2-D Transform Domain Resolution Translation

Jae-Beom Lee and Alexandros Eleftheriadis

Department of Electrical Engineering
and Columbia New Media Technology Center
Columbia University
New York, NY 10027, USA
{jbl,eleft}@ee.columbia.edu

March 9, 1998

Abstract

The extensive use of discrete transforms such as the Discrete Cosine Transform in image and video coding suggests the investigation on filtering before down sampling (FBDS) and filtering after up sampling (FAUS) methods directly acting on the transform domain [1, 2]. On the other hand, Transform Domain Filtering (TDF) was recently introduced as an important tool for implementing linear filtering and other linear operators directly in the compressed domain [3]. In this paper, we introduce the concept of "transform domain resolution translation" as a combined form of the transform domain FBDS and FAUS issues, and then propose the solution with the context of TDF. We, first, generalize the TDF to include non-uniform and multirate cases. 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 pipelined structures that involve matrix-vector product blocks and vector addition, but is not limited to only hardware eventhough the proposed architecture is easily realizable with distributed arithmetic designs and minimizes arithmetic errors due to the small number of processing stages involved. Chang and Messerschmitt showed that TDF is still useful in the form of software architecture since transform data usually take on immensively compressed form [4]. We show step by step examples of how these generalized notions of TDF can provide general solutions to the sub-adjacent block problem, to the 1-D transform domain resolution translation problem and, finally, to the 2-D transform domain resolution translation problem. 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).

1 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 [5, 6, 7], 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 Mbps) 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 assymetric 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 [4, 8] in order to provide fast video compositing algorithms, whereas in [9, 10] it has been used to perform rate changes.

With respect to transform domain filtering, a number of techniques have been applied to provide convolution-multiplication properties to the DCT. Such "DCT filtering" approaches successfully reduce the number of multiplications and additions, but also possess some limitations and imperfections. For example, the scheme in [11] relies on a distortion factor $w(n)$ which is difficult to implement, while the scheme in [12] 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 and cannot also avoid block edge effects.

More recently, an interesting idea was reported by Martucci in [13]. He derived a complete set of symmetrical convolution routines for a family of discrete trigonometric transforms including DCT. His method can be modified to obtain linear convolution algorithms by appropriate zero padding in the convolution domain. In our opinion, unfortunately, these algorithms cannot be used in most of practical applications since the DCT domain data are already given without prior zero padding in the spatial domain.

In recent work [3], we proposed the concept of transform domain filtering (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 [3] that the existing DCT filtering approach can be generalized to TDF, and a pipelining structure was presented as a means to implement it. The TDF scheme overcomes the limitations in filter coefficients and is free from imperfections in the reconstructed data. This filtering scheme completely avoids the block edge effect caused by circular convolution since it employs linear convolution. In addition, the two transforms, T_1 and T_2 of the input and output respectively, need not be of the same kind. That is, T_1 could be the inverse DCT (IDCT) while T_2 could be DFT, WHT, or others, thus directly providing a representation of the filtered signal in a different transform domain. Applications for such conversions include, for example, a DCT-based edge detector where T_1 is IDCT and T_2 is the Haar transform. Note also that it is possible that T_1 is a forward transform, while T_2 is an inverse one, and hence the input and output data are in the spatial (or time) domain. Thus this approach can even be used for operations which are more naturally performed in the transform domain. Moreover, recent works showed that TDF is still useful in software architecture since transform data usually take on immensively compressed form such as DCT [4, 8]. The compression ratio is achieved to 50-100 to 1 in practical DCT-based codec, and hence a small proportion of transform data needs computation in software architecture.

The original TDF approach in [3], however, is only applicable when the sizes of the transforms T_1 and T_2 are the same. And it was not designed for multirate environments. In this paper, we extend recent TDF into the notion of general transform size and multirate processing. This paper also addresses the sub-adjacent block problem, 1-D transform domain resolution translation, and 2-D transform domain resolution translation as a combined form of transform domain FBDS and FAUS issues for which we provide the solution in the form of TDF; However, this paper doesn't mention issues of TDF itself, since they are examined in detail in recent papers [3, 4, 8]. In Section 2 we extend this approach 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 non-uniform 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 [14] . Section 3 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 non-uniform one. An example is given by providing a general solution to the 1-D transform domain resolution translation problem. And then, we generalize it in the next section to 2-D transform domain resolution translation problem, which is actually a 2-D version of generalized transform domain FBDS and FAUS issues given in [1, 2]. Finally, in Section 5 we discuss implementation considerations and present some concluding remarks.

2 Uniform and Non-Uniform Transform Domain Filtering

The Uniform TDF

We define TDF as an operation which has the same functionality as a combination of transform, filtering, and transform [3]. The TDF problem for the case where the input and output transform sizes are equal is depicted in Figure 1 (a) [3]. In this paper, this will be referred to as uniform TDF (UTDF), 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 Figure 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

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

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:

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

resulting to an output from the vector adder:

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

where

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

Non-Uniform TDF

We define Non-Uniform Transform Domain Filtering (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 Figure 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

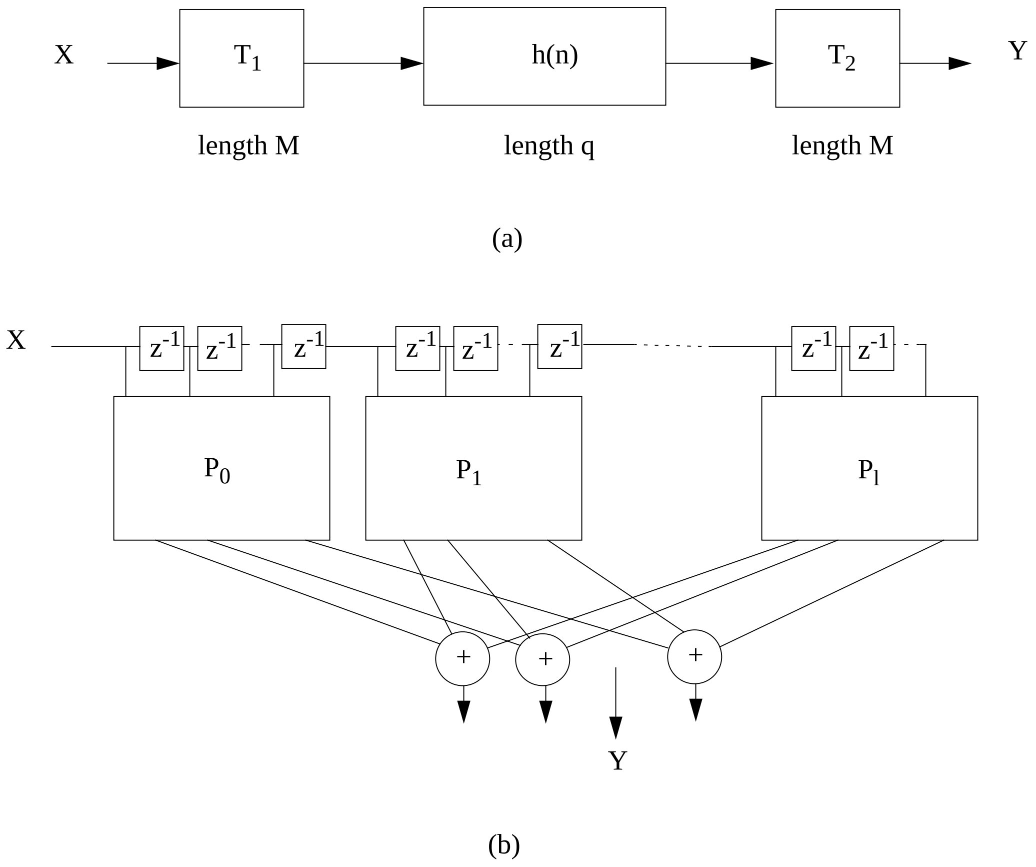

Figure 1: Single-stage Uniform TDF structure: (a) system model, (b) pipelined implementation architecture.

in Section 1. We can map the non-uniform problem to a uniform one by extending the transform matrices. In particular, let

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

and

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

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 $\mathrm{lcm}(M_1, M_2)$ where $\mathrm{lcm}(\cdot)$ 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

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

the various parameters of the system are given only if $M = \mathrm{lcm}(M_1, M_2)$ by Eqs. (1) through (7). The equivalent system model is shown in Figure 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.

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

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.

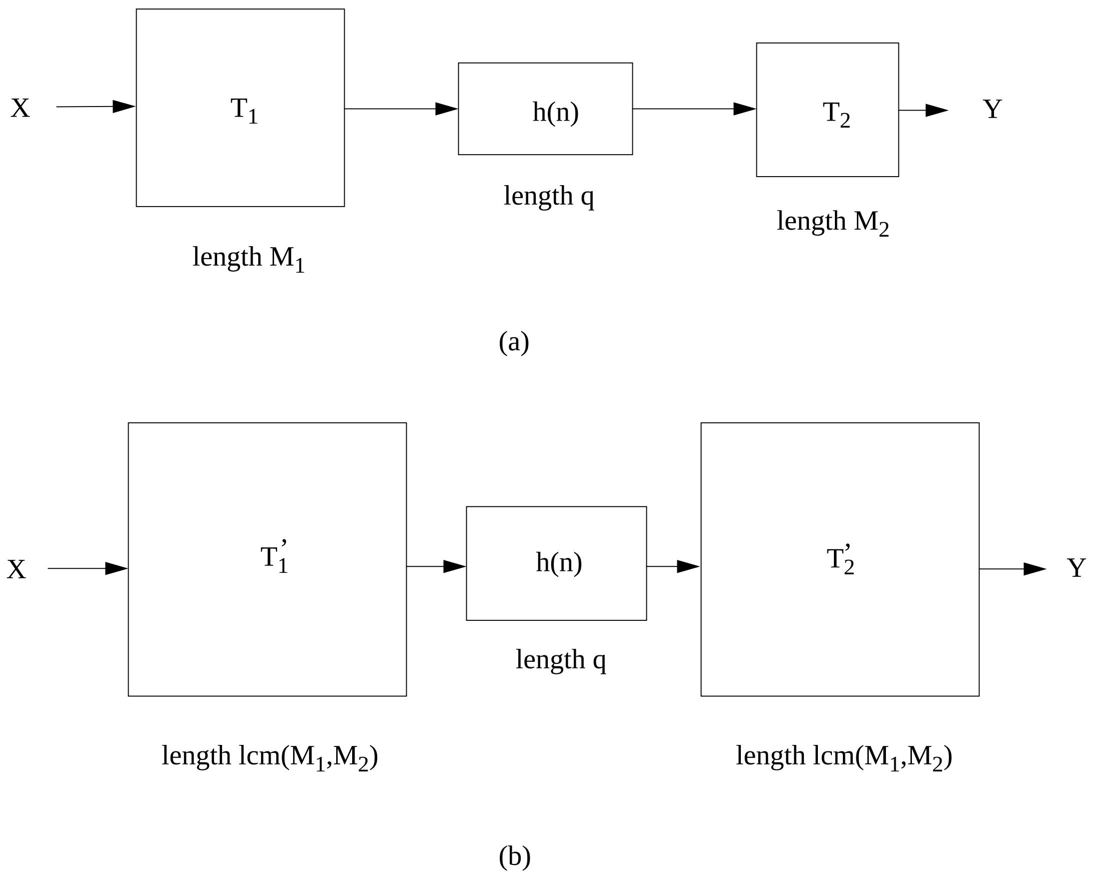

Figure 2: Single-stage Non-Uniform TDF structure: (a) system model, (b) equivalent uniform TDF model.

The Sub-Adjacent Block Problem

As an example of the use of the non-uniform TDF 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 two-dimensional case, such a block overlaps with up to 4 adjacent blocks. In this example, we consider only one dimension for simplicity, and hence overlap will occur with only two blocks; generalization to two dimension is straight forward. 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 [15, 16]. The proposed solution in [15] 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 4 is allowed between the input signal and output signals in the case of an 8-point transform. The proposed method in [16] 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 [17]. 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 non-uniform block size. This is exactly the NTDF structure, and hence can be directly implemented using Eqs. (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 4, the Haar transform block size is 8, and a new block is formed by a shift of 2 samples. This example can be thought of DCT-domain edge detector. A shift of 2 samples implies a second order pure delay filter.

$$M_1 = 4 \quad M_2 = 8 \quad M = \mathrm{lcm}(4, 8) = 8$$

$$\frac{\mathrm{lcm}(M_1, M_2)}{M_1} = 2 \qquad \frac{\mathrm{lcm}(M_1, M_2)}{M_2} = 1$$

$$T'_{\mathrm{IDCT}} = I_{2\times 2} \otimes T_{\mathrm{IDCT}},$$

$$T'_{\mathrm{Haar}} = I_{1\times 1} \otimes T_{\mathrm{Haar}}$$

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

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

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

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

$$P_j \equiv T_2' W_j H T_1'$$

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

$$l = \left\lfloor \frac{10}{8} \right\rfloor = 1$$

The block diagram structure is exactly the same as Figure 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 above expression is more convenient, but we use partitioned matrices expression in Section 4 since the filter coefficient set is fixed.

3 Multirate Transform Domain Filtering

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 Figures 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 UTDF structures, we divide the definition into two cases, as shown in Figures 3 and 4. We define MTDF Case I as the one 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.

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 0 (an example is given below). Consequently, one such valid matrix can be obtained by setting

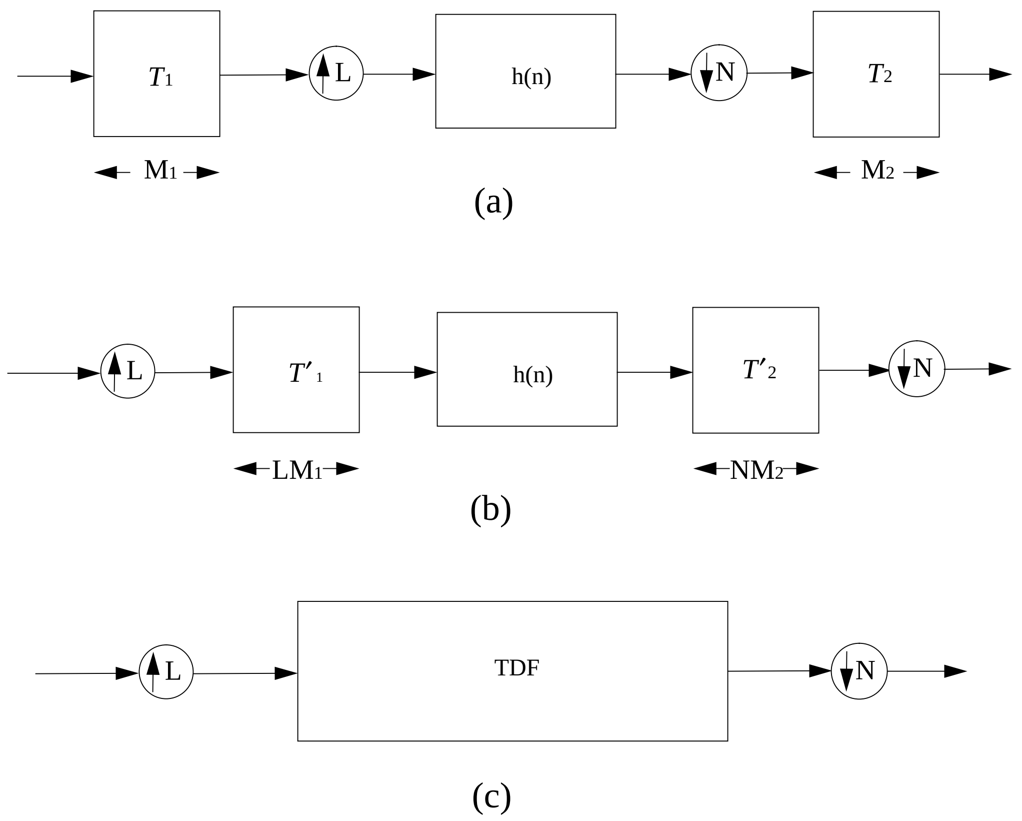

Figure 3: Definition of MTDF (Case I): (a) system model, (b) equivalent MTDF model, (c) equivalent uniform TDF implementation.

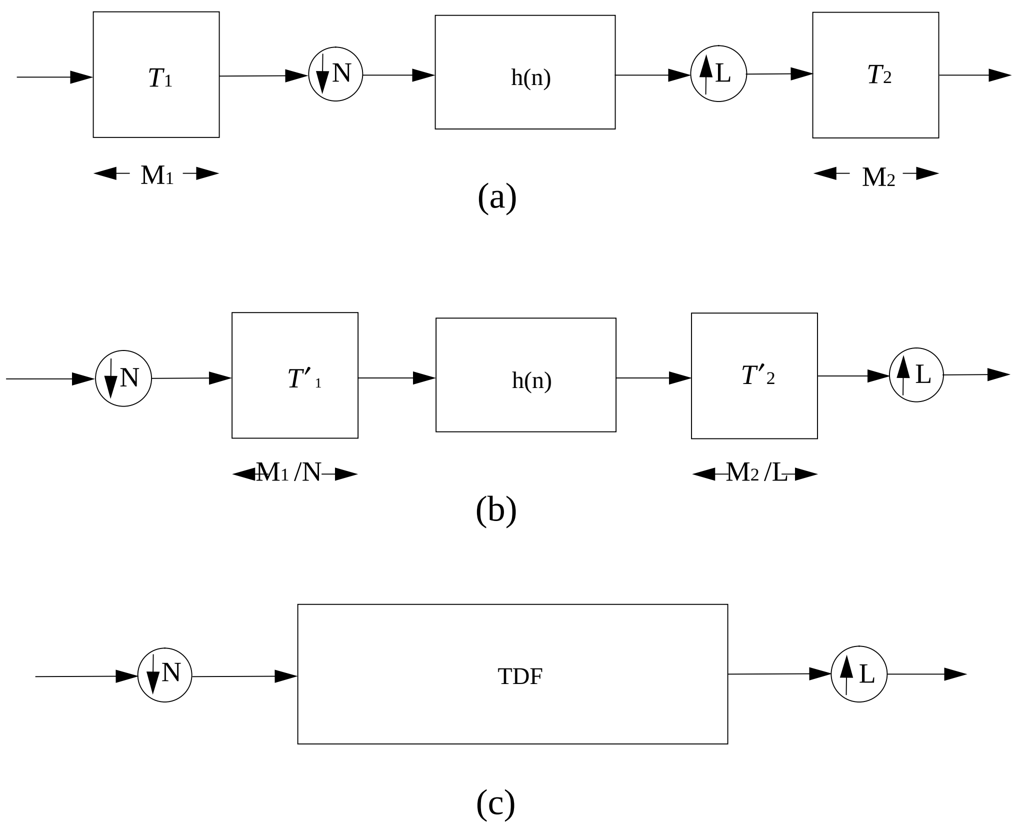

Figure 4: Definition of MTDF (Case II): (a) system model, (b) equivalent MTDF model, (c) equivalent uniform TDF implementation.

all inserted elements to 0; 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 0; when upsampling vertically, the precise value of inserted components becomes irrelevant. Hence a valid choice is obtained by setting all inserted elements to 0, 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 Figure 3 (c). As we see, excluding the upsampling and interpolation stages, the end-system has exactly the structure of a non-uniform TDF. Summarizing, the procedure for the general solution is:

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 non-uniform 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:

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

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

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

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

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

with

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

and

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

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

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

As shown in Figure 3, the total stucture 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.

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.

First, we assume that we have a transform pair T_1', T_2' in MTDF Case II as shown in Figure 4 (b). We then find the transforms T_1 and T_2 as shown in Figure 4 (a), so that we have equivalent functionality to the one in Figure 4 (b). We assume that M_1/N and M_2/L in Figure 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:

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

and

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

where $t^{1'}_{i,j}$ and $t^{2'}_{i,j}$ 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 shown in Eqs. (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 2. 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 Eqs. (1) through (7), after the matrices T'_1 and T'_2 are expanded according to Eqs. (8) and (9).

1-D Transform Domain Resolution Translation Problem

As a practical application of MTDF, we examine a general solution to the transform domain resolution translation 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. Transform domain resolution translation 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: lowpass 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 Figure 3. Let us consider, as a specific example, the case where we use an 8-point DCT transform, a decimation factor of 2 (no interpolator), and a 7-tap lowpass filter. The parameters of the TDF system shown in Figure 3 can be expressed as follows.

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

where C is the 8×8 DCT matrix,

$$N = 2 \quad L = 1 \quad q = 7$$

$$M_1 = 8 \quad M_2 = 8 \quad NM_2 = 16.$$

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

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

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

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

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

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

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

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

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

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

4 2-D Transform Domain Resolution Translation

In this section, we apply MTDF in order to find a solution for 2-D transform domain resolution translaion 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 Figures 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

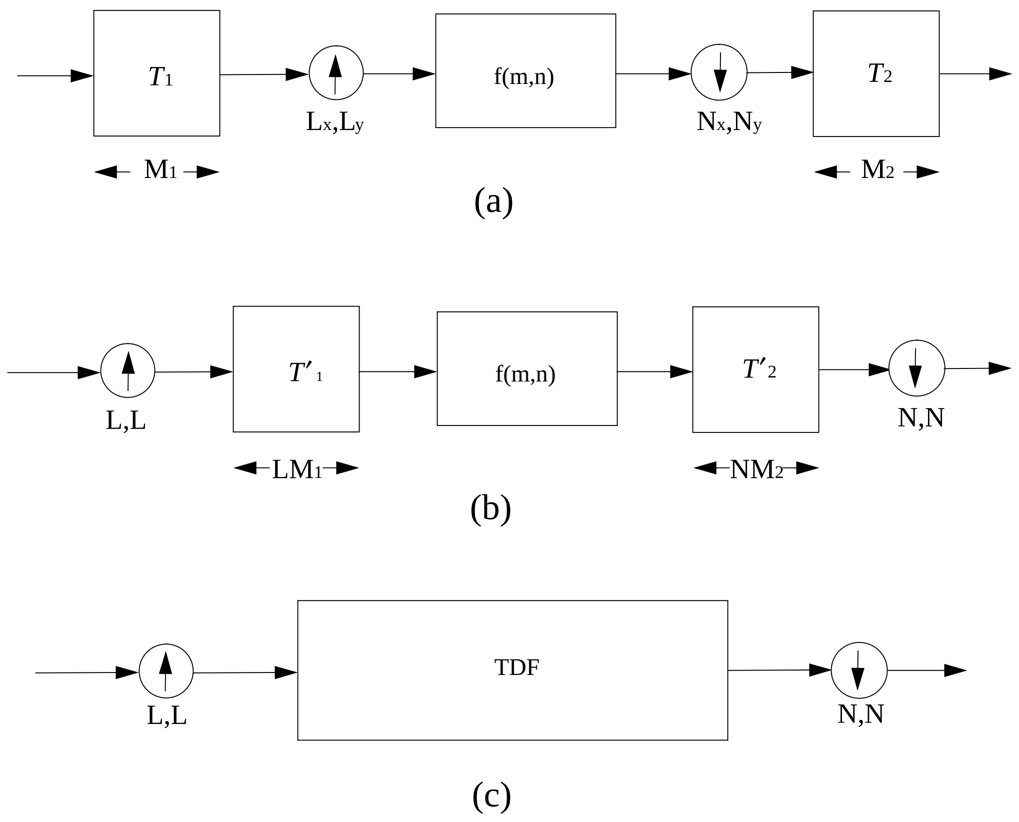

Figure 5: Definition of 2-D MTDF : (a) system model, (b) equivalent 2-D MTDF model, (c) equivalent 2-D uniform TDF implementation.

factors for down/up sampled 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 $-r \leq m,n \leq r$ in which $r = lcm(LM_1, NM_2)$. 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. Figure 5 (a) shows up and down sampling factors as L and N by which both x and y directions are represented. Matrices in Figure 5 (b) can be represented by:

$$\tilde{T}_{2(M_2N \times M_2N)} = T_{2(M_2 \times M_2)} \otimes J_{N \times N} \tag{27}$$

$$\tilde{T}_{1(M_1L \times M_1L)} = T_{1(M_1 \times M_1)} \otimes J_{L \times L}, \tag{28}$$

$$T_1' = I_{\frac{\mathrm{lcm}(LM_1,NM_2)}{LM_1} \times \frac{\mathrm{lcm}(LM_1,NM_2)}{LM_1}} \otimes \tilde{T}_1 \tag{29}$$

and

$$T_2' = I_{\frac{\mathrm{lcm}(LM_1,NM_2)}{NM_2} \times \frac{\mathrm{lcm}(LM_1,NM_2)}{NM_2}} \otimes \tilde{T}_2 \tag{30}$$

Then, we define a virtical 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 limitting filter length in the second assumption.

$$V = \begin{pmatrix} v_{-r} & v_{1-r} & v_{2-r} & . & . & v_{-1} & v_0 & . & v_{r-1} & v_r & 0 & 0 & . & . & 0 & 0 \\ 0 & v_{-r} & v_{1-r} & . & . & v_{-2} & v_{-1} & . & v_{r-2} & v_{r-1} & v_r & 0 & . & . & 0 & 0 \\ 0 & 0 & v_{-r} & . & . & v_{-3} & v_{-2} & . & v_{r-3} & v_{r-2} & v_{r-1} & v_r & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & v_r & 0 \\ 0 & 0 & . & . & . & v_{-r} & v_{1-r} & . & v_0 & v_1 & v_2 & v_3 & . & . & v_{r-1} & v_r \end{pmatrix} \tag{31}$$

and

$$H = \begin{pmatrix} h_{-r} & h_{1-r} & h_{2-r} & . & . & h_{-1} & h_0 & . & h_{r-1} & h_r & 0 & 0 & . & . & 0 & 0 \\ 0 & h_{-r} & h_{1-r} & . & . & h_{-2} & h_{-1} & . & h_{r-2} & h_{r-1} & h_r & 0 & . & . & 0 & 0 \\ 0 & 0 & h_{-r} & . & . & h_{-3} & h_{-2} & . & h_{r-3} & h_{r-2} & h_{r-1} & h_r & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . & . & . & h_r & 0 \\ 0 & 0 & . & . & . & h_{-r} & h_{1-r} & . & h_0 & h_1 & h_2 & h_3 & . & . & h_{r-1} & h_r \end{pmatrix} \tag{32}$$

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

$$V_1 = \begin{pmatrix} v_{-r} & v_{1-r} & v_{2-r} & . & . & v_{-2} & v_{-1} \\ 0 & v_{-r} & v_{1-r} & . & . & v_{-3} & v_{-2} \\ 0 & 0 & v_{-r} & . & . & v_{-4} & v_{-3} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & . & . & . & 0 & v_{-r} \end{pmatrix} \tag{33}$$

$$V_2 = \begin{pmatrix} v_0 & v_1 & v_2 & . & . & v_{r-2} & v_{r-1} \\ v_{-1} & v_0 & v_1 & . & . & v_{r-3} & v_{r-2} \\ v_{-2} & v_{-1} & v_0 & . & . & v_{r-4} & v_{r-3} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ v_{1-r} & v_{2-r} & v_{3-r} & . & . & v_{-1} & v_0 \end{pmatrix} \tag{34}$$

and

$$V_3 = \begin{pmatrix} v_r & 0 & 0 & . & . & 0 & 0 \\ v_{r-1} & v_r & 0 & . & . & 0 & 0 \\ v_{r-2} & v_{r-1} & v_r & . & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & 0 & 0 \\ . & . & . & . & . & v_r & 0 \\ v_1 & v_2 & v_3 & . & . & v_{r-1} & v_r \end{pmatrix} \tag{35}$$

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:

$$x = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \tag{36}$$

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

$$y_{22} = VxH^t. \tag{37}$$

$$y_{22} = \sum_{i=1}^{3}\sum_{j=1}^{3} V_i x_{ij} H_j^t \tag{38}$$

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

$$Y_{22} = \sum_{i=1}^{3}\sum_{j=1}^{3} T_2' V_i x_{ij} H_j^t T_2'^t \tag{39}$$

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:

$$Y_{22} = \sum_{i=1}^{3}\sum_{j=1}^{3} T_2' V_i T_1' X_{ij} T_1'^t H_j^t T_2'^t \tag{40}$$

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

$$P_i^{col} \equiv T_2' V_i T_1' \tag{41}$$

and

$$P_j^{row} \equiv T_1'^t H_j^t T_2'^t, \tag{42}$$

then

$$Y_{22} = \sum_{i=1}^{3}\sum_{j=1}^{3} P_i^{col} X_{ij} P_j^{row} \tag{43}$$

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 3. 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 :

$$Y_{\frac{NP}{2}+1,\frac{NP}{2}+1} = \sum_{i=1}^{NP}\sum_{j=1}^{NP} P_i^{col} X_{ij} P_j^{row} \tag{44}$$

2-D Transform Domain Resolution Translation Example

Recently, a direct DCT domain implementation technique for image resizing was presented in [1, 2]. We have driven that 2-D MTDF can be used a core part of DCT domain image resizing issue, which is called 2-D transform domain resolution translation 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 Figure 5. Let us consider, as a specific example, the case where we use an 8-point DCT transform, a decimation factor of 2 (no interpolator), and a 32-tap lowpass filter. The parameters of the TDF system shown in Figure 5 can be expressed as follows.

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

where C is the 8×8 DCT matrix,

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

$$M_1 = 8 \quad M_2 = 8 \quad NM_2 = 16.$$

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

The pipeline matrices for columns and rows are

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

$$P_j^{row} \equiv T_1'^t H_j^t T_2'^t,$$

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,

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

$$= P_1^{col} X_{11} P_1^{row} + P_1^{col} X_{12} P_2^{row} + P_1^{col} X_{13} P_3^{row}$$

$$+ P_2^{col} X_{21} P_1^{row} + P_2^{col} X_{22} P_2^{row} + P_2^{col} X_{23} P_3^{row}$$

$$+ P_1^{col} X_{31} P_1^{row} + P_3^{col} X_{32} P_2^{row} + P_3^{col} X_{33} P_3^{row}$$

Figure 6 shows the proposed hardware/software structure of 2-D transform domain resolution translation. We believe that this example shows most practical cases; Eventhough we provide a generalized expression in (44), we usually don't use very high order filters in filtering.

5 Concluding Remarks

In this paper, we defined non-uniform transform domain filtering (NTDF), which is a generalized version of conventional TDF in terms of allowing differing input and output transform sizes (as well

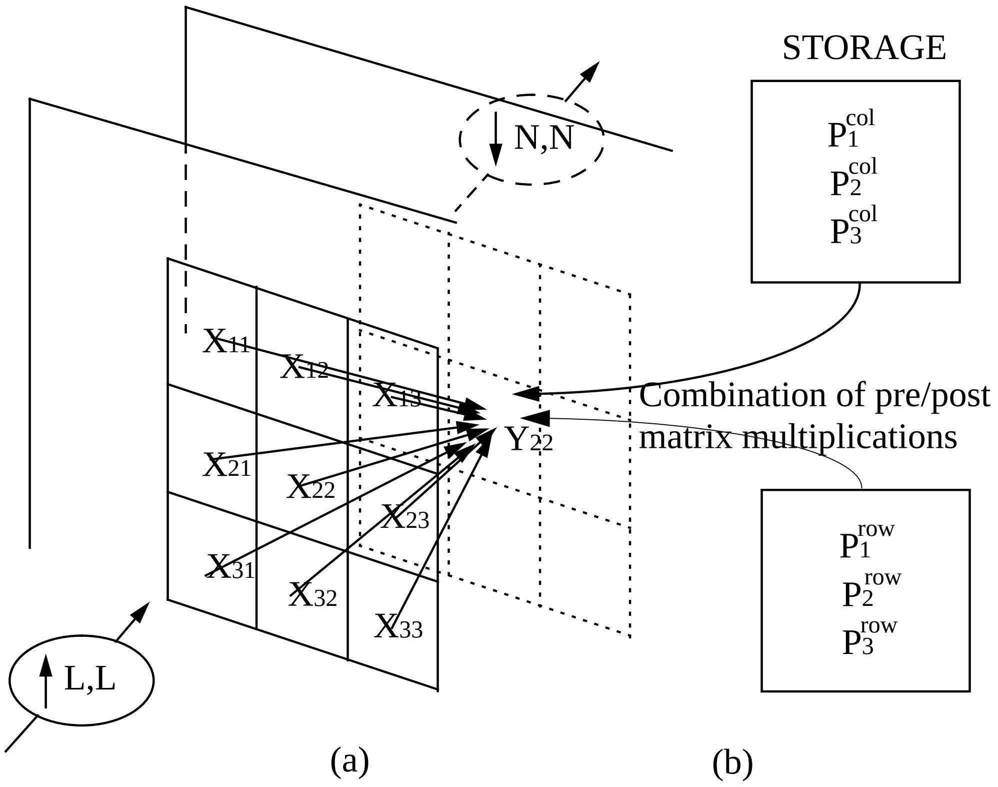

Figure 6: Software/Hardware pipelining structure of 2-D transform domain translation : (a) structure, (b) sub-routine/sub-processor.

as different types). We have analyzed the structure of NTDF systems and have 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 multirate TDF (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 transform domain resolution translation (conversion) problem. These solutions sidestep all the limitations of existing approaches in terms of allowable transform type or filter structure [15] and generalize the issues in [16, 1, 2] 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 all 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 pre-calculate 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 gurantees a small and uniformly distributed arithmetic error which is only due to the individual matrix-vector multiplications.

These results directly generalize those reported in [3]: NTDF eliminates the limitation of regular TDF in terms of allowing different transform sizes, whereas MTDF eliminates the limitation of NTDF by allowing differrent 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.

References

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

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

[3] 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.

[4] S. F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE JSAC, Special Issue on Intelligent Signal Processing*, pp. 1–11, Jan. 1995.

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

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

[7] 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.

[8] S. F. Chang and D. G. Messerschmitt, "A new approach to decoding and compositing motion compensated DCT-based images," *in Proc. of IEEE ICASSP '93 Minneapolis, Minnesota*, pp. V421–V424, Apr. 1993.

[9] A. Eleftheriadis and D. Anastassiou, "Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Coded Digital Video," in *Proceedings, 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, (Durham, New Hampshire), pp. 95–106, April 1995.

[10] A. Eleftheriadis and D. Anastassiou, "Constrained and General Dynamic Rate Shaping of Compressed Digital Video," in *Proceedings, 2nd IEEE International Conference on Image Processing*, (Washington, DC), pp. III.396–399, October 1995.

[11] W. H. Chen and S. C. Fralic, "Image enhancement using cosine transform filtering," *presented at the Image Sci. Math. Symp. Monterey, CA*, Nov. 1976.

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

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

[14] 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)*, pp. 312–324, 1985.

[15] 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.

[16] W. Kou and T. Fjallbrant, "Fast computation of transform coefficients for a subadjacent block for a transform family," *IEEE Trans. Signal Processing*, vol. 39, pp. 1695–1699, July 1991.

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