Improving MPEG-4 coding performance by jointly optimising compression and blocking effect elimination

Wan-Fung Cheung and Yuk-Hee Chan

Centre for Multimedia Signal Processing
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hong Kong

ABSTRACT

In most current block-based image/video coding systems, the compression stage and the deblocking stage operate separately and hence they cannot make use of each other to optimise the overall coding performance. In this research, we suggest modifying the basic structure of the encoding systems such that the deblocking to be performed can be taken into account in the compression and the two processes can be jointly optimised. An example is also provided to show how this idea works successfully in a MPEG-4 codec to boost the rate-distortion performance when the suggested deblocking filter is exploited in the post-processing stage. The modification does not change the bit stream format of the codec and hence is fully compatible with the original standard.

INTRODUCTION

In low bit-rate block-based DCT [1] image/video coding, noise caused by the coarse quantization of transform coefficients is noticeable as visible discontinuities among adjacent blocks in the reconstructed images. In order to remove this blocking artifacts while maintaining the compatibility with current industrial standards, various post-processing techniques such as block-boundary filtering [2-5], maximum a posteriori (MAP) methods [6], projections onto convex sets (POCS) [7-10], transform domain filtering [11,12] and others [13-15] have been proposed. Most of these post-processing techniques were proved to be very effective to eliminate the blocking effect.

Figure 1 shows the general structure of current image/video encoding systems. In this structure, a post-processing is performed to a decompressed image in the receiver. Since the compression and the post-processing processes operate independently in this configuration, the encoder does not make use of the a priori knowledge about the post-processing technique adopted in the decoder to optimise the overall coding performance.

We suggest modifying the basic structure of the encoding system as shown in Figure 2 to improve its coding performance. In this approach, the compression and the post-processing processes are jointly optimised in the encoder according to *f-f'*. Note the approach proposed in H.263+'s Annex J also includes a de-blocking filter in the coding loop[16]. However, its purpose is to use the post-processed output instead of the original of frame n in coding frame n+1 so as to improve the subjective quality. It is not for jointly optimising the two processes.

In this paper, we will use an example to show how this idea works. We optimise a MPEG-4 [17] encoder in a case that a deblocking filter [18] is used in the decoder to eliminate the blocking effect such that the quality of the output will be better than that without optimisation.

MPEG4 ENCODER OPTIMISATION

In conventional schemes, the MPEG-4 encoder partitions the original images **I** into blocks of size 8x8, say $\mathbf{I}_{m,n}$'s and then encodes the blocks with DCT block by block with a raster scan strategy. At the decoder side, the deblocking filter [18] performs one-dimensional filtering along the 8x8 block edges to eliminate the discontinuity among adjacent blocks. It is applied to all the block boundaries first along the horizontal edges followed by the vertical edges. If a pixel intensity is changed by the previous filtering operation, the updated pixel intensity is used for the next filtering. This filter has two separate filtering modes: *Smooth*

region mode and *Default mode*. *Smooth region mode* is applied in smooth region to smooth 8-pixel line segments which are perpendicular to the block's boundary, while *default mode* is applied in complex regions to smooth 2-pixel line segments.

We modify the MPEG-4 encoder according to the post-processing process such that the compression and the deblocking processes can make use of each other to optimise the overall coding performance. The philosophy of our approach is very simple. Instead of encoding block $\mathbf{I}_{\mathbf{m,n}}$ directly, we encode its modified version, say $\mathbf{Y}_{\mathbf{m,n}}$, such that $\|F\{(T^{-1}\{Q\{T\{\mathbf{Y}_{\mathbf{m,n}}\}\}\})^*\} - \mathbf{I}_{\mathbf{m,n}}^*\| < \|F\{(T^{-1}\{Q\{T\{\mathbf{I}_{\mathbf{m,n}}\}\}\})^*\} - \mathbf{I}_{\mathbf{m,n}}^*\|$. Here, $\mathbf{B}_{\mathbf{m,n}}^*$ denotes the greater area of a block named as $\mathbf{B}_{\mathbf{m,n}}$, and, operators $T\{.\}$, $Q\{.\}$, $T^{-1}\{.\}$ and $F\{.\}$, respectively, perform the DCT transformation, the quantization, the inverse DCT transformation and the deblocking filtering [18] proposed for MPEG4 standard. The greater area of a block is shown in Figure 3. Note the support region of operation $F\{.\}$ is the greater area of the processing block instead of the block itself and hence we have to consider the distortion introduced in the greater area by the modification. One may consider the modification as a compensation for the noise introduced by deblocking. Since this modification is carried out before transform coding, it is referred to as pre-processing in this paper.

The modification of $\mathbf{I}_{\mathbf{m,n}}$ is carried out with an optimisation filter. Its design corresponds to that of the deblocking filter used in the post-processing stage. Accordingly, it is one-dimensional and has two separate operation modes. In particular, it operates in *Smooth Region Mode* to handle smooth region and *Default Mode* to handle complex region as the deblocking filter does.

COMPENSATION FOR POST-PROCESSING

Without loss of generality, consider we are now going to encode block $I_{m,n}$. In order to let readers have a better insight of the approach we proposed, we start the formulation of our approach from the MPEG-coded version of $\mathbf{Y}_{\mathbf{m,n}}$, say $\mathbf{C}_{\mathbf{m,n}}$, instead of $\mathbf{I}_{\mathbf{m,n}}$. Let $\vec{v} = (v_0, v_1, v_2, ...v_8, v_9)$ be a particular pixel vector across the boundary of block $\mathbf{C}_{\mathbf{m,n}}$ and block $\mathbf{C}_{\mathbf{m-1,n}}$ as shown in Figure 4. At the decoder, this pixel vector will be post-processed with the one-dimensional deblocking filter[18]. To determine in which mode the deblocking filter will operate, the following measurement is performed.

$$M(\vec{v}) = \sum_{i=0}^{8} \phi(v_i - v_{i+1}), \text{ where } \phi(\Delta) = \begin{cases} 1 & if \ |\Delta| \le T_1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where T_1 is a pre-defined threshold. Function $M(\vec{v})$ is used to determine the flatness of the local region. If $\vec{v}$ is complex enough to make $M(\vec{v})$ smaller than a predefined threshold T_2, we assume that the deblocking filter will operate in *default mode*. Otherwise, *smooth region mode* is assumed. Compensation for the deblocking process is then performed in the encoder accordingly.

A. Compensation for smooth region mode deblocking

In MPEG4, when the deblocking filter works in the smooth region mode, a nine-tap smoothing filter is applied inside the block as well as on the block boundaries if no edge is detected. There is a detailed description of the filter operated in this mode in [18]. In particular, the post-processing output of a particular pixel vector $\vec{v}_A=(v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8)$, say $\vec{p}=(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8)$, is given in matrix form by

$$\vec{p}=\vec{v}_A F + s_0 P_a + s_1 P_b \tag{2}$$

where $s_0 = \begin{cases} v_0 & if \ |v_1 - v_0| < QP \\ v_1 & otherwise \end{cases}$

$s_1 = \begin{cases} v_9 & if \ |v_9 - v_8| < QP \\ v_8 & otherwise \end{cases}$

$P_a = (6 \quad 4 \quad 2 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0)/16,$

$P_b = (0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 4 \quad 6)/16,$

$F = \begin{pmatrix} F_U \\ F_L \end{pmatrix},$

$F_U = \frac{1}{16}\begin{pmatrix} 4 & 2 & 2 & 1 & 1 & 0 & 0 & 0 \\ 2 & 4 & 2 & 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 4 & 2 & 2 & 1 & 1 & 0 \\ 1 & 2 & 2 & 4 & 2 & 2 & 1 & 1 \end{pmatrix}$ and $F_L = \frac{1}{16}\begin{pmatrix} 1 & 1 & 2 & 2 & 4 & 2 & 2 & 1 \\ 0 & 1 & 1 & 2 & 2 & 4 & 2 & 2 \\ 0 & 0 & 1 & 1 & 2 & 2 & 4 & 2 \\ 0 & 0 & 0 & 1 & 1 & 2 & 2 & 4 \end{pmatrix}.$

Here, QP is the Quantization Parameter defined in MPEG4 standard.

The mean square deviation of $\vec{p}$ from its corresponding pixel vector in the original image, say $\vec{i}$, is then given by:

$$\xi = \left\| (\vec{p} - \vec{i}) \right\|^2 \tag{3}$$

In theory, we should look for an appropriate $\vec{v}_A$ to minimize ξ. However, in order to reduce

the intervention between blocks so as not to complicate the situation, we only adjust the values of $\vec{v}_2 = (v_5, v_6, v_7, v_8)$ in the current processing block to minimize ξ. By doing so, any adjustment performed during processing the current block $\mathbf{I_{m,n}}$ is limited to the block itself.

The values of $\vec{v}_2$ that minimize ξ can be obtained by solving equations $\partial\xi / \partial\vec{v}_2 = 0$. To get the formulation, let us first rewrite (2) as

$$\vec{p} = \vec{v}_A F + s_0 P_a + (l_0 v_8 + l_1 v_9) P_b \tag{4}$$

where $l_0 = \begin{cases} 1 & \text{if } s_1 = v_8 \\ 0 & \text{otherwise} \end{cases}$ and

$$l_1 = \begin{cases} 1 & \text{if } s_1 = v_9 \\ 0 & \text{otherwise} \end{cases}$$

Then, from (3) and (4), we have

$$\frac{\partial\xi}{\partial\vec{v}_2} = 2(\vec{p} - \vec{i})(F_L + l_0 P_c)^T \tag{5}$$

where $$P_c = \frac{\partial(v_8 P_b)}{\partial\vec{v}_2} = \frac{1}{16}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 4 & 6 \end{pmatrix}$$

By solving $\partial\xi / \partial\vec{v}_2 = 0$, we have

$$\vec{v}_2 = (\vec{i} - \vec{v}_0 F_U - s_0 P_a - l_1 v_9 P_b) F_c^T (F_c F_c^T)^{-1} \tag{6}$$

where $F_c = F_L + l_0 P_c$ and $\vec{v}_0 = (v_1, v_2, v_3, v_4)$.

B. <u>Compensation for default mode deblocking</u>

In the default mode, the block being processed is considered to be in a complex region and the deblocking filter adopted in MPEG-4 will smooth the two block-boundary pixel v_4 and v_5 only. The values of v_4 and v_5 are modified according to the feature information carried by the pixel vector. In particular, the feature information is described by

$$a_{3,i} = \vec{v}_i \cdot K \qquad \text{for } 0 \le i \le 2 \tag{7}$$

where $K = (k_3, -k_1, k_1, -k_3)^T$ is a basis function of a 4-point DCT and $\vec{v}_i$'s are 3 sub-vectors of the pixel vector $\vec{v}$ as shown in Figure 4. Note that $a_{3,i}$ is actually the highest four-point DCT coefficient of $\vec{v}_i$. Accordingly, we have $k_3 = \frac{1}{\sqrt{2}}\cos(3\pi/8)$ and $k_1 = \frac{1}{\sqrt{2}}\cos(\pi/8)$.

To a certain extent, coefficient $a_{3,1}$ is correlated with the blocking artifacts and block discontinuity can be reduced by scaling down $a_{3,1}$ adequately. In particular, the post-processing output of pixels $\vec{v}_A$, say $\vec{q}$, is given by

$$\vec{q} = \vec{v}_A + d \cdot D \tag{8}$$

where $D = (0,0,0,-1,1,0,0,0)$, $d = CLIP(\ k_1(a'_{3,1} - a_{3,1}),\ 0,\ (v_4 - v_5)/2)$, and $a'_{3,1} = a_{3,1} \cdot \min(|a_{3,0}|, |a_{3,1}|, |a_{3,2}|) / |a_{3,1}|$. The clipping operation is used to make sure that the magnitude of the gradient at the boundary is reduced without a change in direction and that v_4 and v_5 are not out of range. Clipping is a nonlinear operation and hence makes the situation very complicated in solving an optimisation problem. In order to solve this problem, we ignore the clipping operation during the formulation of our approach. A checking mechanism will be used to compensate for any error introduced by this action. This mechanism will be presented later on.

The mean square deviation of $\vec{q}$ from its corresponding pixel vector in the original image, say $\vec{i}$, is then given by:

$$\xi = \|(\vec{q} - i)\|^2 \tag{9}$$

Again, we want to look for an appropriate $\vec{v}_2$ to minimize ξ. Here, we consider the two possible cases, namely, (i) $|a_{3,1}| = \min(|a_{3,0}|, |a_{3,1}|, |a_{3,2}|)$ and (ii) $|a_{3,1}| \neq \min(|a_{3,0}|, |a_{3,1}|, |a_{3,2}|)$, separately. For the former case, we have $a'_{3,1} = a_{3,1}$, which results in d=0 and hence no postprocessing will be performed at the decoder. Accordingly, no pre-processing is required in this case. As for the latter case, we get the desirable $\vec{v}_2$ by solving equations $\partial\xi / \partial\vec{v}_2 = 0$.

First of all, we rewrite eqn. (8) as

$$\vec{q} = \vec{v}_A + (h_0 a_{3,0} + h_2 a_{3,2} - a_{3,1}) k_1 D \tag{10}$$

where $h_0 = \begin{cases} 1 & \text{if } |a_{3,0}| \text{ is min, } |a_{3,1}| \neq 0 \text{ and } a_{3,1} \times a_{3,0} > 0 \\ -1 & \text{if } |a_{3,0}| \text{ is min, } |a_{3,1}| \neq 0 \text{ and } a_{3,1} \times a_{3,0} < 0 \\ 0 & \text{otherwise} \end{cases}$ and

$h_2 = \begin{cases} 1 & \text{if } |a_{3,2}| \text{ is min, } |a_{3,1}| \neq 0 \text{ and } a_{3,1} \times a_{3,2} > 0 \\ -1 & \text{if } |a_{3,2}| \text{ is min, } |a_{3,1}| \neq 0 \text{ and } a_{3,1} \times a_{3,2} < 0 \\ 0 & \text{otherwise} \end{cases}$

By substituting (7) into (10), we have

$$\vec{q} = \vec{v}_A + h_0 k_1 D\vec{v}_0 K + h_2 k_1 D\vec{v}_2 K - k_1 D\vec{v}_1 K \tag{11}$$

where $\vec{v}_0 = (v_1, v_2, v_3, v_4), \vec{v}_1 = (v_3, v_4, v_5, v_6)$ and $\vec{v}_2 = (v_5, v_6, v_7, v_8)$.

Then, from (9) and (11), we have

$$\frac{\partial \xi}{\partial \vec{v}_2} = 2(\vec{q} - \vec{i})(h_2 C_1 + R_1)^T \tag{12}$$

where $R_1 = \begin{pmatrix} 0 & 0 & 0 & k_1^2 & 1-k_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -k_1 k_3 & k_1 k_3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ and $C_1 = \begin{pmatrix} 0 & 0 & 0 & -k_1 k_3 & k_1 k_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1^2 & -k_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -k_1^2 & k_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & k_1 k_3 & -k_1 k_3 & 0 & 0 & 0 \end{pmatrix}$

The desirable $\vec{v}_2$ can then be obtained by solving the equation $\partial \xi / \partial \vec{v}_2 = 0$. Specifically, we have

$$\vec{v}_2 = (\vec{i} - \vec{v}_0 C_2 - h_0 \vec{v}_0 C_1) W^T (W W^T)^{-1} \tag{13}$$

where $W = h_2 C_1 + R_1$ and $C_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & k_1 k_3 & -k_1 k_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-k_1^2 & k_1^2 & 0 & 0 & 0 \end{pmatrix}$

Pixel vectors in the processing block are processed accordingly one by one in a predefined order until the whole desirable $\mathbf{C_{m,n}}$ is obtained. If a pixel is modified, its most updated value will be used as the input for the current processing.

After obtaining the desirable $\mathbf{C}_{m,n}$, $\mathbf{Y}_{m,n}$ can then be determined. In theory, $\mathbf{Y}_{m,n}$ should be an array that makes $\mathbf{C}_{m,n} = T^{-1}\{Q\{T\{\mathbf{Y}_{m,n}\}\}\}$. However, this cannot be practically achieved as quantization is involved and $\mathbf{C}_{m,n}$ may not be one of the possible DCT-coded outputs provided by the encoder. In our approach, we let $\mathbf{Y}_{m,n} = \mathbf{C}_{m,n}$.

By doing so, sometimes the modification made by the optimisation filter cannot guarantee that $F\{(T^{-1}\{Q\{T\{\mathbf{Y}_{m,n}\}\}\})^*\}$ is more faithful to $\mathbf{I}_{m,n}^*$ than $F\{(T^{-1}\{Q\{T\{\mathbf{I}_{m,n}\}\}\})^*\}$. Hence, the encoder should determine whether $\mathbf{Y}_{m,n}$ or $\mathbf{I}_{m,n}$ should be encoded. The selection is carried out by checking their post-processed results against the original image. The one which provides the minimum mean square error (MSE) is then selected. We proceed to process the next block until the whole image is processed. This checking mechanism also excludes any possible degradation caused by the presumptions we have made. That there is no clipping in the compensation for default mode deblocking is one of the presumptions. Figure 5 summaries the flow of the proposed pre-processing scheme. In Figure 5, $\Gamma_s\{.\}$ and $\Gamma_d\{.\}$ are, respectively, the operators that perform the compensation for smooth region mode and default mode deblocking. The operator $\Lambda\{.\}$ in $\mathbf{Y'}_{m,n} = \Lambda\{\mathbf{Y}_{m,n}, \vec{v} \rightarrow \vec{v}'\}$ replaces pixel vector $\vec{v}$ in $\mathbf{Y}_{m,n}$ by $\vec{v}'$ to output $\mathbf{Y'}_{m,n}$.

Note the flow presented in Figure 5 is for illustration only. The realisation effort required is much less than it appears to be as some steps involved in the flow can be reformulated and combined to reduce the computation effort. For instance, it is not necessary to perform two DCTs for each pixel vector to compute $\mathbf{\Omega'}$. Since T{.} is a linear operation, we have $T\{\mathbf{Y'}_{m,n}\}=T\{\mathbf{Y}_{m,n}\}+T\{\mathbf{Y'}_{m,n}-\mathbf{Y}_{m,n}\}$, where $T\{\mathbf{Y}_{m,n}\}$ is known in the current stage. There are only 4 nonzero elements in $\mathbf{Y'}_{m,n}-\mathbf{Y}_{m,n}$ and hence the computation effort of $T\{\mathbf{Y'}_{m,n}\}$ is little. Similarly, as $T^{-1}\{.\}$ is also a linear operation and most of the components of $\{Q\{T\{\mathbf{Y'}_{m,n}\}\}-Q\{T\{\mathbf{Y}_{m,n}\}\}\}$ are zero after quantization, $T^{-1}\{Q\{T\{\mathbf{Y'}_{m,n}\}\}\}$ can be obtained with a few number of arithmetic operations. Though eqns. (6) and (13) look complex, the realisation of $\Gamma_s\{\vec{v}\}$ and $\Gamma_d\{\vec{v}\}$ is simple as it involves a lot of zero multiplication. Finally, as a matter of fact, the encoded result of $\mathbf{Y}_{m,n}$ has already been determined in the pre-processing stage and hence no transform coding is actually required after the pre-processing even though it is termed as "pre-processing".

Only blocks that contain non-zero quantized DCT ac-coefficients are processed as not doing so will eventually require more bits to encode the modified block even though there is some gain in terms of MSE.

OPTIMISATION ORDER

As the optimisation filter performs one-dimensional filtering along the 8x8 block edges, corner pixels may be modified twice. The two modifications may be counteracted with each other and the consequence is difficult to predict. In order to minimise the interference among the modification results, a working pattern which specifies the order of pixels to be filtered must be carefully planned. In this paper, we propose two ordering strategy, say, *Raster Scanning Strategy* (RSS) and *Checker Scanning Strategy* (CSS), to modify $\mathbf{I}_{m,n}$'s to $\mathbf{Y}_{m,n}$'s.

In RSS, blocks are processed one by one from left to right and from top to bottom. For a particular block, the optimisation filtering starts at the left-lower corner of a block and moves clockwise to the right-upper corner of the block. We proceed to process and encode the next block until the whole image is encoded. This scheme does not optimise pixel vectors across the right and the bottom boundaries of a block.

In CSS, blocks are processed in an order as shown in Figure 6. We process and encode the blocks at the gray color area first and then white color area until the whole image is encoded. The optimisation filtering starts at the left-lower corner of a block and moves clockwise around the block. Hence, we can modify all pixel vectors around the boundaries of the block. Besides, the blocks in the area of the same colour can be processed in parallel to reduce processing time as the modification of these blocks does not affect each other. Simulation result shows that CSS is better than RSS.

As the modification made in a particular block will influence the deblocking result of the previously modified blocks next to it, the pre-processing is performed in an iterative way in order to reach an optimised output. From our simulation result, we found that it converged rapidly and was able to reach an optimised rate-distortion value after 2 iterations when CSS was adopted.

SIMULATION RESULT

Simulations were carried out with various testing sequences such as *Akiyo, Car Phone, Claire, Foreman* and *Silent* to demonstrate that a corresponding pre-processing step could definitely improve the coding performance. For the sake of reference, the coding scheme without either pre- or post-processing, the coding scheme with pure post-processing [18] and the proposed coding scheme are, respectively, referred to as CS, CPS and PCPS hereafter.

More specifically, we will use $\mathrm{PCPS_r}$ and $\mathrm{PCPS_c}$, respectively, to denote the PCPS using RSS and CSS.

In our simulations, the MPEG-4 video VM 9.1 coder [17] was used. The ITU-T Recommendation H.263 [19] quantization method was adopted and all frames were encoded with a fixed quantization parameter (QP). Threshold values T_1 and T_2 were, respectively, selected to be 2 and 6. As it is suggested in [18], k_1 and k_3 were, respectively, approximated to be 0.625 and 0.25 so as to reduce realisation effort. The motion search range was [-16, 15.5]. For comparison, we modify the criterion for the selection between $\mathbf{Y_{m,n}}$ and $\mathbf{I_{m,n}}$ in the realisation of PCPS. In particular, $\mathbf{Y_{m,n}}$ is selected only if it provides a smaller MSE and the encoded bit stream generated so far is shorter than that generated so far by CPS for the current frame. This makes PCPS always produce a shorter bit stream as compared with PS and CPS. All PCPS results shown in the paper are obtained with 2 iterations unless it is specified.

In our first simulation, the performance of the proposed approach was investigated in terms of PSNR. Here, PSNR is defined to be $\mathrm{PSNR} \equiv 10\log_{10}\left(255^2 \cdot S / \|f-g\|^2\right)$, where f and g are, respectively, the original image and the reconstructed image, and, S is the size of the images in terms of number of pixels. Sequences of various motion characteristics were used for evaluation. Figures 7a, c, e and g show the case that all frames were encoded as I frames (Intra-mode) while Figures 7b, d, f and h show the case that all frames except the 1st one were encoded as P frames (Inter-mode). Observations can be obtained form the figures. First, in all circumstances, the proposed pre-processing schemes definitely improve the PSNR performance. Second, $\mathrm{PCPS_c}$ can provide a higher PSNR gain than $\mathrm{PCPS_r}$. Third, the ratio of PCPS's gain in SNR with respect to CS to CPS's is larger in Inter-mode than in Intra-mode. This is favorable as a video codec usually operates in Inter-mode instead of Intra-mode in most circumstances.

A simulation was carried out to investigate the convergence of $\mathrm{PCPS_c}$. In this simulation, video sequences *Akiyo* and *Grandma* were encoded in Inter-mode. Figure 8 shows the simulation results. One can see that, at different bit rates, it converges quickly and there is nearly no further SNR improvement after 2 iterations. Figure 8 also shows that $\mathrm{PCPS_c}$ can provide a greater rate-distortion improvement with respect to CPS at high and low bit rate.

Table 1 shows the rate-distortion performance of various schemes at different conditions for testing video sequences of different image characteristics. In this part of simulation, all video sequences are encoded in Inter-mode and the checker scanning strategy is adopted in the pre-processing scheme. One can see that $PCPS_c$ can generally achieve a two-fold SNR improvement as compared with CPS while the bit stream is 2 to 20% shorter than that of the original on average. The improvement is robust with respect to various image characteristics and compression rates.

Figure 9 shows the PSNR improvements achieved by $PCPS_c$ and CPS with respect to CS at a spread range of compression rates. One can easily see that the two-fold improvement achieved by $PCPS_c$ can be maintained over a spread range of compression rates for video sequences of different image characteristics.

Figure 10 shows some deblocking results of different coding schemes. Obviously, the proposed pre-processing scheme can reduce blocking artifacts further. This can be observed by examining the texture detail around the body and the tail of the fish in Figure 10.

Based on the simulation results, it is clear that the proposed pre-processing scheme does not only reduce blocking artifacts effectively, but also improve the rate-distortion performance when it is used in MPEG-4 video coding compression. Note the improvement is robust with respect to various conditions. This supports that taking post-processing into account to jointly optimise encoding and de-blocking is encouraging.

CONCLUSION

In this paper, a pre-processing method is proposed and applied in MPEG-4 video coding to improve the coding performance by jointly optimising both compression and block-effect elimination. The simulation results show that the proposed method can improve video quality both subjectively and objectively without changing the core structure and the bit-stream format of a MPEG4 codec. Specifically, a two-fold gain in rate-distortion performance can be achieved by the proposed method when the suggested deblocking filter is exploited in the post-processing stage.

ACKNOWLEDGEMNTS

The work described in this paper was substantially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region (Project No. PolyU 5092/98E).

REFERENCES

[1] Rao, K.R. and Yip, P.: "Discrete Cosine Transform: Algorithms, Advantages, Applications," *Academic Press,* 1990.

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

[3] Chan, Y.H., Hong, S.W. and Siu, W.C.: "A practical post-processing technique for real-time block-based coding system," *IEEE trans. on CAS for Video Technology*, Feb 1998, Vol.8, No.1, pp.4-8.

[4] Kuo, C.J. and Hsieh, R.J.: "Adaptive postprocessor for block encoded images", *IEEE Trans. CAS. Video Technol.*, Aug 1995, Vol. 5, No. 4, pp.298-304.

[5] Ramamurthi, B. and Gersho, A.: "Nonlinear space-variant postprocessing of block coded images", *IEEE Trans.*, Oct 1986, Vol. ASSP-34, No. 5, pp.1258-1267.

[6] O'Rourke, T.P. and Stevenson, R.L.: "Improved image decompression for reduced transform coding artifacts" *IEEE Trans. CAS. Video Technol.*, Dec 1995, Vol. 5, No. 6, pp.490-499.

[7] Zakhor, A.: "Iterative procedures for reduction of blocking effects in transform image coding", *IEEE Trans. CAS. Video Technol.*, Mar 1992, Vol. 2, No. 1, pp.91-95.

[8] Hsung, T.C., Lun, P.K. and Siu, W.C.: "A deblocking technique for block-transform compressed image using wavelet transform modulus maxima," IEEE Transactions on Image Processing, Oct 1998, Vol.7, No.10, pp.1488 -1496.

[9] Yang, Y., Galatsanos, N.P. and Katsaggelos, A.K.: "Projection-based spatially adaptive reconstruction of block-transform compressed images", *IEEE Trans. CAS Video Technol.*, Jul 1995, Vol. 4, No. 7, pp.896-908.

[10] Yang, Y., Galatsanos, N.P. and Katsaggelos, A.K.: "Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images", *IEEE Trans. CAS Video Technol.*, Dec 1993, Vol. 3, No. 6, pp.421-432.

[11] Choy, S.S.O., Chan, Y.H. and Siu, W.C.: "Reduction of block-transform image coding artifacts by using local statistics of transform coefficients", *IEEE Signal Processing Letters*, Jan 1997, Vol.4, No.1, pp.5-7.

[12] Xiong, Z., Orchard, M.T. and Zhang, Y.Q.: "A deblocking algorithm for jpeg compressed images using overcomplete wavelet representations", *IEEE Trans. CAS.*

Video Technol., Apr 1997, Vol. 7, No. 2, pp.433-437.

[13] Hong, S.W., Chan, Y.H. and Siu, W.C.: "A new approach for real-time reduction of the blocking effect," *Signal Processing*, Mar 1998, Vol. 65, No. 3, pp.337-346.

[14] Minami, S. and Zakhor, A.: "An optimisation approach for removing blocking effects in transform coding", *IEEE Trans. Circuits Syst. Video Technol*., Apr 1995, Vol. 5, No. 2, pp.74-82.

[15] Wu, S.W. and Gersho, A.: "Improved decoder for transform coding with application to the JPEG baseline system," *IEEE Transactions on Communications*, Feb 1992, Vol.40, No.2, pp.251-254.

[16] ITU-T, "Draft Text of Recommendation H.263 version 2 ("H.263+") for Decision", Jan 1999.

[17] "MPEG-4 video verification model V.9.1," ISO/IEC JTC1/SC 29/WG11/N2502a, Oct 1998.

[18] Kim, S. D., Yi, J., Kim, H. M. and Ra, J. B.: "A deblocking filter with two separate modes in block-based video coding", *IEEE Trans. Circuits Syst. Video Technol.*, Feb 1999, Vol. 9, No. 1, pp. 156-160.

[19] ITU-T Recommendation H.263, "Video coding for low bit-rate communication," May 1996.

Sequence	QP	Rate-distortion Performance									
		H263+		MPEG4							
		Annex J : de-blocking filter included in the coding loop[16]		Without Pre-processing			With Pre-processing			Gain in SNR of the proposed scheme, $G_{SNR} \equiv \frac{Q_{pcp} - Q_c}{Q_{cp} - Q_c}$	Gain in Rate of the proposed scheme, $G_{rate} \equiv \frac{B_{pcp} - B_c}{B_c}$ x100%
				Without Post-processing (CS)	With Post-Processing (CPS) [18]	Average bits per frame B_c	Without Post-processing (PCS)	With Post-Processing (PCPS)	Average bits per frame B_{pcp}		
		Average PSNR per frame	Average bits per frame	Average PSNR per frame, Q_c	Average PSNR per frame, Q_{cp}		Average PSNR per frame, Q_{pc}	Average PSNR per frame, Q_{pcp}			
Akiyo	8	35.23	1748	35.51	35.83	1862	35.83	36.31	1592	2.50	-14.50
Akiyo	12	32.89	1290	33.14	33.47	1412	33.51	34.00	1118	2.61	-20.82
Car Phone	14	31.73	3338	31.76	32.03	3326	32.11	32.46	3249	2.59	-2.32
Car Phone	18	30.56	3110	30.65	30.90	3091	30.94	31.31	2973	2.64	-3.82
Claire	9	36.88	1983	36.95	37.32	1987	37.33	37.91	1778	2.59	-10.52
Claire	16	33.88	1647	33.85	34.22	1616	34.28	34.77	1328	2.49	-17.82
Foreman	18	29.34	3973	29.52	29.68	3847	29.74	29.99	3798	2.94	-1.27
Foreman	23	28.32	3812	28.62	28.77	3701	28.81	29.02	3609	2.67	-2.49

Table 1 Rate-distortion performance of various coding schemes. (Units: dB for average PSNR per frame and bits for average bits per frame.)

List of captions to illustrations

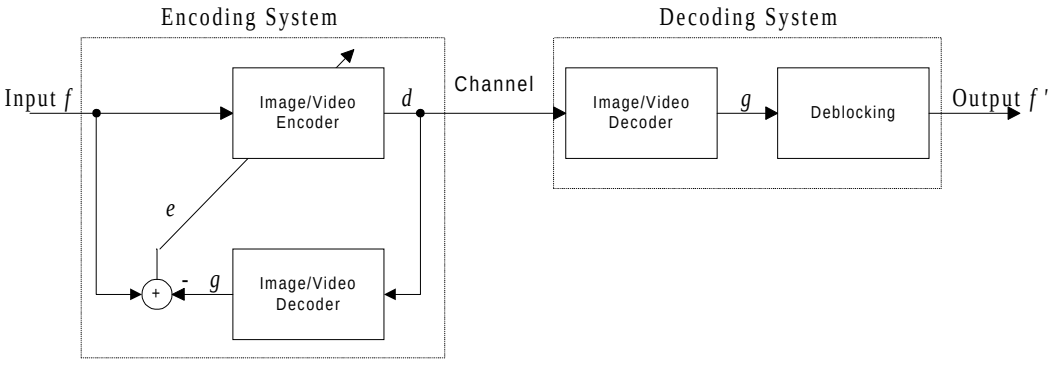


Figure 1. Basic structure of conventional image/video coding systems

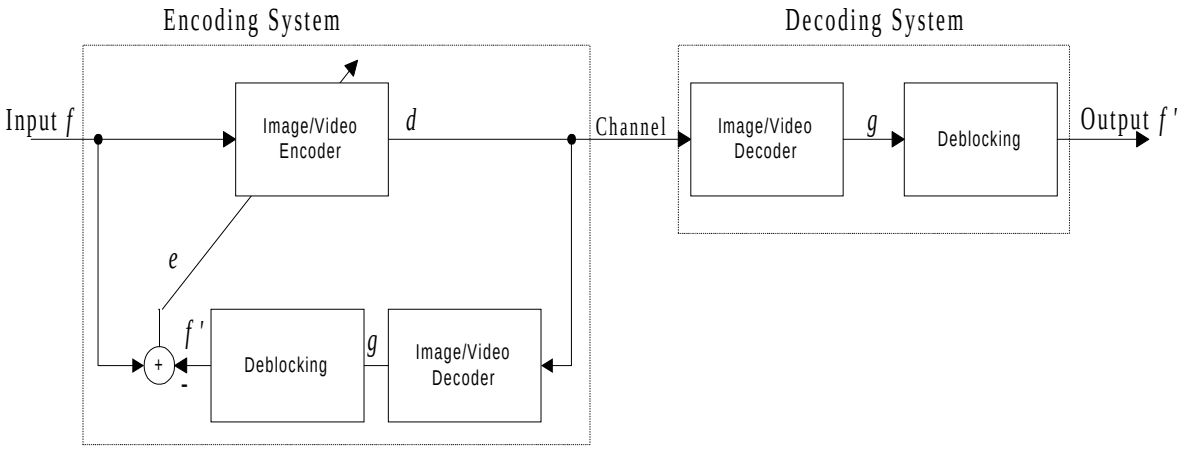


Figure 2. Basic structure of the suggested image/video coding system.

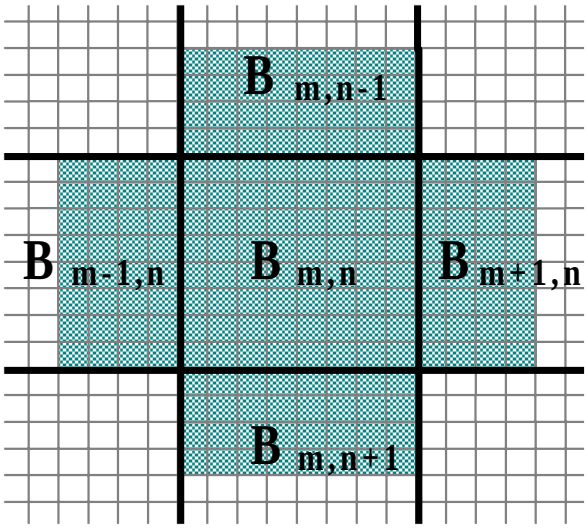


Figure 3. Greater area of block $\mathbf{B_{m,n}}$

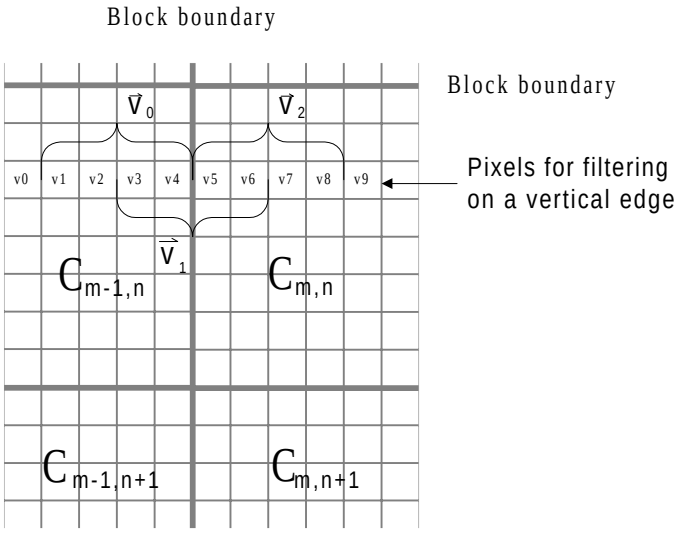


Figure 4. Block boundaries of 8x8 blocks

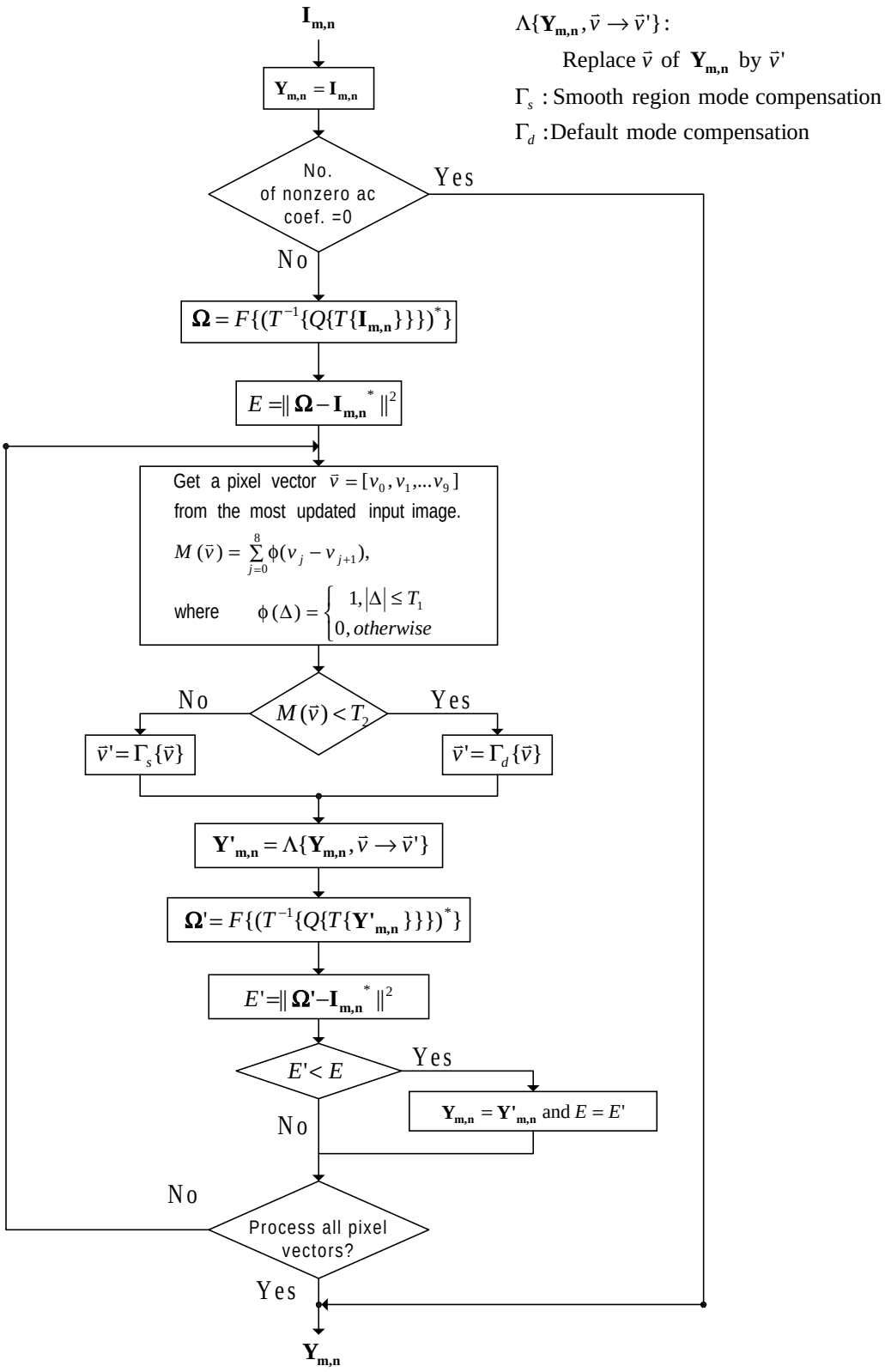
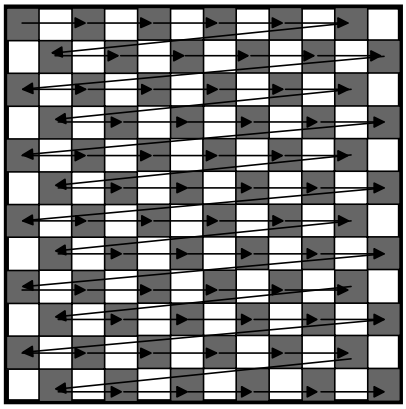


Figure 5. Structure of the proposed pre-processing scheme



(Arrow shows the scanning path)

Figure 6. Checker scanning strategy

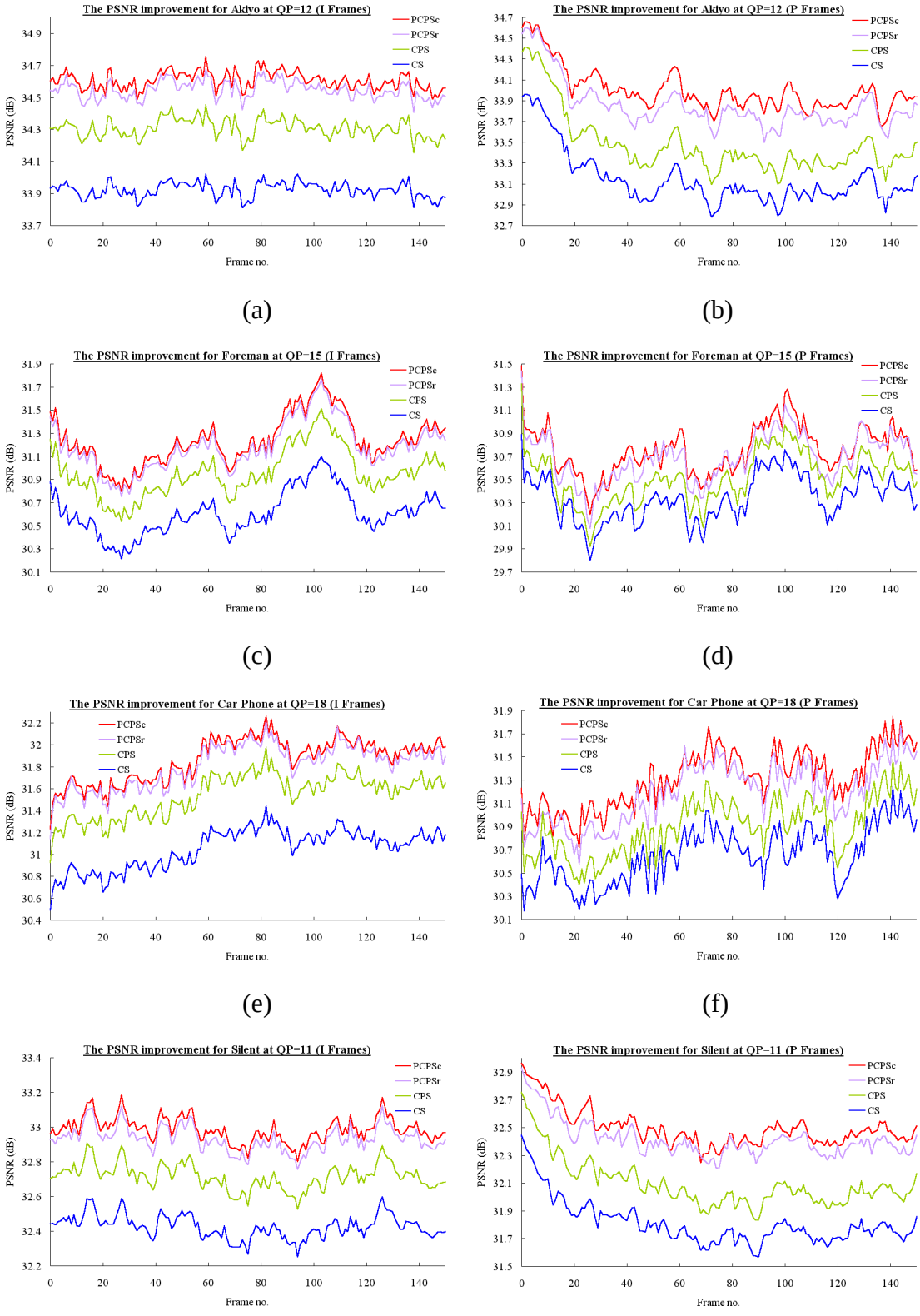
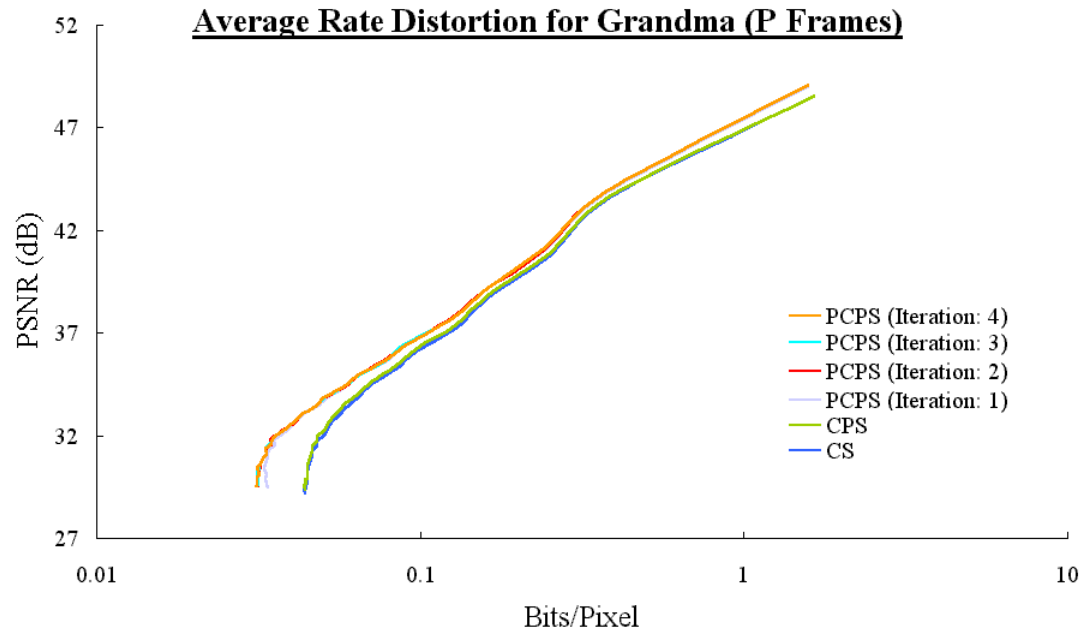
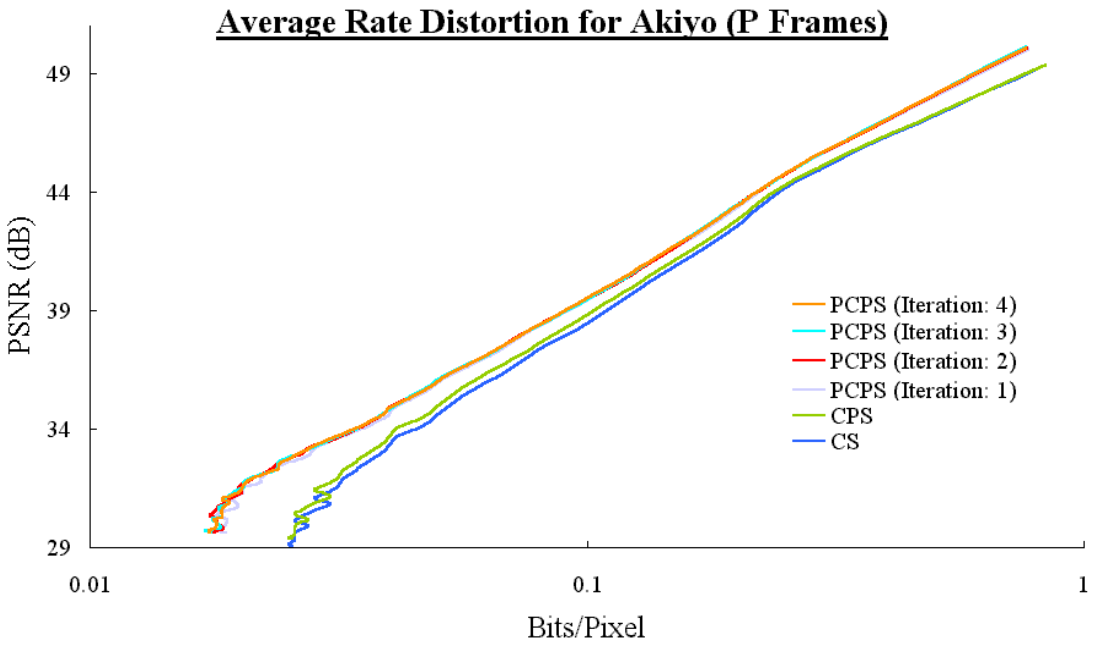


Figure 7. PSNR performance of various coding schemes: a) and b) for *Akiyo* sequence at QP=12, c) and d) for *Foreman* sequence at QP=15, e) and f) for *Car Phone* sequence at QP=18, *and* f) and g) for *Silent* sequence at QP=11.

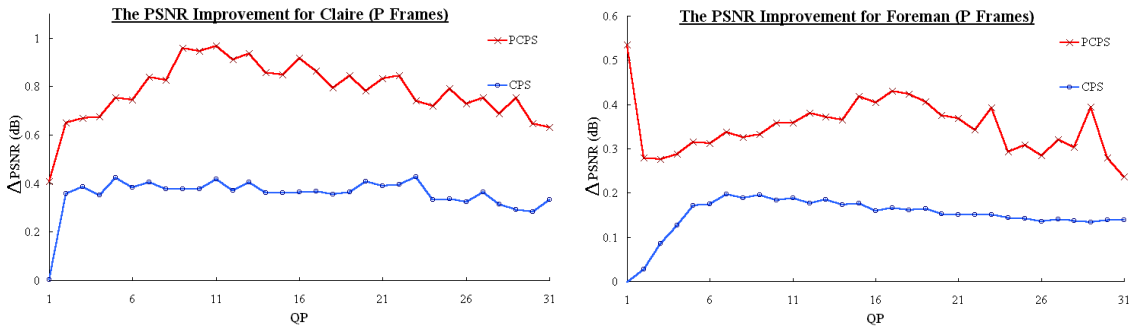


(a)



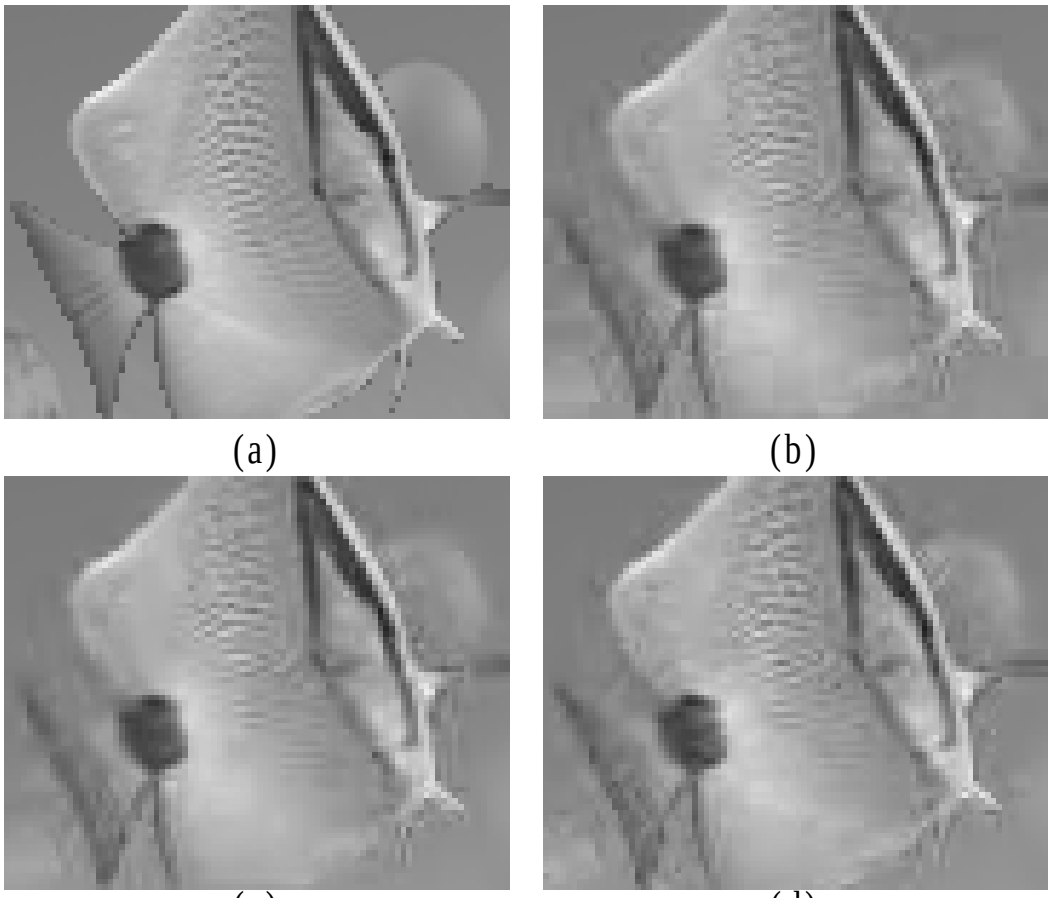
(b)

Figure 8. Average rate-distortion for *Grandma* and *Akiyo*



(a) (b)

Figure 9. PSNR improvements of PCPS and CPS with respect to CS at various QPs: (a) *Claire* (slow motion) and (b) *Foreman* (fast motion) sequences



(a) (b)

(c) (d)

Figure 10. Coding results for *Bream* sequence (QP=13): (a) the 13th original frame; (b) CS result; (c) CPS result and (d) $PCPS_c$ result.