Real-Time Video Postprocessing Algorithms and Metrics

by

Wenfeng Gao

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2003

Program Authorized to Offer Degree: Department of Electrical Engineering

UMI Number: 3090997

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to UMI Dissertation Services, 300 North Zeeb Road, P.O. Box 1346, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature Wafa Cass

Date 4/15/2003

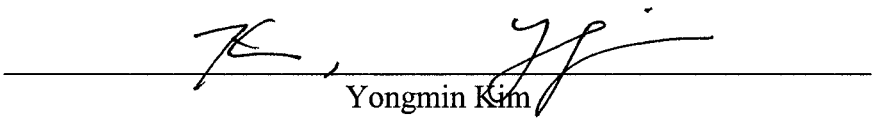
University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Wenfeng Gao

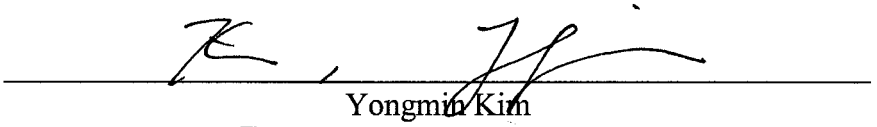
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:



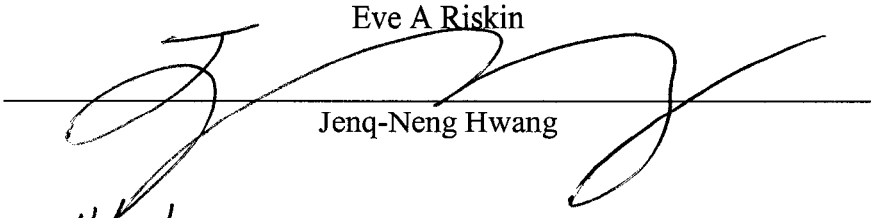
Yongmin Kim

Reading Committee:



Yongmin Kim

Eve A Riskin



Jenq-Neng Hwang

Date: 4/12/03

University of Washington

Abstract

REAL-TIME VIDEO POSTPROCESSING ALGORITHMS AND METRICS

by Wenfeng Gao

Chairperson of the Supervisory Committee: Professor Yongmin Kim
Departments of Bioengineering and Electrical Engineering

The main purpose of video postprocessing is to reduce compression artifacts, such as blocking, ringing, and temporal noise. Many existing postprocessing algorithms either require very high computational cost or cannot visually remove the artifacts, especially at very low bit rates. In this research, we propose an adaptive de-blocking algorithm, a clustering-based de-ringing algorithm, and a motion-compensated temporal filtering with a down-sampled pattern. These algorithms can smooth out the compression artifacts while preserving the strong edges and texture areas. In addition to effective artifact reduction, our algorithms have low computational cost. The implementation of our de-blocking and de-ringing algorithms on a mediaprocessor demonstrates the feasibility of real-time video postprocessing. In addition, we have also developed new metrics to measure the blocking artifacts in the reconstructed or postprocessed images. To validate our algorithms and metrics, we have designed statistics-based subjective quality assessment experiments. The experimental results confirm that our proposed postprocessing algorithms have achieved a significant improvement over other methods

while the proposed blockiness metric is more consistent with subjective evaluation than peak signal-to-noise ratio (PSNR).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First of all, I would like to express my greatest appreciation to my advisor Dr. Yongmin Kim for his supports and important contributions to this project. I would also like to thank the rest of my committee members Dr. Jenq-Neng Hwang, Dr. Ravi Managuli, Dr. Eve A. Riskin, Dr. Ming-Ting Sun, and Dr. Kristiina A. Vogt for their useful suggestions.

This research was sponsored in part by Hitachi Ltd. Thanks to Dr. Atsuo Kawaguchi for facilitating this research. In addition, I would like to thank all the Image Computing Systems Laboratory (ICSL) members and in particular Coskun Mermer and Dr. Lixin Gong.

Thanks also to my wife, Yuhang, for her patience and the constant support and thanks to my parents for their encouragement and advice.

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND AND SIGNIFICANCE

In order to reduce the transmission cost and/or save storage space, image/video compression techniques have been widely used in many applications, including image archiving, video conferencing, videophones, remote monitoring and control, video on demand, internet video, wireless video, and multimedia communication.

Currently, most image/video compression standards are based on two major transform techniques: discrete cosine transform (DCT) and discrete wavelet transform (DWT). For still images, the DWT-based JPEG2000 [37] can outperform the DCT-based JPEG [38] when the bit rate is very low (e.g., < 0.25 bits per pixel). However, the DCT-based standards, such as MPEG-1 [30][31], MPEG-2 [32], MPEG-4 [35][36], and H.26x series [40][41][42], still dominate video compression. Fig. 1-1 shows a MPEG-like DCT-based encoding scheme. The motion estimation (ME) step estimates the motion vectors between the current frame and its reference frame. After motion compensation, the residual image is divided into 8×8 blocks, and each block is transformed into 64 DCT coefficients. The quantization step reduces the resolution of the DCT coefficients so that fewer bits can be used to represent them. Usually, the larger the quantization step size, the higher the compression ratio. Since quantization is irreversible, it is impossible to completely recover the original DCT coefficients after quantization. As a result, low bit rate reconstructed video has blocking and ringing artifacts. Blocking artifacts arise from the independent coding of each $N \times N$ (e.g., $N=8$) block, and they appear as false horizontal and vertical edges along block boundaries. Ringing artifacts come from the loss of high frequency details. In still images, ringing artifacts appear as spurious oscillations in the vicinity of major edges [73]. Ringing artifacts in video signals are also

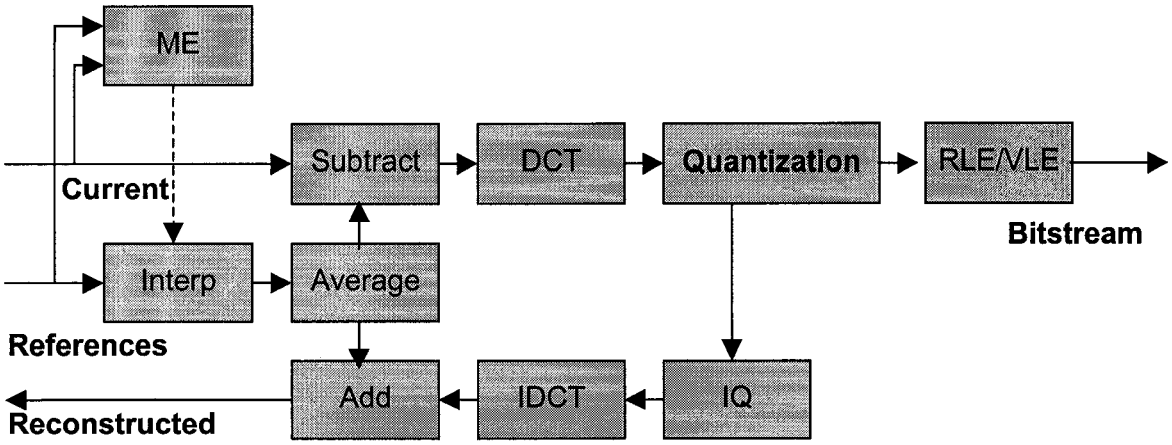


Figure 1-1. Example of DCT-based coding scheme.

called mosquito noise because they appear as moving ripples and swarm around sharp edges [43].

Figure 1-2 shows an example of blocking and ringing artifacts. Figure 1-2(a) is the original FOREMAN image while Fig. 1-2(b) is its reconstructed image compressed at quality=5 via cjpeg [82]. We can easily observe severe blocking artifacts around the relatively homogeneous areas, such as face and walls. Ringing artifacts are not so obvious as blocking artifacts. However, if we look carefully at those sharp edges, we can see many perturbations around them.

In addition to blocking and ringing artifacts, videos involve temporal noise as well. Temporal noise in decoded videos is usually caused by interframe abrupt changes. It can be more annoying than spatial noise because it makes video look "busy." As an example, Fig. 1-3 shows the pixel value at a fixed location (104, 312) over different frames. The

solid line and the dashed line represent the pixel values extracted from the original and the reconstructed FOREMAN sequences, respectively. It is clear that the pixel value in the original FOREMAN sequence usually changes smoothly over frames (except a few frames with scene changes) while more abrupt changes appear after compression.

(a) (b)

Figure 1-2. The original and reconstructed FOREMAN images. (a) The original FOREMAN image and (b) the reconstructed FOREMAN image compressed at quality=5 via cjpeg.

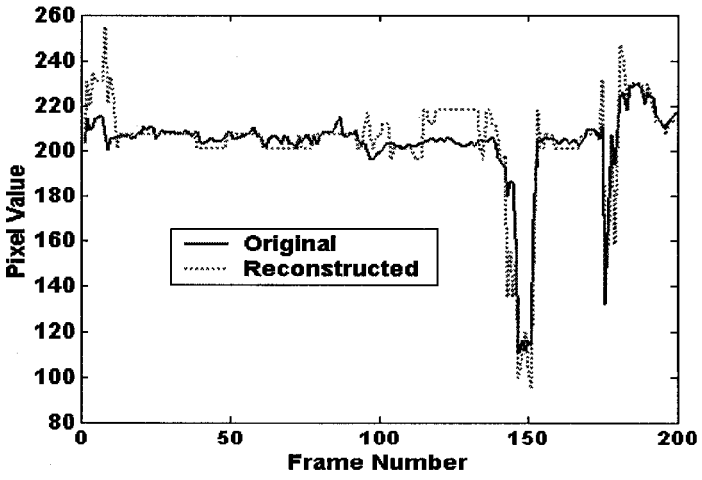


Figure 1-3. Pixel value at (104, 312) over different frames.

The main purpose of video postprocessing is to remove/reduce these compression artifacts. Many postprocessing algorithms have been proposed. Most of them require very high computational cost. For real-time video postprocessing, noniterative algorithms with low computational cost would be desirable. However, the results of many current noniterative algorithms are not satisfactory, especially at very low bit rates. Therefore, we believe that there is a clear need for postprocessing algorithms with both low computational cost and good artifact-reduction ability.

The focus of the presented research work is to develop noniterative algorithms, which can effectively remove compression artifacts and can be implemented on inexpensive processors for real-time video postprocessing. To achieve this goal, we propose the following specific aims:

1) De-blocking algorithm.

   The noniterative de-blocking algorithms we have found in published work cannot effectively remove the blocking artifacts in highly compressed images. Some of them introduce new artifacts (blurring strong edges and textures). We aim to provide a noniterative algorithm that can visually remove the blocking artifacts and keep the strong edges and textures untouched.

2) Blockiness metric.

   Many authors are still using peak signal-to-noise ratio (PSNR) or mean square error (MSE) to measure blocking artifacts although they are poor indicators. We aim to establish an objective blockiness metric that is more consistent with subjective evaluation than PSNR/MSE.

3) De-ringing algorithm.

   Ringing artifacts are only noticeable in blocks with strong edges. We aim to develop a de-ringing algorithm that can smooth out the small perturbations while keeping the strong edges.

4) Implementation on mediaprocessors

   Our proposed de-blocking and de-ringing algorithms should be implementable on mediaprocessors for real-time de-blocking and de-ringing.

5) Temporal filtering algorithm.

   Temporal filtering has been widely used in video preprocessing. In this study, we will apply it in video postprocessing to remove the interframe artifacts.

6) Algorithm validation.

   The proposed postprocessing algorithms will be validated by subjective quality assessment experiments.

Figure 1-4 shows the diagram of our proposed video postprocessing system. The input is the decoded video sequence. The semaphore extraction part is the control unit of the system. Semaphore is a binary marker. The extracted blocking semaphores and ringing semaphores determine which blocks are going to be de-blocked and/or de-ringed. After the selective de-blocking and de-ringing, temporal filtering is applied to reduce the interframe noise. The final output is the postprocessed video sequence.

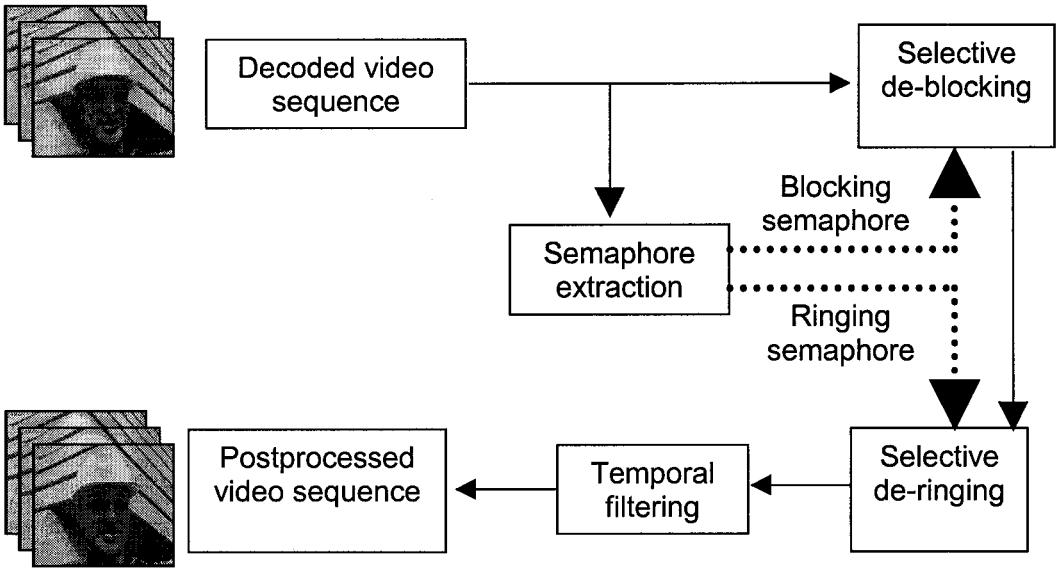


Figure 1-4. Diagram of proposed video postprocessing system.

1.2 THESIS ORGANIZATION

Chapter 2 describes the adaptive de-blocking algorithm and the objective blockiness metric. Chapter 3 explains the de-ringing algorithm. Chapter 4 presents the implementation of de-blocking and de-ringing on a mediaprocessor called MAP-CA. The temporal filtering scheme is studied in Chapter 5. Chapter 6 presents the algorithm validation methodology. Finally, in Chapter 7, the conclusions of our research are provided with a discussion of future directions.

CHAPTER 2: A DE-BLOCKING ALGORITHM AND A BLOCKINESS METRIC

2.1 INTRODUCTION

The block-based discrete cosine transform (BDCT) has been adopted in most image/video compression standards, including JPEG [38], MPEG [34], and H.26L [42], because it has three main advantages: 1) good energy compaction property; 2) low computational cost; and 3) easy hardware implementation [45]. For the low bit-rate video coded by BDCT, the high-frequency DCT coefficients tend to be removed because they are coarsely quantized. Since each $N \times N$ (e.g., N=8) block is coded independently, the reconstructed image can generate discontinuities along block boundaries, commonly referred to as blocking artifacts. Usually, the lower the bit rate, the more severe the blocking artifacts.

Many algorithms have been proposed for reducing the blocking artifacts. These algorithms can be generally grouped into two major categories. One is to use different encoding schemes, such as the interleaved block transform [19][77], the lapped transform [28][66][67], and the combined transform [103]. The other is to postprocess the reconstructed images [74][76][84][97][102]. Since the postprocessing techniques do not require changes to the existing standards, they are more practical solutions. A popular de-blocking method is the projection onto convex sets (POCS) proposed by Zakhor [102]. The basic idea is to represent every known property of an original image by a closed convex set. Since the original image is located within the intersection of all the closed convex sets, the goal of this iterative algorithm is to find an image that is also located within this intersection by alternating projections onto each closed convex set. Some other de-blocking algorithms are statistics-based. O'Rourke and Stevenson [74]

proposed using the maximum *a posteriori* probability of the desired image given the decompressed image. Yang *et al.* [97] converted the blocking artifacts into some penalty using a cost function. Then, the task of de-blocking becomes how to minimize the cost function. The major disadvantage of these methods is their computational complexity. One application area of de-blocking algorithms is real-time low bit-rate video decoding, where noniterative de-blocking algorithms with low computational complexity would be desirable.

To reduce the computational cost of de-blocking, Shen and Kuo [84] proposed using a lookup table method. Park and Lee [76] used the DCT coefficients from the compressed data to extract the semaphores for blocking artifacts. Then, they applied a one-dimensional (1-D) low-pass filter to reduce the blocking artifacts both horizontally and vertically. These noniterative algorithms are computationally more efficient, but they have a common disadvantage in that only a few pixels around the block boundary are smoothed after de-blocking. For highly compressed images, several connected blocks can have the same gray value, and thus the blocking artifacts appear as the discontinuity between two uniform regions. As a result, smoothing a few pixels around the block boundary is, in many cases, not enough to reduce the blocking artifacts visually. In this chapter, we propose an adaptive de-blocking algorithm. This algorithm first identifies those block pairs that need de-blocking. Then, it uses the number of connected blocks in a relatively homogeneous region, the magnitude of abrupt changes across block boundaries, and the quantization step size of DCT coefficients to adapt the filtering.

In addition, it is well known that the common image quality metrics, such as PSNR and MSE, have a poor predictive value on the degree of blocking artifacts. Thus, it would be desirable to establish a new metric that is more robust and can better correlate blocking

artifacts with its value. Several metrics have been proposed to measure blocking artifacts [48][60][70][87][92][94][101]. Karunasekera and Kingsbury [48] used the edge information in the error image to measure the blocking artifacts in the reconstructed images. This metric requires the original image, which might not be available at the decoding end. Liu *et al.* [60] used the slope information between two neighboring pixels to define the blocking and blurring degrees. However, the blocking and blurring strengths of the postprocessed images depend on the availability of the original image. Wu and Yuen [94] proposed a generalized block-edge impairment metric (GBIM) without the original image. The basic idea is to count the difference between columns $(8n+1)$ and $(8n)$ as the blocking artifacts. In general, it works for the reconstructed images. On the other hand, it may count strong edges as blocking artifacts and could lead to erroneous results because the original discontinuities along the block boundaries may shift to a new location after postprocessing. In this chapter, based on the frequency analysis of two neighboring blocks with blocking artifacts, we propose a new blocking artifact metric that can be applied to not only evaluating the reconstructed images, but also comparing the postprocessed images by different algorithms.

In the remainder of this chapter, Section 2.2 describes the adaptive de-blocking algorithm, while Section 2.3 explains the new blocking metric in detail. The simulation results are presented and discussed in Section 2.4. Finally, we summarize our conclusions in Section 2.5.

2.2 ALGORITHM

Our proposed algorithm consists of two steps: blocking semaphore extraction and de-blocking.

2.2.1 BLOCKING SEMAPHORE EXTRACTION

Blocking semaphore extraction is an important part in postprocessing because unnecessary de-blocking can blur the edges and texture areas and can also increase the computational cost. A good semaphore extraction algorithm only identifies those blocks that actually need de-blocking. Since the blocking artifacts in the texture and edge areas could be concealed by the textures and edges, our algorithm is to find those blocks located at the boundary of two different relatively homogeneous regions, where blocking artifacts are most noticeable. Here, we only explain how to extract semaphores for horizontal de-blocking because vertical de-blocking is similar to horizontal de-blocking.

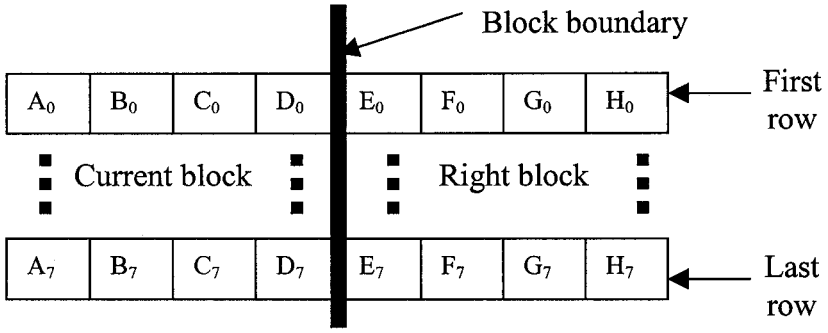


Figure 2-1. Current and right blocks in determining horizontal de-blocking semaphores.

Figure 2-1 shows a pair of 8×8 blocks, which consists of the current block and its right block. Basically, we look for those pairs of blocks that have relatively homogeneous pixel values in the direction perpendicular to the block boundary, while simultaneously exhibiting a discontinuity at the boundary. (A_0, B_0, C_0, D_0) and (A_7, B_7, C_7, D_7) represent the right four pixels of the first and last row of the current block, respectively, while (E_0, F_0, G_0, H_0) and (E_7, F_7, G_7, H_7) represent the left four pixels of the first and last row of the right block, respectively. Since there usually exists a high spatial correlation in a local image region, in this semaphore extraction step, we assume that the homogeneity of

the first and last rows can generally represent that of all eight rows. The algorithm to extract horizontal de-blocking semaphores is as follows.

(1) Compute the maximum variation (MV_{c0}) of A_0, B_0, C_0, and D_0 in Fig. 2-1 from their DC component using $MV_{c0} = max\{|A_0 - \mu_{c0}|, |B_0 - \mu_{c0}|, |C_0 - \mu_{c0}|, |D_0 - \mu_{c0}|\}$, where μ_{c0} is the mean value of A_0, B_0, C_0, and D_0. Similarly, compute the maximum variation (MV_{r0}) of E_0, F_0, G_0, and H_0.

(2) Find the abrupt change of the first row between the current block and its right block: $\Delta_0 = |E_0 - D_0|$.

(3) Calculate the maximum variation (MV_{c7} and MV_{r7}) and abrupt change (Δ_7) of the last row in a similar way.

(4) Finally, the maximum variation values (MV_{c0}, MV_{r0}, MV_{c7}, and MV_{r7}) and abrupt changes (Δ_0 and Δ_7) are used to determine whether this block suffers from blocking effect. If $max\{MV_{c0}, MV_{r0}, MV_{c7}, MV_{r7}\} < k * max\{\Delta_0, \Delta_7\}$ and $max\{\Delta_0, \Delta_7\} \leq T_{edge}$, then we mark this block pair for de-blocking. The first constraint excludes the block pairs with textures or edges. Here, k is set to 0.5. If the variation within a block is comparable to or even bigger than the boundary difference, then the abrupt change across the boundary would not be observable, so no de-blocking is applied. The first constraint also excludes pairs of two neighboring blocks with the same gray value to save the computational cost. We set a threshold T_{edge} in the second constraint to exclude the possibility of considering some strong edges exactly located at the block boundary as blocking artifacts. T_{edge} is set to $2*QP$, where QP is the quantization parameter of H.263 [18]. This is because the abrupt change in pixel values caused by quantization cannot exceed twice the quantization step size q in the pixel domain and q can be approximated by the quantization step size QP in the DCT domain [69].

2.2.2 DE-BLOCKING

The de-blocking algorithm is only applied to those block pairs identified in the semaphore extraction step. For highly compressed images, blocking artifacts can appear as strong discontinuities between two different uniform regions. In this case, smoothing only a few pixels around the block boundary cannot visually remove the blocking artifacts. Here, we propose an adaptive de-blocking algorithm. The basic idea is that the number of pixels to be updated is related to the magnitude of the abrupt change (Δ) and the number of connected homogeneous blocks on both sides of the block boundary. The detailed algorithm for horizontal de-blocking is given as follows (vertical de-blocking can be easily derived by symmetry).

(1) Count the number of connected homogeneous blocks that are located left to the boundary (N_c) using the following algorithm. Here, we label the current block as $LEFT_0$, its first neighbor to the left as $LEFT_1$, its second neighbor to the left as $LEFT_2$, and so on. Also, we use the same notation as in Fig. 1 in referring to the pixels of any block pair.

Initialize a temporary counter $i = 1$;

Form a block pair $\{LEFT_i, LEFT_{i-1}\}$;

While $(A_0=B_0=C_0=D_0=E_0)$ and $(A_7=B_7=C_7=D_7=E_7)$ and $(i < N_c\_max)$ {

$i = i + 1$;

Form a new block pair $\{LEFT_i, LEFT_{i-1}\}$;

}

$N_c = i$;

The loop can be iterated up to $N_c\_max = ceil(2*QP/N)$, where N is the block width (e.g., N=8) and *ceil* means that if the fraction part is nonzero, it is rounded up to the next integer. N_c_max is the number of blocks necessary to disperse any abrupt change (which is assumed to be $\leq 2*QP$ as explained in Section II-A) in the most gradual fashion (i.e., one gray-level value change between two consecutive pixels).

(2) Count the number of homogeneous blocks that are located right to the boundary (N_r) using a similar algorithm as in (1).

(3) For each row, apply a boxcar filter (i.e., an FIR filter where all coefficients are equal to 1) to $\min\{\Delta, \min\{N_c, N_r\}*N\}$ pixels distributed symmetrically around the block boundary. The length of the filter is also given by the same formula, i.e., $\min\{\Delta, \min\{N_c, N_r\}*N\}$.

2.3 A NEW BLOCKING ARTIFACT METRIC

This section describes a new blocking artifact metric to be used in evaluating the reconstructed images and comparing the images postprocessed by different de-blocking algorithms. The metric is derived from the $2N$-point 1D DCT coefficients of a typical block pair with blocking artifacts. Since vertical blocking artifacts are similar to horizontal blocking artifacts, we only explain how to analyze and measure the horizontal blocking artifacts. The final blocking artifact value (BAV) is the average of horizontal BAV (BAV_h) and vertical BAV (BAV_v).

2.3.1 FREQUENCY ANALYSIS OF BLOCKING ARTIFACTS

Figure 2-2 gives an example of two neighboring blocks forming a horizontal blocking artifact. Each of these two blocks has uniform pixel values and an abrupt change occurs across the block boundary. To be uniform within each block implies that the N-point horizontal 1-D DCT coefficients of each block are all zeros except the DC component (frequency index k=0). If we look at the horizontal $2N$-point DCT coefficients of the two blocks combined, the abrupt change in the block boundary results in some nonzero high-frequency coefficients. Figure 2-3 shows the combined $2N$-point (N=8) 1-D sequence $z(n)$, $0 \le n < 2N$, which is obtained by concatenating the N points from one row of the current block, $x(m)$, $0 \le m < N$, and the N points from the same row of the right block,

$$z(n) = \begin{cases} x(n) & 0 \le n < N \\ y(n-N) & N \le n < 2N \end{cases} \tag{2.1}$$

$$X(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos[\frac{(2n+1)k\pi}{2N}] \tag{2.2}$$

$$Y(k) = \alpha(k) \sum_{n=0}^{N-1} y(n) \cos[\frac{(2n+1)k\pi}{2N}] \tag{2.3}$$

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & k = 0 \\ \sqrt{\frac{2}{N}} & k \neq 0 \end{cases} \tag{2.4}$$

$$\begin{aligned} Z(k) &= \frac{1}{\sqrt{2}} \alpha(k) \sum_{n=0}^{2N-1} z(n) \cos[\frac{(2n+1)k\pi}{4N}] \\ &= \begin{cases} \frac{1}{\sqrt{2}} [X(\frac{k}{2}) + (-1)^{k/2} Y(\frac{k}{2})] & k = 0,2,4,\ldots,2N-2 \\ \frac{1}{\sqrt{2}} \alpha(k) \sum_{n=0}^{N-1} \{x(n) \cos[\frac{(2n+1)k\pi}{4N}] & \\ \quad + y(n) \cos[\frac{(2n+1)k\pi}{4N} + k\frac{\pi}{2}]\} & k = 1,3,5,\ldots,2N-1 \end{cases} \end{aligned} \tag{2.5}$$

$y(m)$, $0 \le m < N$. The procedure to derive $Z(k)$, $0 \le k < 2N$, the $2N$-point DCT transform of $z(n)$, is given in Eqs. (2.1)-(2.5).

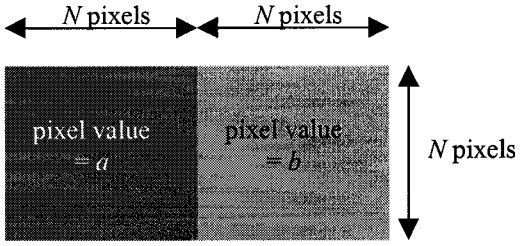


Figure 2-2. Example of blocking artifacts.

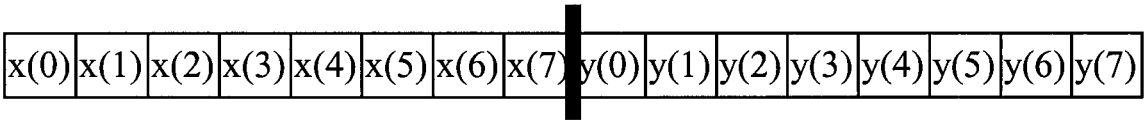


Figure 2-3. Combined 1-D sequence $z(n)$.

Eq. (2.5) shows that the even DCT coefficients $Z(k)$ can be directly derived from $X(k/2)$ and $Y(k/2)$. The odd DCT coefficients $Z(k)$ are a weighted summation of $x(m)$ and $y(m)$. If we assume that $x(m)$ has a value of a and $y(m)$ has a value of b as shown in Fig. 2-2, then the DCT coefficients $Z(k)$ can be explicitly written as Eq. (2.6).

Figure 2-4 shows the DCT coefficients $Z(k)$, where a and b are chosen as 80 and 100, respectively. In order to more clearly see the high-frequency components, the DC component was scaled down by four in Fig. 2-4. As stated in Eq. (2.6), all of the even DCT coefficients are zeros (except k=0). The odd DCT coefficients are proportional to the difference $(a\text{-}b)$ with the magnitude almost inversely proportional to frequency k

because $\sin(k\pi/4N)$ can be approximated by $k\pi/4N$ (when $k\pi/4N$ is small). The above analysis suggests that $Z(1)$ can be a good indicator for the strength of blocking artifacts because it is proportional to the abrupt change between $x(m)$ and $y(m)$, and it has the largest amplitude of all the non-DC coefficients.

Eq. (2.6) is derived based on the assumption that the N points within each block have the same pixel value (a or b). When the compression is not so high or noise is introduced,

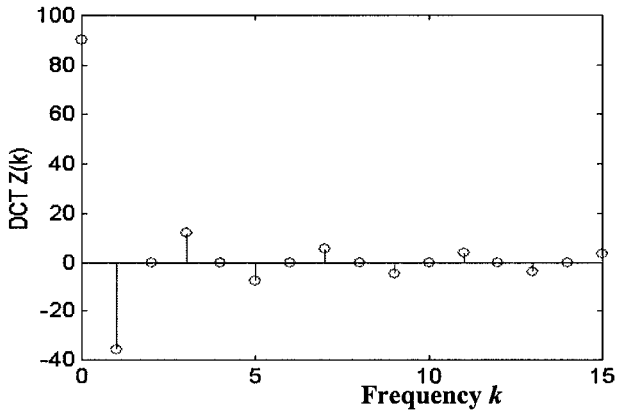


Figure 2-4. DCT coefficients of $z(n)$ where $x(n)$=80 and $y(n)$=100.

$$Z(k)=\begin{cases}\sqrt{\dfrac{N}{2}}(a+b) & k=0\\ 0 & k=2,4,6,\ldots,2N-2\\ \dfrac{(-1)^{\frac{k-1}{2}}(a-b)}{2\sqrt{N}\sin\dfrac{k\pi}{4N}} & k=1,3,5,\ldots,2N-1\end{cases} \tag{2.6}$$

there could be some small variation within each block. In this case, $z(n)$ can be modeled as Eq. (2.7).

$$z(n)=\begin{cases} a+\Delta_a(n) & |\Delta_a(n)|\le T_a \quad \sum_{n=0}^{N-1}\Delta_a(n)=0 & 0\le n<N \\ b+\Delta_b(n-N) & |\Delta_b(n-N)|\le T_b \quad \sum_{n=0}^{N-1}\Delta_b(n)=0 & N\le n<2N \end{cases} \tag{2.7}$$

where a and b are the N-point mean values of $x(m)$ and $y(m)$, respectively, and $\Delta_a(m)$ and $\Delta_b(m)$ represent the small variation ($0 \le m < N$). T_a and T_b are the upper limits of $\Delta_a(m)$ and $\Delta_b(m)$, respectively. Also, we should note the two implicit conditions in Eq. (2.7), which are $\sum_{m=0}^{N-1}\Delta_a(m)=0$ and $\sum_{m=0}^{N-1}\Delta_b(m)=0$ since a and b are defined as mean values.

Plugging Eq. (2.7) into Eq. (2.5) and setting k=1, we have

$$Z(1)=\frac{a-b}{2\sqrt{N}\sin\frac{\pi}{4N}}+\frac{1}{\sqrt{N}}\sum_{m=0}^{N-1}\Delta_a(m)\cos\frac{(2m+1)\pi}{4N}+\frac{1}{\sqrt{N}}\sum_{m=0}^{N-1}\Delta_b(m)\cos\left[\frac{(2m+1)\pi}{4N}+\frac{\pi}{2}\right] \tag{2.8}$$

The first term of Eq. (2.8), hereinafter called $Z_{a-b}(1)$, is the contribution of the abrupt change (a-b) to $Z(1)$. The second term, hereinafter called $Z_{\Delta a}(1)$, represents the contribution of $\Delta_a(m)$ while the last term, hereinafter called $Z_{\Delta b}(1)$, comes from the contribution of $\Delta_b(m)$.

Since $\cos\frac{(2m+1)\pi}{4N}$ decreases with m ($0 \le m < N$), it can be proven that $|Z_{\Delta a}(1)|$ reaches its maximum if and only if $\Delta_a(m)$ is given as either

$$\Delta_a(m) = \begin{cases} T_a, & 0 \le m < \dfrac{N}{2} \\ -T_a, & \dfrac{N}{2} \le m < N \end{cases}$$

or **(2.9)**

$$\Delta_a(m) = \begin{cases} -T_a, & 0 \le m < \dfrac{N}{2} \\ T_a, & \dfrac{N}{2} \le m < N \end{cases}$$

So, we have the upper bound on $|Z_{\Delta a}(1)|$ as

$$|Z_{\Delta a}(1)| \le \frac{1}{\sqrt{N}} \left[\sum_{m=0}^{\frac{N}{2}-1} T_a \cos\frac{(2m+1)\pi}{4N} - \sum_{m=\frac{N}{2}}^{N-1} T_a \cos\frac{(2m+1)\pi}{4N} \right] \quad \textbf{(2.10)}$$

Eq. (2.10) can be simplified as

$$|Z_{\Delta a}(1)| \le \frac{(\sqrt{2}-1)T_a}{2\sqrt{N}\sin\dfrac{\pi}{4N}} \quad \textbf{(2.11)}$$

Similarly, we can prove that

$$|Z_{\Delta b}(1)| \le \frac{(\sqrt{2}-1)T_b}{2\sqrt{N}\sin\dfrac{\pi}{4N}} \quad \textbf{(2.12)}$$

Based on Eqs. (2.11) and (2.12), the total contribution of $\Delta_a(m)$ and $\Delta_b(m)$ is limited by Eq. (2.13).

$$\left|Z_{\Delta a}(1)+Z_{\Delta b}(1)\right| \le \frac{(\sqrt{2}-1)(T_a+T_b)}{2\sqrt{N}\sin\frac{\pi}{4N}} \tag{2.13}$$

Therefore, we can conclude from Eqs. (2.8) and (2.13) that the contributions of $\Delta_a(m)$ and $\Delta_b(m)$ can be ignored when the following condition is met:

$$|a-b| >> (\sqrt{2}-1)(T_a+T_b) \tag{2.14}$$

Eq. (2.14) shows that when the small variation within each block is far less than the difference between DC components, $Z(1)$ is a good indicator for the strength of blocking artifacts. The threshold for ">>" should be set such that the second and the third terms in Eq. (2.8) become negligible compared to the first term. Note that Eq. (2.14) is always satisfied in the case of two uniform (T_a=T_b=0) blocks with an abrupt change across their boundary.

2.3.2 PROPOSED BLOCKING ARTIFACT METRIC FOR RECONSTRUCTED IMAGES

Based on the above frequency analysis of blocking artifacts, we propose the blocking artifact metric for the reconstructed images as follows:

1) Find those horizontal block pairs $z(n)$ satisfying Eq. (2.14).

2) If $|a-b| > 2*QP$, do not process this block pair since the boundary must contain a strong edge.

3) For all those horizontal block pairs left after 2), compute their first DCT coefficient $Z_i(1)$, which reflects the strength of blocking artifacts.

4) The horizontal blocking artifact value (BAV_h) is given by Eq. (2.15)

$$BAV_h = 2\sqrt{N}\sin(\pi/4N)\sqrt{\frac{\sum_i Z_i^2(1)}{image\_size}} \tag{2.15}$$

where *image_size* is the product of width and height of the image. To evaluate BAV in images of different sizes, we should normalize the summation by *image_size*. $2\sqrt{N}\sin(\pi/4N)$ is a constant that appears in the denominator of $Z_i(1)$ in Eq. (6) and is equal to 0.554 for N=8. Since it is only a scaling factor, it does not affect the relative BAV results between images.

5) Compute the vertical blocking artifact value (BAV_v) in a similar way. The BAV of the whole image is the average of BAV_h and BAV_v. The bigger the result, the worse the blocking artifacts.

2.3.3 PROPOSED BLOCKING ARTIFACT METRIC FOR POSTPROCESSED IMAGES

The metric introduced in Section 2.3.2 assumes that the blocking artifacts mainly come from those block pairs that satisfy Eq. (2.14). This is generally true for the reconstructed image because blocking artifacts in the reconstructed image usually involve two neighboring homogeneous blocks with abrupt changes at the boundary. However, after postprocessing, the pixels within each block are no longer homogeneous in most cases. To evaluate the blocking artifacts in such postprocessed images, we propose to use the

reconstructed image as the reference image. The reconstructed image, which is already available at the decoding/postprocessing end, is used to extract those block pairs with blocking artifacts. Since the increased variation within each block can significantly reduce the visibility of blocking artifacts, the blocking artifact value of postprocessed blocks should decrease as $(T_a+T_b)/|a-b|$ increases. The blocking artifact metric to compare the postprocessed images is as follows:

1) Use the reconstructed image to find those horizontal block pairs satisfying Eq. (2.14).

2) Strong edges are excluded if $|a-b| > 2*QP$ is met.

3) For all those horizontal block pairs left after 2), compute $Z_i(1)$ and its weight w_i for all the postprocessed images from the same reconstructed image. The weight w_i is given by Eq. (2.16). If the variation within each block is small compared to the DC difference, then the weight is bigger. On the contrary, if the variation is comparatively big, then the weight is small.

$$w_i = \begin{cases} \max\left\{1 - \dfrac{(\sqrt{2}-1)(T_a+T_b)}{|a-b|}, 0\right\} & a \neq b \\ 0 & a = b \end{cases} \qquad \textbf{(2.16)}$$

4) The horizontal blocking artifact value (BAV_h) is given by Eq. (2.17):

$$BAV_h = 2\sqrt{N}\sin(\pi/4N)\sqrt{\frac{\sum_i w_i Z_i^2(1)}{image\_size}} \qquad \textbf{(2.17)}$$

5) Similarly, the BAV of the whole image is the average of BAV_h and BAV_v. The smaller, the better.

Note that the metric to compute the BAV of the reconstructed images (Section 2.3.2) can be considered as a special case of the metric for the postprocessed images, where $T_a=T_b=0$ and hence $w_i=1$.

2.4 EXPERIMENTAL RESULTS AND DISCUSSION

Our proposed de-blocking algorithm and blocking artifact metric have been tested with several still images and images from video sequences. Figure 2-5 shows six test images: "LENA," "BARBARA," "MANDRILL," "FOREMAN," "TENNIS," and "GARDEN." "LENA" and "FOREMAN" have many smooth areas while "BARBARA," "MANDRILL," and "GARDEN" include a lot of high-frequency textures. In addition to smooth areas and textures, "TENNIS" includes some artificial strong edges, which we intentionally introduced at the block boundaries. In addition, we fixed the value of *QP* at 16 for all the images in the experiment because the results of our proposed algorithm and metric are not sensitive to *QP*.

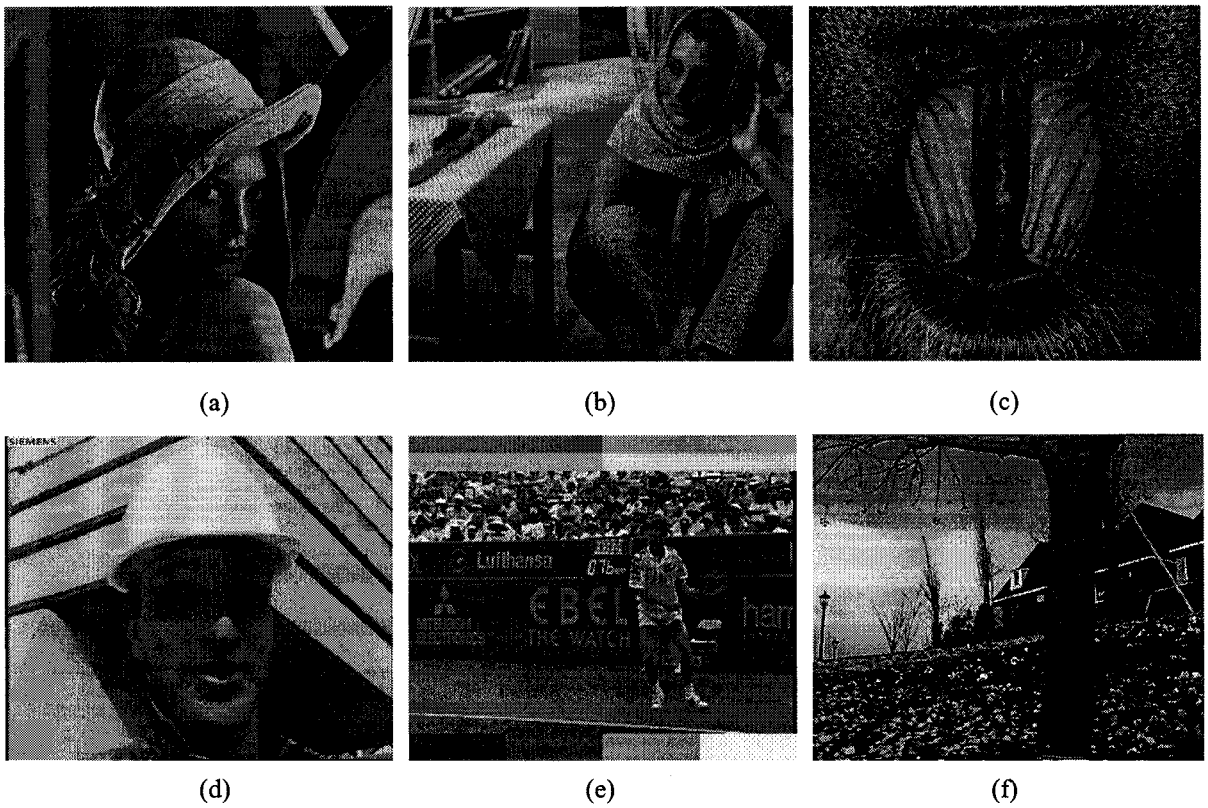
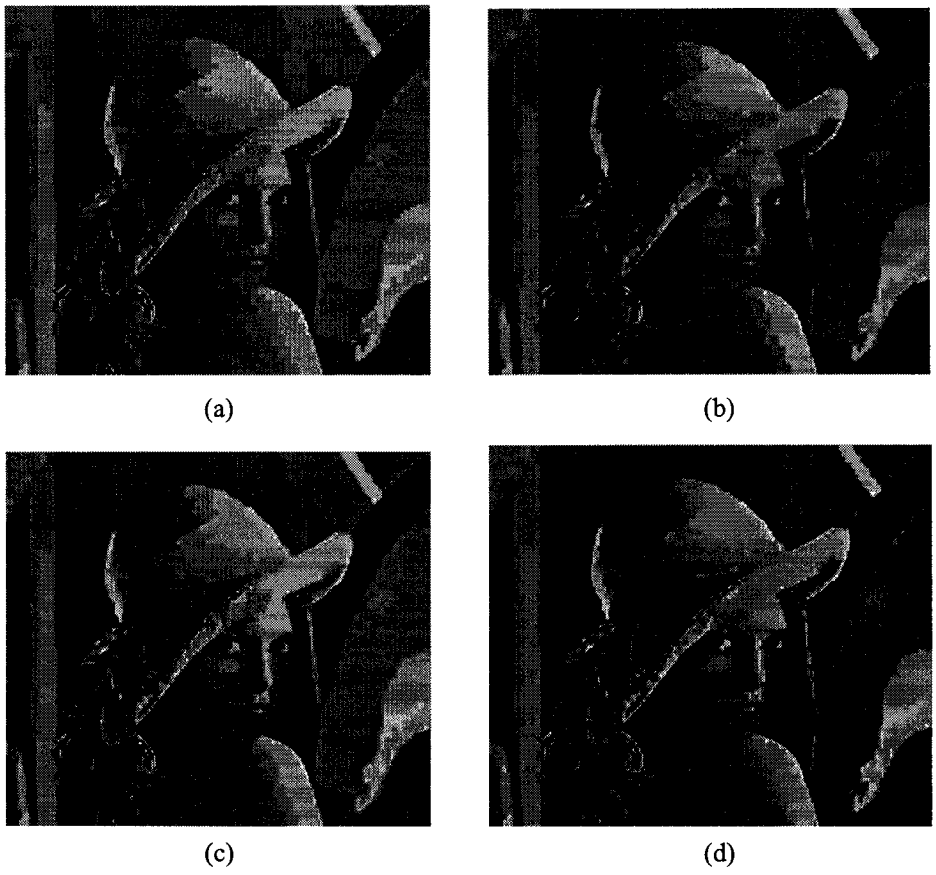


Figure 2-5. Test images in our experiments. (a) "LENA" (512×512). (b) "BARBARA" (512×512). (c) "MANDRILL" (512×480). (d) "FOREMAN" (352×288). (e) "TENNIS" (352×288). (f) "GARDEN" (352×240).

2.4.1 RESULTS OF ADAPTIVE DE-BLOCKING

Figure 2-6 shows the de-blocked results on "LENA". Figure 2-6(a) shows the reconstructed JPEG-compressed image at quality=1 via cjpeg. Figure 2-6(b) is the result after de-blocking by using the Park and Lee's algorithms (PL) [76]. Figure 2-6(c) shows the result by using the Shen and Kuo's algorithm (SK) [84]. Figure 2-6(d) presents the result from our de-blocking algorithm. As shown in Fig. 2-6, the PL and SK algorithms can reduce the blocking artifacts by smoothing a few pixels around the block boundary. However, after de-blocking, we can still find the discontinuous patches around the shoulder, face, and hat areas. In contrast, the image by our de-blocking algorithm looks

smoother. This is mainly because the number of pixels to be updated and the filter length in our proposed algorithm are adaptive to the number of connected blocks in a relatively homogeneous region.



(a) (b)

(c) (d)

Figure 2-6. The de-blocked results on "LENA" image. (a) The reconstructed "LENA" at quality=1 via cjpeg. (b) Result by the PL algorithm. (c) Result by the SK algorithm. (d) Result by the proposed algorithm.

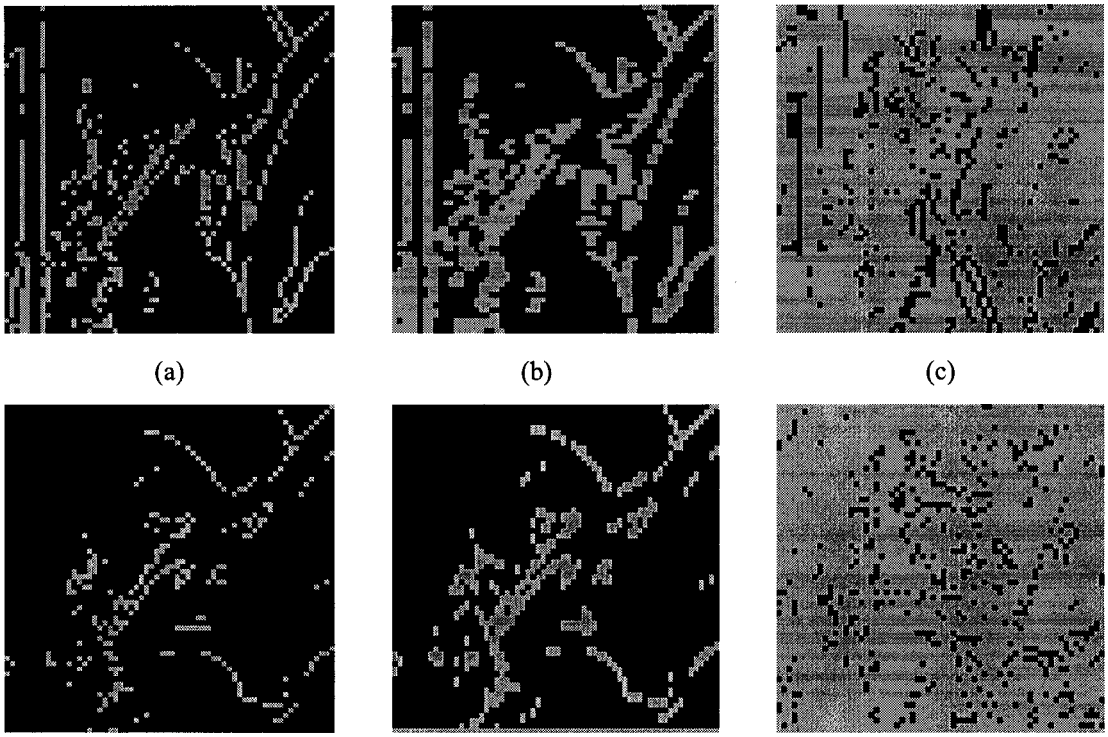
Figure 2-7 compares the number of blocks extracted for horizontal and vertical de-blocking by each algorithm. The black areas represent those blocks extracted to be de-blocked while the gray areas are not processed. The PL algorithm and SK algorithm marked about 90% (3,501 horizontal blocks and 3,841 vertical blocks) and 80% (2,900

horizontal blocks and 3,522 vertical blocks) for de-blocking, respectively. While most of marked blocks by the PL and SK algorithms are located within the same uniform region as their neighboring blocks, our algorithm only extracts those block pairs that are located in the boundary of two different regions. As a result, only about 13% (603 horizontal blocks and 476 vertical blocks) blocks are identified, which greatly reduces the computational cost in de-blocking. To save computational cost further, our semaphore extraction algorithm uses 2 rows/columns to approximate the whole block. Compared to the results by using all 8 rows/columns, more than 95% of the extracted blocks (averaged over 24 images) are the same. Also, the PSNR difference between the de-blocked results is less than 0.01 dB.

Figure 2-8 compares the results of the same algorithms on "BARBARA." Figure 2-8(a) shows the reconstructed JPEG-compressed image at quality=10 via cjpeg. Figures 2-8(b), (c), and (d) show the de-blocked results by PL, SK, and our de-blocking algorithm, respectively. Since "BARBARA" has many high-frequency textures, we do not wish to blur them via de-blocking. Compared to the SK and our proposed algorithms, the PL algorithm blurs the textures around the tablecloth, hanging scarf, shin areas, etc. Our semaphore extraction algorithm is based on the relationship between the maximum variation in each block and the abrupt change along the block boundary. Since the maximum variation inside texture blocks is usually comparable to or even greater than the difference between boundary pixels, these blocks are not marked for de-blocking. In addition, our results look smoother than the others in the floor areas, where the blocking artifacts are quite noticeable.

The results on "TENNIS" are given in Fig. 2-9 for evaluating the ability of preserving the strong edges. Figure 2-9(a) shows the reconstructed JPEG-compressed image at

quality=10 via cjpeg. Since the edges are exactly located along the block boundary in the original image, the reconstructed image still keeps them. Figures 2-9(b) and (c) show the results de-blocked by the PL and SK algorithms. Both of them have blurred the strong edges after de-blocking. In contrast, our proposed algorithm does not touch the edges because the abrupt change is bigger than $2*QP$, which means that they are not resulting from blocking effect [69]. We can also observe that our algorithm outperforms the others in de-blocking the playground areas.



(a) (b) (c)

(d) (e) (f)

Figure 2-7. Comparison of extracted semaphores (black represents the marked areas for de-blocking). (a) Horizontal semaphores by the PL algorithm (3,501). (b) Horizontal semaphores by the SK algorithm (2,900). (c) Horizontal semaphores by the proposed algorithm (603). (d) Vertical semaphores by the PL algorithm (3,841). (e) Vertical semaphores by the SK algorithm (3,522). (f) Vertical semaphores by the proposed algorithm (476).

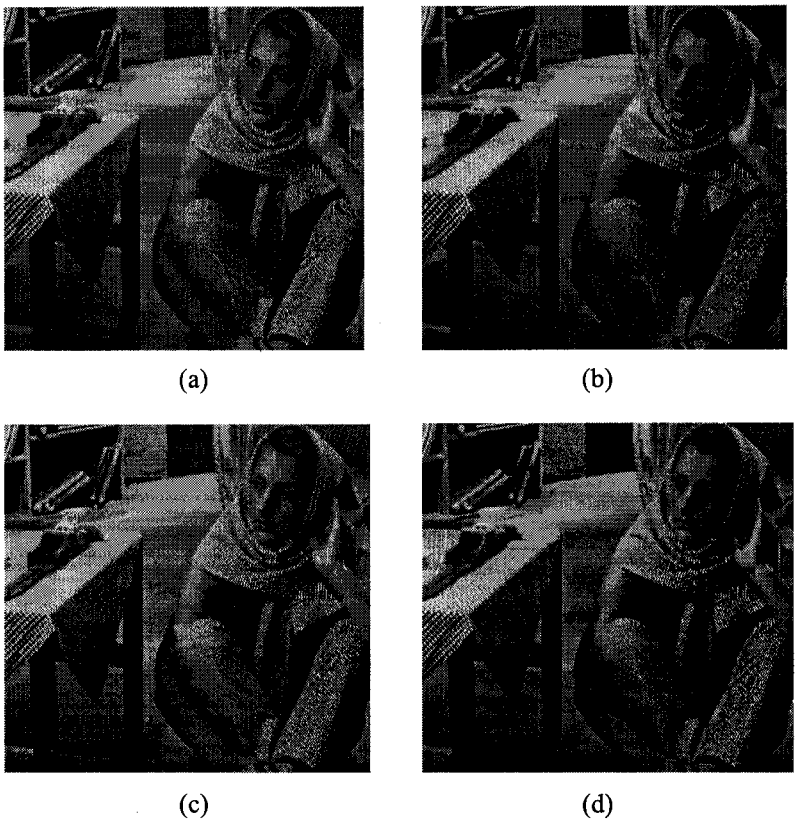
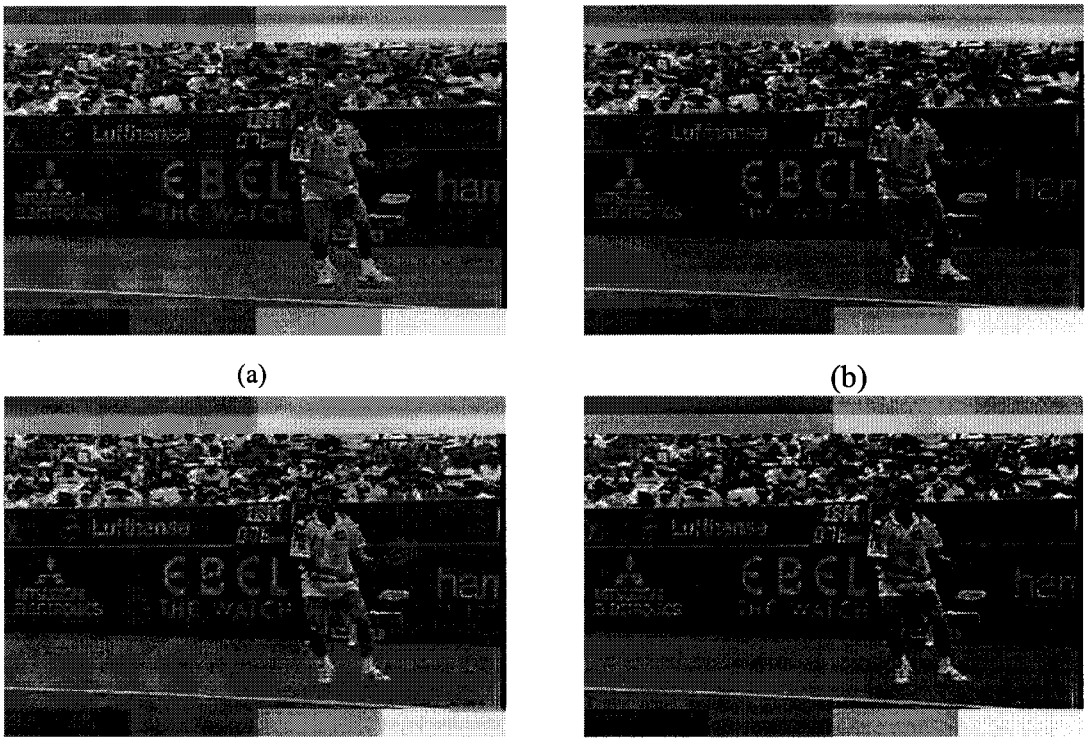


Figure 2-8. The de-blocked results on "BARBARA" image. (a) The reconstructed "BARBARA" at quality=10 via cjpeg. (b) Result by the PL algorithm. (c) Result by the SK algorithm. (d) Result by the proposed algorithm.

The test results of the six images are summarized in Table 2-1 for four different compression qualities (quality 100 means the original image and quality 1 means the worst case). The 24 reconstructed images were de-blocked by the PL, SK, and proposed algorithms. In addition to the BAV, Table 2-1 also gives the PSNR of each postprocessed image for comparison. According to our proposed blocking artifact metric introduced in Section 2.3.3, our de-blocking algorithm achieves the best results in all cases. In addition, we should mention that our de-blocking algorithm did not process the

blocks with strong edges and textures because if needed, those blocks can be processed by de-ringing algorithms [1][3][29][88] to smooth out the perturbations around the edges, which is included in Chapter 3.



(a) (b)

(c) (d)

Figure 2-9. The de-blocked results on "TENNIS" image. (a) The reconstructed "TENNIS" at quality=10 via cjpeg. (b) Result by the PL algorithm. (c) Result by the SK algorithm. (d) Result by the proposed algorithm.

Table 2-1. PSNR and BAV of the reconstructed images and the postprocessed images by the PL, SK, and proposed algorithms.

		Reconstructed		PL		SK		Proposed	
		PSNR (dB)	BAV	PSNR (dB)	BAV	PSNR (dB)	BAV	PSNR (dB)	BAV
LENA	Q=25	32.52	**0.45**	32.36	**0.32**	32.59	**0.32**	32.46	**0.29**
	Q=10	29.57	**1.42**	29.72	**1.03**	29.77	**1.03**	29.82	**0.86**
	Q=5	26.78	**2.56**	27.22	**1.85**	27.19	**1.81**	27.24	**1.43**
	Q=1	24.11	**4.14**	24.88	**3.05**	24.79	**2.76**	24.66	**2.41**
BARBARA	Q=25	29.31	**0.33**	28.36	**0.24**	29.33	**0.24**	29.26	**0.22**
	Q=10	25.70	**1.22**	25.22	**0.90**	25.76	**0.89**	25.77	**0.79**
	Q=5	23.86	**2.51**	23.74	**1.88**	24.07	**1.85**	24.09	**1.57**
	Q=1	22.10	**4.66**	22.46	**3.48**	22.68	**3.15**	22.49	**2.82**
MANDRILL	Q=25	25.55	**0.11**	25.41	**0.07**	25.54	**0.08**	25.51	**0.07**
	Q=10	23.13	**0.87**	23.01	**0.62**	23.15	**0.63**	23.14	**0.55**
	Q=5	21.29	**2.27**	21.31	**1.57**	21.37	**1.53**	21.36	**1.34**
	Q=1	19.73	**4.76**	20.02	**3.15**	20.03	**2.84**	19.90	**2.61**
FOREMAN	Q=25	32.71	**0.40**	32.69	**0.27**	32.78	**0.28**	32.66	**0.26**
	Q=10	29.48	**1.31**	29.76	**0.94**	29.65	**0.93**	29.73	**0.79**
	Q=5	26.58	**2.51**	27.04	**1.81**	26.96	**1.76**	27.03	**1.49**
	Q=1	23.61	**4.79**	24.65	**3.53**	24.52	**3.15**	24.23	**2.73**
TENNIS	Q=25	31.19	**0.40**	30.06	**0.34**	30.41	**0.32**	31.21	**0.28**
	Q=10	27.32	**1.00**	26.67	**0.77**	26.99	**0.73**	27.39	**0.67**
	Q=5	24.21	**1.65**	24.03	**1.18**	24.14	**1.16**	24.33	**0.99**
	Q=1	21.76	**3.21**	21.86	**2.23**	21.87	**2.10**	21.75	**2.02**
GARDEN	Q=25	24.69	**0.25**	24.51	**0.19**	24.69	**0.20**	24.68	**0.18**
	Q=10	21.82	**1.01**	21.70	**0.74**	21.84	**0.74**	21.84	**0.64**
	Q=5	20.06	**2.21**	20.00	**1.61**	20.13	**1.57**	20.12	**1.37**
	Q=1	18.53	**4.07**	18.66	**2.81**	18.71	**2.54**	18.67	**2.24**

2.4.2 RESULTS OF PROPOSED BLOCKING ARTIFACT METRIC

As stated in Eq. (2.14), the first DCT coefficient represents the strength of blocking artifacts when the abrupt change is far greater (>>) than the variation inside each block. In this simulation, only those block pairs with T_a=T_b=0 and $|a-b|$>0 are included in BAV computation [in Steps 2.3.2 (1) and 2.3.3(1)] so that Eq. (2.14) is always satisfied. Table 2-1 shows the BAV and PSNR of the reconstructed images and postprocessed images by different algorithms. From Table 2-1, as the compression quality decreases from 25 to 10, 5, and 1, the BAV of each reconstructed image increases significantly. Since "MANDRILL" is rich with high-frequency textures, its blocking artifacts at quality 25 are hardly visible. As the compression quality decreases from 25 to 1, many high-frequency components in the comparatively smooth areas (e.g., areas around the nose) are removed. As a result, the BAV increases sharply by more than 40 times. Similar results are observed in the texture-rich "BARBARA" and "GARDEN" images. "LENA" and "FOREMAN", on the contrary, have many smooth areas. Their blocking artifacts at quality 25 are visible. As the quality decreases, the blocking artifacts become more severe. "TENNIS" has some manually added strong edges in the top and bottom areas. However, these strong edges were not counted as blocking artifacts because the edges are stronger than 2**QP*. In addition, the BAV of each original image (reconstructed at quality = 100) is zero except "FOREMAN", which has a BAV of 0.04. This is because a few uniform block pairs with weak edges around the top of the white hat area in "FOREMAN" were counted as blocking artifacts while no such uniform block pairs were detected in the other five original images.

As shown in Table 2-1, the PSNR of the postprocessed images by the PL, SK, and our de-blocking algorithms is quite similar except for "TENNIS", where the SK and PL

algorithms get much worse results because they blur the manually added strong edges. In general, the PSNR of the postprocessed images is equal to or a little better than that of the corresponding reconstructed image. However, we do find several cases where the PSNR of the postprocessed image is even worse than that of the reconstructed image. If we used the PSNR as an indicator of blocking artifacts, the decreased PSNR would imply that blocking artifacts have become more severe after postprocessing, which obviously contradicts our subjective assessment. By applying our proposed blocking artifact metric, the BAV of each postprocessed image has been reduced by 20%~45% over that of the reconstructed image. Figure 2-10 compares the BAV of each reconstructed image and its postprocessed images. In terms of BAV, all of the postprocessed images are better than the reconstructed image and our proposed algorithm achieves significant improvement over the PL and SK algorithms, which is consistent with our subjective assessment.

As shown in Fig. 2-6, the postprocessed "LENA" images by the PL and SK algorithms still look blocky in the shoulder, hat, and face areas. However, their PSNRs are a little better than ours. By applying our proposed blocking artifact metric, the BAV of our result is about 20% less than that of the PL-processed image and 10% less than that of the SK-processed image. We should mention that the 10% or 20% improvement is the average value over the entire image while this improvement can be quite large in some local areas. As an example, Fig. 2-11 compares the results in a shoulder area where the BAVs of the reconstructed image, the PL result, and the SK result are 156%, 80%, and 60% larger than that of ours, respectively.

Also, BAV can be presented in horizontal (BAV_h) and vertical (BAV_v) directions separately. For the images in Fig. 2-11(a) ~ 11(d), their horizontal BAV and vertical BAV (BAV_h, BAV_v) are (5.54, 4.04), (4.12, 2.62), (3.70, 2.31), and (2.38, 1.36), respectively. The bigger BAV_h indicates that the horizontal direction has more blocking artifacts. After counting the number of block pairs with blocking artifacts in different directions, we found that the reconstructed image in Fig. 2-11(a) has 47 horizontal block pairs and only 25 vertical block pairs, which leads to the difference in BAV_h and BAV_v.

It is worth mentioning that although both BAV and blocking artifact semaphores are indicators of blockiness, they are different. Blocking artifact semaphores provide the position information on where blocking artifacts are located, but they do not offer any information on the strength of blockiness. On the other hand, BAV represents the averaged blockiness strength in an image, but it does not contain the location of blockiness. In real-time postprocessing, it is desirable for the semaphore extraction algorithm to be fast with reasonable accuracy. Therefore, our semaphore extraction algorithm in Section 2.2.1 makes some assumptions (e.g., it uses only 2 rows/columns per block) and involves simple arithmetic operations. However, our blockiness metric is designed without the real-time constraint. Therefore it involves more complex operations and uses all rows/columns in a block to be more accurate. On the other hand, since BAV represents the overall blockiness strength in an image, it can be used to evaluate different postprocessing algorithms. Also, it can be used as a metric to adjust the parameters in source encoding so that the blockiness can be maintained less than a certain level.

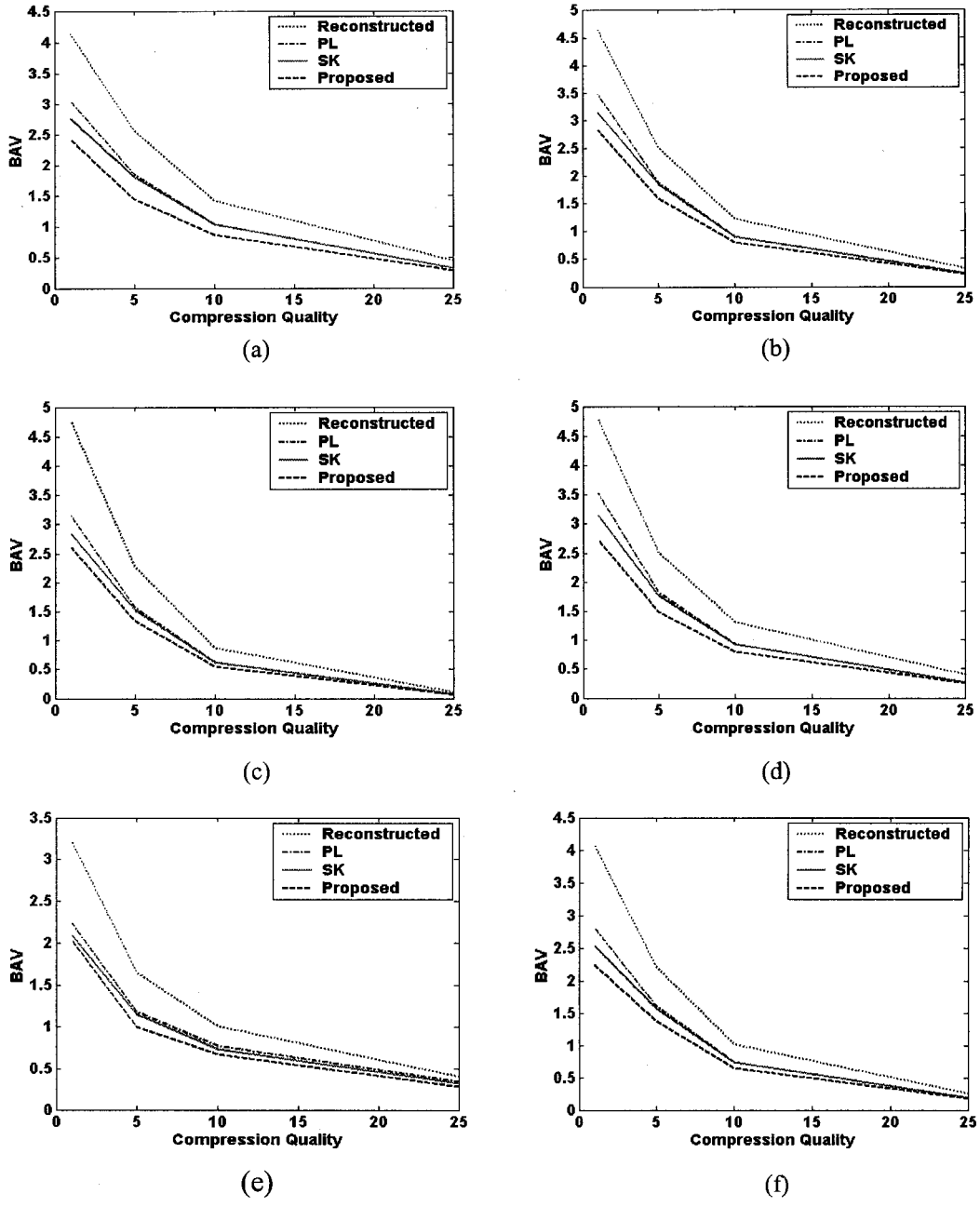


Figure 2-10. BAV comparison of de-blocked images by different algorithms. (a) "LENA." (b) "BARBARA." (c) "MANDRILL." (d) "FOREMAN." (e) "TENNIS." (f) "GARDEN."

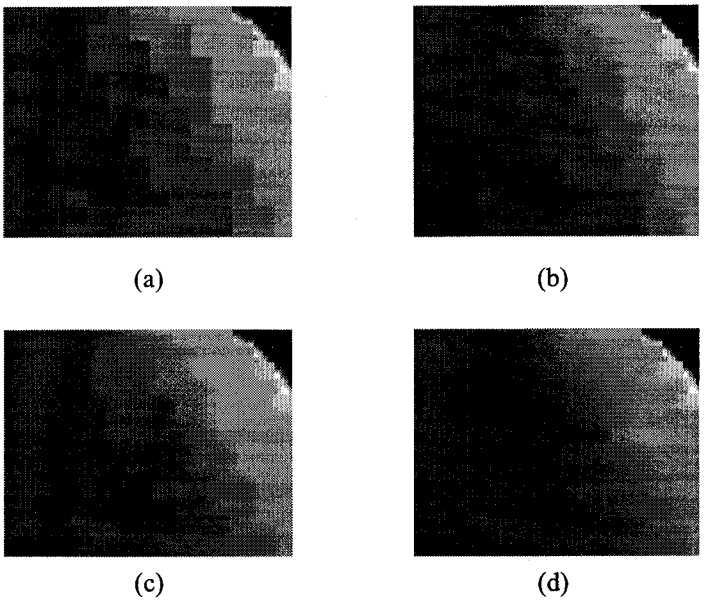


Figure 2-11. Comparison of the postprocessed shoulder area on "LENA." (a) Reconstructed BAV=4.79. (b) The PL result BAV=3.37. (c) The SK result BAV=3.00. (d) The proposed result BAV=1.87.

2.5 CONCLUSIONS

We have proposed an adaptive de-blocking algorithm and a new blocking artifact metric for block DCT-based compressed images. Our de-blocking method can efficiently and effectively reduce the blocking artifacts while keeping the strong edges untouched. The experimental results with highly compressed images have shown significant improvement over the existing algorithms, both subjectively and objectively.

Our proposed blocking artifact metrics can be used to measure blocking artifacts in the reconstructed images and evaluate different postprocessing algorithms. They do not need the original image and do not consider the strong edges as blocking artifacts. The simulation results demonstrate that our proposed metric is more consistent with subjective evaluation than the PSNR metric.

CHAPTER 3: DE-RINGING ALGORITHM

3.1 INTRODUCTION

Ringing artifacts come from the loss of high frequency details. In still images, ringing artifacts appear as spurious oscillations in the vicinity of major edges [73]. Ringing artifacts in video signals are also called mosquito noise because they appear as moving ripples and swarm around sharp edges [43]. Ringing is the dominant artifact in wavelet-based compression methods, e.g., JPEG 2000 [37]. DCT-based compression methods also generate ringing artifacts, but they are constrained within each block.

Many algorithms have been proposed for reducing the ringing artifact. Lynch *et al.* [65] proposed sending the edge information in a side channel so that it can be retrieved at the decoder side. However, this will change the existing standards and also increase the bit rate. Several authors formed ringing reduction as a regularized iterative image restoration problem [5][9][17][54][55][99][100]. Similar to de-blocking, POCS [98], MAP [57][58], and maximum likelihood (ML) [22][29][79][96] methods have also been proposed in de-ringing. The major disadvantage of these methods is their computational complexity. To reduce the computational cost, Shen and Kuo [84] proposed a nonlinear filter to replace the iterative MAP method in de-ringing. However, its computational cost is still high. Oguz *et al.* [73] proposed a morphological post-filtering in ringing reduction. However, its performance depends on an accurate segmentation of edges. Park and Lee [76] proposed using a 2-D adaptive filter in de-ringing. Basically, they classify pixels into edge pixels and non-edge pixels and then smooth the non-edge pixels while keeping the edge pixels. The problem is that ringing artifacts, in many cases, can be falsely classified as edge pixels, and therefore evade the filtering.

In this chapter, we propose a simple clustering-based de-ringing algorithm. First, we classify image blocks into smooth blocks, texture blocks, and edge blocks. Then, we keep the smooth and texture blocks unchanged and process the pixels in the edge blocks. To avoid blurring edges, only pixels that have similar gray values are smoothed.

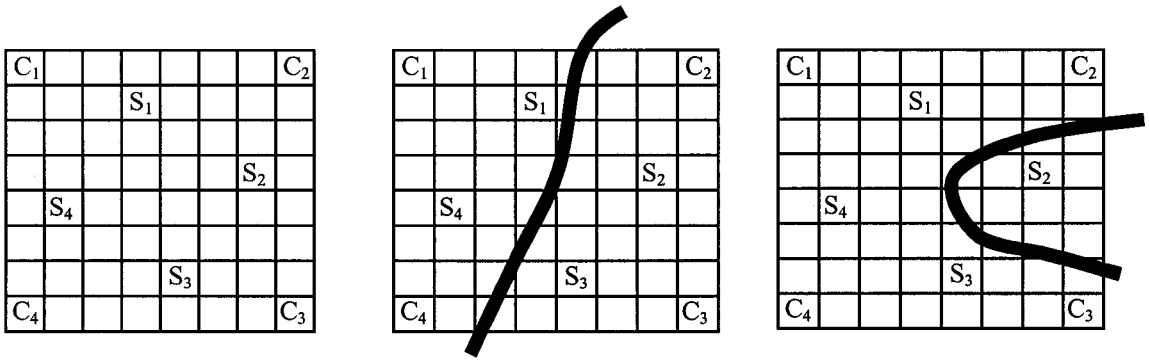
3.2 DE-RINGING ALGORITHM

Our de-ringing algorithm consists of two steps: ringing semaphore extraction and de-ringing. Semaphore extraction is an important part in de-ringing because unnecessary de-ringing can blur the texture areas and can also increase the computational cost. A good semaphore extraction algorithm only identifies those blocks that actually need de-ringing.

3.2.1 RINGING SEMAPHORE EXTRACTION

Since ringing artifacts are only noticeable in edge blocks and edge blocks should have a big dynamic range in pixel values, the first criterion to mark a block for de-ringing is that its range should be bigger than a threshold T_r. To reduce the computational cost, we only compute the range of 8 pre-determined points (C_1~C_4 and S_1~S_4) shown in Fig. 3-1(a). Since a strong edge usually separates a block into two relatively homogeneous regions, it is reasonable to assume that these 8 locations usually lead to at least one point belonging to each region. Figures 3-1(b) and 3-1(c) show two different edge scenarios. In Fig. 3-1(b), the edge enters and exits the block from two different sides. In this case, the four corner pixels (C_1~C_4) are enough to approximate the range of the whole block because C_2 and C_3 are located in a different region from C_1 and C_4. If the edge enters and exits the block from the same side as shown in Fig. 3-1(c), the four side pixels (S_1~S_4) can capture the range information. In this example, S_2 is located in a different region from the others.

Since a texture block can also have a large dynamic range, the range alone cannot reliably differentiate an edge block that should be de-ringed from a texture block where ringing artifacts would be unnoticeable. The key difference between an edge block and a texture block is that the edge block has a dominant edge along a certain direction while edge pixels in a texture block are scattered over many directions. For a texture block located within a texture area, all its 8 neighboring blocks shown in Fig. 3-2 are probably texture blocks as well. As for a block containing a dominant edge, it is unlikely that this edge can cross all its 8 neighboring blocks. Therefore, some of its neighboring blocks should be non-edge blocks.



(a) 8 selected locations (b) Edge crosses 2 sides (c) Edge crosses one side

Figure 3-1. The pixels used to approximate the range of an edge block.

Based on these observations, we propose a simple method to differentiate texture blocks from edge blocks. If the range in a given block is greater than T_r, we mark it as an edge block candidate. Then, we check the ranges of its 8 neighboring blocks. If all of them are also larger than T_r, we mark this block and its 8 neighbors as texture blocks because they are located within a texture area. The ringing semaphore algorithm can be summarized as follows:

(1) Compute the range of each block using the selected 8 pixels shown in Fig. 3-1(a).

(2) If the range is larger than T_r, mark this block as an edge block candidate. Here, T_r is set to be 2*QP.

(3) Repeat Steps (1) and (2) for all the other blocks.

(4) For each edge block candidate, check the marks of its 8 neighboring blocks. If all its 8 neighboring blocks were marked as edge block candidates, mark this block and its 8 neighbors as texture blocks. Otherwise, select this block for de-ringing.

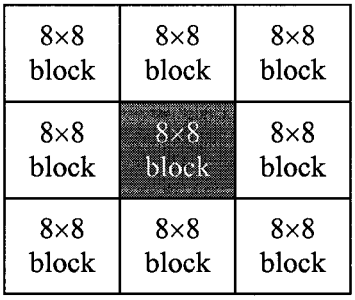


Figure 3-2. Neighboring blocks used to differentiate texture and edge blocks.

3.2.2 DE-RINGING

Ringing artifacts can increase the intensity of some pixels and decrease the intensity of others. Since we do not know whether a specific pixel value has increased or decreased, smoothing is usually applied to reduce the amount of perturbation.

To avoid blurring strong edges, we use a clustering algorithm when smoothing ringing artifacts. For each pixel $x_{i,j}$ to be processed, we define a 3×3 window centered at $x_{i,j}$ as its

neighboring pixels. Then, we check whether each neighboring pixel belongs to the same cluster as $x_{i,j}$ based on the similarity of pixel values. $x_{i,j}$ is replaced by the average value of the neighboring pixels that belong to the same cluster as $x_{i,j}$. Our de-ringing algorithm is as follows:

(1) Initialize the number (*num*) of pixels in $x_{i,j}$'s cluster to 1 and the summation (*sum*) of the pixel values in $x_{i,j}$'s cluster to $x_{i,j}$ since $x_{i,j}$ always belongs to its cluster.

(2) For each neighbor of $x_{i,j}$, if the absolute difference between the neighbor and $x_{i,j}$ is less than a threshold T_d, then this neighbor belongs to the same cluster as $x_{i,j}$. Therefore, the summation and the number of pixels are updated as: *sum* = *sum* + the neighbor's value and *num* = *num* + 1. Otherwise, this neighbor is ignored because it belongs to another cluster.

(3) Finally, $x_{i,j}$ is replaced by the mean value of its cluster, i.e., *sum*/*num*.

We set the threshold in clustering (T_d) to be QP (the quantization step size for DCT coefficients in the frequency domain). T_d should depend on QP because the higher the QP, the stronger the ringing artifacts. This correlation between the QP and the strength of ringing artifacts can be explained as follows. Assuming the quantization errors of DCT coefficients are uniformly distributed from –QP/2 to QP/2, the mean square error (MSE) of DCT coefficients after quantization is $QP^2/12$. Since DCT is a unitary and orthogonal transform, it can be proven that the MSE of pixel values obtained after inverse DCT transform (IDCT) is also $QP^2/12$. Therefore, in the sense of MSE, quantizing DCT coefficients by QP is equivalent to quantizing the pixel values by QP. The experimental results by Martin and Forchhammer [69] also showed that the quantization step size q in the spatial domain can be approximated by QP in the DCT domain. So, in most cases, quantization of DCT coefficients by QP can cause discrepancies of up to QP in pixel

values. Therefore, we assume that the small perturbation ($\leq$ QP) of pixel values in an edge block comes from ringing artifacts.

3.3 RESULTS AND DISCUSSION

Our postprocessing algorithms have been tested with various still images and video sequences at different compression quality levels. In this simulation, we fixed the value of QP at 16 for all the images and videos because the results of our algorithms are not sensitive to it. As an example, Fig. 3-3 shows the postprocessing results of the first frame in the FOREMAN sequence. Figure 3-3(a) is the original FOREMAN. Figure 3-3(b) shows the reconstructed FOREMAN image at the bit rate of 0.25 bit per pixel (bpp). The extracted horizontal blocking semaphores, vertical blocking semaphores, and ringing semaphores are given in Figs. 3-3(c), 3-3(d), and 3-3(e), respectively. The gray area represents those blocks to be de-blocked or de-ringed. The result after de-blocking and de-ringing is presented in Fig. 3-3(f).

In Fig. 3-3(b), there are many blocking artifacts in the relatively smooth areas, such as walls and the face. After de-blocking, these artifacts are visually removed. Since the common image quality metrics, such as MSE and PSNR, have a poor predictive value on the degree of blocking artifacts, we used the blockiness metric proposed in [23] to measure the blocking artifact value (BAV) for both reconstructed and postprocessed images. In Fig. 3-3, the BAV of the reconstructed FOREMAN image decreased from 1.31 to 0.79 after de-blocking (the smaller the BAV, the better). The ringing artifacts in the reconstructed image are observable around the strong edges between walls. Figure 3-3(e) shows the extracted ringing semaphores (the gray areas), which well locates the strong edges. Comparing Figs. 3-3(b) and 3-3(f), we can find that the spurious

oscillations around the wall edges are greatly reduced after de-ringing and the edges are preserved without blurring.

Figure 3-4 shows the postprocessed results of LENA, BARBARA, and GARDEN images compressed at different bit rates. For the highly compressed images, such as LENA, our de-blocking algorithm shows obvious visual improvement in image quality because it can distribute the abrupt change over two homogeneous regions adaptively. For each image, the BAV significantly decreased after de-blocking.

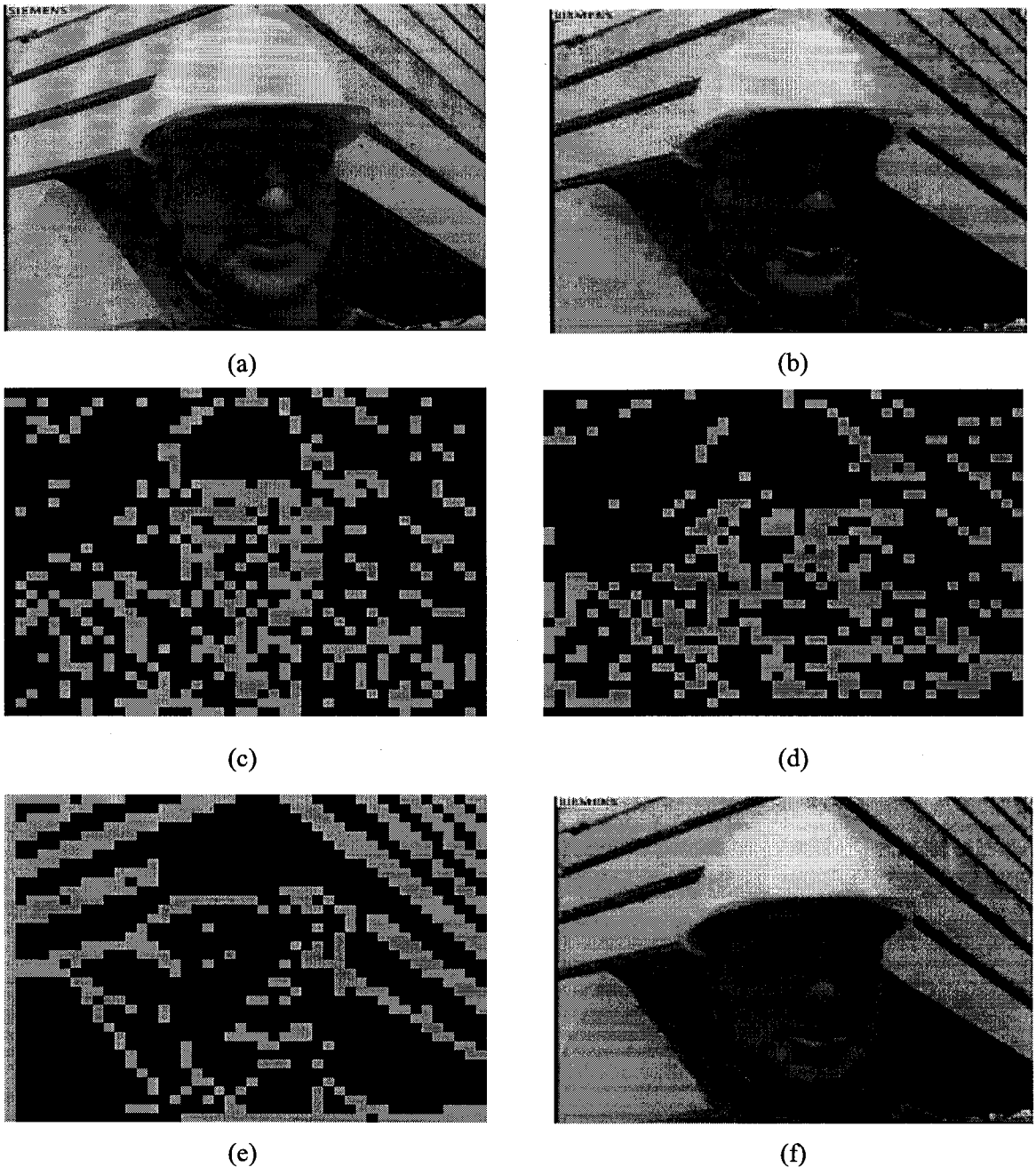
Our proposed ringing semaphore extraction algorithm is effective and efficient. Compared to the method where the range is computed using all the pixels in a block, our 8-pixel method in Section 3.2.2 correctly identified about 95% of the edge block candidates (averaged over 30 different images). Since the range approximated by 8 pixels cannot exceed that computed by all 64 pixels, there is no false detection. Also, some of the missed 5% edge block candidates belong to texture blocks, which do not need de-ringing anyway. Since our approximation only uses one eighth of the pixels in a block, the ringing semaphore extraction becomes much faster.

The method in differentiating texture and edge blocks also works well. For the FOREMAN image shown in Fig. 3-3(a), no texture blocks were detected by our algorithm. However, for the texture-rich images, such as TENNIS (352×288), MANDRILL (512×480), and GARDEN (352×240), the number of identified texture blocks is 656, 1748, and 566, respectively. Figure 3-5 shows the identified texture blocks (the white areas) for MANDRILL and TENNIS images, which corresponds well to the texture areas in the reconstructed images. For the identified texture blocks, we can either keep them unchanged or smooth them by setting a smaller threshold T_d to prevent blurring. Since texture itself has masking effect on artifacts, we finally decided to keep

them unprocessed. In this way, we can save computational cost and avoid smearing textures.

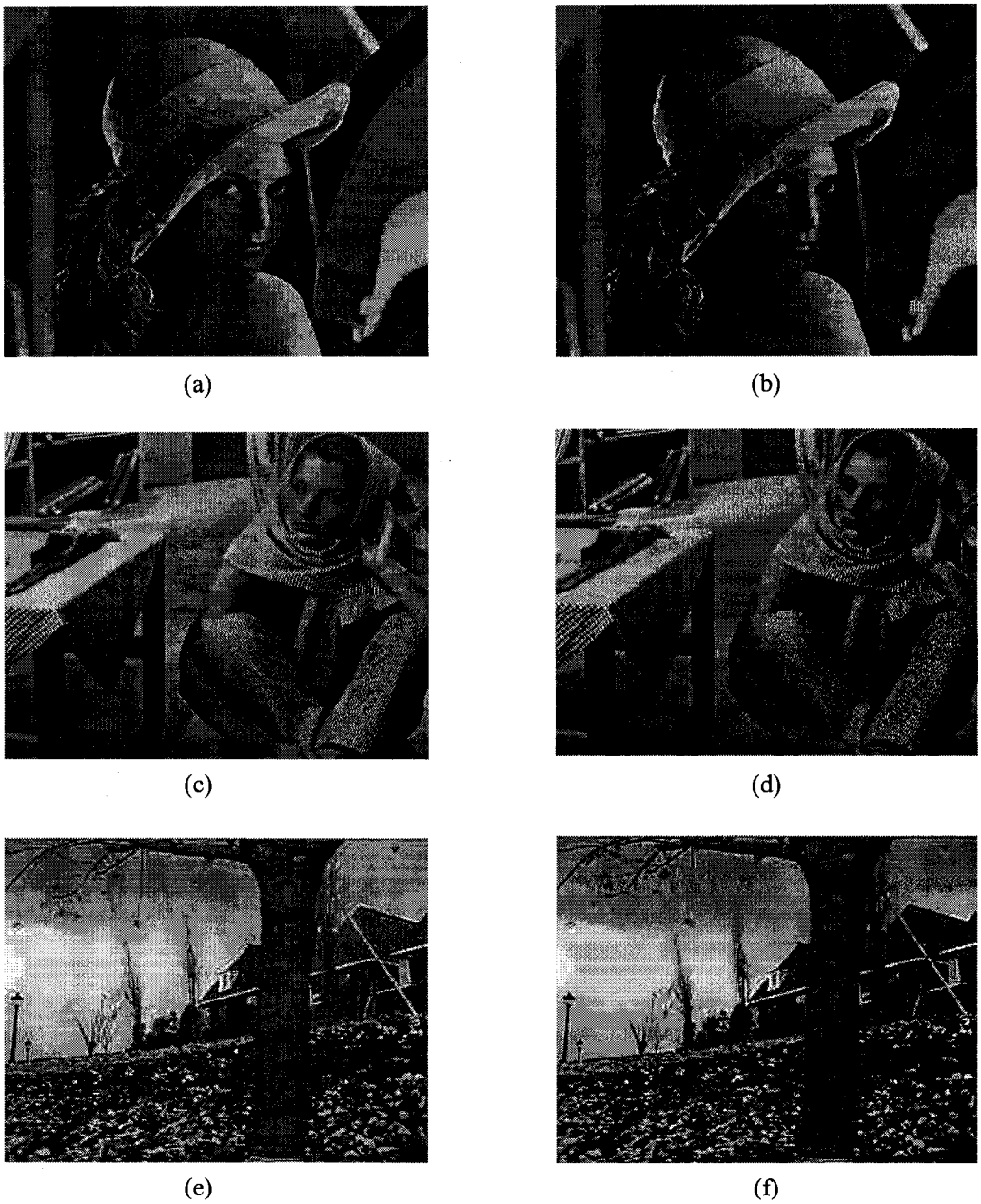
3.4 CONCLUSIONS

We have developed an efficient ringing semaphore extraction algorithm and a clustering-based de-ringing algorithm for BDCT-based compressed images and videos. Our methods can efficiently and effectively reduce the ringing artifacts while preserving the strong edges and textures. In addition to the strong ability in artifact reduction, our algorithms also have low computational cost.



(a) (b) (c) (d) (e) (f)

Figure 3-3. Postprocessing results of FOREMAN. (a) The original FOREMAN (352×288, BAV = 0.04). (b) Reconstructed FOREMAN at the bit rate of 0.25 bpp (BAV = 1.31) (c) Horizontal blocking semaphores (432). (d) Vertical blocking semaphores (408). (e) Ringing semaphores (472). (f) Result after de-blocking and de-ringing (BAV = 0.79).

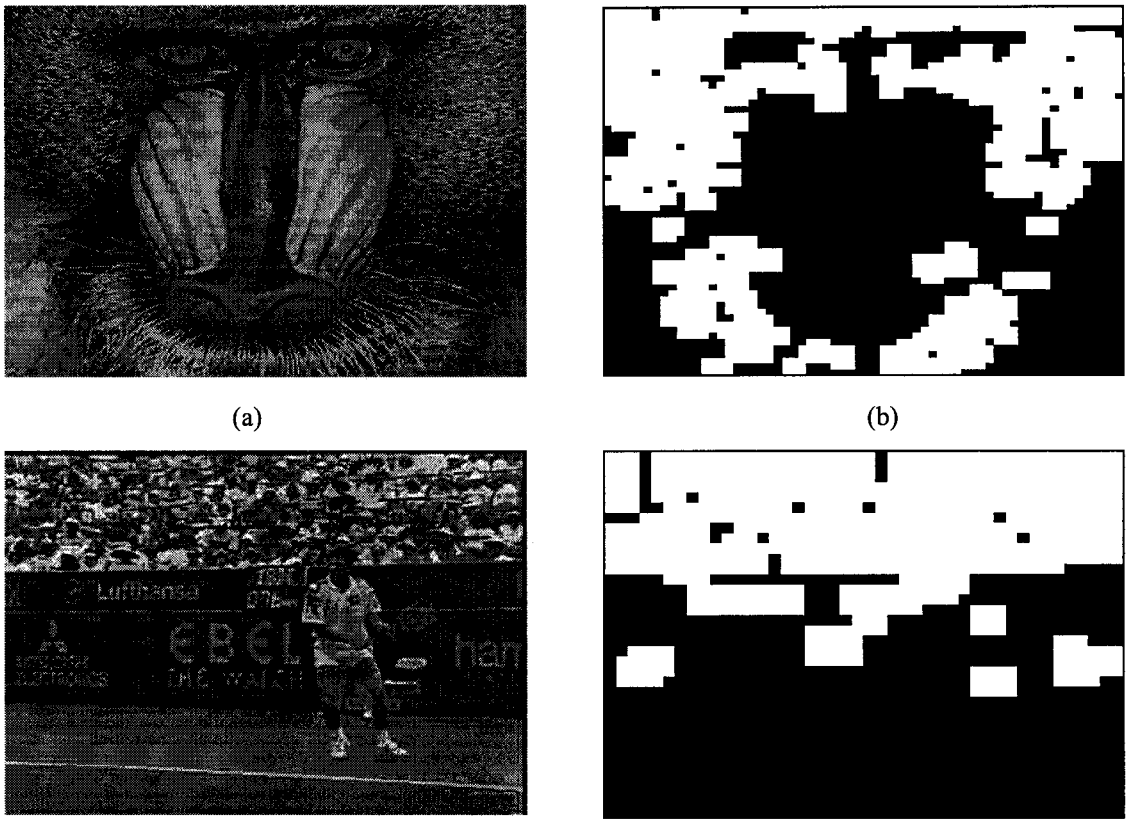


(a) (b)

(c) (d)

(e) (f)

Figure 3-4. Images before and after postprocessing. (a) Reconstructed LENA (512×512, 0.20 bpp, BAV = 2.56). (b) Postprocessed LENA (BAV = 1.57). (c) Reconstructed BARBARA (512×512, 0.35 bpp, BAV = 1.22). (d) Postprocessed BARBARA (BAV = 0.79). (e) Reconstructed GARDEN (352×240, 0.45 bpp, BAV = 1.01). (f) Postprocessed GARDEN (BAV = 0.64).



(a) (b)

(c) (d)

Figure 3-5. Identified texture blocks. (a) Reconstructed MANDRILL (512×480, 0.45bpp). (b) Identified texture blocks of MANDRILL (1,784). (c) Reconstructed TENNIS (352×288, 0.45bpp). (d) Identified texture blocks of TENNIS (656).

CHAPTER 4: IMPLEMENTATION ON MEDIAPROCESSORS

4.1 INTRODUCTION

In Chapters 2 and 3, we have proposed de-blocking and de-ringing algorithms. Our proposed algorithms not only reduce compression artifacts effectively, but also have low computational cost. Ideally, they should be able to run on inexpensive processors for real-time postprocessing. Mediaprocessors are optimized to efficiently handle a large amount of computation and data required by multimedia applications, such as image and video processing. They typically have a superscalar or very long instruction word (VLIW) architecture with multiple execution units. High computing power in mediaprocessors is provided by parallelisms at both instruction and data levels. Instruction-level parallelism allows multiple execution units to execute their own operations in a single clock cycle while data-level parallelism allows multiple data to be processed in a single operation. Some mediaprocessors, e.g., Hitachi/Equator Technologies MAP-CA [16] and Texas Instruments TMS320C64x [89], contain a direct memory access (DMA) controller for efficient handling of data flow. In this Chapter, we present how we have efficiently implement our proposed de-blocking and de-ringing algorithms on MAP-CA for real-time de-blocking and de-ringing.

In the remainder of this chapter, Section 4.2 describes the implementation on MAP-CA. The hardware performances are presented and discussed in Section 4.3. Section 4.4 shows the graphic user interface (GUI) for the hardware demonstration. Finally, we summarize our conclusions in Section 4.5.

4.2 IMPLEMENTATION ON A MEDIAPROCESSOR

To demonstrate that our proposed algorithm is suitable for real-time postprocessing, we mapped it on a mediaprocessor called MAP-CA.

4.2.1 Architecture of the MAP-CA mediaprocessor

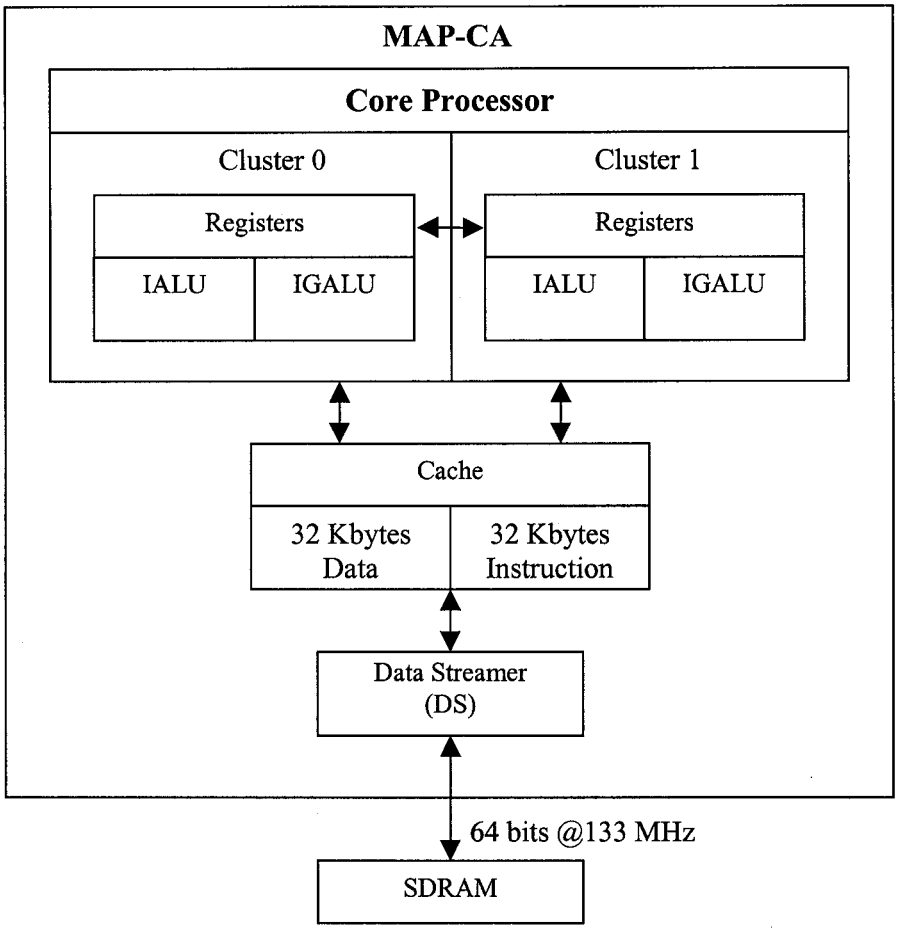


Figure 4-1. MAP-CA architecture.

Figure 4-1 shows a simplified block diagram of the MAP-CA architecture. MAP-CA has two major components: the core processor and the programmable DMA controller called Data Streamer (DS). The core processor consists of two clusters. Each cluster has an Integer Arithmetic Logic Unit (IALU) and an Integer Graphics Arithmetic Logic Unit (IGALU). The IALU can perform either 32-bit fixed-point arithmetic operations or 64-bit load/store operations while the IGALU can perform 64-bit partitioned arithmetic operations. Many IGALU instructions can specify data partitions of 32, 16, or 8 bits, providing a performance improvement by a factor of approximately 2, 4, or 8, respectively, via data-level parallelism. The IGALU also supports powerful instructions for filtering operations used extensively in image and video applications. For example, a single *inner-product* instruction can perform eight 16-bit multiplications in parallel, summing the results into a 32-bit output [68]. The two clusters together are capable of executing four different instructions (e.g., two on IALUs and two on IGALUs) in each clock cycle.

In addition to the powerful core processor and instruction set, the MAP-CA has a 32-kbyte 4-way set-associative data cache, a 32-kbyte 2-way set-associative instruction cache, and the DS. Since the DS can transfer data between on-chip (data cache) and off-chip memory in parallel with the processing performed by the clusters, high performance can be achieved by overlapping the data transfers with the computation as much as possible. The processor currently runs at 450 MHz and the external memory is a 133-MHz SDRAM.

4.2.1 DATA FLOW MANAGEMENT OF VIDEO POSTPROCESSING ON MAP-CA

Our postprocessing algorithms involve three passes: (1) vertical blocking semaphore extraction and vertical de-blocking, (2) horizontal blocking semaphore extraction,

horizontal de-blocking, and range computation of each block, and (3) ringing semaphore extraction and de-ringing. Since the on-chip data cache is limited in size (32 Kbytes for the MAP-CA) and cannot hold the entire image, an image slice (e.g., N rows or columns) is processed at a time.

In the first pass, a vertical slice of the decompressed image is first loaded from the external memory to the on-chip cache by the DS. Then, the vertical blocking semaphore (VBS) extraction algorithm is used to determine which blocks will be vertically de-blocked. After vertical de-blocking, the intermediate results are stored in an external memory ($IMAGE_V$).

The second pass is to extract horizontal blocking semaphores (HBS), de-block the intermediate image horizontally, and compute the range of each block. The DS in the second pass has two input paths: one to transfer a horizontal slice from the decompressed image for HBS, the other to transfer the corresponding slice from the intermediate result $IMAGE_V$ for horizontal de-blocking. After horizontal de-blocking, the results are stored as $IMAGE_{HV}$ in the external memory. In addition, for each slice loaded for HBS, the range of each block is computed. If the range is bigger than the threshold T_r, "1" is stored in an on-chip cache (RANGE) as its range mark. Otherwise, "0" is stored.

After the second pass, the stored range marks in RANGE are used to differentiate texture blocks from edge blocks as described in Step (4) in Section II-B. The final ringing semaphores are stored in an on-chip cache. In the third pass, the DS has one input path to load a horizontal slice from $IMAGE_{HV}$ and one output path to store the results in the external memory after de-ringing. However, the number of image rows transferred in the

input path and the output path are different. To de-ring the pixels at the slice boundary by a 3×3 filtering window, 2 additional rows located at the top and the bottom of the slice need to be loaded in the input path.

4.2.2 POSTPROCESSING ALGORITHM MAPPING

As explained in Chapters 2 and 3, our de-blocking and de-ringing algorithms mainly require simple operations. The MAP-CA has a diverse instruction set in computing these operations using data-level parallelism. For example, the IGALU has a *sum2* instruction, which can sum 16 8-bit values into a 64-bit register in one clock cycle. For our de-blocking algorithm, a 16-tap boxcar filter for each pixel can be achieved by one *sum2* instruction. The de-ringing filter includes addition and division operations. Due to the low throughput [27] of division instructions, they are time-consuming and therefore should be avoided. To achieve better performance, we use a lookup table (LUT) to convert the division into an integer multiplication. For example, to find the mean value of 7 pixels ($a \sim g$) in the de-ringing filter, the straightforward method is to compute "$(a + b + c + d + e + f + g)/7$" directly, which involves an expensive division operation. Since the MAP-CA has an *inner-product* instruction, we convert the above expression into an inner product as "$9362*a + 9362*b + 9362*c + 9362*d + 9362*e + 9362*f + 9362*g$". In this way, we can efficiently exploit the computation power of the *inner-product* instruction in de-ringing. The 16-bit integer "9362" is derived from "1/7" after 16-bit left shift, which is pre-calculated. For a 3×3 de-ringing window, the denominator ranges from 1 to 9. Therefore, the LUT has only 9 entries, which does not occupy much memory.

4.3 RESULTS AND DISCUSSION

Our algorithms have been implemented on MAP-CA using C with intrinsics, which are hints to the compiler about the assembly instructions to use for particular operations. Table 4-1 lists the number of cycles each pass takes in postprocessing a 352×288 FOREMAN image. From Table 1, we can see that each pass is compute-bound, i.e., the I/O cycles of each pass are hidden behind its computation time. Table 4-2 shows the postprocessing time of different images measured on a 300-MHz MAP-CA board. It takes 3.17 ms in de-blocking and de-ringing the FOREMAN image, which accounts for 10.6% of one MAP-CA's computing power at the rate of 30 frames per second. As the image size increases, the execution time increases accordingly. Since the number of blocks for de-blocking and de-ringing is image-dependent, the execution time can vary to some extent for different images with the same size.

Table 4-1. Execution cycles of postprocessing a 352×288 FOREMAN image.

Pass Number	$t_{compute}$	$t_{I/O}$	$\max\{t_{compute}, t_{I/O}\}$
Pass 1 (vertical de-blocking)	215,581	178,864	215,581
Pass 2 (horizontal de-blocking)	314,469	127,545	314,469
Pass 3 (de-ringing)	330,830	97,113	330,830
t_{total}	860,880	403,522	860,880

Table 4-2. Postprocessing performance on a 300-MHz MAP-CA.

Image/Video	Size	Cycles	Execution Time (msec)
GARDEN	352x240	664,966	2.22
FOREMAN	352x288	950,242	3.17
TENNIS	352x288	737,514	2.46
MANDRILL	512x480	1,715,492	5.72
LENA	512x512	2,102,552	7.01
BARBARA	512x512	2,083,724	6.94

In implementing these algorithms on MAP-CA, we used three passes in managing the data flow. The order of these three passes is critical to the actual performance. As mentioned in Section 4.2.2, the second pass (horizontal de-blocking) has one more input path than the first pass (vertical de-blocking). However, the I/O cycles of the first pass in Table 4-1 are about 40% higher than those of the second pass. This is because loading/storing a vertical slice takes more time in most memory systems compared to loading/storing a horizontal slice. If we exchange the order of the passes for horizontal and vertical de-blocking, the I/O cycles of vertical de-blocking increase by 50% since three images (2 inputs and 1 output) will be accessed instead of two. As a result, the I/O cycles will exceed the compute cycles, thus the execution time of the vertical de-blocking pass will increase. On the other hand, since the I/O cycles of the horizontal de-blocking pass are already fewer than its compute cycles, its further-decreased I/O cycles do not result in an improvement in the actual performance. The main reason for the inefficiency

in vertical memory accesses is that the DS transfers consecutively stored data in bursts. Each transfer burst involves an overhead time in addition to the data transfer time. Loading or storing a horizontal/vertical slice of width W pixels and height H rows requires the DS to initiate H transfer bursts of W bytes each. In the case of vertical slices, H is a big number and W is small (e.g., H=288 rows and W=8 bytes), so the transfer burst overhead can become significant in comparison to the data transfer time. Therefore, the W/H ratio needs to be maximized as much as possible for better performance. For example, we have found that the I/O cycles for an entire image by loading 8 columns at a time are about 80% higher than those of loading 16 columns at a time. The numbers shown in Table 4-1 are based on loading 16 columns at a time. For a larger image, if 16 columns cannot be completely fit into the data cache, the DS could transfer 16 half-columns.

Since our algorithms only involve simple operations, e.g., addition, absolute difference, comparison, shift, and multiplication, they can be easily extended to other mediaprocessors. For different mediaprocessors, the major task in mapping our algorithms is to find the corresponding instructions. For example, *dif_pu8* is a MAP-CA instruction to compute the absolute differences between eight byte-pairs. The equivalent instruction in TMS320C64x is *subabs4*. For most mediaprocessors, the frequently-used instructions, e.g., partitioned add, partitioned multiply, and partitioned compare, are all supported under different mnemonics. However, it is possible that some special instructions in a mediaprocessor are not supported by others. As an example, TMS320C64x does not support the *sum2* instruction described in Section 4.2.3. In this case, we need to find a set of instructions to replace *sum2* in an efficient way. Since the summation of pixels can be considered as the inner-product operation between the pixels and 1s, we can use *inner-product* in TMS320C64x as an alternative.

Similarly, the data flow management can be mapped to other mediaprocessors as well. Since different mediaprocessors may have different sizes of on-chip data cache, the size of an image slice should be adjusted accordingly. In many cases, a large on-chip data cache helps managing the data flow more efficiently. For example, TMS320C64x has a 1-Mbyte on-chip cache. In this case, the intermediate results after vertical or horizontal de-blocking can be stored in the on-chip buffer without sending to the external memory. Therefore, all the postprocessing tasks can be done using one DMA pass only.

4.4 REAL-TIME DE-BLOCKING AND DE-RINGING DEMONSTRATION

Figure 4-2 shows the GUI to demonstrate the real-time de-blocking and de-ringing algorithms running on the MAP-CA. The features and usage of this GUI are summarized below.

First, click the "Load Image" button to load an input image, which is the reconstructed image with blocking and ringing artifacts. The input can be either a black-and-white image in the raw format or a color image in the YCbCr (4:2:0) format. The GUI recognizes these two types by the file extension. If the file extension is ".yuv", the input is assumed to be a color image in YCbCr format. Otherwise, it is assumed to be a grayscale image. After loading the input image, click the "Load Dimension" button to enter its width and height. The default dimensions used are "width = 352" and "height = 288". To be robust to human errors, the GUI can check and flag if the input dimensions do not match the file size. After setting the correct dimensions, click the "VPP" button to de-block and de-ring the input image. If the input is a color image, de-blocking and de-ringing is applied to Y, Cb, and Cr components separately. After postprocessing, the input color image, the output image with Y component postprocessed, and the output

image with all components (Y, Cb, and Cr) postprocessed are displayed on a monitor for evaluation. For a black-and-white image, only the Y component exists, thus it is postprocessed. The total de-blocking and de-ringing time (in μs) can be displayed by clicking the "View Profile" button.

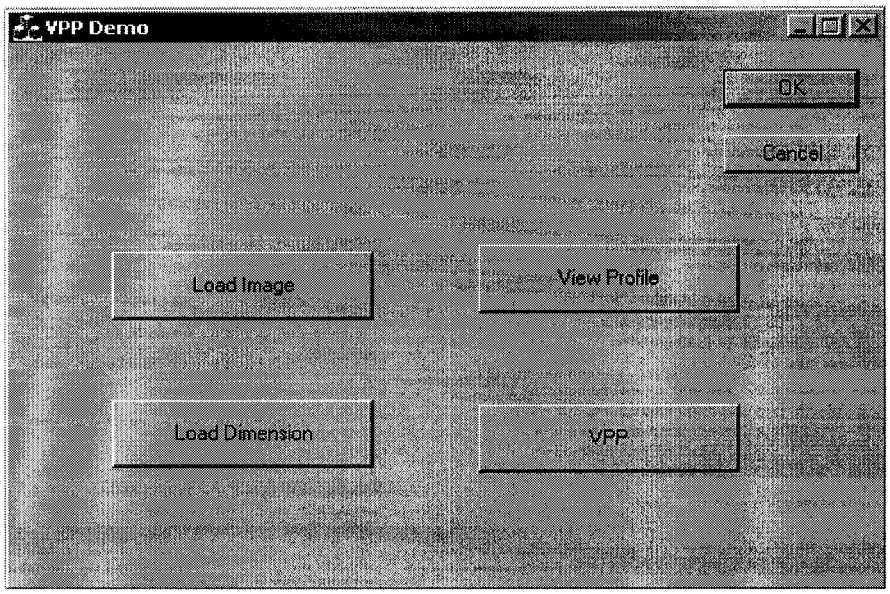


Figure 4-2. GUI to demonstrate real-time de-blocking and de-ringing on the MAP-CA.

As an example, Fig. 4-3 shows the results of the FOREMAN color image in the YCbCr format. Figures 4-3(a)~(d) present the original, reconstructed, Y-postprocessed, and YCbCr-postprocessed images, respectively. Compared to the reconstructed image in Fig. 7(b), the postprocessed images in Figs. 4-3(c) and 4-3(d) have significantly less blocking and ringing artifacts in the wall and face areas. Comparing Figs. 4-3(c) and 4-3(d) on the color screen, we could not see any obvious difference between them. This is because human eyes are not as sensitive to the chrominance components as to the luminance component. As for the computational cost, it takes 4.7 ms to de-block and de-ring the

whole color image, of which about 1.7 ms is used to postprocess the Cb and Cr components. Since postprocessing Cb and Cr does not provide a noticeable visual improvement, we could save about one third of the computation time by postprocessing the Y component only.

(a) Original

(b) Reconstructed

(c) Y postprocessed

(d) YCbCr postprocessed

Figure 4-3. Results of FOREMAN in the YCbCr format.

4.5 CONCLUSIONS

We have implemented our proposed de-blocking and de-ringing algorithms on MAP-CA. The implementation has achieved 3.17 ms for each frame in the FOREMAN sequence, which demonstrates the feasibility of real-time video postprocessing. In addition, our algorithms, only involving simple operations, can be easily extended to other processors.

CHAPTER 5: TEMPORAL FILTERING

5.1 INTRODUCTION

Most image/video sequences are both spatially and temporally correlated. To take advantage of these correlations, three-dimensional (3-D) spatiotemporal filters can be applied for maximum noise reduction. Many 3-D spatiotemporal filters have been proposed in enhancing image sequences [7][12][18][49][51][75][93]. A brief review of these filters was presented in [8]. The major disadvantage of such a filter is its high computational cost. As a result, most proposed 3-D spatiotemporal filters can only be used in those applications without real-time requirements. Since temporal noise could be more annoying than spatial noise, the 1-D temporal filtering [15][52][64][83] has been proposed for faster processing.

1-D temporal filtering has been mainly used in video preprocessing to improve the visual quality and improve the subsequent coding efficiency. Some authors [13][95] have proposed using temporal filtering to reduce the compression artifacts in the reconstructed video sequences. However, temporal filtering alone is not enough to remove those specific blocking and ringing artifacts [62].

In Chapters 2 and 3, we have proposed efficient de-blocking and de-ringing algorithms to reduce blocking and ringing artifacts. After de-blocking and de-ringing, each frame looks much smoother. However, since these de-blocking and de-ringing algorithms were applied to each frame independently, the interframe noise (or temporal noise) still makes the de-blocked and de-ringed video look "busy". To reduce the interframe noise, temporal filtering can be applied after de-blocking and de-ringing.

In the remainder of this chapter, Section 5.2 describes the adaptive motion-compensated temporal filtering algorithm. The simulation results are presented and discussed in Section 5.3. Finally, we summarize our conclusions in Section 5.4.

5.2 TEMPORAL FILTERING SCHEMES

Temporal filtering involves three key issues: motion compensation, adaptive weights, and filter types. Temporal filtering without motion compensation tends to blur the moving objects while non-adaptive temporal filtering cannot effectively deal with local scene changes. In addition, temporal filtering has two types: finite impulse response (FIR) and infinite impulse response (IIR). Investigating these issues can help us select the optimal temporal filtering scheme in terms of quality and computational cost.

5.2.1 MOTION COMPENSATION

Motion compensation can avoid blurring motion objects. The key issue in motion compensation is how to estimate motion vectors. Two major groups of motion estimation techniques have been developed, namely, pel-recursive algorithms [6][14][86][90] and block matching algorithms [56][61][63]. In the pel-recursive algorithms, motion vectors are estimated for individual pixels by recursively using the relative luminance change. In addition to expensive computational cost, poor estimation on edges and uncovered background is the other major drawback of pel-recursive algorithms [91]. To reduce the computational cost and improve the prediction performance, block matching algorithms have been proposed where all pixels within a block are assumed to have the same motion vector.

The block matching algorithms have been adopted in many video standards [30]-[36]. Firstly, the current frame of a video sequence is divided into blocks of $m \times n$. Then, for each block in the current frame, the previous frame is searched within a specified

neighborhood (search area). The motion vector is finally derived based on the best match with respect to a pre-defined matching criterion, such as mean absolute difference (MAD). The most straightforward block matching algorithm is the full search (FS) method. FS checks all possible candidate blocks within the search area. Massive computation is, therefore, required in its implementation. To alleviate the computation burden of FS, several fast search algorithms, such as logarithm search (LOGS) [44], three step search (TSS) [53], conjugate direction search (CDS) [86], cross search (CS) [25], dynamic search-window adjustment (DSWA) [56], new three step search (NTSS) [59], four step search (FSS) [78], block-based gradient descent search (BBGDS) [60], gray prediction search (GPS) [47], and diamond search (DS) [104] have been developed. These fast search algorithms are based on the assumption that the matching error (e.g., MAD) increases monotonically as the search moves away from the global minimum position [10].

The computational cost of these fast algorithms depends on which processor they are implemented on. For example, the NTSS method usually requires fewer operations than the TSS algorithm. However, NTSS involves many conditional and branch operations, which are difficult to be pipelined for parallel processing. As a result, the performance of NTSS can be even worse than that of TSS when they are implemented on a mediaprocessor. In consideration of easy hardware implementation and computational cost, we adopted TSS for motion estimation in our temporal filtering scheme. To further reduce the computational cost, we propose a down-sampled pattern in computing the block matching error. The down-sampled pattern is designed mainly based on two rules: (1) the pixels should be spread over the whole block and (2) the pixels should be easy for memory access. As an example, Fig. 5-1 shows a down-sampled pattern of a 16×16 block. To satisfy the second rule, we use 8 neighboring pixels because they can be loaded in one instruction by many mediaprocessors. Since this down-sampled pattern only uses one quarter of pixels, ideally, it can speed up the performance by four times.

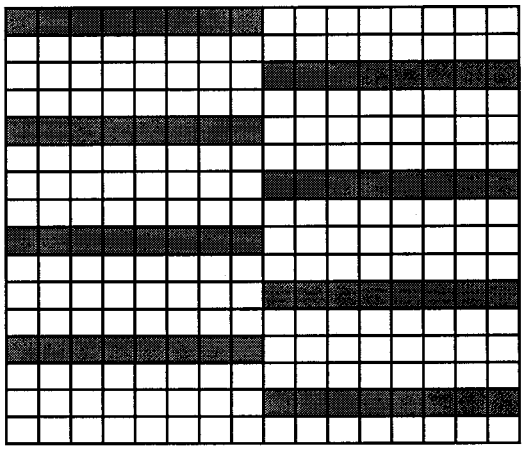


Figure 5-1. Down-sampled pattern in computing the block matching error.

5.2.2 FIR AND IIR TEMPORAL FILTERS

Based on the impulse response of the transfer function, temporal filters can be classified into FIR and IIR. As an example, a motion-compensated FIR filter is given as follows.

$$f(i,j,k) = \sum_{l=-M}^{M} w(i,j,l)\, g(i - d_x^{k,k-l}(i,j),\ j - d_y^{k,k-l}(i,j),\ k-l) \qquad \textbf{(5.1)}$$

where i and j are the spatial locations, k is the frame number, $d^{k,k-l}$ means the motion vector from frame k - l to frame k, $w(i,j,l)$ is the filter weight, $g(i,j,k)$ and $f(i,j,k)$ are the input and output, respectively, and M represents the number of neighboring frames that contribute to the filtering before and after the current frame.

For a FIR temporal filter with equal weights, its noise reduction ability mainly depends on the selection of M. The larger the value of M, the more the noise suppression ability. However, the computational cost increases with M as well. From Eq. (5.1), a ($2M$+1)-tap FIR filter requires the motion vectors between the current frame and its $2M$ neighboring

frames. Due to the high computational cost of motion estimation, a motion-compensated FIR filter is usually more computationally expensive than an IIR filter.

Different from a FIR filter, an IIR filter does not require a large value of M (thus, a filter tap value of $2M+1$) [8]. Eq. (5.2) shows a motion-compensated IIR temporal filter, which only involves the current frame $g(i,j,k)$ and the filtered previous frame $f(i,j,k-1)$.

$$f(i,j,k)=[1-\alpha(i,j,k)]f(i-d_x^{k,k-1}(i,j),\ j-d_y^{k,k-1}(i,j),k-1)+\alpha(i,j,k)\,g(i,j,k) \quad \textbf{(5.2)}$$

where $\alpha(i,j,k)$ is called the filter gain.

5.2.3 ADAPTIVE WEIGHTS

Another key issue in temporal filtering is how to assign the filter weights. Since video sequences involve scene changes, even the best-match blocks can be quite different from each other. If we just average them without considering these changes, severe distortions could result. To avoid the severe distortions while maintaining good noise suppression, the filter weights should be adaptive.

Many methods have been proposed to compute the adaptive weights. Basically, the weights can be adaptive to the noise level and the local changes. As an example, the filter weight $w(i,j,l)$ of the FIR filter shown in Eq. (5.1) can be assigned based on Eqs. (5.3)-(5.6).

$$w(i,j,l) = \frac{\beta}{1 + MAX(\varepsilon^2, AME(i,j,l)^2))} \tag{5.3}$$

$$AME(i,j,l) = \frac{1}{16 \times 16} \sum_{(i,j) \in current\ MB} \left| g(i,j,k) - g(i - d_x^{k,k-l}(i,j), j - d_y^{k,k-l}(i,j), k-l) \right| \tag{5.4}$$

$$\beta = \frac{1}{\sum_{l=-M}^{M} \frac{1}{1 + MAX(\varepsilon^2, AME(i,j,l)^2))}} \tag{5.5}$$

$$\varepsilon^2 = 2 \times E\left\{(g(m,n) - f(m,n))^2\right\} = \frac{QP^2}{6} \tag{5.6}$$

where ε^2 represents the noise variance, $AME(i,j,l)$ is the averaged matching error between the current block, where the pixel $g(i,j,k)$ belongs, and its best-match block at the frame k-l, β is a normalization factor to satisfy $\sum_{l=-M}^{M} w(i,j,l) = 1$, and QP is the quantization parameter.

Eqs. (5.3)-(5.6) show that if the squared $AME(i,j,l)$ is less than or equal to the noise variance ε^2, then the block at the frame (k-l) has an equal weight to the current block (l=0); otherwise, its weight decreases with its squared matching error. The noise variance ε^2 in Eq. (5.6) is approximated by twice of the uniform quantization noise. The constant 2 comes from the assumption that the quantization noise at each frame is independent of any other frames. Therefore, the noise variance of the differentiated signal between two frames is equal to the summation of the noise variance at each frame.

For the IIR filter in Eq. (5.2), the filter gain $\alpha(i,j,k)$ can be adaptively derived based on the absolute difference between pixel values of $f(i,j,k\text{-}1)$ and $g(i,j,k)$. Eq. (5.7) shows how $\alpha(i,j,k)$ is set based on QP.

$$\alpha(i,j,k) = \frac{\min\left\{\left|g(i,j,k) - f(i - d_x^{k,k-1}(i,j), j - d_y^{k,k-1}(i,j), k-1)\right|,\ QP\right\}}{QP} \quad \textbf{(5.7)}$$

Here, we compare the absolute pixel difference with QP. The reasons are given as follows. Temporal filtering is trying to smooth those pixels with similar gray values before quantization. For two pixels a and b before quantization, their quantized values ($\hat{a}$ and $\hat{b}$) should locate within $[a - q/2, a + q/2]$ and $[b - q/2, b + q/2]$, respectively. q is the quantization step size in the spatial domain. If we assume $a = b$, then the maximum difference between $\hat{a}$ and $\hat{b}$ is q. As explained in Section 3.2.2, the quantization step size q in the spatial domain can be approximated by QP in the sense of MSE. Therefore, it is unlikely for two similar pixels to have a difference of more than QP after quantization. In Eq. (5.7), if the absolute difference between the current pixel and its corresponding pixel in the previous frame is bigger than QP, $\alpha(i,j,k)$ is set to 1, i.e., no filtering is applied to the current pixel because local scene changes have happened. As the absolute difference decreases, the current pixel becomes closer and closer to the previous output at its motion-compensated position. Since the previous output was already filtered, it is more reliable than the current pixel. Therefore, more weight should be assigned to the previous output. Finally, $\alpha(i,j,k)$ reaches zero if the current pixel is equal to the previous output. In this special case, the previous output completely replaces the current pixel.

5.2.4 PROPOSED TEMPORAL FILTERING SCHEME

Based on the above analysis in Sections 5.2.1-5.2.3, we propose using a motion-compensated adaptive IIR temporal filter. The filter expression is given in Eq. (5.2). The

motion vectors are estimated using TSS plus our proposed down-sampled pattern. The filter type is set as IIR, whose adaptive weights are derived from Eq. (5.7).

5.3 SIMULATION RESULTS AND DISCUSSIONS

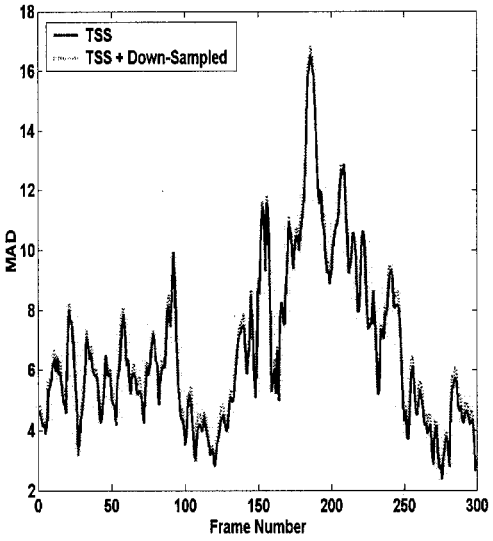
We have tested our temporal filtering schemes by two typical video sequences: the 300-frame FOREMAN and TENNIS. The FOREMAN sequence can represent those video sequences with slow and/or moderate movements while the TENNIS involves fast motions. In terms of video contents, both FOREMAN and TENNIS include smooth areas and strong edges. However, TENNIS has much more high frequency textures, such as the audience areas, than FOREMAN. These two sequences were encoded at a fixed *QP* of 31 separately. Their bit rates are about 0.125 bpp. After decoding, our proposed de-blocking and de-ringing algorithms were applied to reduce the blocking and ringing artifacts within each frame. The de-blocked and de-ringed FOREMAN and TENNIS sequences are used as the inputs of all temporal filtering schemes.

5.3.1 RESULTS OF THE DOWN-SAMPLED PATTERN IN MOTION ESTIMATION

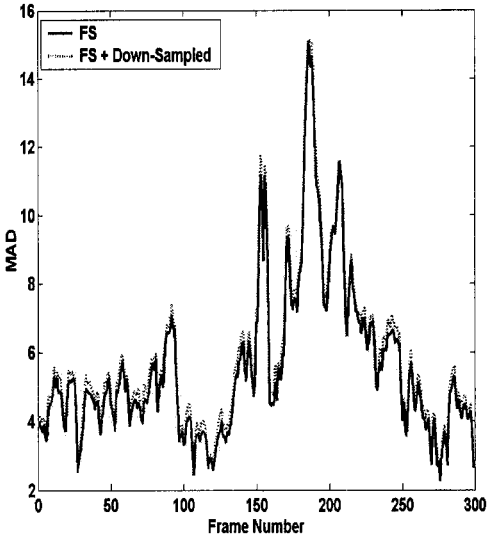
To evaluate our proposed down-sampled pattern (shown in Fig. 5-1) for motion estimation, we also computed the motion vectors using all the pixels in block matching. After deriving these two sets of motion vectors, we use them to generate two motion-compensated frames for each current frame. Then the MAD is calculated based on the current frame and the corresponding motion compensated frame, separately. The performance of the down-sampled pattern is measured by the relative difference (percent increase) between these two calculated MADs. Since the performance of the down-sampled pattern may depend on the search algorithm itself, we tested it with two different search algorithms: FS and TSS. Fig. 5-2 shows the results of FOREMAN and TENNIS. The solid line in each sub-figure represents the MAD without the down-sampled pattern while the dashed line shows the MAD with the down-sampled pattern. For each

sequence, we have observed that the MAD with the down-sampled pattern is very close to that using all pixels regardless of the search algorithm. Table 5-1 summarizes the MAD for different cases. For the FOREMAN using TSS, the average MAD for each frame without the down-sampled pattern is 6.67. After applying the down-sampled pattern, the average MAD increased 3.30% to 6.89. Similar results can be observed for the other three cases. For the motion-compensated temporal filtering, the slightly increased MAD will not affect its filtering ability. Since the down-sampled pattern only uses one quarter of the total pixels in motion estimation, ideally it can speed up the performance by up to four times.

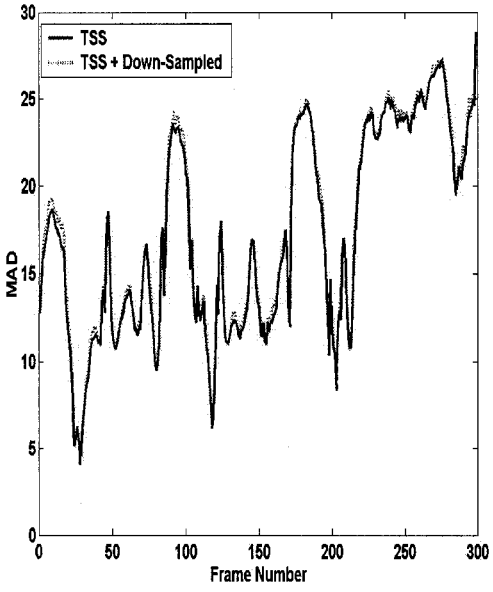
As shown in the last column of Table 5-1, the MAD percent increase using FS is higher than that using TSS for the same sequence. For example, the MAD of TENNIS using FS increased 3.20% while its MAD using TSS only increased 2.10%. One obvious reason is that FS can always find the best match while TSS cannot. Therefore, the overall MAD using FS is smaller than that using TSS, i.e., the same amount increase of MAD in FS and TSS would result in a higher percentage for FS because of the smaller denominator. However, this is not the only reason. Another important reason is that TSS with the down-sampled pattern can sometimes outperform TSS with all pixels. It is easy to understand that the MAD using FS with all pixels is always less than or equal to that using FS plus the down-sampled pattern because FS with the down-sampled pattern may not find the global minimum. However, this statement does not hold for TSS. Since TSS can be trapped into a local minimum, TSS with the down-sampled pattern may sometimes find a better match than TSS with all pixels. For example, we have found that TSS with the down-sampled pattern has a lower MAD than TSS with all pixels for 23 frames in the 300-frame FOREMAN sequence. These 23 frames contribute to the reduction of the relative difference between TSS with all pixels and TSS with the down-sampled pattern, which also explains why the overall MAD increase of TSS has a lower percentage than FS.



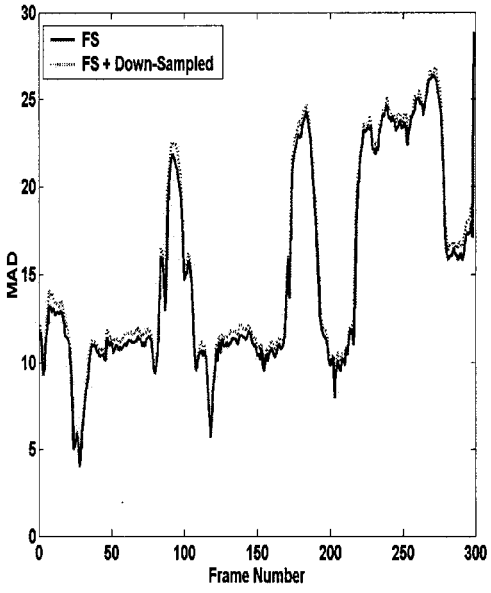
(a) FOREMAN by TSS with/without down-sampled pattern.



(b) FOREMAN by FS with/without the down-sampled pattern.



(c) TENNIS by TSS with/without the down-sampled pattern.



(d) TENNIS by FS with/without the down-sampled pattern.

Figure 5-2. Motion estimation results with and without the down-sampled pattern.

Table 5-1 Mean absolute difference with and without the down-sampled pattern averaged over all frames.

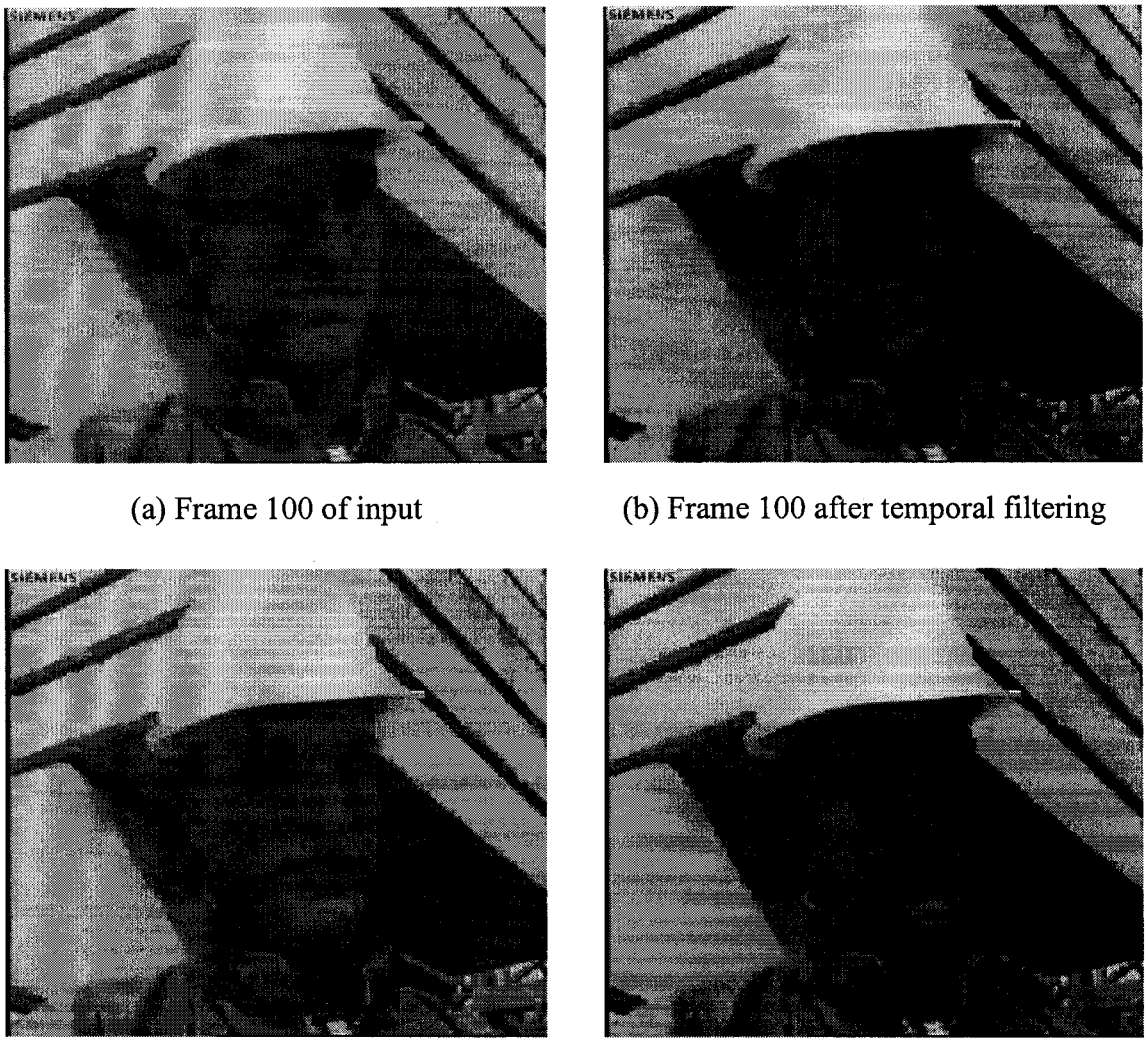
Sequence and search method	Average MAD		
	Without down-sampled	With down-sampled	Increase
FOREMAN by TSS	6.67	6.89	3.30%
FOREMAN by FS	5.59	5.86	4.83%
TENNIS by TSS	17.15	17.51	2.10%
TENNIS by FS	14.98	15.46	3.20%

In addition, the MAD of TENNIS is higher than that of FOREMAN. This is mainly because TENNIS has faster motions. In this simulation, we set the motion search range as 15×15 for both TENNIS and FOREMAN. Since the motions in FOREMAN are slower than TENNIS, they are better compensated. Also, TENNIS has a lot of high frequency textures at each frame, such as audience areas. Since these audience areas involve many local scene changes, they are more difficult to be well compensated than those smooth walls in FOREMAN.

5.3.2 RESULTS OF THE MOTION-COMPENSATED ADAPTIVE IIR TEMPORAL FILTERING

Based on the motion vectors estimated by TSS with the down-sampled pattern, we applied a motion-compensated adaptive IIR filter to the de-blocked and de-ringed FOREMAN and TENNIS sequences. The filter gain $\alpha(i,j,k)$ (or adaptive weight) is adaptively derived based on the difference between $f(i,j,k\text{-}1)$ and $g(i,j,k)$ as given in Eq. (5.7). After temporal filtering, we did subjective quality evaluation of the input and output video sequences. The interframe abrupt changes in the filtered FOREMAN and TENNIS sequences were significantly reduced. Fig. 5-3 shows the 100^{th} and 101^{st} frames extracted from the FOREMAN sequences before and after temporal filtering. Observing

Fig. 5-3, we found that some abrupt changes around the face, mouth, eye, and walls were smoothed after temporal filtering.



(a) Frame 100 of input (b) Frame 100 after temporal filtering

(c) Frame 101 of input (d) Frame 101 after temporal filtering

Figure 5-3. Results of temporal filtering.

To see the temporal smoothing more clearly, Fig. 5-4 compares the luminance value of a pixel at the face area of the FOREMAN sequence before and after temporal filtering over different frames. At the first frame, we picked a face pixel located at (159, 180). Then,

we tracked its new positions at later frames through its motion vectors. The motion vectors were estimated using TSS plus the down-sampled pattern. For example, its new location at the 150th frame becomes (130, 172). Figure 5-4 plots the pixel value along its motion trajectory before and after temporal filtering. As shown in Fig. 5-4, the dashed line represents the pixel value before temporal filtering while the solid line shows its filtered value. Obviously, the interframe abrupt changes were significantly reduced after temporal filtering.

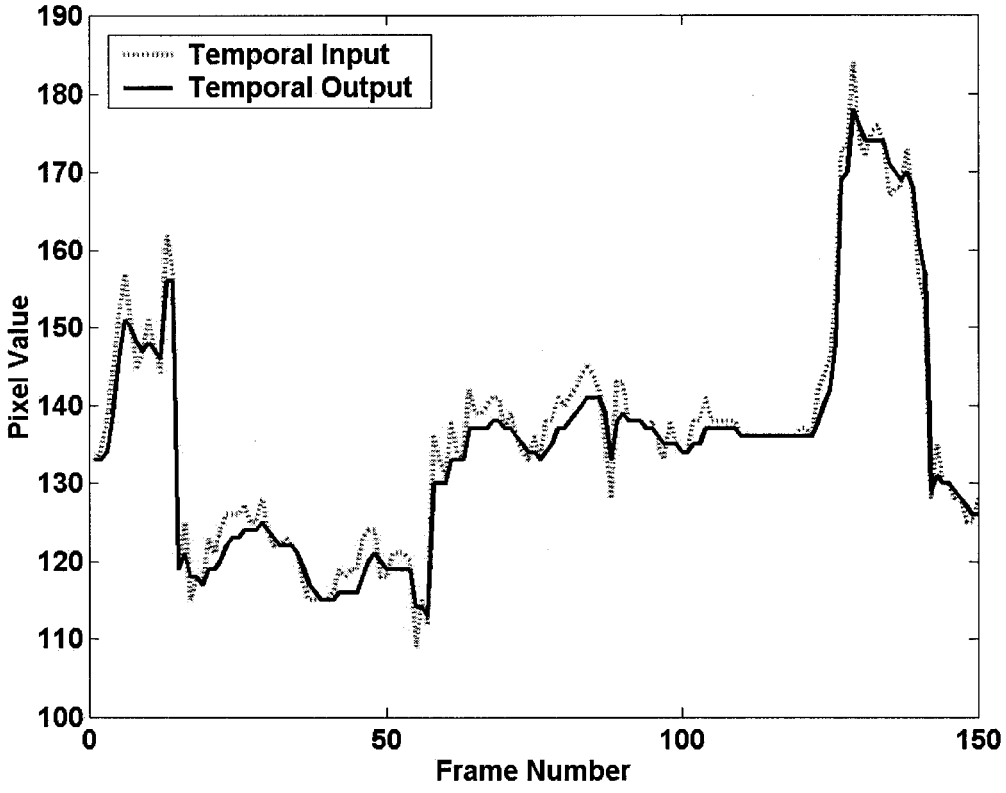
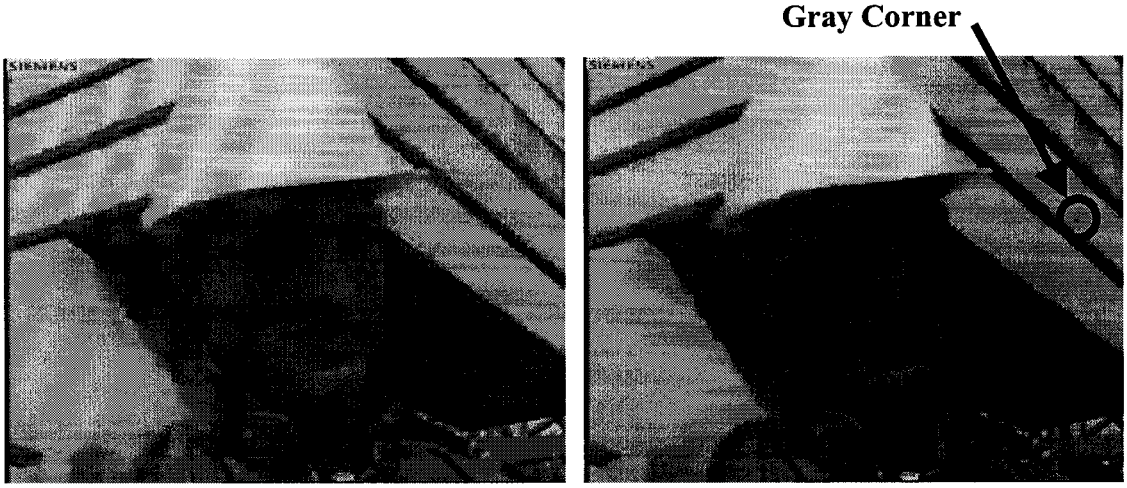


Figure 5-4. A pixel at the face area of FOREMAN before and after temporal filtering over different frames.

In the above experiment, the weight $\alpha(i,j,k)$ was adaptively computed for each pixel $g(i,j,k)$. To further reduce the computational cost, the adaptive weight can be assigned based on each block, i.e., all pixels within a block share the same weight, which is derived from the MAD between this block and its motion-compensated block. The block-based adaptive weights can smooth pixels of a bigger dynamic range. More specifically, if the MAD of a block is less than or equal to a threshold, it allows some individual absolute differences to be much bigger than the threshold. Therefore, block-based adaptive weights have stronger ability to smooth the extreme noise. We have found the block-based adaptive weights can sometimes provide better visual results than the pixel-based adaptive weights. However, block-based weights could cause severe "corner error" problems as well. For example, a "white" block with a small "black" corner can be the best match of another "white" block because the MAD between them is small. Since the filter weight is determined by the MAD, the small "black" corner will contribute to the "white" block as a "gray" corner. And this "gray" corner could be propagated into its following frames. As a result, the video is degraded by some flying "gray" spots. Figure 5-5 shows an example of the "corner error" problem. Figure 5-5(a) is the 106^{th} frame extracted from the FOREMAN sequence after de-blocking and de-ringing. Figure 5-5(b) is its output after the temporal filtering using block-based adaptive weights. Here, the threshold for the MAD is set as $QP/2$. Observing Fig. 5-5(b), we can see a "gray corner" generated in the smooth wall area, which resulted from the block-based smoothing. The severity of the "corner error" problem depends on the MAD threshold. A smaller threshold can alleviate the "corner error" problem, but cannot eliminate it completely. Also, as the MAD threshold decreases, the filter's noise suppression ability decreases as well. To avoid the "corner error" problem and achieve good noise reduction ability, we decided to compute the weight based on each pixel. In this case, each pixel's value cannot change by more than QP after filtering.

We also compared the IIR filter with a 7-tap motion-compensated adaptive FIR filter. The 7-tap FIR filter requires the motion vectors between the current frame and its 6

neighboring frames. These motion vectors were estimated using TSS plus the down-sampled pattern. The adaptive weights were derived based on Eqs. (5.3)-(5.6). During subjective evaluation, we did not observe obvious quality difference between the FIR and IIR results. Since FIR requires much more motion vectors, which is time-consuming to compute, IIR is preferred in real-time applications.



(a) Frame 106 of input (b) Frame 106 of output

Figure 5-5. Example of "corner error" caused by block-based adaptive weights.

5.4 CONCLUSIONS

We have proposed a motion-compensated adaptive IIR temporal filtering scheme with a down-sampled pattern. Our proposed down-sampled pattern speeds up the motion estimation by up to four times while the MAD using this pattern increases less than 5% compared to that using all pixel values. The slightly increased MAD can be well handled by using adaptive weights in the subsequent temporal filtering. We have also studied

different temporal filtering schemes regarding adaptive weights and filter types. In terms of computational cost and video quality, the adaptive IIR filter is a better choice.

CHAPTER 6: ALGORITHM VALIDATION

6.1 INTRODUCTION

Since PSNR is a poor indicator in measuring compression artifacts, currently most researchers evaluate their postprocessing algorithms by presenting their results for visual inspection. Some authors used the ITU-R 5-point quality scale [39] (5 = excellent, 4 = good, 3 = fair, 2 = poor, and 1 = bad) to score their results. However, the quality number depends on the selection and sophistication of observers. Since the postprocessed images/videos involve several artifacts, such as blocking, ringing, and temporal noise, it is unreliable for a non-expert observer to give a quality number without full consideration of these artifacts.

To reduce the risk of misjudgment, the paired comparisons method by Kendall [50] has been used in various areas [4][20][21][72][81][85] for subjective quality assessment. Given two objects, the basic idea of the paired comparisons method is to determine which one is preferable. For example, optometrists ask patients to identify which of the two lenses enables them to see better. Obviously, it is easier for a subject to select a preferable lens than to score it by a number. Due to the advantages of simplicity and low risk, several researchers [26][46][71] used the paired comparisons method in evaluating image/video qualities. To evaluate n images/videos, it is necessary for observers to compare every possible combination, i.e., $n*(n-1)/2$ comparisons for each observer. The preferences of each observer are then recorded in a preference table. Finally, the images/videos are ranked in accordance with their sum of preferences (SOP) from all observers.

The paired comparisons method is suitable for ranking n given images. To evaluate several different postprocessing algorithms, we can apply the same input to them and compare their outputs. In other words, the paired comparisons method could be used in evaluating different postprocessing algorithms through ranking their results. However, one potential issue here is that the performance of different postprocessing algorithms may be image-dependent, i.e., different test images may cause different rankings for the same algorithms. For fair evaluation, these algorithms should be tested with as many images as possible. Traditionally, the paired comparisons method is to rank all the images by comparing any two of them. However, we should not form two different test images as a pair because they are seldom comparable. Therefore, we need to modify the traditional paired comparisons method before applying it in evaluating different postprocessing algorithms.

In the remainder of this chapter, Section 6.2 describes the modified paired comparisons methodology. The experimental results are presented and discussed in Section 6.3. Section 6.4 summarizes the conclusions.

6.2 MODIFIED PAIRED COMPARISONS METHODOLOGY

Our postprocessing algorithms consist of de-blocking, de-ringing, and temporal filtering. The de-blocking and de-ringing algorithms are evaluated together using still images and images from videos while the temporal filtering algorithm is evaluated using videos. Since the performance of de-blocking and de-ringing may vary with different images, we tested them using various images, which include edges, textures, and smooth areas. Also, these images are compressed at different bit rates so that we can evaluate the postprocessing performances under different artifact levels. For each reconstructed image at a given compression quality, three postprocessed images were generated using the PL [76], SK [84], and our proposed algorithms [23][24], respectively. The reconstructed image and its three postprocessed images form 6 pairs for comparisons.

Since it is possible that the two images are too close to be differentiated, we allowed "no preference" to avoid pure guesses. To evaluate the temporal filtering, we compared different videos before and after the temporal filtering. Due to the lack of comparability between different test images/videos, they were divided into different groups. All the images/videos at each group were generated from the same original test image/video. After these modifications, the paired comparisons method was used to evaluate each group as follows:

1) Organize the images/videos in the same group as pairs.
2) Focus on the specific artifacts (blocking and ringing/temporal noise).
3) Compare the two images/videos and select the preferable one ("no preference" is allowed).
4) Repeat steps 1) to 3) for the next pair until all the combinations are evaluated.

In the experiment, the preferences from all observers were automatically recorded. As an example, Table 6-1 shows the preferences of one observer in comparing four still images (A~D). As shown in Table 6-1, if A is preferable to B, the preference in row A and column B is 1 and the preference in row B and column A is 0. For convenience, we use A→B to represent A is preferable to B.

Table 6-1. Preferences of four images from one observer.

Images	A	B	C	D	SOP
A	—	1	0	1	2
B	0	—	1	1	2
C	1	0	—	1	2
D	0	0	0	—	0

The last column of Table 6-1 gives the SOP for each image. Ideally, the SOP should be different so that images can be ranked without any ties. However, 3 images (A, B, and C) in Table 6-1 have the identical SOP value, which may come from "no preferences" (0.5 scores) or inconsistency of the observer. Since Table 6-1 has no 0.5 scores recorded, the tied SOPs must result from the observer's inconsistency. For example, we can find an inconsistent preference circle (A→B→C→A) by examining the relationship among A, B, and C in Table 6-1. Obviously, one observer could generate several circles in his/her preference tables. The more cycles, the more inconsistent. In this study, we used the Kendall's coefficient of consistence ζ [50] to measure the consistency of each observer. Eq. (6.1) shows how to compute ζ.

$$\zeta = \begin{cases} 1 - \dfrac{24d}{n^3 - n}, & n \ odd \\ 1 - \dfrac{24d}{n^3 - 4n}, & n \ even \end{cases} \tag{6.1}$$

where n is the number of images to be compared and d is the number of total preference circles for one observer.

If the preferences of an observer are completely consistent, i.e., no circles ($d = 0$), then ζ reaches its maximum value 1. The minimum value of ζ is 0. The larger the ζ, the more consistent the observer.

ζ describes the consistency of each observer. To measure the degree of agreement between all m observers, we compute the coefficient of agreement u [50]. To compute u, we first sum the preference tables from all m observers together into a new table. Denote $s_{i,j}$ as the value at row i and column j in the new table, then we have

$$u = \frac{4 * \sum_{i=1}^{m} \sum_{j=1}^{n} s_{i,j}(s_{i,j} - 1)}{m(m-1)n(n-1)} - 1 \tag{6.2}$$

When and only when the preferences of all m observers are completely agreed to each other and no 0.5 scores are recorded, u reaches its maximum value 1. The minimum value of u is $-1/(m\text{-}1)$ when m is even, or $-1/m$ when m is odd. A negative u means observers tend to disagree with each other while a positive u indicates a certain agreement between observers. u=0 means observers' opinions are independent.

Statistically, ζ and u are all random variables. The significance of ζ or u can be examined by considering their probability distribution if all the preferences are allotted randomly. As n is larger than 7, the probability of ζ can be well approximated by χ^2 – distribution [50]. Similarly, the probability of u can be well approximated by χ^2 – distribution if $m*n$ is larger than 24. Eqs. (6.3) and (6.4) give the χ^2 – distribution parameters for ζ and u, respectively.

$$\left.\begin{aligned} \nu_\zeta &= \frac{n(n-1)(n-2)}{(n-4)^2} \\ \chi_\zeta^{\,2} &= \frac{8}{n-4}\left\{\frac{1}{4}\binom{n}{3} - d + \frac{1}{2}\right\} + \nu_\zeta \end{aligned}\right\} \tag{6.3}$$

$$\left.\begin{aligned} \nu_u &= \binom{n}{2}\frac{m(m-1)}{(m-2)^2} \\ \chi_u^{\,2} &= \frac{2}{m-2}\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} s_{i,j}(s_{i,j}-1) - \binom{n}{2}\binom{m}{2}\frac{m-3}{m-2}\right\} \end{aligned}\right\} \tag{6.4}$$

Eqs. (6.3) and (6.4) tell us that the χ^2 of ζ or u is distributed in the usual form with ν degrees of freedom. With the χ^2 and ν given in Eqs (6.3) and (6.4), we can compute the probability of ζ and u, respectively. For small n and m, the probability of ζ and u was provided by Kendall through Appendix Tables (9) and (10) in [50].

Given the probability p of ζ or u, its significance is defined as the complement of its probability (1 - p), which can be examined by a significance level α (threshold to the significance). If (1 - p) is less than or equal to α, the result is considered as significant at the α level. In general, if the experiment is carried out by a small group of experts, the significance level α is set as 0.1. If the observers are non-experts, α is usually set as 0.25 [26].

The significance of ζ or u is to check the reliability in terms of observers. To assign different ranks to two images/videos, we still need to check the least significance difference (LSD) of SOP. In this study, we used the Fisher's LSD [11] at the 0.05 significance level to examine the significance of the SOP difference. Only when the SOP difference between two images/videos is larger than the LSD, they are considered as significantly different.

6.3 EXPERIMENTAL RESULTS AND DISCUSSION

Our subjective evaluation experiments were carried out using 10 non-expert observers. The image/video pair was displayed on a 19-inch computer screen in a dark room. The distance between the observer and the computer screen is about 20 inches. Each observer was asked to compare 108 image pairs for de-blocking and de-ringing. The 108 image pairs come from the six original images as shown in Fig. 2-6 (the manually added strong edges in TENNIS were removed for general purpose). Each image was compressed at three different bit rates: 0.125, 0.25, and 0.5 bpp. For each image at a given bit rate, the reconstructed image and its three de-blocked and de-ringed images using the PL, SK, and our proposed algorithms formed 6 pairs for comparison. The observers were asked to click the preferable image (click its original image if "no preference"), and the clicks were automatically recorded.

To evaluate our temporal filtering algorithm, we compressed the 300-frame FOREMAN and TENNIS sequences using three different quantization parameters (QP = 10, 20, and 30). The six reconstructed video sequences were first de-blocked and de-ringed using our proposed de-blocking and de-ringing algorithms. The de-blocked and de-ringed video sequences were then filtered using the proposed temporal filtering scheme. The video sequences before and after the temporal filtering formed 6 video pairs for comparison.

To evaluate two videos, researchers [1][85] usually play one video after the other. However, this requires observers to memorize the first played video. To avoid the memory issue, we horizontally integrated the video pair into a new video. For example, the original frame size of the FOREMAN sequence is 352×288. After the integration, the new video sequence has a frame size of 704×288 instead. Since the two videos to be compared are located side by side within the same sequence, their synchronization is guaranteed. The observers were asked to play each integrated video sequence twice before determining whether the video at the left or at the right is better. For both the still images and videos, their locations (the left side or the right side) were randomly allocated.

6.3.1 RESULTS OF EVALUATING DE-BLOCKING AND DE-RINGING

To evaluate different de-blocking and de-ringing algorithms, we recorded 180 (10 observers, 6 images, and 3 compression ratios) preference tables from 10 observers. Table 6-2 is the summation of all these preference tables. The SOP of the reconstructed image and the postprocessed results using the PL, SK, and proposed algorithms are 21, 310.5, 322, and 426.5, respectively. Thus, according to the SOP, we can see that the postprocessed images have much better visual quality than the images without any postprocessing. Also, the PL and SK algorithms have similar performance while our proposed algorithms achieve a significant improvement over the other two algorithms.

Table 6-3 shows the coefficient of consistency ζ and its probability P_ζ for each observer at three different bit rates. The mean value of ζ for all observers over different testing images is 0.8 and its significance ($1 - P_\zeta = 0.15$) satisfies the 0.25 significance level for non-expert observers. Comparing ζ at different bit rates, we can find that observers are much more consistent at lower bit rates (0.125 and 0.25 bpp) than the high bit rate (0.5 bpp). This is mainly because the artifacts in images at the high bit rates are hardly visible. Several observers preferred some reconstructed images at 0.5 bpp to their postprocessed ones because they cannot consistently differentiate them. As a result, much more preference circles were generated. Table 6-3 shows that the consistency of 5 observers at the bit rate of 0.5 bpp cannot meet the 0.25 significance level. However, all of them exhibited significant consistency at the lower bit rates.

Table 6-2. Summation of 180 preference tables.

	Reconstructed	PL	SK	Proposed	SOP
Reconstructed	—	6	7	8	**21**
PL	174	—	85.5	51	**310.5**
SK	173	94.5	—	54.5	**322**
Proposed	172	129	125.5	—	**426.5**

The coefficients of agreement u and their probabilities P_u at different bit rates are presented in Table 6-4. $u = 0.43$ means that all 10 observers agreed with each other pretty well. The significance of u is 0.000009, which means the agreement between

observers is very unlikely caused by random preferences. Obviously, u satisfies the 0.25 significance level for non-expert observers. Similar to ζ, observers agreed more with each other at lower bit rates than 0.5 bpp. However, even at 0.5 bpp, $u = 0.27$ still shows significant agreement ($P_u = 0.998$) between observers. This is because observers selected "no preference" for many image pairs at this bit rate.

Table 6-3. Coefficients of consistence and their probabilities.

	0.125 bpp		0.25 bpp		0.5 bpp		Mean over bpps	
	ζ	P_ζ	ζ	P_ζ	ζ	P_ζ	ζ	P_ζ
OB1	0.75	0.83	0.83	0.88	0.67	0.79	0.75	0.83
OB2	0.75	0.83	1.00	1.00	**0.58**	**0.73**	0.78	0.85
OB3	0.67	0.75	0.83	0.88	**0.67**	**0.66**	0.72	0.76
OB4	0.92	0.94	1.00	1.00	0.83	0.88	0.92	0.94
OB5	0.83	0.88	0.92	0.94	**0.50**	**0.66**	0.75	0.83
OB6	0.83	0.88	0.75	0.83	**0.50**	**0.66**	0.71	0.79
OB7	1.00	1.00	1.00	1.00	0.67	0.77	0.89	0.92
OB8	0.83	0.88	0.67	0.75	0.83	0.88	0.78	0.84
OB9	0.92	0.94	1.00	1.00	0.75	0.81	0.89	0.92
OB10	0.75	0.81	1.00	1.00	**0.58**	**0.73**	0.78	0.85
Mean	0.83	0.87	0.90	0.93	0.66	0.75	0.80	0.85

Since both ζ and u satisfy the significance level for non-expert observers, the subjective evaluation results can be considered as statistically reliable. However, to rank these

images using their SOP, we still need to check the significance of the SOP difference. In this experiment, only when the SOP difference between two images is larger than or equal to the Fisher's LSD using the 0.05 significance level, these two images are considered as significantly different. For the SOP in Table 6-2, the Fisher's LSD is 37. Obviously, the SOP difference between our proposed algorithm and the PL or SK algorithm far exceeds this threshold. Therefore, we can say that our algorithm achieved a significant quality improvement over both of them. Also, since the SOP difference between the PL and SK algorithms is less than 37, they should have the same rank.

Table 6-4. Coefficients of agreement and their probabilities.

	0.125 bpp	0.25 bpp	0.5 bpp	Overall
u	0.50	0.63	0.27	0.43
P_u	0.9999994	0.99999996	0.998	0.999991

To evaluate different postprocessing algorithms in more details, Tables 6-5~6-7 show the SOP at bit rates of 0.125, 0.25, and 0.5 bpp, respectively. For the SOP in Tables 6-5~6-7, the Fisher's LSD is 22 because of fewer preferences. Similar to the overall evaluation, our proposed algorithm achieves much better results than the PL and SK algorithms at the bit rates of 0.125 and 0.25 while the PL and SK algorithms have the same performance. At the bit rate of 0.5 bpp, the SK algorithm has a slightly higher SOP value than the PL and our proposed algorithms. However, statistically the SOP difference is too small to be significant. Therefore, we can conclude that all three algorithms have similar performance at the bit rate of 0.5 bpp. In addition, all three algorithms still achieved a significant quality improvement over the reconstructed images at 0.5 bpp.

We should mention that the subjective assessment results in this experiment are consistent with those using our objective blockiness metric in Chapter 2.

Table 6-5. Summation of 60 preference tables at 0.125 bpp.

	Reconstructed	PL	SK	Proposed	SOP
Reconstructe	—	1	0.5	2.5	**4**
PL	59	—	31.5	13.5	**104**
SK	59.5	28.5	—	15	**103**
Proposed	57.5	46.5	45	—	**149**

Table 6-6. Summation of 60 preference tables at 0.25 bpp.

	Reconstructed	PL	SK	Proposed	SOP
Reconstructe	—	0.5	0.5	0	**1**
PL	59.5	—	29	9	**97.5**
SK	59.5	31	—	7.5	**98**
Proposed	60	51	52.5	—	**163.5**

Table 6-7. Summation of 60 preference tables at 0.5 bpp.

	Reconstructed	PL	SK	Proposed	SOP
Reconstructe	—	4.5	6	5.5	**16**
PL	55.5	—	25	28.5	**109**
SK	54	35	—	32	**121**
Proposed	54.5	31.5	28	—	**114**

6.3.2 RESULTS OF EVALUATING TEMPORAL FILTERING

To evaluate the temporal filtering, we recorded 60 (10 observers, 2 videos, and 3 compression ratios) preferences from 10 observers. Table 6-8 shows the SOP for videos compressed using three different QPs. In this experiment, only one observer preferred a video before temporal filtering to that after temporal filtering. All the other SOP of the video before temporal filtering resulted from the 0.5 scores ("no preference"). Different from evaluating images, several observers always selected the temporally filtered videos for all pairs. This indicates that observers are more sensitive to the changes in videos than in still images even at the high bit rate.

Table 6-8. SOP of videos before and after temporal filtering.

	Video before temporal filtering	Video after temporal filtering
QP = 10	2.5	17.5
QP = 20	0.5	19.5
QP = 30	1.5	18.5
Total	4.5	55.5

Since each video at a given QP only generated one pair for comparison, the preference circle is not available in this experiment. The coefficient of agreement u between all 10 observers is 0.70, which is high. Its significance 0.005 (P_u = 0.995) is well below the 0.25 level for non-expert observers. Similarly, we applied the Fisher's LSD to examine the significance of the SOP difference. The SOP difference between the filtered and

unfiltered videos is 51 while the Fisher's LSD using the 0.05 significance level is 24. Therefore, our temporal filtering scheme significantly improves the video quality. Also, we examined the significance of the SOP difference for videos compressed at each QP. For the SOP of videos at each QP, the Fisher's LSD is 14. Table 6-8 shows that the SOP differences in all three cases are larger than 14. Therefore, a significant quality improvement is achieved at each compression quality too.

6.4 CONCLUSIONS

We have modified and applied the paired comparisons method in validating our postprocessing algorithms using multiple groups of images/videos. The experimental results confirm that our proposed de-blocking and de-ringing algorithms achieved a significant quality improvement over some other algorithms, especially at the low and moderate bit rates. Also, the subjective assessment results are consistent with our proposed blockiness metric. In addition, we have evaluated our temporal filtering scheme. The proposed temporal filtering achieved a significant visual quality improvement at various compression qualities.

CHAPTER 7: CONCLUSIONS AND FUTURE DIRECTIONS

7.1 CONCLUSIONS

A real-time video postprocessing system has been proposed. Several innovative algorithms in de-blocking, de-ringing, and temporal filtering have been developed. In addition to low computational cost, these algorithms have achieved very promising results in our experiments. Also, we have established an objective blockiness metric. These algorithms and metrics were validated by statistics-based subjective assessment experiments, and they could be used in many applications that involve image/video compression.

Our proposed adaptive de-blocking and clustering-based de-ringing can efficiently and effectively reduce the blocking and ringing artifacts while keeping the strong edges and textures untouched. The experimental results with the highly or moderately compressed images have shown a significant improvement over the existing algorithms, both subjectively and objectively.

Our proposed blocking artifact metrics can be used to quantitatively measure the blocking artifacts in the reconstructed images and evaluate different postprocessing algorithms. Also, they can be used in the encoder to keep the blocking artifact under a certain level. These metrics do not need the original image and do not consider the strong edges as blocking artifacts. The simulation results demonstrate that our proposed metric is more consistent with subjective evaluation than the PSNR/MSE metric.

In addition to the strong ability in artifact reduction, our de-blocking and de-ringing algorithms also have low computational cost. The implementation of our algorithms on MAP-CA by C with intrinsics has achieved more than 300 frames per second for the FOREMAN sequence, which meets the real-time requirements in video postprocessing. In addition, our algorithms, only involving simple operations, can be easily extended to other mediaprocessors, e.g., TMS320C64x and Pentium 4.

To reduce the temporal noise in videos, we have proposed a motion-compensated adaptive IIR temporal filtering scheme with a down-sampled pattern. Our proposed down-sampled pattern speeds up the motion estimation by up to four times while the MAD using this pattern increases less than 5% compared to that using all pixel values. The slightly increased MAD can be well handled by using adaptive weights in the subsequent temporal filtering. We have also studied different temporal filtering schemes regarding adaptive weights and filter types. In terms of computational cost and video quality, the adaptive IIR filter would be a preferred choice.

To validate our postprocessing algorithms and metrics, we have modified and applied the paired comparisons method in evaluating multiple groups of images/videos. The subjective assessment results confirm that our proposed de-blocking and de-ringing algorithms achieved a significant quality improvement over some other algorithms, especially at the low and moderate bit rates. These subjective results are also consistent with those using our objective blockiness metric. In addition, we have evaluated our temporal filtering scheme. The proposed temporal filtering achieved a significant visual quality improvement at various compression qualities.

7.2 CONTRIBUTIONS

The major contribution of this research is to the field of image or video postprocessing. Several new algorithms and metrics, which have low computational cost and reduce the compression artifacts effectively, have been proposed.

Adaptive de-blocking: In this research, we designed an efficient blocking semaphore extraction algorithm. It only identifies those block pairs that really need de-blocking. Compared to other algorithms, our blocking semaphore extraction algorithm greatly reduces the subsequent de-blocking time. After blocking semaphore extraction, we apply an adaptive de-blocking to those extracted blocks. The key idea of our adaptive de-blocking is to consider the blocking artifact as a discontinuity between two homogeneous regions instead of just two blocks. In this case, we can distribute the abrupt change over a longer distance adaptively. For the highly compressed images/videos, our proposed de-blocking algorithm achieved much better results both visually and quantitatively.

Blockiness metric: One of the most challenging problems in image compression has been how to measure the blocking artifact. Based on the frequency analysis of blocking artifacts, we proposed a new objective blockiness metric. This metric can be used not only in evaluating different de-blocking algorithms, but also in the encoder to keep the blocking artifact under a certain level. The experiment results show that it is more consistent with our subjective evaluation than PSNR/MSE.

Clustering-based de-ringing: Ringing artifacts only appear in edge blocks. In this research, we proposed a new algorithm to identify the edge blocks with only 8 pixels, which reduces the computational cost by up to 8 times. Also, we proposed using the 8 neighboring blocks of an edge block candidate to differentiate an edge block from a

texture block. Since texture blocks have a masking effect on compression artifacts, they are not processed to save computation. Our proposed de-ringing algorithm is clustering-based. It smoothes the perturbation of those pixels within the same cluster so that the strong edges are well preserved.

Implementation on mediaprocessors: Our proposed de-blocking and de-ringing algorithms have been implemented on a mediaprocessor called MAP-CA for real-time postprocessing. We have designed an efficient scheme in managing the data flow between external memory and on-chip memory. Several optimization techniques were utilized to maximize the computing power of MAP-CA. For the 352×288 FOREMAN sequence, more than 300 frames can be de-blocked and de-ringed within one second, which demonstrates the feasibility of real-time postprocessing.

Temporal filtering: We have proposed a down-sampled pattern in motion estimation. This down-sampled pattern can speed up the motion estimation by up to four times. Compared to using all pixels, the MAD using this down-sampled pattern only slightly increased. Also, we have studied different temporal filtering schemes in terms of adaptive weights and filter types. We have found the "corner error" problem caused by the block-based adaptive weights. In consideration of computational cost and video quality, we finally selected the motion-compensated adaptive IIR temporal filter as our temporal filtering scheme.

Algorithm validation: We have modified and applied the paired comparison method in validating our postprocessing algorithms and metrics using multiple groups of images/videos. This modified paired comparison method also avoids pure guesses from the observers and makes their subjective evaluation task much easier. After subjective evaluations, we statistically analyzed the experimental results, which confirmed our

previous conclusions about the proposed postprocessing algorithms and metrics. In addition, the modified experiment methodology can be extended to many other applications.

7.3 FUTURE DIRECTIONS

Postprocessing algorithms: We have proposed a real-time video postprocessing system with several innovative algorithms and metrics. Our proposed algorithms mainly focus on the DCT-based compression schemes. For the DWT-based compression scheme, such as JPEG2000, the major artifact is ringing. Ringing artifacts in JPEG2000-based reconstructed images can spread over a longer distance, i.e., no longer constrained within each block. Our current de-ringing algorithm smoothes ringing artifacts around the edges. To smooth ringing artifacts over a longer distance, a new de-ringing algorithm can be developed in the future.

Objective metrics: We have developed a blockiness metric. As JPEG2000 becomes more popular in the future, it would be desirable if we can design an objective ringing artifact metric. Similarly, there is no good objective metric in measuring temporal noise. Developing an objective metric for temporal noise will benefit many researchers related to image sequences, such as digital video broadcasting, video phone, video conferencing, medical imaging, and defense etc.

Error concealment: Our postprocessing algorithms assume that no errors happen during the network transmission. When the network condition is bad, the packet loss and the bit

errors have to be dealt with. If we can incorporate the error concealment techniques into our postprocessing system, the system would become more robust.

BIBLIOGRAPHY

[1] R. Aldridge, J. Davidoff, M. Ghanbari, D. Hands, and D. Pearson, "Recency effect in the subjective assessment of digitally-coded television pictures," *Proc. IEE, Image processing and its Applications*, vol. 410, pp. 336-339, 1995.

[2] V. R. Algazi, G. E. Ford, H. Chen, and R. R. Estes, "Improving the quality of coded still images by postprocessing," *Proc. SPIE,* vol. 2663, pp. 168-176, 1996.

[3] J. G. Apostolopoulos and N. S. Jayant, "Postprocessing for very low bit-rate video compression," *IEEE Trans. Image Processing*, vol. 8, pp. 1125-1129, 1999.

[4] T. R. Athey and J. E. Hautaluoma, "Effects of applicant overreduction, job status, and job gender stereotype on employment decisions," *Jounal of Social Psychology,* 134, pp. 439-452, 1994.

[5] T. Berger, J. O. Stromberg, and T. Eltoft, "Adaptive regularized constrained least squares image restoration," *IEEE Trans. Image Processing,* vol. 8, pp. 1191-1203, 1999.

[6] J. Biemond, L. Looijenga, D. E. Boekee, and R.H. Plompen, "A pel-recursive Wiener-based displacement estimation algorithm," *Signal Processing*, vol. 13, pp. 399-412, 1987.

[7] K. J. Boo, and N. K. Bose, "A motion-compensated spatio-temporal filter for image sequences with signal-dependent noise," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 287-298, 1998.

[8] J. C. Brailean, R. P. Kleihorst, S. Efstratiadis, A. K. Katsaggelos, and R. L. Lagendijk, "Noise reduction filters for dynamic image sequences: a review," *Proc. IEEE*, vol. 83, pp. 1270-1291, 1995.

[9] Y. Cao, P. P. B. Eggermont, and S. Terebey, "Cross Burg entropy maximization and its application to ringing suppression in image reconstruction," *IEEE Trans. Image Processing*, vol. 8, pp. 286-292, 1999.

[10] E. Chan and S. Panchanaman, "Review of block matching based motion estimation algorithms for video compression," *Proc. IEEE CCECE,* vol. 1, pp. 151-154, 1993.

[11] G. W. Cobb, *Introduction to Design and Analysis of Experiemnts*. NY: Springer, 1998.

[12] F. Dekeyser, P. Bouthemy, and P. Perez, "Spatio-temporal Wiener filtering of image sequences using a parametric motion model," *Proc. IEEE Image Processing*, vol. 1, pp. 208-211, 2000.

[13] C. Derviaux, F. X. Coudoux, M. G. Gazalet, and P. Corlay, "Blocking artifact reduction of DCT coded image sequences using a visually adaptive postprocessing," *Proc. IEEE Image Processing,* vol. 1, pp. 5-8, 1996.

[14] J. N. Driessen and J. Biemond, "Motion field estimation by 2-D Kalman filtering," *Proc. SPIE Visual Commun. And Image Processing,* vol. 1605, pp. 511-521, 1991.

[15] E. Dubois and S. Sabri, "Noise reduction in image sequences using motion compensated temporal filtering," *IEEE Trans. Commun.,* vol. 32, pp. 826-831, 1984.

[16] http://www.equator.com

[17] A. T. Erdem and A. M. Tekalp, "Decision-directed adaptive image restoration using multiple image and blur models," *Proc. IEEE ICCON Control and Applications*, pp. 251-256, 1989.

[18] A. N. Evans and M. S. Nixon, "Biased motion-adaptive temporal filtering for speckle reduction in echocardiography," *IEEE Trans. Medical Imaging,* vol. 15, pp. 39-50, 1996.

[19] P. Farrelle and A. Jain, "Recursive block coding—A new approach to transform coding," *IEEE Trans. Commun.*, vol. 34, pp. 161-179, 1986.

[20] R. C. Feldt and K. L. Witte, "Mnemonic benefits of digit-list organization: Test of the development lag hypothesis of reading retardation," *Journal of Social Psychology,* 149, pp. 459-469, 1988.

[21] M. R. Frater, J. F. Arnold, and A. Vahedian, "Impact of audio on subjective assessment of video quality in videoconferencing applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 1059-1062, 2001.

[22] M. Fuderer, "Ringing artifact reduction by an efficient likelihood improvement method," *Proc. SPIE Sci. Eng. Med. Imaging,* vol. 1137, pp. 84-90, 1989.

[23] W. Gao, C. Mermer, and Y. Kim, "A de-blocking algorithm and a blockiness metric," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 1150-1159, 2002.

[24] W. Gao, C. Mermer, and Y. Kim, "Real-time video postprocessing for de-blocking and de-ringing on mediaprocessors," submitted to *International Journal of Imaging Systems and Technology*, August 2002.

[25] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun.,* vol. 38, pp. 950-953, 1990.

[26] C. Glasman, V. Andronov, A. Bukina, and O. Vasilyev, "Subjective assessment of compression systems by trained and untrained observers," *Proc. IEE*, vol. 447, pp. 476-481, 1997.

[27] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach.* San Francisco, CA: Morgan Kaufmann Publishers, 1996.

[28] B. Hinman, J.Bernstein, and D. Staelin, "Short-space Fourier transform image processing," *Proc. IEEE ICASSP, Acoust. Speech and Signal Processing*, vol. 1, pp. 481-484, 1984.

[29] Y. L. Huang and R. F. Chang, "Adaptive MLP post-processing for block-based coded images," *Proc. IEE, Image and Signal Processing,* vol. 147, pp. 463-473, 2000.

[30] ISO/IEC/JTC1/SC2/WG11, *MPEG Video Simulation Model Three (SM3),* MPEG 90/041, 1990.

[31] ISO/IEC 11172, *Information Technology: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbit/s: Systems/Video/Audio*, 1993.

[32] ISO/IEC 13818-2, *Generic Coding of Moving Pictures and Associated Audio, (MPEG-2), Part2: Video*, 1993

[33] ISO/IEC/JTC1/SC29, *Coding of Moving Pictures and Associated Audio: Recommendation H.262: ISO/IEC 13818*, 1993.

[34] ISO/IEC/JTC1/SC29/WG11, *Test Model 5,* MPEG 93/457, Document AVC-491, 1993.

[35] ISO/IEC/JTC11/SC29/WG11 N2323, *Overview of the MPEG-4 Standard*, 1998.

[36] ISO/IEC/JTC11/SC29/WG11 W2502, *ISO/IEC 14496-2: Final Draft International Standard,* 1998.

[37] ISO/IEC/JTC1/SC29/WG1, *Coding of Still Pictures*, 1999.

[38] ISO/IEC/JTC1/WG8, Joint Photographic Expert Group, *JPEG Technical Specification, Revision 8*, 1990.

[39] ITU-R Recommendation 500-5, *Method for the Subjective Assessment of the Quality of Television Pictures*, 1992.

[40] ITU-T Recommendation H.261, *Video Codec for Audiovisual Services at p×64 kbit/s,* 1990.

[41] ITU-T Recommendation H.263, *Video Coding for Low Bit Rate Communication*, 1995.

[42] ITU-T SG16/Q15, *H.26L Test Model Long Term Number 1 (TML-2)*, 1999.

[43] A. Jacquin, H. Okada, and P. Crouch, "Content-adaptive postfiltering for very low bit rate video," *Proc. IEEE DCC Data Compression*, pp. 111-120, 1997.

[44] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, pp. 1799-1808, 1981.

[45] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[46] B. L. Jones and J. A. Turner, "Subjective assessment of cable impairments on television picture quality," *IEEE Trans. Consumer Electronics,* vol. 38, pp. 850-861, 1992.

[47] J. M. Jou, P. Chen, and J. Sun, "The gray prediction search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 9, pp. 843-848, 1999.

[48] S. A. Karunasekera and N. G. Kingsbury, "A distortion measure for blocking artifacts in images based on human visual sensitivity," *IEEE Trans. Image Processing*, vol. 4, pp. 713-724, 1995.

[49] A. K. Katsaggelos, R. P. Kleihorst, S. N. Efstratiadis, and R. L. Lagendijk, "Adaptive image sequence noise filtering methods," *Proc. SPIE Image Processing*, vol. 1606, pp. 716-727, 1991.

[50] M. Kendall, *Rank Correlation Methods*. NY: Oxford University Press, 1971.

[51] J. Kim, and J. W. Woods, "Spatio-temporal adaptive 3-D Kalman filter for video," *IEEE Trans. Image Processing*, vol. 6, pp. 414-424, 1997.

[52] J. H. Kim, G. W. Song, and S. D. Kim, "Lowpass temporal filter using motion adaptive spatial filtering and its systolic realization," *IEEE Trans. Consumer Electronics*, vol. 38, pp. 452-459, 1992.

[53] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *Proc. Nat. Telecommun. Conf.,* pp. G5.3.1-5.3.5., 1981.

[54] R. L. Lagendijk, J. Biemond, and D. E. Boekee, "Regularized iterative image restoration with ringing reduction," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 36, pp. 1874-1887, 1988.

[55] N. F. Law, W. C. Siu, and D. Feng, "Suppression of ringing artifacts with an adaptive shrinkage algorithm," *Proc. IEEE Communications, Computers and Signal Processing*, pp. 181-184, 1999.

[56] L. W. Lee, J. F. Wang, J. Y. Lee, and J. D. Shie, "Dynamic search-window adjustment and interlaced search for block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 85-87, 1993

[57] A. H. Lettington and Q. H. Hong, "Ringing artifact reduction for Poisson MAP superresolution algorithms," *IEEE Signal Processing Letters,* vol. 2(5), pp. 83-84, 1995.

[58] J. Li and C.-C. J. Kuo, "Coding artifact removal with multiscale postprocessing," *Proc. IEEE ICIP Image Processing*, vol. 1, pp. 45-48, 1997.

[59] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 4, pp. 438-442, 1994.

[60] C. M. Liu, J. Y. Lin, K. G. Wu, and C. N. Wang, "Objective image quality measure for block-based DCT coding," *IEEE Trans. Consumer Electronics*, vol. 43, pp. 511-516, 1997.

[61] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 6, pp. 419-422, 1996.

[62] T. S. Liu and N. Jayant, "Adaptive postprocessing algorithms for low bit rate video signals," *IEEE Trans. Image Processing*, vol. 4, pp. 1032-1035, 1995.

[63] L. Luo, C. Zou, X. Gao, and Z. He, "A new prediction search algorithm for block motion estimation in video coding," *IEEE Trans. Consumer Electronics,* vol. 43, pp. 56-61, 1997.

[64] C.-C. Lu, "Application of short-kernel filter pairs for temporal filtering of image sequences," *IEEE Trans. Consumer Electronics*, vol. 41, pp. 49-52, 1995.

[65] W. E. Lynch, A. R. Reibman, and B. Liu, "Edge compensated transform coding," *Proc. IEEE ICIP Image Processing*, vol. 1, pp. 105-109, 1994.

[66] H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts," *IEEE Trans. Signal Processing*, vol. 46, pp. 1043-1053, 1998.

[67] H. S. Malvar and D. H. Staelin, "The LOT: Transform coding without blocking effects," *IEEE Trans. Acoust. Speech and Signal Processing*, vol. 37, pp. 553-559, 1989.

[68] R. Managuli, G. York, D. Kim, and Y. Kim, "Mapping of two-dimensional convolution on very long instruction word mediaprocessors for real-time performance," *Journal of Electronic Imaging*, vol. 9, pp. 327-335, 2000.

[69] B. Martins and S. Forchhammer, "A unified approach to restoration, deinterlacing and superresolution of MPEG-2 decoded video," *Proc. IEEE ICIP Image Processing*, vol. 1, pp. 1008-1011, 2000.

[70] M. Miyahara and K. Kotani, "Block distortion in orthogonal transform coding—analysis, minimization, and distortion measure," *IEEE Trans. Commun.,* vol. COM-33, pp. 90-96, 1985.

[71] N. Narita, "Subjective-evaluation method for quality of coded images," *IEEE Trans. Broadcasting,* vol. 40, pp. 7-13, 1994.

[72] A. C. Neuman and L. S. Eisenberg, "Evaluation of a dereverberation technique. Speical Issue: Advances in sensory aids for the hearing impaired," *Journal of Communicatin Disorders,* 24, pp. 211-221, 1991.

[73] S. H. Oguz, Y. H. Hu, and T. Q. Nguyen, "Image coding ringing artifact reduction using morphological post-filtering," *IEEE Workshop on Multimedia Signal Processing*, pp. 628-633, 1998.

[74] T. P. O'Rourke and R. L. Stevenson, "Improved image decompression for reduced transform coding artifacts," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 490-499, 1995.

[75] M. K. Ozkan, A. T. Erdem, M. I. Sezan, and A. M. Tekalp, "Efficient multiframe Wiener restoration of blurred and noisy image sequences," *IEEE Trans. Image Processing*, vol. 1, pp. 453-476, 1992.

[76] H. W. Park and Y. L. Lee, "A postprocessing method for reducing quantization effects in low bit-rate moving picture coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 161-171, 1999.

[77] D. Pearson and M. Whybray, "Transform coding of images using interleaved blocks," *Proc. IEE, Radar and Signal Processing*, vol. 131, pp. 466-472, 1984.

[78] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 313-317, 1996.

[79] G. Qiu, "MLP for adaptive postprocessing block-coded images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1450-1454, 2000.

[80] B. Ramamurthi and A. Gersho, "Nonlinear space-invariant postprocessing of block coded images," *IEEE Trans. Acoust. Speech and Signal Processing,* vol. 34, pp. 1258-1268, 1986.

[81] D. E. Riechard, "Paired comparisons intransitivity related to age of subjects," *Educational & Psychological Measurement,* 50, pp. 105-110, 1990.

[82] http://www.rt.com/man/cjpeg.1.html

[83] M. I. Sezan, M. K. Ozkan, and S. V. Fogel, "Temporally adaptive filtering of noisy image sequences using a robust motion estimation algorithm," *Proc. IEEE ICASSP, Acoust. Speech and Signal Processing*, vol. 4, pp. 2429-2432, 1991.

[84] M. Y. Shen and C.-C. J. Kuo, "Real-time compression artifact reduction via robust nonlinear filtering," *Proc. IEEE ICIP Image Processing*, vol. 2, pp. 565-569, 1999.

[85] K. Soh and S. Iai, "A subjective quality assessment method for audiovisual signals based on paired comparison with multiple reference signals," *IEEE Workshop on Multimedia Communications*, pp. 6/3/1-6/3/5, 1994.

[86] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. 33, pp. 888-896, 1985.

[87] K. T. Tan and M. Ghanbari, "Measuring blocking artefacts using harmonic analysis," *Electronics Letters,* vol. 35, pp. 1322-1323, 1999.

[88] J. Y. Tham, S. Ranganath, A. A. Kassim, and S. Y. Tan, "Non-iterative adaptive post-processing for ringing artifact suppression in compressed images," 4th International Conference on Information System, Analysis and Synthesis, vol. 3, pp. 203-210, 1998.

[89] http://ti.com.

[90] G. Tziritas, "Recursive and/or iterative estimation of the two-dimensional velocity field and reconstruction of the three-dimensional motion," *Signal Processing,* vol. 16, pp. 53-72, 1989.

[91] Q. Wang and R. J. Clarke, "Motion estimation and compensation for image sequence coding," *Signal Processing: Image Communication,* vol. 4, pp. 161-174, 1992.

[92] Z. Wang, A. C. Bovik, and B. L. Evans, "Blind measurement of blocking artifacts in images," *Proc. IEEE Image Processing*, vol. 3, pp. 981-984, 2000.

[93] D. L. Wilson, K. N. Jabri, and R. Aufrichtig, "Perception of temporally filtering X-ray fluoroscopy images," *IEEE Trans. Medical Imaging,* vol. 18, pp. 22-31, 1999.

[94] H. R. Wu and M. Yuen, "A generalized block-edge impairment metric for video coding," *IEEE Signal Processing Letters*, vol. 4, pp. 317-320, 1997.

[95] L. Yan, "A nonlinear algorithm for enhancing low bit-rate coded motion video sequence," *Proc. IEEE Image Processing*, vol. 2, pp. 923-927, 1994.

[96] S. Yang, Y. H. Hu, T. Q. Nguyen, and D. L. Tull, "Maximum-likelihood parameter estimation for image ringing-artifact removal," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 963-973, 2001.

[97] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Regularized reconstruction to reduce blocking artifacts of block discrete transform compressed images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 421-432, 1993.

[98] Y. Yang and N. P. Galatsanos, "Removal of compression artifacts using projections onto convex sets and line process modeling," *IEEE Trans. Image Processing*, vol. 6, pp. 1345-1357, 1997.

[99] S. Yao and X. Lin, "Post-processing for removing coding artifacts using edge-preserving regularization," *Proc. International Symposium on Intelligent Multimedia, Video and Speech processing,* pp. 121-124, 2001.

[100] Y. L. You and M. Kaveh, "Ringing reduction in image restoration by orientation-selective regularization," *IEEE Signal Processing Letters,* vol. 3, pp. 29-31, 1996.

[101] Z. Yu, H. R. Wu, S. Winkler, and T. Chen, "Vision-model-based impairment metric to evaluate blocking artifacts in digital video," *Proc. IEEE*, vol. 90, pp. 154-169, 2002.

[102] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 2, pp. 91-95, 1992.

[103] Y. Zhang, R. Pickholtz, and M. Loew, "A new approach to reduce the blocking effect of transform coding," *IEEE Trans. Commun.,* vol. 41, pp. 299-302, 1993.

[104] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Processing*, vol. 9, pp. 287-290, 2000.

VITA

Wenfeng Gao

University of Washington

2003

EDUCATION

2003: Ph.D., Major in Signal and Image Processing, Electrical Engineering
University of Washington, Seattle, WA
Thesis topic: Real-Time Video Postprocessing Algorithms and Metrics
Thesis advisor: Yongmin Kim, Ph.D.

1995: M.S., Major in Communications and Systems, Electronics Engineering
Beijing University of Posts and Telecommunications, Beijing, China
Thesis topic: HDTV Storage Series
Thesis advisor: Liu Cheng, Ph.D.

1992: B.S., Major in Electronic Techniques and Info Systems, Electronics Engineering
Tsinghua University, Beijing, China
Thesis topic: Digital Video Input Converter
Thesis advisor: Wang Yiliang, Ph.D.

HONORS

1. AT&T Sponsored Fellowship for Outstanding Graduate Students in Beijing University of Posts and Telecommunications, 1993.
2. Second-Class Prize in the College Mathematics Contest in Beijing, 1990.
3. Outstanding Undergraduate Student Fellowships in 1988, 1989, and 1990, in Tsinghua University.
4. Hong Kong Tsinghua Alumni Fellowship, 1989, in Tsinghua University
5. Second-Class Prize in National Olympic Mathematics Contest for high school students, 1986.

6. Third-Class Prize in National Olympic Physics Contest for high school students, 1986.

PUBLICATIONS

W. Gao, C. Mermer, and Y. Kim, "A de-blocking algorithm and a blockiness metric for highly compressed images", IEEE Transaction Circuits System Video Technology, vol. 12, pp. 1150-1159, 2002.

W. Gao, C. Mermer, and Y. Kim, "Real-time video postprocessing for de-blocking and de-ringing on mediaprocessors", submitted to International Journal of Imaging Systems and Technology, August 2002.

M. Azimi-Sadjadi, W. Gao, T. Haar, and D. Reinke, "Temporal Updating Scheme for Probabilitic Neural Network with Application to Satellite Cloud Classification—Further Results", IEEE Transactions on Neural Network, vol. 12, pp. 1196-1203, 2001.

B. Tian, M. Azimi-Sadjadi, and W. Gao, "Comparison of two different PNN training approaches for satellite cloud data classification", Proc. IEEE Neural Networks, vol. 6, pp. 3791-3795, 1999.