A Complexity Scalable Universal DCT Domain Image Resizing Algorithm

Carlos Salazar and Trac D. Tran, *Member, IEEE*

***Abstract*—This paper presents a computationally flexible method for producing a mapping from one discrete cosine transform (DCT) domain to another that results in a decoded image that has been arbitrarily resized in the spatial dimension. A notable feature of the proposed mapping is its computational scalability; final image quality can be traded off for lower implementation complexity and a wide range of complexity versus final quality operation points can be realized for each scale factor. Current existing methods often suffer from a lack of flexibility (i.e., work for only one or at most a few resizing factors, have only one or two levels of complexity) or require more operations to achieve similar levels of final image quality. When constructing the mapping, a multiplierless DCT approximation can also be employed, yielding fast implementation with excellent results. The use of the DCT approximation confers several benefits upon the proposed mapping including multiplierless implementation or at most integer operations rather than floating point operations.**

***Index Terms*—Computational flexibility, DCT domain, integer implementation, multiplierless, resizing, transcoding.**

I. Introduction

ARBITRARY image resizing is an important problem. Images and video streams often need to be resized spatially in order to ensure that they are tailored to the communications networks over which they travel and to the end-user display devices upon which they will be presented [1]–[4]. Since so much multimedia material is compressed using the popular block discrete cosine transform (DCT) framework, e.g., JPEG images, MPEG videos, and H.26X video conferencing streams, a spatial resizing operation that is efficient, computationally flexible, and occurs in the block DCT domain is desirable. Computational flexibility is especially important because a wide range of design options is often needed in practice in order to meet specific power consumption and performance design requirements.

A. Prior DCT Domain Resizing Efforts

A number of DCT domain spatial resizing methods have been proposed [5]–[15]. Many of them are limited to power-of-two scaling factors [5], [12]–[15] or their performance has been surpassed by newer methods [10], [11]. Dugad [5] put forth a particularly efficient method for upsampling and downsampling an image by powers of two that is carried out entirely in the DCT domain. Their technique used simple DCT scaling [16]–[18] and took advantage of clever factorization to dramatically reduce the computations required. Mukherjee [12] used subband DCT processing [19] to improve upon the final image quality of [5] while requiring a bit more computation. Patil [20] developed a method that operates on 16 by 16 macroblocks for arbitrary downsizing.

Park [6] used symmetric convolution [16] to achieve arbitrary resizing factors and produced higher final image quality than the method of Dugad [5] but with a higher level of computational complexity. By approximating their initial mapping they were able to greatly reduce its computational complexity while retaining most of the final image quality improvements of the original. Wang [7] exploited the spatial relationship between a DCT domain block and its subblocks to realize their resizing method. In order to achieve a given scale factor of $S = O/I$, both techniques in [6] and [7] require a magnification step of O followed by a reduction step of I which as implemented is computationally inefficient (e.g., the scale factor 2/3 requires a magnification of 2 followed by a reduction of 3).

Zhao [8] presented an algorithm for scaling images by factors of 1.25 and 1.5 directly in the DCT domain. This method implicitly uses interpolation and scaling matrices and is intended for applications such as face recognition in images. While the authors indicated that multiple scale factors can be designed, they did not address any way of varying either the output quality or the computational load. Their method also required quite a few more additions than multiplications for a given scale factor. Mukhopadhyay [13] proposed a method similar to ours [21] in some respects but lacking the features of computational scalability and with no discussion of taking advantage of the symmetry of the mappings to reduce the computational burden. Finally, the method of Mehta [9] is quite computationally expensive compared to other techniques.

B. A Computationally Scalable DCT Resizing Algorithm

In this paper, we propose a method to arbitrarily rescale an image based upon a generalization of the technique proposed in [5] which uses simple DCT scaling as described in [16]–[18]. We endow our algorithm with computational scalability by changing the size of the inverse and forward transforms used and by varying the number of input and output coefficients used and produced by those transforms.

In our proposed approach illustrated in Fig. 1, a single mapping is constructed that implicitly involves a combined resizing and inverse transform back into the spatial domain followed by a combined resizing and forward transform into the 8×8 DCT domain. Because our proposed method resizes at both the inverse and forward transformations, we can get any $S = O/I$ scale factor (rather than just powers of two). By choosing different N-point inverse and M-point forward DCTs more or less of the original image data can be used to vary the final image quality for a given $S = O/I$ scale factor. On the other hand,

Manuscript received August 9, 2005; revised April 9, 2006 and September 5, 2006. This work has been supported in part by the National Science Foundation under Grants CCF-0093262 and CCF-0430869. This paper was presented in part at the IEEE International Conference in Image Processing, Singapore, October 2004 and the IEEE International Symposium on Circuits and Systems, Kobe, Japan, May 2005. This paper was recommended by Associate Editor H. Chen.

The authors are with the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: cls@jhu.edu; trac@jhu.edu).

Digital Object Identifier 10.1109/TCSVT.2007.893836

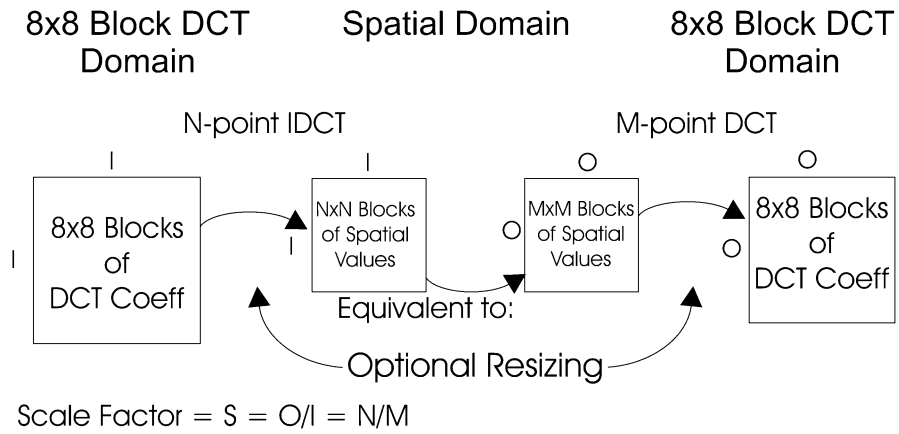

Fig. 1. Arbitrary scaling by resizing at either or both transform locations. Note that the dimensions of each image are given in blocks (i.e., the size of the first image is I blocks by I blocks where each blocks is of size 8×8, the size of the second image is still I blocks by I blocks but the block size is now $N \times N$, etc.).

N and M can be fixed while reducing the number of input and output coefficients used and produced by the mapping to dramatically decrease the number of required computations.

At this point, our resizing algorithm's complexity is already lower than that of the methods of [6], [7], and [9]. By factoring the mapping into an upsizing stage and a downsizing stage we can take advantage of the inherent symmetry of these factors. Recognition of additional symmetry allows us to reduce the computational complexity beyond even that of the very efficient technique of Dugad. It can be shown [21] that we can reduce the number of multiplications required for downsizing by 20% over that of [5]. Additionally, a variety of multiplierless DCT approximations including the various binDCT factorizations and approximations [22]–[24] can be used to obtain a multiplierless implementation or use at most integer instead of floating point operations.

C. Notation

The following notation and convention are used throughout the paper. Bold uppercase and lowercase symbols represent matrices and vectors, respectively. $\mathbf{X}$ will be used to represent an 8×8 block of spatial pixels while $\mathbf{Y}$ will be used to represent an 8×8 block of DCT coefficients. $\mathbf{C}_M$ is the M-point DCT matrix and $\mathbf{C}_N^t$ is the N-point IDCT matrix. $\mathbf{I}_n$ is the $n \times n$ identity matrix whereas $\mathbf{O}_{n \times m}$ is the $n \times m$ null matrix.

If a symbol is not in boldface then it corresponds to a scalar quantity. The scale factor is denoted throughout as $S = O/I$ where O is the number of output blocks and I is the number of input blocks. C_I and C_O represent the number of input and output coefficients, respectively.

II. Problem Formulation

Fig. 2(a) illustrates the typical processing flow for transcoding an image or video frame from one spatial resolution to a different spatial resolution where the image or video frame has been encoded using a block based DCT method such as JPEG [4] or MPEG-2 [1]. The steps of the transcoding operation are: 1) variable length decoding; 2) inverse quantization; 3) inverse DCT; 4) spatial domain resizing operation; 5) forward DCT; 6) quantization; and 7) variable length encoding. Our goal is to perform the resizing operation directly in the DCT domain in order to decrease the computational effort required. The revised processing flow is as shown in Fig. 2(b) with the following steps: 1) variable length decoding; 2) inverse quantization using a modified quantization matrix; 3) DCT domain resizing; 4) quantization using a modified quantization matrix; and 5) variable length encoding.

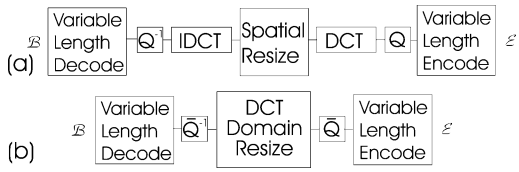

Fig. 2. (a) Standard process flow for resizing compressed images. (b) Alternate process flow for resizing compressed images. Note that $\bar{\mathbf{Q}}^{-1}$ and $\bar{\mathbf{Q}}$ can be modified from $\mathbf{Q}^{-1}$ and $\mathbf{Q}$ to assist in implementing the resize operation.

III. Resizing by an Arbitrary Scale Factor Via Simple DCT Scaling

The DCT itself can be used to resize an image as described in [16]–[18] and is referred to in [16] as simple DCT scaling. Simple DCT scaling is used in [5] to downsize or upsize an image by a factor of two. Examination of the method of [5] reveals that although the final mapping operates entirely in the DCT domain it implicitly involves a transformation into the spatial domain while simultaneously resizing the image followed by a standard transformation back into the 8×8 block DCT domain.

In [17] and [18], it was shown that lowpass filtering and decimation could be combined into one operation in the DCT domain, i.e., spatial resizing. Let $\mathbf{y}_L$ be the L-point DCT of $\mathbf{x}_L$, a spatial domain vector of length L and let $\mathbf{C}_N$ be the N-point DCT transform matrix. Then from [18] the lowpass filtered and decimated spatial domain vector of length N (where $N < L$) is

$$\mathbf{x}_N = \sqrt{\frac{N}{L}} \cdot \mathbf{C}_N^t [\mathbf{I}_N \; \mathbf{0}_{N \times (L-N)}] \mathbf{y}_L. \tag{1}$$

If $N > L$ then we must zero pad $\mathbf{y}_L$ before applying the transform $\mathbf{C}_N$ and this operation corresponds to upsampling and lowpass filtering

$$\mathbf{x}_N = \sqrt{\frac{N}{L}} \cdot \mathbf{C}_N^t \begin{bmatrix} \mathbf{y}_L \\ \mathbf{0}_{(N-L)\times 1} \end{bmatrix}. \tag{2}$$

We go beyond [5] and propose to generalize the resizing to arbitrary scale factors by performing the resizing at *both* transformation points. Our mapping which scales an image by a factor $S = O/I$ is constructed by merging *two* combined transform and resizing operations.

- First, by applying an N-point IDCT to the 8-point DCT coefficients of the original image we can resize the image by a factor of $N/8$ while simultaneously taking the image into the spatial domain.
- Then applying an M-point DCT and retaining only the 8 lower-frequency transform coefficients; thus resizing the image by a factor of $8/M$ and simultaneously taking the image back into the 8×8 DCT domain as shown in Fig. 1.

This way, we have thus scaled the original image by $(N/8) * (8/M)$. To implement a scale factor of $S = O/I$ simply select N and M such that $N/M = O/I$. Since many different choices of M and N can correspond to a given scale factor $S = O/I$, a designer is free to choose appropriate values of M and N that satisfy the final image quality requirement while not exceeding the implementation complexity budget. Further flexibility can be introduced by using fewer coefficients C_I from the input image and producing fewer coefficients C_O in the output image (where $C_I \leq N$ and $C_O \leq M$). The mapping thus produced can often be factored to reduce implementation complexity even further but the details are beyond the scope of this paper.

Since the DCT is a separable transform the DCT resizing operation can be applied both horizontally and vertically to the coefficients of an image and different scale factors can be used for each axis (i.e., M_x, M_y, N_x, N_y) although this will result in distortion of the image since the aspect ratio is being altered if square pixels are used to display the image. For simplicity, the following explanation assumes that $M_x = M_y$ and that $N_x = N_y$. Fig. 3 shows the four different cases when M and N or alternatively C_O and C_I are not equal to 8.

When S is not equal to 1/2, the problem is slightly more complicated. The basic constraint is the need to produce the coefficients of the final resized image in the 8-point DCT domain. For a scale factor of $S = O/I$ applied both horizontally and vertically this corresponds to replacing I^2 8×8 blocks of the original image with O^2 8×8 blocks in the resized image. As seen in Fig. 1, I^2 blocks of the original image can be grouped together and the mapping applied to this group resulting in a final group of O^2 blocks. More generally, as illustrated in Fig. 4, $I_y \times I_x$ blocks of the original image can be grouped together and processed to produce the final group of $O_y \times O_x$ blocks.

Let us define two matrices $\mathbf{D}_{N\times 8}$ and $\mathbf{F}_{8\times M}$ as follows:

$$\mathbf{D} = \begin{bmatrix} \mathbf{I}_{C_I} & \mathbf{0}_{C_I\times(8-C_I)} \\ \mathbf{0}_{(N-C_I)\times C_I} & \mathbf{0}_{(N-C_I)\times(8-C_I)} \end{bmatrix}$$
$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_{C_O} & \mathbf{0}_{C_O\times(M-C_O)} \\ \mathbf{0}_{(8-C_O)\times C_O} & \mathbf{0}_{(8-C_O)\times(M-C_O)} \end{bmatrix}.$$

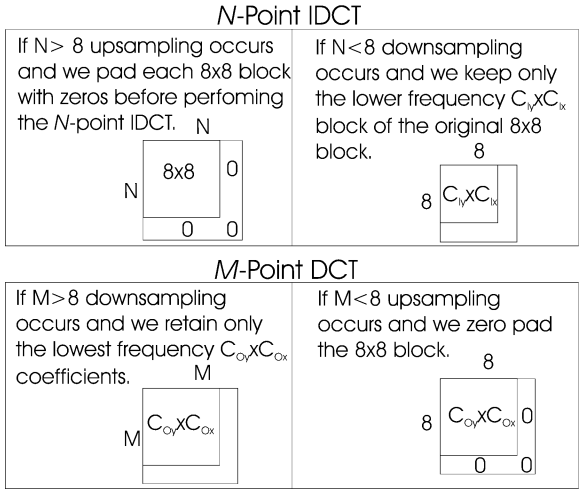

Fig. 3. Four possible conditions when N and M are not equal to 8.

We now define four block diagonal matrices. $\mathbf{\Lambda_O} = \mathbf{diag(F)}$, $\mathbf{\Lambda_I} = \mathbf{diag(D)}$, $\mathbf{\Lambda_{OM}} = \mathbf{diag(C_M)}$ and $\mathbf{\Lambda_{IN}} = \mathbf{diag(C_N^t)}$. $\mathbf{\Lambda_O}$ and $\mathbf{\Lambda_{OM}}$ have O blocks along their diagonals while $\mathbf{\Lambda_I}$ and $\mathbf{\Lambda_{IN}}$ have I blocks along their diagonals.

Mathematically our proposed mapping can now be expressed in matrix form

$$\begin{bmatrix} \mathbf{Y'}_{11} & \cdots & \mathbf{Y'}_{1O_x} \\ \vdots & \ddots & \vdots \\ \mathbf{Y'}_{O_y1} & \cdots & \mathbf{Y'}_{O_yO_x} \end{bmatrix} = \sqrt{\frac{N_xN_y}{M_xM_y}} \cdot \mathbf{\Lambda}_{O_y} \cdot \mathbf{\Lambda}_{OM_y} \cdot \mathbf{\Lambda}_{IN_y} \cdot \mathbf{\Lambda}_{I_y} \cdot \begin{bmatrix} \mathbf{Y}_{11} & \cdots & \mathbf{Y}_{1I_x} \\ \vdots & \ddots & \vdots \\ \mathbf{Y}_{I_y1} & \cdots & \mathbf{Y}_{I_yI_x} \end{bmatrix} \cdot \mathbf{\Lambda}_{I_x}^t \cdot \mathbf{\Lambda}_{IN_x}^t \cdot \mathbf{\Lambda}_{OM_x}^t \cdot \mathbf{\Lambda}_{O_x}^t. \tag{3}$$

In this equation, $\mathbf{Y'}_{ij}$ and $\mathbf{Y}_{kl}$ are the 8×8 DCT blocks of the $O_y \times O_x$ output and $I_y \times I_x$ input arrays, respectively. The input and output conditioning matrices are, respectively, $\mathbf{\Lambda_O}$ and $\mathbf{\Lambda_I}$.

IV. Image Resizing Experiments

We now examine the performance of various resizing operations proposed in previous sections. For comparison, Table I provides the results from various other methods cited previously. In addition to the cases presented here, we have included a few results for a variety of scale factors and choices of N, M, C_I, and C_O in Table II. In all cases, peak sungal-to-noise ratio (PSNR) is computed by reversing the resizing operation and comparing the resulting image with the original.

Fig. 5 shows the PSNR of the downsampled-by-two Lena images versus the number of multiplications using various algorithms. For clarity, we have refrained from showing the number of additions. Finally, Fig. 6 demonstrates subjective image quality by comparing our method using two complexity/final quality operating points against the spatial method and that of [5]. While Fig. 6(a) does exhibit blocking artifacts in the

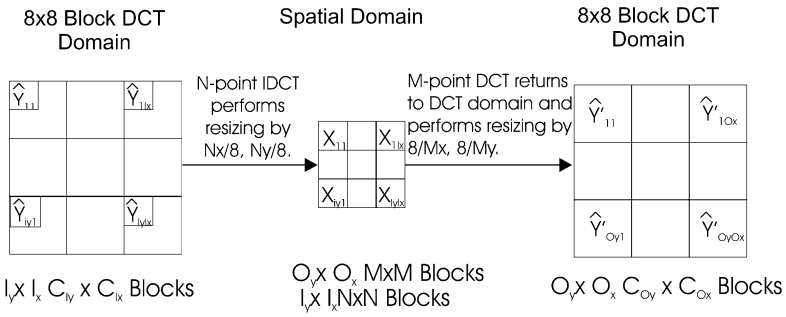

Fig. 4. Resizing by $S_x = O_x/I_x$ in the horizontal dimension and by $S_y = O_y/I_y$ in the vertical dimension in the DCT domain using our proposed method. Each $\hat{\mathbf{Y}}$ block consists of $C_{I_y} \times C_{I_x}$ coefficients while each $\hat{\mathbf{Y}}'$ contains $C_{O_y} \times C_{O_x}$ coefficients.

TABLE I
PSNR VERSUS COMPLEXITY FOR VARIOUS PUBLISHED ALGORITHMS

Algorithm	Scale Factor	PSNR (in dB)		# of Mults	# of Adds
		Lena	Boat		
Spatial[5]	1/2	30.33	28.24	3.44	9.81
Dugad[5]	1/2	34.69	31.5	1.25	1.25
Mukherjee[12]	1/2	34.89	31.55	1.31	1.25
Park[6]	1/2	35.36	31.96	3.38	3.75
Park[6]	1/2	35.08	31.83	1.89	2.06
Hu(FC)[15]	1/2	28.2	-	1.44	2.19
Hu(FM)[15]	1/2	29.24	-	0.25	0
Spatial[6]	1/3	26.72	24.99	3.06	8.94
Park[6]	1/3	31.53	28.74	2.39	3.67
Wang[7]	1/3	31.41	28.61	0.9	1.11
Zhao[8]	2/3	35.56	-	2.46	9.14
Wang[7]	3/4	31.72	28.96	7.91	11.92
Zhao[8]	4/5	36.65	-	2.31	7.68
Wang[7]	3/2	39.82	36.99	3.39	5.19

TABLE II
OUR METHOD'S PSNR VERSUS COMPLEXITY FOR VARIOUS N, M, C_I, C_O

Scale Factor	N	M	C_I	C_O	# of Mult	# of Add	PSNR (in dB)	
							Lena	Boat
1/2	2	4	2	4	0.13	0.19	28.75	26.25
1/2	3	6	3	5	0.26	0.43	30.72	28.01
1/2	4	8	4	8	1	1.25	35.02	31.62
1/2	5	10	5	8	1.41	1.76	35.34	31.94
1/2	6	12	6	8	1.88	2.34	35.36	31.96
1/3	2	6	2	6	0.25	0.29	28.88	26.31
1/3	3	9	3	7	0.5	0.64	30.39	27.66
1/3	3	9	3	8	0.65	0.77	31.41	28.61
1/3	4	12	4	8	1.04	1.25	31.53	28.73
2/3	4	6	4	5	1.15	1.60	33.09	30.12
2/3	4	6	4	6	1.50	1.92	34.61	31.39
2/3	6	9	4	8	3.01	3.55	34.9	31.66
2/3	6	9	5	7	2.72	3.68	36.58	33.04
2/3	6	9	6	8	4.19	5.19	38.65	35.39
3/4	3	4	3	4	0.56	0.94	31.71	28.92
3/4	6	8	4	7	3.04	4.44	35.06	31.69
3/4	6	8	6	7	4.75	5.76	38.48	35.03
3/4	6	8	6	8	5.63	6.56	40.02	37.38
4/5	4	5	3	3	0.51	1.11	29.64	27.22
4/5	4	5	4	4	1.08	1.80	32.52	29.74
4/5	8	10	5	6	4.29	5.88	36.46	33.03
4/5	8	10	6	8	7.75	9.46	40.05	37.31
3/2	3	2	3	2	0.13	0.42	31.67	28.91
3/2	6	4	6	4	1.50	1.92	40.51	37.62
3/2	9	6	7	5	2.72	3.63	43.13	42.4
3/2	9	6	7	6	3.50	4.33	44.26	44.09

zoomed-in image it should be noted that these are difficult to see in the downsized image and the computational effort

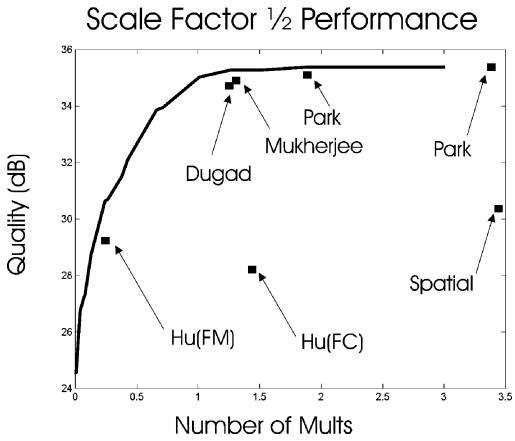

Fig. 5. Final image quality of Lena versus number of multiplications required compared to various other algorithms.

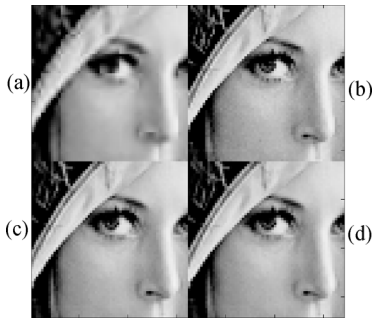

Fig. 6. Comparison of various methods for $S = 1/2$. (a) Our method with $N = 2$, $C_I = 2$, $C_O = 4$, PSNR $= 28.75$ dB, Multiplications Per Pixel (MPP) $= 0.13$. (b) Method of [5], PSNR $= 34.69$ dB, MPP 1.25. (c) Our method with $N = 5$, $C_I = 5$, $C_O = 8$, PSNR $= 35.34$ dB, MPP $= 1.41$. (d) Spatial domain PSNR $= 30.33$ dB, MPP $= 3.44$.

required to produce this image is approximately a tenth that of the higher quality image Fig. 6(b) [5].

Fig. 7 shows the typical behavior of the PSNR as we vary the input parameters and similar results were obtained for a variety of images and scale factors. The optimal values of N_x, N_y, M_x, M_y, C_{I_x}, C_{I_y}, C_{O_x}, and C_{O_y} can be determined for a given scale factor in the following fashion. First, multiply the scale factor by 8 (the maximum number of input coefficients), round down to the nearest integer, and label that number z. Then, choose the smallest integer multiple of the numerator in the scale factor S that is greater than or equal to $z + 1$ and set that

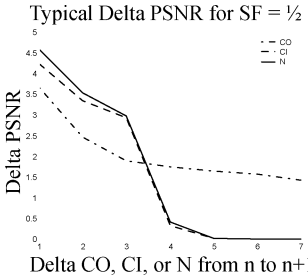

Fig. 7. Typical change in PSNR versus a change in N, C_I, and C_O for a scale factor of 1/2.

number equal to N, e.g. for $S = 2/3$, $z = 5$, $z + 1 = 6$, and $N = 3{\times}2 = 6$. C_I can be set equal to either z or $z{+}1$ depending on whether the maximum PSNR is desired and C_O should be set equal to the minimum of 8 and M. This will ensure that extra computation is not performed for minimal to no extra performance gain. The choice of N puts an upper limit on the PSNR. If N is set too low varying the other parameters will not increase the PSNR. However, N can be set at the previously mentioned level and the computational complexity can be lowered dramatically by reducing C_I and/or C_O. Observation of Fig. 7 reveals several things. It should be noted that C_O has a large effect on PSNR every time it's changed and should be reduced only when necessary to get the computational complexity below a desired threshold, while C_I can be lowered initially with little impact on final image quality.

V. Conclusion

A flexible method for resizing images by an arbitrary scale factor in the DCT domain has been presented. This method can be used to produce mappings with lower complexity and/or better final image quality than other current state of the art methods. Our generalization produces a mapping from the 8×8 DCT domain to another with arbitrary scale factors. The mapping is based upon the synthesis of a combined transform and resizing at either or both transform stages using a variable number of coefficients from the original image and producing a variable number of coefficients in the output image. Our method can also be implemented in a multiplierless fashion [21] with advantages including lower complexity with nearly identical performance and an integer arithmetic implementation. All of the methods produced better final image quality with lower complexity than a direct implementation (i.e., transformation into the spatial domain, spatial domain processing, and transformation back into the DCT domain). While our method allows a wide range of implementation complexity/final image quality choices to be made, we also provided guidelines for choosing the optimal parameters to maximize PSNR with the least computational effort. We also indicated how to reduce the complexity in a way that would have the smallest effect on the final image quality. The extension of this work to the problem of resizing arbitrary region-of-interest and inter-frame video is currently under investigation [24].

References

[1] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*. New York: Champman & Hall, 1997.

[2] R. Xiong, J. Xu, and F. Wu, "A lifting based wavelet transform supporting non-dyadic spatial scalability," in *Proc. 2006 IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 1861–1864.

[3] A. Segall and A. Katsaggelos, "Resampling for spatial scalability," in *Proc. 2006 IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 181–184.

[4] W. B. Pannebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand, 1993.

[5] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 461–474, Apr. 2001.

[6] H. Park, Y. Park, and S. Oh, "L/M-fold image resizing in block-DCT domain using symmetric convolution," *IEEE Trans. Image Process.*, vol. 12, no. 9, pp. 1016–1034, Sep. 2003.

[7] C. Wang, H. Yu, and M. Zheng, "A fast scheme for arbitrarily resizing of digital image in the compressed domain," *IEEE Trans. Consum. Electron.*, vol. 49, no. 2, pp. 466–471, May 2003.

[8] Y. Zhao, M. Kankanhalli, and T.-S. Chua, "Fractional scaling of image and video in DCT domain," in *Proc. 2003 IEEE Int. Conf. Image Process.*, Sep. 2003, vol. 1, pp. 185–188.

[9] D. Mehta and U. B. Desai, "A primer to video transcoding: Image transcoding," presented at the 8th Nat. Conf. Commun. (NCC2002), Jan. 2002.

[10] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image down-sampling and for inverse motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, pp. 468–476, Jun. 1997.

[11] S. F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE J. Sel. Areas Commun.*, vol. 13, pp. 1–11, Jan. 1995.

[12] J. Mukherjee and S. K. Mitra, "Image resizing in the compressed domain using subband DCT," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 620–627, Jul. 2002.

[13] J. Mukhopadhyay and S. K. Mitra, "Resizing of images in the DCT space by arbitrary factors," in *Proc. 2004 IEEE Int. Conf. Image Process.*, Mar. 2004, pp. 2801–2804.

[14] B. K. Natarajan and B. Vasudev, "A fast approximate algorithm for scaling down digital images in the DCT domain," in *Proc. 1995 IEEE Int. Conf. Image Process.*, Oct. 1995, vol. 2, pp. 241–243.

[15] Q. Hu and S. Panchanathan, "Image/video spatial scalability in compressed domain," *IEEE Trans. Ind. Electron.*, vol. 45, no. 2, pp. 23–31, Feb. 1998.

[16] S. A. Martucci, "Image resizing in the discrete cosine transform domain," in *Proc. 1995 IEEE Int. Conf. Image Process.*, Washington, D.C., 1995, pp. 244–247.

[17] K. N. Ngan, "Experiments on two-dimensional decimation in time and orthogonal transform domains," *Signal Process.*, pp. 249–263, Oct. 1986.

[18] J. M. Adant *et al.*, "Block operations in digital signal processing with application to TV coding," *Signal Process.*, pp. 385–397, Dec. 1987.

[19] S.-H. Jung, S. K. Mitra, and D. Mukherjee, "Subband DCT: Definition, analysis and applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 273–286, Jun. 1996.

[20] V. Patil, R. Kumar, J. Mukherjee, and S. S. Prasad, "A fast arbitrary downsizing algorithm for video transcoding," in *Proc. 2006 IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 857–860.

[21] C. Salazar and T. D. Tran, "On resizing images in the DCT domain," in *Proc. 2004 IEEE Int. Conf. Image Process.*, Oct. 2004, vol. 4, pp. 2797–2800.

[22] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," *IEEE Trans. Signal Process.*, vol. 49, no. 12, pp. 3032–3044, Dec. 2001.

[23] T. D. Tran, "The binDCT :Fast multiplierless approximation of the DCT," *IEEE Signal Process. Lett.*, vol. 7, pp. 141–144, Jun. 2000.

[24] C. Salazar and T. D. Tran, "Flexible resizing algorithms for video transcoding," in *Proc. 2005 IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 916–919.