Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 57291, 16 pages
doi:10.1155/2007/57291

Research Article

Efficient Hybrid DCT-Domain Algorithm for Video Spatial Downscaling

Nuno Roma and Leonel Sousa

INESC-ID/IST, TULisbon, Rua Alves Redol 9, 1000-029 Lisboa, Portugal

Received 30 August 2006; Revised 16 February 2007; Accepted 6 June 2007

Recommended by Chia-Wen Lin

A highly efficient video downscaling algorithm for any arbitrary integer scaling factor performed in a hybrid pixel transform domain is proposed. This algorithm receives the encoded DCT coefficient blocks of the input video sequence and efficiently computes the DCT coefficients of the scaled video stream. The involved steps are properly tailored so that all operations are performed using the encoding standard block structure, independently of the adopted scaling factor. As a result, the proposed algorithm offers a significant optimization of the computational cost without compromising the output video quality, by taking into account the scaling mechanism and by restricting the involved operations in order to avoid useless computations. In order to meet any system needs, an optional and possible combination of the presented algorithm with high-order AC frequency DCT coefficients discarding techniques is also proposed, providing a flexible and often required complexity scalability feature and giving rise to an adaptable tradeoff between the involved scalable computational cost and the resulting video quality and bit rate. Experimental results have shown that the proposed algorithm provides significant advantages over the usual DCT decimation approaches, both in terms of the involved computational cost, the output video quality, and the resulting bit rate. Such advantages are even more significant for scaling factors other than integer powers of 2 and may lead to quite high PSNR gains.

1. INTRODUCTION

In the last few years, 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 the digital form. However, once video signals are compressed, delivery systems and service providers frequently face the need for further manipulation and processing of such compressed bit streams, in order to adapt their characteristics not only to the available channel bandwidth but also to the characteristics of the terminal devices.

Video transcoding has recently emerged as a new research area concerning a set of manipulation and adaptation techniques to convert a precoded video bit stream into another bit stream with a more convenient set of characteristics, targeted to a given application. Many of these techniques allow the implementation of such processing operations directly in the compressed precoded video streams, thus offering significant advantages in what concerns the computational cost and distortion level. This processing may include changes on syntax, format, spatial and temporal resolutions, bit-rate adjustment, functionality, or even hardware requirements. In addition, the computational resources available in many target scenarios, such as portable, mobile, and battery supplied devices, as well as the inherent real-time processing requirements, have raised a major concern about the complexity of the adopted transcoding algorithms and of the required arithmetic structures [1–4].

In this context, spatial frame scale is often required to reduce the image resolution by a given scaling factor ($\mathcal{S}$) before transmission or storage, thus reducing the output bit rate. From a straightforward point of view, image resizing of a compressed video sequence can be performed by cascading (i) a video decoder block; (ii) a pixel domain resizing module, to process the decompressed sequence; and (iii) an encoding module, to compress the resized video. However, this approach not only imposes a significant computational cost, but also introduces a nonnegligible distortion level, due to precision and round-off errors resulting from the several involved compressing and decompressing operations.

Consequently, several different approaches have been proposed in order to implement this downscaling process directly in the discrete cosine transform (DCT) domain, as it is described in [2, 5, 6]. However, despite the several different strategies that have been presented, most of such proposals are only directly applied to scaling operations using a scaling factor given by an integer power of 2 ($\mathcal{S} = 2, 4, 8, 16$, etc.). Nevertheless, downscaling operations using any other arbitrary integer scaling factor are often required. In the last few years, some proposals have arisen in order to implement these algorithms for any integer scale factors [7–11]. However, although these proposals provide good video quality for integer powers of 2 scaling ratios, their performance significantly degrades when other scaling factors are applied. One other important issue is concerned with the block structure adopted by these algorithms: the $(N \times N)$ pixels block structure (usually, with $N = 8$) adopted by most digital image (JPEG) and video (MPEG-x, H.261 and H.263) coding standards requires that both the input original frame and the output downscaled frame, together with all the data structures associated to the processing algorithm, are organized in $(N \times N)$ pixels blocks. As a consequence, other feasible and reliable alternatives have to be adopted in order to obtain better quality performances for any arbitrary scaling factor and to achieve the block-based organization found in most image and video coding standards.

Some authors have also distinguished the scaling algorithms in what concerns their output domains [12]. While the input and output blocks of some proposed algorithms are both in the DCT-domain, other approaches process encoded input blocks (DCT-domain) but provide their output in the pixel domain. The processing of such output blocks can then either continue in the pixel-domain or an extra DCT computation module can yet be applied, in order to recover the output of these algorithms into the DCT domain. As a consequence, this latter kind of approaches is often referred to as *hybrid* algorithms [12].

Hence, contrary to the most recent proposals [7–11], the algorithm proposed in this paper and described in Section 3 offers a reliable and very efficient video downscaling method for any arbitrary integer scaling factor, in particular, for scaling factors other than integer powers of 2. The algorithm is based on a *hybrid* scheme that adopts an averaging and subsampling approach performed in a hybrid pixel-transform domain, in order to minimize the introduction of any inherent distortion. Moreover, the proposed method also offers a minimization of the computational complexity, by restricting the involved operations in order to avoid spurious and useless computations and by only performing those that are really needed to obtain the output values. Furthermore, all the involved steps are properly tailored so that all operations are performed using $(N \times N)$ coefficient blocks, independently of the adopted scaling factor ($\mathcal{S}$). This characteristic was never proposed before for this kind of algorithms and is of extreme importance, in order to comply the operations with most image and video coding standards and simultaneously optimize the involved computational effort.

An optional and possible combination of the presented algorithm with high-order AC frequency DCT coefficients discarding techniques is also proposed [13–15]. These techniques, usually adopted by DCT decimation algorithms, provide a flexible and often required complexity scalability feature, thus giving rise to an adaptable tradeoff between the involved scalable computational cost and the resulting video quality and bit rate, in order to meet any system requirements.

The experimental results, presented in Section 4, show that the proposed algorithm provides significant advantages over the usual DCT decimation approaches, both in terms of the involved computational cost, the output video quality, and the resulting bit rate. Such advantages are even more significative when scaling factors other than integer powers of 2 are considered, leading to quite high peak signal-to-noise ratio (PSNR) gains.

2. SPATIAL DOWNSCALING ALGORITHMS

The several spatial-resolution downscaling algorithms that have been proposed over the past few years are usually classified in the literature according to three main approaches [2, 3, 6]:

(i) *filtering and down-sampling*, which adopts a traditional digital signal processing approach, where the downsampled version of a given block is obtained either by applying a given n-tap filter and dropping a certain amount of the filtered pixels [16]; or by following a frequency synthesis approach [17]; or by taking into account the symmetric-convolution property of the DCT [18];

(ii) *averaging and down-sampling*, in which every $(\mathcal{S}_x \times \mathcal{S}_y)$ pixels block is represented by a single pixel with its average value [5, 19–22]; some approaches have even adopted optimized factorizations of the filter matrix, in order to minimize the involved computational complexity [20];

(iii) *DCT decimation*, which downscales the image by discarding some high-order AC frequency DCT coefficients, retaining only a subset of low-order terms [8, 23–27]; some authors have also proposed the usage of optimized factorizations of the DCT matrix, in order to reduce the involved computational complexity [25, 27].

In the following, a brief overview of each of these approaches will be provided.

2.1. Pixel filtering/averaging and down-sampling approaches

From a strict digital signal processing point of view, the first two techniques may be regarded as equivalent approaches, since they only differ in the lowpass filter that is applied along the decimation process. As an example, by considering a simple downscaling procedure that converts each set of (2×2) adjacent blocks $\mathbf{b_{i,j}}$ (each one with (8×8) pixels) into one single (8×8) pixels block $\hat{\mathbf{b}}$ (see Figure 1), these two algorithms can be generally formulated as follows:

$$\hat{\mathbf{b}} = \sum_{i=0}^{1} \sum_{j=0}^{1} \mathbf{h_{i,j}} \cdot \mathbf{b_{i,j}} \cdot \mathbf{w_{i,j}}, \tag{1}$$

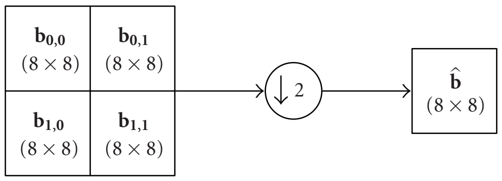

$\mathbf{b}_{0,0}$ (8×8)
$\mathbf{b}_{0,1}$ (8×8)
$\mathbf{b}_{1,0}$ (8×8)
$\mathbf{b}_{1,1}$ (8×8)
$\downarrow 2$
$\hat{\mathbf{b}}$ (8×8)

Figure 1: Downscaling four adjacent blocks in order to obtain a single block.

where $\mathbf{h}_{\mathbf{i,j}}$ and $\mathbf{w}_{\mathbf{i,j}}$ are the considered down-sampling filter matrices.

For the particular case of the application of the *averaging* approaches (usually referred to as *pixel averaging and down-sampling (PAD)* methods [12]), these filters are defined as [5, 19–22]

$$\begin{aligned}\mathbf{h}_{0,0} = \mathbf{h}_{0,1} = \mathbf{w}_{0,0}{}^{t} = \mathbf{w}_{1,0}{}^{t} &= \frac{1}{2}\begin{bmatrix}\mathbf{u}_{4\times 8}\\ \varnothing_{4\times 8}\end{bmatrix},\\ \mathbf{h}_{1,0} = \mathbf{h}_{1,1} = \mathbf{w}_{0,1}{}^{t} = \mathbf{w}_{1,1}{}^{t} &= \frac{1}{2}\begin{bmatrix}\varnothing_{4\times 8}\\ \mathbf{u}_{4\times 8}\end{bmatrix},\end{aligned} \tag{2}$$

where $\mathbf{u}_{4\times 8}$ is defined as

$$\mathbf{u}_{4\times 8} = \begin{bmatrix}1&1&0&0&0&0&0&0\\0&0&1&1&0&0&0&0\\0&0&0&0&1&1&0&0\\0&0&0&0&0&0&1&1\end{bmatrix}, \tag{3}$$

and $\varnothing_{4\times 8}$ is a (4×8) zero matrix.

These scaling schemes can be directly implemented in the DCT-domain, by applying the DCT operator to both sides of (1) as follows:

$$\mathrm{DCT}(\hat{\mathbf{b}}) = \mathrm{DCT}\left(\sum_{i=0}^{1}\sum_{j=0}^{1}\mathbf{h}_{\mathbf{i,j}}\cdot\mathbf{b}_{\mathbf{i,j}}\cdot\mathbf{w}_{\mathbf{i,j}}\right). \tag{4}$$

By taking into account that the DCT is a linear and orthonormal transform, it is distributive over matrix multiplication. Hence, (4) can be rewritten as

$$\hat{\mathbf{B}} = \sum_{i=0}^{1}\sum_{j=0}^{1}\mathbf{H}_{\mathbf{i,j}}\cdot\mathbf{B}_{\mathbf{i,j}}\cdot\mathbf{W}_{\mathbf{i,j}}, \tag{5}$$

where $\mathbf{X} = \mathrm{DCT}(\mathbf{x})$. Since the $\mathbf{H}_{\mathbf{i,j}}$ and $\mathbf{W}_{\mathbf{i,j}}$ terms are constant matrices, they are usually precomputed and stored in memory.

2.2. DCT decimation approaches

DCT decimation techniques take advantage of the fact that most of the DCT coefficients block energy is concentrated in the lower frequency band. Consequently, several video transcoding manipulations that have been proposed make use of this technique by discarding some high-order AC frequency DCT coefficients and retaining only a subset of the low-order terms. As a consequence, this approach has also been denoted as *modified inverse transformation and decimation (MITD)* [12] and has been particularly adopted in DCT-domain inverse motion compensation [13–15] and spatial-resolution downscaling [8, 23–26] schemes.

One example of such approach was presented by Dugad and Ahuja [23], who proposed an efficient DCT decimation scheme that extracts the (4×4) low-frequency DCT coefficients corresponding to each of the four (8×8) original blocks (see Figure 1). Each of these subblocks is then inverse DCT transformed, in order to obtain a subset of the original $(N\times N)$ pixels area that will represent the scaled version of the original block. The four (4×4) subblocks are then merged and combined together, in order to obtain an (8×8) pixels block.

This scheme can be formulated as follows: let $\mathbf{B}_{0,0}$, $\mathbf{B}_{0,1}$, $\mathbf{B}_{1,0}$ and $\mathbf{B}_{1,1}$ represent the four original (8×8) DCT coefficients blocks; $\mathbf{B}'_{0,0}$, $\mathbf{B}'_{0,1}$, $\mathbf{B}'_{1,0}$ and $\mathbf{B}'_{1,1}$ represent the four (4×4) low-frequency subblocks of $\mathbf{B}_{0,0}$, $\mathbf{B}_{0,1}$, $\mathbf{B}_{1,0}$, and $\mathbf{B}_{1,1}$, respectively; $\mathbf{b}'_{\mathbf{i,j}} = \mathrm{IDCT}(\mathbf{B}'_{\mathbf{i,j}})$, with $i, j \in \{0,1\}$. Then,

$$\mathbf{b}' = \begin{bmatrix}[\mathbf{b}'_{0,0}]_{4\times 4} & [\mathbf{b}'_{0,1}]_{4\times 4}\\ [\mathbf{b}'_{1,0}]_{4\times 4} & [\mathbf{b}'_{1,1}]_{4\times 4}\end{bmatrix}_{8\times 8} \tag{6}$$

is the downscaled version of

$$\mathbf{b} = \begin{bmatrix}[\mathbf{b}_{0,0}]_{8\times 8} & [\mathbf{b}_{0,1}]_{8\times 8}\\ [\mathbf{b}_{1,0}]_{8\times 8} & [\mathbf{b}_{1,1}]_{8\times 8}\end{bmatrix}_{16\times 16}. \tag{7}$$

To compute $\mathbf{B}' = \mathrm{DCT}(\mathbf{b}')$ directly from $\mathbf{B}'_{0,0}$, $\mathbf{B}'_{0,1}$, $\mathbf{B}'_{1,0}$, and $\mathbf{B}'_{1,1}$, Dugad and Ahuja [23] have proposed the usage of the following expression:

$$\begin{aligned}\mathbf{B}' &= \mathbf{C}_8\mathbf{b}'\mathbf{C}_8^t\\ &= \begin{bmatrix}\mathbf{C}_L & \mathbf{C}_R\end{bmatrix}\begin{bmatrix}\mathbf{C}_4^t\mathbf{B}'_{0,0}\mathbf{C}_4 & \mathbf{C}_4^t\mathbf{B}'_{0,1}\mathbf{C}_4\\ \mathbf{C}_4^t\mathbf{B}'_{1,0}\mathbf{C}_4 & \mathbf{C}_4^t\mathbf{B}'_{1,1}\mathbf{C}_4\end{bmatrix}\begin{bmatrix}\mathbf{C}_L^t\\ \mathbf{C}_R^t\end{bmatrix}\\ &= (\mathbf{C}_L\mathbf{C}_4^t)\mathbf{B}'_{0,0}(\mathbf{C}_L\mathbf{C}_4^t)^t + (\mathbf{C}_L\mathbf{C}_4^t)\mathbf{B}'_{0,1}(\mathbf{C}_R\mathbf{C}_4^t)^t\\ &\quad + (\mathbf{C}_R\mathbf{C}_4^t)\mathbf{B}'_{1,0}(\mathbf{C}_L\mathbf{C}_4^t)^t + (\mathbf{C}_R\mathbf{C}_4^t)\mathbf{B}'_{1,1}(\mathbf{C}_R\mathbf{C}_4^t)^t,\end{aligned} \tag{8}$$

where $\mathbf{C}_4$ is the 4-point DCT kernel matrix and $\mathbf{C}_L$ and $\mathbf{C}_R$ are, respectively, the four left and the four right columns of $\mathbf{C}_8$, the 8-point DCT kernel matrix.

2.3. Arbitrary downscaling algorithms

Besides the simplest half-scaling setups previously described, many applications have arisen which require arbitrary non-integer scaling factors ($\mathcal{S}$). From the digital signal processing point of view, an arbitrary-resize procedure using a scaling factor $\mathcal{S} = U/D$ (where U and D may take any nonnull relative prime integer values) can be accomplished by cascading an integer upscaling module (by a factor U), followed by an integer downscaling module (by a factor D).

Based on the DCT decimation technique, Dugad and Ahuja [23] have shown that the upscaling step can be efficiently implemented by padding with zeros, at the high frequencies, the DCT coefficients of the original image subblocks, in order to obtain the corresponding target $(N\times N)$ DCT coefficient blocks of the upscaled image. According to

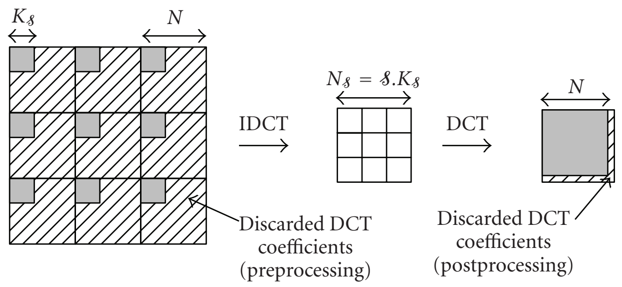

$K_{\mathcal{S}}$
N
IDCT
$N_{\mathcal{S}} = \mathcal{S}.K_{\mathcal{S}}$
DCT
N
Discarded DCT coefficients (preprocessing)
Discarded DCT coefficients (postprocessing)

FIGURE 2: Discarded DCT coefficients in arbitrary downscale DCT decimation algorithms.

Dugad, since each upsampled block will contain all the frequency content corresponding to its original subblocks, this approach provides better interpolation results when compared with the usage of bilinear interpolation algorithms.

Nevertheless, the same does not always happen in what concerns the implementation of the downscaling step using this approach, as it will be shown in the following. Meanwhile, several improved DCT decimation strategies have been presented [8, 24–26]. Some authors have even proposed the usage of optimized factorizations of the DCT kernel matrix, in order to reduce the involved computational complexity [25]. However, most of such proposals are only directly applied to scaling operations using a scaling factor that is a power of 2 ($\mathcal{S} = 2, 4, 8, 16$, etc.). Nevertheless, downscaling operations using any other arbitrary integer scaling factors are often required. As a consequence, in the last few years proposals have arisen in order to implement *DCT decimation* algorithms for any integer scale factor [7–11, 27]. However, not only are they directly influenced by the degradation effect resulting from the coefficient discard, but they often suffer from computational inefficiency on their processing, either by storing a large amount of data matrices [7] or by operating with large matrices [9–11, 27]. One of such proposals was recently presented by Patil et al. [27], who proposed a DCT-decimation approach based on simple matrix multiplications that processes each original DCT frame as a whole, without fragmenting the involved processing by the several macroblocks. However, in practical implementations such approach may lead to serious degradations in what concerns the processing efficiency, since the manipulation of such wide matrices may hardly be efficiently carried out in most current processing systems, namely, due to the inherent high cache missing rate that will be necessarily involved. Such degradation will be even more serious when the processing of high-resolution video sequences is considered. By using an alternative and somewhat simpler approach, Lee et al. [8] proposed an arbitrary downscaling technique by generalizing the previously described DCT decimation approach, in order to achieve arbitrary-size downscaling with scale factors ($\mathcal{S}$) other than powers of 2 (e.g., 3, 5, 7, etc.). Their methodology is illustrated in Figure 2 and can be described as follows:

(1) for each original block $\mathbf{B_{i,j}}$, retain the low-frequency $(K_{\mathcal{S}} \times K_{\mathcal{S}})$ DCT coefficients $\mathbf{B'_{i,j}}$, thus discarding the remaining AC frequency DCT coefficients, with $K_{\mathcal{S}}$ defined as $K_{\mathcal{S}} = \lceil N/\mathcal{S} \rceil$;

(2) inverse transform each subblock $\mathbf{B'_{i,j}}$ to the pixel domain, using $\mathbf{b'_{i,j}} = \mathbf{C}^t_{K_{\mathcal{S}}}(\mathbf{B'_{i,j}})\mathbf{C}_{K_{\mathcal{S}}}$, where $\mathbf{C}_{K_{\mathcal{S}}}$ is the $K_{\mathcal{S}}$-point DCT kernel matrix;

(3) concatenate $(\mathcal{S} \times \mathcal{S})$ subblocks, in order to form an $(N_{\mathcal{S}} \times N_{\mathcal{S}})$ pixels block $\mathbf{b}'$, with $N_{\mathcal{S}}$ defined as $N_{\mathcal{S}} = \mathcal{S} \cdot K_{\mathcal{S}}$:

$$\mathbf{b}' = \begin{bmatrix} \mathbf{b'_{0,0}} & \cdots & \mathbf{b'_{0,\mathcal{S}}} \\ \vdots & \ddots & \vdots \\ \mathbf{b'_{\mathcal{S},0}} & \cdots & \mathbf{b'_{\mathcal{S},\mathcal{S}}} \end{bmatrix}_{(N_{\mathcal{S}} \times N_{\mathcal{S}})}; \tag{9}$$

(4) compute $\mathbf{B}' = \mathrm{DCT}(\mathbf{b}') = \mathbf{C}_{N_{\mathcal{S}}}\mathbf{b}'\mathbf{C}^t_{N_{\mathcal{S}}}$, where $\mathbf{C}_{N_{\mathcal{S}}}$ is the $N_{\mathcal{S}}$-point DCT kernel matrix;

(5) extract the $(N \times N)$ low frequency DCT coefficients of $\mathbf{B}'$ (with $N = 8$), in order to obtain the (8×8) DCT-domain scaled block $\widehat{\mathbf{B}}$.

However, although this methodology is often claimed to provide better performance results than bilinear downscaling approaches in what concerns the obtained video quality [12, 23], it can be shown that such statement is not always true. In particular, when these generalized DCT decimation downscaling schemes are applied using a scaling factor other than an integer power of 2, it can be shown that the obtained video quality is clearly worse than the provided by the previously described pixel averaging approaches. The reason for the introduction of such degradation comes as a result of the additional DCT coefficients discarding procedure that is performed in step (5), described above (see Figure 2). Contrary to the first discarding step (performed in step (1)), this second discard of high-order AC frequency DCT coefficients only occurs for scaling factors other than integer powers of 2 and introduces serious block artifacts, mainly in image areas with complex textured regions. To better understand such phenomenon, in Table 1 it is presented the number of DCT coefficients that is considered along the implementation of this algorithm. As it can be seen, the number of discarded coefficients during the last processing step may be highly significative and its degradation effect will be thoroughly assessed in Section 4.

To overcome the introduction of this degradation by downscaling algorithms using any arbitrary integer scaling factor, a different approach is now proposed based on a

Table 1: Number of DCT coefficients considered by Lee et al.'s [8] arbitrary downscaling algorithm.

Scaling factor	$\mathcal{S}$	2	3	4	5	6	7	8
Number of preserved coefficients in each direction during preprocessing	$K_{\mathcal{S}} = \lceil N/\mathcal{S} \rceil$	4	3	2	2	2	2	1
Reconstructed downscaled block size	$N_{\mathcal{S}} = \mathcal{S} \cdot K_{\mathcal{S}}$	8	9	8	10	12	14	8
Number of discarded coefficients in each direction during post-processing	$N_{\mathcal{S}} - N$	0	1	0	2	4	6	0

highly efficient implementation of a pixel averaging downscaling technique. Such approach is described in the following section.

3. PROPOSED DOWNSCALING APPROACH

Considering an arbitrary integer scaling factor $\mathcal{S} = (\mathcal{S}_x, \mathcal{S}_y) \in \mathbb{N}^2$, where $\mathcal{S}_x$ and $\mathcal{S}_y$ are the horizontal and the vertical down-sizing ratios, respectively, the purpose of an arbitrary downscaling algorithm is to compute the $(N \times N)$ DCT encoded block corresponding to a set of $(\mathcal{S}_x \times \mathcal{S}_y)$ original blocks, each one with $(N \times N)$ DCT coefficients.

According to the previously described pixel averaging approach, a generalized arbitrary integer downscaling procedure can be formulated as follows: by denoting $\mathbf{b}$ as the pixels area corresponding to the set of $(\mathcal{S}_x \times \mathcal{S}_y)$ original blocks $\mathbf{b_{i,j}}$, each one with $(N \times N)$ pixels,

$$\mathbf{b} = \begin{bmatrix} [\mathbf{b_{0,0}}] & [\mathbf{b_{0,1}}] & \cdots & [\mathbf{b}_{0,\mathcal{S}_x-1}] \\ [\mathbf{b_{1,0}}] & [\mathbf{b_{1,1}}] & \cdots & [\mathbf{b}_{1,\mathcal{S}_x-1}] \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{b}_{\mathcal{S}_y-1,0}] & [\mathbf{b}_{\mathcal{S}_y-1,1}] & \cdots & [\mathbf{b}_{\mathcal{S}_y-1,\mathcal{S}_x-1}] \end{bmatrix}, \tag{10}$$

the downscaled $(N \times N)$ pixels block $(\hat{\mathbf{b}})$ can be obtained by multiplying $\mathbf{b}$ with the subsampling and filtering matrices $\mathbf{f}_{\mathcal{S}_x}$ and $\mathbf{f}_{\mathcal{S}_y}$ as follows:

$$\hat{\mathbf{b}} = \left(\frac{1}{\mathcal{S}_x \mathcal{S}_y}\right) \times \mathbf{f}_{\mathcal{S}_y} \cdot \mathbf{b} \cdot \mathbf{f}^{\mathbf{t}}_{\mathcal{S}_x}, \tag{11}$$

where $\mathbf{f}_{\mathcal{S}_q}$ is an $(N \times N\mathcal{S}_q)$ matrix with the following structure:

$$[\mathbf{f}_{\mathcal{S}_q}](i,j) = \begin{cases} 1, & \text{for } i = \left\lfloor \dfrac{j}{\mathcal{S}_q} \right\rfloor, \text{ with } j \in [0, N\mathcal{S}_q - 1] \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

These matrices are used to decimate the input image along the two dimensions. To simplify the description, from now on it will be adopted a common scaling factor for both the horizontal and vertical directions $(\mathcal{S} = \mathcal{S}_x = \mathcal{S}_y)$. Such simplification does not introduce any restriction or limitation in the described algorithm. As an example, the $\mathbf{f_3}$ matrix $(\mathcal{S} = 3)$, considering $N = 5$, is given by (13). This matrix may be used to perform image downscaling by a factor of 3: each set of (3×3) pixel blocks, each one composed by (5×5) pixels, is subsampled in order to obtain a single (5×5) pixels block,

$$\mathbf{f_3} = \Bigg[\underbrace{\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}}_{\mathbf{f}^0_3} \; \underbrace{\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}}_{\mathbf{f}^1_3} \; \underbrace{\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{matrix}}_{\mathbf{f}^2_3} \Bigg]. \tag{13}$$

However, the computation of (11) using the filtering matrices defined in (12) is usually difficult to handle, since it may involve the manipulation of large matrices. Furthermore, although these filtering matrices may seem reasonably sparse in the pixel domain, this does not happen when this filtering procedure is transposed to the DCT domain (as it was described in the previous section), leading to the storage of a significant amount of data corresponding to these precomputed filtering matrices. The computation of (11) is even harder to accomplish if we take into account that the $(N \times N)$ block structure adopted in image and video coding (usually with $N = 8$) requires that the several involved operations are performed directly on blocks with $(N \times N)$ elements, which makes this approach even more difficult to be adopted.

To circumvent all these issues, a different and more efficient approach is now proposed. Firstly, by splitting the $\mathbf{f}_{\mathcal{S}}$ matrix into $\mathcal{S}$ submatrices $\mathbf{f}^{\mathbf{0}}_{\mathcal{S}}, \mathbf{f}^{\mathbf{1}}_{\mathcal{S}}, \ldots, \mathbf{f}^{\mathcal{S}-\mathbf{1}}_{\mathcal{S}}$, each one with $(N \times N)$ elements, the computation of (11) can be decomposed in a series of product terms and take a form entirely similar to (1):

$$\hat{\mathbf{b}} = \frac{1}{\mathcal{S}^2}\left(\mathbf{f}^{\mathbf{0}}_{\mathcal{S}}\mathbf{b_{00}}\mathbf{f}^{\mathbf{0}^{\mathbf{t}}}_{\mathcal{S}} + \mathbf{f}^{\mathbf{0}}_{\mathcal{S}}\mathbf{b_{01}}\mathbf{f}^{\mathbf{1}^{\mathbf{t}}}_{\mathcal{S}} + \cdots + \mathbf{f}^{(\mathcal{S}-\mathbf{1})}_{\mathcal{S}}\mathbf{b}_{(\mathcal{S}-\mathbf{1})(\mathcal{S}-\mathbf{1})}\mathbf{f}^{(\mathcal{S}-\mathbf{1})^{\mathbf{t}}}_{\mathcal{S}}\right) \tag{14}$$

or equivalently,

$$\hat{\mathbf{b}} = \frac{1}{\mathcal{S}^2} \sum_{i=0}^{\mathcal{S}-1} \sum_{j=0}^{\mathcal{S}-1} \mathbf{f}^{\mathbf{i}}_{\mathcal{S}} \cdot \mathbf{b_{ij}} \cdot \mathbf{f}^{\mathbf{j}^{\mathbf{t}}}_{\mathcal{S}}, \tag{15}$$

where $\mathbf{b_{ij}}$ are the several input blocks involved in the downscaling operation, directly obtained from the input video sequence. In the bottom of (13), it was represented the set of three $(N \times N)$ $\mathbf{f}^x_{\mathcal{S}}$ submatrices, for the case with $\mathcal{S} = 3$ and $N = 5$, with $x \in [0, \mathcal{S} - 1]$.

Secondly, the computation of these terms can be greatly simplified if the sparse nature, and the high number of zeros of each $\mathbf{f}_{\mathcal{S}}^{x}$ matrix are taken into account. In particular, it can be shown that each $\mathbf{f}_{\mathcal{S}}^{\mathbf{i}} \cdot \mathbf{b_{ij}} \cdot \mathbf{f}_{\mathcal{S}}^{\mathbf{j}\,\mathbf{t}}$ term only contributes to the computation of a restricted subset of pixels of the subsampled block ($\hat{\mathbf{b}}$), within an area delimited by lines ($l_{\min}(i) : l_{\max}(i)$) and by columns ($c_{\min}(j) : c_{\max}(j)$), where

$$\begin{aligned} l_{\min}(i) &= \left\lfloor \frac{i * N}{\mathcal{S}} \right\rfloor, & l_{\max}(i) &= \left\lfloor \frac{i * N + (N-1)}{\mathcal{S}} \right\rfloor, \\ c_{\min}(j) &= \left\lfloor \frac{j * N}{\mathcal{S}} \right\rfloor, & c_{\max}(j) &= \left\lfloor \frac{j * N + (N-1)}{\mathcal{S}} \right\rfloor, \end{aligned} \tag{16}$$

with $i, j \in [0, \mathcal{S} - 1]$. By denoting the contribution of each block $\mathbf{b_{i,j}}$ to the sampled pixels block $\hat{\mathbf{b}}$ by the $(n_l(i) \times n_c(j))$ matrix $\overline{\mathbf{p_{i,j}}}$, one has

$$\overline{\mathbf{p_{i,j}}} = \underbrace{\overline{\mathbf{f}_{\mathcal{S}}^{\mathbf{i}}} \cdot \mathbf{b_{i,j}} \cdot \overline{\mathbf{f}_{\mathcal{S}}^{\mathbf{j}}}^{\mathbf{t}}}_{n_l(i) \times n_c(j) \text{ matrix}}, \tag{17}$$

where $\overline{\mathbf{f}_{\mathcal{S}}^{\mathbf{i}}}$ and $\overline{\mathbf{f}_{\mathcal{S}}^{\mathbf{j}}}$ are $(n_l(i) \times N)$ and $(n_c(j) \times N)$ matrices, respectively, with $n_l(i) = l_{\max}(i) - l_{\min}(i) + 1$ and $n_c(j) = c_{\max}(j) - c_{\min}(j) + 1$, that are obtained from $\mathbf{f}_{\mathcal{S}}^{\mathbf{i}}$ and $\mathbf{f}_{\mathcal{S}}^{\mathbf{j}}$ by only considering the lines with nonnull elements (see dashed boxes in (13)).

The resulting $(N \times N)$ pixels sampled block ($\hat{\mathbf{b}}$) is obtained by summing up the contributions of all these terms:

$$\hat{\mathbf{b}} = \frac{1}{\mathcal{S}^2} \cdot \left(\sum_{i=0}^{\mathcal{S}-1} \sum_{j=0}^{\mathcal{S}-1} \mathbf{p_{i,j}} \right), \tag{18}$$

where

$$[\mathbf{p_{i,j}}](l, c) = \begin{cases} \overline{\mathbf{p_{i,j}}}, & \text{for} \begin{cases} l_{\min}(i) \le l \le l_{\max}(i), \\ c_{\min}(j) \le c \le c_{\max}(j) \end{cases} \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

with $0 \le l, c \le (N-1)$. By applying such decomposition, the overall number of computations is greatly reduced, since most of the null terms of the $\mathbf{f}_{\mathcal{S}}$ matrices are not considered any more.

It is also worth noting that some pixels of the sampled block ($\hat{\mathbf{b}}$) may be obtained from several of these product-terms. Such situation will occur whenever the set of $\mathcal{S}$ non-null elements of a given line of the $\mathbf{f}_{\mathcal{S}}$ matrix is split into two distinct $\mathbf{f}_{\mathcal{S}}^{x}$ submatrices (see (13)). In such situation, the value of the output pixel will be the sum of the mutual contribution of adjacent $\mathbf{b_{i,j}}$ blocks, each one with $(N \times N)$ pixels. One example of such scenario can be observed in the previously described case with $\mathcal{S} = 3$ and $N = 5$ (see $\mathbf{f_3}$ matrix in (13)) and illustrated in Figure 3. While the pixels of the first row of the sampled $(N \times N)$ output block are obtained with only the subset of blocks $\{\mathbf{b_{00}}, \mathbf{b_{01}}, \mathbf{b_{02}}\}$, the pixels of the second row are the result of the mutual contribution of the set of blocks $\{\mathbf{b_{00}}, \mathbf{b_{01}}, \mathbf{b_{02}}, \mathbf{b_{10}}, \mathbf{b_{11}}, \mathbf{b_{12}}\}$. The same situation can be verified in what concerns the columns of the output block: while the first column is obtained with blocks $\{\mathbf{b_{00}}, \mathbf{b_{10}}, \mathbf{b_{20}}\}$, the second column is computed with blocks $\{\mathbf{b_{i0}}, \mathbf{b_{i1}}\}$, with $i \in \{0, \dots, (\mathcal{S} - 1)\}$.

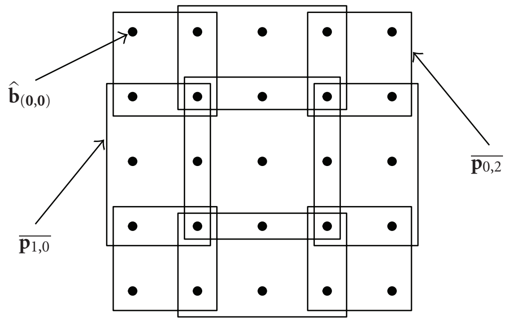

$\hat{\mathbf{b}}_{(0,0)}$
$\overline{\mathbf{p}_{0,2}}$
$\overline{\mathbf{p}_{1,0}}$

Figure 3: Contributions of the several blocks of the original image ($\overline{p_{i,j}}$) to the final value of each pixel of the sampled block $\hat{\mathbf{b}}$ ($\mathcal{S} = 3, N = 5$).

A particular situation also occurs whenever the original frame dimension in any of its directions is not an integer multiple of $\mathcal{S}$. In such case, the pixels of the last column (or line) cannot be obtained from the $\mathcal{S}^2$ input pixels, since only a subset of pixels remains to be considered in that line or column. To overcome such situation, the corresponding averaging weights should be adjusted to the available number of pixels at the end of that line ($W_c - \mathcal{S} \cdot \lfloor W_c/\mathcal{S} \rfloor$) or column ($W_l - \mathcal{S} \cdot \lfloor W_l/\mathcal{S} \rfloor$), where W_c and W_l denote the number of columns and lines of the original image. As an example, the last sampled pixel of a given line should be computed as

$$\hat{\mathbf{b}}\left(:, \left\lfloor \frac{W_c}{\mathcal{S}} \right\rfloor \right) = \frac{1}{\mathcal{S}(W_c - \mathcal{S} \cdot \lfloor W_c/\mathcal{S} \rfloor)} \times p_{i, \lfloor W_c/\mathcal{S} \rfloor}. \tag{20}$$

This adjustment can be compensated a posteriori, by multiplying the pixels of the last column of the sampled block ($\hat{\mathbf{b}}$) by

$$\hat{\mathbf{b}}\left(:, \left\lfloor \frac{W_c}{\mathcal{S}} \right\rfloor \right) = \left[\frac{\mathcal{S}}{W_c - \mathcal{S} \cdot \lfloor W_c/\mathcal{S} \rfloor} \right] \times \hat{\mathbf{b}}\left(:, \left\lfloor \frac{W_c}{\mathcal{S}} \right\rfloor \right). \tag{21}$$

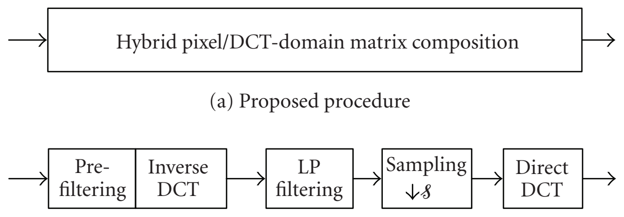
The same applies for the vertical direction of the sampled image.

3.1. Hybrid downscaling algorithm

As it was referred in Section 2, since the DCT is an unitary orthonormal transform, it is distributive to matrix multiplication. Consequently, the described scaling procedure can be directly performed in the DCT domain and still provide the previously mentioned computational advantages. By considering the matrix decomposition to compute the DCT coefficients of a given pixels block $x : \mathbf{X} = \mathbf{C} \cdot \mathbf{x} \cdot \mathbf{C^t}$, (18) can be directly computed in the DCT domain as

$$\hat{\mathbf{B}} = \mathbf{C} \cdot \hat{\mathbf{b}} \cdot \mathbf{C^t} = \frac{1}{\mathcal{S}^2} \cdot \mathbf{C} \cdot \left(\sum_{i=0}^{\mathcal{S}-1} \sum_{j=0}^{\mathcal{S}-1} \mathbf{p_{i,j}} \right) \cdot \mathbf{C^t}. \tag{22}$$

The computation of this expression may be greatly simplified if the definition of matrices $\mathbf{p_{i,j}}$ in (19) is taken into



(a) Proposed procedure

(b) Equivalent approach

FIGURE 4: DCT-domain frame scaling procedure.

account. In particular, the computation of its $(n_l(i) \times n_c(j))$ nonnull elements ($\overline{\mathbf{p_{i,j}}}$) can be carried out as follows:

$$\overline{\mathbf{p_{i,j}}} = \overline{\mathbf{f}^{\mathbf{i}}_{\mathcal{S}}} \cdot \mathbf{b_{i,j}} \cdot \overline{\mathbf{f}^{\mathbf{j}}_{\mathcal{S}}}^{\mathbf{t}} = \overline{\mathbf{f}^{\mathbf{i}}_{\mathcal{S}}} \cdot \mathbf{C^t} \cdot \mathbf{B_{i,j}} \cdot \mathbf{C} \cdot \overline{\mathbf{f}^{\mathbf{j}}_{\mathcal{S}}}^{\mathbf{t}}. \qquad (23)$$

By denoting the product $\overline{\mathbf{f}^{\mathbf{i}}_{\mathcal{S}}} \cdot \mathbf{C^t}$ by the $(n_l(i) \times N)$ matrix $\overline{\mathbf{F}^{\mathbf{i}}_{\mathcal{S}}}$ and the product $\overline{\mathbf{f}^{\mathbf{j}}_{\mathcal{S}}} \cdot \mathbf{C^t}$ by the $(n_c(j) \times N)$ matrix $\overline{\mathbf{F}^{\mathbf{j}}_{\mathcal{S}}}$, the above expression can be represented as

$$\overline{\mathbf{p_{i,j}}} = \underbrace{\overline{\mathbf{F}^{\mathbf{i}}_{\mathcal{S}}} \cdot \mathbf{B_{i,j}} \cdot \overline{\mathbf{F}^{\mathbf{j}}_{\mathcal{S}}}^{\mathbf{t}}}_{n_l(i) \times n_c(j) \text{ matrix}}, \qquad (24)$$

where $\mathbf{B_{i,j}}$ is the $(N \times N)$ DCT coefficients block directly obtained from the partially decoded bit stream. Since all the $\overline{\mathbf{F}^{\mathbf{x}}_{\mathcal{S}}}$ terms (with $0 \leq x \leq \mathcal{S} - 1$) are constant matrices, they can be precomputed and stored in memory.

The overall complexity of the described procedure can still be further reduced if the usage of *partial DCT information* [13–15] techniques is considered, as it will be shown in the following.

```
I-Initialization:
  Compute and store in memory the set of $\overline{\mathbf{F}^{\mathbf{x}}_{\mathcal{S}}}$ matrices;
II-Computation:
  for lin$S = 0$ to $\left(\frac{W_l}{\mathcal{S}} - 1\right)$, lin$S+ = N$ do
    for col$S = 0$ to $\left(\frac{W_c}{\mathcal{S}} - 1\right)$, col$S+ = N$ do
      for $l = 0$ to $(\mathcal{S} - 1)$ do
        for $c = 0$ to $(\mathcal{S} - 1)$ do
          $[\overline{\mathbf{p_{l,c}}}]_{n_l \times n_c} = [\overline{\mathbf{F}^{\mathbf{l}}_{\mathcal{S}}}]_{n_l \times K} \cdot [\widetilde{\mathbf{B}}_{\mathbf{l,c}}]_{K \times K} \cdot [\overline{\mathbf{F}^{\mathbf{c}}_{\mathcal{S}}}^{t}]_{K \times n_c}$
          $\widehat{\mathbf{b}}(l_{\min} : l_{\max}, c_{\min} : c_{\max}) + = \frac{1}{\mathcal{S}^2} [\overline{\mathbf{p_{i,j}}}]_{n_l \times n_c}$
        end for
      end for
      $[\widehat{\mathbf{B}}]_{N \times N} = [\mathbf{C}]_{N \times N} \cdot [\widehat{\mathbf{b}}]_{N \times N} \cdot [\mathbf{C^t}]_{N \times N}$
    end for
  end for
```

FIGURE 5: Proposed hybrid downscaling algorithm.

3.2. DCT-domain prefiltering for complexity reduction

The complexity advantages of the previously described hybrid downscaling scheme can be regarded as the result of an efficient implementation of the following cascaded processing steps: *inverse DCT*, *lowpass filtering* (averaging), *subsampling,* and *direct DCT* (see Figure 4). However, the efficiency of this procedure can be further improved by noting that the signal component corresponding to most of the high-order AC frequency DCT coefficients, obtained from the first implicit processing step (*inverse DCT*), is discarded as the result of the second step (*lowpass filtering*). Hence, the overall complexity of this scheme can be significantly reduced by introducing a lowpass *prefiltering* stage in the inverse DCT processing step, which is directly implemented by only considering a subset of the original DCT coefficients. By denoting K as the maximum bandwidth of this lowpass prefilter, given by the highest line/column index of the considered DCT coefficients, only the coefficients $\widetilde{\mathbf{B}}_{\mathbf{i,j}}(\mathbf{m},\mathbf{n}) = \{\mathbf{B_{i,j}}(\mathbf{m},\mathbf{n}) : m, n \leq K\}$ will be used for the inverse DCT operation. In practice, this prefiltering can be formulated as follows:

$$\begin{aligned} \widetilde{\mathbf{B}}_{\mathbf{i,j}} &= \begin{bmatrix} [\mathbf{I}]_{K\times K} & 0 \\ 0 & 0 \end{bmatrix} \cdot \mathbf{B_{i,j}} \cdot \begin{bmatrix} [\mathbf{I}]_{K\times K} & 0 \\ 0 & 0 \end{bmatrix}^t \\ &= \begin{bmatrix} [\mathbf{B_{i,j}}]_{K\times K} & 0 \\ 0 & 0 \end{bmatrix}, \end{aligned} \qquad (25)$$

where $[\mathbf{I}]_{K\times K}$ is the $(K\times K)$ identity matrix corresponding to the considered prefilter and $[\mathbf{B}_{i,j}]_{K\times K}$ is a $(K \times K)$ submatrix of $\mathbf{B_{i,j}}$, obtained by extracting the $(K \times K)$ lower-order DCT coefficients. Thus, the representative contribution of $\mathbf{B_{i,j}}$ to the output pixels $\overline{\mathbf{p_{i,j}}}$ (see (24)) can be obtained as

$$[\overline{\mathbf{p_{i,j}}}]_{n_l(i)\times n_c(j)} = \left[\overline{\mathbf{F}^{\mathbf{i}}_{\mathcal{S}}}\right]_{n_l(i)\times K} \cdot \left[\widetilde{\mathbf{B}}_{\mathbf{i,j}}\right]_{K\times K} \cdot \left[\overline{\mathbf{F}^{\mathbf{j}}_{\mathcal{S}}}^{t}\right]_{K\times n_c(j)}. \qquad (26)$$

By adopting this scheme, the proposed procedure provides a full control over the resulting accuracy level in order to fulfill any real-time requirements, thus providing a trade-off between speed and accuracy. Furthermore, by considering that the $\mathbf{B_{i,j}}$ matrices usually have most of their high-order AC frequency coefficients equal to zero and provided that K is not too small, the distortion resulting from this scheme is often negligible, as it will be shown in Section 4.

3.3. Algorithm

In Figure 5, it is formally stated the proposed hybrid downscaling algorithm, where (lin$S$, col$S$) are the block coordinates within the target (scaled) image; (l, c) are the coordinates within the set of $\mathcal{S}^2$ blocks being sampled; and $l_{\min}$, $l_{\max}$, $c_{\min}$, and $c_{\max}$, defined in (16), respectively, are the bounding coordinates of the target block area affected by each iteration.

To evaluate the computational complexity of the proposed algorithm, the number of multiplications ($\mathcal{M}$) required

Table 2: Comparison of the several considered downscaling approaches in what concerns the involved computational cost.

Algorithm	DCT coefficents	$\mathcal{M}$	Comparison
CPAT	N	$2N$	$\frac{\mathcal{M}(\texttt{HDT})}{\mathcal{M}(\texttt{CPAT\_N})} \propto \mathcal{O}\left(\frac{1}{\mathcal{S}}\right)$
DDT	K	$\frac{2K^3}{N^2}(\mathcal{S}+1)$	$\frac{\mathcal{M}(\texttt{HDT})}{\mathcal{M}(\texttt{DDT})} \propto \mathcal{O}\left(\frac{1}{\mathcal{S}^2}\right)$
HDT	K	$\frac{KN\mathcal{S}(K+4)+2(N^3+K^2\mathcal{S}^2)}{N^2\mathcal{S}^2}$	1

to process each of the $(W_c \times W_l)$ pixels of the original frame was considered as the main figure of merit. Furthermore, to assess the provided computational advantages, the following different downscaling algorithms were also considered and their computational costs were evaluated, as fully described in the appendix section:

(i) *cascaded pixel averaging transcoder* (CPAT), as depicted in Figure 4(b), where the filtering and sub-sampling processing steps are entirely implemented in the pixel domain, by firstly decoding the whole set of DCT coefficients received from the incoming video stream;
(ii) *DCT decimation transcoder* (DDT) for arbitrary integer scaling factors, as formulated by Lee et al. [8] and described in Section 2.3;
(iii) *hybrid downscaling transcoder* (HDT), corresponding to the proposed algorithm.

In Table 2, it is presented the obtained comparison in what concerns the involved computational cost, both in terms of the adopted scaling factor $(\mathcal{S})$ and of the considered number of DCT coefficients (K). This comparison clearly evidences the complexity advantages provided by the proposed algorithm when compared with other considered approaches and, in particular, with the DCT decimation transcoder (DDT). Such advantages are even more significant when higher scaling factors are considered, as it will be demonstrated in the following section.

4. EXPERIMENTAL RESULTS

Video transcoding structures for spatial downscale comprise several different stages that must be implemented in order to resize the incoming video sequence. In fact, while in INTRA-type images only the space-domain information corresponding to the DCT coefficients blocks has to be downscaled, in INTER-type frames the downscale transcoder must also to take into account several processing tasks, other than the described down-sampling of the DCT blocks, as a result of the adopted temporal prediction mechanism. Some of such tasks involve the reusage and composition of the decoded motion vectors, scaling of the composited motion vectors, refinement of the scaled motion vectors, computation of the new prediction difference obtained by motion compensation, and so forth. All of such processing steps have been jointly or separately studied in the last few years [2, 3].

This manuscript focuses solely on the proposal of an efficient computational scheme to downscale the DCT coefficients blocks decoded from the incoming video stream by any arbitrary integer scaling factor. As it was previously stated, this task is a fundamental operation in most video downscaling transcoders and has been treated by several other proposals presented up to now. The evaluation of its performance was carried out by integrating the proposed algorithm in a reference closed-loop H.263 [28] video transcoding system, as shown in Figure 6. In this transcoding architecture, both the motion compensation (MC-DCT) and the motion estimation (ME-DCT) modules were implemented in the DCT domain. In particular, the motion estimation module of the encoding part of the transcoder implements a DCT-domain least squares motion reestimation algorithm, by considering a ± 1 pixel search range [4]. By adopting such structure, the encoder loop may compute a new reduced-resolution residual, providing a realignment of the predictive and residual components and thus minimizing the involved drift [17]. Nevertheless, to isolate the proposed algorithm from other encoding mechanisms (such as motion estimation/compensation) that could interfere in this assessment, a first evaluation considering the provided static video quality using solely INTRA-type images was carried out in Section 4.2. An additional evaluation that also considers its real performance when processing video sequences that apply the traditional temporal prediction mechanisms was carried out in Section 4.3.

The implemented system was applied in the scaling of a set of several CIF benchmark video sequences (*Akiyo*, *Silent*, *Carphone*, *Table-tennis,* and *Mobile*) with different characteristics and using different scaling factors $(\mathcal{S})$. Although some of the presented results were obtained using the *Mobile* video sequence and a quantization setup with $Q = 4$, the algorithm was equally assessed with all the considered video sequences and using a wide range of quantization steps, leading to entirely equivalent results. For all these experiments, it was considered the block size (N) adopted by most image and video coding standards, with $N = 8$ [28].

In Figure 7, it is represented the first frame of both the input and output video streams, considering the *Mobile* video sequence and $\mathcal{S} = 2, 3, 4,$ and 5. To evaluate the influence of the video scaling on the output bit stream, the same format (CIF) was adopted for both video sequences, by filling the remaining area of the output frame with null pixels. By doing

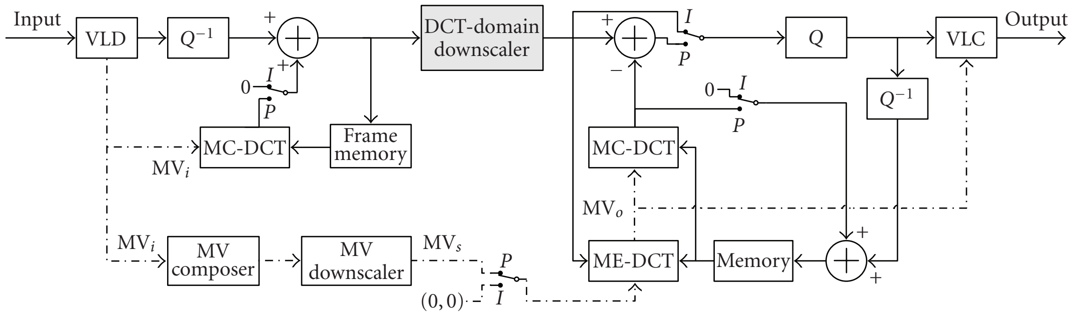


FIGURE 6: Integration of the proposed DCT-domain downscaling algorithm in an H.263 video transcoder.

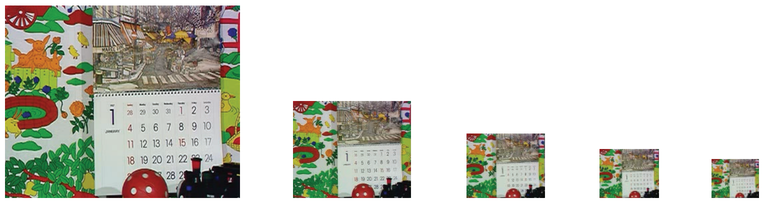


FIGURE 7: Space scaling of the CIF *Mobile* video sequence ($Q = 4$): (a) original frame; (b) $\mathcal{S} = 2$; (c) $\mathcal{S} = 3$; (d) $\mathcal{S} = 4$; (e) $\mathcal{S} = 5$.

so, not only do the two video streams share a significant amount of the variable length coding (VLC) parameters, thus simplifying their comparison, but it also provides an easy encoding of the scaled sequences, since their dimensions are often noncompliant with current video coding standards. Nevertheless, only the representative area corresponding to the scaled image was actually considered to evaluate the output video quality (PSNR) and drift. At this respect, several different approaches could have been adopted to evaluate this PSNR performance. One methodology that has been adopted by several authors is to implement and cascade an up-scaling and a down-scaling transcoders, in order to compare the reconstructed images at the full-scale resolution [23]. However, since such approach also introduces a nonnegligible degradation effect associated with the auxiliary up-scaling stage, it was not adopted in the presented experimental setup. As a consequence, the PSNR quality measure was calculated by comparing each scaled frame (obtained with each algorithm under evaluation), with a corresponding reference scaled frame, that was carefully computed in order to avoid the influence of any lossy processing step related to the encoding algorithm. An accurate quantization-free pixel filtering and down-sampling scheme was specially implemented for this specific purpose. This solution has proved to be a quite satisfactory alternative when compared with other possible approaches to compute the scaled reference frame (such as DCT-decimation), since it may provide a precise control over the inherent filtering process.

In the following, the proposed algorithm will be compared with the remaining considered downscaling algorithms, by considering several different evaluation metrics, namely, the *computational cost*, the *static video quality*, the introduced *drift,* and the resulting *bit rate*.

4.1. Computational cost

In Table 3(a), it is represented the comparison of the proposed HDT algorithm with the pixel-domain transcoder (CPAT) and the DCT decimation transcoder (DDT) in what concerns the involved computational complexity. As it was mentioned before, such computational cost was evaluated by counting the total amount of multiplication operations ($\mathcal{M}$) that are required to implement the downscaling procedure. In order to obtain comparison results as fair as possible, all the involved algorithms adopted the same number of DCT coefficients (K) for each of these comparisons and were implemented for several integer scaling factors ($\mathcal{S}$).

The presented results evidence the clear computational advantages provided by the proposed scheme to downscale the input video sequences by any arbitrary integer scaling factor. In particular, when compared with the DCT decimation transcoder (DDT), the HDT approach presented more significant advantages for scaling factors other than integer powers of 2, leading to a reduction of the computational cost as high as 5 ($\mathcal{S} = 7$). Such phenomenon was already expected and is a direct consequence of the computational inefficiency inherent to the postprocessing discarding stage of the DDT algorithm, illustrated in Figure 2. This computational advantage will be even more significant for higher values of the difference $\mathcal{S} - 2^{\lfloor \log_2 \mathcal{S} \rfloor}$. The presented results also evidence the clear computational advantage provided by the

Table 3: Computational cost comparison of the several considered downscaling algorithms (CIF mobile video sequence, $Q = 4$).

A. Variation of the algorithms computational cost with the scaling factor ($\mathcal{S}$)

$\mathcal{S}$	2	3	4	5	6	7	8	9	10	K
$\frac{\mathcal{M}(\texttt{HDT})}{\mathcal{M}(\texttt{CPAT})}$	0.5	0.3	0.2	0.2	0.2	0.2	0.1	0.1	0.1	$K_{\texttt{HDT}} = K_{\texttt{CPAT}} = N$
$\frac{\mathcal{M}(\texttt{HDT})}{\mathcal{M}(\texttt{DDT})}$	0.9	0.7	0.9	0.5	0.3	0.2	0.9	0.7	0.5	$K_{\texttt{HDT}} = K_{\texttt{DDT}} = \left\lceil \frac{N}{\mathcal{S}} \right\rceil$

B. Variation of the algorithms computational cost with the number of considered DCT coefficients (K)

		K									K							
	$\mathcal{S}$	8	7	6	5	4	3	2	1	$\mathcal{S}$	8	7	6	5	4	3	2	1
$\mathcal{M}(\texttt{CPAT})$		30.4	—	—	—	—	—	—	—		24.8	—	—	—	—	—	—	—
$\mathcal{M}(\texttt{HDT})$	2	14.8	13.0	11.4	10.1	8.9	7.9	7.1	6.4	6	4.1	3.4	2.7	2.2	1.7	1.3	1.0	0.8
$\mathcal{M}(\texttt{DDT})$		—	—	—	—	9.8	—	—	—		—	—	—	—	—	—	3.0	—
$\mathcal{M}(\texttt{CPAT})$		27.0	—	—	—	—	—	—	—		24.7	—	—	—	—	—	—	—
$\mathcal{M}(\texttt{HDT})$	3	9.3	8.0	6.8	5.7	4.8	4.1	3.5	3.1	7	4.0	3.3	2.6	2.1	1.6	1.2	0.9	0.8
$\mathcal{M}(\texttt{DDT})$		—	—	—	—	—	5.6	—	—		—	—	—	—	—	—	4.1	—
$\mathcal{M}(\texttt{CPAT})$		25.7	—	—	—	—	—	—	—		24.5	—	—	—	—	—	—	—
$\mathcal{M}(\texttt{HDT})$	4	5.3	4.5	3.8	3.2	2.7	2.3	2.0	1.7	8	2.1	1.8	1.4	1.2	0.9	0.7	0.6	0.5
$\mathcal{M}(\texttt{DDT})$		—	—	—	—	—	—	2.2	—		—	—	—	—	—	—	—	0.6
$\mathcal{M}(\texttt{CPAT})$		25.2	—	—	—	—	—	—	—		24.3	—	—	—	—	—	—	—
$\mathcal{M}(\texttt{HDT})$	5	5.4	4.4	3.6	2.9	2.3	1.9	1.5	1.3	9	3.2	2.6	2.0	1.5	1.1	0.8	0.5	0.4
$\mathcal{M}(\texttt{DDT})$		—	—	—	—	—	—	2.7	—		—	—	—	—	—	—	—	0.6

proposed scheme over the trivial pixel-domain approach using the whole set of DCT coefficients (CPAT).

Table 3(b) presents the variation of the computational cost of the considered schemes when a different number of DCT coefficients (K) are used by the proposed algorithm to downscale the input frame using several scaling factors $\mathcal{S}$. For such experimental setups, the pixel-domain transcoder (CPAT) adopted the whole set of DCT coefficients, while the DCT decimation transcoder (DDT) adopted $K = \lceil N/\mathcal{S} \rceil$ coefficients, as defined in [8]. As it was predicted before (see Table 2), the computational cost of the proposed HDT algorithm significantly decreases when the number of considered DCT coefficients decreases.

The presented results also evidence a direct consequence of the computational advantage provided by the proposed algorithm: for the same amount of computations ($\mathcal{M}$) and a given scaling factor ($\mathcal{S}$), the proposed algorithm is able to process a greater amount of decoded DCT coefficients (K) than the DCT-decimation transcoder (DDT). This fact can be easily observed for the transcoding setup using $\mathcal{S} = 3$, illustrated in Table 3(b). By approximately using the same number of operations, the DCT decimation transcoder processes only $K^2 = 9$ DCT coefficients of each block, while the proposed transcoder may process $K^2 = 25$ coefficients. As it will be shown in the following, such advantage will allow this algorithm to obtain scaled images with greater PSNR values in transcoding systems with restricted computational resources.

4.2. Static video quality

To isolate the proposed algorithm from other processing issues (such as motion vector scaling and refinement, drift compensation, predictive motion compensation, etc.), a first evaluation and assessment of the considered algorithms was performed using solely INTRA-type images. The comparison of such static video quality performances will provide the means to better understand the advantages of the proposed approach, by focusing the attention on the most important aspects under analysis, which are the accuracy and the computational cost of the spatial downscaling algorithms. A dynamic evaluation of the obtained video quality, by considering the inherent drift that is introduced when temporal prediction schemes are applied, will be presented in the following subsection.

Table 4 presents the PSNR measure that was obtained after the space scaling operation over the *Mobile* video sequence, considering a quantization setup with $Q = 4$. Several different scaling factors ($\mathcal{S}$) and number of considered DCT coefficients (K) were used in these implemented setups. Similar results were also obtained for all the remaining video sequences and quantization steps, evidencing that the overall quality of the resulting sequences is better when the proposed HDT algorithm is applied. These performance results were also thoroughly validated by undergoing a perceptual evaluation of the resulting video sequences using several different observers who have confirmed the obtained quality levels.

The first observation that should be retained from these results is the fact that the proposed algorithm is consistently better than the trivial cascaded pixel-domain architec-

Table 4: Comparison of the PSNR quality level [dB] obtained with the several considered downscaling algorithms (CIF mobile video sequence, $Q = 4$).

		K									K							
	$\mathcal{S}$	8	7	6	5	4	3	2	1	$\mathcal{S}$	8	7	6	5	4	3	2	1
CPAT		36.0	—	—	—	—	—	—	—		36.2	—	—	—	—	—	—	—
HDT	2	36.5	36.4	35.2	31.3	31.3	24.6	21.5	18.6	6	36.8	36.8	36.8	36.5	36.5	34.8	32.0	24.6
DDT		—	—	—	—	31.4	—	—	—		—	—	—	—	—	—	30.2	—
CPAT		36.1	—	—	—	—	—	—	—		36.3	—	—	—	—	—	—	—
HDT	3	36.7	36.6	36.3	35.6	32.8	28.4	24.8	20.7	7	36.7	36.7	36.7	36.4	35.4	34.1	31.5	25.2
DDT		—	—	—	—	—	27.9	—	—		—	—	—	—	—	—	28.6	—
CPAT		36.2	—	—	—	—	—	—	—		36.3	—	—	—	—	—	—	—
HDT	4	36.7	36.6	36.6	36.0	36.0	32.5	32.5	22.0	8	37.0	37.0	37.0	37.0	37.0	37.0	37.0	37.0
DDT		—	—	—	—	—	—	32.6	—		—	—	—	—	—	—	—	37.0
CPAT		36.1	—	—	—	—	—	—	—		36.3	—	—	—	—	—	—	—
HDT	5	36.7	36.7	36.5	35.9	34.8	33.8	29.5	23.6	9	37.0	37.0	36.9	36.6	36.0	35.4	34.2	27.0
DDT		—	—	—	—	—	—	28.6	—		—	—	—	—	—	—	—	28.9

Table 5: PSNR gains provided by the proposed approach over the DDT algorithm when the number of considered DCT coefficients (K) is adjusted, so that both schemes make use of the same computational resources.

$\mathcal{S}$	2	3	4	5	6	7	8	9
K_{DDT}	4	3	2	2	2	2	1	1
K_{HDT}	5	5	3	5	6	8	2	2
ΔPSNR	−0.1 dB	+7.7 dB	−0.1 dB	+7.3 dB	+6.6 dB	+8.1 dB	+0.0 dB	+5.3 dB

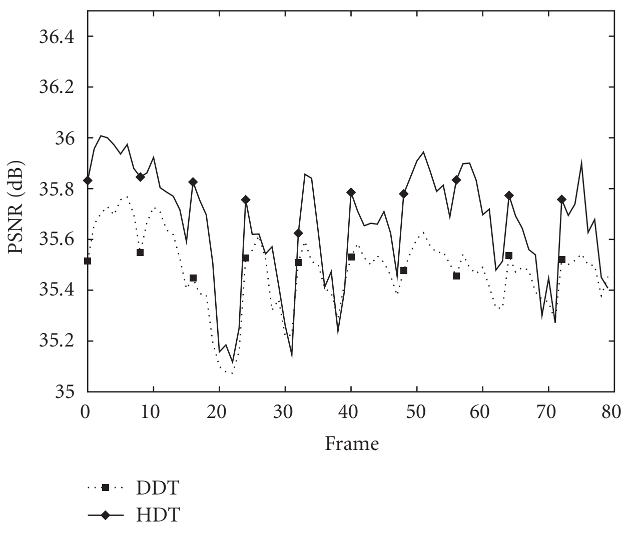
ture (CPAT) for the whole range of considered scaling factors. It should be noted, however, that these better results are not directly owed to the scaling algorithm itself. In fact, when the whole set of decoded DCT coefficients is considered ($K = N$), these two algorithms actually make use of quite similar down-sampling filters. Nevertheless, by processing the incoming blocks of DCT coefficients directly in the DCT domain, the proposed algorithm reduces the total number of arithmetic operations involved in the scaling, thus reducing the inherent degradation influence of round-off and truncation errors.

The second observation that is worth noting about the HDT algorithm is the expected decrease of the PSNR measures when the number of discarded coefficients increases. Although such decrease may be negligible for greater scaling factors, its importance is highly significant for smaller scalings of the original image.

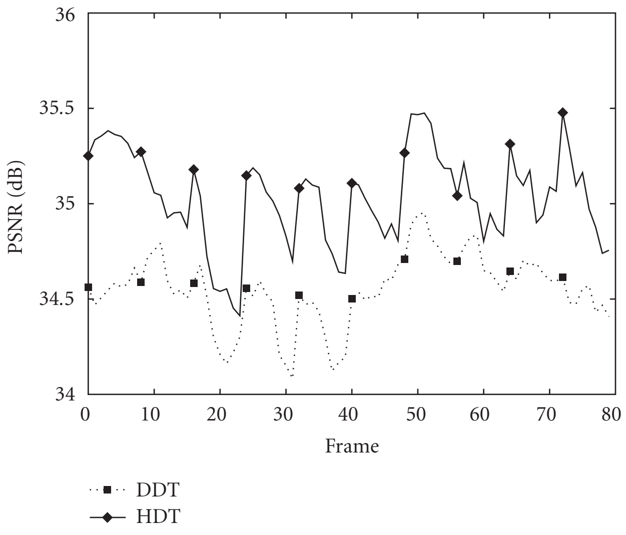
Finally, a careful observation should be devoted to the comparison of the performances obtained with the proposed algorithm and with the DCT decimation approach (DDT). As it was previously predicted, although both algorithms provide quite similar quality performances for scaling factors given by integer powers of 2, the same does not happen when other scaling factors are considered. In such cases, the proposed HDT algorithm proves to provide significantly better results than the DDT algorithm. Moreover, by analyzing the results presented in Tables 3(b) and 4, it should be noted that such better performances are obtained with fewer operations. As a consequence, for downscaling operations implemented in restricted computational environments, where the available amount of arithmetic operations that may be carried out to process each pixel in real-time is limited, the proposed hybrid algorithm offers the possibility to process more decoded DCT coefficients than the DCT decimation algorithm, thus potentially providing much better quality results. Table 5 illustrates such situation. For each scaling factor $\mathcal{S}$, it was presented the number of DCT coefficients that are considered by the DCT decimation algorithm (DDT) as well as the number of coefficients that may be processed by the proposed hybrid algorithm (HDT), when both approaches roughly make use of the same number of operations. For each of these experimental setups, it was also presented the corresponding PSNR gain, provided by the proposed HDT approach. As it can be observed, while for scaling factors given by integer powers of 2, the performances of these algorithms are quite similar (with a slight advantage for the DDT algorithm), for scaling factors other than integer powers of 2 and under similar computational constraints, the proposed algorithm is capable of providing much better quality results than the DCT decimation approach.

4.3. Drift

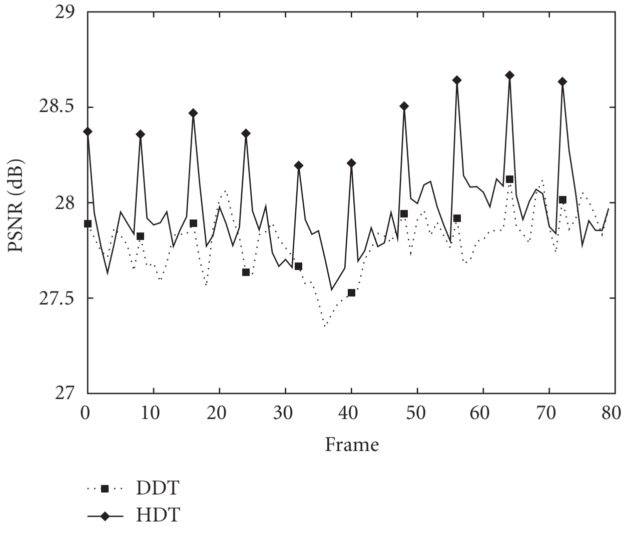
After a first evaluation of the static video quality provided by the considered algorithms, a thorough assessment of their performances when processing video sequences that apply the traditional temporal prediction mechanisms was carried out. Such evaluation was conducted by downscaling encoded video sequences with CIF resolution (352×288) and group of pictures (GOPs) composed by 8 frames, considering both the



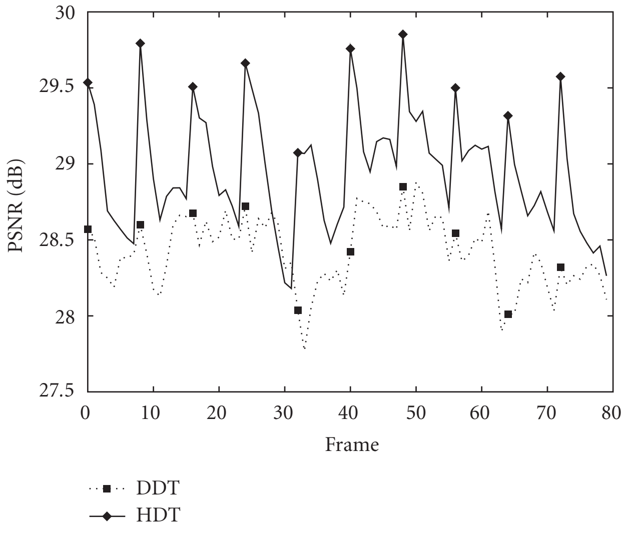
(a) Akiyo, $\mathcal{S} = 3$



(b) Akiyo, $\mathcal{S} = 5$



(c) Mobile, $\mathcal{S} = 3$



(d) Mobile, $\mathcal{S} = 5$

Figure 8: PSNR obtained by downscaling the *Akiyo* and *Mobile* video sequences, considering $Q = 4$ and GOP= 8 frames.

Table 6: Video quality (PSNR) gains provided by the proposed HDT algorithm over the DDT approach, for different scaling factors ($\mathcal{S}$) and considering $K_{\text{HDT}} = K_{\text{DDT}} = \lceil N/\mathcal{S} \rceil$.

$\mathcal{S}$	2	3	4	5	6	7	8
Akiyo	−0.28 dB	+0.19 dB	−0.34 dB	+0.51 dB	+0.19 dB	+3.68 dB	−0.03 dB
Silent	−0.09 dB	+4.29 dB	−0.54 dB	+8.35 dB	+4.58 dB	+4.17 dB	−0.22 dB
Carphone	−0.23 dB	−0.11 dB	−0.28 dB	+0.25 dB	+1.19 dB	+3.81 dB	−0.10 dB
Table-tennis	−0.15 dB	+0.34 dB	−0.32 dB	+1.03 dB	+1.24 dB	+2.32 dB	−0.01 dB
Mobile	−0.61 dB	−0.06 dB	−0.36 dB	+0.33 dB	+1.35 dB	+2.24 dB	−0.10 dB

proposed hybrid approach (HDT) and the DCT decimation transcoding algorithm (DDT). To obtain comparison results as fair as possible, both approaches used the same amount of decoded DCT coefficients: $K_{\text{HDT}} = K_{\text{DDT}} = \lceil N/\mathcal{S} \rceil$.

In Figure 8, it is presented the variation of the PSNR measure obtained for the *Akiyo* and *Mobile* video sequences along the first 80 frames, when downscaled by scaling factors $\mathcal{S} = 3$ and $\mathcal{S} = 5$ and considering a quantization parameter of $Q = 4$. These two video sequences feature distinct content

Table 7: Bit-rate gains provided by the proposed HDT algorithm over the DDT approach, for different scaling factors ($\mathcal{S}$) and considering $K_{\text{HDT}} = K_{\text{DDT}} = \lceil N/\mathcal{S} \rceil$.

$\mathcal{S}$	2	3	4	5	6	7	8
Akiyo	−5.85%	−7.05%	−2.44%	+1.70%	−0.63%	+5.40%	+0.84%
Silent	−7.70%	−10.67%	−3.84%	−4.05%	−5.05%	+0.17%	−0.80%
Carphone	−8.67%	−14.13%	−4.55%	−8.10%	−3.70%	−1.17%	+2.85%
Table-tennis	−9.50%	−13.73%	−4.03%	−5.14%	−4.20%	+0.62%	−2.73%
Mobile	−12.30%	−21.46%	−7.68%	−14.79%	−7.68%	−2.77%	+1.18%

characteristics: while the *Akiyo* sequence is characterized by a reduced amount of spatial and motion activity, the *Mobile* video sequence features a significant amount of spatial detail and movement. From the obtained results, it can be observed that the proposed hybrid algorithm (HDT) consistently provides better quality levels than the DCT decimation approach (DDT), thus confirming the conclusions that were previously driven from their static behavior.

In Table 6, it is represented the average PSNR gain provided by the proposed HDT approach over the DCT decimation scheme for several other different video sequences and scaling factors ($\mathcal{S}$). Such gain was evaluated by computing the average of the corresponding PSNR difference for a time period corresponding to 300 frames. Once again, the obtained values demonstrate that while for scaling factors given by integer powers of 2, the two considered approaches provide similar quality levels (with a slight advantage for the DDT scheme), for scaling factors other than integer powers of 2 the proposed HDT algorithm provides significantly better quality performances. In particular, the results that were obtained with the *Silent* video sequence revealed a notable advantage of the proposed scheme when processing this video sequence. Such advantage comes as a result of the significant amount of spatial detail that exists in the background of this sequence, which is particularly affected by the degradation effect introduced by the postprocessing discarding of the DCT coefficients, inherent to the DCT-decimation approach. Hence, these results fully comply with the previously presented static video quality behavior.

Moreover, the charts presented in Figure 8 also evidence that the effect of the inherent drift on the proposed scheme is not significantly different from the DCT-decimation approach. In fact, by adopting this reference closed-loop architecture (see Figure 6) to evaluate the proposed hybrid scaling algorithm, a new reduced resolution residual is computed in the encoder loop, thus providing a realignment of the predictive and residual components and minimizing the involved drift [17]. Such drift mainly arises from requantization, elimination of some nonzero DCT coefficients and arithmetic errors caused by integer truncation, which will degrade the reference picture used in the temporal prediction mechanism.

To compensate for this gradual degradation along the scaling process, Yin et al. [17] proposed four drift compensating architectures that attempt to reduce the influence of such degradation based on a drift error analysis. Although some of such proposals are mainly targeted to be applied in open-loop downscaling architectures (which are naturally more prone to the influence of this degradation), some of the presented approaches could equally be applied to the closed-loop transcoding architecture considered in this paper (e.g., *Intra_Refresh*). However, since the main scope of this paper is not the actual video transcoding architecture that is adopted but it is the proposal of a computational efficient and more accurate arbitrary resizing algorithm, such compensation architectures were not considered. In fact, the proposed downscaling algorithm could equally be implemented in the down-sample conversion modules of all architectures proposed in [17].

4.4. *Bit rate*

In Table 7, it is represented the average bit-rate gain provided by the proposed HDT approach over the DCT decimation scheme for all the considered video sequences and scaling factors ($\mathcal{S}$), where

$$\Delta \text{ bit-rate } [\%] = 100 \times \frac{\text{bits (HDT)} - \text{bits (DDT)}}{\text{bits (DDT)}}. \tag{27}$$

As before, such gain was evaluated by averaging the differences between the amount of bits required to encode each frame by the two considered algorithms over a time period corresponding to 300 frames, considering $Q = 4$ and $K_{\text{HDT}} = K_{\text{DDT}} = \lceil N/\mathcal{S} \rceil$.

The obtained results evidence a clear advantage of the proposed algorithm over the DDT approach, thus requiring fewer bits (up to 15% less) to encode each frame of the video sequences. Such advantage comes as a result of using a more accurate reduced-resolution reference frame, which will provide a much better temporal prediction mechanism, thus resulting in smaller residuals. In fact, the observed advantage is more significative in video sequences that present greater amounts of movement, such as the *Carphone*, the *Table-tennis,* and the *Mobile*, where such prediction mechanism influences most the efficiency of the video encoder.

5. CONCLUSION

An innovative and efficient transcoding algorithm for video downscaling in the transform domain by any arbitrary integer scaling factor was proposed in this paper. Such algorithm offers a considerable efficiency in what concerns the computational cost, by taking advantage of the scaling mechanism and by only performing the operations that are really needed to compute the desired output values. All the involved steps are properly tailored so that all operations are performed using the coding standard block structure, independently of the adopted scaling factor. In order to meet a variety of system needs, an optional and adaptable tradeoff between the involved computational cost and the resulting video quality is also proposed, by combining the presented algorithm with high-order AC frequency DCT coefficients discarding techniques. Experimental results have shown that the proposed algorithm provides significant advantages over the usual DCT decimation approaches, both in terms of the involved computational cost, the output video quality and the resulting bit rate. Such advantages are even more significant for scaling factors other than integer powers of 2, leading to a reduction of the computational cost as high as 5 and to quite significant PSNR gains, when compared with the usual DCT decimation techniques.

APPENDIX

COMPUTATIONAL COMPLEXITY ANALYSIS

As it was mentioned along the text, to evaluate the computational complexity of the considered algorithms, the number of multiplications ($\mathcal{M}$) required to process each of the ($W_c \times W_l$) pixels of the original frame was considered as the main figure of merit. In the following, the computational complexity of each algorithm will be derived.

A.1. *Cascaded pixel averaging transcoder (CPAT)*

In this approach (see Figure 4(b)), the filtering and subsampling processing steps are entirely carried-out in the pixel domain. For each scaled block ($\widehat{\mathbf{B}}$), most operations are performed in the computation of the $\mathcal{S}^2$ IDCTs, each one requiring $2N^3$ multiplications, since one single multiplication is required to compute the average of each set of ($\mathcal{S} \times \mathcal{S}$) pixels:

$$\mathcal{M}(\texttt{CPAT}) = \frac{1}{W_l W_c}\left[\underbrace{\frac{W_l W_c}{N^2}\cdot 2N^3}_{\text{IDCTs}} + \underbrace{\frac{W_l W_c}{S^2}\cdot 1}_{\text{averaging}} + \underbrace{\frac{W_l W_c}{S^2 N^2}\cdot 2N^3}_{\text{DCTs}}\right]. \tag{A.1}$$

By considering that $1/\mathcal{S}^2 \ll 1$, it can be approximately formulated as

$$\mathcal{M}(\texttt{CPAT}) \approx 2N. \tag{A.2}$$

A.2. *DCT decimation transcoder (DDT)*

As it was described in Section 2.3, most operations that are required to process each scaled block ($\widehat{\mathbf{B}}$) are performed in the computation of the $\mathcal{S}^2$ IDCTs, each one requiring $2K^3$ multiplications, and of the final $(K\mathcal{S})$-point DCT:

$$\begin{aligned}\mathcal{M}(\texttt{DDT}) &= \frac{1}{W_l W_c}\left(\underbrace{\frac{W_l W_c}{N^2}2K^3}_{\text{IDCTs}} + \underbrace{\frac{W_l W_c}{\mathcal{S}^2 N^2}2(K\mathcal{S})^3}_{\text{DCT}}\right)\\ &= \frac{2K^3}{N^2}(1+\mathcal{S}),\end{aligned} \tag{A.3}$$

which can be approximately formulated as

$$\mathcal{M}(\texttt{DDT}) \approx \frac{2K^3\mathcal{S}}{N^2}. \tag{A.4}$$

A.3. *Hybrid downscaling transcoder (HDT)*

To estimate the overall computational complexity of the proposed algorithm, one shall start by evaluating the cost of computing each $\overline{\mathbf{p_{i,j}}}$ matrix:

$$\mathcal{M}(\overline{\mathbf{p_{i,j}}}) = n_l(i)KK + n_l(i)Kn_c(j). \tag{A.5}$$

Hence, it follows that

$$\begin{aligned}&\mathcal{M}(\texttt{HDT})\\ &\quad= \frac{1}{W_l W_c}\left[\frac{W_l W_c}{\mathcal{S}^2 N^2}\left(\underbrace{\sum_{i=0}^{\mathcal{S}-1}\sum_{j=0}^{\mathcal{S}-1}\mathcal{M}(\overline{\mathbf{p_{i,j}}})}_{\text{IDCTs + averaging + scaling}} + \underbrace{2N^3}_{\text{DCTs}}\right)\right],\end{aligned} \tag{A.6}$$

where

$$\sum_{i=0}^{\mathcal{S}-1}\sum_{j=0}^{\mathcal{S}-1}\mathcal{M}(\overline{\mathbf{p_{i,j}}}) = K^2\mathcal{S}\sum_{i=0}^{\mathcal{S}-1}n_l(i) + K\left(\sum_{i=0}^{\mathcal{S}-1}n_l(i)\right)\left(\sum_{j=0}^{\mathcal{S}-1}n_c(j)\right). \tag{A.7}$$

By generically defining

$$n_q = \left\lfloor\frac{qN+(N-1)}{\mathcal{S}}\right\rfloor - \left\lfloor\frac{qN}{\mathcal{S}}\right\rfloor + 1 \tag{A.8}$$

as the number of lines of each $\overline{f_{\mathcal{S}}^{q}}$ matrix, it can be shown that

$$N \le \sum_{q=0}^{\mathcal{S}-1} n_q < \mathcal{S}\left(\left\lfloor\frac{N}{\mathcal{S}}\right\rfloor + 2\right), \tag{A.9}$$

where the lower limit of the previous expression corresponds to the case when N is an integer multiple of $\mathcal{S}$, whereas the upper limit corresponds to a hypothetical worst case situation when the set of $\mathcal{S}$ nonnull elements of both the upper and the lower lines of each $\overline{f_{\mathcal{S}}^{q}}$ matrix are split across different $\overline{f_{\mathcal{S}}^{q}}$ matrices (see (13)). Thus

$$\sum_{i=0}^{\mathcal{S}-1}\sum_{j=0}^{\mathcal{S}-1}\mathcal{M}(\overline{\mathbf{p_{i,j}}}) < K^2\mathcal{S}^2\left(\left\lfloor\frac{N}{\mathcal{S}}\right\rfloor+2\right)+K\mathcal{S}^2\left(\left\lfloor\frac{N}{\mathcal{S}}\right\rfloor+2\right)^2. \tag{A.10}$$

By using the above relation, as well as (A.6), one can obtain

$$\begin{aligned}\mathcal{M}(\mathtt{HDT}) < \frac{1}{\mathcal{S}^2N^2}\Bigg[&K^2\mathcal{S}^2\left(\left\lfloor\frac{N}{\mathcal{S}}\right\rfloor+2\right)\\ &+K\mathcal{S}^2\left(\left\lfloor\frac{N}{\mathcal{S}}\right\rfloor+2\right)^2+2N^3\Bigg],\end{aligned} \tag{A.11}$$

which can be approximately formulated as

$$\mathcal{M}(\mathtt{HDT}) \approx \frac{KN\mathcal{S}(K+4)+2(N^3+K^2\mathcal{S}^2)}{N^2\mathcal{S}^2}. \tag{A.12}$$

A.4. Comparison ratios

From the above estimates, it can be extracted a comparison relation of the overall complexity of the several methods in terms of the scaling factor $\mathcal{S}$, not considering the impact of the inherent discarded constant factors:

$$\begin{aligned}\frac{\mathcal{M}(\mathtt{HDT})}{\mathcal{M}(\mathtt{CPAT})} &\approx \frac{K^2\mathcal{S}+2N^2}{2N^2\mathcal{S}^2} \propto \mathcal{O}\left(\frac{1}{\mathcal{S}}\right),\\ \frac{\mathcal{M}(\mathtt{HDT})}{\mathcal{M}(\mathtt{DDT})} &\approx \frac{K}{N}\frac{1}{\mathcal{S}^2} \propto \begin{cases}\mathcal{O}\left(\frac{1}{\mathcal{S}^2}\right) & \text{for } K=N,\\ \mathcal{O}\left(\frac{1}{\mathcal{S}}\right) & \text{for } K=\frac{N}{\mathcal{S}}.\end{cases}\end{aligned} \tag{A.13}$$

The obtained ratios clearly evidence the complexity advantages provided by the proposed algorithm.

REFERENCES

[1] P. A. A. Assunção and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953–967, 1998.

[2] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: an overview of various techniques and research issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 793–804, 2005.

[3] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, 2005.

[4] N. Roma and L. Sousa, "Least squares motion estimation algorithm in the compressed DCT domain for H.26x/MPEG-x video sequences," in *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS '05)*, pp. 576–581, Como, Italy, September 2005.

[5] W. Zhu, K. H. Yang, and M. J. Beacken, "CIF-to-QCIF video bitstream down-conversion in the DCT domain," *Bell Labs Technical Journal*, vol. 3, no. 3, pp. 21–29, 1998.

[6] T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats," *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 101–110, 2000.

[7] H. Shu and L.-P. Chau, "An efficient arbitrary downsizing algorithm for video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 887–891, 2004.

[8] Y.-R. Lee, C.-W. Lin, S.-H. Yeh, and Y.-C. Chen, "Low-complexity DCT-domain video transcoders for arbitrary-size downscaling," in *IEEE 6th Workshop on Multimedia Signal Processing (MMSP '04)*, pp. 31–34, Siena, Italy, September - October 2004.

[9] C. L. Salazar and T. D. Tran, "On resizing images in the DCT domain," in *Proceedings of IEEE International Conference on Image Processing (ICIP '04)*, vol. 4, pp. 2797–2800, Singapore, October 2004.

[10] Y. S. Park and H. W. Park, "Arbitrary-ratio image resizing using fast DCT of composite length for DCT-based transcoder," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 494–500, 2006.

[11] H. Shu and L.-P. Chau, "A resizing algorithm with two-stage realization for DCT-based transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 2, pp. 248–253, 2007.

[12] T. Shanableh and M. Ghanbari, "Hybrid DCT/pixel domain architecture for heterogeneous video transcoding," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 601–620, 2003, special issue on multimedia adaptation.

[13] H. Li and H. Shi, "A fast algorithm for reconstructing motion-compensated blocks in compressed domain," *Journal of Visual Languages & Computing*, vol. 10, no. 6, pp. 607–623, 1999.

[14] C.-W. Lin and Y.-R. Lee, "Fast algorithms for DCT-domain video transcoding," in *Proceedings of IEEE International Conference on Image Processing (ICIP '01)*, vol. 1, pp. 421–424, Thessaloniki, Greece, October 2001.

[15] S. Liu and A. C. Bovik, "Local bandwidth constrained fast inverse motion compensation for DCT-domain video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 309–319, 2002.

[16] B. K. Natarajan and B. Vasudev, "A fast approximate algorithm for scaling down digital images in the DCT domain," in *Proceedings of IEEE International Conference on Image Processing (ICIP '95)*, vol. 2, pp. 241–243, Washington, DC, USA, October 1995.

[17] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 1009–1020, 2002.

[18] S. A. Martucci, "Image resizing in the discrete cosine transform domain," in *Proceedings of IEEE International Conference on Image Processing (ICIP '95)*, vol. 2, pp. 244–247, Washington, DC, USA, October 1995.

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

[20] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image down-sampling and for inverse motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 468–476, 1997.

[21] B. Shen and I. K. Sethi, "Block-based manipulations on transform-compressed images and videos," *Multimedia Systems*, vol. 6, no. 2, pp. 113–124, 1998.

[22] Q. Hu and S. Panchanathan, "Image/video spatial scalability

in compressed domain," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 23–31, 1998.

[23] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, 2001.

[24] Y.-R. Lee, C.-W. Lin, and C.-C. Kao, "A DCT-domain video transcoder for spatial resolution downconversion," in *Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems (VISUAL '02)*, pp. 207–218, Hsin Chu, Taiwan, March 2002.

[25] J. Ridge, "Efficient transform-domain size and resolution reduction of images," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 621–639, 2003.

[26] Y.- Lee and C.- Lin, "DCT-domain spatial transcoding using generalized DCT decimation," in *Proceedings of IEEE International Conference on Image Processing (ICIP '05)*, vol. 1, pp. 821–824, Genoa, Italy, September 2005.

[27] V. Patil, R. Kumar, and J. Mukherjee, "A fast arbitrary factor video resizing algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 9, pp. 1164–1171, 2006.

[28] ITU-T. ITU-T Recommendation H.263, "Video coding for low bitrate communication," February 1998.

Nuno Roma received the M.S. degree in electrical and computers engineering from Instituto Superior Técnico (IST), Technical University of Lisbon, Portugal, in 2001. He is currently a Lecturer at the Department of Information Systems and Computer Engineering at IST and a Researcher of the Signal Processing Systems (SiPS) Group of Instituto de Engenharia de Sistemas e Computadores R&D (INESC-ID), where he has been pursuing his Ph.D. studies in the area of video coding and transcoding algorithms in the compressed DCT domain. He has also maintained a long-term research interest in the area of dedicated and specialized circuits for digital signal processing, with a special emphasis on image processing and video coding.

Leonel Sousa received the Ph.D. degree in electrical and computer engineering from Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Lisbon, Portugal, in 1996. He is currently an Associate Professor of the Electrical and Computer Engineering Department at IST and a Senior Researcher at Instituto de Engenharia de Sistemas e Computadores-R&D (INESC-ID). His research interests include VLSI architectures, computer architectures, parallel and distributed computing, and multimedia systems. He has contributed to more than 100 papers in journals and international conferences. He is currently an Associate Editor of the Eurasip Journal on Embedded Systems and member of the technical program of several conferences. He is a Senior Member of IEEE and a Member of ACM.