

# Deblocking of Block-Transform Compressed Images Using Weighted Sums of Symmetrically Aligned Pixels

A. Averbuch A. Schclar\* D. L. Donoho†

November 8, 2001

## Abstract

A new class of related algorithms for deblocking block-transform compressed images and video sequences is proposed in this paper. The algorithms apply weighted sums on pixel quartets, which are symmetrically aligned with respect to block boundaries. The basic weights, which are aimed at very low bitrate images, are obtained from a 2-D function which obeys predefined constraints. Using these weights on images compressed at higher bitrates produces a deblocked image which contains blurred “false” edges near real edges. We refer to this phenomenon as the *ghosting effect*. In order to prevent its occurrences, the weights of pixels, which belong to non-monotone areas, are modified by dividing each pixel’s weight by a predefined factor called a *grade*. This scheme is referred to as weight adaptation by grading (WABG). Better deblocking of monotone areas is achieved by applying three iterations of the WABG scheme on such areas followed by a fourth iteration which is applied on the rest of the image. We refer to this scheme as deblocking frames of variable size (DFOVS). DFOVS automatically adapts itself to the activity of each block. This new class of algorithms produces very good subjective results and PSNR results which are competitive relative to available state-of-the-art methods.

*keywords*—deblocking, symmetrically aligned pixels, block-transform, postprocessing

## 1 Introduction

Block-transform codecs (BTC) are among the most common compression tools available for still images and video sequences. These codecs divide the image to non-overlapping square blocks and apply a transform on each individual block as part of the encoding process. Among the available transforms, the DCT is the most widely adopted as it exhibits very good energy compaction and decorrelation properties. This transform is adopted by the JPEG standard for still images compression [1] and the MPEG standards [2, 3, 4, 5] for compression of video sequences.

At low bitrates, BTC compressed images, exhibit a visually annoying phenomenon, known as *the blocking artifact*. This phenomenon is characterized by visually noticeable changes in pixel values along block boundaries. Blocking artifacts become more noticeable in video compression, due to the fact that

---

\*A. Averbuch and A. Schclar are with School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel.

†D. L. Donoho is with the Department of Statistics, Stanford University, Stanford, CA 94305, USA

moving images amplify the blocking effect. Thus, if we wish to achieve high compression rates (low bitrates), using BTC (e.g. JPEG), with visually acceptable results, we have to eliminate the blocking artifacts. This procedure is usually referred to as *deblocking*.

Among the common approaches, post-processing appears to be the most practical solution. It does not require changes to existing standards, and with the rapid increase of available computing power more sophisticated methods can be implemented. The blocking-effect is a major obstacle for using BTC to achieve very low bitrate compression.

Various post-processing techniques have been suggested for the reduction of blocking artifacts, but they often introduce excessive blurring, ringing and in many cases they produce poor deblocking results at certain areas of the image. Moreover, they fail to handle a wide range of bitrates. Most of them cannot automatically adapt themselves to different block activities.

The MPEG4 standard [5] offers a deblocking algorithm which operates in two modes: DC offset mode for low activity blocks and default mode. Block activity is determined according to the amount of changes in the pixels near the block boundaries. All modes apply a 1-D filter in a separable way. The default mode filter uses the DCT coefficients of the pixels being processed and the DC offset mode uses a Gaussian filter. However, this is not exactly a “pure” postprocessing method since every quantization factor from each macro block has to be fed into the algorithm.

The use of a rational filter is suggested in [6, 7, 8] for deblocking. The weights of this filter are determined using a ratio of two polynomials and they are adapted according to local characteristics. The nominator has a low-pass behavior and the denominator is a function of the differences between pairs of pixels used in the mask. Detailed properties of this filter class were described in [9]. Ramponi and Carrato used in [8] a rational filter in order to interpolate an image using only its DC components. A  $3 \times 3$  rational filter, which is applied on block boundaries, was proposed by Castagno et al. in [7] for post-processing of a blocky image. This filter incorporates variance-based measurements for detection of details in the  $3 \times 3$  neighborhood of the filtered pixel.

In [10], special wavelet filters were developed that were utilized to interpolate an image using only the DC component in each block.

Wu *et al.* described in [11] an algorithm which is applied in the transform domain. The algorithm filters each DCT coefficient using the DCT coefficients of its *shifted* blocks. A shifted block is an  $8 \times 8$  block with 0,1 or 2 displacement in each direction.

The theory of *Projection onto Convex Sets* (POCS) is the basis for the iterative methods which are described in [12] and [13]. Each iteration was comprised of two steps: low pass filtering and projection of the low passed DCT coefficients onto their quantization intervals. In [12] low pass filtering was obtained by applying a  $3 \times 3$  low pass filter on the image. In [13] the low pass step bounded the total intensity variations between block boundary pixels of adjacent blocks.

Many of the algorithms described above process only the pixels near the blocks boundaries [5, 7, 13]. This prevents the algorithms from achieving good results at extremely low bit-rates (below 0.25bpp). The algorithm in [12], all of the image pixels are modified, however, it blurred pixels which did not suffer from blocking. Moreover, the execution time of the algorithm in [12] rendered itself as inappropriate for real time applications. Almost all of the algorithms produced poor results when applied on blocks

in which all the AC components were quantized to zero. We call an image which is solely comprised of such blocks, a *DC image*. Exceptions are the algorithms described in [8] and [10] which were specifically tailored to this case. However, in general, they can not handle blocky images other than DC images. Additional information on deblocking can be read in the following survey papers: [14] and [15].

A new class of algorithms for deblocking BTC compressed images and video sequences is proposed in this paper. All the algorithms deblock an image by employing weighted sums of pixel quartets which are symmetrically aligned with respect to block boundaries.

First, we describe a basic deblocking approach which is aimed at very low bit-rate images. In particular, images where all the blocks are *uniform*. In a uniform block, all the pixels are equal. The weights that are used by the algorithm are obtained from a 2-D function which obeys certain constraints. Applying the basic algorithm on images compressed at higher bitrates produces deblocked images which contains an artifact we refer to as the *ghosting effect*. The ghosting effect is characterized by blurred “false” edges which are adjacent to real edges. We alter the weights of the pixels, which belong to non-uniform blocks, in order to prevent the occurrences of the ghosting effect. This is accomplished by dividing each pixel’s weight by a predefined factor we refer to as a *grade*. We refer to this approach as *Weight Adaptation By Grading* (WABG). The result of the enhanced algorithm still exhibits very minor blocking artifacts in *monotone areas*. Monotone areas are regions in the image which are comprised of uniform blocks.

In order to accommodate better deblocking of monotone areas, we form two lists. The first list includes all the uniform blocks and the second list is comprised of all the non-uniform blocks. We refer to non-uniform blocks as *detailed blocks*. Deblocking of the uniform blocks is accomplished by applying three iterations of the WABG scheme. Each of the three iterations processes only the uniform blocks. A fourth iteration is used in order to deblock the detailed blocks. We refer to this approach as *Deblocking Frames of Variable Size* (DFOVS).

The proposed class of algorithms visually outperforms the current state-of-the-art deblocking algorithms such as [5, 7, 11, 12, 13]. In our algorithms, all the pixels are visited. The WABG and the DFOVS approaches automatically adapt themselves to different bitrates, including the case of DC images, and they are considered to be a refinement of the basic scheme. No manual intervention is required. Hence, our algorithms produce very good results for decompressed images ranging from extremely low to medium bitrates.

The rest of this paper is organized as follows. Section II presents the proposed methods in detail. In section III, we provide the experimental results. Finally, conclusions are given in section IV.

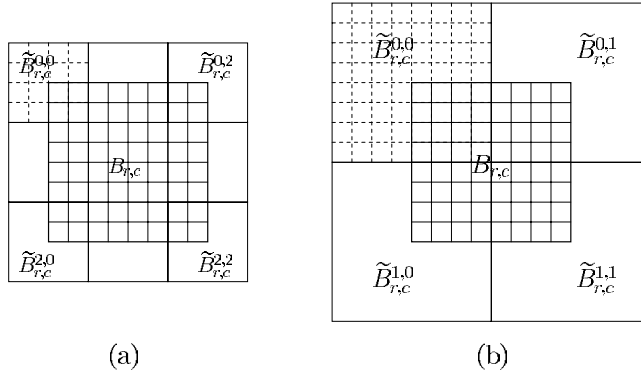


Figure 1: Examples of  $\mathbf{B}(S_f)_{r,c}$  in 2 and 3. (a)  $S_f = 8$ . (b)  $S_f = 4$ .

## 2 Deblocking by weighted sums of symmetrically aligned pixels (WS-SAP)

### 2.1 The kernel of the algorithm

Let  $I$  be an image of size  $R \times C$ ,  $I = \{p_{i,j}\}_{i=0,\dots,R-1, j=0,\dots,C-1}$ . The image is divided into  $8 \times 8$  blocks  $I = \{B_{r,c}\}_{r=0,\dots,\frac{R}{8}-1, c=0,\dots,\frac{C}{8}-1}$  where the pixels in each block  $B_{r,c}$  are given by

$$B_{r,c} = \{p_{8r+i,8c+j}\}_{i,j=0,\dots,7}. \quad (1)$$

We define a *deblocking frame* as a square of adjacent pixels on which the algorithm will be applied. The size of the frame is  $S_f \times S_f$ .

#### 2.1.1 Description of the approach

A new block  $B'_{r,c} = \{p'_{8r+i,8c+j}\}_{i,j=0,\dots,7}$   $r = 0, \dots, \frac{R}{8} - 1$ ,  $c = 0, \dots, \frac{C}{8} - 1$  is constructed for every block  $B_{r,c}$  that was defined in Eq. 1.

Given a block  $B_{r,c}$ , we define the set of  $S_f \times S_f$  deblocking frames

$$\mathbf{B}(S_f)_{r,c} = \{\tilde{B}_{r,c}^{m,n}\} \quad m, n = 0, \dots, \frac{8}{S_f} \quad (2)$$

where the pixels in each deblocking frame  $\tilde{B}_{r,c}^{m,n}$  are given by

$$\tilde{B}_{r,c}^{m,n} = \{p_{8r+(m-\frac{1}{2})S_f+i,8c+(n-\frac{1}{2})S_f+j}\} \quad i, j = 0, \dots, S_f - 1. \quad (3)$$

Figure 1 illustrates the structure of  $\mathbf{B}(S_f)_{r,c}$  for  $S_f = 8$  and  $S_f = 4$ .

Each new block  $B'_{r,c}$  is calculated by applying a weighted sum on four symmetrically aligned pixels in every deblocking frame  $\tilde{B}_{r,c}^{m,n} \in \mathbf{B}(S_f)_{r,c}$ .

For simplicity, we ignore the  $8r + (m - \frac{1}{2})S_f$  and  $8c + (n - \frac{1}{2})S_f$  terms in Eq. 3 and we use the notation  $S'_f = S_f - 1$  to get

$$\tilde{B}_{r,c}^{m,n} = \{p_{i,j}\}_{i,j=0,\dots,S'_f}. \quad (4)$$

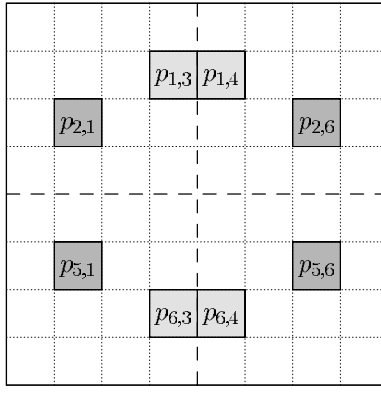


Figure 2: Illustration of the symmetry notion. For example,  $p_{2,1}$  is symmetrically aligned to  $p_{2,6}$  as well as to  $p_{5,1}$ . The dashed lines indicate the central axes of the deblocking frame and also the block boundaries induced by the BTC.

The new value  $p'_{i,j}$  of each pixel  $p_{i,j} \in \tilde{B}_{r,c}^{m,n}$  is determined using a weighted sum of four pixels such that for all  $i, j = 0, \dots, S'_f - 1$

$$p'_{i,j} = \alpha_{i,j} \cdot p_{i,j} + \beta_{S'_f-i,j} \cdot p_{S'_f-i,j} + \gamma_{i,S'_f-j} \cdot p_{i,S'_f-j} + \delta_{S'_f-i,S'_f-j} \cdot p_{S'_f-i,S'_f-j} \quad (5)$$

where  $p_{S'_f-i,j}$  and  $p_{i,S'_f-j}$  are the two image pixels in  $\tilde{B}_{r,c}^{m,n}$ , which lie symmetrically to  $p_{i,j}$  with respect to the horizontal and vertical central axes of the deblocking frame  $\tilde{B}_{r,c}^{m,n}$ , respectively. Pixel  $p_{S'_f-i,S'_f-j}$  lies symmetrically to  $p_{i,j}$  with respect to the center of the deblocking frame. The weights are given by  $\alpha_{i,j}$ ,  $\beta_{S'_f-i,j}$ ,  $\gamma_{i,S'_f-j}$  and  $\delta_{S'_f-i,S'_f-j}$ .

Thus, each deblocking frame  $\tilde{B}_{r,c}^{m,n}$  is divided into *quartets* of symmetrically aligned pixels. Figure 2 illustrates the positions of two quartets. Each quartet has a different shade. The choice of weights is determined to satisfy the following properties:

**Symmetry:**  $\alpha_{i,j} = \alpha_{S'_f-i,j} = \alpha_{i,S'_f-j} = \alpha_{S'_f-i,S'_f-j} \quad i, j = 0, \dots, \frac{S'_f}{2} - 1$ .

Similar constraints are imposed on  $\beta$ ,  $\gamma$  and  $\delta$ , as well. Thus, the computation of every pixel in the quartet uses four distinct permutations of the four weights.

**Monotony:** Given a two variable function  $f(i, j) : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$ , we define its monotony using its gradient  $\nabla f = \left( \frac{\partial f}{\partial i}, \frac{\partial f}{\partial j} \right)$ . We denote the monotony of  $f$  by the signs of its gradient components.

By dividing each deblocking frame  $\tilde{B}_{r,c}^{m,n}$  to  $\frac{S'_f}{2} \times \frac{S'_f}{2}$  quarters, we require

	$\alpha_{i,j}$	$\beta_{S'_f-i,j}$	$\gamma_{i,S'_f-j}$	$\delta_{S'_f-i,S'_f-j}$
$0 \leq i, j \leq \frac{S'_f}{2} - 1$	--	+-	-+	++
$0 \leq i \leq \frac{S'_f}{2} - 1, \frac{S'_f}{2} \leq j \leq S'_f$	-+	++	--	+-
$\frac{S'_f}{2} \leq i \leq S'_f, 0 \leq j \leq \frac{S'_f}{2} - 1$	+-	--	++	-+
$\frac{S'_f}{2} \leq i, j \leq S'_f$	++	-+	+-	--

**Partial ordering:**  $\alpha_{i,j} > \beta_{S'_f-i,j}, \gamma_{i,S'_f-j} > \delta_{S'_f-i,S'_f-j}$ .

**Normalization:** The weights  $\alpha_{i,j}, \beta_{S'_f-i,j}, \gamma_{i,S'_f-j}, \delta_{S'_f-i,S'_f-j}$  are normalized to 1.

In order to determine the weights, we begin by examining the 1-D problem.

### 2.1.2 The 1-D case

We have a row vector  $\mathbf{v} = (p_0, \dots, p_{8 \cdot l-1})$  with groups of 8 elements vectors  $V_c = (p_{8c}, \dots, p_{8c+7})$   $c = 0, \dots, l-1$ . The blocking artifact in a 1-D vector manifests itself as discontinuities between pixels  $p_{8 \cdot k-1}$  and  $p_{8 \cdot k}$   $k = 1, \dots, l-1$ . A deblocking frame is defined by a set of  $S_f$  adjacent pixels.

Given an 8 elements vector  $V_c$ , we define the set of row vectors with  $S_f$  elements to be

$$\mathbf{V}(S_f)_c = \left\{ \tilde{V}_c^m \right\} \quad m = 0, \dots, \frac{8}{S_f}$$

where the elements of  $\tilde{V}_c^m$  are given by

$$\tilde{V}_c^m = \left\{ p_{8c + (m - \frac{1}{2})S_f + i} \right\}_{i=0, \dots, S'_f}.$$

As before, in order to simplify the notation, we ignore the  $8c + (m - \frac{1}{2})S_f$  term and denote  $S'_f = S_f - 1$  to get

$$\tilde{V}_c^m = \{p_i\}_{i=0, \dots, S'_f}.$$

The new value  $p'_i$  of each element  $p_i$  is determined using a weighted sum of two pixels

$$p'_i = \omega_i \cdot p_i + \psi_{S'_f-i} \cdot p_{S'_f-i}.$$

The weights will obey the properties mentioned above:

(i) **Symmetry:**  $\omega_i = \omega_{S'_f-i}$  and  $\psi_{S'_f-i} = \psi_i$  for  $i = 0, \dots, \frac{S_f}{2} - 1$ .

(ii) **Monotony:**  $\omega_i > \omega_{i+1}$  and  $\psi_{S'_f-i} < \psi_{S'_f-(i+1)}$  for  $0 \leq i \leq \frac{S_f}{2} - 2$   
 $\omega_i < \omega_{i+1}$  and  $\psi_{S'_f-i} > \psi_{S'_f-(i+1)}$  for  $\frac{S_f}{2} \leq i \leq S'_f - 1$ .

(iii) **Partial ordering:**  $\omega_i \geq \psi_{S'_f-i}$  for  $i = 0, \dots, S'_f$ .

(iv) **Normalization:**  $\omega_i + \psi_{S'_f-i} = 1$  for  $i = 0, \dots, S'_f$ .

We regard the weights as *continuous* functions  $\omega(x), \psi(x) : [0, S'_f] \rightarrow [0, 1]$ . We will solve for  $\omega$ .  $\psi$  is immediately derived from property (iv). The solution for  $\omega$  is not unique. Thus, we need to supply some additional data in order to find  $\omega$ . For convenience, we denote  $k = \frac{S_f}{2}$  and get  $S'_f = 2k - 1$  and add the constraint  $\omega(0) = \eta$  and  $\omega(k-1) = \vartheta$ . The values of  $\eta, \vartheta$ , which produce the best results, are found empirically. The chosen values of  $\eta$  and  $\vartheta$  must also follow the properties mentioned above. Consequently, properties (iii) and (iv) impose  $\eta, \vartheta \geq 0.5$  and from property (i) we get  $\omega(2k-1) = \eta$

and  $\omega(k) = \vartheta$ . There is no unique solution that satisfies the above conditions. We will consider two possible solutions: linear and quadratic.

**The linear solution** Let the linear solution  $\omega_L$  be of the form  $\omega_L(x) = ax + b$ . First, a separate solution is given for each of the intervals  $[0, k - 1]$  and  $[k, 2k - 1]$  denoted by  $\omega_L^{[0, k-1]}$  and  $\omega_L^{[k, 2k-1]}$ , respectively. We get:

$$\begin{aligned}\omega_L^{[0, k-1]}(x) &= \frac{\vartheta - \eta}{k-1}x + \eta && \text{if } x \in [0, k-1] \\ \omega_L^{[k, 2k-1]}(x) &= \frac{\eta - \vartheta}{k-1}x + \frac{(2\vartheta - \eta)k - \vartheta}{k-1} && \text{if } x \in [k, 2k-1].\end{aligned}$$

Second, observing that  $\omega_L^{[0, k-1]}(x) + \omega_L^{[k, 2k-1]}(x) = \frac{(2k-1)\vartheta - \eta}{k-1}$ , a more compact formula can be derived for  $\omega_L$ . Specifically,

$$\omega_L^{[0, 2k-1]}(x) = \max \left\{ \omega_L^{[0, k-1]}(x), \frac{(2k-1)\vartheta - \eta}{k-1} - \omega_L^{[0, k-1]}(x) \right\} \quad x \in [0, 2k-1].$$

**The quadratic solution** Let the quadratic solution  $\omega_Q$  be of the form  $\omega_Q(x) = ax^2 + bx + c$ . Since  $\omega_Q$  is symmetric, there is no need to divide the interval  $[0, 2k - 1]$ . The solution for  $a$ ,  $b$  and  $c$  is obtained by solving the linear system  $\mathbf{M}\mathbf{x} = \mathbf{y}$  where

$$\mathbf{M} = \begin{pmatrix} 0 & 0 & 1 \\ k^2 & k & 1 \\ (2k-1)^2 & 2k-1 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} \eta \\ \vartheta \\ \eta \end{pmatrix} \quad k > 1.$$

Since  $\mathbf{M}$  is invertible, the system has a unique solution given by  $\mathbf{M}^{-1}\mathbf{y}$  which is

$$\omega_Q(x) = \frac{(\vartheta - \eta)}{k(1-k)}x^2 + \frac{(\vartheta - \eta)(1 - 2k)}{k(1-k)}x + \eta. \quad (6)$$

Figure 3 shows a graph of a linear and quadratic 1-D  $\omega(i)$ ,  $\psi(i)$  with  $\eta = 1$ ,  $\vartheta = \frac{4}{7}$  and  $\eta = 1$ ,  $\vartheta = \frac{1}{2}$ , respectively.

One dimensional deblocking is illustrated in Fig. 4. Figure 4(a) is a step function. Figures 4(b) and 4(c) depict the 1-D deblocking of 4(a) using linear and quadratic weights, respectively.

### 2.1.3 The 2-D case

In order to obtain a deblocking solution for the 2-D case, we observe that the deblocking frame column of pixels  $p_{i,j}$  and  $p_{S'_f-i,j}$  can be treated as 1-D vector of eight elements (Fig. 2). A similar argument holds for the pairs  $p_{i,j}$ ,  $p_{i,S'_f-j}$  and  $p_{S'_f-i,j}$ ,  $p_{S'_f-i,S'_f-j}$ . Since, all of these arguments hold at the same time, a separable solution is given for the 2-D case. Specifically, the weights  $\alpha_{i,j}$ ,  $\beta_{S'_f-i,j}$ ,  $\gamma_{i,S'_f-j}$ ,  $\delta_{S'_f-i,S'_f-j}$  are given by a product of two 1-D solutions:

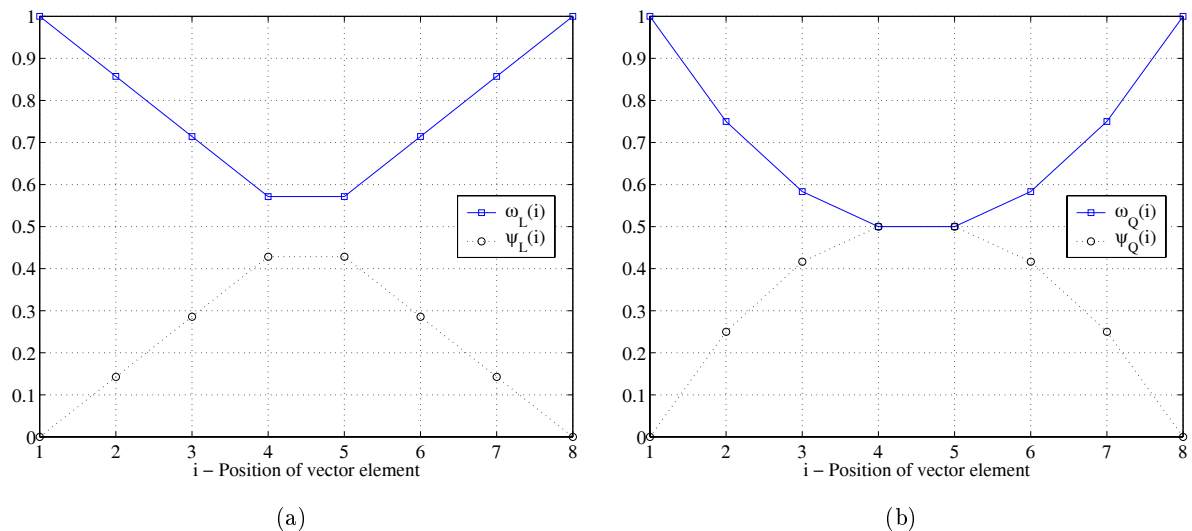


Figure 3: 1-D linear and quadratic  $\omega(i)$  and  $\psi(i)$ . (a) Linear with  $\eta = 1$  and  $\vartheta = \frac{4}{7}$ . (b) Quadratic  $\eta = 1$  and  $\vartheta = \frac{1}{2}$ .

$$\begin{aligned}
\alpha_{i,j} &= \omega(i) \cdot \omega(j) \\
\beta_{S'_f-i,j} &= \psi(S'_f - i) \cdot \omega(j) \\
\gamma_{i,S'_f-j} &= \omega(i) \cdot \psi(S'_f - j) \\
\delta_{S'_f-i,S'_f-j} &= \psi(S'_f - i) \cdot \psi(S'_f - j)
\end{aligned}$$

where  $\omega(\cdot)$  and  $\psi(\cdot)$  are both either linear or quadratic.

Graphs of the 2-D *linear* weights  $\alpha_{i,j}^L, \beta_{i,j}^L, \gamma_{i,j}^L, \delta_{i,j}^L$  using  $\eta = 1$  and  $\vartheta = \frac{4}{7}$  are depicted in Fig. 5. Illustrations of the 2-D *quadratic* weights  $\alpha_{i,j}^Q, \beta_{i,j}^Q, \gamma_{i,j}^Q, \delta_{i,j}^Q$  using  $\eta = 1$  and  $\vartheta = \frac{1}{2}$  are shown in Fig. 6. These graphs demonstrate that the calculated weights behave as expected: they satisfy the four constraints e.g. monotony and symmetry.

#### 2.1.4 Implementation issues

**Handling of image boundaries** The set of deblocking frames  $\mathbf{B}(S_f)_{r,c}$  partially reaches beyond the bounds for blocks at the boundary of the image. Therefore, we symmetrically extend the boundary of the *original image* by  $S_f$  pixels. The algorithm is applied on the *extended image*. The resulting image has an additional band of  $\frac{S_f}{2}$  pixels at its boundary. The subimage without these pixels is taken as the output. This is equivalent to applying 1-D deblocking on each row (column) in the  $\frac{S_f}{2}$  band of pixels at the image boundary. A one dimensional vector is deblocked by applying  $\omega$  and  $\psi$  as explained in section 2.1.2.

**The ghosting effect** Figure 7 illustrates the application of the kernel algorithm (WSSAP) on the *Peppers* image compressed at different bitrates. Figure 7(a) is the *Peppers* image compressed at 0.21bpp and Fig. 7(c) is the DC image of *Peppers*. Figures 7(b) and 7(d) are the result of the basic algorithm applied on figures 7(a) and 7(c), respectively. The basic algorithm uses linear weights with

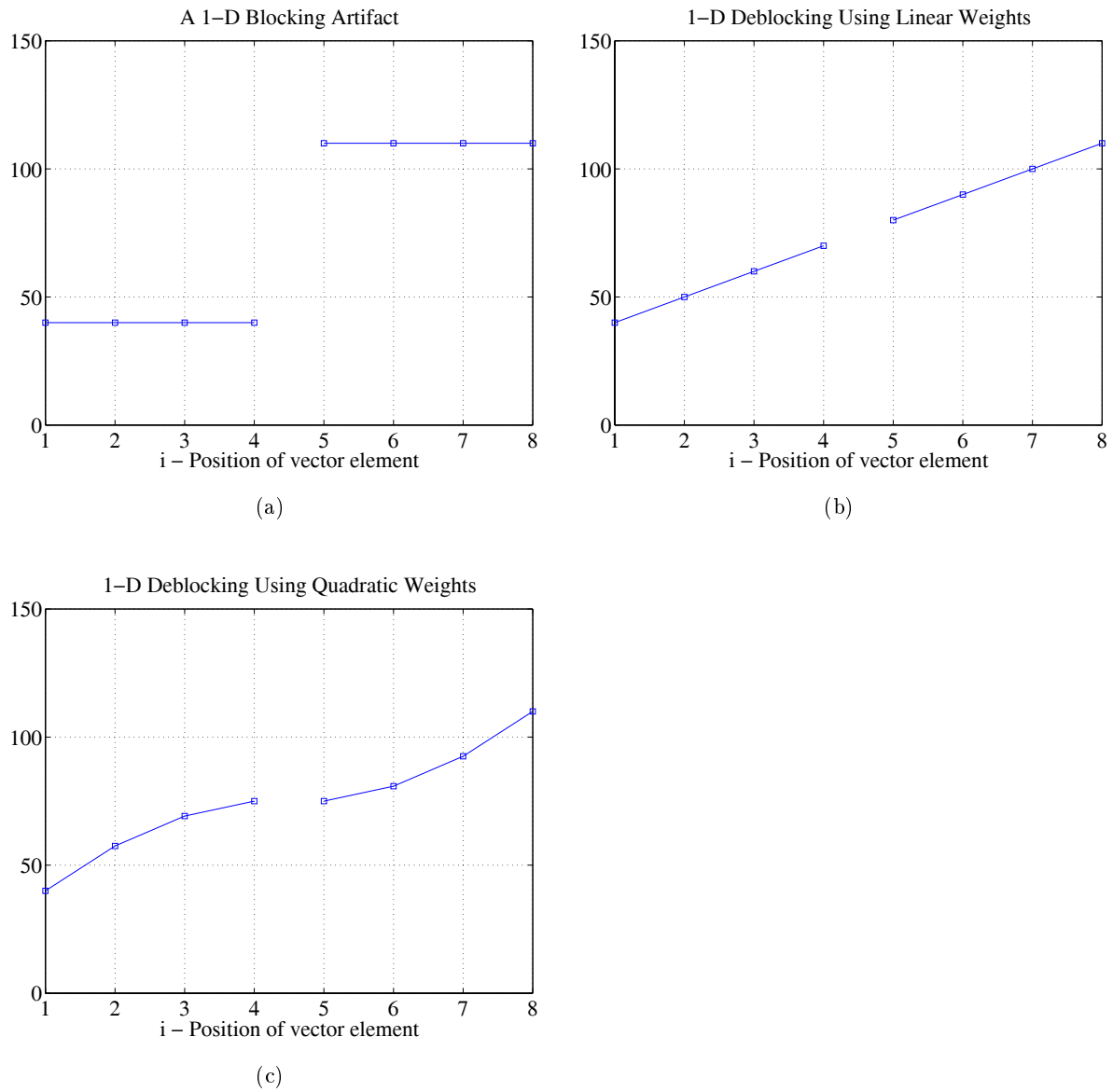
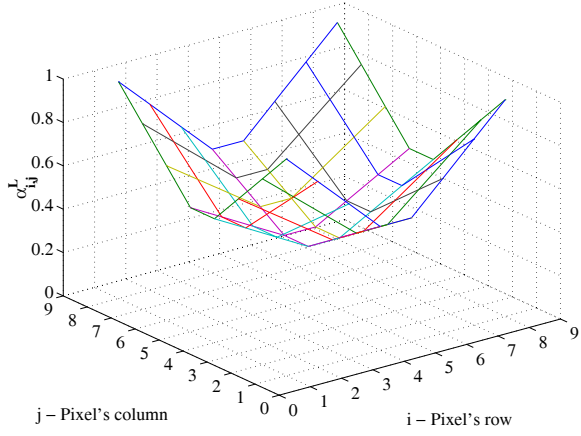
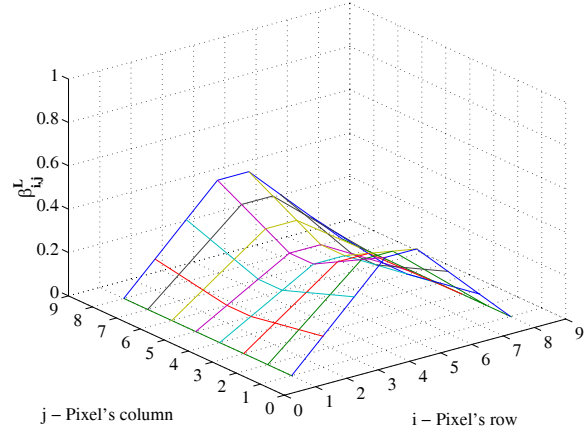


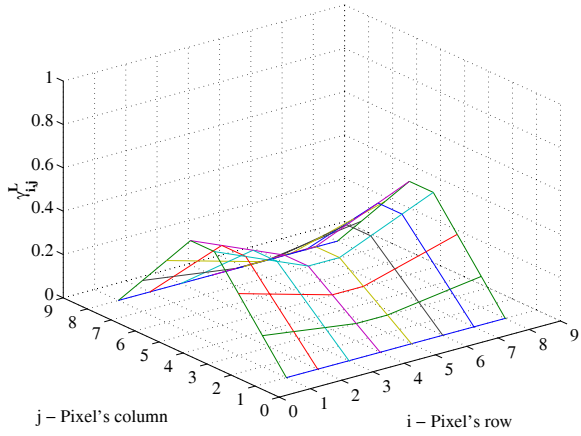
Figure 4: 1-D blocking artifact and its deblocking. (a) A 1-D blocking artifact. (b) Deblocking with linear weights using  $\eta = 1$  and  $\vartheta = \frac{4}{7}$ . (c) Deblocking with quadratic weights using  $\eta = 1$  and  $\vartheta = \frac{1}{2}$ .



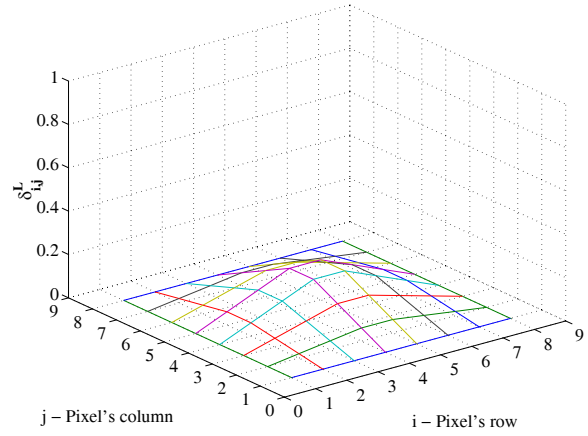
(a)



(b)

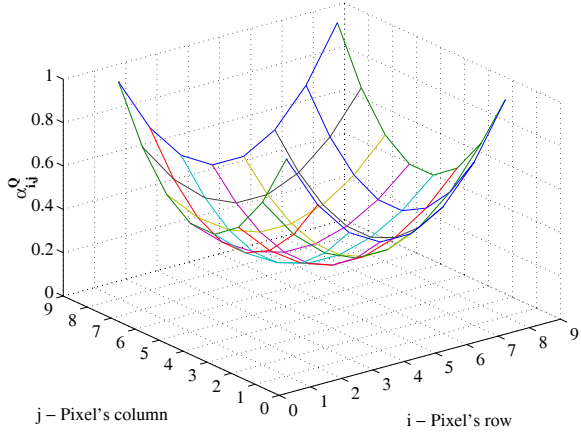


(c)

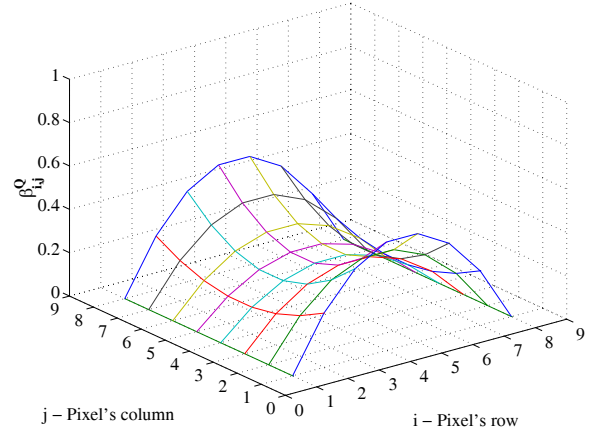


(d)

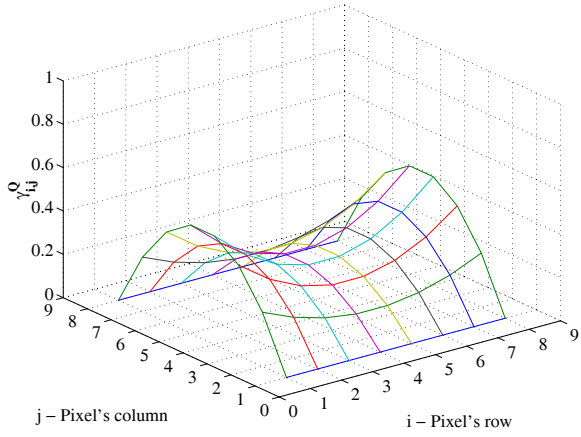
Figure 5: 2-D *linear* weights  $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \delta_{i,j}$  using  $\eta = 1$  and  $\vartheta = \frac{4}{7}$ . (a)  $\alpha_{i,j}^L$ . (b)  $\beta_{i,j}^L$ . (c)  $\gamma_{i,j}^L$ . (d)  $\delta_{i,j}^L$ .



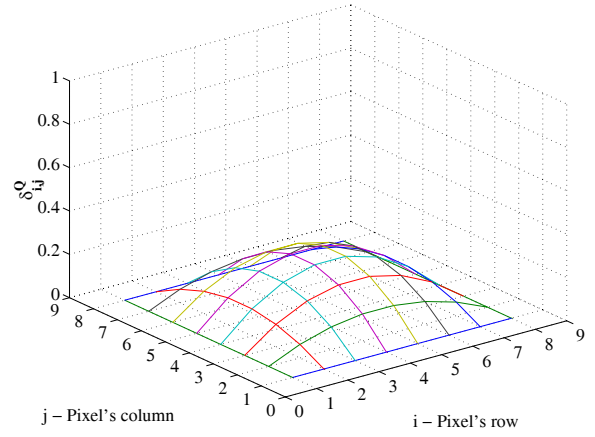
(a)



(b)



(c)



(d)

Figure 6: 2-D *quadratic* weights  $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \delta_{i,j}$  using  $\eta = 1$  and  $\vartheta = \frac{1}{2}$ . (a)  $\alpha_{i,j}^Q$ . (b)  $\beta_{i,j}^Q$ . (c)  $\gamma_{i,j}^Q$ . (d)  $\delta_{i,j}^Q$ .

$\eta = 1$ ,  $\vartheta = 0.63$  and deblocking frames of size  $S_f = 8$ .

The basic approach works well at very low bit rates (under  $0.25\text{bpp}$ ). At higher bit rates this approach produces deblocking results of high quality but introduces a visually noticeable side effect. We refer to this side effect as the *ghosting effect*. It is characterized by blurred “false” edges which are symmetrically located in parallel to real edges, as illustrated in Figure 8. Figures 8(a) and 8(b) are the magnified versions of the marked rectangle in figures 7(a) and 7(b), respectively. A “false” edge is easily noticed in the middle of Fig. 7(b).

This phenomenon is caused by unbalanced weight contribution, which occurs when the differences among the pixels participating in the calculation are high. As a consequence, the values of the pixels in the “false” edge are significantly higher from their neighboring pixels and so they appear as edge pixels. The ghosting effect usually occurs at low to medium bit rates (above  $0.25\text{bpp}$ ).

## 2.2 Prevention of the ghosting effect by weight adaptation

In order to prevent the occurrence of the ghosting effect, we propose a scheme, which reduces the weights  $\beta_{S'_f-i,j}$ ,  $\gamma_{i,S'_f-j}$ ,  $\delta_{S'_f-i,S'_f-j}$  of pixels that are located in non-uniform blocks. This reduces their weighted influence on the calculated pixel. We refer to non-uniform blocks as *detailed blocks*. Three details preservation modes of operation are available. One of the modes, which is intended to preserve the largest amount of details, increases  $\alpha_{i,j}$  while reducing  $\beta_{S'_f-i,j}$ ,  $\gamma_{i,S'_f-j}$ ,  $\delta_{S'_f-i,S'_f-j}$ .

In order to reduce the weights as described above, we assign a grade to each pixel. The grade provides an indication to the amount of details in the pixel’s neighborhood. All the grades form a matrix we refer to as a *grading matrix*. Without loss of generality, we choose grades which increase as the amount of details in the pixel’s neighborhood increases. The grades are confined to be in the range of 1–16. The weight reduction is achieved by dividing each pixel’s weight by its corresponding grade. Weight increase is obtained by multiplying the pixel’s weight by its grade. After the division, the modified weights are normalized and used in the calculation of the pixel’s new value in the same way they were employed by the basic approach. It should be mentioned that the symmetry property no longer holds for the modified weights.

### 2.2.1 Weights adaptation by grading (WABG)

Let a *grading matrix* be  $G = \{g_{r,c}\}_{r=0,\dots,R-1, c=0,\dots,C-1}$ . Let  $p_{i,j}$ ,  $p_{S'_f-i,j}$ ,  $p_{i,S'_f-j}$  and  $p_{S'_f-i,S'_f-j}$  be a symmetrically aligned pixels quartet and  $\alpha_{i,j}$ ,  $\beta_{S'_f-i,j}$ ,  $\gamma_{i,S'_f-j}$ ,  $\delta_{S'_f-i,S'_f-j}$  their corresponding weights from the basic algorithm. We denote by  $g_{i,j}$ ,  $g_{S'_f-i,j}$ ,  $g_{i,S'_f-j}$  and  $g_{S'_f-i,S'_f-j}$  the grades of pixels  $p_{i,j}$ ,  $p_{S'_f-i,j}$ ,  $p_{i,S'_f-j}$  and  $p_{S'_f-i,S'_f-j}$ , respectively.

We denote the details preservation modes by  $M_{low}$ ,  $M_{medium}$  and  $M_{high}$ . In order to achieve detail preservation, the weight  $\alpha_{i,j}$  is multiplied by a factor  $\lambda_{i,j}$  whose value is determined according to the mode of operation:

$$\lambda_{i,j} = \begin{cases} 1 & \text{if } mode = M_{low} \\ g_{i,j} & \text{if } mode = M_{high}. \end{cases}$$



(a)



(b)



(c)



(d)

Figure 7: Application of the basic approach using linear weights with  $\eta = 1$ ,  $\vartheta = 0.63$ ,  $S_f = 8$  on the *Peppers* image. (a) compressed at 0.21bpp (PSNR = 27.32 dB). (b) after the application of the basic approach (PSNR = 27.49 dB). (c) DC image (PSNR = 20.32 dB). (d) deblocking result of (c) using the basic approach (PSNR = 21.45 dB).

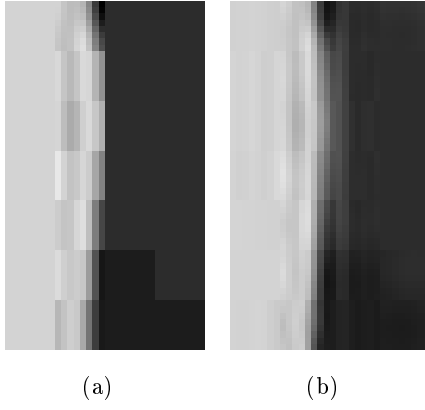


Figure 8: Example of the ghosting effect in the *Peppers* image compressed at 0.21bpp. (a) magnification of the marked area in Fig. 7(a). (b) magnification of the marked rectangle area in Fig. 7(b) - the ghosting effect is manifested as the blurred “false” edge in the middle of the figure.

When the mode is  $M_{medium}$ , the value of  $\lambda_{i,j}$  is determined by:

$$\lambda_{i,j} = \begin{cases} 1 & \text{if } i \text{ or } j \text{ is equal to } \frac{S_f}{2} \text{ or } \frac{S_f}{2} - 1 \\ g_{i,j} & \text{otherwise.} \end{cases}$$

We denote the sum of the adapted weights by

$$W = \lambda_{i,j} \cdot \alpha_{i,j} + \frac{\beta_{S'_f-i,j}}{g_{S'_f-i,j}} + \frac{\gamma_{i,S'_f-j}}{g_{i,S'_f-j}} + \frac{\delta_{S'_f-i,S'_f-j}}{g_{S'_f-i,S'_f-j}}.$$

The new value of the pixel is calculated as follows:

$$p'_{i,j} = \alpha'_{i,j} \cdot p_{i,j} + \beta'_{S'_f-i,j} \cdot p_{S'_f-i,j} + \gamma'_{i,S'_f-j} \cdot p_{i,S'_f-j} + \delta'_{S'_f-i,S'_f-j} \cdot p_{S'_f-i,S'_f-j}$$

where  $\alpha'_{i,j}$ ,  $\beta'_{S'_f-i,j}$ ,  $\gamma'_{i,S'_f-j}$  and  $\delta'_{S'_f-i,S'_f-j}$  are given by  $\frac{\alpha_{i,j}}{W}$ ,  $\frac{\beta_{S'_f-i,j}}{W}$ ,  $\frac{\gamma_{i,S'_f-j}}{W}$  and  $\frac{\delta_{S'_f-i,S'_f-j}}{W}$ , respectively.

### 2.2.2 Grading matrix implementations

We suggest two grading matrix implementations. In both implementations each block is graded according to its details. These grades form a *block grading matrix*. In the first implementation, the grade of each block is based on its variance. In the second implementation the grade is determined using the block’s DCT coefficients. The grade we use for each pixel is determined according to three attributes: (a) its block grade (b) its proximity to the calculated pixel and (c) the difference between its grey level and the grey level of the calculated pixel.

**Block grade** We denote by  $b_{i,j}$ ,  $b_{S'_f-i,j}$ ,  $b_{i,S'_f-j}$  and  $b_{S'_f-i,S'_f-j}$  to be the block grades of the pixels  $p_{i,j}$ ,  $p_{S'_f-i,j}$ ,  $p_{i,S'_f-j}$  and  $p_{S'_f-i,S'_f-j}$ , respectively.

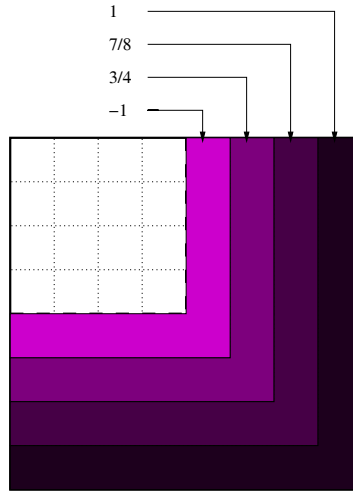


Figure 9: The values of  $\kappa(m, n)$  for the calculation of pixels  $p_{i,j}$  where  $0 \leq i, j \leq \frac{S_f}{2} - 1$ . The values of  $\kappa(m, n)$  for pixels  $p_{i,j}$  where  $0 \leq i \leq \frac{S_f}{2} - 1$  and  $\frac{S_f}{2} \leq j \leq S_f - 1$ , are obtained by rotating the figure  $90^\circ$  clockwise.  $\kappa(m, n)$  values for the pixels in quarters  $\frac{S_f}{2} \leq i \leq S_f - 1$ ,  $0 \leq j \leq \frac{S_f}{2} - 1$  and  $\frac{S_f}{2} \leq i, j \leq S_f - 1$  are obtained by rotating the figure  $180^\circ$  and  $270^\circ$ , respectively.

**Proximity to the calculated pixel** We define a factor  $\kappa(m, n)$  to control the influence of the block grade according to the proximity to the calculated pixel

$$\kappa(m, n) = \begin{cases} -1 & \text{if } n - m = 1 \\ \frac{3}{4} & \text{if } n - m = 3 \\ \frac{7}{8} & \text{if } n - m = 5 \\ 1 & \text{if } n - m = 7. \end{cases}$$

where  $m, n$  are either the row or column coordinates of the pixels. Figure 9 illustrates the values of  $\kappa(m, n)$ .

**Grey level difference** Given two pixels  $p_i$  and  $p_j$  in a quartet we denote the sum of the pixels' blocks grades in the quartet by

$$G_T = b_{i,j} + b_{S'_f-i,j} + b_{i,S'_f-j} + b_{S'_f-i,S'_f-j}$$

and define a grey level difference function

$$\text{Diff}(G_T, p_{i,j}, p_{k,l}) = \begin{cases} 1 & \text{if } G_T = 4 \\ e^{-\frac{|p_{i,j} - p_{k,l}|}{64}} & \text{otherwise.} \end{cases} \quad 0 \leq i, j \leq R - 1, 0 \leq k, l \leq C - 1$$

The corresponding grades  $g_{S'_f-i,j}$ ,  $g_{i,S'_f-j}$ ,  $g_{S'_f-i,S'_f-j}$  of pixels  $p_{S'_f-i,j}$ ,  $p_{i,S'_f-j}$ ,  $p_{S'_f-i,S'_f-j}$ , are given by:

$$\begin{aligned}
g_{S'_f-i,j} &= \max \left\{ \kappa \left( i, S'_f - i \right) \cdot b_{S'_f-i,j}, 1 \right\} \cdot \text{Diff} \left( G_T, p_{i,j}, p_{S'_f-i,j} \right) \\
g_{i,S'_f-j} &= \max \left\{ \kappa \left( j, S'_f - j \right) \cdot b_{i,S'_f-j}, 1 \right\} \cdot \text{Diff} \left( G_T, p_{i,j}, p_{i,S'_f-j} \right) \\
g_{S'_f-i,S'_f-j} &= \max \left\{ \kappa \left( i, S'_f - i \right) \cdot b_{S'_f-i,S'_f-j}, 1 \right\} \cdot \text{Diff} \left( G_T, p_{i,j}, p_{S'_f-i,S'_f-j} \right).
\end{aligned}$$

The *max* operation is required in order to ensure that the weight will always be greater or equal to 1, otherwise the partial ordering property (see section 2.1.1) of the weights might not hold.

We suggest two ways to implement block grading: variance based and DCT based.

### 2.2.3 The variance grading implementation

Variance-based measurements produce a reliable and simple estimation of details level in a pixel environment. Thus, we can base the grade of the block on its variance. The variance of a *uniform* block is zero. The value of the variance increases with the increase in the amount of details in the block. The variance of an  $8 \times 8$  block ranges from zero to  $\approx 10^5$ , and the range of pixel values is 0–255. Therefore, in order to use the variance based grades, they are set on a squared logarithmic scale in the range of 1–16. The grades are given to  $8 \times 8$  blocks of the original image. Formally, let an  $8 \times 8$  block be

$$B_{r,c} = \{p_{8r+i,8c+j}\}_{i,j=0,\dots,7} \quad r = 0, \dots, \frac{R}{8} - 1, \quad c = 0, \dots, \frac{C}{8} - 1.$$

We denote its mean by  $\mu_{r,c} = \frac{1}{64} \sum_{i=0}^7 \sum_{j=0}^7 p_{8r+i,8c+j}$  and its variance by  $\sigma_{r,c}^2 = \frac{1}{64} \sum_{i=0}^7 \sum_{j=0}^7 (p_{8r+i,8c+j} - \mu_{r,c})^2$ . Let  $b'_{r,c}$  be  $b'_{r,c} = \max \left\{ 1, \log_{10}^2 \left( \sigma_{r,c}^2 + 1 \right) \right\}$   $r = 0, \dots, \frac{R}{8} - 1, c = 0, \dots, \frac{C}{8} - 1$  and denote the blocks' maximal and minimal grades by  $b^{\max}$  and  $b^{\min}$ , respectively, where

$$\begin{aligned}
b^{\max} &= \max_{r=0,\dots,\frac{R}{8}-1, c=0,\dots,\frac{C}{8}-1} \{b_{r,c}\} \\
b^{\min} &= \min_{r=0,\dots,\frac{R}{8}-1, c=0,\dots,\frac{C}{8}-1} \{b_{r,c}\}.
\end{aligned}$$

The grade of the block  $B_{r,c}$  is

$$b_{r,c} = 15 \cdot \frac{b'_{r,c} - b^{\min}}{b^{\max} - b^{\min}} + 1.$$

Clearly, one can see that this choice is in agreement with the requirements of the grading system, which was described in section 2.2.

### 2.2.4 The DCT grading implementation

Another proposed implementation for a block grading matrix uses the DCT coefficients of the block since they give an indication to the amount of details in a block. Specifically, in a uniform block, all the AC components are equal to zero. As the amount of details increases in a block, more non-zero AC coefficients are introduced. Thus, the number of non-zero DCT coefficients can give us a fair indication of the amount of details in a block. Therefore, we determine the grade of a block to be the number of non-zero DCT coefficients translated to the range 1–16. Specifically, let

$$B_{r,c}^d = \left\{ p_{8r+i,8c+j}^d \right\}_{i,j=0,\dots,7} \quad r = 0, \dots, \frac{R}{8} - 1, \quad c = 0, \dots, \frac{C}{8} - 1$$

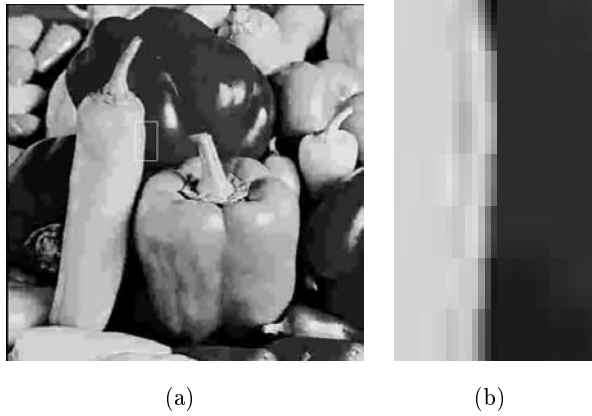


Figure 10: Application of the WABG scheme on *Peppers* compressed at 0.21bpp. The algorithm uses linear weights with  $\eta = 1$ ,  $\vartheta = 0.63$  and the variance based block grading implementation. (a) The image after the application of the WABG scheme (PSNR = 27.96 dB). (b) Magnification of the rectangular marked area in (a) which corresponds to the rectangular marked area in Fig. 8(b) - no ghosting effect is visible.

be the DCT coefficients of the block  $B_{r,c} = \{p_{8r+i,8c+j}\}_{i,j=0,\dots,7}$ . Each block's grade  $b_{r,c}$  is defined as the translation of

$$b'_{r,c} = \left| \{p_{8r+i,8c+j}^d \neq 0 \quad : \quad i, j = 0, \dots, 7\} \right| \quad r = 0, \dots, \frac{R}{8} - 1, \quad c = 0, \dots, \frac{C}{8} - 1$$

to the interval  $[1, 16]$  where the outer vertical lines stand for the *magnitude* of the set.

Figure 10 demonstrates the application of the WABG scheme on *Peppers* compressed at 0.21bpp. The algorithm uses linear weights with  $\eta = 1$ ,  $\vartheta = 0.63$  and the variance based block grading implementation. The detail preservation mode is set to  $M_{low}$ . Figure 10(a) is the deblocked image and Fig. 10(b) is the magnification of the rectangular marked area in Fig. 10(a). Figure 10(b) reveals that there is no ghosting effect.

The WABG scheme does not introduce “false” edges, but the blocking artifacts are still noticeable in monotone areas. This observation is corroborated by looking at the long pepper in Fig. 10(a). Therefore, we refine the results obtained by the WABG scheme in order to accommodate better deblocking of monotone areas.

### 2.3 Deblocking frames of variable size (DFOVS)

In order to achieve better deblocking we can suggest an iterative algorithm which uses deblocking frames of variable sizes. Specifically, this scheme consists of three stages:

1. A classification procedure is executed to form a list of all the uniform blocks. Uniform blocks are blocks whose grade is equal to one. We denote this list by UBL. The list of the non-uniform (detailed) blocks is denoted by DBL. Figure 11 illustrates the UBL and DBL for *Barbara* compressed at 0.24bpp.

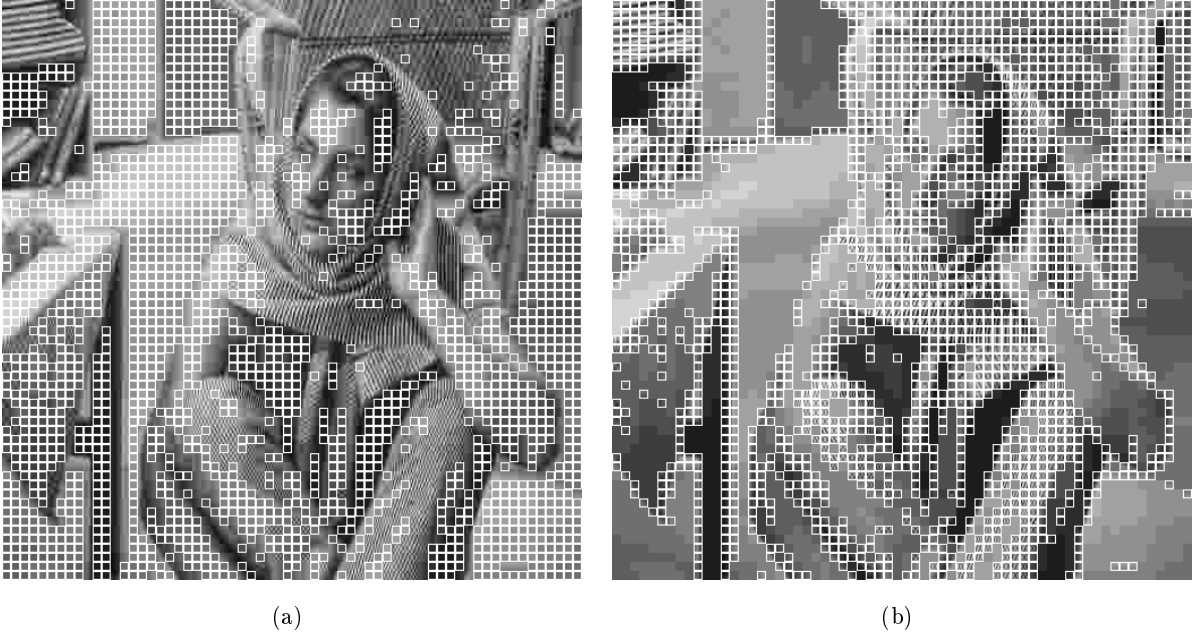


Figure 11: UBL and DBL for Barbara compressed at 0.24bpp (PSNR = 24.52 dB). Each block in either the DBL or the UBL is marked by a white frame. (a) UBL. (b) DBL.

2. This stage is comprised of three iterations of the WABG scheme using detail preservation mode  $M_{high}$ ,  $M_{medium}$  and  $M_{low}$ , in this order. Each iteration is performed on the blocks in the UBL and uses a deblocking frame of different size. The first iteration uses a deblocking frame of size eight. The second and third iterations use deblocking frames of sizes four and two, respectively.
3. A single iteration is performed using a deblocking frame of size four ( $S_f = 4$ ). The detail preservation mode is set to  $M_{low}$ . This stage uses a slightly modified version of  $\kappa(m, n)$

$$\kappa(m, n) = \begin{cases} -1 & \text{if } n - m = 1 \\ \frac{1}{8} & \text{if } n - m = 3 \end{cases}$$

Deblocking frames, which are located inside a single block, are not processed.

Figure 12 demonstrates the deblocking results of the WABG (fig. 12(a)) and DFOVS (fig. 12(b)) schemes when applied on monotone areas in the *Peppers* image. Comparison between the results reveals that the DFOVS scheme, achieves better deblocking than the WABG scheme, in monotone areas .

## 2.4 Deblocking of a DC image

When the iterative algorithm is applied on a DC image, it uses a different *Diff* function in order to avoid excessive blurring of the image and to allow contours reconstruction. DC images are easily recognized by checking whether the size of the UBL is equal to  $\frac{R \times C}{64}$ , i.e. whether all the blocks in the

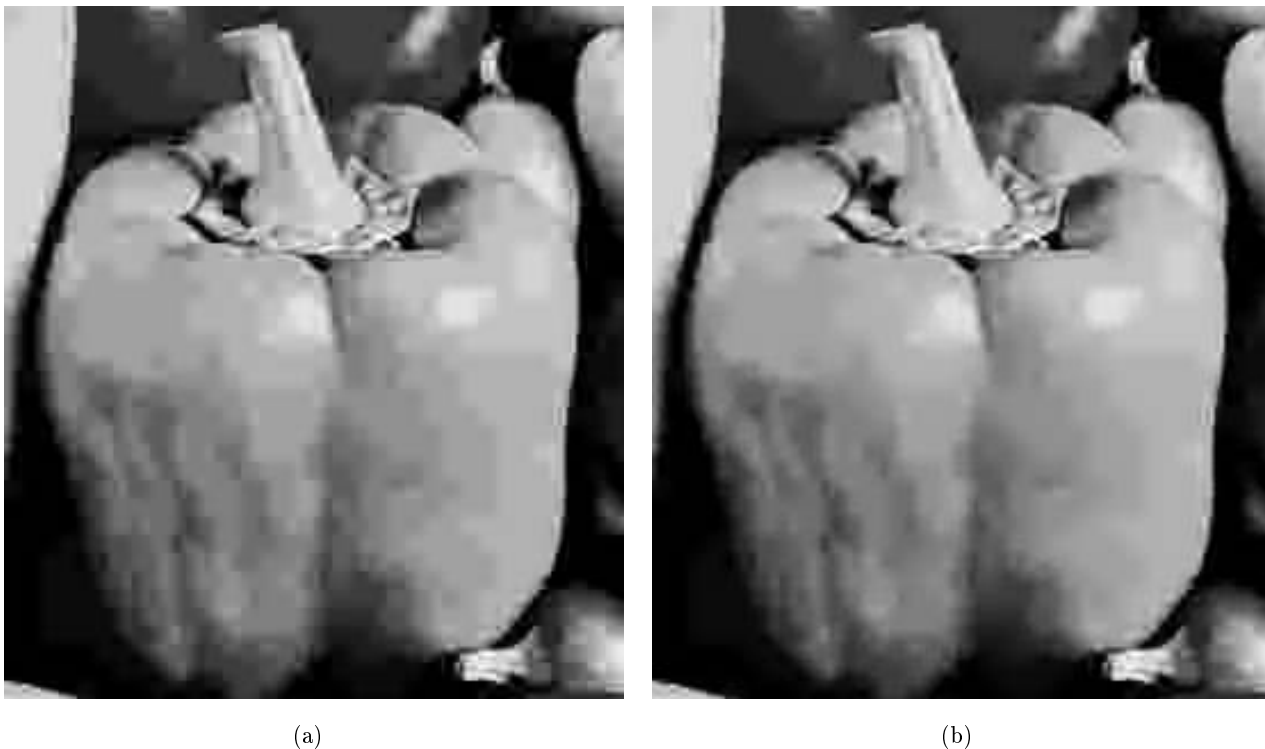


Figure 12: Application of the DFOVS scheme on *Peppers* compressed at 0.21bpp (PSNR = 27.32 dB). (a) After the application of the WABG scheme (PSNR = 27.96 dB). The scheme uses linear weights with  $\eta = 1$ ,  $\vartheta = 0.63$  and the variance based block grading implementation. (b) After the application of the DFOVS scheme (PSNR = 28.20 dB). The algorithm uses the variance based block grading implementation. The second and third stages use linear weights with  $\eta = 0.8$ ,  $\vartheta = 0.7$  and  $\eta = 0.9$ ,  $\vartheta = 0.55$ , respectively.

image are uniform. In this case, the algorithm uses the following *Diff* function

$$\text{Diff}(p_{i,j}, p_{k,l}) = \frac{1 + |p_{i,j} - p_{k,l}|}{256} \quad 0 \leq i, j \leq R - 1, 0 \leq k, l \leq C - 1$$

### 3 Experimental results

In order to evaluate the deblocking capabilities of our proposed algorithms we use a number of  $512 \times 512$  JPEG compressed images which are widely used in the literature: *Lena*, *Barbara*, *Peppers*.

First, we investigate the visual quality of the deblocked images obtained by our DFOVS algorithm. Then, we compare between its results and the results of a selection of related mentioned algorithms. The deblocking algorithms, which are used for comparison, are the ones which were described in [5, 7, 8, 11, 12, 13]. All images were compressed using the *cjpeg* UNIX [16].

The DFOVS approach, which uses linear weights with  $\eta = 0.8$ ,  $\vartheta = 0.7$  for the second stage and linear weights with  $\eta = 0.9$ ,  $\vartheta = 0.55$  for the third stage, was used in the experiments.

#### 3.1 Visual results

We first analyze the visual results which were obtained by applying the DFOVS scheme on the test images. This is followed by a comparison with the results that were obtained by the other related algorithms.

Each deblocking method is evaluated according to its performance in monotone areas and in detailed areas. This is due to the fact that blocking artifacts are more visible in monotone areas and should be eliminated while preserving details. Examples of monotone areas are located in the shoulder of *Lena* in figure 13(b) and the forearm of *Barbara* in fig. 15(b). The faces of *Lena* and *Barbara* are examples of regions containing both detailed and monotone areas.

The DFOVS algorithm achieved very good deblocking results both in monotone and detailed areas. Furthermore, it preserves details. In monotone areas, the blocking artifacts were reduced to a point where they are barely noticeable.

The preservation of details is demonstrated in 15(c) where the pattern of the table cloth, trousers and shawl in *Barbara* are preserved after the application of the DFOVS algorithm. Additionally, examining the faces of *Lena* and *Barbara* confirms that this method does not damage any details.

The result of the application of the DFOVS method on DC images is illustrated in Fig. 17(a). We can see that the algorithm successfully avoids damaging object contours while dramatically reducing the blocking artifacts. Specifically, the contours of the peppers are not damaged during the elimination of the blocking artifacts.

#### 3.2 Comparison to other algorithms

The algorithm in [7] achieved satisfactory deblocking of detailed areas, but fails to do so in monotone areas. This is due to the fact that this algorithm processes only the two pixels adjacent to the boundary of each block. This is confirmed by examining 13(d) and 15(d). Application of this algorithm on DC images produces very poor results.

The deblocking algorithm in MPEG4 standard [5] achieved the best visual results among all the algorithms used for comparison. However, this algorithm does not provide enough deblocking in detailed areas. Furthermore, the results of this algorithm contain small regions which exhibit blocking artifacts. These artifacts are manifested as isolated detailed blocks in monotone areas. The major deficiency of this algorithm is its inability to deblock DC images. Nonetheless, it is important to comment that this deblocking algorithm was intended to serve video sequences. Thus, it assumes that the quantization factor from each macroblock is available. We used it for still images with a fixed quantization factor.

The algorithm in [11] achieves better deblocking in monotone areas than the algorithms in [7] and [13]. However, blocking artifacts are still visible in monotone area. Moreover, minor blurring is introduced in detailed areas. Application of this algorithm on DC images produces images that still suffer from blocking artifacts.

The algorithm in [12] facilitates very good deblocking of monotone areas but it introduces excessive blurring to the image. This algorithm produces a very blurred image which still exhibits blocking artifacts when applied on DC image.

The results of [13] are similar results to the results obtained by the algorithm in [7]. However, the blocking artifacts, which are exhibited in monotone areas, are more noticeable than those in [7]. Deblocked detailed areas still exhibit blocking artifacts. When applied on DC images, blocking artifacts are still noticeable.

The algorithm in [8] produces good deblocking results for DC images. These results bear some resemblance to the results of the DFOVS approach.

### 3.3 PSNR results

It is well established that PSNR fails to provide a decisive indication to the visual quality of an image. Often, images which have superior visual quality have a lower PSNR (Figs. 13(c),(d)). Nevertheless, for the sake of completeness, we present the PSNR results obtained by the DFOVS algorithm and the related algorithms chosen for comparison [5, 7, 8, 11, 12, 13]. Table 1 displays the PSNR values for different algorithms compared to our DFOVS method. Table 2 specifies PSNR values for the DC version of the test images. The bit-rate for these images is 0.18, therefore, we exclude a bit-rate column from this table.

### 3.4 PSNR versus bit rates

The graphs in Fig. 19 depict PSNR values versus bitrate (bits per pixel). The test images were postprocessed by the new and the related algorithms. The PSNR results, obtained by the DFOVS method for *Barbara* (Fig. 19(a) ), are the highest at bitrates below 0.24bpp. In higher bitrates, the results of the DFOVS method are among the top three algorithms. When applied on *Peppers* (Fig. 19(b)), the DFOVS method produces the best PSNR results, at bitrates below 0.2bpp. It is among the top two algorithms for 0.2bpp–0.29bpp bitrates. In higher bitrates, the results obtained by the DFOVS method are among the top three algorithms.



(a)



(b)



(c)



(d)

Figure 13: Deblocking using DFOVS and related algorithms for *Lena*. (a) Original image. (b) Compressed at 0.22bpp (PSNR = 29.47 dB). (c) Our DFOVS approach (PSNR = 30.27 dB). (d) Castagno *et al.* [7] (PSNR = 30.34 dB).



(a)



(b)



(c)



(d)

Figure 14: Deblocking results from related algorithms for *Lena* compressed at 0.22bpp (PSNR = 29.47 dB). (a) MPEG4 standard [5] (PSNR = 29.96 dB). (b) Wu *et al.* [11] (PSNR = 30.37 dB). (c) Zakhor [12] (PSNR = 28.80 dB). (d) Galatsanos *et al.* [13] (PSNR = 30.19 dB).



(a)



(b)



(c)



(d)

Figure 15: Deblocking using DFOVS and related algorithms for *Barbara*. (a) Original image. (b) Compressed at 0.24bpp (PSNR = 24.52 dB). (c) Our DFOVS approach (PSNR = 25.04 dB). (d) Castagno *et al.* [7] (PSNR = 24.83 dB).



(a)



(b)



(c)



(d)

Figure 16: Deblocking results from related algorithms for *Barbara* compressed at 0.24bpp (PSNR = 24.52 dB). (a) MPEG4 standard [5] (PSNR = 24.72 dB). (b) Wu *et al.* [11] (PSNR = 25.08 dB). (c) Zakhor [12] (PSNR = 24.42 dB). (d) Galatsanos *et al.* [13] (PSNR = 24.97 dB).



(a)



(b)



(c)



(d)

Figure 17: Deblocking using DFOVS and related algorithms for the DC image of *Peppers*. (a) DC image - 0.18bpp (PSNR = 20.32 dB). (b) Our DFOVS approach (PSNR = 21.31 dB). (c) MPEG4 standard [5] (PSNR = 20.41 dB). (d) Castagno *et al.* [7] (PSNR = 20.48 dB).



(a)



(b)



(c)



(d)

Figure 18: Deblocking results from related algorithms for the DC image of *Peppers* - 0.18bpp (PSNR = 20.32 dB). (a) Ramponi and Carrato [8] (PSNR = 22.27 dB). (b) Zakhor [12] (PSNR = 21.26 dB). (c) Galatsanos *et al.* [13] (PSNR = 21.07 dB). (d) Wu *et al.* [11] (PSNR = 21.40 dB).

Image	Bit Rate (bpp)	PSNR						
		After JPEG	DFOVS	[5]	[7]	[12]	[13]	[11]
Lena	0.16	26.49	27.63	27.06	27.26	26.13	27.36	27.56
	0.19	28.19	29.2	28.77	29.13	27.75	29.00	29.22
	0.22	29.47	30.27	29.96	30.34	28.80	30.19	30.37
	0.24	30.40	31.01	30.84	31.17	29.63	31.03	31.15
	0.27	31.09	31.56	31.48	31.79	30.26	31.63	31.73
Barbara	0.20	23.46	24.13	23.75	23.86	23.52	23.93	24.13
	0.24	24.52	25.04	24.72	24.83	24.42	24.97	25.08
	0.28	25.24	25.66	25.39	25.45	25.06	25.67	25.71
	0.32	25.84	26.21	25.97	25.94	25.63	26.27	26.25
	0.36	26.40	26.69	26.51	26.34	26.13	26.81	26.72
peppers	0.18	25.62	26.72	26.13	26.45	25.21	26.40	26.49
	0.21	27.32	28.20	27.78	28.22	26.61	27.94	27.86
	0.24	28.43	29.07	28.83	29.28	27.65	28.93	28.80
	0.27	29.14	29.65	29.50	29.94	28.34	29.57	29.37
	0.30	29.76	30.11	30.05	30.47	29.75	30.11	29.89

Table 1: Comparison of PSNR results among different postprocessing deblocking methods with our new proposed methods. The column “After JPEG” means that no postprocessing to deblock was performed.

Image	PSNR							
	After JPEG	DFOVS	[5]	[7]	[8]	[12]	[13]	[11]
Lena	23.75	24.5	23.91	24.01	25.28	23.65	24.43	24.73
Barbara	19.57	19.69	19.57	19.60	19.93	19.31	19.68	19.81
peppers	20.32	21.31	20.41	20.48	22.27	21.26	21.07	21.40

Table 2: Comparison among PSNR results of available postprocessing deblocking methods with our proposed methods for the DC version of the test images. The bitrate for these coded images is 0.18bpp. The column “After JPEG” means that no postprocessing to deblock was performed.

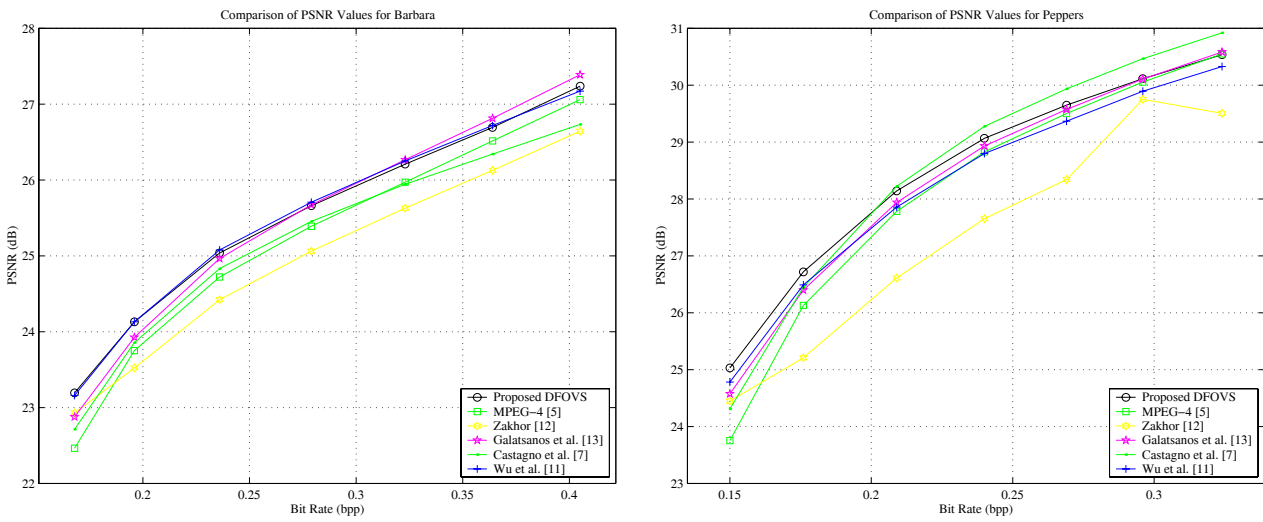


Figure 19: PSNR values versus bit rate (bits per pixel) charts. (a) *Barbara*. (b) *Peppers*.

## 4 Conclusions

This paper presented a new class of related algorithms for the elimination of blocking artifacts in BTC compressed images. The algorithms were based on using weighted sums on symmetrically aligned pixels (WSSAP). The basic algorithm was aimed at deblocking of images compressed at extremely low bitrate. The application of this scheme on images compressed at low to medium bitrates produced deblocked images which contained a ghosting effect. The weight adaptation by grading scheme (WABG) was introduced, in order to prevent the occurrences of the ghosting effect. The deblocking frames of variable size (DFOVS) scheme was proposed in order to achieve better deblocking in monotone areas. Experimental results demonstrated that our proposed algorithms achieve excellent visual and PSNR results. Nevertheless, the PSNR values were not always superior compared to related algorithms, although their corresponding images demonstrated better visual quality. These methods should be incorporated into video systems such as: MPEG-4 [4, 5], H.263[17] etc. that aim at low to medium bitrates. Combination of the new methods with MPEG type decoders can be very useful to video processing at low bitrates.

## References

- [1] ISO/IEC JTC1 Committee Draft 10918-1, "Digital compression and coding of continuous-tone still images, part 1, requirements and guidelines," February 1991.
- [2] ISO/IEC 11172, "Information technology - coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s.," 1993.
- [3] ISO/IEC 13818, "Information technology - generic coding of moving pictures and associated audio information.," 1994.

- [4] ISO/IEC 14496-2, “Information technology - coding of audio-visual objects, part 2: visual, amendment 1: visual extensions,” December 1999.
- [5] ISO/IEC 14496-2, “Mpeg4 video verification model version 18.0,” .
- [6] G. Ramponi S. Carrato and S. Marsi, “A simple edge-sensitive image interpolation filter,” in *Proc. Third IEEE Intern. Conf. on Image Processing, ICIP-96*, Lausanne (Switzerland), September 16–19 1996, IEEE, pp. III-711 – III-714.
- [7] R. Castagno S. Marsi and G. Ramponi, “A simple algorithm for the reduction of blocking artifacts in images and its implementation,” *IEEE Trans. Consumer Electron.*, vol. 44, no. 3, August 1998.
- [8] G. Ramponi and S. Carrato, “Interpolation of the dc component of coded images using a rational filter,” in *Proc. of the Int. Conf. on Image Processing, ICIP*, Santa Barbara, CA, October 1997.
- [9] G. Ramponi, “The rational filter for image smoothing,” *IEEE Signal Processing Lett.*, vol. 3, no. 3, pp. 63–65, March 1996.
- [10] D. L. Donoho, “Smooth wavelet decompositions with blocky coefficient kernels,” Tech. Rep., Department of Statistics, Stanford University, Available <ftp://playfair.stanford.edu>, 1993.
- [11] H. G. Wu T. Chen and B. Qiu, “Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 584–602, May 2001.
- [12] A. Zakhor, “Iterative procedures for reduction of blocking artifacts in transform image coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 91–95, March 1992.
- [13] N. P. Galatsanos Y. Yang and A. K. Katsaggelos, “Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 421–432, December 1993.
- [14] M.-Y. Shen and C. C. Jay Kuo, “Review of postprocessing techniques for compression artifacts removal,” *Journal of Visual Communication and Image Representation*, vol. 9, no. 1, pp. 2–14, March 1998.
- [15] M. Yuen and H. R. Wu, “A survey of hybrid mc/dpcm/dct video coding distortions,” *Signal Processing*, vol. 70, no. 3, pp. 247–278, July 1998.
- [16] “<ftp.uu.net:/graphics/jpeg/jpegsrc.v6b.tar.gz>,” .
- [17] ITU-T, “Video coding for narrow telecommunication channels at 64 kbit/s, draft recommendation h.263.,” May 1996.

# List of Figures

1	Examples of $\mathbf{B}(S_f)_{r,c}$ in 2 and 3. (a) $S_f = 8$ . (b) $S_f = 4$ . . . . .	4
2	Illustration of the symmetry notion. For example, $p_{2,1}$ is symmetrically aligned to $p_{2,6}$ as well as to $p_{5,1}$ . The dashed lines indicate the central axes of the deblocking frame and also the block boundaries induced by the BTC. . . . .	5
3	1-D linear and quadratic $\omega(i)$ and $\psi(i)$ . (a) Linear with $\eta = 1$ and $\vartheta = \frac{4}{7}$ . (b) Quadratic $\eta = 1$ and $\vartheta = \frac{1}{2}$ . . . . .	8
4	1-D blocking artifact and its deblocking. (a) A 1-D blocking artifact. (b) Deblocking with linear weights using $\eta = 1$ and $\vartheta = \frac{4}{7}$ . (c) Deblocking with quadratic weights using $\eta = 1$ and $\vartheta = \frac{1}{2}$ . . . . .	9
5	2-D <i>linear</i> weights $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \delta_{i,j}$ using $\eta = 1$ and $\vartheta = \frac{4}{7}$ . (a) $\alpha_{i,j}^L$ . (b) $\beta_{i,j}^L$ . (c) $\gamma_{i,j}^L$ . (d) $\delta_{i,j}^L$ . . . . .	10
6	2-D <i>quadratic</i> weights $\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j}, \delta_{i,j}$ using $\eta = 1$ and $\vartheta = \frac{1}{2}$ . (a) $\alpha_{i,j}^Q$ . (b) $\beta_{i,j}^Q$ . (c) $\gamma_{i,j}^Q$ . (d) $\delta_{i,j}^Q$ . . . . .	11
7	Application of the basic approach using linear weights with $\eta = 1$ , $\vartheta = 0.63$ , $S_f = 8$ on the <i>Peppers</i> image. (a) compressed at 0.21bpp (PSNR = 27.32 dB). (b) after the application of the basic approach (PSNR = 27.49 dB). (c) DC image (PSNR = 20.32 dB). (d) deblocking result of (c) using the basic approach (PSNR = 21.45 dB). . . . .	13
8	Example of the ghosting effect in the <i>Peppers</i> image compressed at 0.21bpp. (a) magnification of the marked area in Fig. 7(a). (b) magnification of the marked rectangle area in Fig. 7(b) - the ghosting effect is manifested as the blurred “false” edge in the middle of the figure. . . . .	14
9	The values of $\kappa(m, n)$ for the calculation of pixels $p_{i,j}$ where $0 \leq i, j \leq \frac{S_f}{2} - 1$ . The values of $\kappa(m, n)$ for pixels $p_{i,j}$ where $0 \leq i \leq \frac{S_f}{2} - 1$ and $\frac{S_f}{2} \leq j \leq S_f - 1$ , are obtained by rotating the figure 90° clockwise. $\kappa(m, n)$ values for the pixels in quarters $\frac{S_f}{2} \leq i \leq S_f - 1$ , $0 \leq j \leq \frac{S_f}{2} - 1$ and $\frac{S_f}{2} \leq i, j \leq S_f - 1$ are obtained by rotating the figure 180° and 270°, respectively. . . . .	15
10	Application of the WABG scheme on <i>Peppers</i> compressed at 0.21bpp. The algorithm uses linear weights with $\eta = 1$ , $\vartheta = 0.63$ and the variance based block grading implementation. (a) The image after the application of the WABG scheme (PSNR = 27.96 dB). (b) Magnification of the rectangular marked area in (a) which corresponds to the rectangular marked area in Fig. 8(b) - no ghosting effect is visible. . . . .	17
11	UBL and DBL for Barbara compressed at 0.24bpp (PSNR = 24.52 dB). Each block in either the DBL or the UBL is marked by a white frame. (a) UBL. (b) DBL. . . . .	18
12	Application of the DFOVS scheme on <i>Peppers</i> compressed at 0.21bpp (PSNR = 27.32 dB). (a) After the application of the WABG scheme (PSNR = 27.96 dB). The scheme uses linear weights with $\eta = 1$ , $\vartheta = 0.63$ and the variance based block grading implementation. (b) After the application of the DFOVS scheme (PSNR = 28.20 dB). The algorithm uses the variance based block grading implementation. The second and third stages use linear weights with $\eta = 0.8$ , $\vartheta = 0.7$ and $\eta = 0.9$ , $\vartheta = 0.55$ , respectively. . . . .	19

13	Deblocking using DFOVS and related algorithms for <i>Lena</i> . (a) Original image. (b) Compressed at 0.22bpp (PSNR = 29.47 dB). (c) Our DFOVS approach (PSNR = 30.27 dB). (d) Castagno <i>et al.</i> [7] (PSNR = 30.34 dB). . . . .	22
14	Deblocking results from related algorithms for <i>Lena</i> compressed at 0.22bpp (PSNR = 29.47 dB). (a) MPEG4 standard [5] (PSNR = 29.96 dB). (b) Wu <i>et al.</i> [11] (PSNR = 30.37 dB). (c) Zakhor [12] (PSNR = 28.80 dB). (d) Galatsanos <i>et al.</i> [13] (PSNR = 30.19 dB). . .	23
15	Deblocking using DFOVS and related algorithms for <i>Barbara</i> . (a) Original image. (b) Compressed at 0.24bpp (PSNR = 24.52 dB). (c) Our DFOVS approach (PSNR = 25.04 dB). (d) Castagno <i>et al.</i> [7] (PSNR = 24.83 dB). . . . .	24
16	Deblocking results from related algorithms for <i>Barbara</i> compressed at 0.24bpp (PSNR = 24.52 dB). (a) MPEG4 standard [5] (PSNR = 24.72 dB). (b) Wu <i>et al.</i> [11] (PSNR = 25.08 dB). (c) Zakhor [12] (PSNR = 24.42 dB). (d) Galatsanos <i>et al.</i> [13] (PSNR = 24.97 dB). . .	25
17	Deblocking using DFOVS and related algorithms for the DC image of <i>Peppers</i> . (a) DC image - 0.18bpp (PSNR = 20.32 dB). (b) Our DFOVS approach (PSNR = 21.31 dB). (c) MPEG4 standard [5] (PSNR = 20.41 dB). (d) Castagno <i>et al.</i> [7] (PSNR = 20.48 dB). . .	26
18	Deblocking results from related algorithms for the DC image of <i>Peppers</i> - 0.18bpp (PSNR = 20.32 dB). (a) Ramponi and Carrato [8] (PSNR = 22.27 dB). (b) Zakhor [12] (PSNR = 21.26 dB). (c) Galatsanos <i>et al.</i> [13] (PSNR = 21.07 dB). (d) Wu <i>et al.</i> [11] (PSNR = 21.40 dB). . . . .	27
19	PSNR values versus bit rate (bits per pixel) charts. (a) <i>Barbara</i> . (b) <i>Peppers</i> . . . . .	29

## Contact information

### **Amir Averbuch :**

School of Computer Science  
Tel Aviv University, Tel Aviv 69978, Israel  
Phone: +972-3-6422020  
Fax: +972-3-6409357  
e-mail: amir@math.tau.ac.il

### **Alon Schclar :**

School of Computer Science  
Tel Aviv University, Tel Aviv 69978, Israel  
Phone: +972-3-6422020  
Fax: +972-3-6409357  
e-mail: shekler@post.tau.ac.il

### **David Donoho :**

Department of Statistics  
Stanford University  
Sequoia Hall  
Stanford, CA 94305, USA  
Phone: +1-650-723-3350  
Fax: +1-650-725-8897  
e-mail: donoho@stat.stanford.edu