ON RESIZING IMAGES IN THE DCT DOMAIN

Carlos L. Salazar and Trac D. Tran

The Johns Hopkins University
ECE Department
Baltimore, MD 21218
cls/trac@jhu.edu

ABSTRACT

This paper presents a general method for producing a mapping from the DCT domain to another DCT domain that results in an image that has been resized in the spatial dimension. Although the mapping is implemented entirely in the DCT domain, it can be thought of as a transformation into the spatial domain using a combined inverse DCT and resizing operator followed by a combined forward DCT and another resizing operator. Final image quality can be traded off for lower implementation complexity and a wide range of complexity versus final quality operation points can be chosen 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 be used for 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 rather than floating point operations.

1. INTRODUCTION

Arbitrary image resizing is an important problem. It is needed to ensure that images and video streams are tailored to the communications networks over which the streams travel and to the end-user display devices upon which they will be presented. Since so much multimedia material (especially images and video) is compressed using the popular block DCT framework (e.g. JPEG images, MPEG videos, and H.26X video conferencing streams) a resizing operation that is efficient (i.e. offers less complexity than a direct inverse transformation, spatial domain resizing, and forward transformation) and occurs in the block DCT domain is desirable.

Dugad and Ahuja [1]proposed a method for resizing an image by powers of two (both upsampling and downsampling) that is carried out entirely in the DCT domain. Their technique used simple DCT scaling and took advantage of clever factorization to dramatically reduce the computations required. Park, Park, and Oh [2]used symmetric convolution to implement arbitrary resizing factors and produced higher final image quality than the method of Dugad and Ahuja but at a greater computational complexity. By approximating the derived mapping they were able to greatly reduce the computational complexity while maintaining quite a bit of the final image quality. Zhao, Kankanhalli, and Chua [3]presented an algorithm for scaling images by factors of 1.25 and 1.5 directly in the DCT domain. Their method implicitly uses interpolation and scaling matrices.

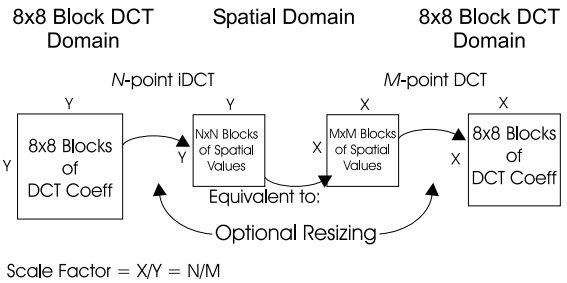

Fig. 1. Arbitrary scaling by resizing at either or both transform locations. Note that the dimensions of each image are given in blocks (i.e. the size of the first image is Y blocks by Y blocks where the blocks are of size 8x8, the size of the second image is still Y blocks by Y blocks but the block size is now NxN, etc.)

Because the method of Dugad and Ahuja is constrained to factors that are powers of two, it is unsuitable for applications where a finer granularity is required. The general method proposed by Park, Park, and Oh requires more computational effort than that of Dugad and Ahuja for the same factors. In addition, in order to achieve a given scale factor of X/Y, they require a magnification step of X followed by a reduction step of Y which is computationally inefficient. The method of Zhao, Kankanhalli, and Chua works only for factors of 1.25 and 1.5 (although other scale factors can be designed) and is intended for applications such as face recognition in images.

Our proposed method used to arbitrarily rescale an image is based upon a generalization of the technique proposed in [1]. In our approach, 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 by 8 DCT domain (see Fig. 1). By combining these operations into a single mapping, complexity can be reduced over that of the method of Park, Park, and Oh. Recognition of additional symmetry allows us to reduce the computational complexity beyond even that of the technique of Dugad and Ahuja (for a scale factor of 1/2 we are able to reduce the number of multiplications by 20%). Because we resize at both the inverse and forward transformations we can get any X/Y scale factor (rather than just powers of two as in the method of Dugad and Ahuja). By choosing different N-point inverse and M-point forward DCTs we can use more or less of the original image data to vary the final image quality

0-7803-8554-3/04/$20.00 ©2004 IEEE.

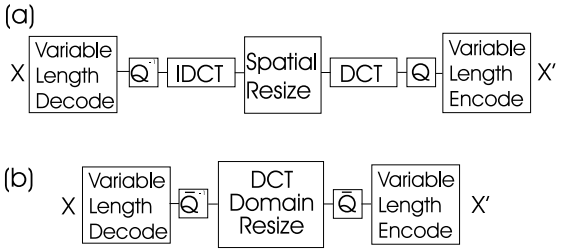

Fig. 2. (a) Standard process flow for resizing compressed images; (b) Alternate process flow for resizing compressed images. Note that $\bar{\mathbf{Q}}^{-1}$ and $\bar{\mathbf{Q}}$ can be modified from $\mathbf{Q}^{-1}$ and $\mathbf{Q}$ to assist in implementing the resize operation.

for a given X/Y scale factor. Finally, we can use a variety of multiplierless DCT approximations (such as the various binDCT factorizations and approximations [4]) to obtain a multiplierless implementation or use at most integer operations vice floating point operations. Thus our method allows for general resizing of an image by a scale factor of X/Y with a selectable tradeoff between final image quality and implementation complexity while offering an integer (possibly multiplierless) implementation.

2. USING SIMPLE DCT SCALING TO RESIZE AN IMAGE BY AN ARBITRARY SCALE FACTOR

The DCT itself can be used to resize an image as described in [5, 6, 7] and is referred to in [5] as simple DCT scaling. Simple DCT scaling is used in [1] to downsize an image by a factor of two, see Fig.3. By examining the method of [1] we recognize 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 8x8 block DCT domain. We go beyond [1] and generalize the resizing to arbitrary scale factors by performing the resizing at *both* transformation points. We propose to construct a mapping that scales an image by a factor X/Y. This mapping is built by merging *two* combined transform and resizing operations. 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. We then apply an M-point DCT and retain only the lower 8 coefficients thus resizing the image by a factor of $8/M$ and simultaneously taking the image back into the 8x8 DCT domain (see Fig.1). We have thus scaled the original image by $(N/8)*(8/M)$. To implement a scale factor of X/Y we simply choose N and M such that N/M = X/Y. Since many different choices of M and N can correspond to a given scale factor X/Y, a designer is free to choose appropriate values of M and N that satisfy the final image quality requirements while not exceeding the implementation complexity budget. The final mapping thus produced can be further factored to reduce implementation complexity.

In [6, 7] it was shown that low pass filtering and decimation could be combined into one operation in the DCT domain i.e. spatial resizing. Let $\mathbf{X_L}$ be the L-point DCT of x_L, a spatial domain vector of length L and let $\mathbf{G_N}$ be the N-point DCT transform matrix. Then from [7] the lowpass filtered and decimated spatial domain vector of length N (where $N < L$) is:

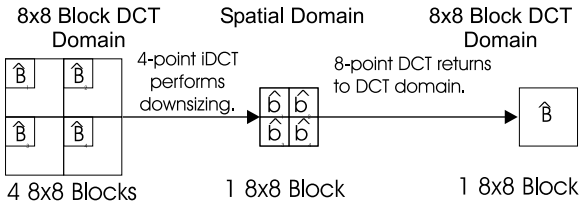

Fig. 3. Downsampling by two in both dimensions in the DCT domain using the method of [1]

$$x_N = \sqrt{N/L} \cdot \mathbf{G_N^t} \mathbf{X_L}(0 \cdots N-1). \tag{1}$$

If $N > L$ then we must zero pad $\mathbf{X_L}$ before applying the transform $\mathbf{G_N}$ and this operation corresponds to upsampling and lowpass filtering, thus:

$$x_N = \sqrt{N/L} \cdot \mathbf{G_N^t}[\mathbf{X_L}, 0 \cdots 0], \tag{2}$$

where $[\mathbf{X_L}, 0 \cdots 0]$ is a column vector of length N.

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. Fig. 4 shows the four different cases when M and N are not equal to 8. When N is less than 8 we keep only the lower $N \times N$ DCT coefficients and when N is greater than 8 we must zero pad the 8x8 DCT coefficient block before performing the N-point iDCT. Similarly when M is less than 8 we assign the MxM coefficients produced in the M-point DCT to the lower frequency coefficients of the final 8x8 block and zero pad the rest. When M is greater than 8 we retain only the lower frequency 8x8 coefficients of the $M \times M$ coefficients produced in the M-point DCT (we construct the mapping such that only the 8 lowest frequency coefficients are computed).

The existing method of [1] works well for a scale factor of two because the 8-point DCT block structure is well suited to this operation geometrically. We can simply fit four downsized blocks in the area previously occupied by a single original resolution block (see Fig. 3). For a scale factor of X/Y (applied both horizontally and vertically) this corresponds to replacing Y^2 8x8 blocks of the original image with X^2 8x8 blocks in the resized image. The mapping is applied to Y^2 blocks of the original image resulting in a final group of X^2 blocks (see Fig. 1).

The mapping can be expressed in matrix form as follows:

$$\mathbf{v_O} = \sqrt{N/M} \cdot \mathbf{\Lambda_O} \cdot \mathbf{\Lambda_{XM}} \cdot \mathbf{\Lambda_{YN}} \cdot \mathbf{\Lambda_I} \cdot \mathbf{v_I}. \tag{3}$$

Where $\mathbf{\Lambda_{XM}}$ and $\mathbf{\Lambda_{YN}}$ are block diagonal matrices with the M-point forward and N-point inverse DCT matrices, respectively, along their diagonals. $\mathbf{\Lambda_O}$ and $\mathbf{\Lambda_I}$ are input and output matrices designed to implement the four conditions shown in Fig. 4.

3. IMAGE RESIZING EXPERIMENTS

We now look at the performance of the various resizing operations. In all cases PSNR has been computed after restoring an image to its original size and comparing the result with the original image (e.g. in the case of downsizing by 1/2 we restored the smaller

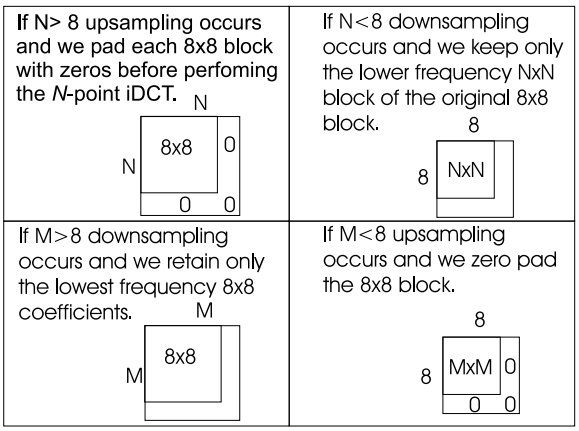

Fig. 4. Four possible conditions when N and M are not equal to 8.

images to their original size using our resizing algorithm with N equal to 8 and M equal to 16). If we wish to resize an image by a factor of 1/2 (i.e. downsize by a factor of two), we can choose $N = 4$ and $M = 8$. Using these factors we produce a mapping that conceptually looks like the following (see Fig. 3):

$$\frac{1}{2}\mathbf{G_8}\begin{bmatrix} \mathbf{H_4} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_4} \end{bmatrix}\begin{bmatrix} \hat{\mathbf{B}}_1 & \hat{\mathbf{B}}_2 \\ \hat{\mathbf{B}}_3 & \hat{\mathbf{B}}_4 \end{bmatrix}\begin{bmatrix} \mathbf{H_4^t} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_4^t} \end{bmatrix}\mathbf{G_8}^{\mathbf{t}}, \quad (4)$$

where $\mathbf{G_8}$ is the 8-point forward DCT matrix and $\mathbf{H_4}$ is the 4-point inverse DCT matrix. This reduces to:

$$1/2\mathbf{\Lambda_O}\begin{bmatrix} \mathbf{K_L}\hat{\mathbf{B}}_1(\mathbf{K_L})^{\mathbf{t}} & \mathbf{K_L}\hat{\mathbf{B}}_2(\mathbf{K_R})^{\mathbf{t}} \\ \mathbf{K_R}\hat{\mathbf{B}}_3(\mathbf{K_L})^{\mathbf{t}} & \mathbf{K_R}\hat{\mathbf{B}}_4(\mathbf{K_R})^{\mathbf{t}} \end{bmatrix}\mathbf{\Lambda_O}^{\mathbf{t}}, \quad (5)$$

where $\hat{\mathbf{B}}_\mathbf{i}$ is the upper right 4 by 4 block of DCT coefficients selected by $\mathbf{\Lambda_I}$, $\mathbf{K_L}$ is $\mathbf{G_{8L}H_4}$ and $\mathbf{K_R}$ is $\mathbf{G_{8R}H_4}$ (i.e. $\mathbf{H_4}$ times the left and right halves of $\mathbf{G_8}$). $\mathbf{\Lambda_O}$ is the 8x8 identity matrix.

Using the factorization technique laid out in [1] we see that we can express $\mathbf{K_L}$ as $C + D$ and $\mathbf{K_R}$ as $C - D$ which allows us to more efficiently compute this mapping. By modifying the pre and post quantization matrices of Fig. 2b (in this case multiplying all the terms in $\mathbf{Q^{-1}}$ by $\sqrt{2}$) we can reduce the number of non-zero and non-unity terms in these matrices to 8 each. The total computational burden is then only 1 mult and 1.25 add per pixel of the original image (see Table 1).

If lower computational complexity is required we can choose $N = 3$ and $M = 6$. Computational complexity and final image quality will be reduced because we are using only the first three coefficients and zeroing out the last five. Additionally the final two coefficients of the eight produced at the end of the transcoding will always be zero (see Fig. 4). Conceptually this mapping has the same form as (5) with $\mathbf{K_L}$ defined as $\mathbf{G_{6L}H_3}$ and $\mathbf{K_R}$ as $\mathbf{G_{6R}H_3}$. $\hat{\mathbf{B}}_\mathbf{i}$ is the upper right 3 by 3 block of DCT coefficients selected by $\mathbf{\Lambda_I}$. $\mathbf{\Lambda_O}$ will now be used to zero pad the 6 by 6 blocks to 8 by 8 blocks. Using the same factorization and pre and post multiplication techniques as before, we can reduce the number of non-zero and non-unity terms in the two matrices C and D to 3 and 6, respectively. This results in a computational complexity of 0.42 mults and 0.56 adds per pixel of the original image (see Table 1).

Table 1. Comparison of PSNR and complexity for various choices of N, M, and Park's algorithms. See Fig. 5 for images.

Algorithm	Scale Factor	PSNR (in dB)		# of Ops
		Lena	Boat	
Park[2]	1/3	31.53	28.74	2.39M+3.67A
N=3,M=9	1/3	31.53	28.61	1.09M+1.12A
N=4,M=12	1/3	31.53	28.73	1.74M+1.77A
Park[2]	1/2	35.36	31.96	3.38M+3.75A
Park[2]	1/2	35.08	31.83	1.89M+2.06A
N=3,M=6	1/2	32.08	29.18	0.42M+0.56A
N=4,M=8	1/2	35.02	31.62	1.00M+1.25A
N=5,M=10	1/2	35.34	31.94	1.41M+1.76A
N=4,M=6	2/3	34.64	31.41	3.00M+2.83A
N=6,M=9	2/3	38.63	35.36	8.50M+7.91A
N=4,M=5	4/5	34.23	31.07	3.60M+3.30A

If better final image quality is desired we can choose $N = 5$ and $M = 10$. Computational complexity and final image quality will increase because we are using five coefficients (more information than either of the first two cases) and zeroing out the last three. The final two coefficients of the ten point DCT will not be computed (see Fig. 4). Conceptually this mapping also has the same form as (5) except that $\mathbf{\Lambda_O}$ is absorbed into the $\mathbf{K}$ entries to eliminate the computation of the last two terms of the 10-point DCT. $\mathbf{K_L}$ is defined as $\mathbf{G_{10LT8}H_5}$ and $\mathbf{K_R}$ as $\mathbf{G_{10RT8}H_5}$, where $\mathbf{G_{10L8}}$ and $\mathbf{G_{10R8}}$ are the top 8 rows of the left and right halves of the 10-point DCT matrices, respectively. $\hat{\mathbf{B}}_\mathbf{i}$ is the upper right 5 by 5 block of DCT coefficients selected by $\mathbf{\Lambda_I}$. The number of non-zero and non-unity terms in the matrices C and D is 8 and 12, respectively, resulting in a computational complexity of 1.46 mults and 1.76 adds per pixel of the original image (see Table 1).

4. MULTIPLIERLESS IMPLEMENTATION USING THE BINDCT

Rather than use the standard DCT and iDCT we can use multiplierless approximations to these transforms to produce the desired resizing. We chose the binDCT [4]as a multiplierless approximation to the DCT for this application. The binDCT is derived from Loeffler's and Chen's factorizations of the DCT. Each of these factorizations is parameterized and approximate the DCT with varying levels of fidelity. Experiments show that very little final image quality is sacrificed when using these transform approximations. Thus rather than using floating point multiplications we can use shifts and adds to perform the desired resizing operation. The binDCT is a biorthogonal transform rather than an orthogonal transform like the DCT but it still possesses many of the same symmetry properties of the DCT and the factorizations discovered thus far can be directly applied to the binDCT. We can replace $\mathbf{G_8}$ and $\mathbf{H_4}$ in (4) with their binDCT counterparts.

$$\frac{1}{2}\mathbf{G_{8b}}\begin{bmatrix} \mathbf{H_{4b}} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_{4b}} \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{B}}_1 & \tilde{\mathbf{B}}_2 \\ \tilde{\mathbf{B}}_3 & \tilde{\mathbf{B}}_4 \end{bmatrix}\begin{bmatrix} \mathbf{H_{4b}^t} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_{4b}^t} \end{bmatrix}\mathbf{G_{8b}}^{\mathbf{t}},$$

where $\mathbf{G_{8b}}$ is the 8-point forward binDCT and $\mathbf{H_{4b}}$ is the 4-point inverse binDCT. Since the binDCT is proportional to the true DCT

Fig. 5. The top row shows the original images. Second row: transcoded images scaled by 4/5 (N=4, M=5). Third row: images scaled by 2/3 (N=6, M=9). Fourth row: images scaled by 1/2 (N=5, M=10) and 1/3 (N =3, M =9.

we have to premultiply the DCT coefficients before applying the binDCT modified mapping ($\tilde{\mathbf{B}}_\mathbf{i}$ is the coefficient block resulting from the premultiplication of $\hat{\mathbf{B}}_\mathbf{i}$) and postmultiply the remapped coefficients afterward. Since compressed coefficients are stored in a quantized state we can absorb these multiplications into the quantization matrices $\mathbf{Q}$ and $\mathbf{Q}^{-1}$ (see Fig. 2).

Several images were tested using various versions of the binDCT. In [4] 9 different versions of the binDCT were produced for both Chen's and Loeffler's factorization and ordered according how well they approximate the true DCT. They were then labeled CX or LX (where X ranges from 1 to 9 with 1 being the finest and 9 being the coarsest approximation). In this application the results indicated that approximations down to 4 or 5 of Chen's factorization and 3 of Loeffler's factorization provided quality nearly indistinguishable from that of [1] (see Table 2). The penalty in image quality for using integer operations rather than floating point calculations is very low and many levels of binDCT approximations exist permitting a tradeoff between required final image quality and allowable complexity.

Table 2. Comparison of transcoding image quality (PSNR in dB) between our binDCT-method and Dugad's[1] DCT-based method.

Algorithm	C1	C5	Dugad[1]
Lena	34.63	34.20	34.69
Boat	31.46	31.17	31.50
Barbara	25.51	25.46	25.52
Goldhill	31.85	31.68	31.89
# of Ops	4S+3.88A	3.88S+3.94A	1.25M+1.25A

5. CONCLUSION

A generalization of the method of [1] to arbitrary scale factors has been presented that can be used to produce mappings with lower complexity and/or better final image quality than other current methods. This generalization produces a mapping from the 8x8 DCT domain to the 8x8 DCT domain with arbitrary scale factors with the mapping based upon the synthesis of a combined transform and resizing at either or both transform stages. We have also shown the advantages of implementing the method of [1] in a multiplierless fashion. These advantages include 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, transformation into the DCT domain). Our method also allows a wide range of implementation complexity/final image quality choices to be made. These performance points can be varied by choosing different values of N and M for the N-point iDCT and M-point DCT that are used to construct the mapping for a given scale factor and by choosing DCT approximations with various levels of implementation complexity.

6. REFERENCES

[1] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 461–474, Apr. 2001.

[2] Y. Park H. Park and S. Oh, "L/m-fold image resizing in block-dct domain using symmetric convolution," *IEEE Trans. Image Proc.*, vol. 12, pp. 1016–1034, Sep. 2003.

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

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

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

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

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