UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

ADAPTIVE ANYSOTROPIC FILTERING (AAF) FOR REAL-TIME VISUAL ENHANCEMENT OF MPEG-CODED VIDEO SEQUENCES

L. Atzori, F. G. B. De Natale, and F. Granelli

January 2002

Technical Report # DIT-02-0001

ADAPTIVE ANISOTROPIC FILTERING (AAF) FOR REAL-TIME VISUAL ENHANCEMENT OF MPEG-CODED VIDEO SEQUENCES

Luigi Atzori[1], *Member, IEEE*, Francesco G.B. De Natale[2], *Member, IEEE*,
Fabrizio Granelli[2], *Student Member, IEEE*

Abstract

Current standards for video compression achieve good performances in terms of data compaction and signal to noise ratio of the decoded signal. Nevertheless, there are some known problems concerning the visual quality of reconstructed images, which can be partially solved using appropriate post-processing algorithms. The paper proposes a new adaptive anisotropic filter (AAF) that aims at unifying the treatment of different sources of perceptive distortion in MPEG sequences. The process is driven by a local classification of blocks and single pixels of decoded frames, taking into account several parameters (distribution of DCT coefficient energy, presence of sharp variations, spatial position of DCT block boundaries). Experimental results show that the proposed algorithm outperforms existing enhancement approaches, in particular when constraints on complexity and real-time processing are compelling.

1. Introduction

The most widely used techniques for source coding in visual communication applications are DCT-based video coding techniques. They afford good results in terms of compaction ratio, which is essential for the transmission of real-time multimedia application data at contained costs in most transport networks, and make it feasible to use digital video in several application frameworks. Several standards based on this coding strategy (such as MPEG-1,2 [1] and H.26x [2]) have been defined. These are used in many applications, ranging from home entertainment to industrial systems, and soon they will also be deployed in mobile entertainment. The price to pay is a perceivable distortion in the decoded video, mainly consisting of annoying visual artifacts that are especially noticeable at medium and low bitrates. The best known is the

blocking or *tiling* effect, consisting of the introduction of artificial edges at the 8x8 DCT block boundaries, due to independent quantization of the transform coefficients in each block. These edges are perceived by the human eye as unnatural geometrical contours, and lead to serious degradation of the overall subjective quality of the encoded video sequence. Quantization of DCT-coefficients also causes *blurring* of real contours present in the image, due to the cut in high frequency components in the transformed blocks. This cut has an additional side effect, i.e. some retained frequencies remain 'unbalanced' causing ripples near edge locations. Such a distortion is particularly annoying during rendering of video sequences, where small regions at high frequency appear, move, and disappear at random points of the scene, capturing the observer's attention. This behavior is where the name, *mosquito* noise, comes from.

A practical solution to achieve visual enhancement of compressed images is post-processing. It has the advantage of not implying a bitrate increase in the compressed stream, and does not require any changes to existing coding standards. Nevertheless, one only solution for all this is not easy to find, as the problems are strictly intercorrelated. As an example, a classical algorithm for contrast enhancement or edge sharpening will of course reduce edge blurring, but at the same time it will worsen blocking and mosquito effects.

In next section, a brief review of state-of-the-art algorithms for post-processing of decoded video is provided. In section 3, we propose a new method that aims at reducing the visual impact of tiling, edge blurring, and mosquito distortions in one operation. The method is based on an adaptive anisotropic filter (AAF) driven by a pixel classification. Experimental testing demonstrates that AAF outperforms existing enhancement approaches, especially when constraints on complexity and real-time processing are compelling. These properties are displayed in sections 4 and 5, which respectively propose evaluations of the computational complexity of the algorithm and the results achieved with the simulations. Finally, in the appendix some specific distortion measures that have been used to better evaluate the performance of the proposed technique are described and compared with competing state-of-the-art methodologies.

2. Analysis of State-of-the-Art Post-Processing Methodologies for Video Enhancement

In this section, some proposed solutions to the problem of visual quality enhancement in decoded video are reviewed. It should be pointed out that in general these methods are targeted to the removal of a specific type of distortion, and in particular blocking effects. A few solutions are also proposed for mosquito noise. Furthermore, the methodologies presented in this section are characterized by low or medium computational complexity, in order to be comparable with the proposed approach.

As already mentioned, the main source of distortion derives from the independent quantization of the DCT coefficients of each block. This problem can be faced working in the spatial domain after the reconstruction of the video signal, or in the transform domain directly operating on the encoded data stream.

Among the methods working in the spatial domain, a classical approach is to filter the video frames with properly designed convolution kernels. The use of a 3x3 Gaussian smoothing filter for post-processing of the pixels directly adjacent to block boundaries [3] is a simple approach to reduce blocking effects, but it has the drawback of introducing edge and texture blurring close to block borders. Better results are achieved using directional smoothing filters with a stronger effect on the direction orthogonal to the block boundaries [4]. Jarske et al. observed that the performance of such methods is increased by applying high-emphasis filtering after Gaussian smoothing to improve the sharpness of the pixels near block borders [5]. This solution however re-introduces noise problems.

The use of one-dimensional filters is proposed in [6], where the kernel is applied in the direction orthogonal to the block edge to be removed. The filter is spatially adapted by evaluating local distortion and its visibility based on the characteristics of the human visual system: to this extent, gamma function and Weber's law are used to estimate distortion visibility.

In [7], the activity of each block is estimated by computing the maximum difference among all 3x3 sub-blocks inside the block, not counting the internal boundary. The activity parameter is then used to tune a

smoothing filter kernel chosen among three types: inverse gradient, thresholded average and Gaussian. Finally, a contour enhancement is operated by high-pass filtering.

The solution proposed in [8] transposes the problem of estimating and reducing the blocking effect into a *Constrained Least Squares* (CLS) problem, which must be solved for each image block. This approach has many desirable properties, such as non-linear processing to protect edges, efficient characterization of block distortion, and fast convergence. After estimating the amount of blocking in the image, a non-linear filtering that reduces the tiling effect is applied while preserving real contours. Further work on the subject has determined that it is possible to simplify the estimation by considering only a few key transform coefficients.

A different class of methods is based upon projections onto convex sets (*POCS*), usually implemented through iterative algorithms. Discontinuities among blocks can in fact be identified as high horizontal or vertical frequencies, whereby projections of the degraded image onto a set of band-limited signals can be used to decrease blocking, thus introducing a regularization constraint in the quantization levels of the transformed coefficients [9]. In [10], the convex set for blocking reduction is made up of all the images for which the sum of the squared pixel differences across block boundaries is smaller than a given threshold.

In [11], the innovative concept of boundary-orthogonal bi-dimensional functions is introduced. The aim is to calculate the coefficients of a set of boundary orthogonal 2-D functions that, summed pixel-by-pixel to the corrupted image, minimize blocking. Furthermore, it is shown that good artifact reduction is possible if three of the usual DCT baseline functions are used.

The method presented in [12] is based on three parameters: the global slope, which represents the difference between the central pixels of two adjacent blocks; the block slope, which is the difference between internal and boundary block pixels; and the local slope, which is computed as the difference among pixels inside each adjacent block. Based on these values, the method defines an adaptive operator that makes the block slope of two adjacent blocks equal to their common global slope while maintaining

the local slopes. The result is that blocking effect is reduced, while high frequency components close to the block boundary remain unaffected by the process.

Among the algorithms that work in the frequency domain, the one proposed in [13] is based on the classification of the blocks and the estimation of the blocking distortion strength. Here, the basic assumption is that tiling is more likely to be perceived by the human visual system in low activity regions (smooth areas). On this basis each block is classified as *smooth* or *non-smooth*, and only the smooth blocks are filtered to compensate false contours by low-pass FIR filters, which are locally adapted to the magnitude of the undesired contour. Artifact visibility is estimated through the square difference between internal and external block boundaries, and is approximated as a function of the DC coefficient and of the AC coefficients on the first row and column. Thanks to the direct application of the estimates in the DCT domain, it is possible to achieve low complexity implementation.

A similar approach is employed in [14], where an enhancement algorithm is developed starting from two simple assumptions: the first, that smoothing of artificial discontinuities between blocks improves visual image quality; the second, that smoothing of actual image edges degrades image quality. To discriminate between these two cases, the quantization error of each DCT coefficient is estimated for each image block using statistical models. By averaging the estimates over the entire image, a reliable measure of the quantization error associated to each block can be achieved. Artificial discontinuities can then be efficiently removed with a non-linear attenuation technique that aims at reducing the strength of the discontinuity under the visibility threshold set by Weber's law.

Concerning mosquito noise, classical post-processing techniques make use of non-linear filters. As an example, rank-order filters (in particular, the median filter) can remove isolated noise peaks without corrupting the nearby regions [15]. Other approaches have been proposed that need some processing at the encoder. The most interesting are the standard-compliant pre-processing techniques: here, knowledge of the mosquito noise characteristics is exploited to make the image more robust to the distortion [16].

3. Video Enhancement by Adaptive Anisotropic Filtering

As mentioned in the introductory section, the objective of the proposed method is to jointly reduce the appearance of most annoying artifacts introduced by DCT-based video coding standards, namely, blocking, edge blurring, and mosquito effects. The main advantage of a comprehensive method lies in the fact that it allows to reduce one defect without increasing the visibility of another. For instance when an edge enhancement filter is applied to reduce contour blurring, it also causes an increase in tiling and mosquito – a problem that could easily be overcome by a comprehensive method.

The three types of distortion so far introduced can be regarded as different noise sources, characterized by a non-uniform distribution over the image. To achieve effectiveness in the application of the algorithms for distortion removal, therefore, we must first identify which areas are involved by which problems. Blocking distortion is very easy to locate, for it necessarily occurs at the boundary between adjacent blocks. Differently, edge blurring and mosquito noise are content-dependent, as they are a side effect of the compression procedure when applied to image areas containing high spatial frequencies. To obtain better knowledge of these phenomena, the following considerations should be taken into account:

- mosquito noise typically affects image blocks crossed by significant luminance transitions, and is located near edges. It is due to the cut-off of high frequency DCT coefficients;
- blurring affects image blocks containing textures and edges, and causes loss of accuracy in small details and overall sharpness reduction. It is due to severe quantization of DCT coefficients;
- blocking and mosquito artifacts are more critical in smooth areas than on textures, due to spatial masking effects.

Although the above observations provide some conceptual insight into the problem, a more accurate assessment can only be achieved by defining a quantitative measure of the distortion. A very simple but effective approach consists of analyzing the variation of local contrast caused by the co-decoding process.

As a matter of fact, edge and texture blurring produce a reduction in local image definition, while mosquito noise artificially increases contrast. The following formula defines the normalized variation of local contrast [20]:

$$NVLC(i) = \frac{\left[\sigma_{or}^2(i) - \sigma_{dist}^2(i)\right]}{\max\left\{1, \sigma_{dist}^2(i)\right\}}, \qquad (1)$$

where $\sigma_{or}(i)$ and $\sigma_{dist}(i)$ represent the variance computed on a 3x3 window centered on pixel i for the original and compressed images, respectively. The $NVLC$ index is expected to be positive and increasing with compression in edge and texture pixels, due to loss of sharpness. An opposite behavior is expected in the pixels surrounding edges, where mosquito can be present.

An experimental evidence of this fact comes from the observation of the charts in Fig. 1. They have been obtained by analyzing the behavior of $NVLC$ on a video frame compressed by a JPEG-like technique at various compression factors, CF. The chart in Fig. 1.a plots the mean values of $NVLC$ in the image areas surrounding the edges: the curves $\overline{NVLC_e}^{CF}(j)$ are obtained by averaging the $NVLC$ parameter over a set of pixels that lie at an equal distance j from the relevant edge. The chart in Fig. 1.b plots the mean $NVLC$ values in the textured image areas: in this case the curves $\overline{NVLC_t}^{CF}(j)$ are obtained by averaging the $NVLC$ index over a set of pixels lying at an equal distance j from the border of the relevant DCT block. In both cases, the city block distance was used as a topological measure. The first graph shows loss of contrast on edge pixels (j = 0), where the curves have positive values that increase with compression, and enlarged contrast at intermediate distances from the edge (j = 2÷6), where the curves become negative. The second graph shows a completely different behavior, the contrast being increased near the block borders (j = 0) due to the tiling effect, and slightly reduced at higher distances (j ≥ 1) due to texture smoothing.

The above concepts are very important for both the design and testing of the proposed post-processing system. In particular, by imposing the equalization of the $NVLC_{e,t}^{CF}(j)$ curves for a given compression

factor, it is possible to tune the system parameters correctly in order to achieve contrast values similar to those of the original image, thus augmenting edge sharpness and reducing the artifacts on smooth areas. To achieve this goal, it is essential to introduce reliable techniques to extract the spatial features (edges/textures) of the frame to be processed, and to classify the blocks and single pixels for successive filtering.

According to the above considerations, a post-processing architecture has been defined as shown in Fig. 2. It has been called Adaptive Anisotropic Filtering (AAF) as it is based on a tunable filter that adaptively selects local convolution kernels based on the spatial context of the pixel. The architecture includes three main modules: block classification, pixel classification, and filter bank. The first module analyzes the energy distribution within the DCT coefficient matrix, and consequently classifies each block in three classes: edged, flat, or textured. The second module produces a classification map by marking each pixel on the basis of four parameters: the block class, the spatial position within the block, the distance from the closest edge, and the relevant direction. The last two parameters are used for edged blocks only, in which case edges are previously extracted by a separate module. Finally, based on the classification map, a filtering procedure is applied to each pixel using a bank of pre-defined kernels through a discrete convolution.

For the implementation of each specific module, a constraint that was carefully taken into account is the integration of the post-processor in a *hw* decoder architecture at low-costs. This involves the use of algorithms that require light computation, low storage requirements, and slight changes in the decoder architecture. Iterative solutions were therefore discarded, as well as solutions that require heavy access to the bitstream and to decoded data. In particular, it is supposed that only one slice is available at a time, possibly including the relevant motion vectors. The last constraint is typically caused by restrictions in the use of data buses to access the buffered stream and decoded reference frames. The following sub-sections describe in detail the definition of the three main components of the system.

3.1 Block Classification Module

The system architecture defined in Fig. 2 emphasizes the strong dependence of local processing on the result of block and pixel classification, and points out the great influence of classifier performance on overall post-processing achievement. As an example, an inaccurate discrimination among textured and edged regions would imply the selection of inappropriate kernels for local filtering, thus causing further worsening of the distortion.

The first level of classification aims at distinguishing among three block categories: 'edged', 'smooth', and 'textured'. The procedure used in this context works in the DCT domain, with the double advantage of low computational complexity and satisfactory performance. It is based on the approach proposed in [17]. The transformed coefficients are divided into four groups (see Fig. 3): the DC coefficient, the low-frequency group, the edge group, and the high-frequency group. Then, the parameters L, E, and H are computed by summing the magnitudes of the coefficients within the last three groups. It was experimentally established that large values of the ratios $\frac{L+E}{H}$ and $\frac{L}{E}$ denote the presence of edges, while the sum $(E+H)$ provides a good approximation of the block texture energy. Each intra-coded block can therefore be directly classified using simple combinations of the relevant transform coefficients, followed by a heuristic thresholding procedure. The adopted procedure is explained in detail in [17], and uses a set of thresholds obtained from experimental evidence. Fig. 4 shows the result of applying this block classification procedure to an I-frame of the *Claire* sequence.

A special procedure is used for P- and B-frames, where motion information can be used to predict the class of each block, in order to speed up classification. To this purpose, each block in the current frame inherits the class of the dominant block in the reference frame (i.e., the block that shows maximum spatial overlapping with the current one). This procedure ensures an exact result only if the block pointed by the

motion vector exactly matches a DCT block in the reference frame, otherwise it can produce errors with possible propagation to subsequent frames. Fig. 4.c shows the result of the 'inter' classification applied to a P-frame of the *Claire* sequence . A comparison with the corresponding 'intra' classification showed that temporal prediction achieves correct results in 95% of the cases. Further analysis demonstrated that almost all misclassified blocks were overlapped in similar proportion to two or more DCT blocks in the reference frame. This observation suggested that the small percentage of blocks that suffer from this problem should be treated in a special way. The adopted procedure is based on a simple edge detection performed in the spatial domain: after the detection, if the block is crossed by significant contours it is classified as 'edged', otherwise as 'textured'.

3.2 Pixel Classification Module

After block classification, the exact type of filtering to be applied to each pixel must be defined: to this purpose, a pixel-level classification is introduced. Five classes of pixels have been defined: *blocking*, *mosquito*, *edge*, *texture*, and *non-processed*. The first two classes represent the pixels potentially affected by the relevant sources of distortion; *edge* and *texture* pixels are usually affected by a more or less visible blurring effect; *non-processed* pixels identify the areas that do not require post-processing. Obviously, the pixel classifier operates differently depending on its block class, in particular (see Fig. 5):

- pixels belonging to 'smooth' blocks are marked as:
  - 'blocking pixels' at the block boundaries
  - 'non-processed pixels' in the internal area
- pixels belonging to 'textured' blocks are marked as:
  - 'non-processed pixels' at the block boundaries
  - 'texture pixels' in the internal area
- pixels belonging to 'edged' blocks are marked as:

- 'edge pixels' at the block boundary with $D^2 > T_H$, or in internal area with $D^2 > T_L$
- 'blocking pixels' at the block boundaries with $D^2 < T_H$
- 'mosquito pixels' otherwise

As can be noticed, while for smooth or textured blocks the position is the only parameter used to classify pixels, in the case of edged blocks an additional parameter D^2 is used to verify some threshold conditions. D^2 derives from the application of a Sobel operator [18], and provides an estimation of the position and strength of a contour pixel. It is defined as follows:

$$D^2 = D_x^2 + D_y^2, \tag{2}$$

where D_x and D_y approximate the discrete derivatives along the horizontal and the vertical directions, and are achieved by convolution with the following kernels m_x, m_y:

$$m_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad ; \quad m_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3}$$

In the thresholding operation, two thresholds achieved by experimental analysis have been used, T_L and T_H. The lower threshold T_L is used in the internal area of the block to distinguish real edges from background noise. Sobel being a quadratic filter intrinsically quite robust to noise, T_L can be fixed a-priori independently of image type and compression. The higher threshold T_H is used at the block borders. It can be obtained by incrementing T_L by a value proportional to the compression factor in order to avoid the detection of false edges due to blocking. Typical T_H values range from $1.5 \cdot T_L$ to $2 \cdot T_L$, depending on the CF.

The Sobel filter provides an additional parameter, which is useful in the filtering module. In fact, the inverse tangent of D_y / D_x gives an estimate of the contour direction. This parameter can be roughly quantized into four orientations (S-N, SW-NE, W-E, and NW-SE) to drive a directional filtering, as explained in the following section.

3.3 Filtering Bank

In the filtering module, the result of pixel classification is used to select the proper kernel for distortion removal. An exception is represented by pixels, marked as 'non-processed', which are left unchanged. No-process pixels are located at the boundaries of textured blocks as well as in the internal areas of smooth blocks. In the first case, blocking is masked by the texture; furthermore, the application of a blocking reduction filter will produce clear texture blurring, while texture enhancement will very likely heighten tiling artifacts. In the latter case, internal pixels are usually well approximated by DCT.

In the other cases, the pixel class drives the selection of an appropriate kernel within a bank of 3x3 or 5x1 FIR masks. Depending on the pixel type, in fact, a different filtering aimed at removing a specific type of noise, which is most visible in the corresponding area, is applied. Consequently, the following filters are defined:

Edge pixels: a directional edge enhancement filter that increases the contrast along the gradient direction and smoothes in the perpendicular direction is applied. To this aim, the quantized gradient directions extracted by the Sobel filter are used to select one among four possible kernels, obtained by rotating the basic mask of Fig. 6.a (used for vertical edges). The C parameter is used to control filter strength: a small value involves strong edge crispening, while higher values yield to softer effects.

Mosquito pixels: a 3x3 directional smoothing filter is applied. Fig. 6.b shows the mask used if the closest edge pixel has horizontal direction. The masks used for the other three directions are obtained through circular shifts of the eight boundary coefficients around the central value. As can be observed, these filters have a stronger impact in the direction orthogonal to the edge, which is more affected by ripple effects. Also in this case, the strength of the smoothing effect is ruled by a parameter S: small values of S increase the flatness of the filtered image area, while eliminating signal spikes that characterize mosquito noise.

Texture Pixels: a single 3x3 mild contrast enhancement filter is applied (see Fig. 6.c). Similarly to edge filtering, it is ruled by a parameter *K*, small values of which lead to stronger crispening. No directionality is required in this case.

Blocking Pixels: a 1-D low-pass filter is applied to mask the unpleasant visual effect of luminance transition across adjacent blocks. The filter is applied along the direction orthogonal to the block boundary: in particular, in vertical block boundaries the horizontal mask is used and vice versa. The filter is applied to the most external block perimeter as well as to the internal one just after it. The filter uses uniform weights for all 5 pixels (moving average), and is recursively applied from the external block perimeter to the internal one, since it achieves more effective smoothing (see Fig. 7). Pixels marked as edges are not considered in this process.

Summarizing, a total of 10 parametric filter kernels are used at this stage:

– 4 masks (3x3) for directional contrast enhancement;

– 4 masks (3x3) for directional smoothing;

– 1 mask (3x3) for texture enhancement;

– 2 masks (5x1) for removing blocking artifacts.

A correct setting of the filter parameters is a crucial task, since selecting wrong filter coefficients would produce negative effects. A set of optimal parameters could be computed by imposing the flatness of the curves $NVLC_{e,t}^{CF}(j)$ after post-processing as a target. This corresponds to increasing or decreasing the local variance of the image in order to match the original values. An experimental set up of kernel parameters was therefore derived from the study of a set of *NVLC* values observed on a representative image set. To achieve this goal, an empirical relationship was first drawn between image distortion and filter parameters. This was obtained by modulating the parameters of the post-processing filters applied to a test set with uniform distortion, and analyzing the results. Distinct experiments were conducted for edge, mosquito, and texture regions. Then, for each distortion range the parameter settings that maximize the

quality of the post-processed data were chosen. Fig. 8 plots the resulting curves, which provide optimal parameter values as a function of the distortion. Of course, this optimization should be regarded in an average sense.

Fig. 9 provides a practical example, where the same curves are used to set the filter parameters in the post-processing of a frame of the *Claire* video sequence: Figs. 9.a-b depict the parameter settings, while Figs. 9.c-d report the distortion curves after processing. It can be observed that the compression effects in terms of local image variance could not be cancelled completely. The reason is twofold: first, the parameters were obtained based on mean *NVLC* values, thus allowing only global parameter tuning; second, it is often impossible to completely restore the original contrast starting from the distorted image, especially after heavy edge or texture smoothing. Nonetheless, the resulting curves show a significant improvement in the filtered image contrast.

Unfortunately, the parameter setting procedure so far described is not feasible in practical applications, since on order to compute the *NVLC* values, the original data, which are usually unavailable at the decoder, must be known. Nevertheless, it was experimentally found that the behavior of the *NVLC* curves is strictly correlated to the compression factor, while it is not significantly affected by the image characteristics. The charts in Fig. 10 present the average distortion as a function of the compression rate and as a function of the pixel distance from the edge or from the block border. For a better representation, only four compression levels have been considered. It can be observed that the distortion curves obtained for the *Claire* test video sequence do not correspond completely with those in Fig. 10, thus suggesting that other image characteristics may affect the distortion. Consequently, the reduced accuracy in distortion estimation causes a sub-optimal parameter choice, which is particularly evident in the mosquito areas, where the distortion strictly depends on the strength of the edge and the smoothness of the surrounding area. For operational convenience, the graphical representation in Fig. 10 can also be converted into a set of look-up tables, one for each filter type. The LUTs provide a direct correspondence between parameter

values and compression factors, also considering the distance from the nearest edge (for edge and mosquito filtering), and from the block border (for texture filtering).

Summarizing, filter parameter selection is performed as follows:

- Estimate the compression factor *CF* for each I-frame. This value is used for all the frames in the relevant GOP;
- Based on the *CF*, select the parameter values on the specific look-up tables corresponding to the pixel class.

4. Analysis of Computational Complexity

The computational complexity of the algorithm can be evaluated by examining the four main processing steps involved in the procedure.

- **Block Classification.** In I-frames, each block is directly classified in the DCT domain. The calculation of the L, E, and H values requires *nz* sums and 7 comparisons per block, where *nz* is the average number of DCT coefficients per block preceding the EOB (frequently less than 16). Differently, for temporally predicted frames the classification is performed using the motion vectors available in the MPEG-compressed stream and propagating the block class from the reference frame. Such a process involves a fixed computation of 2 sums (for motion compensation) and 2 comparisons (for block classification) per block.
- **Contour extraction.** The Sobel convolution masks are applied to each edge block. For each pixel the following operations are required: 2x(7 sums)+(2 products + 1 sum)+1 complex function (artg). Considering that the direction of the edge is quantized to only 4 directions, the computation of the inverse tangent can be avoided by using the D_y / D_x ratio directly, as follows:

  $D_y / D_x > 1.24$ or $D_y / D_x < -1.24 \Rightarrow$ vertical direction;

  $0.41 < D_y / D_x < 1.24 \Rightarrow$ SW-NE direction;

$-0.41 < D_y / D_x < 0.41 \Rightarrow$ horizontal direction;

$-1.24 < D_y / D_x < -0.41$ NW-SE direction.

This leads to a total of 3 products, 15 sums, and 4 comparisons per pixel (edge blocks only).

- **Pixel classification.** Most computation is implied by the need of finding the nearest edge point to classify mosquito pixels. This can be efficiently accomplished by processing each edge chain and labeling the pixels orthogonal to the edge within the same 8x8 block. It involves checking a maximum of 8 map positions per edge pixel.
- **Filtering.** Filtering is done through convolution with a 3x3 mask or a 5x1 mask, requiring an upper limit of 9 products and 8 sums per pixel.

For example, for a video sequence in CIF format at 25 frames per sec. and 10% edged blocks, the total computation required is about:

- 0.2M products per second
- 1M sums per second
- 0.3M checks per second,

which makes the processing feasible also in software decoding. For higher resolutions, hardware solutions may be required to achieve real-time performance. To this purpose, it should be pointed out that most of the above operations are suitable for vector or parallel processing, and can be efficiently implemented on a standard DSP. Consequently, the entire procedure can be easily implemented by a properly programmed controller in the decoder, supported by a dedicated DSP for the filtering procedure.

5. Experimental results

Extensive experimental testing and comparison were conducted on several MPEG-coded sequences with different spatial and temporal characteristics. In this section, the results from three different video sequences are presented: *Claire* (Fig. 4.a), characterized by a smooth background and short movements, *Flower*

Garden (Fig. 11.a), with large moving areas, and *Rai* (Fig. 11.b), with high motion activity and scene changes. Simulations were carried out on 40 consecutive frames encoded with an MPEG-I standard encoder at 0.75 Mbps. Performance was mainly evaluated by visual judgment, for there is no standard measure currently available to evaluate subjective video quality. Nonetheless, the following objective measures are also provided for better comparison:

- Peak Signal-to-Noise Ratio (PSNR): PSNR is a widely used quality measure based on the evaluation of the mean square error of the decoded data as compared to the original, namely:

$$\mathrm{PSNR} = 10 \cdot \log_{10} \frac{255^2}{\mathrm{MSE}} \quad ; \quad \mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} \left(x_i - \hat{x}_i \right)^2 \tag{4}$$

  where x_i and $\hat{x}_i$ are the original and processed pixels of the image, respectively.

- Block Distortion (BD): different algorithms have been proposed to assess this phenomenon, a short description of the measure used in this paper (called *Normalized Block Distortion*, NBD) is provided in the Appendix.
- Edge Blurring and Mosquito Noise: the measure of edge blurring and mosquito noise was obtained by analyzing the behavior of the *NVLC* parameter across edged areas (see Appendix).

It should be pointed out that PSNR in no way takes into account the type of distortion, or the characteristics of the Human Visual System (HVS). Therefore, it is not unusual for better looking or enhanced images to be associated to a lower PSNR.

Figs. 12-14 report the performance achieved by AAF in terms of the above distortion measures for the *Rai* video sequence. The chart in Fig. 12 shows a very low variation in PSNR between MPEG-decoded and post-processed frames. As already mentioned, this result is to be considered reasonable: in fact, the target of post-processing is not to reduce the mean error (which is performed by the compression technique), but to reduce the visual impact of the distortion. It is commonly accepted that this operation could even increase the MSE. On the contrary, the effectiveness of the proposed approach is witnessed by the results in terms of blocking/mosquito reduction and edge enhancement. In Figs. 13 and 14 a comparison is

provided with two of the major competing approaches, proposed by Zakhor [9] and Jeong [11], respectively. In particular, Fig. 13 compares the performance of the three methods in terms of NBD, while Figs. 14.a-b demonstrate the achievements in terms of edge sharpening and mosquito reduction. It can be observed that the Zakhor method achieves slightly better results than AAF in terms of mosquito reduction, but it introduces excessive blurring at the edges and block boundaries (loss of image contrast). This is due to the application of a strong low-pass filtering, irrespective of the local image characteristics, and to the employment of quantization constraints, which do not take into account edge sharpness. Differently, the Jeong technique achieves satisfactory results in terms of blocking reduction, but it increases blurring of edges and does not properly treat mosquito artifacts. It minimally affects the PSNR, but employs a set of base functions that are not always adapted to the local characteristics of the block boundaries, thus giving different results depending on the test sequence used for the experiments. Similar results were achieved with other test sets: in Table 1 the above comparison is extended to the three test sequences where the mean values of the employed distortion measures are presented.

Although these measures give an objective assessment of the method performance, the most dependable judgement comes from direct observation of the processed images. In Fig. 15 four clips of the *Rai* video sequence are presented, to allow a visual evaluation of the results achieved by AAF as compared to the Zakhor and Jeong algorithms. Attention should be paid to the sharp reduction in visual artifacts, in particular as regards tiling and mosquito, combined with a satisfactory rendering of high frequency areas. Fig. 16 presents three magnified details extracted from a frame of the *Claire* sequence. Here, the performance of AAF in terms of the significant reduction of block distortion is particularly visible in the smooth background area, as well as the suppression of mosquito artifacts, associated to a good contour sharpness. Very good visual quality was also achieved for the *Flower Garden* sequence (Fig. 17).

6. Conclusions

A novel post-processing technique that aims at enhancing the video quality after DCT block transform coding has been presented (in particular, MPEG standards). The proposed algorithm, which is called Anisotropic Adaptive Filtering (AAF), analyzes the local block and pixel characteristics and on this basis for each image zone it selects the most appropriate filtering by choosing among a set of pre-defined parametric convolutional kernels. Pixel classification has been based on the observation of image contrast variation along edged, smooth, and textured areas, as well as along block boundaries. The same contrast variation curves also revealed a strict inter-dependence among distortion level, contrast variation magnitude, pixel position relative to edges and block borders, and compression factor. This allowed to determine also the kernel parameters (filter intensity) automatically.

The major strength points of the proposed approach lie in the fact that different types of distortion, such as edge and texture blurring, mosquito artifacts and blocking, are treated simultaneously, without requiring iterative or cascade processing and with a low-computational complexity as compared to competing techniques. Experimental tests have shown that AAF provides excellent results from both subjectively and objectively, outperforming existing methodologies with similar or even higher computational complexity. Additionally, AAF features very low buffering requirements (only a few rows of the compressed image are needed), thus making its integration in a decoder architecture very convenient.

Appendix

Measures for effective visual quality estimation

The measure of blocking used in the experimental tests exploits some known properties of the DCT transform [19]. With reference to Fig. 18, let consider a block overlapping two smooth image blocks with different gray values (case B_V): the relevant DCT matrix will contain a subset of non-null AC coefficients, located in the first column at even row indices. It is possible to verify that this set of coefficients highlights similar block boundary discontinuities also if the two blocks are not completely smooth. Similarly, if the

central edge is vertical (case B_H), the non-null AC coefficients are located in the first row at even column indices.

Based on these considerations, the proposed measure, *BD*, estimates blocking by computing the normalized energy content of such coefficients, namely:

$$BD = \frac{\sum_{\Xi_V}\sum_{i=1}^{4}(2\cdot i)^2\cdot c_{2\cdot i,1}^2 + \sum_{\Xi_H}\sum_{j=1}^{4}(2\cdot j)^2\cdot c_{1,2\cdot j}^2}{\sum_{\Xi_R}\left(\sum_{i=1}^{4}(2\cdot i)^2\cdot c_{2\cdot i,1}^2 + \sum_{j=1}^{4}(2\cdot j)^2\cdot c_{1,2\cdot j}^2\right)} \quad (5)$$

where $\boldsymbol{X}_H$ and $\boldsymbol{X}_V$ represent the sets of all possible overlapped blocks of type B_H and B_V in the distorted image, while $\boldsymbol{X}_R$ includes all the blocks where the DCT transform has been applied during compression. The index is close to one if no block distortion is present, meaning that the critical coefficients at the block boundaries show an average behavior similar to the normal image blocks, and increases with the compression factor, due to increased tiling. The *BD* parameter is very interesting, since the original image is not required. Nevertheless, since the specific image characteristics could bias the index (no unit value for uncompressed data), it is possible to obtain the more robust *normalized block distortion* (*NBD*) by dividing the actual BD by the BD value computed on the original image.

As regards edge blurring and mosquito noise distortion assessment, the *NVLC* index defined in § 3, which provides a contrast variation measure, is used [20]. *NVLC* is expected to be positive in edged and textured areas where the DCT quantization procedure cuts-off high frequency components causing image smoothing, and negative in the regions surrounding edgels where mosquito is usually present. Two distortion measures D_{BL} and D_{MO} can therefore be obtained by averaging the *NVLC* parameter on the areas subject to edge blurring and mosquito noise, respectively:

$$D_{BL} = \frac{1}{N_{BL}}\sum_{i\in A_{BL}}\left|NVLC(i)\right| \;;\; D_{MO} = -\frac{1}{N_{MO}}\sum_{i\in A_{MO}}\left|NVLC(i)\right| \quad (6)$$

where A_{BL} includes all the image edgels, and A_{MO} all the pixels that belong to edged blocks except the edgels themselves. Both D_{BL} and D_{MO} show values close to zero if no distortion is present, and increasing positive values consistent with the relevant amount of distortion.

References

[1] J.L. Mitchell, W.B. Pennebaker, C.E. Fogg, and D.J. LeGall, *MPEG video compression standard*, Chapman & Hall, New York, 1997.

[2] "Draft recommendation H.263: video coding for low bitrate communication," ITU-T (CCITT), Dec. 1995.

[3] H.C. Reeve and J.S. Lim, "Reduction of blocking effects in image coding," *Optical Engineering*, Vol. 23, No. 1, pp. 34-37, Jan./Feb. 1984.

[4] K.H. Tzou, "Post-filtering for transform-coded images," *Proceedings of SPIE, Applications of Digital Image Processing XI,* San Diego, CA, Vol. 974, pp. 121-126, Aug. 1988.

[5] T. Jarske, P. Haavisto, and I. Defèe, "Post-filtering methods for reducing blocking effects from coded images," *IEEE Transactions on Consumer Electronics*, Vol. 40, pp. 521-526, Aug. 1994.

[6] F.-X. Coudoux, M. Gazalet, and P. Corlay, "Reduction of blocking effect in DCT-coded images based on a visual perception criterion," *Signal Processing: Image Comuunication*, Vol. 11, No. 3, Jan. 98.

[7] T. Jarske, P. Haavisto, and I. Defèe, "Post-filtering methods for reducing blocking effects from coded images," *IEEE Transactions on Consumer Electronics*, Vol. 40, No. 3, 1994.

[8] M. Crouse, and K. Ramchandran, "Nonlinear constrained least squares estimation to reduce artifacts in block transform-coded images," *ICIP 95 (IEEE International Conference on Image Processing)*.

[9] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding,"

IEEE Transactions on Circuits and Systems for Video Technology, Vol. 2, No. 1, March 1992.

[10] Y. Yang, N.P. Galatsanos, and A.K. Katsaggelos, "Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 6, December 1993.

[11] J. Jeong, and B. Jeon, "Use of a class of two-dimensional functions for blocking artifacts reduction in image coding," *ICIP 95*.

[12] H. Paek, J.-W. Park, and S.-U. Lee, "Non-iterative post-processing technique for transform coded image sequence," *ICIP 95*.

[13] S.-C. Hsia, J.-F. Yang, and B.-D. Liu, "Efficient postprocessor for blocky effect removal based on transform characteristics," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 6, Dec. 97.

[14] J. Chou, M. Crouse, and K. Ramchandran, "A simple algorithm for removing blocking artifacts in block-transform coded images," *IEEE Signal Processing Letters*, Vol. 5, No. 2, Feb. 98.

[15] W.F. Schreiber, "Image processing for quality enhancement," *Proceedings of the IEEE*, 66 (1978), pp. 1640-1651.

[16] S.J.P. Westen, P.L. Lagendijk and J. Biemond, "Adaptive spatial noise shaping for DCT based image compression," *ICASSP 96 (IEEE International Conference on Acoustics, Speech and Signal Processing)*.

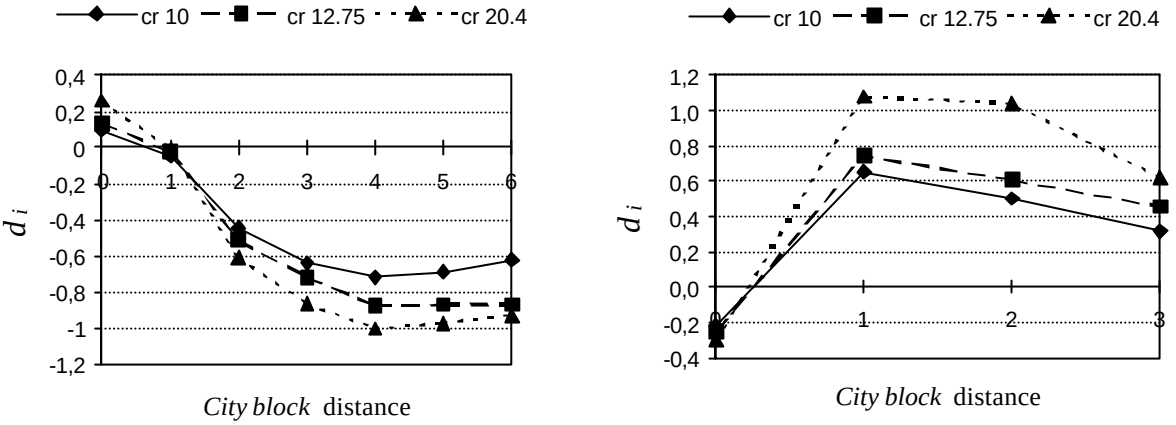
[17] H.H.Y. Tong, and A.N. Venetsanopoulos, "A perceptual model for JPEG applications based on block classification, texture masking, and luminance masking," *IEEE International Conference on Image Processing (ICIP'98)*, Chicago, Ill., October 1998.

[18] D.H. Ballard, C.M. Brown, *Computer Vision*, Prentice Hall, 1982.

[19] M. Cireddu, F.G.B. De Natale, D.D. Giusto, and P. Pes, "Blockness distortion evaluation in

block-coded pictures," *IWISP'96 (International Workshop on Image and Signal Processing), Manchester, UK, 1996.*

[20] P. Fränti, "Blockwise distortion measure for statistical and structural errors in digital images," *Signal Processing: Image Communication*, Vol. 13, pp. 89-98, 1998.



(a) (b)

FIGURE 1

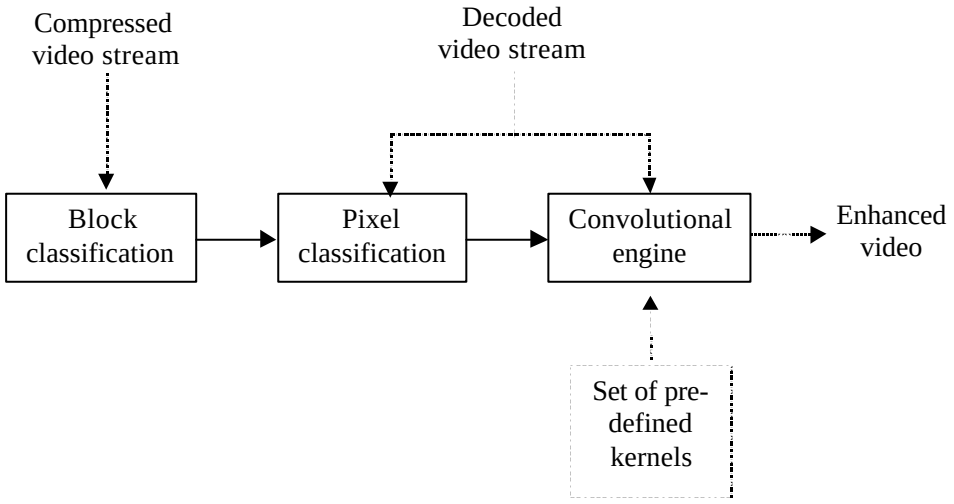


FIGURE 2

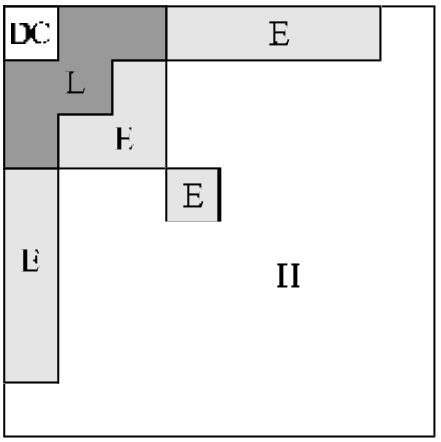


FIGURE 3

(a) (b) (c)

FIGURE 4

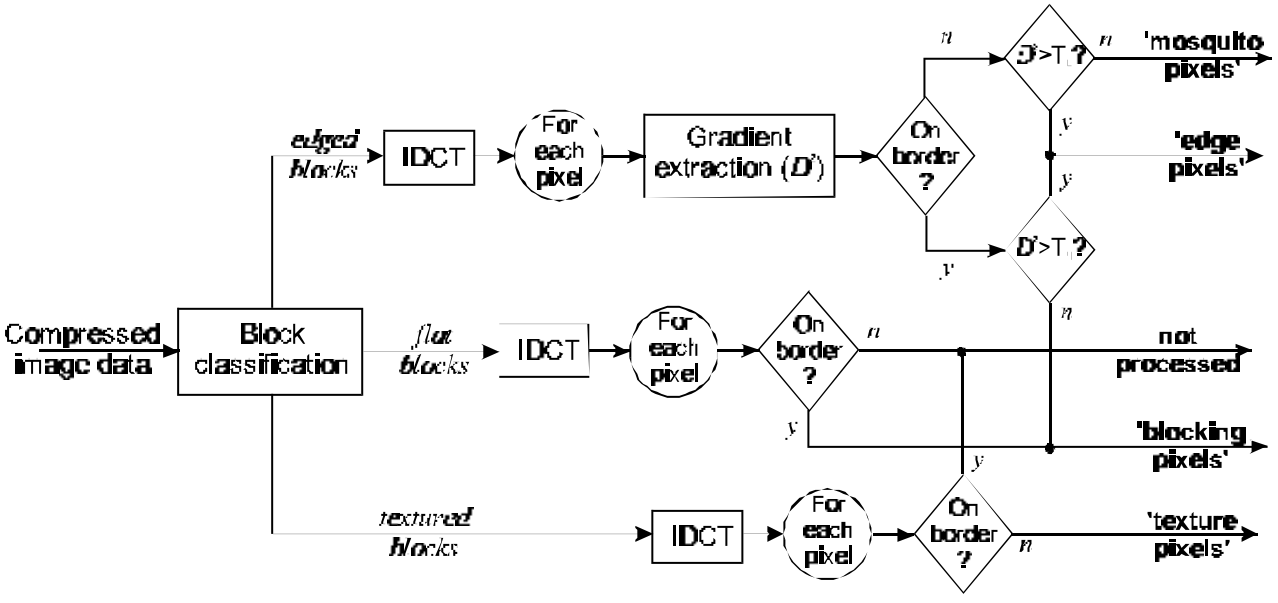


FIGURE 5

$\frac{1}{C}\cdot$

0	-1	0
+1	C	+1
0	-1	0

(a)

$\frac{1}{9+\frac{6}{5}S}\cdot$

1	1+.1S	1
1	1+S	1
1	1+.1S	1

(b)

$\frac{1}{K}\cdot$

-1	-1	-1
-1	8+K	-1
-1	-1	-1

(c)

FIGURE 6

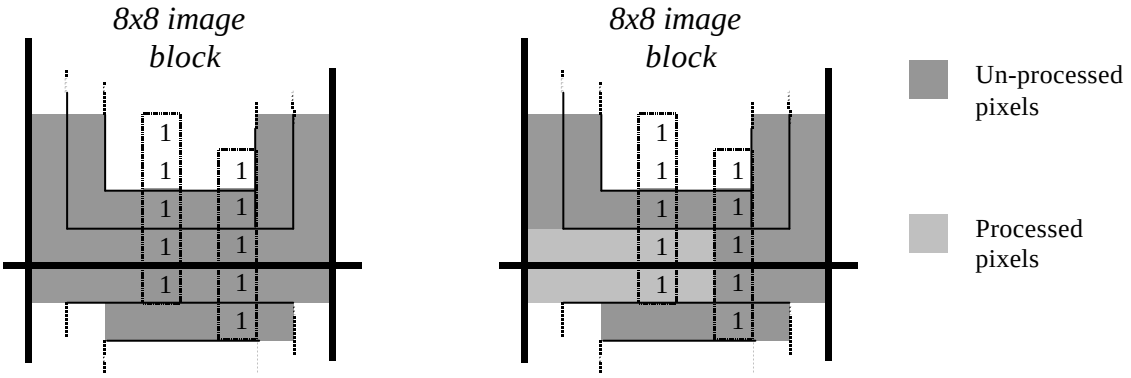
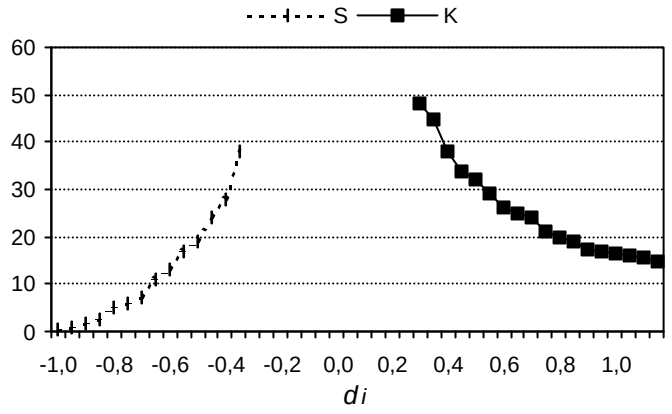


FIGURE 7


S
K
60
50
40
30
20
10
0
-1,0 -0,8 -0,6 -0,4 -0,2 0,0 0,2 0,4 0,6 0,8 1,0
di

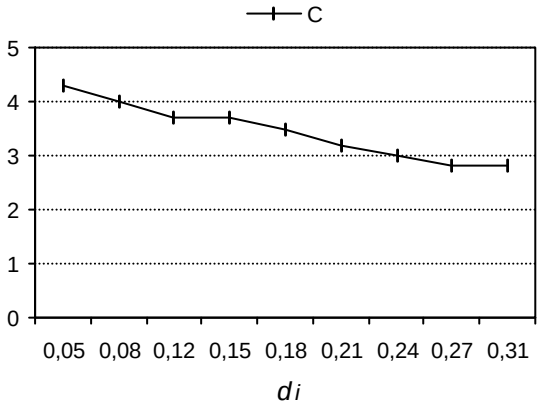

C
5
4
3
2
1
0
0,05 0,08 0,12 0,15 0,18 0,21 0,24 0,27 0,31
di

FIGURE 8

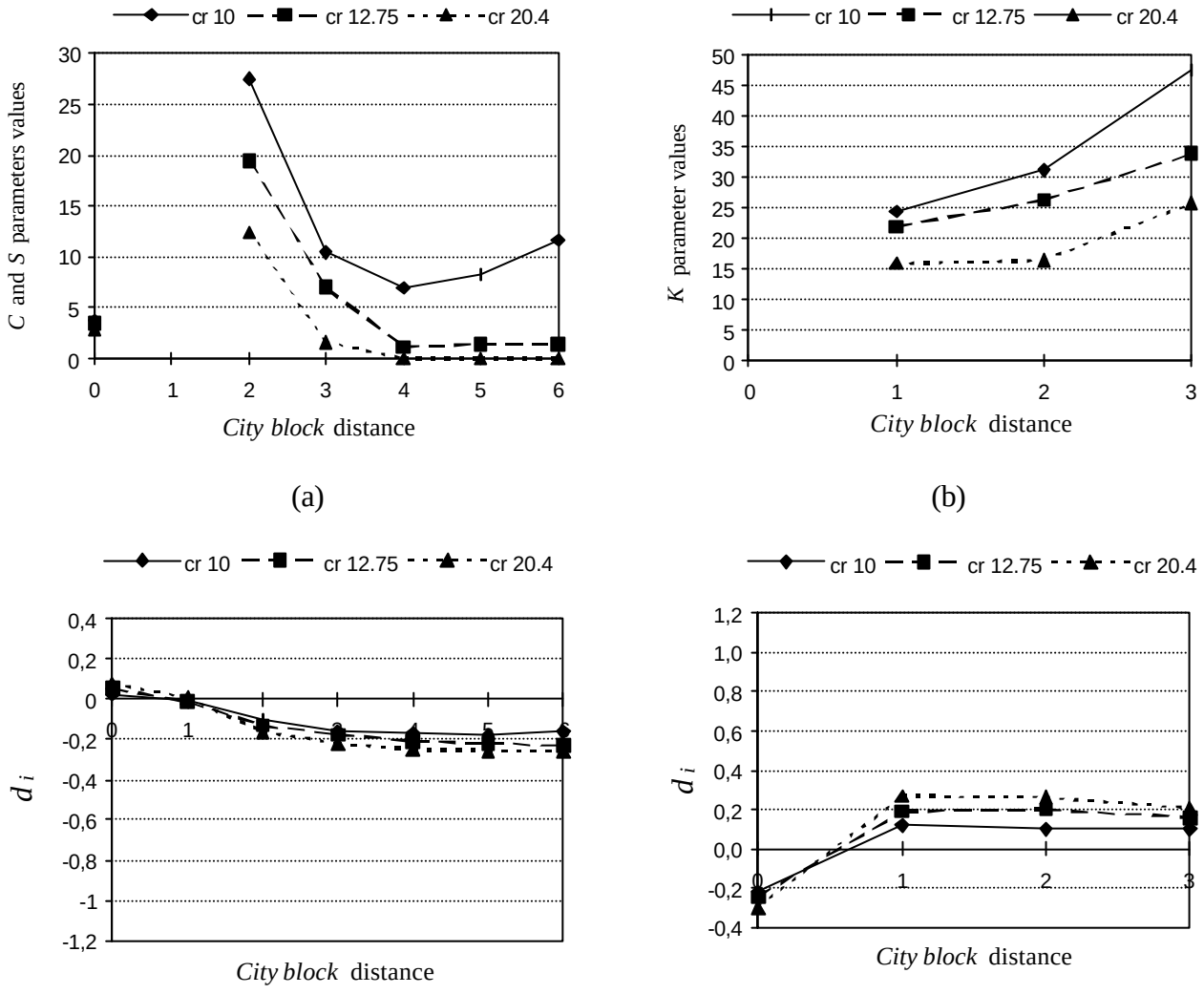


FIGURE 9

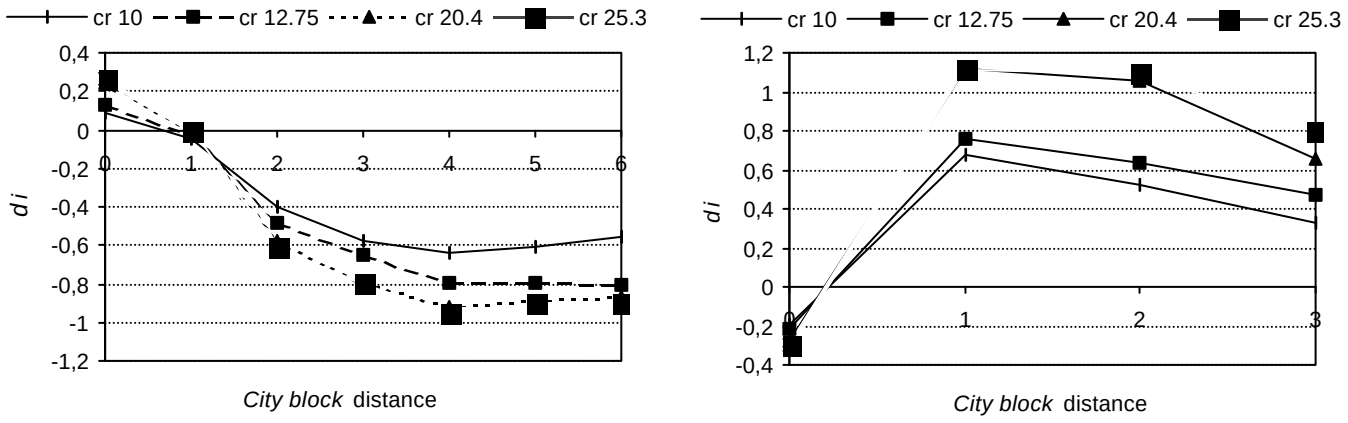
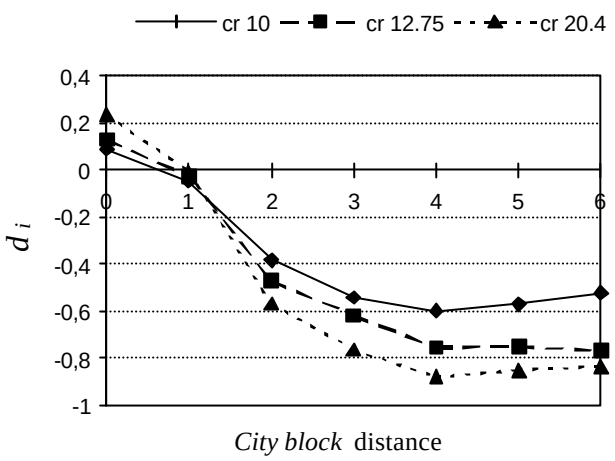
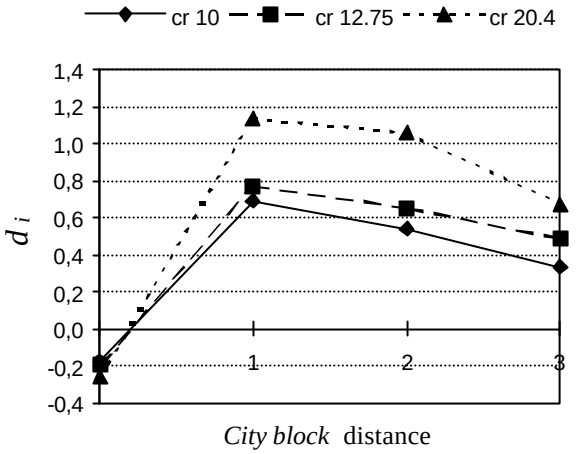


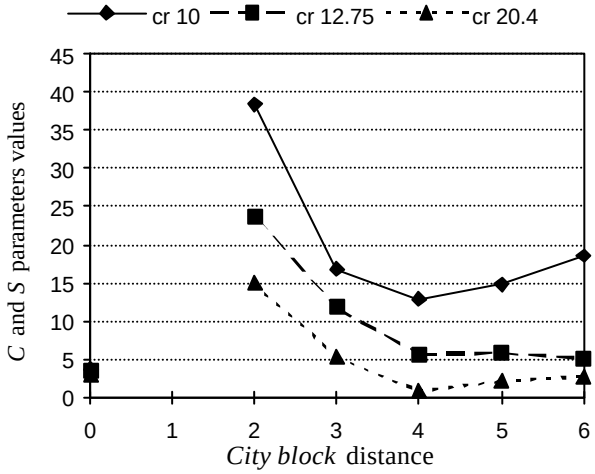
FIGURE 10



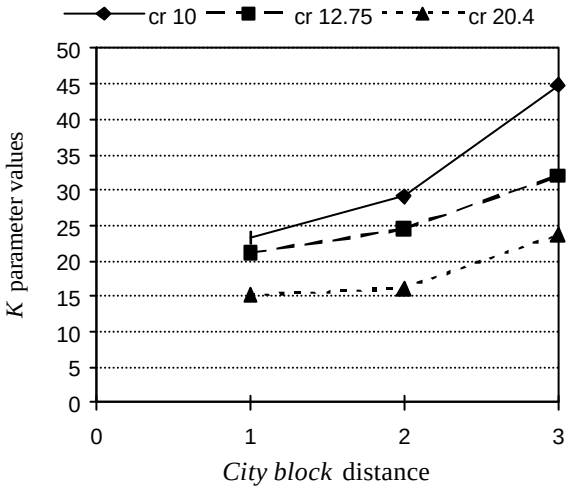
(a)



(b)



(c)



(d)

FIGURE 11

(a) (b)

FIGURE 12

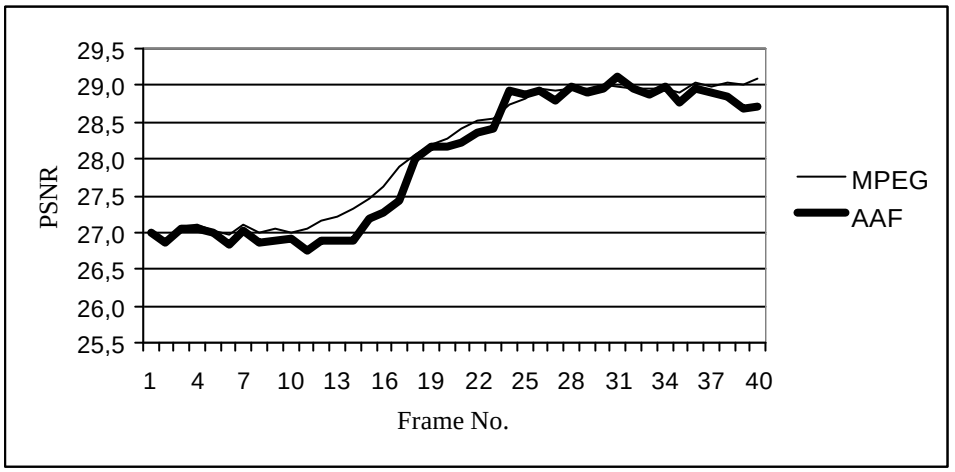


FIGURE 13

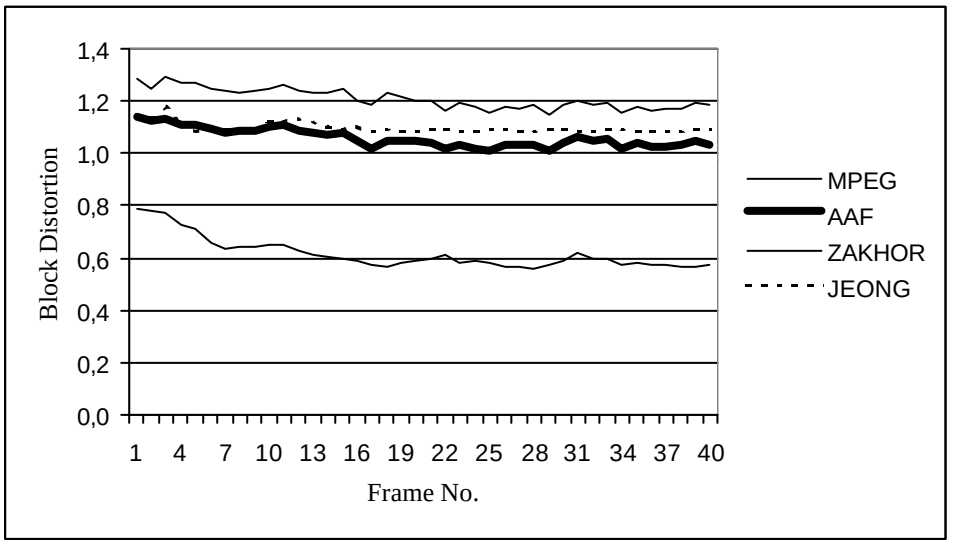
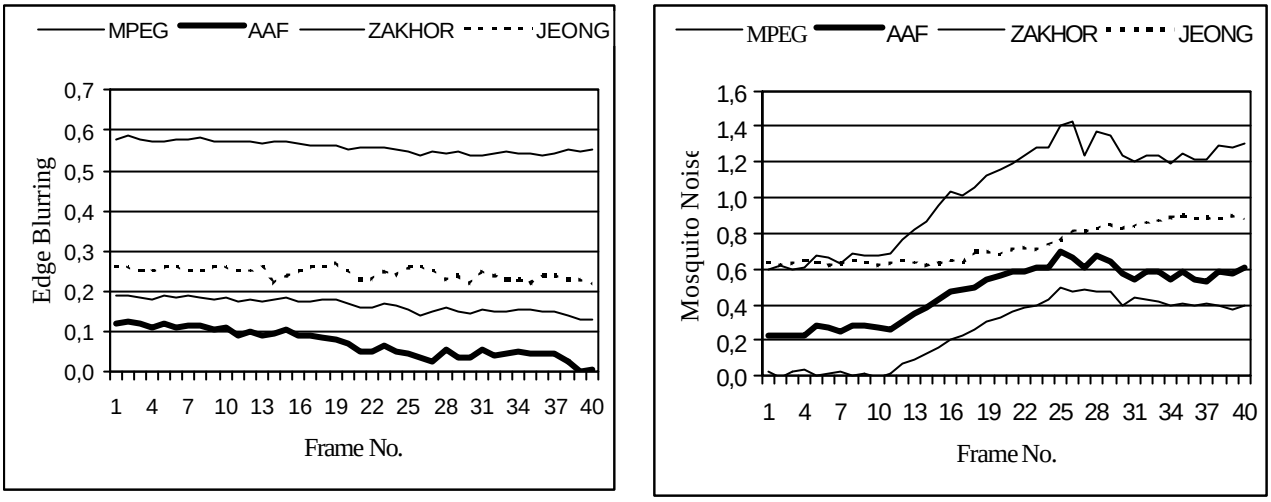


FIGURE 14



(a) (b)

FIGURE 15

(a) (b)

(c) (d)

FIGURE 16

(a) (b)

(c) (d)

FIGURE 17

(a) (b)

(c) (d)

FIGURE 18

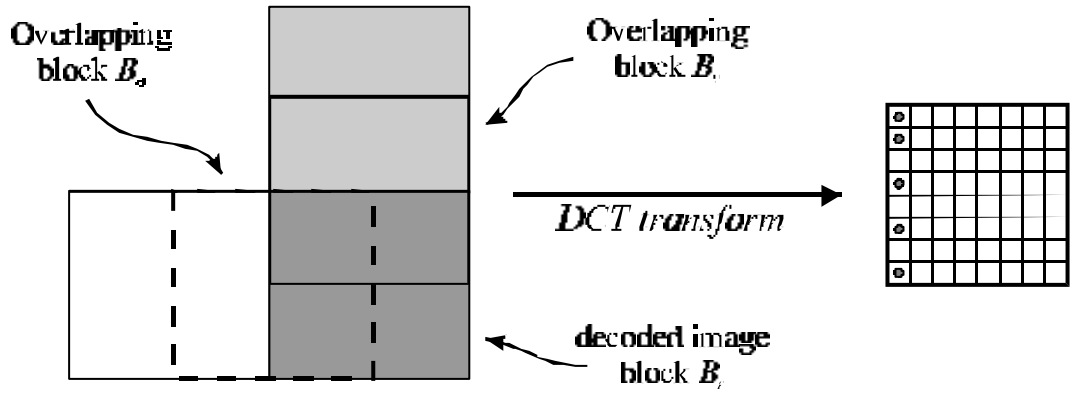


FIGURE 19

TABLE I

	Rai				***Claire***				***Flower Garden***			
	MPEG	Zakhor	Jeong	AAF	MPEG	Zakhor	Jeong	AAF	MPEG	Zakhor	Jeong	AAF
PSNR	28,10	27,56	25,72	27,76	38,46	28,17	34,06	30,72	23,06	20,42	21,48	21,39
D_{BL}	0,17	0,56	0,26	0,07	0,05	0,38	0,11	0,01	0,12	0,59	0,15	0,07
D_{MO}	1,03	0,25	0,63	0,47	0,77	0,43	1,29	0,32	0,31	-0,25	0,34	0,02
NBD	1,21	0,62	1,09	1,06	1,07	1,07	1,05	1,04	1,09	0,56	1,07	1,05

Table and illustration captions

Fig. 1. Average d_i curves: (a) pixels belonging to edged blocks, grouped by city block distance from edges (block boundary pixels were not included to exclude blocking effects); (b) pixels belonging to textured blocks, grouped by city block distance from the block border. Both graphs refer to an intra-coded frame of the *Claire* sequence, at three different compression factors.

Fig. 2. Scheme of the proposed enhancement technique.

Fig. 3. Grouping of DCT coefficients used for block classification: DC (DC), low-frequencies (L), edge (E) and high-frequencies (H).

Fig. 4. (a) Original frame of the *Claire* sequence; (b) block classification from intra-coded data; (c) block classification from inter-coded data, using as a reference the third preceding frame.

Fig. 5. Flow graph for the pixel classification phase.

Fig. 6. FIR filtering kernels: (a) edge enhancement, (b) mosquito smoothing, (c) texture crispening. Kernels (a) and (b) are applied to pixels belonging or close to horizontal edges; otherwise, a rotation is applied according to edge orientation.

Fig. 7. Blocking artifact reduction: left, FIR filtering procedure; right, recursive use of the smoothing filter (inner pixel stripe filtering uses previously filtered boundary values).

Fig. 8. Optimal filtering parameter as a function of the local distortion measure: these data were experimentally obtained from an heterogeneous image set.

Fig. 9. Optimum experimental filter parameters for the *Claire* video sequence, calculated according to the d_i curves of Fig. 1: (a) edged blocks (*C* and *S* parameters); (b) textured blocks (*K* parameter); (c-d) d_i curves after filtering.

Fig. 10. Experimentally obtained distortion curve as a function of the compression ratio.

Fig. 11. Average d_i curves and optimum filtering parameters for the *Rai* sequence at three different compression factors: (a) d_i curves for pixels belonging to edged blocks, grouped by city block distance

from edges (block boundary pixels were not included to exclude blocking effects); (b) d_i curves pixels belonging to textured blocks, grouped by city block distance from the block border. (c-d) optimal filtering parameters values.

Fig. 12. Original frames: (a) *Flower Garden*; (b) *Rai*.

Fig. 13. Comparison between MPEG-decoded and post-processed frames in terms of PSNR for the *Rai* sequence.

Fig. 14. Comparison between different post-processing algorithms in terms of normalized block distortion on 40 frames of the *Rai* sequence. Absence of blocking is represented by a unit value of the distortion measure. Values greatly below unit reveal excessive blurring.

Fig. 15. Comparison between different post-processing algorithms on 40 frames of the *Rai* sequence: (a) edge blurring, (b) mosquito noise. Absence of distortion corresponds to zero values.

Fig. 16. Results on a portion of a frame of the *Rai* sequence: (a) MPEG decoded; (b) after AAF post-processing; (c) after Zakhor's algorithm; (d) after Jeong's algorithm.

Fig. 17. Visual comparison of three portions of a frame of the *Claire* sequence: (a) MPEG decoded; (b) AAF; (c) Zakhor; (d) Jeong.

Fig. 18. Visual comparison of three portions from a frame of the *Flower Garden* sequence: (a) MPEG decoded; (b) AAF; (c) Zakhor; (d) Jeong.

Fig. 19. Evaluation of the tiling effect from the analysis of DCT coefficients calculated on blocks overlapping vertical (B_H) and horizontal (B_V) MPEG block boundaries. The strength of blocking distortion is estimated by analyzing the energy of the B_V and B_H DCT coefficients. On the right, the transform matrix for a B_V block is drawn.

Table 1. Comparison of different post-processing algorithms in terms of PSNR, edge blurring, mosquito noise and blocking distortion for three test sequences.