Some New Algorithms for Processing Images in the Transform Compressed Domain

Shih-Fu Chang

Department of Electrical Engineering
& Center for Telecommunications Research
Columbia University
New York, NY 10027

1. Introduction

In many advanced multimedia applications such as video servers, image databases, and video conferencing, image and video data are often compressed. Interactive retrieval and visualization of vast amount of visual data require efficient image manipulation algorithms. Typical manipulation functions include rate/format conversion, special visualization effects (e.g., image scaling, translation, rotation, warping), and multi-object compositing (e.g., video mixing in the video bridge for multi-point video conferencing).

A major underlying component in the image/video compression algorithms (such as JPEG, MPEG and H.261 [8,9,10]) is the Discrete Cosine Transform (DCT). In the transform domain, lots of coefficients are small and are often quantized to zeroes. To take advantage of this lower data rate in the transform domain, our prior work has studied techniques for manipulating image in the transform compressed domain [1,2,3]. Given transform coefficients (e.g., DCT coefficients) of the input images, we can calculate the transform coefficients of the desired output images without converting images back to the uncompressed spatial domain. The overall implementation cost can thus be greatly reduced. In particular, we have derived a whole set of algorithms for performing image filtering, scaling, translation, and overlapping directly in the transform domain. These algorithms are applicable to any orthogonal separable transforms such as DCT, DFT, and discrete wavelet transform. Independent work [4,6,7] have proposed the transform-domain approach for a subset of operations mentioned above.

This paper expands our prior results to advanced image manipulation functions such as rotation, shearing, and line-wise special effects widely used in video production. These manipulation functions requires more sophisticated mathematical modeling because they cannot be treated as regular 2D separable filtering. We derive mathematical formulas for doing these manipulations in the transform domain and discuss their impact on computational complexity and image quality.

2. Background

Block-based transforms such as DCT can be described as the following

$$T(A) = \bar{A} = CAC^t \qquad (1)$$

CU/CTR Technical Report 390-94-37
also submitted to SPIE Visual Communications and Image Processing, '95

where T is the transform operation, A is the input image block (say N pixels by N pixels), $\bar{A}$ is the image transform[1], C is the transform matrix, and C^t is the matrix transpose ($C^t = C^{-1}$ for unitary transforms). This type of orthogonal transform have some nice properties. First, the orthogonal transform is distributive to matrix multiplication, i.e.,

$$T(AB) = T(A)T(B) \tag{2}$$

This property is useful in deriving many transform-domain manipulations. Second, the correlation operation can be done equivalently in both the spatial and the transform domains, i.e.,

$$A \bullet B = \bar{A} \bullet \bar{B} \tag{3}$$

where $\bullet$ stands for the correlation operation. This property is useful in image pattern matching based on the transform coefficients, although the boundary effect requires some special considerations [3].

3. Linear Filtering and General Geometrical Transformation of Images

Linear filtering can be used to model many useful manipulation functions such as scaling and smoothing. A convenient way to describe 2D separable linear filtering of images is via pre-matrix and post-matrix multiplications. For example, a filtered output image block, B, can be calculated as

$$B = \sum_i V_i A_i H_i \tag{4}$$

where A_i are neighboring image blocks which have contribution to the target output block, V_i are the vertical filters and H_i the horizontal filters. The summation range depends on the filter length. Scaling is a special case of linear filtering. For example, a 1/2 by 1/2 down scaling (for N=8) can be implemented by using the following filter matrices.

$$V_1 = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad V_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

$$H_1 = \left(V_1\right)^t \qquad H_2 = \left(V_2\right)^t \tag{5}$$

1. Regular capital letters represent the spatial image blocks. Letters with the upper bar represent their transform coefficients.

Scaling of different factors can also be easily implemented in the transform domain by choosing appropriate filter matrices (V_i and H_i) of Equation 4. Translation with arbitrary distance (not necessarily integral multiples of the block size) can be done in the transform domain as well [3].

However, advanced image manipulations such as rotation, shearing, and line-wise irregular operations cannot be modeled in the form of separable matrix multiplications of Equation 4. For example, in a horizontal shearing operation, different rows of an image block are shifted by different amount, as shown in Figure 1. This is contrary to the constant scaling factor and filtering effect applied to each row of an image block. To overcome this problem associated with the line-wise irregular operations, we adopt the *divide and conquer* approach. We first extract each row from an image block by multiplying with a special pre-matrix W_i , defined as

$$W_i(k, l) = \delta(k - i, l - i) \tag{6}$$

There is only one non-zero element in the matrix. $W_i A$ extracts the i-*th* row of image block A and set all remaining rows to zeros. After this step, we shift each row independently with the correct distance (as required by the shearing operation) by post-multiplying with a column shifting matrix. For example, the following N by (i+N) matrix M_i shifts the original N by N matrix to the right by i columns and pad zeroes in the first i columns.

$$M_i = [\text{ i zero columns } | I | \ldots] \tag{7}$$

where I is the N by N identity matrix. In practical implementations of shearing and rotation, interpolation (i.e., filtering) and resampling are also performed. In that case, the identity matrix in Equation 7 may be simply replaced by the filter matrix, as shown in Equation 4. The complete output image after shearing becomes

$$\left[B_1 \middle| B_2 \middle| \cdots\right] = \sum_i W_i A_i M_i \tag{8}$$

where B_i are N by N square matrices. The actual width of the output image depends on the exact shearing parameter. Each output target image block now can be calculated as a matrix product sum. For example,

$$B_1 = \sum_i W_i A S_i \tag{9}$$

where W_i extract individual rows, S_i shifts each row with different distances, and the summation combines contributions from different rows. Note that S_i are simply sub-matrices of M_i. For example, to calculate B_1 , S_i is simply the first N columns of M_i.

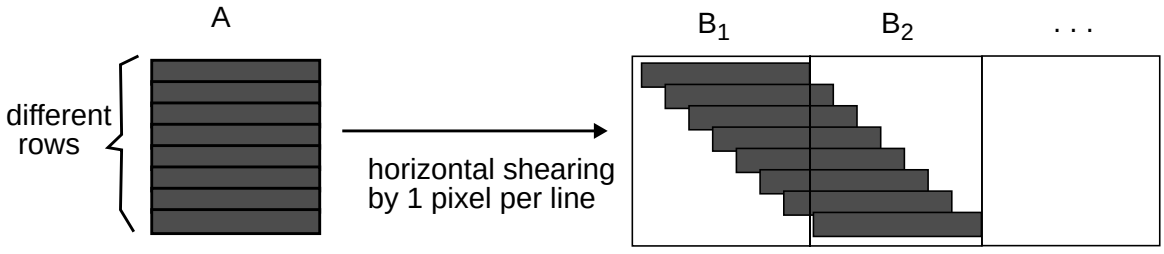

FIGURE 1 One-dimensional horizontal shearing by 1 pixel per line. The result is spread over several target blocks.

By combining the row-extraction and row-wise shifting operations, we now are able to model the shearing operation with regular matrix multiplications. We can use the same equation in the transform domain by simply applying the distributive property of orthogonal transform in Equation 2. As a result,

$$\bar{B}_1 = \sum_i \bar{W}_i \bar{A} \bar{S}_i = \sum_{k,l} \bar{A}(k,l) \left(\sum_i \bar{W}_i(-,k)\, \bar{S}_i(l,-) \right) = \sum_{k,l} \bar{A}(k,l)\, \bar{\phi}_{k,l} \tag{10}$$

where $\bar{\phi}_{k,l}$ is an N by N matrix and is equal to the sum of N column-row product matrices. Coefficients of $\bar{\phi}_{k,l}$ can be calculated in advance and stored in a table. It is worth mentioning that the final summation in Equation 10 involves non-zero $\bar{A}(k,l)$ coefficients only. The number of non-zero transform coefficients is usually small in practical compression. In addition, many elements of $\bar{\phi}_{k,l}$ are small (since in the transform domain) and thus can be approximated by zeroes. Both these two facts imply that there is great potential to reduce the overall implementation complexity of this transform-domain equation.

In practice, image rotation can be implemented by multi-pass techniques such as a horizontal shearing-scaling followed by a vertical shearing-scaling, as shown in Figure 2 [5]. In general, one image block in the rotated coordinate space has contributions from 4-6 neighboring image blocks in the original coordinate space. Since shearing and scaling can both be implemented in a matrix product form of Equation 9. The composite function of these multi-pass processes can also be modeled by the matrix product form. As a result, one output block of the rotated image can be calculated as

$$\bar{B} = \sum_i \left(\sum_{k,l} \bar{A}_i(k,l) \left(\bar{\phi}_{k,l} \right)^i \right) \tag{11}$$

where $\bar{A}_i$ are transform coefficients of those blocks which contribute to the output target block. In other words, given transform coefficients of the original images blocks A_i, we can use the above equation to calculate the transform coefficients of the rotated image blocks B directly without decoding.

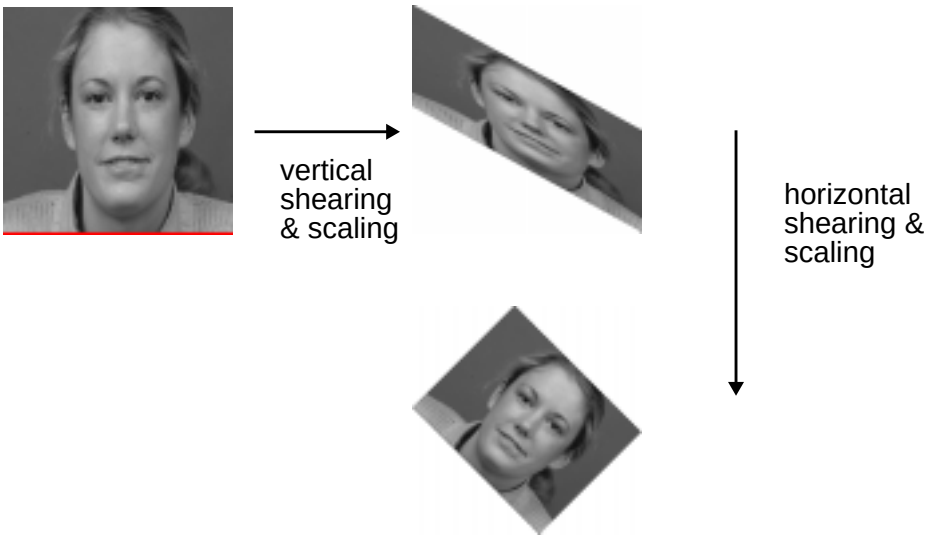

FIGURE 2 Image rotation implemented by multi-pass shearing and scaling, which can be directly implemented in the transform-compressed domain.

The above concept of line extraction followed by regular linear filtering can be applied to other general line-wise image manipulations (e.g., image warping) as well.

4. Performance Analysis and Extensions

Due to the much lower data rate in the transform domain, the transform-domain manipulations provide great potential in reducing overall computational complexity. The actual computational complexity of the proposed transform-domain algorithms depends on the image compression rate (i.e., the number of non-zero transform coefficients) and the sparsity of manipulation matrices $\bar{\phi}_{k,l}$. We are currently conducting simulations to get actual statistics of these critical factors. Special strategies can be explored to find optimal approximation of image manipulation matrices ($\bar{\phi}_{k,l}$) to increase the matrix sparsity and thus reduce the overall complexity.

The proposed transform-domain manipulation algorithms will not affect the image quality except for the minor effect of the computing round-off errors. The transform-domain approach will possibly reduce the latency since the processing delay is reduced.

For video coded by orthogonal transform plus inter-frame motion compensation, we have proposed a modified decoding algorithm to partly decode the video back to the transform domain and then apply the transform-domain manipulation techniques [2,3].

5. References

1. S.-F. Chang, W.-L. Chen and D. G. Messerschmitt, "Video Compositing in the DCT domain," *IEEE Workshop on Visual Signal Processing and Communications*, Raleigh, NC, pp. 138-143, Sep. 1992.
2. S.-F. Chang and D. G. Messerschmitt, "A New Approach to Decoding and Compositing Motion Compensated DCT-Based Images," IEEE ICASSP, Minneapolis, Minnesota, April, 1993.
3. S.-F. Chang and D. G. Messerschmitt, "Manipulation and Compositing of MC-DCT Compressed Video," accepted for publication in IEEE Journal of Selected Areas in Communications, 1994.
4. B. Chiptrasert and K.R. Rao, "Discrete Cosine Transform Filtering," *Signal Processing*, Vol. 19, No. 3, pp. 233-45, Mar. 1990.
5. J.D. Foley, A. V. Dam, S. K. Feiner and J.F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed., Addison-Wesley, 1990.
6. J.B. Lee and B.G. Lee, "Transform Domain Filtering Based on Pipelining Structure," IEEE Transactions on Signal Processing, pp. 2061-4, Vol. 40, No. 8, Aug. 1992.
7. B.C. Smith and L. Rowe, "Algorithms for Manipulating Compressed Images," IEEE Computer Graphics and Applications, pp. 34-42, Sept. 1993.
8. G.K. Wallace, "The JPEG Still Picture Compression Standard," Communications of the ACM, Vol. 34, No. 4, April 1991.
9. M. Liou, "Overview of the p×64 kbits/s Video Coding Standard," Communications of the ACM, Vol. 34, No. 4, April 1991.
10. D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, Vol. 34, No.4, April, 1991.