Image Filtering in the Compressed Domain

Jayanta Mukherjee[1] and Sanjit K. Mitra[2]

[1] Dept. of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, India
[2] Dept. of Electrical and Computer Engineering
University of California, Santa Barbara, USA
jay@cse.iitkgp.ernet.in, mitra@ece.ucsb.edu

Abstract. Linear filtering of images is usually performed in the spatial domain using the linear convolution operation. In the case of images stored in the block DCT space, the linear filtering is usually performed on the sub-image obtained by applying an inverse DCT to the block DCT data. However, this results in severe blocking artifacts caused by the boundary conditions of individual blocks as pixel values outside the boundaries of the blocks are assumed to be zeros. To get around this problem, we propose to use the symmetric convolution operation in such a way that the operation becomes equivalent to the linear convolution operation in the spatial domain. This is achieved by operating on larger block sizes in the transform domain. We demonstrate its applications in image sharpening and removal of blocking artifacts directly in the compressed domain.

1 Introduction

Filtering of images is required in various applications of image processing, such as noise removal, sharpening and edge extraction, anti-aliasing operations in image resizing, etc. These operations are usually performed in the spatial domain. As in many cases the images are stored in a compressed format, it is of interest to perform these operations directly in the compressed domain. This reduces the computational overhead associated with decompression and compression operations with the compressed stream. As DCT based JPEG standard is widely used for image compression, a number of algorithms have been advanced to perform various image processing operations in the DCT space [1]-[8].

In a classic work [9], Martucci has shown how the *convolution-multiplication* property of the *Fourier transform* could be extended to the class of *trigonometric transforms*, namely the *discrete cosine* and *sine transforms*. He has pointed out that like the *discrete Fourier transform* where *circular convolution* holds the convolution-multiplication property, in trigonometric transforms *symmetric convolutions* have similar properties. Hence, a class of linear filtering operations which could be mapped to symmetric convolutions of images, could be easily performed in the transform domain. In our work, we restrict our discussion to images in the *type-II block DCT* format. We demonstrate here that the *Gaussian filtering* could be performed in this domain and show its application in various

P. Kalra and S. Peleg (Eds.): ICVGIP 2006, LNCS 4338, pp. 194–205, 2006.

image processing applications such as smoothing of the blocking artifacts of highly compressed data, image sharpening, edge extraction, etc.

In the case of images stored in the block DCT space, linear filtering is usually performed on the sub-image obtained by applying an inverse DCT to the block DCT data. However, this results in severe blocking artifacts caused by the boundary conditions of individual blocks as pixel values outside the boundaries of the blocks are assumed to be zeros. On the other hand, the symmetric convolution has an advantage over the linear convolution operation in this regard, as in this case due to smooth transitions in the boundaries, the strength of blocking artifacts is reduced in the processed image. Moreover it does not provide similar boundary conditions of individual blocks what would have been there in the case of linear convolution of images in the spatial domain. Our objective in this work is to perform filtering with the blocks in the compressed domain in such a way that the symmetric convolution operation becomes equivalent to the linear convolution in the spatial domain. This has been achieved by operating on larger block sizes in the transform domain. To this end, we have used composition and decomposition of the DCT blocks using the spatial relationship of the DCT coefficients developed by Jiang and Feng [10]. It may be noted that in [8], Shin and Kang used the convolution-multiplication property of the DCT for designing anti-aliasing low pass filters for the purpose of image resizing. However, the approach was restricted for *image halving* and *image doubling* operations with the filtered output in the *type-I DCT* space. On the other hand, in our work *given an image in the type-II block-DCT space, the output is also of the same type.*

2 Symmetric Convolution and Convolution-Multiplication Properties in the DCT Domain

In this section we briefly review the concept of symmetric convolution and its equivalent operation in the DCT domain [9]. For the sake of brevity, we restrict our discussion to the 1-D case, as the concepts are trivially extended to 2-D.

Let $h(n), 0 \leq n \leq N$, be a sequence of length $N+1$. Its N-point 1-D *type-I* DCT is defined by

$$C_{1e}\{h(n)\} = H_I^{(N)}(k) = \sqrt{\tfrac{2}{N}}\alpha(k)\sum_{n=0}^{N} h(n)\cos(\tfrac{n\pi k}{N}), \quad 0 \leq k \leq N. \tag{1}$$

Likewise, $x(n), 0 \leq n \leq N-1$, be a sequence of length N. Its N-point 1-D *type-II* DCT is defined by

$$C_{2e}\{x(n)\} = X_{II}^{(N)}(k) = \sqrt{\tfrac{2}{N}}\alpha(k)\sum_{n=0}^{N-1} x(n)\cos(\tfrac{(2n+1)\pi k}{2N}), \quad 0 \leq k \leq N-1. \tag{2}$$

In Eqs. (1) and (2), $\alpha(k)$ is $\sqrt{\frac{1}{2}}$ for $k = 0$, otherwise its value is 1.

It should be noted that the type-I $N-$point DCT is defined with $(N+1)$ samples, whereas, the type-II DCT is defined with N samples. They can be considered as *generalized discrete Fourier transforms* (GDFT's) [9] of *symmetrically*

extended sequences. After symmetric extensions, the resulting periods in both the cases are $2N$. For the type-I DCT, the *symmetric extension* of the $(N+1)$ samples is carried out as follows:

$$\hat{h}(n) = \begin{cases} h(n), 0 \le n \le N, \\ h(2N-n), N+1 \le n \le 2N-1, \end{cases} \tag{3}$$

whereas, for the type-II DCT the symmetric extension of the length-N input sequence is carried out as follows (before applying GDFT to it):

$$\hat{x}(n) = \begin{cases} x(n), 0 \le n \le N-1, \\ x(2N-1-n), N \le n \le 2N-1. \end{cases} \tag{4}$$

In this paper we refer the symmetric extensions of Eqs. (3) and (4) as *type-I* and *type-II symmetric extensions*, respectively.

The *symmetric convolution* of two finite-length sequences of appropriate lengths is nothing but the *periodic convolution* of their *symmetrically extended sequences* (having the same periods). The output resulting from this operation is observed for a specific interval. This operation is illustrated below.

Let $x(n), 0 \le n \le N-1$, and $h(n), 0 \le n \le N$, be two sequences. Denote the *type-II symmetric extension* of $x(n)$ as $\hat{x}(n)$ and the *type-I symmetric extension* of $h(n)$ as $\hat{h}(n)$. *Symmetric convolution* of $x(n)$ and $h(n)$, denoted by the operator Ⓢ is then defined as follows.

$$\begin{aligned} y(n) &= x(n) Ⓢ h(n) \\ &= \hat{x}(n) \circledast^{2N} \hat{h}(n) \\ &= \Sigma_{k=0}^{n} \hat{x}(k)\hat{h}(n-k) + \Sigma_{k=n+1}^{2N-1} \hat{x}(k)\hat{h}(n-k+2N), \\ &\qquad\qquad 0 \le n \le N-1, \end{aligned} \tag{5}$$

where the operator $\circledast^{2N}$ denotes the $2N$-point *circular* convolution.

In [9] Martucci has discussed how *convolution-multiplication properties* hold for trigonometric transforms with symmetric convolution. In particular, with respect to Eq. (5) this property is given by:

$$C_{2e}\{x(n) Ⓢ h(n)\} = C_{2e}\{x(n)\} C_{1e}\{h(n)\}. \tag{6}$$

It should be noted that as the N-th coefficient of type-II DCT of $x(n)$ (denoted by $X_{II}^{(N)}(N)$) is zero, only N multiplications are involved in Eq. (6).

The above concepts could easily be extended to 2-D. Here, the $M \times N$-point type-I 2-D DCT is defined over $(M+1) \times (N+1)$ samples and the type-II 2-D DCT is defined over $M \times N$ samples. These can also be derived from the *2-D GDFT* defined over symmetrically extended sequences as discussed earlier. We denote the type-I and type-II DCTs of $x(m,n)$ by $C_{1e}\{x(m,n)\}$ and $C_{2e}\{x(m,n)\}$, respectively. Similar convolution multiplication properties hold also in 2-D and a trivial extension of Eq. (6) to 2-D is as follows:

$$C_{2e}\{x(m,n) Ⓢ h(m,n)\} = C_{2e}\{x(m,n)\} C_{1e}\{h(m,n)\} \tag{7}$$

It should be noted here that Eq. (7) involves $M \times N$ multiplications.

3 Filtering in the Block DCT Space

The convolution-multiplication property as expressed by Eq. (7) has a particular significance in its application to filtering of images represented in the *type-II block DCT* space. Given the *type-I DCT* of the *impulse response* of a filter, one can easily compute the filtered output using Eq. (7). In such a situation both the input (image) and the output (filtered image) remain in the type-II DCT space. As different compression schemes such as JPEG and MPEG have adopted the type-II DCT representation of images and videos, filtering in the transform domain itself can be performed directly using Eq. (7). However, this filtering operation in the transform domain is equivalent to symmetric convolution in the spatial (time) domain of an image (signal). Hence, only filters with impulse responses that are even functions can be supported by this operation. For performing the symmetric convolution (in this case), specifications for the first (positive) quadrant (half) of the spatial (time) domain are only required. This also reduces the storage requirement of the filter.

A Filtering Example. We illustrate next the implementation of a Gaussian filter in the block DCT domain. A *Gaussian filter* has an impulse response that is an *even* function and performs *low-pass filtering*. For a symmetric convolution in 2-D, specifications in the first quadrant of the discretized image space are required. The 2-D Gaussian impulse response in the first quadrant of the spatial domain is given by

$$h(m,n) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{m^2}{\sigma_x^2}+\frac{n^2}{\sigma_y^2})}, \quad 0 \le m \le M, 0 \le n \le N. \tag{8}$$

Let $H_I(k,l), 0 \le k \le M, 0 \le l \le N$, denote the type-I DCT of $h(m,n)$. Given a $M \times N$ type-II DCT block $\mathbb{B} = \{B_{II}(k,l), 0 \le k < M, 0 \le l < N\}$, the output $\mathbb{F}$ in the transform domain is then computed as follows:

$$\mathbb{F} = \{F_{II}(k,l) = B_{II}(k,l).H_I(k,l), 0 \le k < M, 0 \le l < N\}. \tag{9}$$

In our work we have assumed $\sigma_x = \sigma_y$, and henceforth both are referred to as σ.

Boundary Conditions. Filtering in the spatial domain is implemented by a linear convolution of an image with a finite length impulse response with the boundaries of the image zero-padded. Because of the sharp transitions at the boundaries, blocking artifacts occur at the boundaries. In a symmetric convolution, symmetric extensions at the boundaries of a block results in smoother transitions at the boundaries. As a result, the symmetric convolution results in better boundary conditions than that obtained using the linear convolution. To arrive at a smoother transition at the boundaries, with the help of Eq. (9), the symmetric convolution is applied to an independent block (of size 8×8 in the present case). It is of interest to compare its performances to that of a linear convolution operation. In Table 1 the PSNR values of the images obtained using

the symmetric convolution operation have been computed by considering the images obtained via a linear convolution as the reference. It is observed that the quality of the filtered images obtained by symmetric convolution of 8×8 blocks suffer heavily due to blocking artifacts. This is also reflected by the low PSNR values in Table 1. One of the objectives of the present work is to outline

Table 1. PSNR Values of Gaussian Filtered Images

	PSNR (dB)		
σ	Pepper	Mandrill	Lena
2.0	28.73	28.89	28.65
3.0	26.10	27.12	26.43
4.0	24.91	26.17	25.41
5.0	24.85	26.09	25.40

filtering in the block DCT domain so that the operation becomes equivalent to filtering of the whole image by a linear convolution with the impulse response of the filter. Consider a block of size N in 1-D. Let the *effective length of the impulse response*[1] be K (beyond $(K-1)$-th position sample values are zero). Hence, the convolved output response (through symmetric or circular convolution) between the sample positions K and $(N-1-K)$ will be the as same as those obtained from linear convolution with the complete input sequence. This implies that the smaller the value of K, the closer the result is to that of linearly convolved output. This may be observed from the PSNR values in Table 1. Increasing values of σ make the *effective length* of the filter longer. For example, *effective half length* of a Gaussian impulse response with σ will be around 2σ. As can be seen in Table 1, PSNR values get degraded with increasing σ. Hence, one should keep *effective half length* small to get performance similar to that of the corresponding linear convolution operation. However, small value of N (e.g. 8) places a severe restriction on the filter design. In addition, filter response also deviates largely from its desirable characteristics due to the truncation errors. As a result, one should consider larger block sizes for this purpose. One could form blocks of larger sizes from smaller sizes directly in the transform domain following the technique of Jiang and Feng [10]. After performing filtering operation with larger blocks, the filtered blocks are decomposed into their original sizes to get back the results in the specified block DCT domain.

Composition and Decomposition of the DCT Blocks. For convenience, we discuss the spatial relationships of the DCT blocks in 1-D. Let $C_i^{(N)}, 0 \leq i \geq M-1$, the i-th N-point DCT block of a sequence $\{x(n)\}, n = 0, 1, ..., M \times N - 1$. Jiang and Feng [10] showed that a block DCT transformation is nothing but an orthonormal expansion of the sequence $\{x(n)\}$ with a set of $M \times N$ basis vectors, each of which is derived from the basis vectors of N-point DCT. Hence, there exists

[1] For symmetric convolution, impulse response is defined for positive half only. In this case we refer K as the *effective half filter length.*

an invertible linear transformation from M blocks of N-point DCT transform to the usual MN-point DCT transform. In other words, for a sequence of N-point DCT blocks $\{C_i^{(N)}\}, i = 0, 1, ...M-1$, the corresponding composite DCT $C^{(MN)}$ (MN-point DCT), there exists a matrix $A_{(M,N)}$ of size $MN \times MN$ such that[2]

$$C^{(M.N)} = A_{(M,N)} \cdot [C_0^{(N)} C_1^{(N)} .....C_{M-1}^{(N)}]^T \tag{10}$$

The above analysis in 1-D can also be extended to 2-D. For details one may refer the discussion made in [6]. It should be noted that the conversion matrices and their inverses are sparse [10]. Hence, lesser number of multiplications and additions of two matrices is required than those required in usual matrix multiplications.

4 Filtering with Block Composition and Decomposition

In our technique $L \times M$ number of 8×8 blocks are merged into a single block (say, $\mathbb{B}^{(LN \times MN)}$). Then, the resulting block is subjected to the filtering operation. Let $h(m,n), 0 \leq m \leq 8L, 0 \leq n \leq 8M$, be the filter response specified in the first quadrant of the image space. Let $H_I(k,l), 0 \leq k \leq 8L, 0 \leq l \leq 8M$, denote the type-I DCT of $h(m,n)$ (i.e., $C_{1e}(h(m,n)) = H_I(k,l)$). The filtered response (say, $\mathbb{B}_f^{(LN \times MN)}$)is computed by multiplying an element of $\mathbb{B}^{(LN \times MN)}$ with the corresponding element of $\mathbb{H}$ (refer Eq. (9)). Finally, $\mathbb{B}_f^{(LN \times MN)}$ is decomposed into $L \times M$ blocks (of size 8×8). We refer this algorithm in our work as the *Block_Filtering_on_Composition_Decomposition* (BFCD) algorithm.

We have performed the same Gaussian filtering given by Eq. (9) using the BFCD algorithm. Table 2 lists the PSNR values of the images obtained using the BFCD algorithm with the Gaussian filtered image obtained via linear convolution in the spatial domain as the reference for different values of L and M. In our simulations we have kept the values of L and M same. It can be seen that the PSNR values increases with increasing block sizes. It is also observed that blocking artifacts are also less visible in the filtered images. One may interestingly note that it is expected that the larger the block size, the closer the result is to that obtained using the convolution. However, in Table 2 it can be seen that for $L(= M) = 4$, the PSNR values are lower for all values of σ compared to that obtained in the case of neighboring L and M values. In fact, the degradation in the PSNR values happens when the block sizes are integral multiples of the image size. In that case, a block at the right and bottom boundaries gets totally fitted within the image. Hence the boundary effect of symmetric convolution is felt from all sides of the block. When the block sizes are not integral multiple of image sizes, a boundary block (containing right and bottom margin of the image) contains a fraction of the image pixel data and the rest are assumed to be zeros. In such a case, the distortion due to the boundary conditions of symmetric convolution is less.

[2] The transpose of a matrix X is denoted here by X^T.

Table 2. PSNR values obtained using Gaussian filtering with BFCD

L(=M)	σ	PSNR (dB)			Per pixel	Equivalent number
		Pepper	Mandrill	Lena	Computational cost	of additions per pixel
2	2.0	29.68	29.65	29.72	28 M + 32 A	116
	3.0	27.46	28.10	27.93		
	4.0	27.00	27.94	27.67		
3	2.0	34.44	32.50	32.59	46.17 M + 53.33 A	191.84
	3.0	31.45	31.01	30.70		
	4.0	30.70	30.87	30.38		
4	2.0	30.56	30.08	30.32	71.5 M + 96 A	310.5
	3.0	28.55	28.48	28.64		
	4.0	28.20	28.32	28.47		

4.1 Computational Costs

In this section we compare the computational costs of the transform domain technique with that of the spatial domain technique. If the costs of a single multiplication and a single addition are $\mathcal{M}$ and $\mathcal{A}$, respectively, then the total cost for an operation requiring a number of multiplications and b number of additions is $a\mathcal{M} + b\mathcal{A}$. We have also considered the cost of a multiplication operations is *three times* of the cost of addition for providing a combined cost measure following the similar practice used in an earlier work [13].

Computational Cost with Spatial Domain Operations. Costs associated with spatial domain operation are due to computations involved in: (i) IDCT of individual blocks, (ii) Convolution of the image with a $(2K-1) \times (2K-1)$ mask, and (iii) DCT of individual blocks. It should be noted that the effective filter size is $(2K-1) \times (2K-1)$. Outside this support, the filter's impulse response samples are taken as zeros. As a result, per pixel, one has to perform $(2K-1)^2$ multiplications and $((2K-1)^2 - 1)$ additions. However, exploiting the symmetry in the impulse response in the spatial domain (for the class of filters under consideration of this paper), the number of multiplications could be reduced to K^2. In addition there are costs involved due to DCT and IDCT. We make use of the computationally efficient algorithm developed by Loeffer et al which computes 8×8 DCT (as well as IDCT) with 176 multiplications and 464 additions [11]. Additionally 5.5 multiplications and 14.5 additions per pixel are needed for performing the 8×8 DCT and IDCT. Typically for $K = 8$, numbers of multiplications and additions per pixel are 69.5 and 238.5, respectively.

Computational Cost with BFCD. Costs associated with BFCD are due to computations involved in: (i) Composition of $L \times M$ blocks into a single block, (ii) Element to element multiplications between DCT coefficients of composed block and type-I DCT coefficient of the impulse response, and (iii) Decomposition of the filtered block into $L \times M$ blocks of 8 size.

Following a similar approach for efficient computation of block composition and decomposition of type-II DCT as discussed in [6], numbers of per pixel operations for the BFCD algorithm are presented in Table 2 for different values

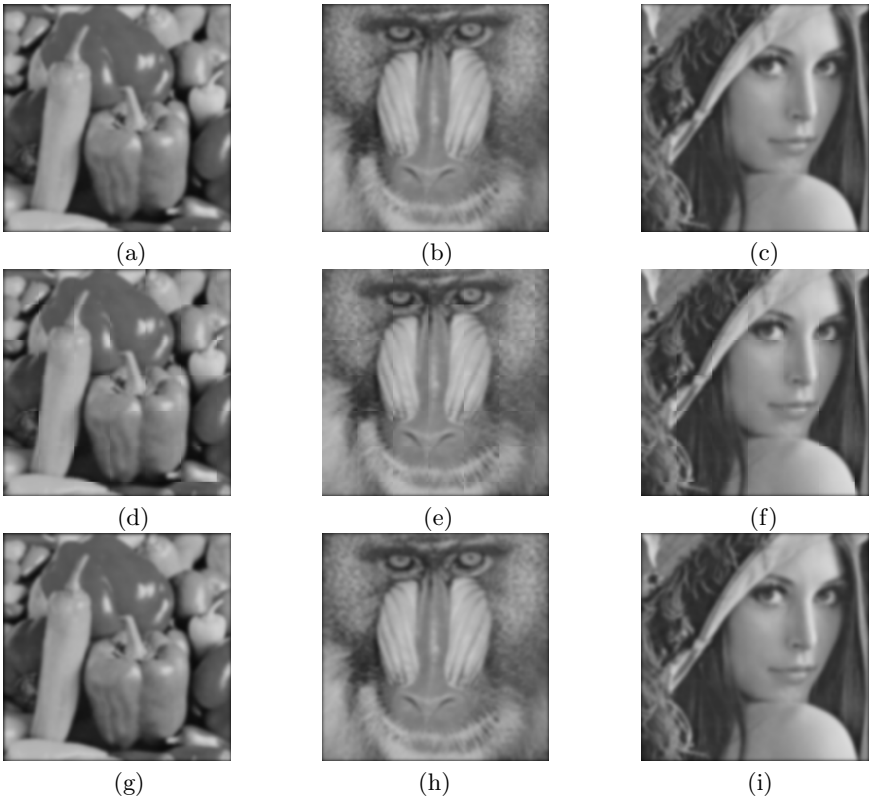


Fig. 1. Gaussian filtering with $\sigma = 2.0$: (a)-(c): Filtering in the spatial domain using linear convolution, (d)-(f) Filtering in the DCT domain using the BFCD algorithm for $L = M = 5$ and (g)-(i): Filtering in the DCT domain using the OBFCD algorithm

of L (or M). Comparing the computational cost of the BFCD algorithm with the corresponding spatial domain operation, it is evident that the BFCD is a faster operation. For example for $L = M = 5$, the number of equivalent addition operations for the BFCD algorithm is 393.45, while the requirement for spatial domain approach (for $K = 8$) is 447. It may be noted however that the BFCD algorithm provides an approximate solution. For $L = M = 5$, the approximate Gaussian filtered image with $\sigma = 2.0$ maintains quite high PSNR values (typically 35.56 dB, 33 dB and 33.14 dB for the images *Pepper*, *Mandrill*, and *Lena*, respectively and refer Figures 1(d)-(f)) with respect to the filtered image obtained through spatial convolution. For obtaining the exact solution, we outline an overlapping block filtering approach in the compressed domain as described in the following subsection.

4.2 Filtering with Overlapping Blocks

One way of removing the boundary effects is to apply BFCD in overlapping set of blocks and retain the results of those blocks which are not affected by

Table 3. Performances of Gaussian Filtering with OBFCD

L(=M)	$\sigma_x(=\sigma_y)$	PSNR (dB)			Per pixel computational cost	Equivalent number of additions per pixel
		Pepper	Mandrill	Lena		
3	2.0	301.42	300.99	301.71	103.88 M + 120 A	431.64
	3.0	301.85	301.31	302.03		
	4.0	302.69	302.10	302.91		
4	2.0	294.23	293.56	294.40	127.11 M + 170.67 A	552
	3.0	294.43	293.75	294.61		
	4.0	295.33	294.61	295.47		
5	2.0	293.18	292.51	293.39	144.92 M + 180 A	614.76
	3.0	293.37	292.66	293.56		
	4.0	294.28	293.56	294.47		

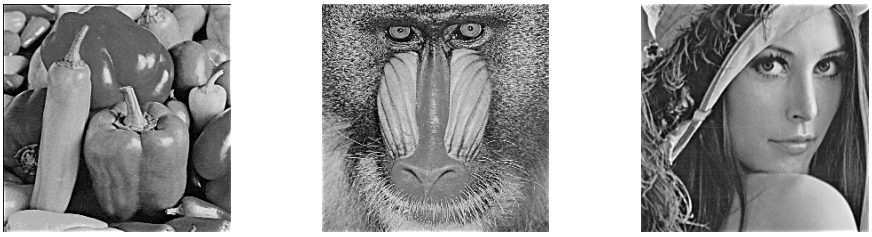
the boundary conditions due to the symmetric extension. For example, if the effective half filter size is $K \times K$ (in spatial domain), BFCD could be applied to $L \times M$ number of blocks producing only the $(L - 2\lceil\frac{K}{8}\rceil) \times (M - 2\lceil\frac{K}{8}\rceil)$ central blocks as the output. Naturally, this will increase the redundancy in the computation and there will be an increase in the number of multiplications and additions as a consequence. We refer this algorithm as *Overlapping_Block_Filtering_on_Composition_Decomposition* (OBFCD). Figures 1(g)-(i) show the results using OBFCD for $L = 3$ and $M = 3$. It can be seen that the filtered images are almost the same as those obtained by the linear convolution (Figures 1(a)-(c)). Table 3 presents the PSNR values obtained by OBFCD. In this case, PSNR values are significantly higher (around 300 dB). This implies that the OBFCD operation is equivalent to the spatial domain convolution. Table 3 also includes the computational cost associated with OBFCD. In this case, it can be seen that as OBFCD requires more computations than BFCD, it is marginally faster than the corresponding spatial domain operation. Typically, for $L = M = 3$, OBFCD requires 431.64 equivalent number of addition operations per pixel of the image, whereas for the spatial domain approach for $K = 8$, the required number is 447. It is also observed from Table 3 that it is not necessary to increase the value of $L(= M)$ beyond 3. Output response does not vary significantly with increasing $L(\geq 3)$. It remains close to the spatially convolved output. However this depends upon the effective half filter size ($K \times K$). We summarize our observations in the following lemma:

Lemma 1: The minimum value of L for *effective half filter size* $K \times K$ is given by $L_{min} = 2\lceil\frac{K}{8}\rceil + 1$.

Proof: As $(L - 2\lceil\frac{K}{8}\rceil) > 0$, $L > 2\lceil\frac{K}{8}\rceil$. Hence, $L_{min} = 2\lceil\frac{K}{8}\rceil + 1$. □

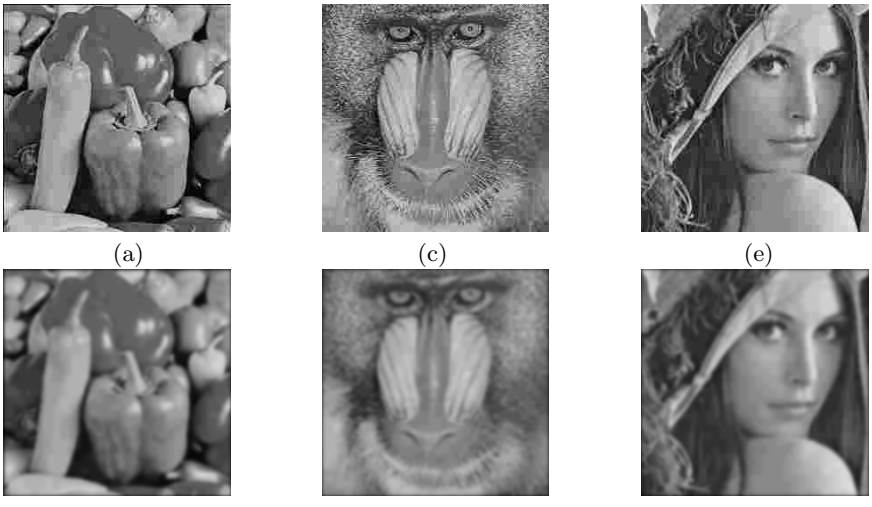
5 Applications of Image Filtering

In this section we demonstrate two specific image processing applications of the proposed image filtering.



(a) (b) (c)

Fig. 2. Image sharpening using OBFCD filtering with $\lambda = 0.7$, $\sigma = 2.0$, $L = M = 3$ and $K = 8$: (a) Peppers, (b) Mandrill, and (c) Lena.



(a) (c) (e)

(b) (d) (f)

Fig. 3. Removal of blocking artifacts of highly compressed JPEG images using OBFCD filtering with $\sigma = 2.0$, $L = M = 3$ and $K = 8$. Images are compressed with JPEG compression scheme with the quality factor 10.0:(a) Peppers: JPEG compressed, (b) Peppers: After OBFCD filtering, (c) Mandrill: JPEG compressed, (d) Mandrill: After OBFCD filtering, (e) Lena: JPEG compressed, and (f) Lena: After OBFCD filtering.

5.1 Image Sharpening

One approach to image sharpening operation is carried out by adding a fraction of the high-pass filtered output to the original image. Let B_f be the low-pass filtered block using BFCD or OBFCD in the transform domain. Let B be its corresponding original block in the transform domain. Hence the sharpened block in the transform domain is computed as follows:

$$B_s = B + \lambda(B - B_f). \tag{11}$$

In Eq. (11) λ (> 0) is the parameter controlling the amount of image sharpening. Figure 2 shows the sharpened images for $\lambda = 0.7$.

5.2 Blocking Artifacts Removal

Blocking artifacts are often visible in images reconstructed from highly compressed data. These blocking artifacts can be masked by applying low pass filtering directly in the compressed domain. We present here examples of such filtering of JPEG compressed images (with *quality factor* =10.0). Blocking artifacts are clearly visible in Figures 3(a), (c), and (e), respectively. Their visibility has been substantially reduced in the filtered images shown in Figures 3(b), (d), and (f), respectively.

6 Concluding Remarks

In this paper we have described filtering in the block DCT space using the convolution-multiplication properties of trigonometric transforms [9]. We have made use of the block composition and decomposition methods of [10] for satisfying the boundary conditions as in the case of linear convolution. We have demonstrated the application of the proposed algorithm in performing two specific image processing operations such as enhancement, and removal of blocking artifacts, in the transform domain.

Acknowledgment

This work was supported in part by a University of California MICRO grant with matching support from Intel Corporation, Qualcomm Corporation and Xerox Corporation.

References

1. B.C. Smith and L. Rowe, "Algorithms for manipulating compressed images," *IEEE Comput. Graph. Applicat. Mag.*, vol. 13, pp. 34–42, September 1993.
2. A. Neri, G. Russo, and P. Talone, "Inter-block filtering and downsampling in DCT domain," *Signal Processing: Image Commun.*, vol. 6, pp. 303–317, August 1994.
3. H.W. Park, Y.S. Park, and S.K. Oh, "L/M-image folding in block DCT domain using symmetric convolution," *IEEE Trans. on Image Processing*, vol. 12, pp. 1016–1034, September 2003.
4. J. Tang and E. Peli, "Image enhancement using a contrast measure in the compressed domain," *IEEE Signal Processing Letters*, vol. 10, pp. 289–292, October 2003.
5. B. Shen, I.K. Sethi, and V. Bhaskaran, "DCT convolution and its application in compressed domain," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8, pp. 947–952, December 1998.
6. J. Mukherjee and S.K. Mitra, "Arbitrary resizing of images in the DCT space," *IEE Proc.: Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 152-164.
7. Y.S. Park and H.W. Park, "Design and analysis of an image resizing filter in the block-DCT domain," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 14, pp. 274–279, February 2004.

8. G.S. Shin and M.G. Kang, "Transform domain enhanced resizing for a discrete-cosine-transform-based codec," *Optical Engineerring.*, vol. 42, pp. 3204–3214, November 2003.
9. S.A. Martucci, "Symmetric convolution and the discrete sine and cosine transforms," *IEEE Trans. on Signal Processing*, vol. 42, pp. 1038–1051, May 1994.
10. J. Jiang and G. Feng, "The spatial relationships of DCT coefficients between a block and its sub-blocks," *IEEE Trans. on Signal Processing*, vol. 50, pp. 1160–1169, May 2002.
11. C. Loeffer, A. Ligtenberg, and G.S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," *Proc. IEEE Int. Conf. Accoustics, Speech and Signal Processing*, vol. 2, pp. 988–991, May 1989.
12. R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, pp. 461–474, April 2001.
13. B. Shen, I.K. Sethi, and V. Bhaskaran, "Adaptive motion vector resampling for compressed video downscaling, " *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, pp. 929-936, September, 1999.