Fast DCT Domain Filtering Using the DCT and the DST

Renato Kresch and Neri Merhav*
HP Israel Science Center†

Abstract

The use of the discrete sine transform (DST), together with the discrete cosine transform (DCT), for processing of compressed digital video and images, is introduced. Using this approach, a method for efficient spatial domain filtering, directly in the DCT-domain, is developed and proposed. It is demonstrated that, in typical applications, the proposed algorithm is significantly more efficient than the conventional spatial domain method, and a recently-proposed advanced method. It assumes a separable kernel, but no symmetry is required. The method is applicable to any DCT based data compression standard, such as JPEG, MPEG, and H.261.

Keywords: DCT-domain filtering, discrete sine transform, data compression.

*Currently on sabbatical leave at Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA.
†Address: HP Israel Science Center, Technion City, Haifa 32000, Israel. E-mail: [renato,merhav]@hp.technion.ac.il

1 Introduction

This work addresses the problem of efficient 2D linear filtering in the discrete Cosine transform (DCT) domain, which is an important problem in the area of processing and manipulation of images and video streams compressed in DCT-based methods, such as JPEG, MPEG, H.261, and others (see, e.g., [1-7]). More specifically, suppose that an input image is given in the format of a sequence of sets of DCT coefficients of 2D blocks. Our aim is to calculate efficiently a filtered image, in the same format, that corresponds to spatial convolution between the input image and a given filter. We derive and propose an efficient filtering algorithm that operate directly in the DCT domain without explicit transformations from the DCT domain (compressed domain) to the spatial domain (uncompressed domain) and vice versa.

There are several advantages to the direct compressed domain approach as opposed to the standard spatial domain approach:

1. It avoids the computationally expensive operations of moving to and from another transform domain.
2. The input data, given in terms of quantized DCT coefficient sets, is typically sparse, i.e., it consists mostly of null coefficients.
3. Since the DCT is closely related to the discrete Fourier transform (DFT) it has relatively simple convolution-multiplication relationships.

None of the previously reported proposed algorithms for DCT-domain filtering enjoy all three advantages at the same time. The approaches developed by Chen and Fralick [1], Ngan and Clarke [2], and Chitprasert and Rao [3] have the three properties, but perform *circular* rather than *linear* convolution, and therefore are not suitable for translation-invariant filtering. Lee and Lee [4], Chang and Messerschmitt [5], and Neri *et al.* [6] proposed algorithms for linear convolution that enjoy the first and the second properties above but not the third one. Merhav and Bhaskaran [7] significantly improved this approach by proposing an algorithm which uses sparser matrices for processing in the compressed domain. This algorithm actually represents a step forward in the direction of property 3. All the above algorithms assume a *symmetric* filter.

In this work, we fully explore the third property, at the expense of the first one. The main underlying idea is that the discrete sine transform (DST) together with the DCT provide simple convolution-multiplication relationships induced by the DFT. Since the DST coefficients are not available in advance, they have to be computed from the given DCT data. To this end, we develop fast algorithms that directly transform from the DCT to the DST (denoted CST) and vice versa (SCT). Incorporating these algorithms in the filtering scheme, we obtain an overall complexity that, for symmetric filters, is 14% smaller than that of [7], and 35-64% smaller than that of the standard spatial approach. In the non-symmetric case (for which no previous work has been reported literature), up to 64% of the computations are saved, compared to the spatial domain filtering method, depending on the kernel size.

The outline of this document is as follows. Section 2 briefly describes the theoretical background behind the derivation of the proposed algorithms. Section 3 presents the mathematical derivation, whose key element is a diagonalization theorem that provides simple convolution-multiplication properties of the DST and the DCT. In section 4, the proposed filtering, and some of its especial cases, are described. Implementation and Complexity are analyzed in section 5, and the conclusion is presented in section 6.

2 Preliminaries and Problem Description

Discrete Cosine Transform

The 8-point 1D DCT-II transforms a vector $\{x(n)\}_{n=0}^{7}$ in the spatial domain into a vector of frequency components $\{X^c(k)\}_{k=0}^{7}$, according to the following equation [8]:

$$X^c(k) = \frac{\gamma(k)}{2} \sum_{n=0}^{7} x(n) \cos(\frac{2n+1}{16} \cdot k\pi), \tag{1}$$

where $\gamma(0) = 1/\sqrt{2}$ and $\gamma(k) = 1$ for $k > 0$. The inverse transform is given by:

$$x(n) = \sum_{k=0}^{7} \frac{\gamma(k)}{2} X^c(k) \cos(\frac{2n+1}{16} \cdot k\pi). \tag{2}$$

In order to rewrite the above in a matrix form, let us define the column vectors $\boldsymbol{x} = \{x(n)\}_{n=0}^{7}$ and $\boldsymbol{X}^c = \{X^c(k)\}_{k=0}^{7}$, and define the 8-point DCT-II matrix $C = \{c(k,n)\}_{k,n=0}^{7}$, where

$$c(k,n) = \frac{\gamma(k)}{2} \cos(\frac{2n+1}{16} \cdot k\pi). \tag{3}$$

Then:

$$\boldsymbol{X}^c = C\boldsymbol{x}, \tag{4}$$

and, similarly:

$$\boldsymbol{x} = C^{-1}\boldsymbol{X}^c = C^t\boldsymbol{X}^c \tag{5}$$

where the superscript t denotes matrix transposition. The second equality follows from the unitarity of C.

Discrete Sine Transform

Similarly to the DCT, the 8-point 1-D DST-II can be represented in the following matricial terms:

$$\boldsymbol{X}^s = S\boldsymbol{x}, \tag{6}$$

where $S = \{s(k,n)\}_{k,n=1}^{8}$ is the 8-point DST-II matrix, defined by [8]:

$$s(k,n) = \frac{\sigma(k)}{2} \sin(\frac{2n-1}{16} \cdot k\pi), \tag{7}$$

with $\sigma(8) = 1/\sqrt{2}$ and $\sigma(k) = 1$ for $k < 8$.

2-D Transforms

Different 8×8 2-D transforms of a 2-D block $\boldsymbol{x} \triangleq \{x(n,m)\}_{n,m=0}^{7}$ are obtained by pre-multiplying and post-multiplying it by the 1-D DCT and DST in the following manner:

$$\boldsymbol{X}^{c} \triangleq C\boldsymbol{x}C^{t} \tag{8}$$

$$\boldsymbol{X}^{sc} \triangleq S\boldsymbol{x}C^{t} \tag{9}$$

$$\boldsymbol{X}^{cs} \triangleq C\boldsymbol{x}S^{t} \tag{10}$$

$$\boldsymbol{X}^{s} \triangleq S\boldsymbol{x}S^{t}. \tag{11}$$

The first and the last of the above transforms characterize, respectively, the 8×8 2-D Discrete Cosine and Sine Transforms. Input data, compressed by the JPEG or MPEG standards are usually available in the form of 2-D DCT blocks. We denote here the transforms (9) and (10) by Mixed DST/DCT and DCT/DST transforms, respectively.

Image Convolution

Filtering, or convolution, of an input image $\{I(i,j)\}$, (where i and j are integers taking on values in ranges that correspond to the size of the image), by a filter with impulse response $\{f(i,j)\}$ (also called kernel), results in an output image $\{J(i,j)\}$ given by:

$$J(i,j) = \sum_{i'}\sum_{j'} f(i',j')I(i-i',j-j') \tag{12}$$

where the range of summation over i' and j' is, of course, according to the support of the impulse response $\{f(i,j)\}$. In this work we assume that the filter $\{f(i,j)\}$ is *separable*, that is, $f(i,j)$ can be factorized as

$$f(i,j) = v_i h_j, \tag{13}$$

for some one-dimensional sequences $\{v_i\}$ and $\{h_j\}$. The supports of $\{v_i\}$ and $\{h_j\}$ are $M^- \leq i \leq M^+$ and $N^- \leq j \leq N^+$, respectively, meaning that $f(i,j) = 0$ outside a $(M^+ - M^- + 1) \times (N^+ - N^- + 1)$ rectangle.

Incorporating the separability assumption into eq. (12), we get

$$J(i,j) = \sum_{i'=M^-}^{M^+} v_{i'} \sum_{j'=N^-}^{N^+} h_{j'} I(i-i',j-j'), \tag{14}$$

namely, one can first perform a one-dimensional convolution on each row with the *horizontal filter component* (HFC) $\{h_j\}$, and then another one-dimensional convolution on each resulting column with the *vertical filter component* (VFC) $\{v_i\}$. Of course, the order can be interchanged and the vertical convolutions can be carried out first without affecting the final result.

An important special case, frequently assumed in previously reported work (see, e.g., [1, 3, 7]), is that of symmetric filter components, namely, $v_i = v_{-i}$ and $h_j = h_{-j}$ for all i and j.

Symmetry, however, will not be assumed in the main parts of this document, but it will be considered as a special case later.

The input image $\{I(i,j)\}$ is given in the compressed domain, that is, we are given a sequence of 8×8 matrices $\boldsymbol{X}_1^c, \boldsymbol{X}_2^c, \ldots$ of DCT coefficients corresponding to spatial domain 8×8 spatial domain blocks $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ that together form the input image $\{I(i,j)\}$. Our task is to compute the sequence of 8×8 matrices $\boldsymbol{Y}_1^c, \boldsymbol{Y}_2^c, \ldots$ of DCT coefficients of the spatial domain blocks $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$ associated with the filtered image $\{J(i,j)\}$, directly from $\boldsymbol{X}_1^c, \boldsymbol{X}_2^c, \ldots$ without going via the spatial domain and performing spatial domain convolution.

We further assume that $|M^+|$, $|M^-|$, $|N^+|$, and $|N^-|$ do not exceed 8 (that is, the filter size is always smaller than 17×17), so that every DCT block $\boldsymbol{Y}^c$ (associated with the spatial domain block $\boldsymbol{y}$) of the filtered image $\{J(i,j)\}$ depends on the corresponding DCT block $\boldsymbol{X}^c$ (associated with the spatial domain block $\boldsymbol{x}$) of the input image $\{I(i,j)\}$ and the eight immediate neighbors of $\boldsymbol{X}^c$. We shall label these neighbors according to their relative location w.r.t the current block $\boldsymbol{X}^c$, i.e., "north", "northeast", "east", etc. Accordingly, the input DCT blocks will be denoted by the appropriate subscript, i.e., $\boldsymbol{X}_N^c$, $\boldsymbol{X}_{NE}^c$, $\boldsymbol{X}_E^c$, and so on. Similarly, the respective spatial domain blocks will be denoted $\boldsymbol{x}_N$, $\boldsymbol{x}_{NE}$, $\boldsymbol{x}_E$, etc. This can be expressed in the following block matrix form:

$$\boldsymbol{y} = V \cdot \begin{pmatrix} \boldsymbol{x}_{NW} & \boldsymbol{x}_N & \boldsymbol{x}_{NE} \\ \boldsymbol{x}_W & \boldsymbol{x} & \boldsymbol{x}_E \\ \boldsymbol{x}_{SW} & \boldsymbol{x}_S & \boldsymbol{x}_{SE} \end{pmatrix} \cdot H^t \tag{15}$$

where V and H are 8×24 matrices defined by:

$$V = \begin{pmatrix} v_8 & v_7 & \cdot & \cdot & \cdot & v_1 & v_0 & v_{-1} & \cdot & \cdot & \cdot & v_{-7} & v_{-8} & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & v_8 & v_7 & \cdot & \cdot & \cdot & v_1 & v_0 & v_{-1} & \cdot & \cdot & \cdot & v_{-7} & v_{-8} & 0 & \cdot & \cdot & 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 & \cdot & \cdot & \cdot & 0 & v_8 & v_7 & \cdot & \cdot & \cdot & v_1 & v_0 & v_{-1} & \cdot & \cdot & \cdot & v_{-7} & v_{-8} \end{pmatrix} \tag{16}$$

$$H = \begin{pmatrix} h_8 & h_7 & \cdot & \cdot & \cdot & h_1 & h_0 & h_{-1} & \cdot & \cdot & \cdot & h_{-7} & h_{-8} & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & h_8 & h_7 & \cdot & \cdot & \cdot & h_1 & h_0 & h_{-1} & \cdot & \cdot & \cdot & h_{-7} & h_{-8} & 0 & \cdot & \cdot & 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 & \cdot & \cdot & \cdot & 0 & h_8 & h_7 & \cdot & \cdot & \cdot & h_1 & h_0 & h_{-1} & \cdot & \cdot & \cdot & h_{-7} & h_{-8} \end{pmatrix} \tag{17}$$

In summary, we are interested in an efficient algorithm that computes $\boldsymbol{Y}^c$ from $\boldsymbol{X}^c$, $\boldsymbol{X}_N^c$, $\boldsymbol{X}_{NE}^c$, $\boldsymbol{X}_E^c$, $\boldsymbol{X}_{SE}^c$, $\boldsymbol{X}_S^c$, $\boldsymbol{X}_{SW}^c$, $\boldsymbol{X}_W^c$, and, $\boldsymbol{X}_{NW}^c$.

3 Mathematical Derivation

3.1 Basic Derivation

The following theorem forms the mathematical foundation for the filtering scheme developed in the sequel. It provides tools for diagonalizing spatial domain convolution operator matrices (demonstrated in the sequel), yielding simple convolution-multiplication relationships.

Theorem 1 *Let* $\{g_i\}$, $i = 0, \ldots, 8$, *be an arbitrary vector of real numbers, and let* $\{G_R(k) + j \cdot G_I(k)\}$, $k = 0, \ldots, 15$, *be the 16-point DFT of its zero-padded extension* $(g_0, \ldots, g_8, 0, \ldots, 0)$.

Define also:

$$G_1 \triangleq \begin{pmatrix} g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 \\ 0 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 & g_2 \\ 0 & 0 & g_8 & g_7 & g_6 & g_5 & g_4 & g_3 \\ 0 & 0 & 0 & g_8 & g_7 & g_6 & g_5 & g_4 \\ 0 & 0 & 0 & 0 & g_8 & g_7 & g_6 & g_5 \\ 0 & 0 & 0 & 0 & 0 & g_8 & g_7 & g_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & g_8 & g_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g_8 \end{pmatrix}, \quad G_2 \triangleq \begin{pmatrix} g_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_1 & g_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 & 0 \\ g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 \\ g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 \\ g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 \\ g_7 & g_6 & g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \end{pmatrix},$$

$$\Phi \triangleq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \Theta \triangleq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{18}$$

The matrices G_1 *and* G_2 *satisfy the following relations:*

$$C\left(\frac{G_2 + G_2^t}{2} + \frac{G_1 + G_1^t}{2}\Phi\right)C^t = diag\{G_R(0), \cdots, G_R(7)\} \tag{19}$$

$$S\left(\frac{G_2 + G_2^t}{2} - \frac{G_1 + G_1^t}{2}\Phi\right)S^t = diag\{G_R(1), \cdots, G_R(8)\} \tag{20}$$

$$C\left(\frac{G_2 - G_2^t}{2} + \frac{G_1 - G_1^t}{2}\Phi\right)S^t = diag\{G_I(0), \cdots, G_I(7)\} \cdot \Theta \tag{21}$$

$$S\left(\frac{G_2 - G_2^t}{2} - \frac{G_1 - G_1^t}{2}\Phi\right)C^t = -diag\{G_I(1), \cdots, G_I(8)\} \cdot \Theta^t \tag{22}$$

The proof is given in Appendix A.

Kernel Decomposition

Our aim is to use the relations in Theorem 1 to derive a simple convolution rule in the DCT-domain, using the DCT and the DST.

Our first step is to rewrite the matrix V, defined in (16), in the form $[V_1^+, V_2^+ + V_2^-, V_1^-]$, where V_1^+, V_2^+, V_2^-, and V_1^- are defined as follows.

$$V_1^+ = \begin{pmatrix} v_8 & v_7 & v_6 & v_5 & v_4 & v_3 & v_2 & v_1 \\ 0 & v_8 & v_7 & v_6 & v_5 & v_4 & v_3 & v_2 \\ 0 & 0 & v_8 & v_7 & v_6 & v_5 & v_4 & v_3 \\ 0 & 0 & 0 & v_8 & v_7 & v_6 & v_5 & v_4 \\ 0 & 0 & 0 & 0 & v_8 & v_7 & v_6 & v_5 \\ 0 & 0 & 0 & 0 & 0 & v_8 & v_7 & v_6 \\ 0 & 0 & 0 & 0 & 0 & 0 & v_8 & v_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & v_8 \end{pmatrix} \tag{23}$$

$$V_2^+ = \begin{pmatrix} \alpha \cdot v_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_1 & \alpha \cdot v_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_2 & v_1 & \alpha \cdot v_0 & 0 & 0 & 0 & 0 & 0 \\ v_3 & v_2 & v_1 & \alpha \cdot v_0 & 0 & 0 & 0 & 0 \\ v_4 & v_3 & v_2 & v_1 & \alpha \cdot v_0 & 0 & 0 & 0 \\ v_5 & v_4 & v_3 & v_2 & v_1 & \alpha \cdot v_0 & 0 & 0 \\ v_6 & v_5 & v_4 & v_3 & v_2 & v_1 & \alpha \cdot v_0 & 0 \\ v_7 & v_6 & v_5 & v_4 & v_3 & v_2 & v_1 & \alpha \cdot v_0 \end{pmatrix} \tag{24}$$

$$V_2^- = \begin{pmatrix} \beta \cdot v_0 & v_{-1} & v_{-2} & v_{-3} & v_{-4} & v_{-5} & v_{-6} & v_{-7} \\ 0 & \beta \cdot v_0 & v_{-1} & v_{-2} & v_{-3} & v_{-4} & v_{-5} & v_{-6} \\ 0 & 0 & \beta \cdot v_0 & v_{-1} & v_{-2} & v_{-3} & v_{-4} & v_{-5} \\ 0 & 0 & 0 & \beta \cdot v_0 & v_{-1} & v_{-2} & v_{-3} & v_{-4} \\ 0 & 0 & 0 & 0 & \beta \cdot v_0 & v_{-1} & v_{-2} & v_{-3} \\ 0 & 0 & 0 & 0 & 0 & \beta \cdot v_0 & v_{-1} & v_{-2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \beta \cdot v_0 & v_{-1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta \cdot v_0 \end{pmatrix} \tag{25}$$

$$V_1^- = \begin{pmatrix} v_{-8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_{-7} & v_{-8} & 0 & 0 & 0 & 0 & 0 & 0 \\ v_{-6} & v_{-7} & v_{-8} & 0 & 0 & 0 & 0 & 0 \\ v_{-5} & v_{-6} & v_{-7} & v_{-8} & 0 & 0 & 0 & 0 \\ v_{-4} & v_{-5} & v_{-6} & v_{-7} & v_8 & 0 & 0 & 0 \\ v_{-3} & v_{-4} & v_{-5} & v_{-6} & v_{-7} & v_{-8} & 0 & 0 \\ v_{-2} & v_{-3} & v_{-4} & v_{-5} & v_{-6} & v_{-7} & v_{-8} & 0 \\ v_{-1} & v_{-2} & v_{-3} & v_{-4} & v_{-5} & v_{-6} & v_{-7} & v_{-8} \end{pmatrix} \tag{26}$$

where α is an arbitrary real number, and $\beta \triangleq 1-\alpha$. In the general case considered herein the choice of α is immaterial from the aspects of computational efficiency. In certain special cases that will be studied later, however, the choice of α will be important.

Note that, if $\{g_i\}_{i=0}^{8}$ is given by

$$g_i = \begin{cases} \alpha \cdot v_0, & i=0 \\ v_i, & i=1,\ldots,8 \end{cases}, \tag{27}$$

then $G_1 = V_1^+$ and $G_2 = V_2^+$, and similarly, if

$$g_i = \begin{cases} \beta \cdot v_0, & i=0 \\ v_{-i}, & i=1,\ldots,8 \end{cases}, \tag{28}$$

then $G_1 = (V_1^-)^t$ and $G_2 = (V_2^-)^t$.

Let

$$V_F^+ \triangleq \mathrm{DFT}\{\alpha \cdot v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, 0,0,0,0,0,0,0\} \tag{29}$$
$$V_F^- \triangleq \mathrm{DFT}\{\beta \cdot v_0, v_{-1}, v_{-2}, v_{-3}, v_{-4}, v_{-5}, v_{-6}, v_{-7}, v_{-8}, 0,0,0,0,0,0,0\} \tag{30}$$

and let $V_R^+ \triangleq \Re\mathrm{e}\{V_F^+\}$, $V_I^+ \triangleq \Im\mathrm{m}\{V_F^+\}$, $V_R^- \triangleq \Re\mathrm{e}\{V_F^-\}$, $V_I^- \triangleq \Im\mathrm{m}\{V_F^-\}$,

We define:

$$V_{R,0}^+ \triangleq \frac{1}{2}\left[\frac{V_2^+ + (V_2^+)^t}{2} + \frac{V_1^+ + (V_1^+)^t}{2}\Phi\right] \tag{31}$$
$$V_{R,1}^+ \triangleq \frac{1}{2}\left[\frac{V_2^+ + (V_2^+)^t}{2} - \frac{V_1^+ + (V_1^+)^t}{2}\Phi\right] \tag{32}$$
$$V_{I,0}^+ \triangleq \frac{1}{2}\left[\frac{V_2^+ - (V_2^+)^t}{2} + \frac{V_1^+ - (V_1^+)^t}{2}\Phi\right] \tag{33}$$
$$V_{I,1}^+ \triangleq \frac{1}{2}\left[\frac{V_2^+ - (V_2^+)^t}{2} - \frac{V_1^+ - (V_1^+)^t}{2}\Phi\right] \tag{34}$$
$$\boldsymbol{V}_{R,0}^+ \triangleq \frac{1}{2}\cdot \mathrm{diag}\{V_R^+(0),\cdots,V_R^+(7)\} \tag{35}$$
$$\boldsymbol{V}_{R,1}^+ \triangleq \frac{1}{2}\cdot \mathrm{diag}\{V_R^+(1),\cdots,V_R^+(8)\} \tag{36}$$
$$\boldsymbol{V}_{I,0}^+ \triangleq \frac{1}{2}\cdot \mathrm{diag}\{V_I^+(0),\cdots,V_I^+(7)\}\cdot\Theta \tag{37}$$
$$\boldsymbol{V}_{I,1}^+ \triangleq \frac{1}{2}\cdot \mathrm{diag}\{V_I^+(1),\cdots,V_I^+(8)\}\cdot\Theta^t. \tag{38}$$

and define similar quantities with all "+" superscripts replaced by "−."

Now, Theorem 1 gives

$$CV^+_{R,0}C^t = \boldsymbol{V}^+_{R,0}; \qquad CV^-_{R,0}C^t = \boldsymbol{V}^-_{R,0} \tag{39}$$
$$SV^+_{R,1}S^t = \boldsymbol{V}^+_{R,1}; \qquad SV^-_{R,1}S^t = \boldsymbol{V}^-_{R,1} \tag{40}$$
$$CV^+_{I,0}S^t = \boldsymbol{V}^+_{I,0}; \qquad CV^-_{I,0}S^t = -\boldsymbol{V}^-_{I,0} \tag{41}$$
$$SV^+_{I,1}C^t = -\boldsymbol{V}^+_{I,1}; \qquad SV^-_{I,1}C^t = \boldsymbol{V}^-_{I,1} \tag{42}$$

In words, the above equations tell us that combinations of row/column DCT and DST operations diagonalize the matrices $V^r_{p,q}$, $p \in \{R, I\}$, $q \in \{0, 1\}$, and $r \in \{+, -\}$. The elements on the main diagonals are directly related to the DFT of the causal and anti-causal parts of the filter. Observe also, that since the elements of V are real, $V_I^+(0) = V_I^+(8) = V_I^-(0) = V_I^-(8) = 0$.

In the same manner, we have for the HFC, H

$$CH^+_{R,0}C^t = \boldsymbol{H}^+_{R,0}; \qquad CH^-_{R,0}C^t = \boldsymbol{H}^-_{R,0} \tag{43}$$
$$SH^+_{R,1}S^t = \boldsymbol{H}^+_{R,1}; \qquad SH^-_{R,1}S^t = \boldsymbol{H}^-_{R,1} \tag{44}$$
$$CH^+_{I,0}S^t = \boldsymbol{H}^+_{I,0}; \qquad CH^-_{I,0}S^t = -\boldsymbol{H}^-_{I,0} \tag{45}$$
$$SH^+_{I,1}C^t = -\boldsymbol{H}^+_{I,1}; \qquad SH^-_{I,1}C^t = \boldsymbol{H}^-_{I,1} \tag{46}$$

with suitable definitions of these matrices.

Input Butterflies

Eq. (15) can be directly rewritten in terms of V^r_i and H^r_i, $i = 1, 2$, $r \in \{+, -\}$, as follows:

$$\begin{aligned}
\boldsymbol{y} &= (V_1^+\boldsymbol{x}_{NW} + V_2^+\boldsymbol{x}_W + V_2^-\boldsymbol{x}_W + V_1^-\boldsymbol{x}_{SW})(H_1^+)^t + \\
&\quad (V_1^+\boldsymbol{x}_N + V_2^+\boldsymbol{x} + V_2^-\boldsymbol{x} + V_1^-\boldsymbol{x}_S)(H_2^+)^t + \\
&\quad (V_1^+\boldsymbol{x}_N + V_2^+\boldsymbol{x} + V_2^-\boldsymbol{x} + V_1^-\boldsymbol{x}_S)(H_2^-)^t + \\
&\quad (V_1^+\boldsymbol{x}_{NE} + V_2^+\boldsymbol{x}_E + V_2^-\boldsymbol{x}_E + V_1^-\boldsymbol{x}_{SE})(H_1^-)^t, \qquad (47)\\
&= \boldsymbol{z}_W(H_1^+)^t + \boldsymbol{z}(H_2^+)^t + \boldsymbol{z}(H_2^-)^t + \boldsymbol{z}_E(H_1^-)^t, \qquad (48)
\end{aligned}$$

where

$$\boldsymbol{z}_W \triangleq V_1^+\boldsymbol{x}_{NW} + V_2^+\boldsymbol{x}_W + V_2^+\boldsymbol{x}_W + V_1^+\boldsymbol{x}_{SW}, \tag{49}$$
$$\boldsymbol{z} \triangleq V_1^+\boldsymbol{x}_N + V_2^+\boldsymbol{x} + V_2^+\boldsymbol{x} + V_1^+\boldsymbol{x}_S, \tag{50}$$
$$\boldsymbol{z}_E \triangleq V_1^+\boldsymbol{x}_{NE} + V_2^+\boldsymbol{x}_E + V_2^+\boldsymbol{x}_E + V_1^+\boldsymbol{x}_{SE}. \tag{51}$$

We next generate "butterflies" at the input level in order to express $\boldsymbol{y}$ in terms of $V^r_{p,q}$ and $H^r_{p,q}$, $p \in \{R, I\}$, $q \in \{0, 1\}$, and $r \in \{+, -\}$, which are diagonalizable by C and S. This will enable us to take advantage of properties (39)-(42) and (43)-(46). The basic idea is to use the identity $ax + by = \frac{1}{2}(a + b)(x + y) + \frac{1}{2}(a - b)(x - y)$.

For instance, $\boldsymbol{z}_E$ is given by

$$\begin{aligned}
\boldsymbol{z}_E &= V_{R,0}^{+}(\boldsymbol{x}_E + \Phi \boldsymbol{x}_{NE}) + V_{R,1}^{+}(\boldsymbol{x}_E - \Phi \boldsymbol{x}_{NE}) + \\
&\quad V_{I,0}^{+}(\boldsymbol{x}_E + \Phi \boldsymbol{x}_{NE}) + V_{I,1}^{+}(\boldsymbol{x}_E - \Phi \boldsymbol{x}_{NE}) + \\
&\quad V_{R,0}^{-}(\boldsymbol{x}_E + \Phi \boldsymbol{x}_{SE}) + V_{R,1}^{-}(\boldsymbol{x}_E - \Phi \boldsymbol{x}_{SE}) + \\
&\quad V_{I,0}^{-}(\boldsymbol{x}_E + \Phi \boldsymbol{x}_{SE}) + V_{I,1}^{-}(\boldsymbol{x}_E - \Phi \boldsymbol{x}_{SE}).
\end{aligned} \tag{52}$$

Similar expressions are obtained for $\boldsymbol{z}$ and $\boldsymbol{z}_W$ by using (52) and replacing $\boldsymbol{x}_{NE}, \boldsymbol{x}_E$, and $\boldsymbol{x}_{SE}$ by $\boldsymbol{x}_N, \boldsymbol{x}, \boldsymbol{x}_S$, and $\boldsymbol{x}_{NW}, \boldsymbol{x}_W, \boldsymbol{x}_{SW}$, respectively.

Similarly, we obtain:

$$\begin{aligned}
\boldsymbol{y} &= (\boldsymbol{z} + \boldsymbol{z}_W \Phi)(H_{R,0}^{+})^t + (\boldsymbol{z} - \boldsymbol{z}_W \Phi)(H_{R,1}^{+})^t + \\
&\quad (\boldsymbol{z} + \boldsymbol{z}_W \Phi)(H_{I,0}^{+})^t + (\boldsymbol{z} - \boldsymbol{z}_W \Phi)(H_{I,1}^{+})^t + \\
&\quad (\boldsymbol{z} + \boldsymbol{z}_E \Phi)(H_{R,0}^{-})^t + (\boldsymbol{z} - \boldsymbol{z}_E \Phi)(H_{R,1}^{-})^t + \\
&\quad (\boldsymbol{z} + \boldsymbol{z}_E \Phi)(H_{I,0}^{-})^t + (\boldsymbol{z} - \boldsymbol{z}_E \Phi)(H_{I,1}^{-})^t.
\end{aligned} \tag{53}$$

The DCT-Domain Equations

So far, our derivations referred to the spatial domain solely. In order to express the DCT of the output block in terms of the DCT sets of the input data, we use the distributive property of the DCT w.r.t matrix multiplication. To this end, we first pre-multiply and post-multiply both sides of (52) and of (53) by C and C^t, respectively. Secondly, in order to use (39)-(42) and (43)-(46), we strategically insert the terms $C^tC = I$ or $S^tS = I$, I being the 8×8 identity matrix, between every two multiplied matrices in eq. (52) and (53). This results in

$$\begin{aligned}
\boldsymbol{Z}_E^c &= \boldsymbol{V}_{R,0}^{+}(\boldsymbol{X}_E^c + \boldsymbol{\Phi} \boldsymbol{X}_{NE}^c) + \boldsymbol{V}_{I,0}^{+}(\boldsymbol{X}_E^{sc} + \boldsymbol{\Phi} \boldsymbol{X}_{NE}^{sc}) + \\
&\quad \boldsymbol{V}_{R,0}^{-}(\boldsymbol{X}_E^c + \boldsymbol{\Phi} \boldsymbol{X}_{SE}^c) - \boldsymbol{V}_{I,0}^{-}(\boldsymbol{X}_E^{sc} + \boldsymbol{\Phi} \boldsymbol{X}_{SE}^{sc}) + \\
&\quad T^t \Big[\boldsymbol{V}_{R,1}^{+}(\boldsymbol{X}_E^{sc} - \boldsymbol{\Phi} \boldsymbol{X}_{NE}^{sc}) - \boldsymbol{V}_{I,1}^{+}(\boldsymbol{X}_E^c - \boldsymbol{\Phi} \boldsymbol{X}_{NE}^c) + \\
&\quad \boldsymbol{V}_{R,1}^{-}(\boldsymbol{X}_E^{sc} - \boldsymbol{\Phi} \boldsymbol{X}_{SE}^{sc}) + \boldsymbol{V}_{I,1}^{-}(\boldsymbol{X}_E^c - \boldsymbol{\Phi} \boldsymbol{X}_{SE}^c) \Big]
\end{aligned} \tag{54}$$

where $T \triangleq SC^t$ is interpreted as the 1-D DCT-to-DST domain transform (CST) operator matrix (hence T^t is the SCT operator matrix), and

$$\boldsymbol{\Phi} \triangleq C \Phi C^t = S \Phi S^t = \mathrm{diag}\left\{ (-1)^k \right\}_{k=0}^{7}. \tag{55}$$

Equation (54) yields an efficient filtering scheme provided that the SCT can be implemented efficiently. This is true because all the $\boldsymbol{V}$-matrices in (54) are diagonal. Note, that multiplication by $\boldsymbol{\Phi}$ is costless.

Nevertheless, unlike $\boldsymbol{X}_i^c$, the Mixed DST/DCT coefficients $\boldsymbol{X}_i^{sc} = T\boldsymbol{X}_i^c$ are not available in advance and therefore a fast CST is required for obtaining these matrices. In section 3.2, fast

CST and SCT algorithms are developed. The overall computational complexity is assessed in section 5.

Equivalent expressions for $\boldsymbol{Z}^c$ and $\boldsymbol{Z}^c_W$ are derived from (54) simply by replacing $\boldsymbol{X}^c_{NE}, \boldsymbol{X}^c_E, \boldsymbol{X}^c_{SE}$ by $\boldsymbol{X}^c_N, \boldsymbol{X}^c, \boldsymbol{X}^c_S$, and $\boldsymbol{X}^c_{NW}, \boldsymbol{X}^c_W, \boldsymbol{X}^c_{SW}$, respectively. Assuming that the blockwise filtering procedure is performed in a raster scan order (i.e., left-to-right and top-to-bottom), then it is easy to note that $\boldsymbol{Z}^c$ is identical to $\boldsymbol{Z}^c_E$ of the previously processed DCT block, and similarly, $\boldsymbol{Z}^c_W$ is the same as $\boldsymbol{Z}^c_E$ two steps ago. Thus, one needs only to calculate $\boldsymbol{Z}^c_E$ in every step and to save it for the next two steps.

Finally, transforming eq. (53) to the DCT domain yields

$$\begin{aligned}\boldsymbol{Y}^c \;=\; & (\boldsymbol{Z}^c + \boldsymbol{Z}^c_W\boldsymbol{\Phi})\boldsymbol{H}^+_{R,0} + (\boldsymbol{Z}^{cs} + \boldsymbol{Z}^{cs}_W\boldsymbol{\Phi})(\boldsymbol{H}^+_{I,0})^t + \\ & (\boldsymbol{Z}^c + \boldsymbol{Z}^c_E\boldsymbol{\Phi})\boldsymbol{H}^-_{R,0} - (\boldsymbol{Z}^{cs} + \boldsymbol{Z}^{cs}_E\boldsymbol{\Phi})(\boldsymbol{H}^-_{I,0})^t + \\ & \Big[(\boldsymbol{Z}^{cs} - \boldsymbol{Z}^{cs}_W\boldsymbol{\Phi})\boldsymbol{H}^+_{R,1} - (\boldsymbol{Z}^c - \boldsymbol{Z}^c_W\boldsymbol{\Phi})(\boldsymbol{H}^+_{I,1})^t + \\ & (\boldsymbol{Z}^{cs} - \boldsymbol{Z}^{cs}_E\boldsymbol{\Phi})\boldsymbol{H}^-_{R,1} + (\boldsymbol{Z}^c - \boldsymbol{Z}^c_E\boldsymbol{\Phi})(\boldsymbol{H}^-_{I,1})^t\Big] T. \end{aligned} \tag{56}$$

Here, the Mixed DCT/DST coefficients $\boldsymbol{Z}^{cs}_E$ must be derived from $\boldsymbol{Z}^c_E$ by using $\boldsymbol{Z}^{cs}_E = \boldsymbol{Z}^c_E T^t$, and it must be stored to be used as $\boldsymbol{Z}^{cs}$ and $\boldsymbol{Z}^{cs}_W$ in the following 2 steps of the scanning process.

3.2 The CST and SCT Algorithms

In this section we derive efficient CST and SCT algorithms, i.e., fast multiplication by T and T^t. The main idea is to factorize these matrices into products of sparse matrices.

First, we shall use the following property relating the DST matrix to the DCT matrix:

$$S = \Phi C \boldsymbol{\Phi} \tag{57}$$

where Φ and $\boldsymbol{\Phi}$ are defined in (18) and (55), respectively.

In addition, we shall use a factorization of C that corresponds to the fastest existing algorithm for 8-point DCT due to Arai, Agui, and Nakajima [9] (see also [10]). According to this factorization, C is represented as follows.

$$C = DPB_1B_2MA_1A_2A_3 \tag{58}$$

where D is a diagonal matrix given by

$$D = \mathrm{diag}\{0.3536, 0.2549, 0.2706, 0.3007, 0.3536, 0.4500, 0.6533, 1.2814\}, \tag{59}$$

P is a permutation matrix given by

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

and the remaining matrices are defined as follows:

$$B_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7071 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.9239 & 0 & -0.3827 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.7071 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.3827 & 0 & 0.9239 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Thus, we have

$$T = SC^t = \Phi C \boldsymbol{\Phi} C^t = \Phi DPB_1B_2MA_1A_2A_3\boldsymbol{\Phi}A_3^tA_2^tA_1^tM^tB_2^tB_1^tP^tD^t \tag{60}$$

The proposed CST algorithm is based on the observation that the product

$$G \triangleq MA_1A_2A_3\boldsymbol{\Phi}A_3^tA_2^tA_1^tM^t \tag{61}$$

is a fairly sparse matrix, given by:

$$G = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & -5.2264 & 0 & 2.1648 & 2 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 1.4142 \\ 0 & 0 & 0 & 0 & 1.0824 & 0 & 2.6132 & 2 \\ 0 & -5.2264 & 0 & 1.0824 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.1648 & 0 & 2.6132 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1.4142 & 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Using part of the inter-relations between the elements of G, one can implement the multiplication of a vector by G with 8 multiplications and 12 additions.

The final implementation is based on:

$$T = \tilde{D}\tilde{T}D, \tag{62}$$

where

$$\tilde{T} \triangleq \Phi PB_1B_2GB_2^tB_1^tP^t. \tag{63}$$

and where $\tilde{D} \triangleq \Phi D \Phi$ is a diagonal matrix having the same elements as D but in reversed order.

The multiplication by $\tilde{T}$ takes 8 multiplications and 28 additions.

In the next section, we show how to avoid the multiplications by the matrices $\tilde{D}$ and D, by absorbing them in the dequantization and quantization tables of the JPEG (or MPEG) decompression and compression processes, respectively, and in the kernel matrices $\boldsymbol{V}_{p,q}^r$, $p \in \{R, I\}$, $q \in \{0, 1\}$, $r \in \{+, -\}$.

The column-wise SCT, corresponding to T^t, is obtained similarly as above.

$$T^t = D\tilde{T}^t\tilde{D} = D(\Phi\tilde{T}\Phi)\tilde{D}, \tag{64}$$

and, therefore, can be implemented with the same number of operations as T.

4 The Proposed Filtering Scheme

We now present the proposed filtering scheme. It consists of a modification of the basic scheme developed in section 3.1, by absorbing the matrices $\tilde{D}$ and D in the dequantization and quantization tables of the JPEG (or MPEG) algorithm, and the kernel matrices, in order to avoid their actual multiplication.

The scheme modification is based on pre-multiplying and post-multiplying both sides of eq. (56) by D^{-1}, and both sides of eq. (54) by D^{-1} and D, respectively. We also strategically insert the terms DD^{-1} or $\tilde{D}^{-1}\tilde{D}$ in the above equations. The resulting filtering scheme is the following.

1. Let Q^d be the dequantization table used in the decoding algorithm. Q^d must be altered in the following way:
$$Q^d \leftarrow D \cdot Q^d \cdot D \tag{65}$$
Therefore, instead of the data blocks $\boldsymbol{X}_r^c$, $r \in \{NW, W, SW, N, \emptyset, S, NE, E, SE\}$, we now manipulate the blocks $\tilde{\boldsymbol{X}}_r^c = D\boldsymbol{X}_r^c D$.

2. The modified DST-block $\tilde{\boldsymbol{X}}_{SE}^{sc}$ is calculated by:
$$\tilde{\boldsymbol{X}}_{SE}^{sc} = \tilde{T}\tilde{\boldsymbol{X}}_{SE}^{c}, \tag{66}$$
and stored to be used as $\tilde{\boldsymbol{X}}_p^{sc}$, $p \in \{NW, W, SW, N, \emptyset, S, NE, E\}$, in future steps of the scanning procedure.

3. Calculate:
$$\begin{aligned}\tilde{\boldsymbol{Z}}_E^c \;=\; & \tilde{\boldsymbol{V}}_{R,0}^{+}(\tilde{\boldsymbol{X}}_E^c + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{NE}^c) + \tilde{\boldsymbol{V}}_{I,0}^{+}(\tilde{\boldsymbol{X}}_E^{sc} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{NE}^{sc}) + \\ & \tilde{\boldsymbol{V}}_{R,0}^{-}(\tilde{\boldsymbol{X}}_E^c + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{SE}^c) - \tilde{\boldsymbol{V}}_{I,0}^{-}(\tilde{\boldsymbol{X}}_E^{sc} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{SE}^{sc}) + \\ & \tilde{T}^t\Big[\tilde{\boldsymbol{V}}_{R,1}^{+}(\tilde{\boldsymbol{X}}_E^{sc} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{NE}^{sc}) - \tilde{\boldsymbol{V}}_{I,1}^{+}(\tilde{\boldsymbol{X}}_E^c - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{NE}^c) + \\ & \tilde{\boldsymbol{V}}_{R,1}^{-}(\tilde{\boldsymbol{X}}_E^{sc} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{SE}^{sc}) + \tilde{\boldsymbol{V}}_{I,1}^{-}(\tilde{\boldsymbol{X}}_E^c - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}_{SE}^c)\Big],\end{aligned} \tag{67}$$
where:
$$\tilde{\boldsymbol{V}}_{R,0}^{+} \triangleq D^{-1}\boldsymbol{V}_{R,0}^{+}D^{-1}; \qquad \tilde{\boldsymbol{V}}_{I,0}^{+} \triangleq D^{-1}\boldsymbol{V}_{I,0}^{+}\tilde{D} \tag{68}$$
$$\tilde{\boldsymbol{V}}_{R,0}^{-} \triangleq D^{-1}\boldsymbol{V}_{R,0}^{-}D^{-1}; \qquad \tilde{\boldsymbol{V}}_{I,0}^{-} \triangleq D^{-1}\boldsymbol{V}_{I,0}^{-}\tilde{D} \tag{69}$$
$$\tilde{\boldsymbol{V}}_{R,1}^{+} \triangleq \tilde{D}\boldsymbol{V}_{R,1}^{+}\tilde{D}; \qquad \tilde{\boldsymbol{V}}_{I,1}^{+} \triangleq \tilde{D}\boldsymbol{V}_{I,1}^{+}D^{-1} \tag{70}$$
$$\tilde{\boldsymbol{V}}_{R,1}^{-} \triangleq \tilde{D}\boldsymbol{V}_{R,1}^{-}\tilde{D}; \qquad \tilde{\boldsymbol{V}}_{I,1}^{-} \triangleq \tilde{D}\boldsymbol{V}_{I,1}^{-}D^{-1}, \tag{71}$$
$\tilde{\boldsymbol{Z}}_E^c$ is stored to be used in future steps as $\tilde{\boldsymbol{Z}}^c$ and $\tilde{\boldsymbol{Z}}_W^c$.

4. The modified DST-block $\tilde{\boldsymbol{Z}}_E^{cs}$ is calculated by:

$$\tilde{\boldsymbol{Z}}_E^{cs} = \tilde{\boldsymbol{Z}}_E^{c} \tilde{T}^t, \tag{72}$$

and used in future steps as $\tilde{\boldsymbol{Z}}^{cs}$ and $\tilde{\boldsymbol{Z}}_W^{cs}$.

5. Calculate:

$$\begin{aligned} \tilde{\boldsymbol{Y}}^c \;=\; & (\tilde{\boldsymbol{Z}}^c + \tilde{\boldsymbol{Z}}_W^c \boldsymbol{\Phi})\tilde{\boldsymbol{H}}_{R,0}^{+} + (\tilde{\boldsymbol{Z}}^{cs} + \tilde{\boldsymbol{Z}}_W^{cs} \boldsymbol{\Phi})(\tilde{\boldsymbol{H}}_{I,0}^{+})^t + \\ & (\tilde{\boldsymbol{Z}}^c + \tilde{\boldsymbol{Z}}_E^c \boldsymbol{\Phi})\tilde{\boldsymbol{H}}_{R,0}^{-} - (\tilde{\boldsymbol{Z}}^{cs} + \tilde{\boldsymbol{Z}}_E^{cs} \boldsymbol{\Phi})(\tilde{\boldsymbol{H}}_{I,0}^{-})^t + \\ & \Big[(\tilde{\boldsymbol{Z}}^{cs} - \tilde{\boldsymbol{Z}}_W^{cs} \boldsymbol{\Phi})\tilde{\boldsymbol{H}}_{R,1}^{+} - (\tilde{\boldsymbol{Z}}^c - \tilde{\boldsymbol{Z}}_W^c \boldsymbol{\Phi})(\tilde{\boldsymbol{H}}_{I,1}^{+})^t + \\ & (\tilde{\boldsymbol{Z}}^{cs} - \tilde{\boldsymbol{Z}}_E^{cs} \boldsymbol{\Phi})\tilde{\boldsymbol{H}}_{R,1}^{-} + (\tilde{\boldsymbol{Z}}^c - \tilde{\boldsymbol{Z}}_E^c \boldsymbol{\Phi})(\tilde{\boldsymbol{H}}_{I,1}^{-})^t\Big] \tilde{T} \end{aligned} \tag{73}$$

where:

$$\tilde{\boldsymbol{H}}_{R,0}^{+} \triangleq D^{-1}\boldsymbol{H}_{R,0}^{+}D^{-1}; \qquad \tilde{\boldsymbol{H}}_{I,0}^{+} \triangleq \tilde{D}\boldsymbol{H}_{I,0}^{+}D^{-1} \tag{74}$$

$$\tilde{\boldsymbol{H}}_{R,0}^{-} \triangleq D^{-1}\boldsymbol{H}_{R,0}^{-}D^{-1}; \qquad \tilde{\boldsymbol{H}}_{I,0}^{-} \triangleq \tilde{D}\boldsymbol{H}_{I,0}^{-}D^{-1} \tag{75}$$

$$\tilde{\boldsymbol{H}}_{R,1}^{+} \triangleq \tilde{D}\boldsymbol{H}_{R,1}^{+}\tilde{D}; \qquad \tilde{\boldsymbol{H}}_{I,1}^{+} \triangleq D^{-1}\boldsymbol{H}_{I,1}^{+}\tilde{D} \tag{76}$$

$$\tilde{\boldsymbol{H}}_{R,1}^{-} \triangleq \tilde{D}\boldsymbol{H}_{R,1}^{-}\tilde{D}; \qquad \tilde{\boldsymbol{H}}_{I,1}^{-} \triangleq D^{-1}\boldsymbol{H}_{I,1}^{-}\tilde{D}. \tag{77}$$

6. The resulting block $\tilde{\boldsymbol{Y}}^c$ is related to the desired output $\boldsymbol{Y}^c$ by: $\boldsymbol{Y}^c = D\tilde{\boldsymbol{Y}}^c D$. Therefore, to obtain the desired quantized block, the quantization table, denoted Q^q, must be altered in the same way as Q^d before it, i.e.:

$$Q^q \leftarrow D \cdot Q^q \cdot D. \tag{78}$$

In summary, if one alters the quantization and dequantization tables, and the kernel ($\boldsymbol{V}$- and $\boldsymbol{H}$-type) matrices as described above, the CST and the SCT can be implemented according to $\tilde{T}$ and $\tilde{T}^t$, respectively, i.e., without explicit multiplications by D and $\tilde{D}$.

4.1 Symmetric and Anti-Symmetric Filters

An important special case is the symmetric case, i.e., $h_{-n} = h_n$ and $v_{-n} = v_n$, $n = 1, \ldots, 8$. Here, by setting $\alpha = \beta = 1/2$ in (29) and (30), we obtain $\tilde{\boldsymbol{V}}_{p,q}^{+} = \tilde{\boldsymbol{V}}_{p,q}^{-}$ and $\tilde{\boldsymbol{H}}_{p,q}^{+} = \tilde{\boldsymbol{H}}_{p,q}^{-}$, for $p \in \{R, I\}$ and $q \in \{0, 1\}$.

Therefore, in this case, by defining $\tilde{\boldsymbol{V}}_{p,q} \triangleq \tilde{\boldsymbol{V}}_{p,q}^{+}$ and $\tilde{\boldsymbol{H}}_{p,q} \triangleq \tilde{\boldsymbol{H}}_{p,q}^{+}$, the schemes presented in (73) and (67) become, respectively:

$$\begin{aligned} \tilde{\boldsymbol{Y}}^c \;=\; & [2\tilde{\boldsymbol{Z}}^c + (\tilde{\boldsymbol{Z}}_W^c + \tilde{\boldsymbol{Z}}_E^c)\boldsymbol{\Phi}]\tilde{\boldsymbol{H}}_{R,0} + (\tilde{\boldsymbol{Z}}_W^{cs} - \tilde{\boldsymbol{Z}}_E^{cs})\boldsymbol{\Phi}(\tilde{\boldsymbol{H}}_{I,0})^t + \\ & \Big\{[2\tilde{\boldsymbol{Z}}^{cs} - (\tilde{\boldsymbol{Z}}_W^{cs} + \tilde{\boldsymbol{Z}}_E^{cs})\boldsymbol{\Phi}]\tilde{\boldsymbol{H}}_{R,1} + (\tilde{\boldsymbol{Z}}_W^c - \tilde{\boldsymbol{Z}}_E^c)\boldsymbol{\Phi}(\tilde{\boldsymbol{H}}_{I,1})^t\Big\} \tilde{T} \end{aligned} \tag{79}$$

$$\begin{aligned}\tilde{\boldsymbol{Z}}^c_E &= \tilde{\boldsymbol{V}}_{R,0}[2\tilde{\boldsymbol{X}}_E^{\ c} + \boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^c_{NE} + \tilde{\boldsymbol{X}}^c_{SE})] + \tilde{\boldsymbol{V}}_{I,0}\boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^{sc}_{NE} - \tilde{\boldsymbol{X}}^{sc}_{SE}) + \\ & \tilde{T}^t\left\{\tilde{\boldsymbol{V}}_{R,1}[2\tilde{\boldsymbol{X}}^{sc}_E - \boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^{sc}_{NE} + \tilde{\boldsymbol{X}}^{sc}_{SE})] + \tilde{\boldsymbol{V}}_{I,1}\boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^c_{NE} - \tilde{\boldsymbol{X}}^c_{SE})\right\}\end{aligned} \tag{80}$$

In the *anti-symmetric* case, i.e., $h_{-n} = -h_n$ and $v_{-n} = -v_n$, $n = 1, \ldots, 8$, $h_0 = v_0 = 0$, a similar scheme is obtained, with some sign changes.

4.2 Causal and Anti-Causal Filters

In the causal case, defined by $h_n = v_n = 0$ for $n < 0$, setting $\alpha = 1 - \beta = 1$, gives

$$\boldsymbol{V}^-_{R,0} = \boldsymbol{V}^-_{R,1} = \boldsymbol{V}^-_{I,0} = \boldsymbol{V}^-_{I,1} = \boldsymbol{H}^-_{R,0} = \boldsymbol{H}^-_{R,1} = \boldsymbol{H}^-_{I,0} = \boldsymbol{H}^-_{I,1} = \boldsymbol{0}, \tag{81}$$

where $\boldsymbol{0}$ is the 8×8 null matrix. Incorporating this fact into the filtering equation (73), one obtains

$$\begin{aligned}\tilde{\boldsymbol{Y}}^c &= (\tilde{\boldsymbol{Z}}^c + \tilde{\boldsymbol{Z}}^c_W\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^+_{R,0} + (\tilde{\boldsymbol{Z}}^{cs} + \tilde{\boldsymbol{Z}}^{cs}_W\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^+_{I,0})^t + \\ & \left[(\tilde{\boldsymbol{Z}}^{cs} - \tilde{\boldsymbol{Z}}^{cs}_W\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^+_{R,1} - (\tilde{\boldsymbol{Z}}^c - \tilde{\boldsymbol{Z}}^c_W\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^+_{I,1})^t\right]\tilde{T}\end{aligned} \tag{82}$$

Notice that, in this case, $\tilde{\boldsymbol{Z}}^c_E$ does not need to be calculated, because only $\tilde{\boldsymbol{Z}}^c$ and $\tilde{\boldsymbol{Z}}^c_W$ are used in (82). Therefore, as opposed to the previous cases, where $\tilde{\boldsymbol{Z}}^c_E$ is calculated and stored for 2 steps, here we have to calculate $\tilde{\boldsymbol{Z}}^c$ and store it for the next one step to be used as $\tilde{\boldsymbol{Z}}^c_W$. The following formula for calculating $\tilde{\boldsymbol{Z}}^c$ is derived from (67) by replacing $\tilde{\boldsymbol{X}}^c_{NE}, \tilde{\boldsymbol{X}}^c_E, \tilde{\boldsymbol{X}}^c_{SE}$ by $\tilde{\boldsymbol{X}}^c_N, \tilde{\boldsymbol{X}}^c, \tilde{\boldsymbol{X}}^c_S$, and incorporating (81) in it.

$$\begin{aligned}\tilde{\boldsymbol{Z}}^c &= \tilde{\boldsymbol{V}}^+_{R,0}(\tilde{\boldsymbol{X}}^c + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^c_N) + \tilde{\boldsymbol{V}}^+_{I,0}(\tilde{\boldsymbol{X}}^{sc} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_N) + \\ & \tilde{T}^t\left[\tilde{\boldsymbol{V}}^+_{R,1}(\tilde{\boldsymbol{X}}^{sc} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_N) - \tilde{\boldsymbol{V}}^+_{I,1}(\tilde{\boldsymbol{X}}^c - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^c_N)\right]\end{aligned} \tag{83}$$

The anti-causal filtering scheme, where $h_n = v_n = 0$, for $n < 0$, is obtained similarly, by setting the V^+-type matrices to zero, instead of the V^--type ones.

4.3 Causal-Symmetric Filtering

The best special case of the proposed scheme, in terms of complexity, is obtained with a 4-pixel delayed *causal-symmetric* filter, for which both the causality and the symmetry properties can be used to save computations.

A k-pixel delayed causal symmetric filter will be defined as a causal filter $\{h_n\}_{n=0}^k$ with $h_n = h_{2k-n}$ for all $0 \leq n \leq 2k$. Obviously, a causal symmetric filter is a delayed version of a non-causal filter that is symmetric about the origin.

In the causal symmetric case, when $k = 4$, the computational benefits of both symmetry and causality are combined. Specifically, on the top of the simplification due to causality for $\alpha = 1$, we also have

$$V^+_R(k) = H^+_R(k) = 0, \qquad k = 1, 3, 5, 7, \tag{84}$$
$$V^+_I(k) = H^+_I(k) = 0, \qquad k = 0, 2, 4, 6, 8. \tag{85}$$

Therefore, the matrices $\tilde{\boldsymbol{V}}^{+}_{p,q}$ and $\tilde{\boldsymbol{H}}^{+}_{p,q}$, $p \in \{R, I\}$, $q \in \{0, 1\}$, have only 4 nonzero elements each. This case can be of much interest for fast symmetric convolution with a short kernel, when the user does not mind a 4-pixel translation of the resulting image.

5 Implementation and Complexity

In this section, we focus on implementation considerations and on comparison between the proposed filtering scheme and previously reported schemes in terms of computational complexity.

So far we have studied merely the computational benefits associated with certain assumptions on the *filter* structure (symmetry, causality, and the combination of both). No assumptions were made, however, on the structure of the input *data*. An additional important factor to be taken into account in the implementation is that of typical sparseness of the quantized DCT input data blocks. Similarly as in [11], [12], and [7], we shall define a DCT coefficient block as *sparse* if only its 4×4 upper left quadrant (corresponding to low frequencies in both directions) contains nonzero elements. A very high percentage of the DCT blocks usually satisfy this requirement, which is fairly easy to check directly in the compressed format.

Incorporating the sparseness assumption in the above definition into the CST algorithm proposed in section 3.2, results in 7 multiplications and 17 additions, as opposed to 8 multiplications and 28 additions in the general case. As demonstrated below, these and other computation savings contribute to a significant increase in the efficiency of the proposed algorithm, in the case of sparse data.

This section is organized as follows. The different subsections correspond to the different degrees of generality of the assumed filter structure, similarly as in Section 4. In each subsection both the sparse and nonsparse input data cases will be studied. Computational complexity will be expressed in the form of $Mm + Aa$, which means M multiplications and A additions. For the purpose of comparison between the different schemes, we will assume a PA-RISC processor, where each multiplication is assumed to be equivalent to three additions on the average.

The General Case

As described in Appendix B.1, the general scheme, described by equations (73), (72), (67), and (66), can be implemented with a total of $1216m+2688a$ (6336 basic PA-RISC operations) per 8×8 image block, if sparseness is not assumed, or $716m + 1516a$ (3664 basic PA-RISC operations), if sparseness is assumed.

Let us compare the above result with the total number of computations associated with the straightforward approach of going explicitly via the spatial domain, detailed as follows:

1. Compute the IDCT $\boldsymbol{x}_{SE} = C^t \boldsymbol{X}^c_{SE} C$, and store for future use as $\boldsymbol{x}_E$, and $\boldsymbol{x}_{SE}$. (total: $16 \times (5m + 29a) = 80m + 464a$)

2. Compute the vertical convolution. $(64(M^+ - M^- + 1)m + 64(M^+ - M^-)a)$

3. Compute the horizontal convolution. $(64(N^+ - N^- + 1)m + 64(N^+ - N^-)a)$

4. Compute the DCT of the filtered block. $(16 \times (5m + 29a) = 80m + 464a)$

The total number of operations associated with the straightforward approach is $(64L + 288)m + (64L + 928)a$, where $L \triangleq (M^+ - M^-) + (N^+ - N^-)$. In terms of PA-RISC operations, the straightforward approach requires $256L + 1792$ operations.

According to the above figures, the proposed approach is preferable for every $L > 17$, when sparseness is not assumed. In the extreme case $L = 32$, it saves 37% of the computations. When operating on sparse data, the proposed approach is preferable for every $L > 7$, and it saves 63% of the computations in the extreme case.

Symmetric Filtering

The implementation of the proposed scheme for symmetric kernels is detailed in Appendix B.2. According to the analysis presented there, it requires $736m + 1984a$ (4192 basic PA-RISC operations) per image block, when sparseness is not assumed, and $448m + 1124a$ (2468 basic PA-RISC operations) when sparseness is assumed.

As before, we compare the results obtained for the proposed scheme to that of the straightforward approach. In the symmetric case, the straightforward approach can also be simplified, by using the kernel symmetry to efficiently perform the convolutions, leading to smaller complexity than in the general case. The symmetric straightforward approach requires $(64L^+ + 288)m + (128L^+ + 928)a$, where $L^+ \triangleq M^+ + N^+$. This represents $320L^+ + 1792$ basic PA-RISC operations.

We also compare the efficiency of the scheme with that of the scheme proposed in [7]. It requires $1152m + 1664a$ (5120 basic PA-RISC operations) when sparseness is not assumed, and $432m + 544a$ (1840 basic PA-RISC operations) when sparseness is assumed. The symmetric and causal-symmetric are the only particular cases of the proposed scheme which can be compared to [7]. This is because symmetry of the kernel is assumed there.

According to the above results, for non-sparse data the proposed scheme provides the best results, in terms of complexity, unless the kernel is short ($L^+ \leq 7$), in which case the straightforward approach is preferable. If the DCT-data is sparse, then the scheme proposed in [7] provides the best results, even for short kernels. As mentioned before, and stressed in the sequel, if a 4-pixel translation of the filtered image is tolerable, and if the kernel has up to 9 nonzero terms, then the proposed scheme using a causal-symmetric offers an efficient alternative to this case.

Causal Filtering

Appendix B.3 analyzes the proposed scheme for causal filtering. It indicates that $736m + 1728a$ are required for non-sparse data, whereas $448m + 980a$ are required for sparse data.

By comparing the above to the straightforward approach using a causal kernel (in which case, $L \leq 16$), the proposed scheme in this case is preferable to the straightforward approach when the data is sparse and $L > 2$, or when the data is not sparse and $L > 8$.

Causal-Symmetric Filtering

The implementation and complexity of the proposed algorithm when using a causal-symmetric kernel are shown in Appendix B.4. It requires $512m + 1280a$ and $296m + 688a$ in the non-sparse and sparse cases, respectively.

The proposed algorithm is compared to the *symmetric* version of the straightforward approach, which requires $(64\tilde{L} + 288)m + (128\tilde{L} + 928)$, where $\tilde{L} \triangleq (M^+ - 1)/2 + (N^+ - 1)/2$. This is equivalent to $320\tilde{L}+1792$ basic PA-RISC operations. Moreover, the causal-symmetric approach is compared to the *symmetric (non-causal)* approach in [7], because causal kernels are not allowed there. The complexity figures for the approach in [7], used in the comparison here are identical to those used in the comparison for the symmetric case above.

By performing the above comparisons, we reach the following conclusions. For sparse data, the proposed approach is the most efficient one, regardless of the size of the filter. Furthermore, for non-sparse data, it is the most efficient approach, for every $\tilde{L} > 3$.

6 Conclusion

In this work, we introduce the use of the Discrete Sine Transform (together with the Discrete Cosine Transform) for image processing in the DCT-domain. This is shown to provide good theoretical basis and tools for analysis in that domain. Moreover, by manipulating some of these theoretical tools, a fast filtering scheme, suitable to be used on compressed digital video and images, using the two transformations, was developed and proposed.

Comparison between the proposed algorithm and the straightforward approach, of converting back to the uncompressed domain, convolving in the spatial domain, and re-transforming to the DCT domain, was carried out. Comparison to the approach recently proposed in [7] is also presented. It was demonstrated that, by taking into account the typical sparseness of the input DCT-data, the proposed algorithm provides the best results for symmetric filtering, if a 4-pixel translation of the image in both directions is allowed. In this case, 35-64% of the computations are saved, depending on the kernel size, in comparison to the straightforward algorithm, and 14% of computations savings are obtained in comparison to the approach in [7]. The approach is also typically the most efficient one for long or medium-length, non-symmetric, kernels. Twice as much memory is required by the proposed algorithm in comparison to the two others, since DCT *and* DST coefficients are to be temporarily stored, instead of only the spatial data (as in the straightforward approach) or only the DCT coefficients (as in [7]).

Appendices

A Proof of Theorem 1

Let us define

$$G_{R,0} \triangleq \left(\frac{G_2 + G_2^t}{2} + \frac{G_1 + G_1^t}{2} \Phi \right). \tag{86}$$

The i-th column of $G_{R,0}$, which we will denote $G_{R,0}^{(i)}$ is given by:

$$\begin{aligned} G_{R,0}^{(i)} &= \frac{1}{2}(g_i,\ldots,g_0,\overbrace{0,\ldots,0}^{7-i\ \text{times}})^t + \frac{1}{2}(\overbrace{0,\ldots,0}^{i\ \text{times}},g_0,\ldots,g_{7-i})^t + \\ & \frac{1}{2}(g_{i+1},\ldots,g_8,\underbrace{0,\ldots,0}_{i\ \text{times}})^t + \frac{1}{2}(\underbrace{0,\ldots,0}_{7-i\ \text{times}},g_8,\ldots,g_{8-i})^t \end{aligned} \tag{87}$$

The i-th column of the product $CG_{R,0}$ is the DCT of $G_{R,0}^{(i)}$. Therefore, the elements $(CG_{R,0})(k,i)$, $k = 0,\ldots,7$, $i = 0,\ldots,7$, of the matrix $CG_{R,0}$ are given by:

$$\begin{aligned} (CG_{R,0})(k,i) = \frac{\gamma(k)}{2}\Bigg\{ & \sum_{n=0}^{i}\frac{g_{i-n}}{2}\cos\left(\frac{2n+1}{16}\pi k\right) + \sum_{n'=i}^{7}\frac{g_{n'-i}}{2}\cos\left(\frac{2n'+1}{16}\pi k\right) + \\ & \sum_{m=0}^{7-i}\frac{g_{i+1+m}}{2}\cos\left(\frac{2m+1}{16}\pi k\right) + \sum_{m'=7-i}^{7}\frac{g_{15-m'-i}}{2}\cos\left(\frac{2m'+1}{16}\pi k\right)\Bigg\}. \end{aligned} \tag{88}$$

We perform the following changes of variable in (88): $n \to i-\ell$, $n' \to \ell+i$, $m \to \ell-i-1$, and $m' = 15-\ell-i$, obtaining:

$$\begin{aligned} (CG_{R,0})(k,i) = \frac{1}{2}\cdot\frac{\gamma(k)}{2}\Bigg\{ & \sum_{\ell=0}^{i} g_\ell \cos\left(\frac{2(i-\ell)+1}{16}\pi k\right) + \sum_{\ell=0}^{7-i} g_\ell\cos\left(\frac{2(i+\ell)+1}{16}\pi k\right) + \\ & \sum_{\ell=i+1}^{8} g_\ell\cos\left(\frac{2(\ell-i-1)+1}{16}\pi k\right) + \sum_{\ell=8-i}^{8} g_\ell\cos\left(\frac{2(15-i-\ell)+1}{16}\pi k\right)\Bigg\}. \end{aligned} \tag{89}$$

Next, we use the following properties, due to the symmetry and periodicity of function cosine:

$$\cos\left(\frac{2(\ell-i-1)+1}{16}\pi k\right) = \cos\left(\frac{2(i-\ell)+1}{16}\pi k\right), \tag{90}$$

$$\cos\left(\frac{2(15-i-\ell)+1}{16}\pi k\right) = \cos\left(\frac{2(i+\ell)+1}{16}\pi k\right), \tag{91}$$

which yield

$$\begin{aligned} (CG_{R,0})(k,i) &= \frac{\gamma(k)}{2}\left\{\sum_{\ell=0}^{8} g_\ell\cdot\frac{1}{2}\left[\cos\left(\frac{2(i-\ell)+1}{16}\pi k\right)+\cos\left(\frac{2(i+\ell)+1}{16}\pi k\right)\right]\right\} \\ &= \left\{\sum_{\ell=0}^{8} g_\ell\cdot\cos\left(\frac{2\ell}{16}\pi k\right)\right\}\left\{\frac{\gamma(k)}{2}\cos\left(\frac{2i+1}{16}\pi k\right)\right\} \qquad (92) \\ &= G_R(k)C(k,i) = (\mathrm{diag}\{G_R(j)\}_{j=0}^{7}\cdot C)(k,i). \qquad (93) \end{aligned}$$

Therefore,

$$C\cdot G_{R,0} = \mathrm{diag}\{G_R(j)\}_{j=0}^{7}\cdot C \tag{94}$$

which leads to (19).

The remaining relations in Theorem 1 are obtained similarly.

B Algorithmic Implementations

In the algorithms detailed throughout this appendix, W is a 8×8 temporary memory allocation. At each algorithm step, we provide between parenthesis the number of arithmetic operations associated with each step in the form of an expression $M \cdot m + A \cdot a$, which means M multiplications plus A additions.

B.1 General Algorithm

The general scheme is described by equations (73), (72), (67), and (66). This scheme can be implemented in the following way:

1. Compute $\tilde{\boldsymbol{X}}^{sc}_{SE} = \tilde{T}\tilde{\boldsymbol{X}}^{c}_{SE}$ and store for future use as $\tilde{\boldsymbol{X}}^{sc}_{E}$ and $\tilde{\boldsymbol{X}}^{sc}_{NE}$.
   (Not assuming sparseness: $8 \times (8m + 28a) = 64m + 224a$;
   assuming sparseness: $4 \times (7m + 17a) = 28m + 68a$)
2. Compute: $W \leftarrow \tilde{\boldsymbol{V}}^{+}_{R,1}(\tilde{\boldsymbol{X}}^{sc}_{E} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_{NE})$. (Non-sparse: $64m + 64a$; sparse: $32m + 32a$)
3. Compute: $W \leftarrow W - \tilde{\boldsymbol{V}}^{+}_{I,1}(\tilde{\boldsymbol{X}}^{c}_{E} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{c}_{NE})$. Notice that the kernel matrices $\tilde{\boldsymbol{V}}^{r}_{I,q}$, $q \in \{0, 1\}$, $r \in \{+, -\}$, have 7 non-null coefficients only, not 8. Therefore, in this case only 7 rows of the image blocks have to be summed. (Non-sparse: $56m + 112a$; sparse: $12m + 24a$)
4. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}^{-}_{R,1}(\tilde{\boldsymbol{X}}^{sc}_{E} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_{SE})$. (Non-sparse: $64m+128a$; sparse: $32m+64a$)
5. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}^{-}_{I,1}(\tilde{\boldsymbol{X}}^{c}_{E} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{c}_{SE})$. (Non-sparse: $56m+112a$; sparse: $12m+24a$)
6. Compute: $W \leftarrow \tilde{T}^{t}W$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $4 \times (8m + 28a) = 32m + 112a$)
7. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}^{+}_{R,0}(\tilde{\boldsymbol{X}}^{c}_{E} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{c}_{NE})$. (Non-sparse: $64m+128a$; sparse: $16m+32a$)
8. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}^{+}_{I,0}(\tilde{\boldsymbol{X}}^{sc}_{E} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_{NE})$. (Non-sparse: $56m+112a$; sparse: $32m+64a$)
9. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}^{-}_{R,0}(\tilde{\boldsymbol{X}}^{c}_{E} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{c}_{SE})$. (Non-sparse: $64m+128a$; sparse: $16m+32a$)
10. Compute: $\tilde{\boldsymbol{Z}}^{c}_{E} \leftarrow W - \tilde{\boldsymbol{V}}^{-}_{I,0}(\tilde{\boldsymbol{X}}^{sc}_{E} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_{SE})$, and store for the next 2 steps, to be used as $\tilde{\boldsymbol{Z}}^{c}$ and $\tilde{\boldsymbol{Z}}^{c}_{W}$. (Non-sparse: $56m + 112a$; sparse: $32m + 64a$)
11. Compute $\tilde{\boldsymbol{Z}}^{cs}_{E} = \tilde{\boldsymbol{Z}}^{c}_{E}\tilde{T}^{t}$ and store for the next 2 steps, to be used as $\tilde{\boldsymbol{Z}}^{cs}$ and $\tilde{\boldsymbol{Z}}^{cs}_{W}$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $8 \times (7m + 17a) = 56m + 136a$)
12. Compute: $W \leftarrow (\tilde{\boldsymbol{Z}}^{cs} - \tilde{\boldsymbol{Z}}^{cs}_{W}\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^{+}_{R,1}$. (Non-sparse: $64m + 64a$; sparse: $64m + 64a$)
13. Compute: $W \leftarrow W - (\tilde{\boldsymbol{Z}}^{c} - \tilde{\boldsymbol{Z}}^{c}_{W}\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^{+}_{I,1})^{t}$. (Non-sparse: $56m+112a$; sparse: $24m+48a$)

14. Compute: $W \leftarrow W + (\tilde{\boldsymbol{Z}}^{cs} - \tilde{\boldsymbol{Z}}^{cs}_{E}\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^{-}_{R,1}$. (Non-sparse: $64m+128a$; sparse: $64m+128a$)

15. Compute: $W \leftarrow W + (\tilde{\boldsymbol{Z}}^{c} - \tilde{\boldsymbol{Z}}^{c}_{E}\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^{-}_{I,1})^t$. (Non-sparse: $56m+112a$; sparse: $24m+48a$)

16. Compute $W \leftarrow W\tilde{T}$. (Non-sparse and sparse: $8 \times (8m + 28a) = 64m + 224a$)

17. Compute: $W \leftarrow W + (\tilde{\boldsymbol{Z}}^{c} + \tilde{\boldsymbol{Z}}^{c}_{W}\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^{+}_{R,0}$. (Non-sparse: $64m + 128a$; sparse: $32m + 64a$)

18. Compute: $W \leftarrow W + (\tilde{\boldsymbol{Z}}^{cs} + \tilde{\boldsymbol{Z}}^{cs}_{W}\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^{+}_{I,0})^t$. (Non-sparse and sparse: $56m + 112a$)

19. Compute: $W \leftarrow W + (\tilde{\boldsymbol{Z}}^{c} + \tilde{\boldsymbol{Z}}^{c}_{E}\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^{-}_{R,0}$. (Non-sparse: $64m + 128a$; sparse: $32m + 64a$)

20. Compute: $\tilde{\boldsymbol{Y}}^{c} \leftarrow W - (\tilde{\boldsymbol{Z}}^{cs} + \tilde{\boldsymbol{Z}}^{cs}_{E}\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^{-}_{I,0})^t$. (Non-sparse and sparse: $56m + 112a$)

The total number of operations is $1216m + 2688a$, in the non-sparse case, and $716m + 1516a$, in the sparse case.

B.2 Proposed Algorithm for Symmetric Kernels

The algorithm is implemented in the following way:

1. Compute $\tilde{\boldsymbol{X}}^{sc}_{SE} = \tilde{T}\tilde{\boldsymbol{X}}^{c}_{SE}$ and store for future use as $\tilde{\boldsymbol{X}}^{sc}_{E}$ and $\tilde{\boldsymbol{X}}^{sc}_{NE}$.
(Not assuming sparseness: $8 \times (8m + 28a) = 64m + 224a$;
assuming sparseness: $4 \times (7m + 17a) = 28m + 68a$)

2. Compute: $W \leftarrow \tilde{\boldsymbol{V}}_{R,1}[2\tilde{\boldsymbol{X}}^{sc}_{E} - \boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^{sc}_{NE} + \tilde{\boldsymbol{X}}^{sc}_{SE})]$. (Non-sparse: $64m + 128a$; sparse: $32m + 64a$)

3. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}_{I,1}\boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^{c}_{NE} - \tilde{\boldsymbol{X}}^{c}_{SE})$. (Non-sparse: $56m + 112a$; sparse: $12m + 24a$)

4. Compute: $W \leftarrow \tilde{T}^t W$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $4 \times (8m + 28a) = 32m + 112a$)

5. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}_{R,0}[2\tilde{\boldsymbol{X}}^{c}_{E} + \boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^{c}_{NE} + \tilde{\boldsymbol{X}}^{c}_{SE})]$. (Non-sparse: $64m + 192a$; sparse: $16m + 48a$)

6. Compute: $\tilde{\boldsymbol{Z}}^{c}_{E} \leftarrow W + \tilde{\boldsymbol{V}}_{I,0}\boldsymbol{\Phi}(\tilde{\boldsymbol{X}}^{sc}_{NE} - \tilde{\boldsymbol{X}}^{sc}_{SE})$, and store for the next 2 steps, to be used as $\tilde{\boldsymbol{Z}}^{c}$ and $\tilde{\boldsymbol{Z}}^{c}_{W}$. (Non-sparse: $56m + 112a$; sparse: $32m + 64a$)

7. Compute $\tilde{\boldsymbol{Z}}^{cs}_{E} = \tilde{\boldsymbol{Z}}^{c}_{E}\tilde{T}^t$ and store for the next 2 steps, to be used as $\tilde{\boldsymbol{Z}}^{cs}$ and $\tilde{\boldsymbol{Z}}^{cs}_{W}$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $8 \times (7m + 17a) = 56m + 136a$)

8. Compute: $W \leftarrow [2\tilde{\boldsymbol{Z}}^{cs} - (\tilde{\boldsymbol{Z}}^{cs}_{W} + \tilde{\boldsymbol{Z}}^{cs}_{E})\boldsymbol{\Phi}]\tilde{\boldsymbol{H}}_{R,1}$. (Non-sparse and sparse: $64m + 128a$)

9. Compute: $W \leftarrow W + (\tilde{\boldsymbol{Z}}^c_W - \tilde{\boldsymbol{Z}}^c_E)\boldsymbol{\Phi}(\tilde{\boldsymbol{H}}_{I,1})^t$. (Non-sparse: $56m + 112a$; sparse: $24m + 48a$)

10. Compute $W \leftarrow W\tilde{T}$. (Non-sparse and sparse: $8 \times (8m + 28a) = 64m + 224a$)

11. Compute: $W \leftarrow W + [2\tilde{\boldsymbol{Z}}^c + (\tilde{\boldsymbol{Z}}^c_W + \tilde{\boldsymbol{Z}}_E)\boldsymbol{\Phi}]\tilde{\boldsymbol{H}}_{R,0}$. (Non-sparse: $64m + 192a$; sparse: $32m + 96a$)

12. Compute: $\tilde{\boldsymbol{Y}}^c \leftarrow W + (\tilde{\boldsymbol{Z}}^{cs}_W - \tilde{\boldsymbol{Z}}^{cs}_E)\boldsymbol{\Phi}(\tilde{\boldsymbol{H}}_{I,0})^t$. (Non-sparse and sparse: $56m + 112a$)

The total number of operations is $736m + 1984a$, in the non-sparse case, and $448m + 1124a$, in the sparse case.

B.3 Proposed Scheme for Causal Filtering

The proposed scheme for causal filtering is described by equations (82), (72), (83), and (66). This scheme can be implemented in the following way:

1. Compute $\tilde{\boldsymbol{X}}^{sc} = \tilde{T}\tilde{\boldsymbol{X}}^c$ and store for future use as $\tilde{\boldsymbol{X}}^{sc}_N$. (Not assuming sparseness: $8 \times (8m + 28a) = 64m + 224a$; assuming sparseness: $4 \times (7m + 17a) = 28m + 68a$)

2. Compute: $W \leftarrow \tilde{\boldsymbol{V}}^+_{R,1}(\tilde{\boldsymbol{X}}^{sc} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_N)$. (Non-sparse: $64m + 64a$; sparse: $32m + 32a$)

3. Compute: $W \leftarrow W - \tilde{\boldsymbol{V}}^+_{I,1}(\tilde{\boldsymbol{X}}^c - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^c_N)$. (Non-sparse: $56m+112a$; sparse: $12m+24a$)

4. Compute: $W \leftarrow \tilde{T}^tW$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $4 \times (8m + 28a) = 32m + 112a$)

5. Compute: $W \leftarrow W + \tilde{\boldsymbol{V}}^+_{R,0}(\tilde{\boldsymbol{X}}^c + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^c_N)$. (Non-sparse: $64m+128a$; sparse: $16m+32a$)

6. Compute: $\tilde{\boldsymbol{Z}}^c \leftarrow W + \tilde{\boldsymbol{V}}^+_{I,0}(\tilde{\boldsymbol{X}}^{sc} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_N) + W$, and store for the next 1 step, to be used as $\tilde{\boldsymbol{Z}}^c_W$. (Non-sparse: $56m + 112a$; sparse: $32m + 64a$)

7. Compute $\tilde{\boldsymbol{Z}}^{cs} = \tilde{\boldsymbol{Z}}^c\tilde{T}^t$ and store for the next 1 step, to be used as $\tilde{\boldsymbol{Z}}^{cs}_W$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $8 \times (7m + 17a) = 56m + 136a$)

8. Compute: $W \leftarrow (\tilde{\boldsymbol{Z}}^{cs} - \tilde{\boldsymbol{Z}}^{cs}_W\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^+_{R,1}$. (Non-sparse: $64m + 64a$; sparse: $64m + 64a$)

9. Compute: $W \leftarrow W - (\tilde{\boldsymbol{Z}}^c - \tilde{\boldsymbol{Z}}^c_W\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^+_{I,1})^t$. (Non-sparse: $56m+112a$; sparse: $24m+48a$)

10. Compute $W \leftarrow W\tilde{T}$. (Non-sparse and sparse: $8 \times (8m + 28a) = 64m + 224a$)

11. Compute: $W \leftarrow W + (\tilde{\boldsymbol{Z}}^c + \tilde{\boldsymbol{Z}}^c_W\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^+_{R,0}$. (Non-sparse: $64m+128a$; sparse: $32m+64a$)

12. Compute: $\tilde{\boldsymbol{Y}}^c \leftarrow W + (\tilde{\boldsymbol{Z}}^{cs} + \tilde{\boldsymbol{Z}}^{cs}_W\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^+_{I,0})^t$. (Non-sparse and sparse: $56m + 112a$)

The total number of operations is $736m + 1728a$, in the non-sparse case, and $448m + 980a$, in the sparse case.

B.4 Proposed Scheme for Causal-Symmetric Filtering

The implementation in this case can be done as follows:

1. Compute $\tilde{\boldsymbol{X}}^{sc} = \tilde{T}\tilde{\boldsymbol{X}}^{c}$ and store for future use as $\tilde{\boldsymbol{X}}^{sc}_{N}$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $4 \times (7m + 17a) = 28m + 68a$)

2. Compute: Odd rows of $W \leftarrow \tilde{\boldsymbol{V}}^{+}_{R,1}(\tilde{\boldsymbol{X}}^{sc} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_{N})$. Note that only the odd rows of the data vectors have to be summed. (Non-sparse: $32m + 32a$; sparse: $16m + 16a$)

3. Compute: Even rows of $W \leftarrow -\tilde{\boldsymbol{V}}^{+}_{I,1}(\tilde{\boldsymbol{X}}^{c} - \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{c}_{N})$. Here only the even rows of the data vectors have to be summed. (Non-sparse: $32m + 32a$; sparse: $8m + 8a$)

4. Compute $W \leftarrow \tilde{T}^{t}W$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $4 \times (8m + 28a) = 32m + 112a$)

5. Compute: Odd rows of $\tilde{\boldsymbol{Z}}^{c} \leftarrow \tilde{\boldsymbol{V}}^{+}_{R,0}(\tilde{\boldsymbol{X}}^{c}+\boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{c}_{N})$+Odd rows of W. (Non-sparse: $32m+64a$; sparse: $8m + 16a$)

6. Compute: Even rows of $\tilde{\boldsymbol{Z}}^{c} \leftarrow \tilde{\boldsymbol{V}}^{+}_{I,0}(\tilde{\boldsymbol{X}}^{sc} + \boldsymbol{\Phi}\tilde{\boldsymbol{X}}^{sc}_{N})$ + Even rows of W. (Non-sparse: $32m + 64a$; sparse: $16m + 32a$)

7. Compute $\tilde{\boldsymbol{Z}}^{cs} = \tilde{\boldsymbol{Z}}^{c}\tilde{T}^{t}$ and store for future use as $\tilde{\boldsymbol{Z}}^{cs}_{W}$. (Non-sparse: $8 \times (8m + 28a) = 64m + 224a$; sparse: $8 \times (7m + 17a) = 28m + 68a$)

8. Compute: Odd columns of $W \leftarrow (\tilde{\boldsymbol{Z}}^{cs} - \tilde{\boldsymbol{Z}}^{cs}_{W}\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^{+}_{R,1}$. Note that only the odd columns of the data vectors have to be summed. (Non-sparse and sparse: $32m + 32a$)

9. Compute: Even columns of $W \leftarrow -(\tilde{\boldsymbol{Z}}^{c} - \tilde{\boldsymbol{Z}}^{c}_{W}\boldsymbol{\Phi})(\tilde{\boldsymbol{H}}^{+}_{I,1})^{t}$. Here only the even columns of the data vectors have to be summed. (Non-sparse: $32m + 32a$; sparse: $16m + 16a$)

10. Compute $W \leftarrow W\tilde{T}$. (Non-sparse and sparse: $8 \times (8m + 28a) = 64m + 224a$)

11. Compute: Odd columns of $\tilde{\boldsymbol{Y}}^{c} \leftarrow$ Odd columns of $W + (\tilde{\boldsymbol{Z}}^{c} + \tilde{\boldsymbol{Z}}^{c}_{W}\boldsymbol{\Phi})\tilde{\boldsymbol{H}}^{+}_{R,0}$. (Non-sparse: $32m + 64a$; sparse: $16m + 32a$)

12. Compute: Even columns of $\tilde{\boldsymbol{Y}}^{c} \leftarrow$ Even columns of $W + (\tilde{\boldsymbol{Z}}^{cs} + \boldsymbol{\Phi}\tilde{\boldsymbol{Z}}^{cs}_{W})(\tilde{\boldsymbol{H}}^{+}_{I,0})^{t}$. (Non-sparse and sparse: $32m + 64a$)

The total number of operations is $512m + 1280a$, when sparseness is not assumed, and $296m + 688a$, when sparseness is assumed. In terms of basic PA-RISC operations, the average complexity is 2816 in the non-sparse case, and 1576 in the sparse case.

7 References

[1] W. H. Chen and S. C. Fralick, "Image enhancement using cosine transform filtering," *Image Sci. Math. Symp.*, Monterey, CA, November 1976.

[2] K. N. Ngan and R. J. Clarke, "Lowpass filtering in the cosine transform domain," *Int. Conf. on Commun.*, Seattle, WA, pp. 37.7.1-37.7.5, June 1980.

[3] B. Chitprasert and K. R. Rao, "Discrete cosine transform filtering," *Signal Processing*, Vol. 19, pp. 233-245, 1990.

[4] J. B. Lee and B. G. Lee, "Transform domain filtering based on pipelining structure," *IEEE Trans. on Signal Processing*, Vol. SP-40, no. 8, pp. 2061-2064, August 1992.

[5] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE J. Selected Areas in Communications*, Vol. 13, no. 1, pp. 1-11, January 1995.

[6] A. Neri, G. Russo, and P. Talone, "Inter-block filtering and downsampling in DCT domain," *Signal Processing: Image Communication*, Vol. 6, pp. 303-317, 1994.

[7] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT domain filtering," HPL Technical Report #HPL-95-56, May 1995.

[8] K. R. Rao, and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press 1990.

[9] Y. Arai, T. Agui, and M. Nakajima, "A Fast DCT-SQ Scheme for Images," *Trans. of the IEICE*, E 71(11):1095, November 1988.

[10] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.

[11] N. Merhav and V. Bhaskaran, "A transform domain approach to spatial domain image scaling," HPL Technical Report #HPL-94-116, December 1994.

[12] N. Merhav and V. Bhaskaran, "A fast algorithm for DCT-domain inverse motion compensation," HPL Technical Report #HPL-95-17, February 1995.