RC22238 (W0111-023) November 7, 2001
Computer Science

IBM Research Report

Fast DCT-Domain Image Shifting and Merging

Timothy J. Trenary, Charles A. Micchelli, J. Q. Trelewicz, Marco Martens, Joan L. Mitchell
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

FAST DCT-DOMAIN IMAGE SHIFTING AND MERGING

Timothy J. Trenary, Charles A. Micchelli, J. Q. Trelewicz, Marco Martens, Joan L. Mitchell

IBM T. J. Watson Research Center, Route 134, Yorktown Heights, NY 10598, USA

ABSTRACT

In high-speed color image processing applications, it is desired to manipulate DCT-domain images in real time. Because of throughput requirements, operations required for the building of a printer page, such as merging and shifting, cannot be performed in the sampled domain – the overhead associated with the DCT slows computation. In this paper we present an efficient method for the merging or shifting of JPEG-compressed images, performed in the DCT domain to avoid additional overhead. The algorithm exploits quantization to achieve further gains. The resulting algorithm can save between 25% and 75% of the required operation cycles in a typical image processing system.

1. INTRODUCTION

Fast color image processing is required in a number of applications. For example, high-speed color printing, where full-image pages are printed at more than 100 pages per minute, requires pages to be built and imaged very rapidly. Web-based e-commerce applications, where high-quality images must be downloaded and displayed quickly, give another class of examples, where the application developer has little or no control over the client equipment that must perform the rapid display of images, precluding the use of specialized hardware. Many of the images used by these applications are stored and transmitted in compressed format, such as the JPEG standard. The images must be manipulated for the building of pages, including merging and shifting images within a larger page. While straightforward in the sampled domain, these operations introduce challenges in the Discrete Cosine Transform (DCT) domain, since partial DCT blocks may need to be combined with other partial DCT blocks without destroying image content. This paper focuses on processing images, such as those compressed with JPEG, in the DCT domain. However, because the methods described herein may be extended to other linear orthogonal transforms, the algorithms are useful for a wide class of image compression formats, including algorithms based on Fast Fourier Transform (FFT) and certain wavelet or other transform bases.

DCT-domain processing algorithms for image are well known. High-quality scaling in the DCT domain (e.g., [1]) for videoconferencing applications has been an area of much recent activity. In [2], a multiplicative alpha mask, which determines the transparency of pixels, is implemented as a convolution in the DCT domain. In [3], nonseparable masks are implemented as sparse matrix multiplications in the DCT domain. For masking and merging applications, the algorithms developed in this manuscript[2] make further reductions in computational complexity by exploiting the nature of merging and shifting in our applications, where blocks are masked along rows and columns, which may not align with the DCT block boundaries. Our algorithms are further distinguished by their combination with quantization and consideration of coefficient magnitude for additional reduction in computational requirements.

2. MERGE AND SHIFT OPERATORS IN THE DCT DOMAIN

The following notation is used in the paper. Let $D : \Re^8 \to \Re^8$ be the 1-D DCT, given by

$$D_{ux} = C_u \cos\left(\frac{\pi}{16} u(2x+1)\right), \quad u, x \in 0, 1, \ldots, 7$$

where

$$C_u = \begin{cases} 1/(2\sqrt{2}) & \text{if } u = 0 \\ 1/2 & \text{otherwise} \end{cases}$$

This matrix is unitary, so that the inverse of the matrix is equal to its transpose, $D^{-1} = D^T$. The input domain of D comprises the real or sampled domain; e.g., a digital photograph. The output is said to be in the DCT domain. Vectors or matrices in the real domain are denoted as F. The vector of points in the DCT domain is denoted by $\tilde{F} = DF$, and the matrix by $\tilde{F} = DFD^T$.

[1]Timothy J. Trenary is with IBM Printing Systems Division. Charles A. Micchelli is Emeritus Research Staff, T. J. Watson Research Center, currently with the State University of New York at Albany as Professor of Mathematics. This work was partially supported by the U. S. National Science Foundation, under grant DMS 997342. J. Q. Trelewicz, the contact author, is with IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, trelewic@us.ibm.com. J. Q. Trelewicz and Joan L. Mitchell are on temporary assignment with the IBM Printing Systems Division. Marco Martens is actually located physically in Yorktown Heights.

[2]Patent applications have been filed.

Figure 1: A shift by σ pels.

Merge operators in one dimension are developed first because of notational simplicity. The 2-D merge operators are shown to be a straightforward extension with the application of appropriate lemmas. Merges along horizontal cuts are discussed, since merges along vertical cuts are a triviality, given the horizontal cut algorithms. The operators defined in this section may also serve as shift operators, since a shift along a horizontal or vertical direction is essentially a merge from two adjacent blocks. A general merge operation in two dimensions can be performed by applying a vertical and horizontal algorithm as described above. All merge operations described herein allow such a decomposition. Figure 1 shows an example of the merging of two DCT blocks required to realize a shift.

Define matrix U_σ, which masks the σ upper rows of its argument, as follows:

$$(U_\sigma)_{x,y} = \begin{cases} 1 & y \in \{0, 1, ..\sigma - 1\}, x = y \\ 0 & \text{otherwise} \end{cases}.$$

The matrix $I - U_{8-\sigma}$ masks the σ lower rows of its argument. Similarly, define V_σ, which masks the σ lower rows of its argument as follows:

$$(V_\sigma)_{x,y} = \begin{cases} 1 & y \in \{0, 1, ..7 - \sigma\}, x = \sigma + y \\ 0 & \text{otherwise} \end{cases}.$$

Similarly, $V_\sigma^T = V_{-\sigma}$, and $V_{-\sigma} \equiv V_{8-\sigma}$ masks the upper σ rows of its argument. Then, the merge operators may be denoted $M = (M_\sigma, M_{8-\sigma})$, where $M_\sigma \in \{U_\sigma, I - U_{8-\sigma}, V_{\pm\sigma}\}$ determines whether the upper or lower rows of the argument are used in the merge.

Suppose that the composite is to be formed from the elements of G and H. The merged sample vector F is given by

$$F = M \begin{pmatrix} G \\ H \end{pmatrix} = M_\sigma G + M_{8-\sigma} H.$$

Because D is unitary,

$$\begin{aligned} \tilde{F} &= DM_\sigma G D^T + DM_{8-\sigma} H D^T \\ &= \tilde{M}_\sigma \tilde{G} + \tilde{M}_{8-\sigma} \tilde{H} \equiv \tilde{M} \begin{pmatrix} \tilde{G} \\ \tilde{H} \end{pmatrix}, \end{aligned}$$

where the merge operator in the DCT domain may be denoted $\tilde{M}$.

A computational improvement on this method can be achieved by working with $S = G + H$ and $D = G - H$, using the sum-difference merge operator M' defined as $M' = (M_\sigma + M_{8-\sigma}, M_\sigma - M_{8-\sigma})/2$. Then

$$\tilde{F} \equiv \tilde{M}' \begin{pmatrix} \tilde{S} \\ \tilde{D} \end{pmatrix}.$$

The advantage of working with $\tilde{M}'$ is that many of the entries in the operator matrix will be zero, reducing significantly the number of operations required for the sum and difference computation. For example, using 8×8 DCT blocks and shifting by 4 positions in one dimension, 75 of 128 entries of $\tilde{M}'$ are zero. The significant number of zero entries occurs because of the structure of the DCT basis, since the block matrices in M' are essentially low-frequency sample matrices.

3. QUANTIZATION

Further computational improvements are realized by incorporating quantization into the merge calculations. Since quantization involves, in effect, scaling the results prior to integer truncation, the incorporation of the quantization scales allows quantization to be performed through integer truncation at the output of the merge procedure.

It is shown in this section that quantization can be incorporated with the other operations, so that the DCT-domain operator $\hat{M}$ becomes an operator of the same form as $\tilde{M}$. Specifically, the matrices of $\hat{M}$ have the same zero entries as $\tilde{M}$, since the matrices Q and $\hat{Q}$ are diagonal.

Let $[\cdot]$ be the integer truncation operator. Consider a 1-D merge operator

$$\tilde{F} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \tilde{S} \\ \tilde{D} \end{pmatrix},$$

with S, D defined as above. Let $q \in \Re^8$ be a quantization vector and let $Q = \text{diag}(q)$ be the corresponding diagonal matrix. Because the quantization is the same for both vectors $\tilde{G}$ and $\tilde{H}$,

$$Q\tilde{G} \pm Q\tilde{H} = Q(\tilde{G} \pm \tilde{H}) \in \left\{ Q\tilde{S}, Q\tilde{D} \right\}.$$

Assume that $\tilde{G}, \tilde{H} \in Z^8$ is input quantized with respect to q, both quantized on the same scale. This is not a restriction if we are considering shift operators. In the case of merging two picture, we have to quantize them on the same scale before performing the merge for the following improvement to be successful.

Let $\hat{q} \in \Re^8$ be the desired quantization vector for the output and $\hat{Q} = \text{diag}(\hat{q})$ be the corresponding diagonal matrix. An elementary computation of the quantized output leads to

$$\tilde{F} = \left[\begin{pmatrix} \hat{Q}^{-1}AQ & \hat{Q}^{-1}BQ \end{pmatrix} \begin{pmatrix} \tilde{S} \\ \tilde{D} \end{pmatrix} \right],$$

where the square bracket indicate the integer part, the actual quantization. Let $\hat{A} = \hat{Q}^{-1}AQ$ and $\hat{B} = \hat{Q}^{-1}BQ$. Then by incorporating the dequantization and quantization we get

$$\tilde{F} = \left[\begin{pmatrix} \hat{A} & \hat{B} \end{pmatrix} \begin{pmatrix} \tilde{S} \\ \tilde{D} \end{pmatrix} \right].$$

The matrices $\hat{A}$ and $\hat{B}$ have the same zero entries as the matrices A and B, since the matrices Q and $\hat{Q}$ are diagonal. In particular,

$$\begin{aligned} \hat{A}_{uv} &= q_v/\hat{q}_u A_{uv} \\ \hat{B}_{uv} &= q_v/\hat{q}_u B_{uv}, \end{aligned}$$

so that the matrices $\hat{A}$ and $\hat{B}$ can be computed once and reused throughout the image.

In a traditional JPEG implementation, the values would first be scaled by q_v, for a total of 8 multiplies for the vector, later scaled down by $\hat{q}_u$, for another 8 multiplies. Since quantization is combined with the matrix operations in our approach, the scaling required for quantization essentially takes no additional operations.

4. EXAMPLES

The shift by four positions, which is equivalent to merging the four left columns of one block with the four right columns of the adjacent block, results in $\tilde{M}'$ having 75/128 (about 58%) of its elements zero. Assuming 1-D DCT and no non-zero DCT coefficients, the calculation of S and D require a total of 16 addition operations. A brute force application of the matrix operation adds 52 multiplies and 58 additions. As stated above, quantization requires no additional operations. On the other hand, conversion to the pel domain requires 2 each inverse and forward DCTs of G and H, in addition to any cycles required for the shift of the samples (not included in the summary numbers). Additionally, dequantization and requantization require 24 multiplications. This conventional approach, using the binDCT-C fast DCT [4] and the AAN fast DCT [5] is summarized in Table 1. The percentage of operations saved through our approach, called "SD", is shown in the last column of the table. The table assumes one execution cycle for each of multiplications, additions, and shifts, a reasonable assumption on modern DSPs and RISC microprocessors. A finessed application of our algorithm, using the Fast Paths discussed below, would save even more cycles through intelligent grouping of operations.

The non-zero coefficients of the matrix in SD lie between 0.01 and 0.5. Thus, the integer methods described in [6] could be useful for a hardware or an embedded software implementation of the transform. Those methods allow the implementation of multiplierless transforms, which have real estate advantages in FPGA implementations. The methods also guide the allocation of precision in calculations, allowing efficient low-error implementation in narrow-bus embedded processors.

Table 1: Operation counts for shift example.

Algorithm	Mult	Add	Shift	SD saves
Tran	24	120	52	$> 35\%$
AAN	20+24	116	0	$> 20\%$
SD	52	58+16	0	-

In contrast, the operator for shift by one position has only about 25% of its elements zero. The algorithms for each of the σ values can be coded and optimized in advance, so that a different approach can be chosen for those cases where the DCT domain algorithm does not perform best.

5. NATURAL IMAGES

A further reduction in computation cycles may be achieved with natural images. The typical quantized coefficient vector from a natural image has the property that only the first several coefficients are non-zero. One could even suppress entries; e.g., if some $\tilde{G}(i,j)$ with $(i,j) \in R$ is small, and if their omission will not appreciably affect image quality.

Assume that only the first $\tau \in 1, 2, \ldots 8$ entries of the input vectors $\tilde{G}$ and $\tilde{H}$ are non-zero. Let $P_\tau : \Re^8 \to \Re^\tau$ be the projection onto the first τ entries. Then

$$P_\tau \tilde{G} \pm P_\tau \tilde{H} = P_\tau(\tilde{G} \pm \tilde{H}) \in \left\{ P_\tau \tilde{S}, P_\tau \tilde{D} \right\}.$$

Consider the merge operator on the DCT domain

$$\tilde{F} = \left[\begin{pmatrix} \hat{A} & \hat{B} \end{pmatrix} \begin{pmatrix} \tilde{S} \\ \tilde{D} \end{pmatrix} \right],$$

with dequantization and quantization incorporated. Let $\tilde{S}_\tau = P_\tau \tilde{S}$ and $\tilde{D}_\tau = P_\tau \tilde{D}$ be $\tau-$vectors and $\hat{A}_\tau$, $\hat{B}_\tau$ be the $8 \times \tau$ matrices formed by the first τ columns of respectively $\hat{A}$ and $\hat{B}$. An elementary computation shows

$$\tilde{F} = \left[\begin{pmatrix} \hat{A}_\tau & \hat{B}_\tau \end{pmatrix} \begin{pmatrix} \tilde{S}_\tau \\ \tilde{D}_\tau \end{pmatrix} \right].$$

This improvement is referred to as "Fast Paths". The improvement of the Fast Paths comes from the effective reduction in matrix size resulting from the omission of coefficients, similar to the technique used in [2] for reducing operations required for convolution.

It should be noted that Fast Paths considerations reduce both the number of operations required for our brute-force DCT-domain approach and for the conventional approach; however, while Fast Paths reduce all of our steps

(sum/difference of DCT coefficients, and matrix operations required for shift/merge/quantization), in the conventional approach, only the operations required for the DCTs are reduced, since the pel-domain blocks may be fully populated. Specifically, in the example described above, assume that only the first two DCT coefficients are non-zero. Then the calculation of S and D require only 4 addition operations, the matrix operation adds 13 multiplications and 11 additions. In contrast, the AAN fast IDCT requires a total of 8 multiplcations and 30 additions (assuming that operations associated with zero coefficients are dropped), and the forward DCT requires the full total 10 multiplies and 58 additions. Dequantization requires only 4 multiplies, but requantization requires all 8. Thus, our Fast Paths approach uses about 75% fewer operations!

6. EXTENSION TO TWO DIMENSIONS

We now extend the description to include 2-D merge operators and the corresponding action on the DCT-domain. In the 2-D case the input space, again called real domain, is the space of 8×8 matrices, $\mathcal{M}(8)$. The 2-D DCT-operator, denoted by $D_2 : \mathcal{M}(8) \to \mathcal{M}(8)$, is

$$D_2 : \tilde{G} \mapsto (D(D\tilde{G})^T)^T,$$

where D is the 1-D DCT operator. The output domain is called the DCT-domain.

It is sufficient to derive the DCT-domain action corresponding to purely vertical merge operators, since general 2-D shifts or merges may be constructed as the composition of vertical and horizontal shifts or merges. Let $M = \begin{pmatrix} A & B \end{pmatrix}$ be a 1-D merge operator. This merge operator defines a 2-D merge operator as follows. Let $G_1, G_2 \in \mathcal{M}(8)$ be elements in the real domain, G_1 is thought to be a block above G_2. Now we can merge the G_1 and G_2 by columns:

$$F = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$$

or $F = AG_1 + BG_2$. This 2-D merge operation can be described completely on the DCT domain. In particular, let $\tilde{F} = D_2(F)$, $\tilde{G}_1 = D_2(G_1)$, $\tilde{G}_2 = D_2(G_2)$ and let $\tilde{M} = \begin{pmatrix} \tilde{A} & \tilde{B} \end{pmatrix}$ be the representation on the 1-D DCT domain of the merge operator M. In particular, $\tilde{A} = DAD^{-1}$ and $\tilde{B} = DBD^{-1}$.

Lemma 6.1 *The 2-D vertical merge operation can be performed on the 2-D DCT domain by applying the corresponding 1-D merge operator on the columns:*

$$\tilde{F} = \begin{pmatrix} \tilde{A} & \tilde{B} \end{pmatrix} \begin{pmatrix} \tilde{G}_1 \\ \tilde{G}_2 \end{pmatrix} = \tilde{A}\tilde{G}_1 + \tilde{B}\tilde{G}_2.$$

The proof of this Lemma is a computation, which follows from $\tilde{F} = (D(DF)^T)^T = DFD^{-1}$.

The application of the operator by columns provides a natural parallelization in the algorithm that allows further computational gains to be made through application of the methods described in [7], which describe a method by which parallel operations can be performed in scalar processors through use of a vector representation. The vector representation allows computation to be reduced by several times, dependent on the size of the registers in the processor and the required precision in the calculations.

A general 2-D merge operation is an operation which combines a vertical 1-D merge operator $\tilde{M}_v$ and a horizontal 1-D merge operator $\tilde{M}_h$. Let $\tilde{M}_v = \begin{pmatrix} \tilde{A}_v & \tilde{B}_v \end{pmatrix}$ and $\tilde{M}_h = \begin{pmatrix} \tilde{A}_h & \tilde{B}_h \end{pmatrix}$ be the representations of the 1-D merge operators in the DCT domain. Now we will describe the action of this 2-D merge operator $\tilde{M}$ on the DCT-domain. Let

$$\tilde{G} = \begin{pmatrix} \tilde{G}_{11} & \tilde{G}_{12} \\ \tilde{G}_{21} & \tilde{G}_{22} \end{pmatrix}$$

be a square consisting of 4 blocks in the DCT-domain, $\tilde{G}_{ij} \in \mathcal{M}(8)$. The following Lemma is obtained by applying the previous Lemma to the vertical and horizontal parts of this 2-D merge.

Lemma 6.2 *The merge operator $\tilde{M}$ on the 2-D DCT-domain is*

$$\tilde{F} = [\tilde{M}_h[\tilde{M}_v\tilde{G}]^T]^T = \tilde{M}_v\tilde{G}\tilde{M}_h^T$$

or in block form

$$\tilde{F} = \begin{pmatrix} \tilde{A}_v & \tilde{B}_v \end{pmatrix} \begin{pmatrix} \tilde{G}_{11} & \tilde{G}_{12} \\ \tilde{G}_{21} & \tilde{G}_{22} \end{pmatrix} \begin{pmatrix} \tilde{A}_h & \tilde{B}_h \end{pmatrix}^T.$$

We finish this section with the application of the three 1-D improvements, S and D, Quantization, and Fast Paths, to the 2-D case. The example uses the notation for $\tilde{G}$, $\tilde{M}_h$, and $\tilde{M}_v$ as above.

6.1. Sum-Difference

Consider the transformed input

$$\begin{pmatrix} \tilde{S}_1 & \tilde{S}_2 \\ \tilde{D}_1 & \tilde{D}_2 \end{pmatrix} = \begin{pmatrix} \tilde{G}_{11} + \tilde{G}_{21} & \tilde{G}_{12} + \tilde{G}_{22} \\ \tilde{G}_{11} - \tilde{G}_{21} & \tilde{G}_{12} - \tilde{G}_{22} \end{pmatrix}.$$

An elementary computation with the block matrices shows

Lemma 6.3 *The merge operator $\tilde{M}$ on the 2-D DCT-domain is*

$$\begin{aligned} \tilde{F} \quad = \quad & (\tilde{A}_v(\tilde{S}_1 + \tilde{S}_2) + \tilde{B}_v(\tilde{D}_1 + \tilde{D}_2))\tilde{A}_h^T \\ & + (\tilde{A}_v(\tilde{S}_1 - \tilde{S}_2) + \tilde{B}_v(\tilde{D}_1 - \tilde{D}_2))\tilde{B}_h^T, \end{aligned}$$

where $\tilde{A}_h, \tilde{B}_h$ and $\tilde{A}_v, \tilde{B}_v$ are the sum-difference representation of the 1-D horizontal and vertical merge, respectively.

A purely vertical (or horizontal) merge operation becomes the operation consisting of applying the 1-D merge operator on the columns (or on the rows).

6.2. Quantization

Let $Q \in \mathcal{M}(8)$ be a 2-D quantization matrix. Generally speaking quantization matrices do not satisfy special properties. This generality implies that we cannot incorporate the dequantization and requantization completely within the merge matrices; instead, we have to work by columns. If X is a matrix we will denote the k^{th} column of X by X^k and the k^{th} row by ${}^kX$. Moreover let $Q_k = \text{diag}(Q^k)$, the diagonal matrix using the entries of the k^{th} column. The quantized input $\tilde{G}_{ij}$ is given and quantized using the same quantization matrix. Let the transformed input be denoted by

$$\begin{pmatrix} \tilde{S}_1 & \tilde{S}_2 \\ \tilde{D}_1 & \tilde{D}_2 \end{pmatrix}$$

After performing the vertical merges we obtain the following unquantized information: $\tilde{X}_1, \tilde{X}_2 \in \mathcal{M}(8)$ with

$$\tilde{X}_j^k = \tilde{A}_v Q_k \tilde{S}_j^k + \tilde{B}_v Q_k \tilde{D}_j^k.$$

The unquantized output is given by

$$\tilde{F} = (\tilde{X}_1 + \tilde{X}_2)\tilde{A}_h^T + (\tilde{X}_1 - \tilde{X}_2)\tilde{B}_h^T.$$

Let $\hat{Q}$ be the desired quantization matrix for the output and $(\hat{Q}^T)_k$ be the diagonal matrix using the entries of the k^{th} row of $\hat{Q}$.

Lemma 6.4 *The k^{th} row of the requantized output is given by ${}^k\tilde{F} = {}^k\tilde{S}_+ \tilde{A}_h^T \hat{Q}_k^{-1} + {}^k\tilde{S}_- \tilde{B}_h^T \hat{Q}_k^{-1}$, where*

$${}^k\tilde{S}_\pm = (\tilde{A}_v Q_k)(\tilde{S}_1^k \pm \tilde{S}_2^k) + (\tilde{B}_v Q_k)(\tilde{D}_1^k \pm \tilde{D}_2^k).$$

The previous Lemma implies that the dequantization and requantization can be completely incorporated in the computation: the sum-difference computation described previously for unquantized input has the same performance as the computation including de- and requantization. Furthermore, the matrices $\tilde{A}_h^T \hat{Q}_k^{-1}$, $\tilde{B}_h^T \hat{Q}_k^{-1}$, $\tilde{A}_v Q_k$, and $\tilde{B}_v Q_k$ can be computed once and used throughout the image, with the same non-zero entries as their counterparts in the sum-difference formula.

In the case of a pure vertical (or horizontal) merge operation we have

Lemma 6.5 *The k^{th} column of the requantized output is given by $\tilde{F}^k = \hat{Q}_k^{-1} \tilde{A}_v Q_k \tilde{S}^k + \hat{Q}_k^{-1} \tilde{B}_v Q_k \tilde{D}^k$.*

The matrices $\hat{Q}_k^{-1} \tilde{A}_v Q_k$ and $\hat{Q}_k^{-1} \tilde{B}_v Q_k$ can be computed once and used throughout the image, with the same zero entries as $\tilde{A}_v$ or $\tilde{B}_v$, respectively.

Using the combined operations as described above, this DCT-domain method saves about 20% of the cycles of conventional approaches, even before considerations are made for zero DCT coefficients. This saving occurs from the elimination of additional operations required for quantization and the reduction of elements in the combined matrix, as shown in 1-D in Section 4. A finessed application of our algorithm would save even more cycles through intelligent grouping of operations. Furthermore, the Fast Paths, discussed in Section 5, significantly reduce the operations required for our approach.

7. CONCLUSIONS

We have outlined a DCT-domain fast algorithm for performing shifts and merges of images along a rectangular grid. The algorithm exploits de- and re-quantization to achieve further gains, both in combining the quantization operation with the algorithm, and in reducing the number of operations required by omitting calculations on zero coefficients. The resulting algorithm can save at least 25% of the required operation cycles in an average image processing system.

8. REFERENCES

[1] C. Yim and M. A. Isnardi, "An efficient method for DCT-domain image resizing with mixed field/frame-mode macroblocks," *IEEE trans. circuits and systems for video technology*, vol. 9, no. 5, pp. 696–700, 1999.

[2] B. Shen, I. K. Sethi, and V. Bhaskaran, "DCT domain alpha blending," in *ICIP98, V.I*, (Chicago, IL), pp. 857–861, 1998.

[3] R. H. Jonsson, "Efficient DCT domain implementation of picture masking and compositing," in *ICIP97, V.II*, (Santa Barbara, CA), pp. 366–369, 1997.

[4] T. D. Tran, "The binDCT: fast multiplierless approximation of the DCT," *IEEE Signal Processing Letters*, vol. 7, no. 6, pp. 141–144, 2000.

[5] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. of the IEICE, E*, vol. 71, no. 11, p. 1095, 1988.

[6] J. Q. Trelewicz, M. T. Brady, and J. L. Mitchell, "Efficient integer implementations for faster linear transforms," in *Proceedings of 35th Asilomar Conference on Signals, Systems, and Computers 2001*, (Pacific Grove, CA), Nov. 2001.

[7] M. T. Brady, J. Q. Trelewicz, and J. L. Mitchell, "Vector processing in scalar processors for signal processing algorithms," in *ICASSP*, (Salt Lake City, UT), 2001.