EFFICIENT DEBLOCKING OF BLOCK-TRANSFORM COMPRESSED IMAGES AND VIDEO USING SHIFTED THRESHOLDING

Alexander Wong and William Bishop
Department of Electrical and Computer Engineering, University of Waterloo
Waterloo, Ontario, Canada
a28wong@engmail.uwaterloo.ca, wdbishop@uwaterloo.ca

ABSTRACT
Image and video compression are active areas of research due to the increasing demand for efficient storage of visual data in a wide variety of fields. Popular image compression schemes are often based on block-transform coding. Block-transform coding is susceptible to blocking artifacts, particularly at low bit-rates due to quantization errors. This paper addresses this problem by presenting an algorithm based on shifted thresholding that reduces blocking artifacts in block-transform compressed images and video. The algorithm is designed to be hardware-efficient and well-suited for parallel hardware implementation, using purely integer computations without the need for integer division. Experimental results demonstrate good subjective quality improvements that are comparable with previous work while maintaining a low computational complexity.

KEY WORDS
deblocking, block-transform coding, shifted thresholding

1 Introduction

Image and video compression are active areas of research given the increasing need for the efficient storage of visual data in a wide variety of fields, ranging from medical imaging to multimedia systems. Popular compression schemes are based on block-transform coding. These include still image compression schemes such as JPEG [1], video compression schemes such as MPEG [2-3] and recent video compression schemes such as H.264/AVC [4]. Block-transform codes that utilize the Discrete Cosine Transform (DCT) are widely used. Blocks are processed independently so block-transform codes are simple to implement and require minimal additional storage.

One of the major drawbacks of block-transform coding is the fact that it also introduces blocking artifacts due to quantization errors at the block boundaries. These artifacts are very noticeable and degrade image quality, particularly when images are compressed at a high compression rate. To deliver high-quality image compression using block-transform codes, the reduction of blocking artifacts is critical.

A large number of methods have been proposed to reduce blocking artifacts in block-transform compressed images and video. These deblocking methods include the following:

1) Projections onto convex sets (POCS) methods [5-7]
2) Spatial block boundary filtering methods [8-9]
3) Wavelet filtering methods [10]
4) Statistical modeling methods [11]
5) Constrained optimization methods [12]

Some of the more effective deblocking techniques in recent development are those based on shifted transforms. The general technique was first introduced by Nosratinia with the re-application of shifted JPEG compressions [13]. This technique has since been modified for improved deringing [14] and it has been applied to wavelet coders to reduce compression artifacts [15]. Shifted transform algorithms have been shown to offer improved performance over traditional techniques based on POCS and wavelet transforms. A structural overview of the deblocking algorithms based on shifted transforms is shown in Figure 1. The decompressed image is shifted based on n shift patterns and the output of these shifts $S_{i,\ldots,n}$ are transformed into another domain using transform operator T. The shifted transforms are then filtered using operator F, inverse transformed with T^{-1} and inverse-shifted based on the corresponding shift pattern. Finally, the images are averaged together to form the final output image. In the simplified version of a shifted JPEG transform [13], the transform operator T is the DCT transform into the spatial frequency domain, the filter operator F is a combined quantization/dequantization process based on a quantization matrix, the averaging process is a simple unweighted average of the inverse-shifted images, and a total of 64 shifts are performed. An improved weighted averaging scheme was proposed [14] to adapt to the input image content.

While available algorithms based on shifted transforms are effective at reducing blocking artifacts, there are some computational drawbacks. While less computationally expensive than methods based on optimization techniques, in its basic implementation, such algorithms

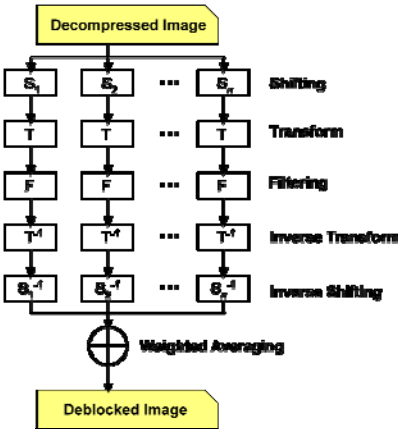

Figure 1. Structure of deblocking algorithms based on shifted transforms

require 64 iterations of DCT transforms, inverse DCT (IDCT) transforms, quantization and dequantization operations per 8×8 block, each requiring fixed-point or floating-point arithmetic operations. The goal of the proposed algorithm is to reduce the computational complexity of the deblocking algorithm while maintaining visual quality.

The main contribution of this paper is the introduction of a hardware-efficient algorithm for image and video deblocking. This algorithm utilizes integer-based shifted transforms and frequency-domain thresholding to enhance deblocking performance. This algorithm also utilizes an improved weighted averaging technique to enhance visual quality. In this paper, the proposed algorithm is described and explained in detail in Section 2. The computational costs are analyzed in Section 3. Finally, experimental results comparing the proposed algorithm to the algorithm proposed by Nosratinia for block-transform compressed images are discussed in Section 4. Finally, conclusions are drawn in Section 5.

2 Proposed Deblocking Algorithm

The proposed deblocking algorithm utilizes the concept of shifted transforms in a practical way to reduce computational complexity while preserving visual quality. Using a 5-stage combination of integer transforms, frequency-domain thresholding, and weighted averaging, the proposed algorithm is hardware-efficient, delivers good performance, and provides a high level of visual quality. Thus, the algorithm is suitable for efficient block artifact reduction in real-time image and video applications. A general overview of the algorithm is shown in Figure 2.

2.1 Shifting Stage

During this stage, the input image is shifted based on four shift patterns: $(\Delta x, \Delta y) = \{(-3,-3),(-1,-1),(1,1),(3,3)\}$,

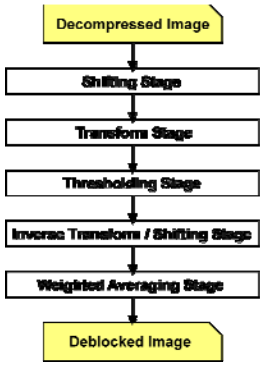

Figure 2. Overview of proposed algorithm

where $(\Delta x, \Delta y)$ represents the shift in the x and y directions. These patterns are illustrated in Figure 3. This produces a total of four shifted images. Therefore, the actual deblocking process is repeated a total of four times, one for each shifted image. Results during testing have shown that this configuration provides good deblocking results for the proposed algorithm while minimizing the number of computations needed. This is significantly less than other algorithms of this type so the computational requirements of this stage are effectively reduced.

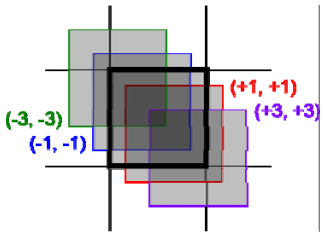

Figure 3. Example of some shift patterns

2.2 Transform Stage

During this stage, the shifted images are transformed to the frequency domain. The basic DCT forward transform on an input block **X** is given by:

$$\mathbf{Y} = (\mathbf{B}\mathbf{X}\mathbf{B}^{\mathbf{T}}) \quad (1)$$

where **X** is the input block, **B** is the DCT transform matrix, and $\mathbf{B}^{\mathbf{T}}$ is the transpose of the DCT transformation matrix.

To reduce the computational complexity required by the proposed algorithm, an integer transform is proposed based on a scaled integer approximation of the DCT transform in the form of:

$$\mathbf{M} = \text{Round}(\alpha\mathbf{B}) \quad (2)$$

where α is the scaling coefficient and **B** is the DCT transformation matrix. When an image is transformed from the spatial domain to the frequency domain using the DCT transform, there is an approximate magnitude gain of $8\alpha^2$ if the ceiling of the integer approximation is taken. In the case of 8 bits per channel, which is common in current image representations, the total number of bits needed becomes $\log_2(8\alpha^2)+8$ bits. As most microprocessors are capable of 32-bit integer computations, the maximum allowed value of α that is a power of 2 to avoid the need for division is α=1024, as $\log_2(8(1024)^2)+8=31$ bits. For the proposed algorithm, a value of α=512 is used and so a shift of 18 bits is needed to compensate for the scaling. To reduce the number of multiplications needed for the transform, a number of elements in the matrix are set to powers of 2 while others are readjusted to compensate for this change. The proposed 8×8 integer transform matrix **M** is given by:

$$M = \begin{bmatrix} 181 & 181 & 181 & 181 & 181 & 181 & 181 & 181 \\ 256 & 206 & 128 & 64 & -64 & -128 & -206 & -256 \\ 256 & 64 & -64 & -256 & -256 & -64 & 64 & 256 \\ 206 & -64 & -256 & -128 & 128 & 256 & 64 & -206 \\ 181 & -181 & -181 & 181 & 181 & -181 & -181 & 181 \\ 128 & -256 & 64 & 206 & -206 & -64 & 256 & -128 \\ 64 & -256 & 256 & -64 & -64 & 256 & -256 & 64 \\ 64 & -128 & 206 & -256 & 256 & -206 & 128 & -64 \end{bmatrix} \quad (3)$$

The transformation matrix can be further decomposed into a scalar multiplication of two matrices, **P** and **E**. The final forward transform is given by the following equation:

$$\mathbf{Y} = (\mathbf{P}\mathbf{X}\mathbf{P}^{\mathbf{T}}) \otimes (\mathbf{E} \otimes \mathbf{E}^{\mathbf{T}}) >> (\log_2(\alpha^2)) \quad (4)$$

where ⊗ indicates a scalar multiplication, **X** is the input block, **P** is the integer transform matrix, $\mathbf{P}^{\mathbf{T}}$ is the transpose of the integer transformation matrix, $(\mathbf{E} \otimes \mathbf{E}^{\mathbf{T}})$ is the scaling transformation matrix, α is the scaling coefficient, and >> indicates a logical bit shift right of the appropriate size. For the proposed algorithm, a shift of 18 bits is used since $\alpha^2 = 2^{18}$. **P** and $(\mathbf{E} \otimes \mathbf{E}^{\mathbf{T}})$ are given by:

$$P = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 256 & 206 & 128 & 64 & -64 & -128 & -206 & -256 \\ 256 & 64 & -64 & -256 & -256 & -64 & 64 & 256 \\ 206 & -64 & -256 & -128 & 128 & 256 & 64 & -206 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 128 & -256 & 64 & 206 & -206 & -64 & 256 & -128 \\ 64 & -256 & 256 & -64 & -64 & 256 & -256 & 64 \\ 64 & -128 & 206 & -256 & 256 & -206 & 128 & -64 \end{bmatrix} \quad (5)$$

$$(E \otimes E^{T}) = \begin{bmatrix} 32761 & 181 & 181 & 181 & 32761 & 181 & 181 & 181 \\ 181 & 1 & 1 & 1 & 181 & 1 & 1 & 1 \\ 181 & 1 & 1 & 1 & 181 & 1 & 1 & 1 \\ 181 & 1 & 1 & 1 & 181 & 1 & 1 & 1 \\ 32761 & 181 & 181 & 181 & 32761 & 181 & 181 & 181 \\ 181 & 1 & 1 & 1 & 181 & 1 & 1 & 1 \\ 181 & 1 & 1 & 1 & 181 & 1 & 1 & 1 \\ 181 & 1 & 1 & 1 & 181 & 1 & 1 & 1 \end{bmatrix} \quad (6)$$

The transformation matrices **P**, $\mathbf{P}^{\mathbf{T}}$, and $(\mathbf{E} \otimes \mathbf{E}^{\mathbf{T}})$ are static. These matrices may be pre-computed to deliver improved computational performance. The use of an integer transform allows the entire deblocking process to be implemented using purely integer computations and no division operations, thereby reducing the computational complexity required to perform this stage of the algorithm.

2.3 Thresholding Stage

During this stage, the transformed images are subjected to a threshold in the spatial frequency domain using a threshold matrix **T**, which is given by:

$$\mathbf{T} = \mathbf{Q} >> 1 \quad (7)$$

where ">>1" represents a logical bit shift right of 1 bit and **Q** is the quantization matrix used to compress the image. The **Q** matrix is easily retrieved from the media itself as it is used to decompress the image.

All frequency coefficients below the corresponding threshold value in the threshold matrix are set to zero. The threshold matrix used in the algorithm is a scaled integer approximation of the quantization matrix used to decompress the image. This threshold matrix allows the decompressed image to be adaptively filtered depending on the image quality. This approach is designed to attenuate the high frequencies. Shifting the image according to several shifting patterns and then applying block-based threshold filtering is similar to applying threshold filtering on the original block boundaries at various positions. Thus, this approach makes use of the correlation between adjacent blocks and effectively reduces the blocking artifacts exhibited at the boundaries.

2.4 Inverse Transform and Shifting Stage

After the transformed images have been threshold filtered, the inverse transform of input block **Y** is given by:

$$\mathbf{X} = (\mathbf{P}^{\mathbf{T}}(\mathbf{Y} \otimes (\mathbf{E} \otimes \mathbf{E}^{\mathbf{T}}))\mathbf{P}) >> (\log_2(\alpha^2)) \quad (8)$$

where **Y** is the input block, **P** is the integer transform matrix, $\mathbf{P}^{\mathbf{T}}$ is the transpose of the integer transformation matrix, $(\mathbf{E} \otimes \mathbf{E}^{\mathbf{T}})$ is the scaling transformation matrix, α is

the scaling coefficient, and >> indicates a logical bit shift right that is used to compensate for scaling done in the inverse transform.

Like the integer transform, a value of α=512 is used and so a logical bit shift right of 18 bits is needed to compensate for the scaling in the integer inverse transform. Finally, the shifted images are inverse-shifted based on their corresponding shift pattern such that all the images are aligned together.

2.5 Weighted Averaging Stage

Once the images have been inverse transformed and inverse shifted, an unweighted averaging process is used to combine the images, as given by:

$$\mathbf{I}_{\mathbf{deblocked}}(x,y)=\left(\sum_{i=1}^{4}\mathbf{I}_i(x,y)\right)>>2 \tag{9}$$

where x and y are the x and y coordinates relative to the image, $\mathbf{I}_i$ is the image that was shifted using pattern i, and ">>2" denotes a bit shift right of 2 bits.

While the thresholding process reduces blocking artifacts, it also introduces a slight degradation to the areas that are not subject to blocking artifacts. To remedy the introduction of this degradation, a number of images with different shift patterns are threshold filtered, effectively introducing a different degradation to each image. Averaging across the set of filtered images effectively reduces the degradation while preserving the image signal.

To avoid unnecessary blurring at pixels that are not near the block boundaries, the deblocked averaged image $\mathbf{I}_{\mathbf{deblocked}}$ and the input image with blocking artifacts $\mathbf{I}_{\mathbf{blocked}}$ are combined using a distance-weighted averaging process to produce the final deblocked output image. The weights used in the weighted averaging process are based on the Euclidean distance between the pixel under evaluation and the center of its corresponding block. The weights are given by the following equation:

$$\mathbf{W}_{\mathbf{a}}(x,y)=\text{Round}\left((\sigma-\beta)\left(\frac{\mathbf{D}(x,y)}{\text{Max}(\mathbf{D}(\text{x},\text{y}))}\right)+\beta\right) \tag{10}$$

where x and y represent the x and y coordinates relative to the block respectively, β and σ represent the lower bound and upper bound of the weight range respectively, and $\mathbf{D}(x, y)$ is the Euclidean distance. $\mathbf{D}(x, y)$ is given by the following equation:

$$\mathbf{D}(x,y)=\sqrt{(x-c)^2+(y-c)^2} \tag{11}$$

where x and y are the x and y coordinates relative to the block respectively, and c is the closest pixel from the center region of the block. In the case of an 8×8 block, the center region is represented by the 2×2 block of pixels at the center of the block. For the proposed algorithm, a weight range of (β, σ) = (100, 256) is used. The weights for $\mathbf{I}_{\mathbf{blocked}}$ are then given by:

$$\mathbf{W}_{\mathbf{o}}=\sigma-\mathbf{W}_{\mathbf{a}} \tag{12}$$

where σ is the upper bound of the weight range. Both weight matrices $\mathbf{W}_{\boldsymbol{a}}$ and $\mathbf{W}_{\boldsymbol{o}}$ are static. These matrices are pre-computed to improve performance. The final pixel value in the output image block is given by:

$$\mathbf{I}_{\mathbf{output}}(x,y)=\begin{pmatrix}\mathbf{W}_{\mathbf{o}}(x,y)\mathbf{I}_{\mathbf{blocked}}(x,y)\\+\mathbf{W}_{\mathbf{a}}(x,y)\mathbf{I}_{\mathbf{deblocked}}(x,y)\end{pmatrix}>>\log_2(\sigma) \tag{13}$$

where $\mathbf{I}_{\mathbf{deblocked}}$ is the average deblocked image, $\mathbf{I}_{\mathbf{blocked}}$ is the input image with blocking artifacts, $\mathbf{W}_{\boldsymbol{a}}$ and $\mathbf{W}_{\boldsymbol{o}}$ are the weight matrices for $\mathbf{I}_{\mathbf{deblocked}}$ and $\mathbf{I}_{\mathbf{blocked}}$ respectively, and σ is the upper bound of the weight range. For a weight range of (β, σ) = (100, 256), a logical bit shift right of 8 bits is performed.

The distance-weighted average is designed such that pixels near the block boundary where blocking artifacts occur are more dependent on the average deblocked image while pixels near the center where blocking artifacts do not occur are more dependent on the input image.

3 Computational Costs

A comparison between the computational complexity of the proposed algorithm and the original Nosratinia algorithm for JPEG compressed images illustrates the clear performance benefits of the proposed algorithm. The complexity of the proposed algorithm can be measured based on the cost of the integer transform and inverse transform operations. The integer transform used by the proposed algorithm requires approximately 6.25 times fewer multiplications than the standard DCT operation. This reduction is the result of multiplications being replaced with bit shifts. Furthermore, the DCT and IDCT are repeated 63 times in the original Nosratinia algorithm, while the integer transform and inverse transform are repeated only 4 times in the proposed algorithm. Therefore, ignoring the computational complexity differences between integer and fixed-point / floating-point operations, the proposed algorithm uses approximately 1% of the multiplications and 6.3% of the additions needed for the Nosratinia algorithm. This reduction in computational complexity makes the proposed algorithm well-suited for the real-time deblocking of compressed image and video sources.

4 Experimental Results

A comparison of the proposed algorithm and the original Nosratinia algorithm designed for JPEG compressed images [13] was conducted by both quantitative and qualitative means. The algorithms were tested on 5 different JPEG-compressed images and video frames from 2 different MPEG-compressed video clips. For a quantitative comparison, the PSNR of the compressed images and video frames were measured. The results are shown in Table 1. It is observed from the quantitative measurements that the PSNR gains using the proposed algorithm are comparable with the original Nosratinia algorithm, despite requiring significantly fewer computations. The subjective results for Lena, Susi, and Tennis after the application of the algorithm are shown in Figures 4, 5, and 6 respectively. From a subjective comparison of the resulting images, it can be observed that the overall quality of the images and video frames produced using the proposed algorithm is noticeably superior to the original decompressed sources. Blocking artifacts have been reduced while preserving edges. The visual quality is comparable to that produced by the Nosratinia algorithm.

TABLE 1
IMPROVEMENTS IN PSNR ON COMPRESSED IMAGE AND VIDEO SOURCES

Image / Video Frame	PSNR (in dB)	PSNR Gain (dB)	
		Proposed Algorithm	Nosratinia Algorithm [13]
Images			
Lena	31.13	0.94	1.06
Elaine	30.48	0.63	0.69
Mandrill	24.06	0.28	0.32
Peppers	31.16	0.84	0.93
Boat	28.85	0.74	0.76
Video Frames			
Susi	34.80	0.91	0.91
Tennis	24.32	0.33	0.34

5 Conclusions

In this paper, we have introduced a new method for deblocking compressed image and video sources based on the concept of shifted thresholding. The algorithm reduces blocking artifacts in a computationally effective manner. Experimental results show that overall visual quality is noticeably improved when compared to the decompressed image or video sources. This highly-efficient algorithm is competitive with the original Nosratinia algorithm. It is our belief that this method is more suitable for implementation in digital multimedia systems and consumer devices.

Acknowledgements

This research has been sponsored in part by Epson Canada and the Natural Sciences and Engineering Research Council of Canada.

References

[1] G. Wallace, The JPEG Still Picture Compression Standard," *Communications of the ACM*, *34*(4), 1991, 30-34.

[2] ISO/IEC 11172, Coding of Moving Pictures and Associated Audio for Digital Storage Media Up to About 1.5 Mbits/s – Part 2: Video, 1993.

[3] ISO/IEC 13818-2, Generic Coding of Moving Pictures and Associated Audio Information – Part 2: Video, 1994.

[4] Joint Video Team of ITU-T and ISO/IEC JTC 1, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), 2003.

[5] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, Projection-Based Spatially Adaptive Reconstruction of Block-Transform Compressed Images, *Transactions on Image Processing*, *4*, 1995, 896-908.

[6] C. Weerasinghe, A. Liew, and H. Yan, Artifact Reduction in Compressed Images Based on Region Homogeneity Constraints Using the Project Onto Convex Sets Algorithm, *IEEE Transactions on Circuits and Systems for Video Technology*, *12*(10), 2002, 891-897.

[7] H. Paek, R. C. Kim, and S. Lee, On the POCS-Based Postprocessing Technique to Reduce Blocking Artifacts in Transform Coded Images, *IEEE Transactions on Circuits and Systems for Video Technology*, *8*(3), 1998, 358–367.

[8] ISO/IEC 14496-2, MPEG-4 Video Verification Model Version 18.0, 2001.

[9] J. Chou, M. Crouse, and K. Ramchandran, A Simple Algorithm for Removing Blocking Artifacts in Block-Transform Coded Images, *IEEE Signal Processing Letters*, *5*(2), 1998, 33-35.

[10] Z. Xiong, M. T. Orchard, and Y. Q. Zhang, A Deblocking Algorithm for JPEG Compressed Images Using Overcomplete Wavelet Representations, *IEEE Transactions on Circuits and Systems for Video Technology*, *7*, 1997, 433-437.

[11] T. O'Rourke and R. L. Stevenson, "Improved Image Decompression for Reduced Transform Coding Artifacts, *IEEE Transactions on Circuits and Systems for Video Technology*, *5*(8), 1995, 298–304.

[12] S. Hong, Y.H. Lee, and W.C. Siu, Subband Adaptive Regularization Method for Removing Blocking Artifacts, *Proceedings of the International Conference on Image Processing*, 1995, Washington, D. C., 523-527.

[13] A. Nosratinia, Enhancement of JPEG-Compressed Images by Re-Application of JPEG, *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, *27*, 2001, 69-79.
[14] R. Samadani, A. Sundararajan, and A. Said, Deringing and Deblocking DCT Compression Artifacts with Efficient Shifted Transforms, *Proceedings of the International Conference on Image Processing*, 2004, Singapore, Republic of Singapore, 1799-1802.
[15] A. Nosratinia, "Postprocessing of JPEG-2000 Images to Remove Compression Artifacts," *IEEE Signal Processing Letters*, *10*(10), 2003, 296-299.

Figure 4. Left: JPEG-compressed 512×512 Lena at PSNR of 31.13 dB
Center: Deblocked image using proposed algorithm
Right: Deblocked image using original Nosratinia algorithm

Figure 5. Left: MPEG-compressed Susi video frame at PSNR of 24.32 dB
Center: Deblocked image using proposed algorithm
Right: Deblocked image using original Nosratinia algorithm

Figure 6. Left: MPEG-compressed Tennis video frame at PSNR of 24.32 dB
Center: Deblocked image using proposed algorithm
Right: Deblocked image using original Nosratinia algorithm