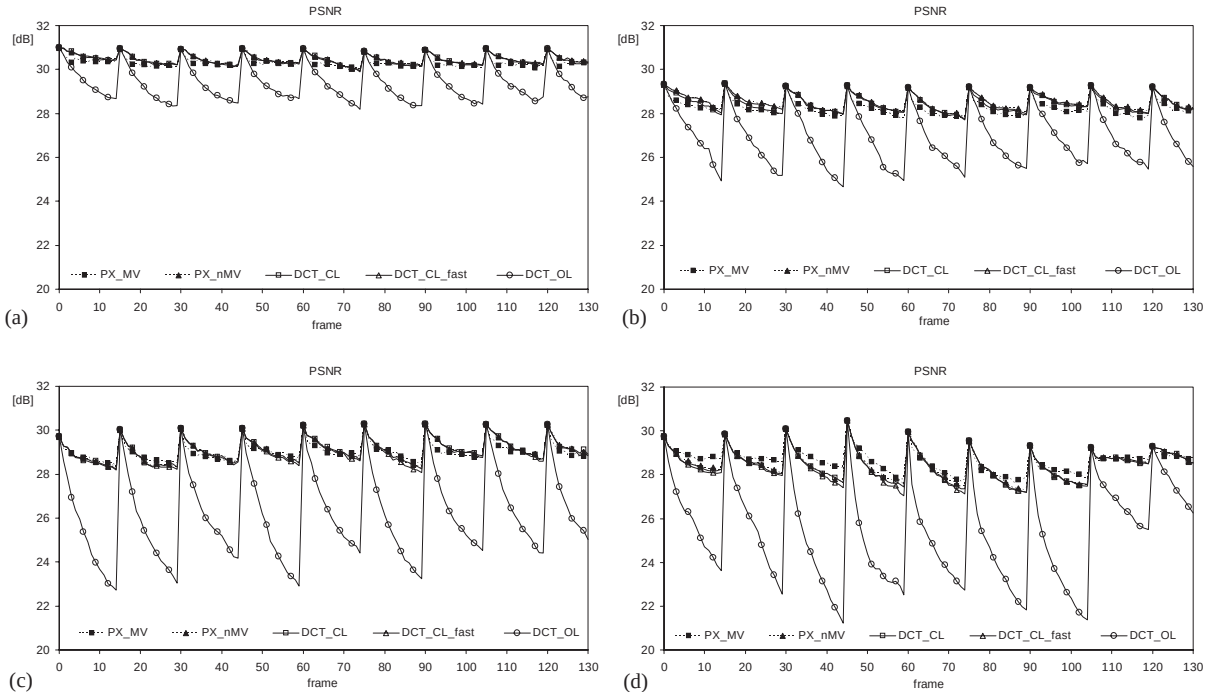

Fig. 20. Obtained PSNR for video sequences (a) *Akiyo*; (b) *Silent-Voice*; (c) *Carphone* and (d) *Table-tennis* using $Q = 15$.

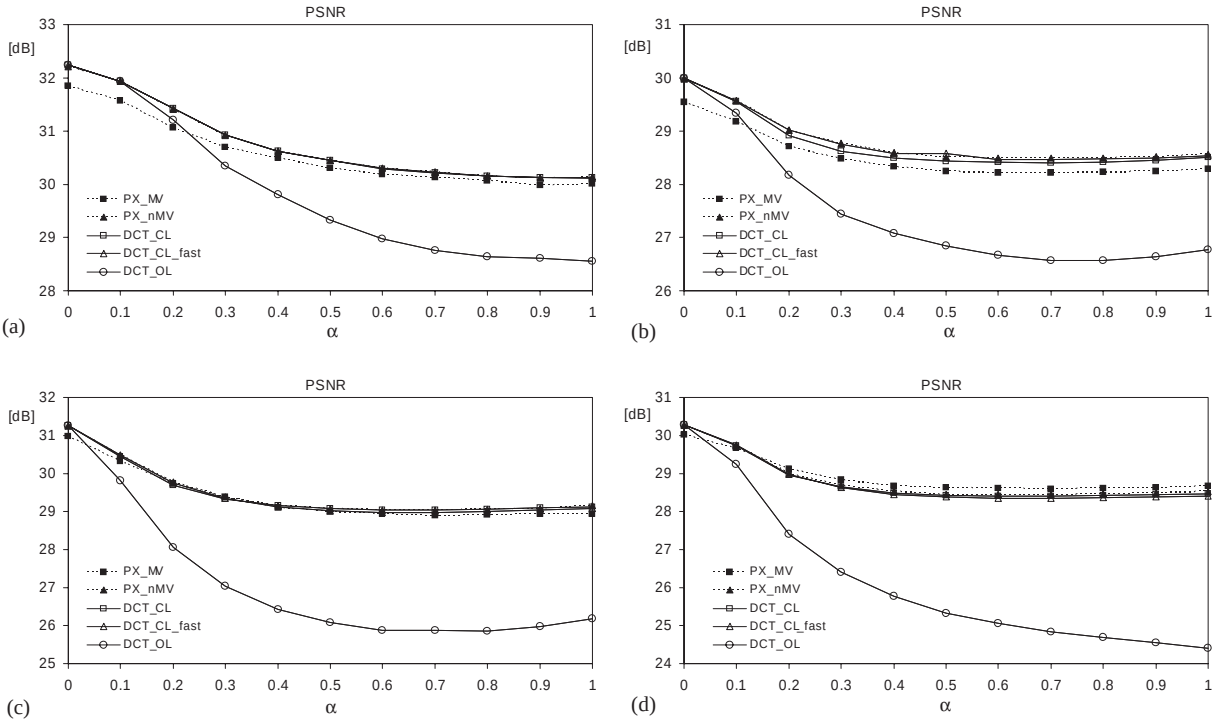

Fig. 21. Average PSNR for video sequences (a) *Akiyo*; (b) *Silent-Voice*; (c) *Carphone* and (d) *Table-tennis* using $Q = 15$.

precision errors during the several computations performed with the DCT coefficients of the video sequences. In fact, since the DCT transform concentrates most of the information in the lower frequency coefficients of each block, a slight precision error in such low-frequency coefficients will have a more significant effect on the PSNR measure than it would have if the same minor error was introduced in the pixel-domain.

However, it is worth noting that the visibility of this degradation due to precision errors is greatly attenuated when real quantizers are used in the coding algorithm. As it is illustrated in Fig. 20, when a non-null quantizer with $Q = 15$ is used the overall quality of both the compressed DCT-domain transcoders and of the pixel-domain transcoders is very similar, presenting a degradation up to 1.5 dB in the codification of the last frame of each GOP due to the accumulated drift introduced by the quantization errors (with exception to the *open-loop DCT-domain transcoder*, whose properties will be further described in Section 5.2.5).

5.2.2. Usage of invalid motion vectors

Despite all strategies that have been proposed by several authors over the last few years to re-use, in the coding of the output bit-stream, as much information decoded from the incoming bit-stream as possible, one now has to take into account that this information may no longer be valid for the MBs in the vicinity of the insertion. In fact, video transcoders usually re-use the decoded MVs, which may no longer point to the best matching MB. Furthermore, it is also possible that MVs that pointed to the MBs where the object has been inserted may have to be re-estimated, since the object content is not expected to be inserted in those regions of the image.

Consequently, this re-usage of the received MVs may lead to the introduction of additional degradative effects, since it will prevent a perfect

matching between the MBs of the current and previous images during the predictive coding phase of the algorithm. This effect will be even more sensed in video sequences with a significant amount of movement. In fact, this is one of the reasons why the pixel-domain insertion scheme PX_MV presents greater values of PSNR in the *Table-tennis* video sequence in Fig. 20(d), superseding the PX_nMV architecture. One possible way to circumvent this problem is to perform a new motion estimation procedure, after the insertion of the logos. However, this approach may introduce other type of degradation, as will be described in Section 5.2.3.

Panusopone [12] has recently studied the problem of inserting translucent objects (see Eq. (1)) in pixel-domain transcoders and proposed several schemes to adapt the MVs and the quantization steps so that the resulting impact on the coding algorithm is small [12]. Despite the fact that the proposed insertion techniques are clearly different from those presented in this paper (since they are intended to be used in the pixel-domain), his proposals concerning the re-usage of the MVs can still be applied to the proposed schemes with minor changes.

5.2.3. Motion estimation based on already distorted images

As it was described in Section 4.1, the transcoding architecture PX_MV performs an entirely new motion estimation at the encoder side of the transcoder. However, this re-estimation may be significantly influenced by the quantization operation. In fact, while the MVs of the incoming bit stream were computed with full-precision pixel values, the search procedure is now performed with decoded pictures that were already affected by the quantization distortion of the coding algorithm [1], which can introduce some more degradation to the overall algorithm.

However, when a new estimation of the MVs is performed at the encoder side of the transcoder, it is the set of processed images, with the NRSOs already inserted on them, that are taken into account. Consequently, it is expected that not only will this improve the efficiency of the coding algorithm on such blocks (with significant influences on the output bit rate), but it may also affect the obtained video quality, due to a more perfect matching between the MBs of the current and previous frame, with benefits in the predictive coding phase of the algorithm. The effect of this option will be more noticeable in video sequences with a significant amount of movement, as it is illustrated in Fig. 10(d) for the *Table-tennis* video sequence. Nevertheless, this option may introduce distortion in the rest of the blocks.

To circumvent this problem, one could conceive an alternative hybrid solution that would re-use the motion vectors of those macroblocks not processed by the insertion algorithm and would re-estimate the motion vectors of the macroblocks where the NRSOs were inserted. However, not only would it imply the loss of the regularity of the algorithm, with dramatic effects in its control units, but it would significantly affect its performance towards the operation in real-time. In fact, as it was shown in Section 5.1, the presence of the motion estimation block in the transcoder algorithm becomes the involved computational cost prohibitively high for real-time applications.

5.2.4. Two-fold dequantization-quantization

It is widely accepted that one of the main causes for distortion in video coding arises from the usage of quantization. In fact, independently of the considered transcoder algorithm (either in the pixel domain or in the DCT-domain), it is always necessary to dequantize the received DCT coefficients at the decoder side of the transcoder. Soon after the insertion procedure, this data will have to be re-quantized at the encoder end of the transcoding system. This extra decoding-encoding process usually leads to the introduction of an additional degradation effect, as a result of this two-fold dequantization-quantization operation [1,3]. In fact, according to the results presented in [3], the losses that are introduced by this cascaded quantization may reach 0.5–1.0 dB.

Paradoxically, this is one of the aspects where the DCT-domain architectures take advantage over the pixel domain schemes. In fact, contrary to the pixel-domain architectures, which perform

the dequantization operation over the entire image, the DCT domain transcoders only carry-out any processing in such blocks where the insertion (or removal) of the NRSOs is to be performed. All the remaining blocks can be skipped from the entire process. Consequently, the degradation effect due to this two-fold dequantization-quantization operation is considerably less severe in the DCT domain algorithms. In fact, this is the main reason why the obtained PSNR of the processed sequences is higher for the DCT_CL and DCT_CL_fast schemes (as it can be seen in Figs. 20(a)–(c) for the *Akiyo*, *Silent-Voice* and *Carphone* video sequences). This advantage is more significant in video sequences with low or medium amount of movement, since other factors may benefit the pixel domain architectures when processing video sequences with high amount of movement, as it was described in Section 5.2.2.

5.2.5. Drift introduced in INTER type images

Despite the great compression capability that the quantization block offers, it is the most responsible for the accumulation of errors (drift). In fact, since INTER type images are coded using a temporal prediction scheme based on the previously coded INTRA/INTER type frame, this degradation effect tends to suffer a gradual aggravation along the codification of a given GOP, as it was already shown in Figs. 19 and 20.

In Fig. 22, it is illustrated a fraction of a coded GOP composed by $M = 15$ frames, obtained by applying the insertion algorithm to the *Table-tennis* video sequence. The reason for choosing this video sequence to illustrate this phenomenon emerges from the fact that it offers the set of worst case conditions to apply the proposed algorithms: moderate spatial detail and a significant amount of motion, simultaneously of translational and of zoom-out types. The image shown in Fig. 22(a) is the first frame of that GOP (INTRA type), which will be used as the reference for the following INTER type image. In Fig. 22(b), it is illustrated the last frame of the same GOP (INTER type). Fig. 22(c) illustrates the first frame (INTRA type) of the following GOP. The set of images illustrated in Fig. 22 were obtained with a full-precision quantizer ($Q = 0$) and using the pixel-domain transcoder (PX_MV). The usage of this ideal quantizer led to the minimization of the drift introduced along the GOP, as it was already shown in the chart of Fig. 19(d).

In Fig. 23, it is illustrated the last image (frame no. 44) of the considered GOP using the same full-precision quantizer and all the other proposed architectures for the insertion algorithm: PX_nMV (Fig. 23(a)), DCT_CL (Fig. 23(b)), DCT_CL_fast (Fig. 23(c)) and DCT_OL (Fig. 23(d)). As it was previously illustrated in Fig. 19(d), the PSNR obtained using the pixel-domain transcoder with re-usage of the motion vectors (PX_nMV) is very similar to the PSNR obtained using the pixel-domain insertion algorithm with re-estimation of the motion vectors (PX_MV). In Figs. 23(b) and (c) it is illustrated the same image obtained with the DCT_CL and DCT_CL_fast transcoders, respectively. As it was already referred, the precision errors introduced during the set of computations involving the DCT coefficients led to the introduction of a slightly distortion. However, considering that the proposed DCT-domain insertion algorithms only affect the MBs where the NRSO is supposed to be inserted (and, eventually, the surrounding MBs), the degradation effect can be assumed to be a localized distortion that only affects a restricted area of the image (which tends to increase along the GOP), corresponding to the fraction affected by the insertion of the NRSO. If one takes into account that the considered NRSOs do not significantly change with time, one can naturally consider that, from the subjective point of view, the degradation that is introduced in such areas will not significantly affect the perception of the rest of the scene. Consequently, they won't be easily perceived by the viewer.

In Fig. 23(d), it is illustrated the same frame but obtained, in this case, with the open-loop DCT-domain transcoder (DCT_OL). The observation of this image provides the means to further understand the source of the significant degradation effect that was shown in the chart of Fig. 19(d). In fact, by comparing this frame with the first image of the considered GOP (INTRA image illustrated in Fig. 22(a)), one can easily realize that the gap-line that crosses the word "WORLD" in the subtitle of Fig. 23(d) is almost parallel to the edge

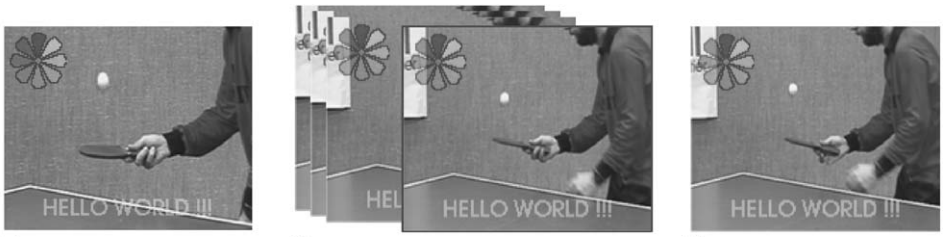

Fig. 22. First frames ((a) and (c)—INTRA type) and last frame ((b)—INTER type) of two consecutives GOPs of the *Table-tennis* video sequence using $Q = 0$ and the pixel-domain transcoder (with re-estimation of the motion vectors). (a) Frame no. 30 [68.93 dB]; (b) frame no. 44 [68.89 dB] and (c) frame no. 45 [68.94 dB].

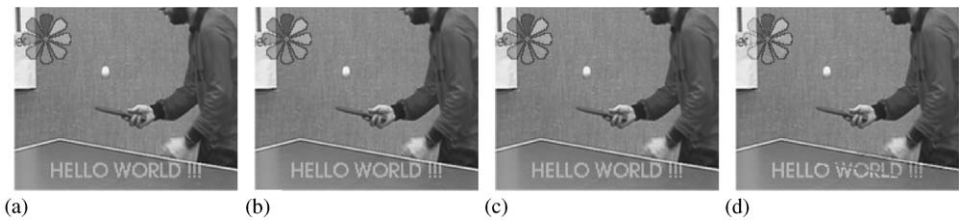

Fig. 23. Last frame (INTER type) of a GOP of the *Table-tennis* video sequence, processed using: (a) the pixel-domain transcoder with re-usage of the motion vectors (frame no. 44 [68.93 dB]); (b) the closed-loop DCT-domain transcoder (frame no. 44 [55.50 dB]); (c) the computational-reduced DCT-domain transcoder (frame no. 44 [55.17 dB]) and (d) the open-loop DCT-domain transcoder (frame no. 44 [32.47 dB]) ($Q = 0$).

of the table. If one takes into account the description of the insertion mechanism of this scheme (see Section 4.5), one can easily conclude that this mismatch is mainly caused by a deficient compensation of the insertion algorithm, as it was previously described. This phenomenon is particularly perceived in the following conditions:

- presence of sharp edges in the background of INTRA type images where the NRSO was previously inserted;
- presence of motion in the area affected by the insertion algorithm.

Thus, the combined effect of the presence of the table edge and of the zooming-out mechanism caused by the video camera provided the worst processing conditions which led to an easy perception of this phenomenon (whose distortion effect is easily noticeable in the chart presented in Fig. 19(d)) and to an increment of the computational load (as it was previously noted). Were these conditions not satisfied and this degradation would not be so evident to the viewer.

Despite this degradation effect, there is still another noticeable phenomenon in the region covered by the logo which contributes to the distortion level presented in the chart of Fig. 19(d). In fact, if one compares the "*flower*" logo inserted in this figure with the logos inserted with the other transcoder architectures it is possible to perceive a certain decrease of its luminance level. This fact can be justified by a deficient processing of the received differences signal (E_l) and of the transparency factor (α) in the processing of INTER type images.

The images shown in Figs. 24 and 25 were obtained in a similar manner as those presented in Figs. 22 and 23, but using, in this case, a real quantizer with $Q = 15$. By observing Figs. 24(b), 25(a), 25(b) and 25(c) one can easily conclude that both pixel-domain insertion algorithms and the closed-loop DCT-domain architectures provide roughly the same video quality, as it was

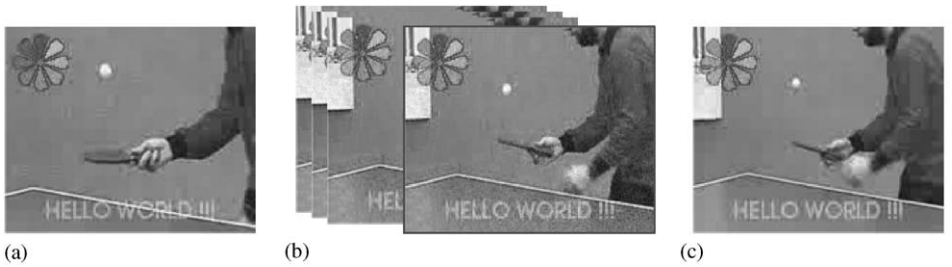

Fig. 24. First frames ((a) and (c)—INTRA type) and last frame ((b)—INTER type) of two consecutives GOPs of the *Table-tennis* video sequence using $Q = 15$ and the pixel-domain transcoder (with re-estimation of the motion vectors). (a) Frame no. 30 [30.09 dB]; (b) frame no. 44 [28.29 dB] and (c) frame no. 30 [30.45 dB].

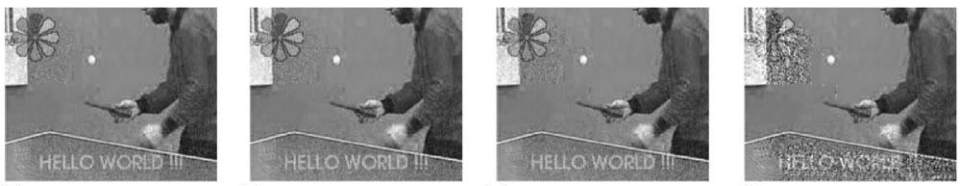

Fig. 25. Last frame (INTER type) of a GOP of the *Table-tennis* video sequence, processed using: (a) the pixel-domain transcoder with re-usage of the motion vectors (frame no. 44 [27.63 dB]); (b) the closed-loop DCT-domain transcoder (frame no. 44 [27.65 dB]); (c) the computational-reduced DCT-domain transcoder (frame no. 44 [27.41 dB]) and (d) the open-loop DCT-domain transcoder (frame no. 44 [21.22 dB]) ($Q = 15$).

previously illustrated in the charts of Fig. 20(d). On the other hand, from the observation of Fig. 25(d), corresponding to the same frame but obtained with the open-loop DCT-domain transcoder (`DCT_OL`), it is possible to perceive a much more serious degradation effect surrounding the NRSO area. This degradation is mainly caused by a deficient removal/insertion of the NRSO data in the processed macroblocks of INTER type images (see Eq. (52)), which is significantly worsened by the quantization errors inserted along the coding of the GOP. In fact, as it can be seen in the chart of Fig. 20(d), this frame corresponds to the minimum PSNR value obtained in the whole set of 130 frames, corresponding to the instant when the zooming-out mechanism is more significant.

Even so, it is worth noting that the significant computational savings offered by the open-loop DCT-domain insertion architecture can still be used in certain applications, where the particular characteristics in terms of the amount of motion and spatial details provide the required conditions to keep the distortion level low. One such application is the processing of video sequences like *Akiyo* and *Silent Voice* with a small GOP length ($M = 8$). In Figs. 26(a) and (b), it is presented the last frame of the first GOP of these video sequences. As it can be seen, for these particular conditions the amount of distortion introduced by this algorithm is still visually acceptable from the subjective point of view.

6. Conclusions

In this paper, an object insertion scheme in the compressed DCT-domain for video transcoding was proposed. This scheme was adapted from the well-known pixel-domain compositing technique and implies the usage of a symmetric convolution operator in the DCT-domain, so that only the pixels corresponding to those objects that are

(a)

(b)

Fig. 26. Last frame (INTER type) of the first GOP (frame no. 8) of the Akyio (a) and Silent Voice (b) video sequences, processed using the open-loop DCT-domain transcoder with a GOP length $M=8$ ($Q=8$). (a) PSNR = 33.07 dB and (b) PSNR = 31.52 dB.

intended to be inserted are considered in the operation. By using this technique, the problem concerning the presence of undesired semi-transparent rectangular regions around irregular-shaped objects (such as logos or subtitles) was circumvented.

Several pixel-domain and DCT-domain transcoding architectures were proposed to implement the NRSO insertion system. A detailed discussion about the obtained quality and degradation effect, and of the computational cost involved in these architectures was presented. In particular, it was shown that DCT-domain insertion architectures can offer significant advantages both in terms of image quality and computational cost. They also provide the possibility to adapt the characteristics of the insertion architecture to the target application, adjusting the desired tradeoff between image quality and computational cost.

References

[1] P. Assunção, M. Ghanbari, A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bitstreams, IEEE Trans. Circuits Systems Video Technol. 8 (8) (December 1998) 953–967.

[2] S.-F. Chang, D.G. Messerschmitt, Manipulation and compositing of MC-DCT compressed video, IEEE J. Sel. Areas Commun. 13 (1) (January 1995) 1–11.

[3] G. Keesman, R. Hellinghuizen, F. Hoeksema, G. Heideman, Transcoding of MPEG bit-streams, Signal Process. Image Commun. 8 (1996) 481–500.

[4] H. Li, H. Shi, A fast algorithm for reconstructing motion compensated blocks in compressed domain, J. Visual Languages Comput. 10 (6) (December 1999) 607–623.

[5] C.-W. Lin, Y.-R. Lee, Fast algorithms for DCT-domain video transcoding, in: Proceedings of IEEE International Conference on Image Processing, Thessaloniki, Greece, October 2001, pp. 421–424.

[6] S. Liu, A.C. Bovik, Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding, IEEE Trans. Circuits Systems Video Technol. 12 (5) (May 2002) 309–319.

[7] S. Martucci, Symmetric convolution and discrete sine and cosine transforms, IEEE Trans. on Signal Process. SP-42 (5) (May 1994) 1038–1051.

[8] J. Meng, S.-F. Chang, Embedding visible video watermarks in the compressed domain, Proceedings of IEEE International Conference on Image Processing (ICIP'1998), Chicago, IL, Vol. 1, 1998, pp. 474–477.

[9] N. Merhav, V. Bhaskaran, A fast algorithm for DCT-domain inverse motion compensation, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Atlanta, GA, USA, May 1996, Vol. 4, pp. 2307–2310.

[10] J.L. Mitchell, W. Pennebaker, C. Fogg, D. LeGall, MPEG Video Compression Standard, Chapman & Hall, London, 1996.

[11] MPEG, MPEG-4 Video Verification Model, version 18.0-ISO/MPEG N3908, January 2001.

[12] K. Panusopone, X. Chen, F. Ling, Logo insertion in MPEG transcoder, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, USA, May 2001.

[13] F. Pereira, T. Ebrahimi (Eds.), *The MPEG-4 Book*, Prentice-Hall PTR, Englewood Cliffs, NJ, 2002.

[14] T. Porter, T. Duff, Compositing digital images, Comput. Graph. (Proc. SIGGRAPH'84) 18 (3) (July 1984) 253–259.

[15] N. Roma, L. Sousa, Insertion of irregular-shaped logos in the compressed DCT domain, in: Proceedings of the 14th International Conference on Digital Signal Processing (DSP'2002), IEEE, Santorini, Greece, July 2002, Vol. 1, pp. 125–128.

[16] N. Roma, L. Sousa, Transcoding architectures for object insertion in compressed video, Technical Report RT/006/2002, INESC-ID–Lisboa, Portugal, October 2002.

[17] T. Shanableh, M. Ghanbari, Transcoding architectures for DCT-domain heterogeneous video transcoding, in: Proceedings of IEEE International Conference on Image Processing—ICIP'2001, Thessaloniki, Greece, October 2001.

[18] B. Shen, K. Sethi, Block-based manipulations of transformed-compressed images and videos, Multimedia Systems J. 6 (2) (March 1998) 113–124.

[19] B. Shen, I.K. Sethi, V. Bhaskaran, DCT convolution and its application in compressed domain, IEEE Trans. Circuits Systems Video Technol. 8 (8) (December 1998) 947–952.

[20] J. Song, B.-L. Yeo, A fast algorithm for DCT-domain inverse motion compensation based on shared information in a macroblock, IEEE Trans. Circuits Systems Video Technol. 10 (5) (August 2000) 767–775.

[21] S.J. Wee, B. Vasudev, Splicing MPEG video streams in the compressed domain, in: Proceedings of IEEE Workshop on Multimedia Signal Processing, Princeton, June 1997.