2-D Transform-Domain Resolution Translation

Jae-Beom Lee, *Member, IEEE,* and Alexandros Eleftheriadis, *Member, IEEE*

***Abstract*—The extensive use of discrete transforms in image and video coding suggests the investigation on filtering before downsampling (FBDS) and filtering after upsampling (FAUS) methods directly acting on the transform domain. In this paper, we describe the "transform-domain resolution translation" technique that gives flexibility to resize windows of each video conferencing session for server compositing without explicit decompression, spatial domain processing, and compression. We generalize transform-domain filtering (TDF) to include nonuniform and multirate cases to implement the transform-domain resolution translator. The former is defined as a TDF problem in which the original transform domain is of different size from the target one, while the latter considers the implementation of sampling rate conversion in the transform domain. The implementation architecture is based on a pipeline that involves matrix–vector product blocks and vector addition, but is not limited to particular hardware. Such techniques are particularly useful for fast algorithms for processing compressed images and video where transform coding is extensively used (e.g., in JPEG, H.261, MPEG-1, MPEG-2, and H.263).**

***Index Terms*—Multirate transform-domain filtering (MTFD), nonuniform transform-domain filtering (NTDF), transform-domain filtering (TDF), transform-domain resolution translation (TDRT).**

I. Introduction

TRANSFORM coding techniques are widely used for compressing digital image and audio signals. In particular, the discrete cosine transform (DCT) is used as the primary means for image and video compression and transmission [1]–[4], and is at the core of several international standards (e.g., JPEG, H.261, MPEG-1, MPEG-2, and H.263). The large data rates (more than 100 Mb/s) associated with uncompressed video necessitate its storage and transmission in a compressed form; in several instances, however, it is desired that processing operations are applied on the compressed data. Examples include traditional image and video production facilities, as well as modern distributed multimedia systems that bring editing and processing capabilities to end-users.

Typical operations are pixel-oriented and applied in the spatial domain, and include overlays, translation, scaling, linear filtering, rotation, etc. The straightforward approach of decompressing, processing, and recompressing is undesirable due to the significant computational overhead associated with compression and decompression (particularly for highly asymmetric codecs like MPEG). This makes it desirable that these operations are applied directly in the compressed or transform domain. The computational overhead can then be significantly reduced, as it only involves parsing of the data up to the point where transform coefficient data is available, and regeneration of data based on the new transform coefficient values. As an example, direct transform-domain manipulation has been employed in [5] and [6] in order to provide fast video compositing algorithms, whereas in [7] and [8], it has been used to perform rate changes.

With respect to transform-domain filtering (TDF), a number of techniques have been applied to provide convolution-multiplication properties to the DCT [9]–[11]. Such "DCT filtering" approaches successfully reduce the number of multiplications and additions, but also possess some limitations and imperfections. For example, the scheme in [10] relies on a distortion factor $w(n)$ which is difficult to implement, while the scheme in [9] requires that filter coefficients are real and symmetric. Furthermore, since both schemes are concerned with circular convolution applied to individual signal blocks, they cannot implement linear filtering or avoid block-edge effects.

In recent work [12], Lee and Lee proposed the concept of TDF as a technique to get around these problems, taking advantage of advanced modern VLSI technology. TDF is a block-based filtering process that is applied to transform-domain data and that can implement the desired time domain filtering. It is shown in [12] that the existing DCT filtering approach can be generalized to TDF, and a pipelining structure was presented as a means to implement it. TDF has been extended into infinite-impulse response (IIR) structures in [13] by Kim and Lee. Moreover, Chang and Messerschmitt showed that TDF is still useful in software architectures since transform data usually take on immensely compressed form such as DCT [5], [6]; the compression ratios achieved in practical DCT-based codecs are 50–100 to 1, and only a small proportion of transform data needs computation in software architecture.

More recently, direct DCT domain image downsampling problems were investigated in [14]–[16] to provide an alternative approach wherein the data stream is processed in the compressed DCT domain without explicit decompression and spatial-domain processing, and so that the output compressed stream, corresponding to the output image, conforms to the standard syntax of 8×8 blocks [15]. Beneficial examples include a multiparty video communication system through a server. As explained in [15], a videoconferencing session of several parties requires each one of them to see everybody else in a separate window on his screen. Every user would like to have the flexibility to resize windows, move them from one location on the screen to another, and so on. Since each workstation is capable of handling one video stream only, the server must compose the bit streams from all parties to a single stream whose architecture depends on user's request. If one

Manuscript received March 13, 1998; revised December 15, 1999. This paper was recommended by Associate Editor R. L. Stevenson.

The authors are with the Department of Electrical Engineering, Columbia University, New York, NY, 10027 USA (e-mail: jbl@ee.columbia.edu; eleft@ee.columbia.edu).

Publisher Item Identifier S 1051-8215(00)06555-1.

user wishes, say, to scale down by a factor of two a window corresponding to another user and move it to a different place on the screen, the server must support a downsampling. The one step-size conversion operation directly in the DCT domain will reduce the computational burden, since the decompression and compression processes are typically computational bottlenecks. For example, the inverse DCT (IDCT) for decompression process requires 38.7% of the overall execution time on a typical workstation [15].

In this paper, we extend the previous idea of [15] into a general case where the size of windows can be changed with arbitrary scale (i.e., fractional or even scaling up). In fact, arbitrary or upscaling modification directly in the DCT domain were not possible in that approach. To make the generalized work possible, we introduce the concept of "transform-domain resolution translation" (TDRT) as a combined form of the transform-domain FBDS and FAUS issues, and then propose the solution with in the context of TDF.

In Section II we review the core of uniform TDF, and extend it by introducing the concept of non-uniform TDF (NTDF), in which the two transform sizes need not be the same. A modified TDF is applied by mapping the nonuniform problem to a uniform one. As an example of the applications of NTDF, we show how it can be used to provide a general solution to the direct computation of transform coefficients in the "sub-adjacent block problem" given in [17]. Section III extends TDF to include multirate processing as well, resulting to multirate TDF (MTDF). Again, a modified TDF is applied, by mapping the multirate problem into a nonuniform one. An example is given by providing a general solution to the 1-D TDRT problem. We generalize it in the next section to address the 2-D TDRT problem, which is actually a 2-D version of the generalized transform-domain FBDS and FAUS issues given in [14] and [15]. Finally, in Section V, we discuss implementation considerations and present some concluding remarks.

II. Uniform and Nonuniform TDF

A. The Uniform TDF

Lee and Lee defined TDF as an operation which has the same functionality as a combination of transform, filtering, and transform [12]. The TDF problem for the case where the input and output transform sizes are equal is depicted in Fig. 1(a) [12]. In this paper, this will be referred to as uniform TDF, since the original transform domain is of same size as the target one. In a traditional approach, T_1 would be the IDCT, T_2 would be the DCT, and $h(n)$ would be the desired linear filter. The pipelined implementation shown in Fig. 1(b) involves a set of matrix-vector multiplication modules (P_i), and a vector adder. Denoting by M the transform block size and by q the filter length, we can define:

$$H = [h_{ij}]_{(M+q-1)\times M} \tag{1}$$

where

$$h_{ij} = \begin{cases} h(i-j), & j \le i < j+q \\ & i = 0, 1, \ldots, M+q-2 \\ & j = 0, 1, \ldots, M-1 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

and

$$W_j = [w_{kl}]_{M\times(M+q-1)} \tag{3}$$

with

$$w_{kl} = \begin{cases} 1, & l = k + jM \\ & k = 0, 1, \ldots, M-1 \\ & l = 0, 1, \ldots, M+q-2 \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

The multiplying matrices can then be obtained as

$$P_j \stackrel{\text{def}}{=} T_2 W_j H T_1 \tag{5}$$

resulting to an output from the vector adder

$$Y_i = \sum_{j=0}^{l} P_j X_{i-j} \tag{6}$$

where

$$l = \left\lfloor \frac{M+q-1}{M} \right\rfloor. \tag{7}$$

VLSI hardware implementation issues have shifted from computational complexity to interconnection complexity. In VLSI, memory and processing power are relatively cheap. Therefore, the main emphasis of design has moved toward reducing the overall interconnection complexity (i.e., keeping the overall architecture highly regular, parallel, and pipelined) [18]. The computational complexity is not a major issue in TDF, since it is designed to take advantage of modern VLSI architectures. That is, to reduce interconnection complexity of the TDF, the proposed structure has a pipelining structure of matrix-vector product units which are well known to be the simplest units for localized implementation [12], [18].

For software implementations, on the other hand, recent research [5], [6] concluded that computational complexity is, once again, not a major issue in TDF; the efficiency of the computation does not come from the low computational complexity of TDF, but from the fact that TDF deals with already immensely compressed data (i.e., DCT).

These arguments also apply to the TDRT, since it is a direct variant of TDF.

B. NTDF

We define NTDF as a transform-domain filter which has the same functionality as a combination of transform, filtering, and transform, where the two transforms are of different sizes. The block diagram for an NTDF system is shown in Fig. 2(a). Note that the sizes of the input (T_1) and output transform (T_2), M_1 and M_2 respectively, are not necessarily the same. In the case where $M_1 = M_2$ we have the uniform TDF problem discussed in Section I-A. We can map the nonuniform problem to a uniform one by extending the transform matrices. In particular, let

$$T_1' = I_{(\mathrm{lcm}(M_1, M_2)/M_1)\times(\mathrm{lcm}(M_1, M_2)/M_1)} \otimes T_1 \tag{8}$$

and

$$T_2' = I_{(\mathrm{lcm}(M_1, M_2)/M_2)\times(\mathrm{lcm}(M_1, M_2)/M_2)} \otimes T_2 \tag{9}$$

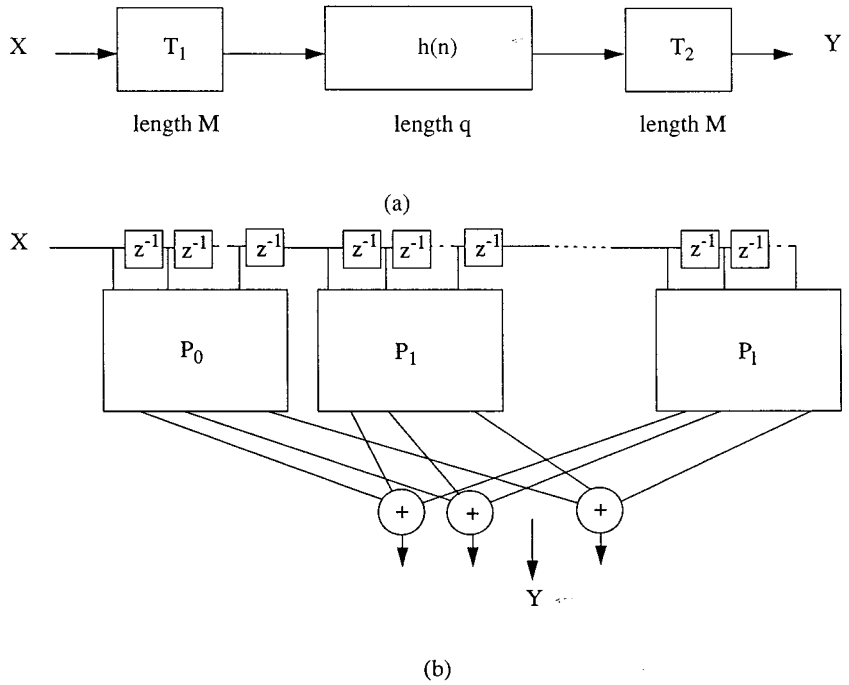


Fig. 1. Single-stage uniform TDF structure. (a) System model. (b) Pipelined implementation architecture.

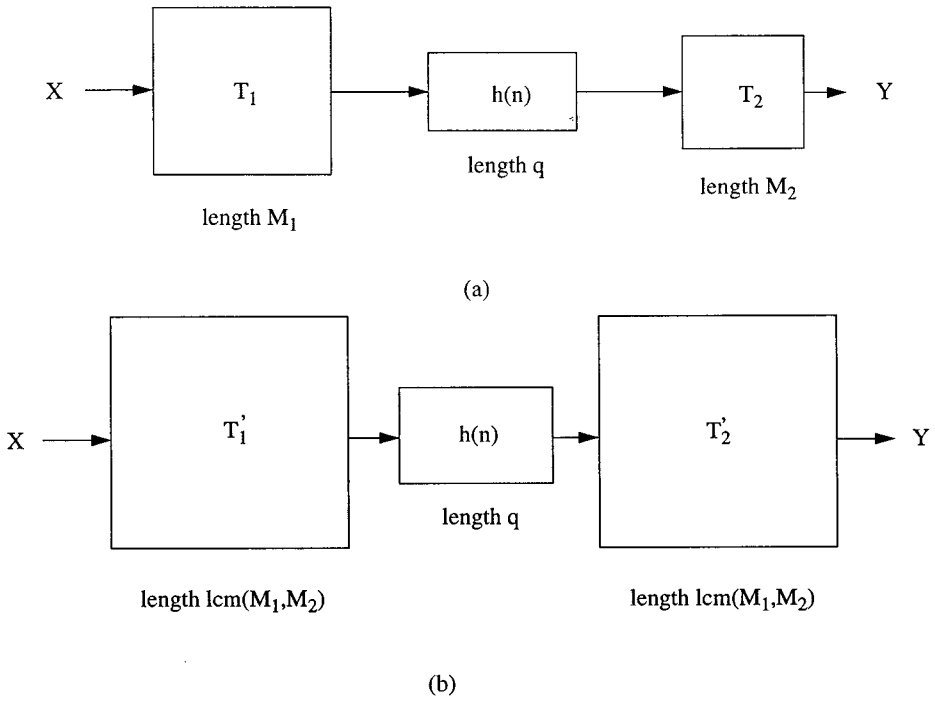


Fig. 2. Single-stage NTDF structure. (a) System model. (b) Equivalent uniform TDF model.

where $\otimes$ denotes the Kronecker matrix product, and where $I_{m\times n}$ is a matrix with ones in the major diagonal and zeros elsewhere. Note that the transform block sizes for T_1' and T_2' are both lcm(M_1, M_2) where lcm$(\cdot)$ denotes the least common multiple, and hence the uniform TDF results can be applied. As a result, the pipelining multiplication matrices become

$$P_j = T_2' W_j H T_1' = (I \otimes T_2) W_j H (I \otimes T_1). \tag{10}$$

By defining

$$M \stackrel{\text{def}}{=} \text{lcm}(M_1, M_2) \tag{11}$$

the various parameters of the system are given only if $M = \text{lcm}(M_1, M_2)$ by (1)–(7). The equivalent system model is shown in Fig. 2(b).

A concern in terms of implementation complexity is the size M of the expanded transform matrices T_1' and T_2'. We should

note, however, that in most typical situations transform sizes are powers of 2. In this case, M would simply be equal to the largest of M_1 and M_2, i.e.

$$M = \text{lcm}(M_1 = 2^k, M_2 = 2^l) = \begin{cases} M_1, & k \geq l \\ M_2, & k < l \end{cases} \qquad (12)$$

thus making the hardware requirements equivalent to that of uniform TDF.

The following example provides a general solution to the direct computation of transform coefficients for the sub-adjacent block problem using the NTDF method.

C. The Sub-Adjacent Block Problem

As an example of the use of the NTDF approach, and due to its importance in practical video-processing applications, we discuss the sub-adjacent block problem. Here we are given the transform coefficients of a rectangular array of blocks, and we are interested in obtaining the transform coefficients of a block which is not aligned perfectly with the original block structure. In the 2-D case, such a block overlaps with up to four adjacent blocks. In this example, we consider only one dimension for simplicity, and hence, overlap will occur with only two blocks; generalization to 2-D is straightforward. The obvious method of solving this problem is to take the inverse transform, and then compute the forward transform for the sub-adjacent block. In recent work, transform-domain manipulation techniques have been used to provide a direct method for the sub-adjacent block problem [19], [20]. The proposed solution in [19] is restricted only to DCT. In addition, a new block is formed by taking the "halves" of two adjacent blocks, so that an offset of only four is allowed between the input signal and output signals in the case of an 8-point transform. The proposed method in [20] gives a particular solution for a special kind of transform family, including WH, for which an output block is formed by taking some fixed pattern (i.e., not halves) of two adjacent blocks in [21]. The solutions are given based on a specific fast algorithm so that for practical applications, several kinds of processors should be designed due to different combination of transforms and delays. In other words, this is not a general solution for arbitrary offset patterns and transform pairs. To overcome these drawbacks, we propose a generalized solution using the NTDF method. To cast the sub-adjacent block problem in an NTDF context, we consider a simple delay filter as the offset operation. We can notice that the procedure takes the form "transform-filter-transform" with potentially nonuniform block size. This is exactly the NTDF structure, and hence can be directly implemented using (8)–(10).

As an example, we give the solution for the sub-adjacent block problem for the IDCT and Haar transform pair; when the IDCT block size is four, the Haar transform block size is eight, and a new block is formed by a shift of two samples. This example can be thought of as a DCT-domain edge detector. A shift of two samples implies a second-order pure delay filter

$$M_1 = 4, \quad M_2 = 8, \quad M = \text{lcm}(4, 8) = 8$$
$$\frac{\text{lcm}(M_1, M_2)}{M_1} = 2, \quad \frac{\text{lcm}(M_1, M_2)}{M_2} = 1$$
$$T'_{\text{IDCT}} = I_{2\times 2} \otimes T_{\text{IDCT}}$$
$$T'_{\text{Haar}} = I_{1\times 1} \otimes T_{\text{Haar}}.$$

Other parameters, including the offset filter, are given as follows:

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

$$W_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$W_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$P_j \equiv T'_2 W_j H T'_1$$
$$Z_i = \sum_{j=0}^{l} P_j X_{i-j}$$
$$l = \lfloor \frac{10}{8} \rfloor = 1.$$

The block diagram structure is exactly the same as Fig. 1.

Note that the matrix product $W_j H$ is operation of partition on filter matrix H implicitly. Once an filter coefficient set is given, we can explicitly write down the partitioned matrices. At this moment, the above expression is more convenient, but we use partitioned matrices expression in Section IV, since the filter coefficient set is fixed.

III. Multirate TDF

In this section, we generalize NTDF to MTDF in order to consider the case where the input and output rates are different. We define MTDF as a transform-domain filter which has the same functionality as a combination of transform, rate-change operation, filtering, rate-change operation, and transform as shown in Figs. 3 and 4. Note that this is a general case of NTDF combined with a decimator and a interpolator. MTDF is applicable to any combination of appropriate transforms, and also provides arbitrary fractional rate change functionality. In order to obtain an explicit representation in a form similar to the NTDF and uniform TDF structures, we divide the definition into two cases, as shown in Figs. 3 and 4. We define MTDF Case I as the one

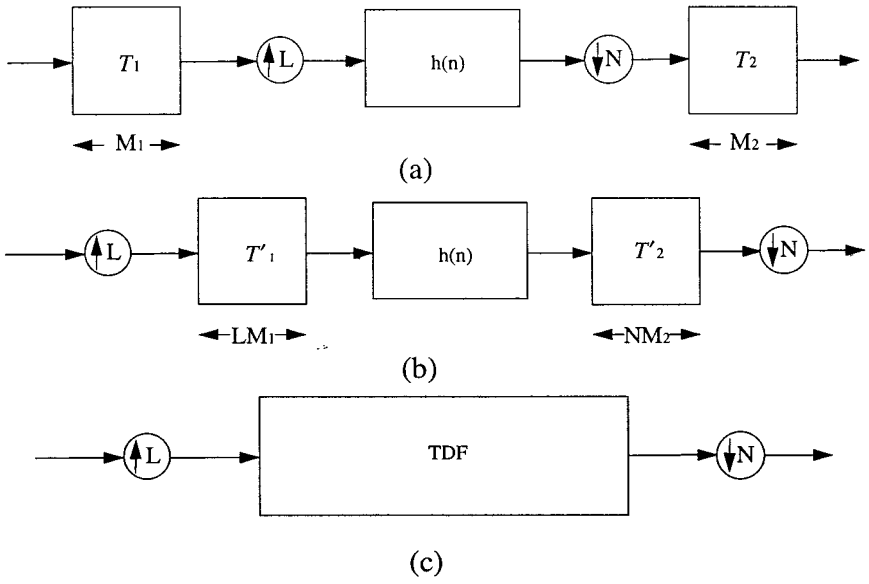


Fig. 3. Definition of MTDF (Case I). (a) System model. (b) Equivalent MTDF model. (c) Equivalent uniform TDF implementation.

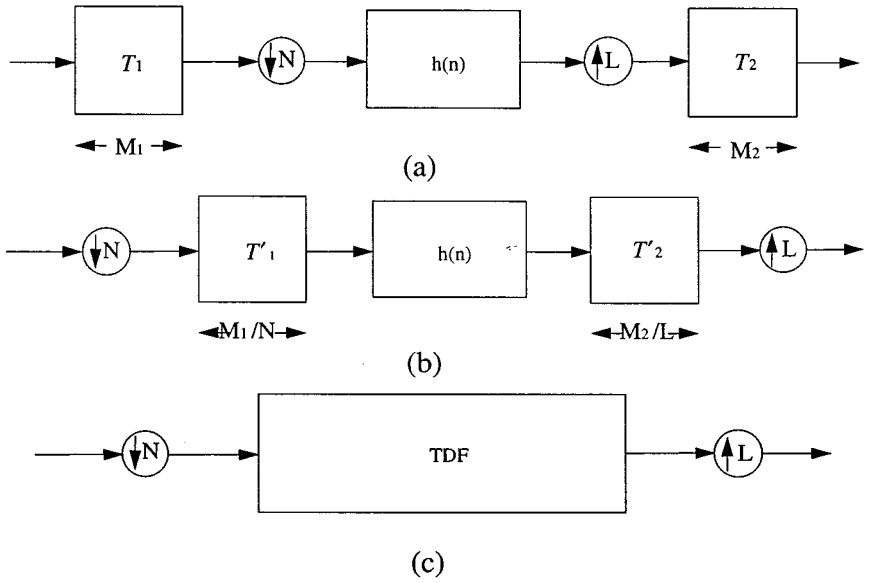


Fig. 4. Definition of MTDF (Case II). (a) System model. (b) Equivalent MTDF model. (c) Equivalent uniform TDF implementation.

where the interpolator precedes the decimator, and MTDF Case II the one where the decimator precedes the interpolator. The two possibilities of equal or unequal transform sizes ($M_1 \neq M_2$ and $M_1 = M_2$) are considered simultaneously.

A. MTDF Case I

We examine first the structure of MTDF Case I. The purpose of our derivation is to obtain the equivalent structure in the form of a uniform TDF. The first step is to exchange the decimator and the transform T_2 in the output. Using the definition of decimation (where $N-1$ out of N samples are discarded), we can easily see that we can exchange the decimator and the transform just by "upsampling" the matrix T_2 both horizontally and vertically. In the horizontal direction, the values of inserted components do not affect the system in any way, since their effect is eliminated by the decimation stage that immediately follows. In the vertical direction, inserted rows must have the value zero (an example is given below). Consequently, one such valid matrix can be obtained by setting all inserted elements to zero; the expanded matrix $\tilde{T}_2$ can then be expressed as

$$\tilde{T}_{2(M_2N\times M_2N)} = T_{2(M_2\times M_2)} \otimes J_{N\times N} \tag{13}$$

where

$$J_{N\times N} = \begin{pmatrix} 1 & 0 & . & . \\ 0 & 0 & . & . \\ . & . & 0 & . \\ . & . & . & . \end{pmatrix}.$$

In the second step, we exchange the interpolator and the transform T_1 in the input. Again, using the definition of interpolation (where $L-1$ zero-valued samples are inserted after each

original sample), we can see that we can exchange the interpolator and the transform just by upsampling the matrix T_1 in the horizontal and vertical directions. In this case, the role of inserted rows and columns is reversed. When upsampling horizontally, inserted values must be zero; when upsampling vertically, the precise value of inserted components becomes irrelevant. Hence, a valid choice is obtained by setting all inserted elements to zero, in which the expanded matrix $\tilde{T}_1$ can be expressed as

$$\tilde{T}_{1(M_1L\times M_1L)} = T_{1(M_1\times M_1)} \otimes J_{L\times L}. \tag{14}$$

We can finally combine the interpolation and decimation exchanges, as shown in Fig. 3(c). As we see, excluding the upsampling and interpolation stages, the end-system has exactly the structure of a nonuniform TDF. Summarizing, the procedure for the general solution is the following.

Step 1: Replace $M_1 \rightarrow LM_1$.
Step 2: Replace $M_2 \rightarrow NM_2$.
Step 3: Replace the transform stages by $\tilde{T}_1$ and $\tilde{T}_2$ as given above.

The nonuniform TDF problem can be converted to a uniform one by expanding to the least common multiple of LM_1 and NM_2. The equations describing the TDF components for the MTDF problem can then be written as follows:

$$H = [h_{ij}]_{(\mathrm{lcm}(LM_1,\,NM_2)+q-1)\times \mathrm{lcm}(LM_1,\,NM_2)} \tag{15}$$

$$h_{ij} = \begin{cases} h(i-j), & j \le i < j+q \\ & i = 0,\,1,\,\ldots,\,\mathrm{lcm}(LM_1,\,NM_2)+q-2 \\ & j = 0,\,1,\,\ldots,\,\mathrm{lcm}(LM_1,\,NM_2)-1 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$W_j = [w_{kl}]_{\mathrm{lcm}(LM_1,\,NM_2)\times(\mathrm{lcm}(LM_1,\,NM_2)+q-1)} \tag{17}$$

$$w_{kl} = \begin{cases} 1, & l = k + j\mathrm{lcm}(LM_1,\,NM_2) \\ & k = 0,\,1,\,\ldots,\,\mathrm{lcm}(LM_1,\,NM_2)-1 \\ & l = 0,\,1,\,\ldots,\,\mathrm{lcm}(LM_1,\,NM_2)+q-2 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

$$P \equiv T_2' W_j H T_1' \tag{19}$$

with

$$T_1' = I_{(\mathrm{lcm}(LM_1,\,NM_2)/LM_1)\times(\mathrm{lcm}(LM_1,\,NM_2)/LM_1)} \otimes \tilde{T}_1 \tag{20}$$

and

$$T_2' = I_{(\mathrm{lcm}(LM_1,\,NM_2)/NM_2)\times(\mathrm{lcm}(LM_1,\,NM_2)/NM_2)} \otimes \tilde{T}_2 \tag{21}$$

$$Z_i = \sum_{j=0}^{l} P_j X_{i-j} \tag{22}$$

$$l = \left\lfloor \frac{\mathrm{lcm}(LM_1,\,NM_2)+q-1}{\mathrm{lcm}(LM_1,\,NM_2)} \right\rfloor. \tag{23}$$

As shown in Fig. 3, the total structure is given by the combination of the interpolator, the TDF pipeline structure, and the decimator. As a result, it still has the merits of the conventional uniform TDF.

B. MTDF Case II

Let us now consider Case II, where the decimator precedes the interpolator. As in Case I, the purpose of our derivation is to obtain the equivalent structure in terms of conventional TDF. Here we will follow a reverse procedure: we will start from the equivalent TDF structure, and work our way back to the MTDF architecture. The reason is that, as we are going to see, the reduction is not always possible.

$$T_1 = \begin{pmatrix} t^{1'}_{0,0} & 0 & 0 & \cdots & t^{1'}_{0,1} & 0 & 0 & \cdots & t^{1'}_{0,2} & 0 & 0 & \cdots \\ \times & \times & \times & \cdots & \times & \times & \times & \cdots & \times & \times & \times & \cdots \\ \times & \times & \times & \cdots & \times & \times & \times & \cdots & \times & \times & \times & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\ t^{1'}_{1,0} & 0 & 0 & \cdots & t^{1'}_{1,1} & 0 & 0 & \cdots & t^{1'}_{1,2} & 0 & 0 & \cdots \\ \times & \times & \times & \cdots & \times & \times & \times & \cdots & \times & \times & \times & \cdots \\ \times & \times & \times & \cdots & \times & \times & \times & \cdots & \times & \times & \times & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \end{pmatrix}_{M_1\times M_1} \tag{24}$$

$$T_2 = \begin{pmatrix} t^{2'}_{0,0} & \times & \times & \cdots & t^{2'}_{0,1} & \times & \times & \cdots & t^{2'}_{0,2} & \times & \times & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\ t^{2'}_{1,0} & \times & \times & \cdots & t^{2'}_{1,1} & \times & \times & \cdots & t^{2'}_{1,2} & \times & \times & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \end{pmatrix}_{M_2\times M_2}. \tag{25}$$

First, we assume that we have a transform pair T_1', T_2' in MTDF Case II as shown in Fig. 4(b). We then find the transforms T_1 and T_2, as shown in Fig. 4(a), so that we have equivalent functionality to the one in Fig. 4(b). We assume that M_1/N and M_2/L in Fig. 4 are both positive integers. In the same way we exchanged the transform operations with the decimator and interpolator in Case I, we can similarly exchange them in this case as well. One can easily verify that the form of the exchanged ("upsampled") transform matrices will be given by (24) and (25), shown at the bottom of the previous page, where $t_{i,j}^{1'}$ and $t_{i,j}^{2'}$ are the elements of T_1' and T_2' respectively, and "$\times$" denotes "don't care" values

We see then that the exchange of the interpolator and the decimator is only possible if T_1 and T_2 have the specific structure in (24) and (25). This limits the applicability of Case II in practical situations, but we should note that a configuration with the interpolator (upsampling) as the last stage is not typical anyway.

Assuming that we are given an MTDF problem where the transform matrices satisfy the above requirements, we can convert it into an NTDF problem and subsequently to a uniform TDF one by following the steps detailed in Section II. The transform size of the equivalent uniform TDF will be

$$M = \text{lcm}(M_1/N,\, M_2/L). \tag{26}$$

The equations describing the various TDF components can be easily obtained from (1) to (7) after the matrices T_1' and T_2' are expanded according to (8) and (9).

C. 1-D TDRT Problem

As a practical application of MTDF, we examine a general solution to the TDRT problem. In this problem, we want to convert the rate (or resolution) of a signal that is provided in the transform domain. The output signal can have the same or different transform size, and it can even be represented in a different transform domain. TDRT is a natural concept for compressed images and video; in this case, the transform is DCT, and the transform sizes are typically the same (8) for both the input (inverse) and output (forward) transforms.

To resample a digital signal we perform two operations: low-pass filtering and sampling rate change. For an MTDF-based resolution translation system, and in order to avoid the transform structure limitations of Case II, we only consider a design that follows Case I. The resolution translation operation for the general case of fractional rate change involves upsampling and low-pass filtering, followed by downsampling. The filter represents the combination of the two interpolation and decimation filters (the ideal filter would have a cutoff at $\min\{\pi/L, \pi/N\}$). The MTDF system block diagram is identical to the one shown in Fig. 3. Let us consider, as a specific example, the case where we use an 8-point DCT transform, a decimation factor of two (no interpolator), and a 7-tap low-pass filter. The parameters of the TDF system shown in Fig. 3 can be expressed as follows:

$$T_1 = C^{-1}\; T_2 = C$$

where C is the 8×8 DCT matrix

$$N = 2,\; L = 1,\; q = 7$$

$$M_1 = 8,\; M_2 = 8,\; NM_2 = 16$$

$$\text{lcm}(LM_1,\, NM_2) = \text{lcm}(8,\, 16) = 16$$

$$H = [h_{ij}]_{22\times 16}$$

$$h_{ij} = \begin{cases} h(i-j), & j \le i < j+7 \\ & i = 0,\, 1,\, \cdots,\, 21 \\ & j = 0,\, 1,\, \cdots,\, 15 \\ 0, & \text{otherwise} \end{cases}$$

$$W_j = [w_{kl}]_{16\times 22}$$

$$w_{kl} = \begin{cases} 1, & l = k + j16 \\ & k = 0,\, 1,\, \cdots,\, 15 \\ & l = 0,\, 1,\, \cdots,\, 21 \\ 0, & \text{otherwise} \end{cases}$$

$$P_j = T_2' W_j H T_1'$$

$$T_1' = I_{2\times 2} \otimes (C^{-1} \otimes J_{1\times 1})$$

$$T_2' = I_{(1\times 1)} \otimes (C \otimes J_{2\times 2})$$

$$Z_i = \sum_{j=0}^{l} P_j X_{i-j}$$

$$l = \lfloor \frac{22}{16} \rfloor = 1.$$

IV. 2-D TDRT

In this section, we apply MTDF in order to find a solution for 2-D TDRT problem. We follow a restriction to generalization procedure. First we obtain the solution under some constraints, and later we relax the constraints for more generalized expression. The problem is shown in Fig. 5 and we consider only MTDF Case I as a potential solution. We restrict ourselves in the beginning in following assumptions with little loss of generality. First, sampling factors for down/upsampled images are same in x and y directions. This means the output images are scaled with same ratio in x and y directions. Second, the 2-D digital filter is separable ($f(m, n) = v_m h_n$) and its length is limited in $-r \le m, n \le r$ in which $r = \text{lcm}(LM_1, NM_2)$. This filter length is still not short, for at least the filter length of 17×17 is guaranteed at $r = 8$.

We first follow same procedures as in MTDF in the previous section. Fig. 5(a) shows upsampling and downsampling factors as L and N, by which both x and y directions are represented. Matrices in Fig. 5(b) can be represented by (13), (14), (20), and (21).

Then, we define a vertical and a horizontal filter in matrix forms. Note that we use now explicit expressions for filter matrices V and H, since the filter coefficient sets are given due to our limiting filter length in the second assumption as in (27) and (28), shown at the bottom of the page.

We partition V and H into $[V_1, V_2, V_3]$ and $[H_1, H_2, H_3]$, where

$$V_1 = \begin{pmatrix} v_{-r} & v_{1-r} & v_{2-r} & . & . & v_{-2} & v_{-1} \\ 0 & v_{-r} & v_{1-r} & . & . & v_{-3} & v_{-2} \\ 0 & 0 & v_{-r} & . & . & v_{-4} & v_{-3} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & . & . & . & 0 & v_{-r} \end{pmatrix} \tag{29}$$

$$V_2 = \begin{pmatrix} v_0 & v_1 & v_2 & . & . & v_{r-2} & v_{r-1} \\ v_{-1} & v_0 & v_1 & . & . & v_{r-3} & v_{r-2} \\ v_{-2} & v_{-1} & v_0 & . & . & v_{r-4} & v_{r-3} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ v_{1-r} & v_{2-r} & v_{3-r} & . & . & v_{-1} & v_0 \end{pmatrix} \tag{30}$$

and

$$V_3 = \begin{pmatrix} v_r & 0 & 0 & . & . & 0 & 0 \\ v_{r-1} & v_r & 0 & . & . & 0 & 0 \\ v_{r-2} & v_{r-1} & v_r & . & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & 0 & 0 \\ . & . & . & . & . & v_r & 0 \\ v_1 & v_2 & v_3 & . & . & v_{r-1} & v_r \end{pmatrix} \tag{31}$$

with similar definitions of H_1, H_2, and H_3.

Now, let x denote a spatial domain input block of size $3r \times 3r$, subdivided into nine $r \times r$ blocks as follows:

$$x = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}. \tag{32}$$

The $r \times r$ output block y_{22} that corresponds to the central input block x_{22} is given by

$$y_{22} = VxH^t \tag{33}$$

$$y_{22} = \sum_{i=1}^{3} \sum_{j=1}^{3} V_i x_{ij} H_j^t. \tag{34}$$

Since 2-D transforms are given by $Y_{22} = Ty_{22}T^t$, the second transform-domain values are $Y_{22} = T_2' y_{22} T_2'^t$

$$Y_{22} = \sum_{i=1}^{3} \sum_{j=1}^{3} T_2' V_i x_{ij} H_j^t T_2'^t. \tag{35}$$

If we define X_{ij} as the first transform-domain values, $x_{ij} = T_1' X_{ij} T_1'^t$. Thus, overall representation is given by

$$Y_{22} = \sum_{i=1}^{3} \sum_{j=1}^{3} T_2' V_i T_1' X_{ij} T_1'^t H_j^t T_2'^t. \tag{36}$$

If we define P_i^{col} and P_j^{row} by

$$P_i^{\text{col}} \equiv T_2' V_i T_1' \tag{37}$$

and

$$P_j^{\text{row}} \equiv T_1'^t H_j^t T_2'^t \tag{38}$$

$$V = \begin{pmatrix} v_{-r} & v_{1-r} & v_{2-r} & . & . & v_{-1} & v_0 & . & v_{r-1} & v_r & 0 & 0 & . & . & 0 & 0 \\ 0 & v_{-r} & v_{1-r} & . & . & v_{-2} & v_{-1} & . & v_{r-2} & v_{r-1} & v_r & 0 & . & . & 0 & 0 \\ 0 & 0 & v_{-r} & . & . & v_{-3} & v_{-2} & . & v_{r-3} & v_{r-2} & v_{r-1} & v_r & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & v_r & 0 \\ 0 & 0 & . & . & . & v_{-r} & v_{1-r} & . & v_0 & v_1 & v_2 & v_3 & . & . & v_{r-1} & v_r \end{pmatrix} \tag{27}$$

$$H = \begin{pmatrix} h_{-r} & h_{1-r} & h_{2-r} & . & . & h_{-1} & h_0 & . & h_{r-1} & h_r & 0 & 0 & . & . & 0 & 0 \\ 0 & h_{-r} & h_{1-r} & . & . & h_{-2} & h_{-1} & . & h_{r-2} & h_{r-1} & h_r & 0 & . & . & 0 & 0 \\ 0 & 0 & h_{-r} & . & . & h_{-3} & h_{-2} & . & h_{r-3} & h_{r-2} & h_{r-1} & h_r & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & h_r & 0 \\ 0 & 0 & . & . & . & h_{-r} & h_{1-r} & . & h_0 & h_1 & h_2 & h_3 & . & . & h_{r-1} & h_r \end{pmatrix}. \tag{28}$$

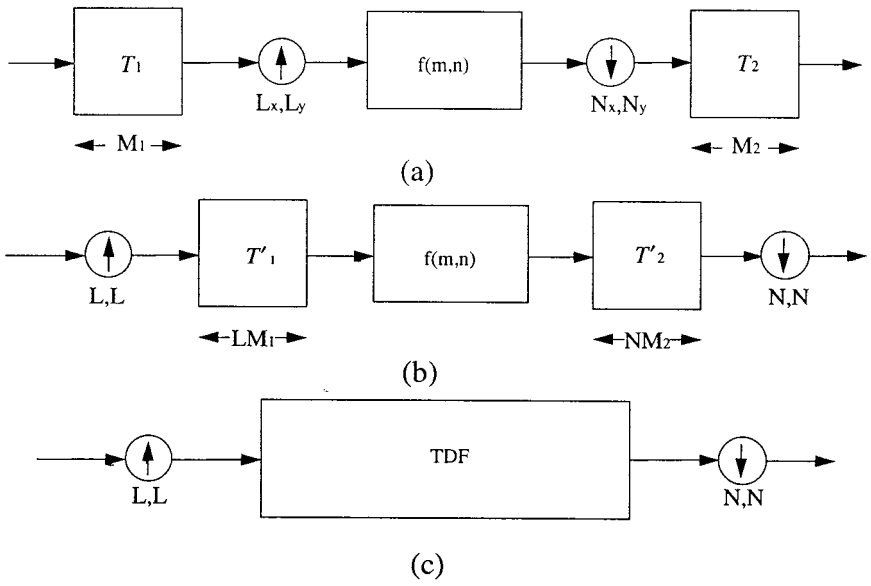


Fig. 5. Definition of 2-D MTDF. (a) System model. (b) Equivalent 2-D MTDF model. (c) Equivalent 2-D uniform TDF implementation.

then

$$Y_{22} = \sum_{i=1}^{3} \sum_{j=1}^{3} P_i^{\text{col}} X_{ij} P_j^{\text{row}}. \tag{39}$$

We now relax the filter-length constrains for more generalized expression. If the filter length is over $2r+1$, we can represent the vertical and horizontal filter matrices with more partitions, not just three. How many partions we can get just affects the upper limit of summations. For example, if we get NP as the number of partitions, above equation is rewritten by

$$Y_{(NP/2)+1,\,(NP/2)+1} = \sum_{i=1}^{NP} \sum_{j=1}^{NP} P_i^{\text{col}} X_{ij} P_j^{\text{row}}. \tag{40}$$

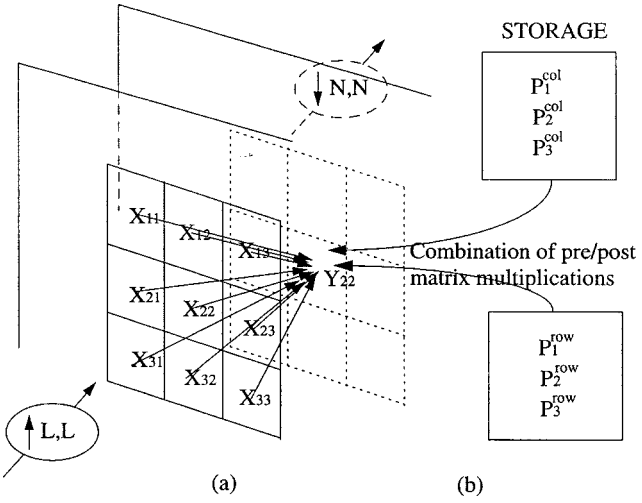


Fig. 6. Software/hardware pipelining structure of 2-D transform-domain translation. (a) Structure. (b) Sub-routine/sub-processor.

A. 2-D TDRT Example

Recently, a direct DCT domain implementation technique for image resizing was presented in [14], [15]. We have driven that 2-D MTDF can be used a core part of DCT domain image resizing issue, which is called 2-D TDRT in this paper. The resolution translation operation for the general case of fractional rate change involves upsampling and low-pass filtering, followed by downsampling. The filter represents the combination of the two interpolation and decimation filters (the ideal filter would have a cutoff at $\min\{\pi/L, \pi/N\}$). The 2-D MTDF system block diagram is identical to the one shown in Fig. 5. Let us consider, as a specific example, the case where we use an 8-point DCT transform, a decimation factor of two (no interpolator), and a 32-tap low-pass filter. The parameters of the TDF system shown in Fig. 5 can be expressed as follows:

$$T_1 = C^{-1}\ T_2 = C$$

where C is the 8×8 DCT matrix

$$N = 2,\ L = 1,\ q = 33$$

$$M_1 = 8,\ M_2 = 8,\ NM_2 = 16$$

$$\text{lcm}(LM_1,\ NM_2) = \text{lcm}(8,\ 16) = 16.$$

The pipeline matrices for columns and rows are

$$P_i^{\text{col}} \equiv T_2' V_i T_1'$$
$$P_j^{\text{row}} \equiv T_1'^{t} H_j^{t} T_2'^{t}$$

where

$$T_1' = I_{2\times 2} \otimes (C^{-1} \otimes J_{1\times 1})$$
$$T_2' = I_{(1\times 1)} \otimes (C \otimes J_{2\times 2})$$

with the same definition of V_i and H_j in (31)–(35) at $r = 16$.

Then

$$\begin{aligned} Y_{22} &= \sum_{i=1}^{3}\sum_{j=1}^{3} P_i^{\text{col}} X_{ij} P_j^{\text{row}} \\ &= P_1^{\text{col}} X_{11} P_1^{\text{row}} + P_1^{\text{col}} X_{12} P_2^{\text{row}} + P_1^{\text{col}} X_{13} P_3^{\text{row}} \\ &\quad + P_2^{\text{col}} X_{21} P_1^{\text{row}} + P_2^{\text{col}} X_{22} P_2^{\text{row}} + P_2^{\text{col}} X_{23} P_3^{\text{row}} \\ &\quad + P_1^{\text{col}} X_{31} P_1^{\text{row}} + P_3^{\text{col}} X_{32} P_2^{\text{row}} + P_3^{\text{col}} X_{33} P_3^{\text{row}}. \end{aligned}$$

Fig. 6 shows the proposed hardware/software structure of 2-D TDRT. We believe that this example shows most practical cases; even though we provide a generalized expression in (40), we usually do not use very high-order filters in filtering.

V. Concluding Remarks

In this paper, we defined NTDF, which is a generalized version of conventional TDF in terms of allowing differing input and output transform sizes (as well as different types). We have analyzed the structure of NTDF systems and shown how they can be converted to an equivalent TDF one, for which the solution is readily available. We also showed how to extend NTDF to MTDF, where the rates of the input and output signals are different, and follow a rational proportionality relationship (fractional rate change). Here we distinguished two different cases, depending on if the interpolator precedes the decimator (Case I) or vice versa (Case II). We showed that Case I can be converted to an NTDF problem, while for Case II this is possible if and only if the transform matrices have a particular structure. We should note that, for both NTDF and MTDF, extensions to multiple dimensions are trivial, as long as the transform operation is a separable one.

As practical examples of the utility of NTDF and MTDF, we showed how they can be used to provide general solutions to the sub-adjacent block problem as well as the 1-D/2-D TDRT (conversion) problem. These solutions sidestep most of the limitations of existing approaches in terms of allowable transform type or filter structure [19] and generalize the issues in [20], [14], and [15] to a combined form of transform domain FBDS and FAUS cases. These two examples are strongly connected with applications involving coded images and video, since transform coding is a core component of most of the standard compression schemes (JPEG, MPEG-1, MPEG-2, H.261, H.263).

We have shown that the fundamental expression in both NTDF and MTDF is that of the matrix-vector product, leading to various advantages for a TDF hardware/software implementation. After the filter coefficients are determined, we can precalculate the matrix coefficient blocks that appear in the TDF analyses; thus, all potential arithmetic (finite precision) errors disappear, except for the matrix block multiplication. Note that the same accuracy remains after the original data is shifted in the course of pipelining. That is, the pipelining structure itself guarantees a small and uniformly distributed arithmetic error, which is only due to the individual matrix-vector multiplications.

These results directly generalize those reported in [12]: NTDF eliminates the limitation of regular TDF in terms of allowing different transform sizes, whereas MTDF eliminates the limitation of NTDF by allowing different input and output signal rates. These two extensions allow the TDF architecture to be applied to a large variety of relevant applications, such as 2-D image processing and compression.

Acknowledgment

The authors wish to acknowledge two anonymous reviewers for their useful comments, and to thank the Associate Editor, Prof. R. Stevenson, for his kind suggestion.

References

[1] K. R. Rao and J. J. Hwang, *Techniques and Standards for Digital Iimage/Video/Audio Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[2] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90–93, Jan. 1974.

[3] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. New York: Springer-Verlag, 1975.

[4] W. H. Chen and C. H. Smith, "Adaptive coding of monochrome and color images," *IEEE Trans. Commun.*, vol. COM-25, pp. 1285–1292, Nov. 1977.

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

[6] ——, "A new approach to decoding and compositing motion compensated DCT-based images," in *Proc. IEEE ICASSP '93*, Minneapolis, MN, Apr. 1993, pp. V421–V424.

[7] A. Eleftheriadis and D. Anastassiou, "Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video," in *Proc. 5th Int. Workshop Network and Operating System Support for Digital Audio and Video*, Durham, NH, Apr. 1995, pp. 95–106.

[8] ——, "Constrained and general dynamic rate shaping of compressed digital video," *Proc. 2nd IEEE Int. Conf. Image Processing*, pp. III.396–399, Oct. 1995.

[9] B. Chitprasert and K. R. Rao, "Discrete cosine transform filtering," *IEEE Trans. Signal Processing*, vol. 19, pp. 233–245, 1990.

[10] W. H. Chen and S. C. Fralic, "Image enhancement using cosine transform filtering," presented at the Image, Science, and Mathematics Symp., Monterey, CA, Nov. 1976.

[11] S. A. Martucci, "Symmetric convolution and discrete sine and cosine transforms," *IEEE Trans. Signal Processing*, vol. 42, pp. 1038–1051, May 1994.

[12] J. B. Lee and B. G. Lee, "Transform domain filtering based on pipelining structure," *IEEE Trans. Signal Processing*, vol. 40, pp. 2061–2064, Aug. 1992.

[13] D. Kim and B. G. Lee, "Transform domain IIR filtering," *IEEE Trans. Signal Processing*, vol. 43, pp. 2431–2434, Oct. 1995.

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

[15] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image downsampling and for inverse motion compensation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 408–476, June 1997.

[16] S. A. Martucci, "Image resizing in the discrete cosine transform domain," *Proc. IEEE ICIP'95*, vol. 3, no. 2, pp. 244–247, 1995.

[17] T. Fjallbrant, "A wide-band approach to adaptive transform coding of speech signals a tms 320 signal processor implementation," in *Proc. IEEE ISCAP-85*, Kyoto, Japan, 1985, pp. 312–324.

[18] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

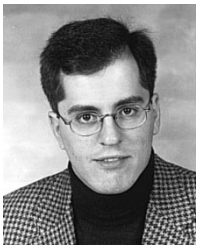
[19] W. Kou and T. Fjallbrant, "A direct computation of dct coefficients for a signal block taken from two adjacent blocks," *IEEE Trans. Signal Processing*, vol. 39, pp. 1692–1695, July 1991.

[20] ——, "Fast computation of transform coefficients for a subadjacent block for a transform family," *IEEE Trans. Signal Processing*, vol. 39, pp. 1695–1699, July 1991.

[21] W. Kou and Z. Hu, "Several methods of constructing discrete orthogonal transforms," *Acta Electron. Sinica*, vol. 14, pp. 95–102, May 1986.

Jae-Beom Lee (M'99) was born in Pusan, Korea, in 1965. He received the B.S. degree in electronics engineering from Yonsei University, Seoul, Korea, in 1988, and the M.S.E. degree from Seoul National University, Seoul, Korea, and Columbia University, NY, in 1990 and 1996, respectively. He is currently pursuing the Ph.D. degree in the Department of Electrical Engineering, Columbia University, where he is a Graduate Research Assistant with the ADVENT Group.

He has been a Senior Engineer, working on video processing/compression at the C-Cube/DiviCom, NY (NYDC), a DiviCom unit of C-Cube Microsystems, since July 1999, where digital video networking/systems issues are focused. In 1998, he was a Summer Technical Employee with the Video Compression Systems Lab, Samoff Corporation, working on MPEG-2 compressed domain logo Insertion. In 1990–1991, he w as a Member of Technical Staff with the Advanced Television Group, Samsung Electronics R&D Center, Suwan, Korea, working on subband-based HDTV proposals. His research interests include MPEG-2/MPEG-4 video and very low bit-rate video-coding technology and its systems issues, 3-D depth camera applications to very low bit-rate video coding, and compressed-domain video manipulation with emphasis on transcoding.



Alexandros Eleftheriadis (S'88–M'95) was born in Athens, Greece, in 1967. He received the Diploma in electrical engineering and computer science from the National Technical University of Athens, Athens, Greece, in 1990, and the M.S., M.Phil., and Ph.D. degrees in electrical engineering from Columbia University, New York, in 1992, 1994, and 1995, respectively.

Since 1995, he has been on the faculty of the Department of Electrical Engineering, Columbia University, currently as an Associate Professor, where he is leading a research team working on media representation, with emphasis on multimedia software, video signal processing and compression, video communication systems (including video-on-demand and Internet video), and the mathematical fundamentals of compression. He is also Co-Principal Investigator of the ADVENT Project (http: //www.ee.columbia.edu/advent), an industrial affiliates program at Columbia University that is performing research on all aspects of digital video representation, communication, and description. He is a member of the ANSI NCITS L3.1 Committee and the ISO-IEC JTC I /SC29/WG I I (MPEG) Group, who develop national and international standards for audio and video coding. He has served as the Editor of the MPEG-4 Systems (ISO/IEC 14 496-1) specification and has authored more than 60 publications in international journals and conferences and holds six patents (11 pending).

Dr. Eleftheriadis is on the Editorial Board of the *Multimedia Tools and Applications Journal*, and has served as a Guest Editor, Committee Member, and Organizer for several international journals and conferences. He is a member of the ACM, the AAAS, and the Technical Chamber of Greece, and the recipient of a National Science Foundation CAREER Award.