

A new class of fast shape-adaptive orthogonal transforms and their application to region-based image compression

Ryszard Stasiński *Member, IEEE* and Janusz Konrad *Member, IEEE*

Abstract—Region-based approaches to image and video compression have been very actively explored in the last few years. It is widely expected that they will result in rate/quality gains and expanded functionalities. In such approaches one of the essential problems is the representation of luminance and color in arbitrarily-shaped regions. For rectangular blocks extracted from natural images the discrete cosine transform (DCT) has been found to perform close to the eigentransform. Although for arbitrarily-shaped regions orthogonalization-based procedures have been shown to perform very well, their computational complexity and memory requirements are prohibitive for today's technology. Therefore, other approaches are presently investigated and particular attention is paid to low implementation complexity. In this paper, we propose a new class of orthogonal transforms that self-adapt to arbitrary shapes. The new algorithms are derived from flowgraphs of standard fast transform algorithms by a suitable modification of certain butterfly operators. First, we show how to derive a shape-adaptive transform from the discrete Walsh-Hadamard transform (DWHT) flowgraph. Then, we discuss modifications needed to arrive at a DCT-based shape-adaptive transform. We give implementation details of this transform and compare its computational complexity with several well-known approaches. We also evaluate the energy compaction performance of the new transform for both synthetic and natural data. We conclude that the proposed DCT-based shape-adaptive transform gives a very beneficial compaction/complexity ratio compared to other well-known approaches. The complexity of the new method does not exceed the complexity of two non-adaptive DCTs on a circumscribing rectangle, and therefore, unlike other tested methods with comparable energy compaction, it is suitable for large regions. This property should prove very valuable in the future when *true* region-based image/video compression methods are developed.

Index Terms—Shape-adaptive transforms, SA-DCT, DCT, transform coding, region-based coding, MPEG-4

I. INTRODUCTION

IMAGE and video compression standards commonly used today are based on a uniform partitioning of data into rectangular blocks; luminance and color of each block are suitably processed and organized into a stream of bits for transmission or storage. Although ubiquitous, such schemes suffer from two

basic drawbacks. First, when the available bit rate is severely limited, block boundaries clearly emerge. This distortion is particularly annoying since human observers are sensitive to regular patterns. Secondly, bit stream organization based on rectangular blocks is very rigid and does not allow for flexible manipulation of data streams (e.g., in video database querying). To alleviate the above problems compression methods that do not use uniform rectangular blocks have been considered. One such approach is based on image partitioning into irregularly-shaped regions [15], [23]. Although several variants of this approach have been proposed, most assume that a segmentation of an image into regions is performed first, followed by independent compression of each region (segment). An immediate benefit of such an approach, as opposed to rectangular image partitioning, is that all pixels associated with an object are treated as one entity. Consequently, such functionalities as segment-by-segment progressive transmission or database query by object can be envisaged. The other benefit is that with bit rate reduction, distortion increases uniformly within the whole region; all pixels within a segment are treated jointly. A possible increase in error at region boundary is largely obscured by the spatial masking property of the human visual system (HVS) [16].

Although the upcoming MPEG-4 standard (a good overview can be found in [20]) addresses the above issues, its framework does not depart too far from a block-based hybrid coding structure [9] proposed in MPEG-1/2. To address the region-based access, alpha planes that identify spatial location of regions have been introduced in MPEG-4. The introduction of such planes permits a division of each block into arbitrarily-shaped parts for the purpose of independent texture coding and motion compensation of each part. Clearly, object boundaries can be handled more efficiently in this approach than in standard block-based coding. However, any region is still divided into (perhaps partial) blocks, and therefore an independent coding of these blocks is likely to result in a distortion inside the region (block visibility) if bit rate is severely limited. We believe that a joint processing of *all* pixels in a region, e.g., by a single transformation applied to all of region's samples, would be more beneficial both from the manipulation and compression point of view than the current block-based proposal of MPEG-4. For instance, to perform an object query by example, only a few transform coefficients could be used to reconstruct a coarse content of the region against which a coarse representation of the query example could be compared; this would allow a rapid elimination of unlikely matches. In compression, as discussed

R. Stasiński is with Høgskolen i Narvik, Lodve Langes gt. 2, N-8501 Narvik, Norway on leave from the Technical University of Poznań, Poland. This work was performed when he was with INRS-Télécommunications. J. Konrad is with INRS-Télécommunications, 16 Place du Commerce, Verdun, QC, Canada H3E 1H6.

This work was supported by the Fonds pour la formation de chercheurs et l'aide à la recherche, Québec, Canada under research grant 96-ER-1577 and by the Natural Sciences and Engineering Research Council of Canada under strategic grant STR192788.

above, not only a segment-wise progressive transmission would be possible, but perhaps the compression efficiency could be improved as well. Although not demonstrated to date, it seems likely that due to the contribution of a single transform coefficient to all pixels in a region, coefficient quantization should result in a uniform distortion throughout the region instead of the potential block visibility of MPEG-4. A *true* region-based transformation is likely to be adopted in future compression standards, however to date it has been judged too complex and insufficiently researched to be considered for MPEG-4.

Regardless of the approach adopted (full regions or full/partial blocks), one of the essential problems in region-based image compression is how to efficiently represent luminance and chrominance information within an arbitrary shape. If the basic concepts from the current standards, such as intra-frame coding and inter-frame motion compensation, are to be retained in the new region-based coder, then either the luminance/chrominance information in a region or the corresponding prediction error would have to undergo a transformation. The resulting transform coefficients would have to be quantized to allow rate control. Therefore, the most important properties that a shape-adaptive transform should possess are:

1. *orthogonality* to allow addition of new coefficients without recomputing the previous ones,
2. *good energy compaction* to assure high efficiency,
3. *frequency-like indexing of transform coefficients* to match HVS characteristics (psychovisual irrelevancy removal),
4. *simplicity* to facilitate a real-time implementation.

To date two approaches to the transformation of arbitrarily-shaped regions have been dominant: extrapolation of data to a rectangular support followed by a standard transform and transformation adapted to region shape. Both approaches will be discussed in detail in Section II.

In this paper we propose a new class of fast shape-adaptive orthogonal transforms that, we believe, should find application in *true* region-based image compression. From flowgraphs of the fast discrete Walsh-Hadamard transform (DWHT) and discrete cosine transform (DCT) algorithms we derive their new shape-adaptive variants. The new transforms are orthogonal and very simple. We evaluate the energy compaction performance [16] of the new shape-adaptive DCT-based algorithm against that of several well-known approaches. We demonstrate that the new method has very good energy compaction properties and that its 2-D implementation is quite insensitive to the order of processing directions. We compare the complexity of the new DCT-based transform with other algorithms; we estimate its complexity to be less than the complexity of two standard DCTs applied to a rectangle circumscribed on the region under transformation. We conclude that the method is suitable for the transformation of large regions unlike other tested methods of comparable compaction performance. Moreover, for a rectangular shape the method simplifies to the standard, but sub-optimally implemented, DCT algorithm; backward compatibility with the DCT is assured. Although the transform coefficient indexing does not correspond to true frequencies in the new transform, for natural images a zig-zag scan of coeffi-

cients similar to that used in the DCT corresponds very well to the descending order of coefficient amplitudes. This interesting observation will undoubtedly have to be explored further in any rate-distortion study of the method.

The paper is organized as follows. In Section II we define the problem and discuss the past research. Then, in Section III we develop the new algorithm and in Section IV we discuss its implementation and compare its complexity with other methods. In Section V we show experimental results. Finally, in Sections VI and VII we discuss the advantages of the developed transforms and we draw conclusions.

II. PROBLEM STATEMENT AND PAST RESEARCH

Let S be a region of arbitrary shape and \mathcal{U} its complement to a surrounding rectangle (Fig. 1); pixel values in \mathcal{U} are undetermined. The goal is to represent luminance and/or color in S in the most efficient manner.

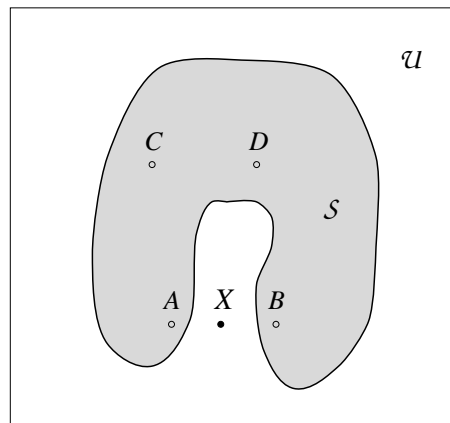


Fig. 1. Example of an arbitrarily-shaped region S within a surrounding rectangle. Pixels in \mathcal{U} are undetermined.

Most of the methods proposed to date can be divided into two classes. The first class comprises methods that exploit standard transforms defined on a rectangular support, such as 2-D DCT. Clearly, this approach requires that samples in \mathcal{U} be somehow determined, for example by an extrapolation from the data within S . In order that the subsequent transform be effective, the extrapolation must preserve statistical properties of the data within S . The other class of methods limits the basis functions of a transform to region's shape; shape-adaptive transforms result.

A. Extrapolation-based approaches

The most straightforward approach to the extrapolation is to pad out \mathcal{U} with a constant intensity and then apply a transform to the resulting rectangular block. Usually, such a method does not perform well since fixed-value padding results in sharp intensity transitions at region boundary that produce long tails in the transform-domain representation. In the case of the DCT, zero padding has been confirmed experimentally to give a very poor performance [4].

A better, yet still very simple, extrapolation can be performed by replicating boundary points of region S everywhere in \mathcal{U} .

For example, a separable operation can be performed as follows: intensity at location X (Fig. 1) can be copied from the horizontally-nearest point in \mathcal{S} (boundary point) either to the left or to the right of X , followed by similar operation vertically but with respect to the resulting rectangle (\mathcal{S} plus the horizontal extrapolation). A variant of such padding has been adopted in phase I of MPEG-4 [9]. Alternatively, a more complicated non-separable extrapolation can be used; for example, intensity at X can be recovered from the nearest point in \mathcal{S} in the Euclidean sense. The computational overhead is small, but since the statistics in the extrapolated region do not match those inside \mathcal{S} (only boundary points are used), the performance is still poor. To approximate region statistics better, mirror reflection of intensities inside \mathcal{S} with respect to region boundary has been proposed; the method performs about 3–4dB better (PSNR) than zero padding for DCT [4] at the cost of increased computational overhead. It is worth noting, however, that any separable implementation of the 2-D (row-column) mirror reflection is sensitive to the order of operations (horizontal followed by vertical or *vice versa*); structural properties of intensity in \mathcal{S} are fairly well propagated into \mathcal{U} only in the direction of the first mirror replication. This will be further discussed in Section V.

A more advanced signal extrapolation based on the method of successive approximation has been proposed in [12]. In the original contribution the authors proposed to project the image vector obtained by scanning \mathcal{S} onto each of the DCT basis vectors limited to \mathcal{S} , and choose the basis vector with the largest projection (largest transform coefficient)¹. This is done iteratively, but since basis vectors are not orthogonal, it is possible that the same basis vector is selected more than once. Another deficiency is that the number of transform coefficients may exceed region size without perfectly reconstructing intensity in \mathcal{S} . To alleviate these problems, various modifications to the original procedure have been proposed [4], [2].

Another interesting technique recently proposed for data extrapolation is an iterative approach based on the theory of successive projections onto convex sets (POCS) [6], [5]. The method can be described by the following steps:

1. extrapolate the undetermined samples in \mathcal{U} from \mathcal{S} (to obtain a signal supported on a rectangle),
2. compute a 2-D DCT on the rectangle,
3. exit, if the result is satisfactory,
4. set to zero some of the DCT coefficients (to limit signal bandwidth),
5. compute inverse 2-D DCT for the bandwidth-limited signal,
6. within \mathcal{S} replace the resulting samples with samples from the original signal (to assure unmodified signal inside \mathcal{S}),
7. return to 2.

The major strength of the method is that it can be implemented in real time using existing hardware with only a slight additional cost. Clearly, the method uses an assumption of limited signal bandwidth to perform extrapolation, a realistic as-

sumption for practical signals. The initial extrapolation and the choice of coefficients for zeroing are design parameters. Usually a simple mirror-image reflection is used as initial extrapolation; the type of initial extrapolation does not change the point of convergence but affects the number of iterations before convergence is reached [5]. As for the zeroing of coefficients, a simple triangular zone in the low-frequency part of the coefficient plane can be used; coefficients inside this zone are retained and those outside are zeroed. According to the POCS theory the method will converge only if the selection of retained DCT coefficients is fixed throughout iterations. This means that the selection of coefficients to be retained is highly dependent on the initial extrapolation; large high-frequency components may be produced by an unsuitable extrapolation and thus may result in a sub-optimal zone for coefficients to be retained. To alleviate this problem it has been proposed to adapt the coefficient selection as the algorithm progresses [2], however this may affect the convergence of the algorithm.

B. Shape-adaptive transforms

Unlike extrapolation methods, shape-adaptive transforms process pixels within \mathcal{S} only; pixels in \mathcal{U} remain undetermined as they do not impact any operation.

Let the intensities in $\mathcal{S} \cup \mathcal{U}$ (Fig. 1) be scanned linearly (e.g., row by row) and represented by vector $\mathbf{x} \in R^M$; M is the total number of pixels in $\mathcal{S} \cup \mathcal{U}$. Similarly, let the intensities in \mathcal{S} be scanned and represented by vector $\mathbf{x}_S \in R^{M_S}$, where M_S is the number of pixels in \mathcal{S} . Since many algorithms discussed in this paper are separable, we will also discuss 1-D shape-adaptive transforms. In fact, we will develop the new transform in detail for the 1-D case. Therefore, the notions of region and undetermined pixels for a 1-D signal need to be suitably modified; we will use the same symbols \mathcal{S} and \mathcal{U} except that we will call the former a (1-D) segment rather than an object. Consequently, N_S will be the number of samples inside the 1-D segment \mathcal{S} and N_U will be the number of undetermined samples; $N = N_S + N_U$.

The optimal transform for arbitrarily-shaped regions is the Karhunen-Loeve transform (KLT) also called the eigentransform. It imposes the upper limit on the compression level for a given signal and therefore often serves as a benchmark [18], [24]. It is not, however, a practical transform; DCT is used instead since for rectangular segments from natural images it gives similar performance.

For an intensity in \mathcal{S} with autocorrelation matrix $\mathbf{R}_S = E\{\mathbf{x}_S \cdot \mathbf{x}_S^T\}$, the KLT is defined as $\mathbf{X}_S = \mathbf{T}\mathbf{x}_S$ where \mathbf{X}_S is the vector of transform coefficients and

$$\mathbf{R}_S = \mathbf{T}^T \cdot \mathbf{\Lambda} \cdot \mathbf{T}. \quad (1)$$

Above, $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues of \mathbf{R}_S , and the superscript T denotes transposition. Since a reliable estimation of the autocorrelation matrix \mathbf{R}_S is not trivial and also since \mathbf{R}_S would have to be transmitted to the receiver, it is much more practical to establish a fixed autocorrelation (autocovariance) model. Due to the high correlation of neighboring pixels in natural imagery, the following exponential model is often used for zero-mean wide-sense stationary processes over rectangular \mathcal{S}

$$[\mathbf{R}_S]_{i,j} = \rho_1^{|m_i - m_j|} \rho_2^{|n_i - n_j|}, \quad (m_i, n_i), (m_j, n_j) \in \mathcal{S}, \quad (2)$$

¹DCT basis vectors limited to \mathcal{S} are obtained by zeroing 2-D DCT basis functions outside \mathcal{S} .

where (m_i, n_i) is the location of i -th pixel.

For very high correlations ($\rho_1, \rho_2 \rightarrow 1$) and rectangular shape \mathcal{S} , the DCT basis functions approach those of the KLT. However, in the case of an arbitrary shape \mathcal{S} resulting from a segmentation process (such as in Fig. 13.b) the assumption of stationarity of intensities is questionable; one would expect a larger correlation between pixels C and D , than between pixels A and B (Fig. 1) despite the same distance between pixels in both pairs. To alleviate this problem an alternative model that relaxes the stationarity assumption has been proposed [24], [25]; it is based on 2-D non-causal first-order Markov process with Neumann boundary conditions. The model has been shown to clearly outperform the stationary model (2) [24]. Unfortunately, since the algorithm requires solving a discrete version of an elliptic partial differential equation, its computational complexity is very high.

In a pivotal contribution Gilge *et al.* [7] have proposed a method to represent the intensity within \mathcal{S} by coefficients of an expansion into orthogonal series with basis functions obtained from polynomial or cosine functions. Although such basis functions can be easily made orthogonal with respect to a rectangular support, they lose orthogonality once limited to an arbitrary shape \mathcal{S} . Therefore, the authors have proposed to perform orthogonalization of the \mathcal{S} -constrained basis functions *via* Gram-Schmidt or Householder procedure. The proposed method applied to DCT basis functions has been shown to have excellent compression capabilities and has been often used as a benchmark algorithm. Again, however, the algorithm requires very large memory for storage of the basis functions (proportional to M_S^2) and is characterized by very high computational complexity.

A very interesting shape-adaptive DCT technique has been devised by Sikora and Makai [19]. Since the method is of row-column type, we first describe its 1-D variant using the 1-D notation established earlier in this section:

1. shift the samples of segment \mathcal{S} to the beginning of the data vector (first N_S vector entries are populated with samples from \mathcal{S} , whereas the remaining entries are undefined),
2. compute the N_S -point DCT using (21) (the DCT size matches the segment size N_S and thus the undetermined samples from \mathcal{U} are not processed),
3. multiply the result by $4/N_S$.

A 2-D separable extension of this algorithm can be performed in a standard way described in Section III-E but the resulting 2-D transform becomes non-orthogonal. To make it orthogonal again, the multiplier above should be proportional to $1/\sqrt{N_S}$ as proposed in [11], [13]. However, the orthogonalized version is not DC-preserving, unlike the original approach of Sikora and Makai [19]; by a DC-preserving transform we mean a transform that gives only one non-zero coefficient for a constant-valued input signal [11]. A transform without DC preservation [13] is suitable for inter-frame coding only, when zero-mean signals are processed. For intra-frame coding it was shown in [11] how to modify the encoder/decoder structure when the properly computed DC coefficient replaces the $X(0, 0)$ sample of the orthogonal transform. These issues are further discussed in

Section III-E.

The method of Sikora and Makai is much less complex computationally than the KLT and Gilge's basis orthogonalization and at the same time has good compression properties [18]. In fact, it is simple enough to be considered as a solution for practical region-based coders, e.g., MPEG-4 [17], [20], [9]; blocks fully-enclosed within a region are treated in a standard fashion whereas boundary blocks containing two or more regions are processed by the shape-adaptive transform. The method is not suitable, however, for the processing of large regions as it requires 1-D DCTs of all lengths from 1 to $\max(K, L)$ for a region with an $K \times L$ circumscribing rectangle.

III. A NEW FAST ALGORITHM FOR SHAPE-ADAPTIVE TRANSFORMATION

Central to the development of a new shape-adaptive transform is the observation that a linear transform algorithm can be represented by a series of matrix multiplications [26]. The proposed methodology is generic and applies to any orthogonal transform, but to make the explanations clear we will make extensive use of examples. Our development below is carried out for a 1-D transform, having in mind an extension to separable multi-dimensional transforms (Section III-E). Therefore, the 1-D notation proposed in Section II-B applies throughout the current section.

A. Matrix representation and permuting butterflies

The underlying idea in the new shape-adaptive algorithm is the replacement of some operations of a fast orthogonal transform algorithm in such a way that the undetermined data samples from \mathcal{U} are ignored, while the overall transform remains orthogonal.

Consider the following matrix representation of a fast orthogonal transform algorithm:

$$\mathbf{T} = \mathbf{T}_K \cdots \mathbf{T}_{i+1} \cdot \mathbf{T}_i \cdot \mathbf{T}_{i-1} \cdots \mathbf{T}_2 \cdot \mathbf{T}_1, \quad (3)$$

where K denotes the number of stages into which the transform can be decomposed. In practical algorithms the matrix \mathbf{T} is usually not strictly orthogonal² but rather proportional to an orthogonal matrix Θ :

$$\mathbf{T} = \gamma \cdot \Theta, \quad \Theta^T \cdot \Theta = \Theta \cdot \Theta^T = \mathbf{I}, \quad (4)$$

where \mathbf{I} is an identity matrix and γ is called an amplitude gain. For an orthogonal transform, its basis vectors form rows of the matrix \mathbf{T} , i.e., $\mathbf{T}^T = [\mathbf{t}_0 \ \mathbf{t}_2 \ \dots \ \mathbf{t}_{N-1}]$, where \mathbf{t}_i is the i -th basis vector.

Let's assume that the products $\mathbf{T}_K \cdots \mathbf{T}_{i+1}$, and $\mathbf{T}_{i-1} \cdots \mathbf{T}_2 \cdot \mathbf{T}_1$ are proportional to orthogonal matrices, and that the matrix \mathbf{T}_i is orthogonal. If the matrix \mathbf{T}_i is replaced by another orthogonal matrix \mathbf{Q}_i , then the new transform:

$$\hat{\mathbf{T}} = \mathbf{T}_K \cdots \mathbf{T}_{i+1} \cdot \mathbf{Q}_i \cdot \mathbf{T}_{i-1} \cdots \mathbf{T}_2 \cdot \mathbf{T}_1,$$

²We will use traditional definitions from linear algebra, although for historical reasons they are somewhat confusing: an orthogonal matrix is a matrix consisting of orthogonal *and* normalized columns, and an orthogonal basis is a basis consisting of orthogonal vectors, not necessarily normalized. Consequently, by an orthogonal transform we will understand a transform with an orthogonal basis.

also represents an orthogonal transform, although different from that represented by \mathbf{T} .

We need to define the new matrix \mathbf{Q}_i in such a way that the overall transform $\widehat{\mathbf{T}}$ be shape-adaptive. We impose the following two requirements:

1. In order for $\widehat{\mathbf{T}}$ to represent the data only within \mathcal{S} , it must not mix samples inside and outside of \mathcal{S} . This will assure that certain transform-domain samples (coefficients) describe only the data within \mathcal{S} , while others correspond exclusively to the undetermined area \mathcal{U} .
2. To maximize the computational efficiency of the algorithm, $\widehat{\mathbf{T}}$ must not process the undetermined samples in \mathcal{U} at all.

These two requirements can be implemented as follows. Let $\mathbf{x} = [x(0) \dots x(N-1)] \in R^N$ and $\mathbf{X} = [X(0) \dots X(N-1)] \in R^N$ be the input and output vectors of the i -th stage of a fast orthogonal transform algorithm described by an $N \times N$ matrix \mathbf{T}_i . Suppose that the input sample $x(k)$ is undetermined and we wish it would appear at position j of the output. Then, in order for $x(k)$ not to be “mixed” with samples from \mathcal{S} , column k of \mathbf{Q}_i (replacing \mathbf{T}_i) should consist of zeros for those row numbers that correspond to samples from \mathcal{S} . Should \mathbf{x} contain another undetermined sample, its corresponding entry in column k could have any value, but since undetermined pixels are irrelevant and we are concerned about efficiency, it should be set to zero as well. However, we let $[\mathbf{Q}_i]_{j,k}$ equal 1 to pass $x(k)$ to the output; setting $[\mathbf{Q}_i]_{j,k}$ to zero would have made \mathbf{Q}_i non-orthogonal. At the same time all entries in row j , except $[\mathbf{Q}_i]_{j,k}$, can be set to zero since the output sample number j remains undetermined. With the above reasoning, the matrix \mathbf{Q}_i takes the following form:

$$\mathbf{Q}_i = \begin{bmatrix} q_{0,0} & \dots & q_{0,k-1} & 0 & q_{0,k+1} & \dots & q_{0,N-1} \\ q_{1,0} & \dots & q_{1,k-1} & 0 & q_{1,k+1} & \dots & q_{1,N-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ q_{j-1,0} & \dots & q_{j-1,k-1} & 0 & q_{j-1,k+1} & \dots & q_{j-1,N-1} \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ q_{j+1,0} & \dots & q_{j+1,k-1} & 0 & q_{j+1,k+1} & \dots & q_{j+1,N-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ q_{N-1,0} & \dots & q_{N-1,k-1} & 0 & q_{N-1,k+1} & \dots & q_{N-1,N-1} \end{bmatrix}.$$

The orthogonality of \mathbf{Q}_i is guaranteed if the matrix:

$$\mathbf{Q}'_i = \begin{bmatrix} q_{0,0} & \dots & q_{0,k-1} & q_{0,k+1} & \dots & q_{0,N-1} \\ q_{1,0} & \dots & q_{1,k-1} & q_{1,k+1} & \dots & q_{1,N-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ q_{j-1,0} & \dots & q_{j-1,k-1} & q_{j-1,k+1} & \dots & q_{j-1,N-1} \\ q_{j+1,0} & \dots & q_{j+1,k-1} & q_{j+1,k+1} & \dots & q_{j+1,N-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ q_{N-1,0} & \dots & q_{N-1,k-1} & q_{N-1,k+1} & \dots & q_{N-1,N-1} \end{bmatrix}$$

obtained from \mathbf{Q}_i by the rejection of column k and row j is orthogonal as well. If there are more undetermined samples at the input to \mathbf{T}_i , the same idea can be applied to \mathbf{Q}'_i again.

The special case of $N=2$ is very important since a 2×2 matrix \mathbf{Q}_i describes a two-point butterfly. Two-point butterflies are building blocks of many fast discrete orthogonal transform algorithms when the size of the data vector is equal to a power of

2, e.g., DWHT (Fig. 2), DCT (Fig. 7), discrete sine transform (DST), fast Fourier transform (FFT), discrete Haar transform (DHT). A two-point orthogonal butterfly can be represented only by either of the two matrices [8]:

$$\mathbf{V}_1 = \begin{bmatrix} \sin(\phi) & \cos(\phi) \\ \cos(\phi) & -\sin(\phi) \end{bmatrix}$$

or

$$\mathbf{V}_2 = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$$

where ϕ is a rotation angle; \mathbf{V}_1 represents *Householder reflection* whereas \mathbf{V}_2 is a *Givens rotation*. The above matrices apply if both input samples belong to \mathcal{S} . However, when either input sample belongs to \mathcal{U} they simplify, as we discussed above, to one of the following matrices³:

$$\mathbf{V}_I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{V}_S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \quad (5)$$

With the above *permuting* butterflies, undetermined samples can be passed through at the same position (\mathbf{V}_I) or they can be moved around (\mathbf{V}_S) without affecting samples from \mathcal{S} .

Since many orthogonal transform algorithms can be represented by a mixture of interleaved orthogonal matrices and diagonal matrices of (scalar) multipliers, the above methodology is sufficient for the derivation of a wide class of shape-adaptive transforms.

B. DWHT example

Due to its particular simplicity, we will use the DWHT algorithm to demonstrate various aspects of the proposed approach. The new methodology will be applied later to the DCT flowgraph for which we will also show experimental results.

Probably the most elegant definition of the N -point non-normalized DWHT ($\gamma = \sqrt{N}$) when N is a power of 2 is given by the following recursion [1]:

$$\begin{aligned} \mathbf{T}_N &= \begin{bmatrix} \mathbf{T}_{N/2} & \mathbf{T}_{N/2} \\ \mathbf{T}_{N/2} & -\mathbf{T}_{N/2} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{T}_{N/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{N/2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_{N/2} & \mathbf{I}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{I}_{N/2} \end{bmatrix}, \end{aligned}$$

where the subscripts show dimensions of submatrices, and $\mathbf{T}_1 = [1]$. Note that the recursion above implies how the fast DWHT algorithm can be constructed. For $N=8$ the recursion stops after two steps resulting in a flowgraph presented in Fig. 2.a and in a matrix that can be decomposed as follows:

$$\mathbf{T} = \mathbf{T}_3 \cdot \mathbf{T}_2 \cdot \mathbf{T}_1, \quad (6)$$

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix},$$

³The multiplier associated with sample in \mathcal{S} may be alternatively set to -1.

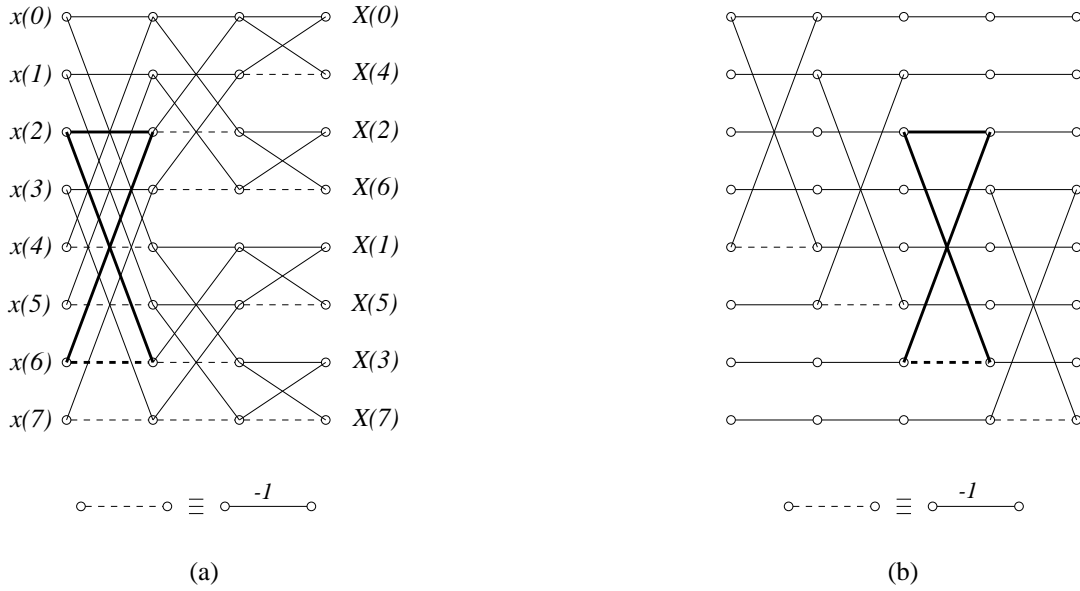


Fig. 2. Flowgraphs of (a) 8-point DWT algorithm and of (b) the decomposition of its first stage.

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix},$$

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

Note that each matrix above corresponds to one stage in the flowgraph from Fig. 2.a. Since each stage consists of four independent operations (butterflies) it can be further decomposed into sub-stages. For example, for stage 1 we have (Fig. 2.b):

$$\mathbf{T}_1 = \mathbf{T}_{1,4} \cdot \mathbf{T}_{1,3} \cdot \mathbf{T}_{1,2} \cdot \mathbf{T}_{1,1}. \quad (7)$$

Matrices \mathbf{T}_1 and $\mathbf{T}_{1,1}$, $\mathbf{T}_{1,2}$, $\mathbf{T}_{1,3}$, $\mathbf{T}_{1,4}$ are not strictly orthogonal and thus a direct substitution of $\mathbf{T}_{1,i}$ ($i \in \{1, 2, 3, 4\}$) by an orthogonal matrix does not apply. However, \mathbf{T}_1 is proportional to an orthogonal matrix Θ_1 , i.e., $\mathbf{T}_1 = \sqrt{2} \cdot \Theta_1$. Consequently,

$$\mathbf{T}_1 = \sqrt{2} \cdot \Theta_{1,4} \cdot \Theta_{1,3} \cdot \Theta_{1,2} \cdot \Theta_{1,1}, \quad (8)$$

where each $\Theta_{1,i}$ ($i \in \{1, 2, 3, 4\}$) is orthogonal and corresponds to one sub-stage in the flowgraph from Fig. 2.b. For example, sub-stage 3 (bold lines in Fig. 2.b) can be described

by the following orthogonal matrix:

$$\Theta_{1,3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & -1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (9)$$

Combining equations (6) and (8) we obtain

$$\mathbf{T} = \mathbf{T}_3 \cdot \mathbf{T}_2 \cdot \sqrt{2} \cdot \Theta_{1,4} \cdot \Theta_{1,3} \cdot \Theta_{1,2} \cdot \Theta_{1,1}, \quad (10)$$

which satisfies the conditions for substitution of the orthogonal matrix $\Theta_{1,3}$ (Section III-A); both $\mathbf{T}_3 \cdot \mathbf{T}_2 \cdot \sqrt{2} \cdot \Theta_{1,4}$ and $\Theta_{1,2} \cdot \Theta_{1,1}$ are proportional to an orthogonal matrix. If at the input to the sub-stage 3 of stage 1 the sample $x(6)$ is undetermined, then there exist only two possible replacements for $\Theta_{1,3}$:

$$\mathbf{Q}_{1,3}^I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

or

$$\mathbf{Q}_{1,3}^S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The above matrices result from replacing entries (2,2), (2,6), (6,2) and (6,6) in the matrix $\Theta_{1,3}$ (9) by the corresponding entries of \mathbf{V}_I and \mathbf{V}_S (5), respectively; orthogonality of $\Theta_{1,3}$ is maintained. Clearly, either of the above matrices may replace $\Theta_{1,3}$ and the overall transform \mathbf{T} (6) will remain orthogonal.

Note that the substitution matrices above do not encompass the $\sqrt{2}$ multiplier (10). Once this multiplier is included, the modified substitution matrices

$$\hat{\mathbf{Q}}_{1,3}^I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\hat{\mathbf{Q}}_{1,3}^S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

apply as direct replacements for $\mathbf{T}_{1,3}$. Note that in column 6 the non-zero entry equals 1 although formally it should be $\sqrt{2}$. This is done to save unnecessary computations since $x(6)$ is undetermined and its value on output is irrelevant.

At the output of the first stage the undetermined sample $x(6)$ will appear as a sample with index 2 or 6 depending whether $\mathbf{Q}_{1,3}^I$ or $\mathbf{Q}_{1,3}^S$ is used. This means that the butterfly from the second stage into which feeds the undetermined sample should also be replaced by a permuting butterfly. The same reasoning applies to subsequent stages. The resulting DWHT-based transform will completely ignore the undetermined sample $x(6)$.

C. Modifications to assure a constant-valued (DC) basis function

Since the proposed transform will be applied to images, it is very important that basis functions match image properties as closely as possible. Although in a general case designing those basis functions is non-trivial, one image property stands out. In many synthetic as well as natural images large areas exhibit constant or almost constant luminance/color; this is a 2-D equivalent of the DC component in 1-D signals. If none of the basis functions of an orthogonal transform is flat, then a good approximation of luminance/color in such an area will require a combination of several basis functions. Since an image compression scheme usually involves quantization of corresponding coefficients, the outcome may be a non-flat (rippled) approximation of a constant-luminance/color area. The situation would be different, however, should one of the basis functions be constant-valued. Then, quantization of its coefficient

would cause a gray level/color change, but no ripples (distortions) would show up. Therefore, we need to design a transform with a constant-valued basis function; an orthogonal transform that includes such a basis function is DC-preserving (see definition in Section II-B).

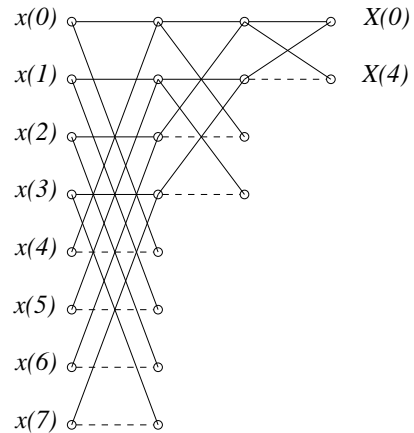


Fig. 3. Example of 8-point structure for computing the DC component in many orthogonal transforms.

For 1-D linear non-adaptive transforms the constant-valued (DC) component can be expressed as follows:

$$DC = \frac{1}{N} \sum_{n=0}^{N-1} x(n). \quad (11)$$

This component is computed as output sample $X(0)$ for the discrete Fourier transform (DFT), all types of DCT, DWHT, DHT, and other transforms. If N is a power of 2, $X(0)$ is usually computed jointly with $X(N/2)$ by the structure presented in Fig. 3. The main difference between various algorithms is in the way the input data samples are permuted, e.g., Fig. 2.a (DWHT) versus Fig. 7 (DCT).

In the case of a shape-adaptive transform special care must be taken to properly compute the DC component. For a butterfly with x_{top} and x_{bot} being the top and bottom input samples, respectively, a logical strategy is to apply the following rules:

1. if $x_{top} \in \mathcal{S}$ and $x_{bot} \in \mathcal{S}$, apply regular butterfly,
2. if $x_{top} \notin \mathcal{S}$ and $x_{bot} \in \mathcal{S}$, swap positions of both samples, i.e., apply \mathbf{V}_S (5),
3. if $x_{top} \in \mathcal{S}$ and $x_{bot} \notin \mathcal{S}$, pass both samples with no change, i.e., apply \mathbf{V}_I ,
4. if $x_{top} \notin \mathcal{S}$ and $x_{bot} \notin \mathcal{S}$, pass both samples with no change, i.e., apply \mathbf{V}_I .

In case 2 above, x_{top} is undefined and therefore \mathbf{V}_S should be used to swap positions and move the undefined sample towards the bottom of the flowgraph. Similarly, in case 3 x_{bot} is undefined but since x_{top} is defined their positions should not be swapped as otherwise the undefined sample would be pushed up in the flowgraph and could preclude computation of the DC component. In case 4 it does not matter what operation is performed (both samples are undefined) but to minimize the complexity both inputs should be passed with no change. With this strategy all undetermined samples are pushed down

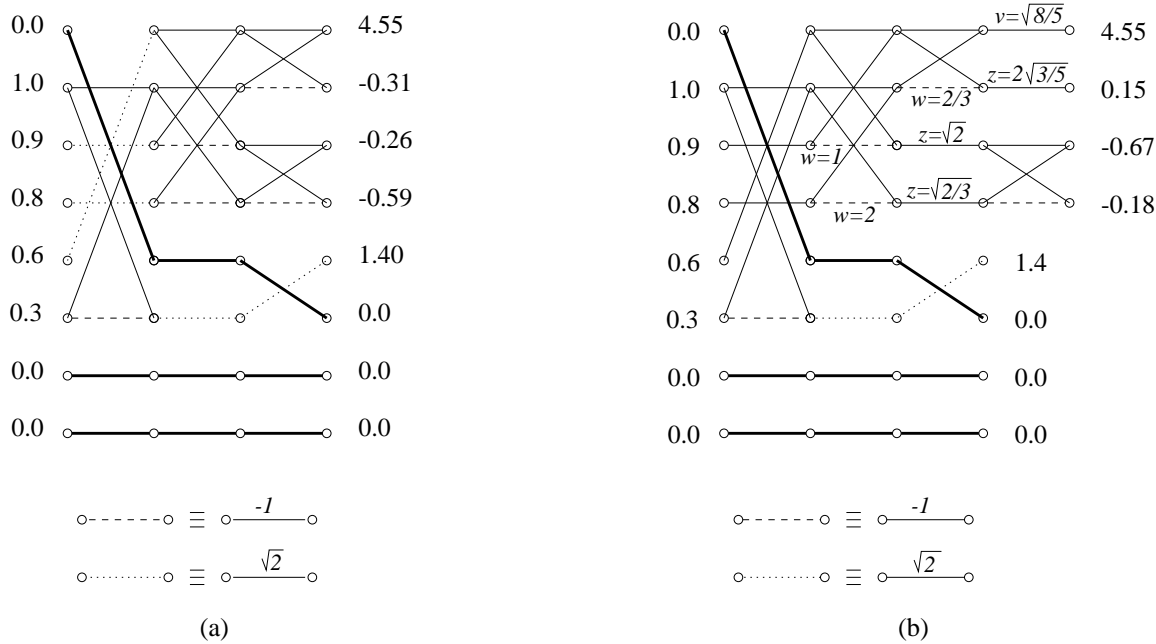


Fig. 4. Flowgraphs of 8-point shape-adaptive DWHT-based fast algorithm for $\mathbf{x} = [0 \ 1 \ .9 \ .8 \ .7 \ .6 \ .3 \ 0 \ 0]$: (a) with non-constant and (b) with constant-valued basis function for $X(0)$. Undetermined samples are assumed to be equal to zero; their flow is shown by bold lines.

to lower parts of the algorithm flowgraph. An open question is whether to apply this strategy to the whole algorithm flowgraph (Fig. 2) or only to its DC part from Fig. 3. From the DWHT flowgraph (Fig. 2.a) it is clear that every time an undetermined sample is pushed down by the \mathbf{V}_S operation (instead of the regular butterfly), its index at the output increases. For example, let us analyze the input sample $x(0)$ (Fig. 2.a). With the \mathbf{V}_I operation between $x(0)$ and $x(4)$, the index of the input sample $x(0)$ would be 0 at the output. However, the \mathbf{V}_S operation would increase the output index to 1. In the second stage, another \mathbf{V}_S operation would increase this index to 3, whereas the final \mathbf{V}_S in the third stage would increase it further to 7 (note that the order of output samples in Fig. 2 is permuted). Consequently, undetermined samples would be moved to less important high-frequency positions, leaving the low-frequency positions for transform samples.

In the case of the DCT algorithm that we will discuss in Section IV [21], there are very few exceptions to these rules, at least for small and moderate values of N . Therefore, we shall apply the above rules to the whole structure of the DWHT and DCT algorithms. An example flowgraph of shape-adaptive DWHT-based⁴ algorithm constructed using the above rules is presented in Fig. 4.

The above algorithm, however, does not yet assure a constant-valued basis function and therefore $X(0)$ does not correspond to the true DC value. This is caused by the $\sqrt{2}$ multipliers in the algorithm structure (Fig. 4.a) that are due to the $\hat{\mathbf{Q}}^I$ and $\hat{\mathbf{Q}}^S$ matrices described in the previous section. Fig. 5.a-b shows basis functions associated with $X(0)$ and $X(N/2)$ for the shape-adaptive DWHT-like algorithm from Fig. 4.a. These basis functions can be easily deduced from Fig. 4.a. Note that

$X(k)$ is a linear combination of $x(n)$, $n = 0, \dots, N - 1$, and therefore that the weights of this combination are equal to the samples of the basis function for $X(k)$. These weights can be inferred by traversing the flowgraph between each input $x(n)$ and a given output $X(k)$, and by compounding the encountered multipliers (continuous line = 1, dashed line = -1, dotted line = $\sqrt{2}$). Note that the basis function associated with $X(0)$ (Fig. 5.a) is not flat. The corresponding basis functions for the 8-point shape-adaptive DCT-based algorithm (Fig. 8.a) consist of the same samples, but in a different order.

If we remove the $\sqrt{2}$ multipliers from the DC-computing substructure (Fig. 3) of the DWHT flowgraph (Fig. 4.a), the basis function associated with $X(0)$ becomes flat and $X(0)$ becomes the true DC component for data within the segment \mathcal{S} . The transform, however, is no longer orthogonal. Fig. 5.c-d illustrates this case by showing basis functions for $X(0)$ and $X(N/2)$ in the algorithm from Fig. 4.a deprived of $\sqrt{2}$ multipliers.

In analogy to sample $X(0)$ representing the DC component, $X(N/2)$ is a high-frequency component and can be expressed as follows:

$$\begin{aligned} X(N/2) &= \sum_{n=0}^{N-1} (-1)^n x(n) \\ &= \sum_{m=0}^{N/2-1} x(2m) - \sum_{m=0}^{N/2-1} x(2m+1). \end{aligned} \quad (12)$$

$X(N/2)$ is also a high-frequency component in DFT, DCT, DST, and DHT. If certain samples $x(n)$ are undetermined, expression (13) must be modified. Let $\zeta(n)$ be an indicator function defined as follows:

$$\zeta(n) = \begin{cases} 1 & \text{if } n \in \mathcal{S}, \\ 0 & \text{if } n \in \mathcal{U}. \end{cases}$$

⁴We call this algorithm DWHT-based (DCT-based) since some DWHT (DCT) butterflies have been replaced by permuting butterflies and therefore the resulting basis functions are not DWHT (DCT) basis functions any more.

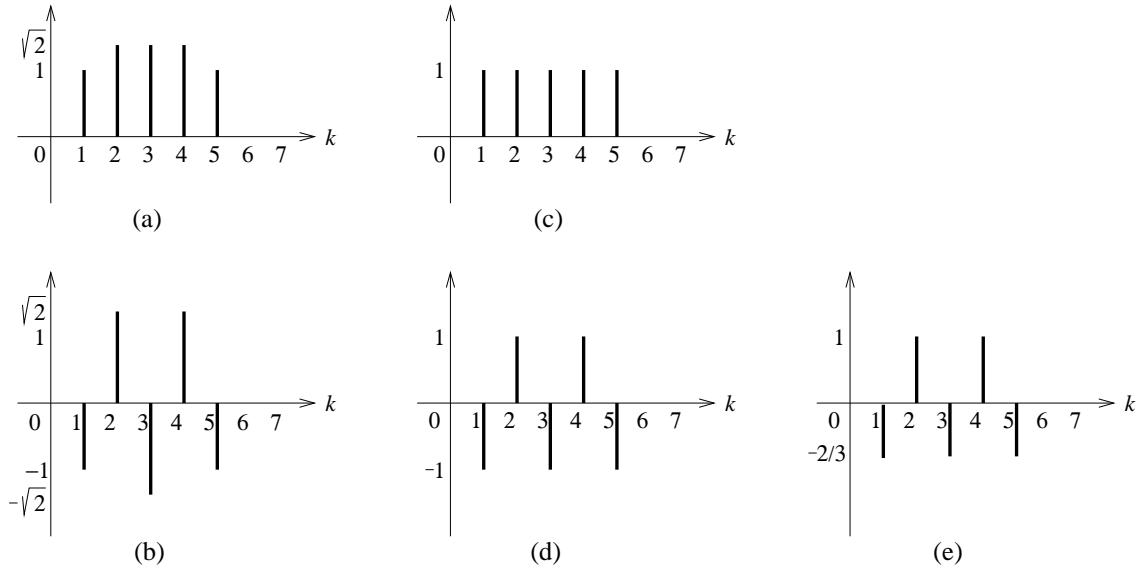


Fig. 5. Basis functions used in the computation of samples (a) $X(0)$ and (b) $X(N/2)$ by the algorithm from Fig. 4, and (c-d) the same functions after removal of the $\sqrt{2}$ multipliers to allow precise calculation of DC. (e) Suitably scaled basis function for $X(N/2)$ to recover orthogonality ($w=2/3$).

Then, counts of even- and odd-indexed samples in \mathcal{S} are

$$\alpha_e = \sum_{m=0}^{N/2-1} \zeta(2m), \quad \alpha_o = \sum_{m=0}^{N/2-1} \zeta(2m+1). \quad (13)$$

Using $\zeta(n)$, $X(N/2)$ for the shape-adaptive algorithm becomes

$$X(N/2) = \sum_{m=0}^{N/2-1} \zeta(2m)x(2m) - \sum_{m=0}^{N/2-1} \zeta(2m+1)x(2m+1). \quad (14)$$

After the removal of $\sqrt{2}$ multipliers from the DC-computing substructure (Fig. 3), the basis vectors of \mathbf{T} that correspond to $X(0)$ and $X(N/2)$ become (even N)

$$\begin{aligned} \mathbf{t}_0 &= [\zeta(0) \zeta(1) \zeta(2) \dots \zeta(N-1)]^T, \\ \mathbf{t}_{N/2} &= [\zeta(0) - \zeta(1) \zeta(2) \dots - \zeta(N-1)]^T, \end{aligned}$$

and then

$$\mathbf{t}_0^T \mathbf{t}_{N/2} = \alpha_e - \alpha_o. \quad (15)$$

In general, this scalar product is non-zero; \mathbf{t}_0 and $\mathbf{t}_{N/2}$ are not orthogonal. To make them orthogonal again, we multiply the odd-indexed samples of $\mathbf{t}_{N/2}$ by w so that $\mathbf{t}_0^T \mathbf{t}_{N/2} = \alpha_e - w\alpha_o$, and we set w to α_e/α_o . In the particular case of functions from Fig. 5.d-e, w equals $2/3$. The multiplication by w is shown in Fig. 6.a; it can be performed in the very last butterfly of the substructure from Fig. 3.

Note that in the particular case shown in Fig. 5, the final basis functions for $X(0)$ and $X(N/2)$ resemble closely their non-adaptive counterparts; both functions have restricted support and the function for $X(N/2)$ (Fig. 5.e) has the odd samples additionally scaled. However, since both the support and

the scaling depend on the number of samples in \mathcal{S} and on their distribution this behavior does not generalize; a general relationship between the form of basis functions and object shape is very complicated.

Note that the above reasoning applies throughout the butterflies of the structure from Fig. 3. More specifically, the upper butterfly in the second stage of the flowgraph from Fig. 3 computes the DC and high-frequency components for all even indices of data samples, and the lower butterfly computes the same components for odd indices. The four butterflies in the preceding stage carry out the same computation for indices *modulo* 4. This means that the local orthogonalization described above applies to all butterflies of that structure, except that the multiplier w must be re-defined as follows:

$$w = \frac{\alpha_a}{\alpha_b} \quad (16)$$

where α_a, α_b denote numbers of samples from \mathcal{S} summed up at nodes a and b (Fig. 6.a). In the very last butterfly, however, $\alpha_a = \alpha_e$, and $\alpha_b = \alpha_o$.

This local orthogonalization guarantees that the DC basis function is orthogonal to other basis functions in the structure from Fig. 3. Unfortunately, energies of these basis functions are, in general, different from the corresponding energies in the non-adaptive version of the algorithm. Therefore, if we carry out computations using a w -corrected structure (butterfly from Fig. 6.a applied throughout the structure from Fig. 3) but with unchanged remaining part of the algorithm flowgraph, the whole transform will be non-orthogonal. This is due to the fact that the basis function energies have been diminished by the rejection of $\sqrt{2}$ multipliers in the structure from Fig. 3. We restore the lost gain by introducing multipliers z in butterflies containing multiplication by w (Fig. 6.a):

$$z = \sqrt{\frac{2^k}{\alpha_a + w^2 \cdot \alpha_b}} = \sqrt{\frac{2^k}{\alpha_a \cdot (1+w)}} = \sqrt{\frac{2^k}{(\alpha_a + \alpha_b) \cdot w}} \quad (17)$$



Fig. 6. (a) Modification of butterflies in the flowgraph from Fig. 3 leading to the orthogonalization of basis functions. (b) Non-normalized inverse butterfly for butterfly from (a).

where k is the stage number in which the butterfly is performed ($k = 1, \dots, \log_2 N$). In the very last butterfly the term in the numerator under the square root (17) is equal to the energy of the basis function of $X(N/2)$ for the non-adaptive algorithm (13), i.e., $\sum_{n=0}^{N-1} ((-1)^n)^2 = N$ for $k = \log_2 N$, whereas the term in the denominator is equal to the energy of the same function for the shape-adaptive algorithm (15) taking into account the multiplier w , i.e., $\sum_{m=0}^{N/2-1} (\zeta(2m))^2 + \sum_{m=0}^{N/2-1} (w\zeta(2m+1))^2 = \alpha_e + w^2\alpha_o$ also for $k = \log_2 N$.

Finally, we must compensate for the loss of energy of the DC basis function caused by the rejection of $\sqrt{2}$ multipliers. We do so by multiplying $X(0)$ by

$$v = \frac{\gamma}{\sqrt{N_S}} = \sqrt{\frac{N}{N_S}} \quad (18)$$

in the very last butterfly (recall that for DWHT the amplitude gain is $\gamma = \sqrt{N}$). The normalized and orthogonalized version of the 8-point shape-adaptive DWHT-based transform from Fig. 4.a that assures a constant-valued basis function is shown in Fig. 4.b.

D. Inverse algorithms

An inverse shape-adaptive transform is needed for the reconstruction of region \mathcal{S} in the decoder. The matrix representation of a transform algorithm greatly simplifies the derivation of its inverse because

$$\mathbf{T}^{-1} = \mathbf{T}_1^{-1} \dots \mathbf{T}_{i-1}^{-1} \cdot \mathbf{T}_i^{-1} \cdot \mathbf{T}_{i+1}^{-1} \dots \mathbf{T}_K^{-1}.$$

Since only orthogonal and diagonal matrices appear in representations of many fast orthogonal transform algorithms (e.g., FFT, DCT/DST, DWHT, DHT), the derivation of inverse algorithms is particularly simple. The only remark to be made here is that many multiplications may be saved if we allow the inverse algorithm to introduce a gain. Then, the true inverse can be obtained by pre- or post-multiplication of the data by the product of gains for forward and inverse shape-adaptive transforms (4), which may be included in the decoder.

Taking into account the above simplification, the non-normalized inverse of the operation from Fig. 6.a is shown in Fig. 6.b, where $\Delta = 1 + w$ is the determinant of the matrix representing operation from Fig. 6.a taken with negative sign. Notice that the input transform sample $X(0)$ should be multiplied by the inverse of v (18) applied at the output of the forward algorithm.

E. Multi-dimensional extensions

A one-dimensional transform can be always generalized to a two-dimensional separable transform [10]; first, the 1-D transform is applied to each of the image columns, and then to each of the rows of the intermediate result (or *vice versa*). The approach is very simple and it guarantees that the computational complexity of the 2-D algorithm is not greater than $O(N^3)$, and only $O(N^2 \log N)$ for fast transforms such as DFT, DCT, DST, DWHT. This approach has been used by Sikora and Makai [19]. We propose a similar separable 2-D generalization for the shape-adaptive transforms described here.

In the case of shape-adaptive approaches, a difficulty emerges immediately. The usual separable 2-D transform supported on a rectangle (non-shape-adaptive) is described by the following formula [10, Section 5.2]

$$X(k, l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [\mathbf{T}]_{k,m} [\mathbf{P}]_{n,l} x(m, n), \quad (19)$$

where \mathbf{T} and \mathbf{P} are row- and column-wise transforms, respectively. Note that all rows are processed by the same transform \mathbf{T} , whereas all the columns are processed by \mathbf{P} . This formulation does not hold for a shape-adaptive transform. Since, in general, rows (columns) of the rectangle surrounding \mathcal{S} have different numbers of pixels belonging to \mathcal{S} , the row-wise (column-wise) 1-D transforms are different. In our approach, the flowgraphs of 1-D transforms (and therefore the \mathbf{T} and \mathbf{P} matrices) differ and in the case of the Sikora and Makai method the lengths of the DCTs differ. Therefore, the formulation (19) must be modified as follows:

$$X(k, l) = \sum_{m=0}^{M-1} [\mathbf{T}_l]_{k,m} \sum_{n=0}^{N-1} [\mathbf{P}_m]_{n,l} x(m, n), \quad (20)$$

where \mathbf{T}_l ($l = 0, \dots, N-1$) and \mathbf{P}_m ($m = 0, \dots, M-1$) are again row- and column-wise 1-D transforms but, in general, different for each row and column. Note that when the order of computations is changed, i.e., a row-wise transformation is followed by a column-wise transformation, the resulting 2-D transform is usually different. Indeed, this can be observed in the experimental results of Section V. The generalization to the multi-dimensional case can be done similarly.

Recall that orthogonality and good energy compaction are two of the properties required of a transform in the context of compression (Section I). In constant-luminance/color areas, often encountered in images, the latter condition translates to DC

preservation as discussed in Section III-C. However, for shape-adaptive separable 2-D transforms that can be formulated as in (20), the orthogonality and the DC preservation are in conflict [11]. Assume that all 1-D transforms in (20) are orthogonal, that they compute correct DC value (11), and that their gain is a function of N_S . In order to construct an orthogonal separable 2-D transform, all 1-D transform gains need to be equalized to one value. Hence, the algorithm should proceed as follows:

1. apply 1-D transforms \mathbf{P}_m ($m = 0, \dots, M - 1$) to columns,
2. scale the results to equalize gains between all columns,
3. apply 1-D transforms \mathbf{T}_l ($l = 0, \dots, N - 1$) to rows of the intermediate result,
4. scale the results to equalize gains between all rows.

Since the 1-D transforms compute correct DC values, if step 2 were omitted the 2-D transform sample $X(0, 0)$ would give the correct 2-D DC value. However, the scaling in step 2 breaks this paradigm down⁵. The conflict between orthogonality and DC preservation is a serious problem of the Sikora and Makai method [11], [13] as well as of the approach proposed here.

Note that the gain equalization in the new 1-D shape-adaptive transforms \mathbf{P}_m and \mathbf{T}_l is assured by the $\sqrt{2}$ multipliers in the permuting butterflies, by the z multipliers in Fig. 6, and by the v multiplier to scale the $X(0)$ sample. Since the multiplier v depends on N_S (18), it causes the breakdown of correct DC computation as discussed above. However, we can make our shape-adaptive 2-D separable transform both orthogonal and DC-preserving; butterflies from Fig. 6 have been devised for this purpose. First, all v multipliers at the output of 1-D column transforms \mathbf{P}_m are removed; the correct 2-D DC value is now computed, but the 2-D transform becomes non-orthogonal. Then, we need to recover the orthogonality. Lets imagine that the sample $X(0, 0)$ is computed in a huge $2 \log_2 N$ -stage structure similar to that from Fig. 3; the last $\log_2 N$ stages are performed by a 1-D algorithm processing the row of intermediate samples $X(0)$ of image columns, while the first $\log_2 N$ stages are parts of column-wise algorithms computing those intermediate samples. In order to make the 1-D transforms orthogonal several output butterflies of this structure have been transformed in accordance with Fig. 6. What makes the 2-D transform non-orthogonal is: 1) the breakdown in passing α_a and α_b from the part of the structure belonging to column transforms to that processed inside the row transform, and 2) the inappropriate values of multipliers z in the row transform.

As for the first problem, recall that α_a and α_b (16) are counts of samples belonging to \mathcal{S} that are summed up at nodes a and b . Since the algorithm for \mathbf{T}_0 begins with butterflies from Fig. 6, the nodes a and b for the first-stage butterfly in \mathbf{T}_0 are nodes at which samples $X(0)$ of two different column transforms, e.g., \mathbf{P}_m and \mathbf{P}_n , are available. This means that the counts α_a and α_b in \mathbf{T}_0 are equal to N_S for columns m and n ; the w multiplier (16) is uniquely defined.

As for the second problem, as explained in Section III-C

⁵Although a transform without scaling computes the correct DC value, it does not have to be DC-preserving, in contrast to the original proposal of Sikora and Makai (see [11]). In this case, the transform is not orthogonal and can produce several non-zero AC coefficients for strictly flat intensities.

for the case of 1-D transforms, the multiplier z restores the lost transform gain due to the rejection of $\sqrt{2}$ multipliers (17). Now, however, we deal with the whole 2-D transform and we need to perform gain correction due to the removal of multipliers v . Recall that the transform gain for non-adaptive column (1-D) transforms is simply γ (4); this is the target gain for the shape-adaptive 2-D transform prior to the row transform on the intermediate samples $X(0)$. This means that all z coefficients of this row transform can be simply multiplied by γ to track the gain of the non-adaptive 2-D structure. Finally, the $X(0, 0)$ sample should be corrected with respect to the whole 2-D transform gain which means that the multiplier v should be $v = \gamma^2 / \sqrt{N_S^{2D}}$, with N_S^{2D} being the number of samples in the 2-D region \mathcal{S} . We need not even write a special procedure for this particular row transformation; appropriately initialized standard procedure will do the job correctly.

Of course, the construction of a shape-adaptive transform based on multidimensional fast transforms like vector-radix FFTs is also possible. This is due to the fact that a finite multidimensional vector can be always mapped into a one-dimensional vector, and hence a two-dimensional transform flowgraph can be interpreted as a one-dimensional one. Then, the reasoning described for the one-dimensional transforms can be applied as well.

IV. IMPLEMENTATION AND COMPLEXITY OF THE SHAPE-ADAPTIVE DCT-BASED ALGORITHM

Although DCT is more complex computationally than DWHT, it performs much better on natural images. In fact, DCT is the most ubiquitous transform used in image/video compression today.

In the subsequent discussion of the new shape-adaptive DCT-based algorithm, we shall use DCT given by the following formula

$$X(k) = c(k) \sum_{n=0}^{N-1} x(n) \cdot \cos\left[\left(n + \frac{1}{2}\right) \cdot k \cdot \frac{\pi}{N}\right], \quad (21)$$

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0, \\ 1 & \text{if } k = 1, \dots, N - 1, \end{cases} \quad k = 0, \dots, N - 1,$$

for which the amplitude gain γ is $\sqrt{N/2}$ [29]. In particular, we shall study a DCT algorithm introduced in [21] that belongs to the group of algorithms requiring the smallest known number of arithmetical operations. The first DCT algorithm having this property was presented in [28]. It is difficult to say (if possible at all) which of the best DCT algorithms has the simplest realization. However, it has been shown that the algorithm from [21] has good error rounding properties [27].

A. Algorithm implementation

As can be seen from Fig. 7, the flowgraph of a DCT algorithm is not as regular as that of the DWHT algorithm (Fig. 2). However, a more detailed analysis shows that the only important complication is related to the appearance of general orthogonal butterflies (i.e., with non-trivial multipliers). Transformation of these butterflies into permuting butterflies is straightforward

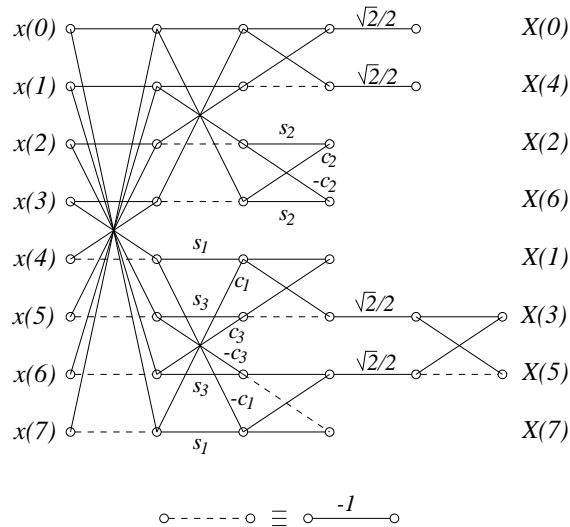


Fig. 7. Flowgraph of the 8-point DCT algorithm used in the paper ($s_i = \sin \pi i/16$, $c_i = \cos \pi i/16$).

ward (Section III). Fig. 8 shows flowgraphs of 8-point shape-adaptive DCT-based fast algorithm; in Fig. 8.a flowgraph resulting in a non-constant basis function for $X(0)$ is shown, whereas Fig. 8.b shows the same flowgraph after normalization described in Section III-C that assures a constant-valued basis function for $X(0)$. An important yet not obvious element in this example is the value of multiplier $v = \sqrt{N/(2N_S)}$ resulting from the fact that the DCT amplitude gain γ is equal to $\sqrt{N/2}$, i.e., it is $\sqrt{2}$ times smaller than that for the DWHT. The factor of 2, however, disappears from the denominator of v for shape-adaptive versions of DFT, DWHT, and DHT algorithms.

The input data to a shape-adaptive algorithm consists of image and segmentation vectors. While the former contains intensity and color of an image region, the latter comprises labels identifying whether a given location belongs to the segment S . The one-dimensional shape-adaptive forward algorithm (encoder) may be implemented in one step by means of S -controlled “if” statements preceding each butterfly (decision whether to use regular or permuting butterflies). The inverse algorithm (decoder), however, requires two steps: the first step to permute the transmitted shape of segment S into S' and recover the order of transmitted transform coefficients, and the second step to perform the actual S' -controlled computations.

B. Computational complexity

The computational complexity of the new algorithm is of the same order as that of the DCT. In the worst case, when the input vector contains no undetermined samples (standard DCT), in addition to DCT operations the new algorithm tests if a sample is determined and performs (redundant) switching before each DCT butterfly.

Since the shape-adaptive algorithm consists of up to two S -dependent stages (decoder), it actually performs up to two additional complex control operations per each DCT butterfly⁶.

⁶Such an operation requires an if-elseif-elseif-...-endif rather than if-endif structure to handle the four possible combinations of the 2 input samples (determined versus undetermined).

It should be noted that even a standard DCT implementation should perform a binary decision testing before its butterflies as some of them are multiplierless while others contain multiplications. This is caused by asymmetry characteristics of the optimal DCT algorithms that are too complicated to be realized by pure “do” loops.

Let’s consider a 1-D N -point transform and a 2-D separable transform with $M=N \times N$ points; we implicitly assume that the surrounding rectangle (square) is $N \times N$. Our conservative estimate is that the computational complexity of the new algorithm is approximately that of two standard DCT algorithms. With the appearance of undetermined samples its complexity actually declines since one such sample causes that on average $1/N$ of the butterflies lose their status as standard DCT operations. The burden associated with the calculation of w and z , and multiplication by them is of rank $O(N)$. Moreover, (integer) values of α_a and α_b necessary for the calculation of w and z are small; a look-up table can be used instead of performing real-time computation of w and z . The only exception is the algorithm operating on $X(0)$ samples of the first processing direction (see Section III-E). However, its extra cost is negligible compared to the complexity of the whole transform. Thus, the overall complexity for an N -point shape-adaptive DCT-based transform is of the order $O(2N(1 + \log N))=O(N \log N)$. In the case of 2-D separable transform with an $N \times N$ surrounding square, the 1-D transform is repeated N times for image rows and then N times for its columns. Clearly, the 2-D transform’s complexity is $2N$ times higher than the complexity of the 1-D transform, i.e., of the order $O(N^2 \log N)$.

Of the methods discussed in Section II the lowest computational complexity is attained by techniques based on data extrapolation followed by a rectangle-supported DCT; exception is the POCS method. Their complexity is equal to that of one DCT algorithm plus $O(N_U)$ operations for filling-in U , thus giving the order of $O(N \log N + N_U)$ for 1-D case and approximately $O(N^2 \log N + M_U)$ for 2-D separable case (M_U is the number of undetermined samples in the surrounding $N \times N$ square). It should be underlined, however, that $O(N_U)$ or

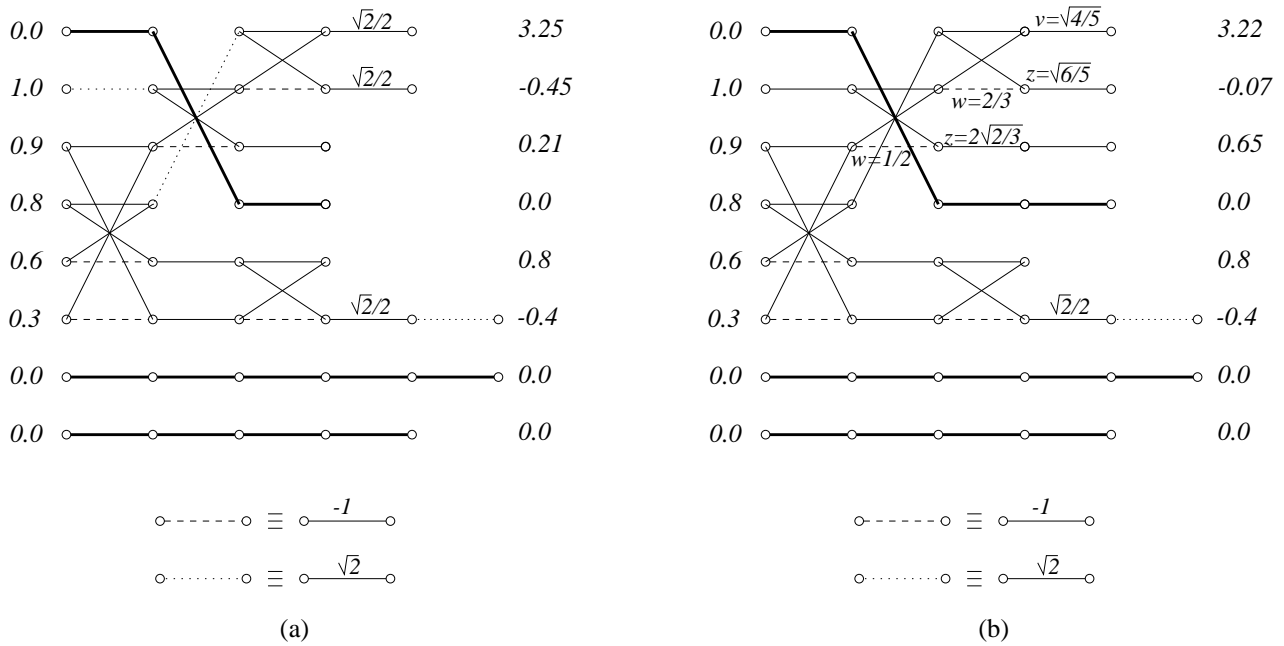


Fig. 8. Flowgraphs of 8-point shape-adaptive DCT-based fast algorithm for $\mathbf{x} = [0 \ 1 \ .9 \ .8 \ .7 \ .6 \ .3 \ 0 \ 0]$: (a) with non-constant and (b) with constant-valued basis function for $X(0)$. Undetermined samples are assumed to be equal to zero; their flow is shown by bold lines. For easier comparison with Fig. 7, redundant flowgraph structures are not removed.

$O(M_U)$ may be significant. For example, for the face segment (Fig. 13.c) consisting of $M_S=2261$ object samples, the smallest $2^K \times 2^L$ surrounding rectangle necessary for implementing a fast DCT algorithm has the size of $M=64 \times 128=8192$ samples; $M_U=5931$. Moreover, implementation of mirror reflections requires a sophisticated combination of control instructions. In conclusion, for large M_U implementation of mirror reflections may be of comparable complexity to that of the DCT.

The POCS method requires some data extrapolation for initialization and then each iteration requires two DCTs plus $O(N)$ operations for data testing/zeroing. Since in our implementation usually about 5 iterations were necessary, the complexity of the method was high; about 10 times the complexity of the DCT. However, existing hardware can be used for implementation and the decoder is very simple since just one inverse DCT is needed. This is advantageous for broadcasting, but in a point-to-point transmission where both the encoder and decoder complexity count, the overall computational complexity of the algorithm proposed here is much lower.

The method proposed by Sikora and Makai [19] is the only shape-adaptive method that under certain conditions may have computational complexity comparable to the new algorithm. This is the case when small data blocks and fast DCT algorithms are used. However, a practical implementation of all necessary DCT algorithms (for a given maximum block size) is complicated, and for N exceeding 10 becomes extremely complex. For example, for $N \leq 10$ we need 5 separate algorithms [?]: for $N = 2^K$ ($N = 2, 4, 8$), for $N = 2^K \cdot 3$ ($N = 3, 6$), for $N = 2^K \cdot 5$ ($N = 5, 10$), for $N = 7$, and for $N = 9$. An alternative is to compute the DCT directly from the definition, but although the method becomes conceptually much simpler, it also becomes much more complex computationally. It suffices to say that the complexity of a fast DCT algorithm grows with $O(N \log N)$, in contrast to $O(N^2)$ for the direct formula. For

an $N \times N$ -point 2-D separable transform the complexity would grow to $O(N^3)$. Clearly, the complexity of Sikora’s method is comparable to the proposed algorithm only for small N .

Other shape-adaptive transform methods tested here are much more complex computationally and/or memory inefficient. Computational complexity of both KLT and Gilge’s method is of the order of $O(N^3)$ for 1-D case, which gives $O(N^6)$ for $N \times N$ -point 2-D (non-separable) transform. Clearly, the complexity of definition-based Sikora’s method for 2-D vectors is lower ($O(N^3)$), as is that of the algorithm proposed here ($O(N^2 \log N)$).

V. EXPERIMENTAL RESULTS

To evaluate the performance of the new algorithm in terms of the energy compaction efficiency, we have performed three types of experiments: synthetic region shapes with synthetic data, synthetic region shapes with natural data, and natural region shapes with natural data. Fig. 9 shows two of the several shapes tested: a regular elliptic shape and a highly irregular non-convex shape that will be called “atol”.

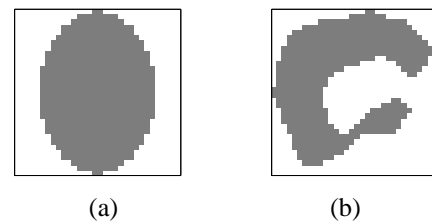


Fig. 9. Synthetic shapes used in experiments: (a) ellipse, and (b) atol with a 32×32 surrounding square.

We compare the performance of various approaches using

the basis restriction (truncation) error

$$e = \sum_{i=1}^{N_S} (x(i) - \tilde{x}(i))^2 \quad (22)$$

where $\tilde{\mathbf{x}} = \mathbf{T}^{-1}\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}$ retains fraction p of the coefficients:

$$\tilde{X}(i) = \begin{cases} X(i) & \text{if } i \leq p \cdot N_S \\ 0 & \text{otherwise} \end{cases}$$

under the assumption that transform coefficients in \mathbf{X} are ordered from highest to lowest energy. We perform the energy ordering of coefficients to eliminate the impact of coefficient scan on the comparison of different transforms; the proposed DCT-based transform is built from regular *as well as* permuting butterflies and thus its coefficients do not correspond to the true frequencies, unlike in the case of the DCT. We express the basis restriction error e in decibels as follows

$$\varepsilon = 10 \log_{10} \frac{\sum_{i=1}^{N_S} x(i)^2}{e} \quad [\text{dB}]. \quad (23)$$

and we use ε to evaluate the energy compaction performance of various algorithms.

In addition to the proposed DCT-based shape-adaptive algorithm, that we shall call the SK algorithm, we have software-simulated (Matlab and C) and tested 6 approaches: KLT, Gilge's approach (GEM), Sikora's approach (SM), DCT with extrapolation based on mirror-image extension (DCT-M), DCT with zero padding (DCT-0), and POCS. Although the DCT-0 method is not a state-of-the-art approach, we included it in our tests since it has the lowest computational complexity of all the methods tested. A few implementation details are in order here. All separable algorithms have been applied either in vertical-horizontal or horizontal-vertical mode; the latter is called subsequently "transposed algorithm". In the GEM approach, Matlab's Gram-Schmidt orthogonalization was used and the subsequent basis functions for orthogonalization were selected by zig-zag scanning in the transform coefficient space. Special care has been taken in the implementation of the SM method to assure transform's orthogonality, which was not the case in the original proposal [19]. Not only does this simplify the computation of the basis restriction error, but also it makes the SM method more efficient [13], [11]. Mirror-image extrapolation for DCT was performed periodically but always in the forward direction (increasing indices); it was done vertically first in the regular mode and horizontally first in the transposed mode. We have implemented a very basic version of the POCS method; we have selected beforehand a triangular zone of coefficients to keep and we did not change it afterwards. This permitted us to generate results for a fixed p only. We initialized the method with mirror-image extrapolation.

Fig. 10 shows the basis restriction error ε as a function of p for ellipse and atol with synthetic data. To generate the synthetic data we have used a 2-D autoregressive (AR) process based on Markov-1 model in horizontal and vertical directions with $\rho_1=\rho_2=0.9$ (2) similarly to the approach undertaken in [18]. The KLT clearly outperforms the other methods. Note

the very close compaction performance of the SM and GEM methods in the case of the ellipse but the better performance of the GEM approach (by about 2dB) for atol. This can be explained by observing that while the GEM method builds true 2-D basis functions, the SM method is separable and by "pushing" the pixels together may mix up distant and unrelated region characteristics. The SK algorithm performs about 1dB below the GEM and SM methods for ellipse, while for atol it attains the SM compaction performance for mild coefficient restriction (high p 's) but loses up to 1dB for severe restrictions (low p 's). DCT-M and DCT-0 perform much poorer compaction at mild restrictions for ellipse; for more severe restrictions, however, DCT-M performs similarly to the proposed method but in this range all differences are small anyway. For atol, DCT-M performs slightly better than SK at severe truncation but loses up to 1dB for mild truncations.

We have tested the same shapes on natural images as well. We have extracted the flower bed from field #0 of interlaced ITU-R 601 sequence "Flowergarden" (Fig. 11). Fig. 12 shows ε for the same algorithms as before except for the KLT⁷. Note that because of the interlace statistical properties of the image in vertical and horizontal directions are very different. For SM, SK and DCT-M approaches two curves are shown (one for regular and one for transposed processing); the GEM approach is non-separable, and DCT-0 does not depend on the direction. Note the high sensitivity of the SM algorithm to the direction of processing; differences of up to 8dB can be observed. Smaller but also very significant sensitivity can be observed for the DCT-M algorithm. The SK method performs very similarly in both direction modes for ellipse and has a much smaller gap than the other two algorithms for atol. Overall SK outperforms DCT-M significantly and performs better (ellipse) or much better (atol) than SM in the worse direction. Interestingly, the better version of the SM algorithm for both shapes and the better version of the SK algorithm for atol outperform the compaction of the GEM approach. This may be due, however, to the fact that for this particular image the selection of basis functions during orthogonalization in the GEM approach was not optimal.

Finally, we have compared the methods for natural region shapes and natural data. Fig. 13.a shows image #6 from QCIF sequence "Carphone" that we have used for experiments. The segmentation from Fig. 13.b has been computed by an advanced method that takes into account both intensity of individual images as well as motion between them [14]. Although far from perfect, this segmentation clearly identifies primary moving objects. Fig. 13.c shows luminance of the face segment containing 2261 pixels. The basis restriction error (as a function of p) for this segment is shown in Fig. 14. While in Fig. 14.a only the better curves are shown over the whole range of p 's, in Fig. 14.b both curves (regular and transposed processing) are shown but in a narrow range of p 's. Note that GEM outperforms SM by about 1dB and SK by about 2dB; POCS, DCT-0 and DCT-M are well below. Interestingly, whereas SM and DCT-M perform better compaction for transposed processing, the SK method does a better job for regular processing. To compare the results

⁷Using basis functions derived from 2-D AR Markov-1 model would do no justice to the method, while there are too few data samples to obtain a reliable estimate of the autocorrelation function.

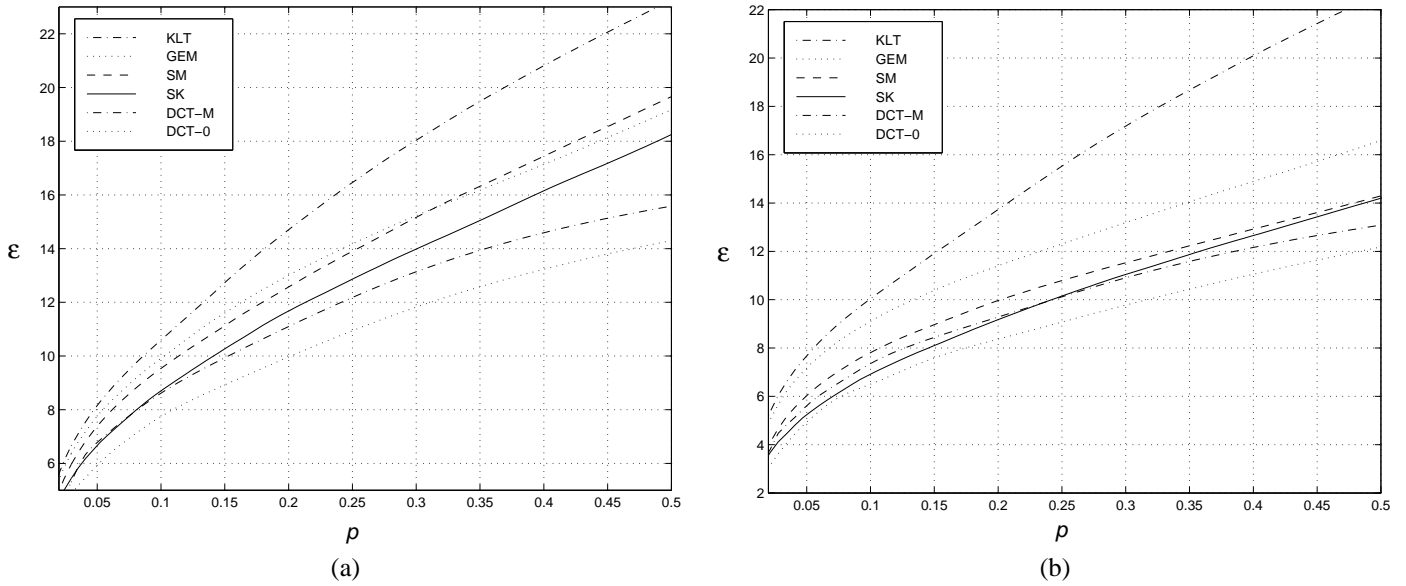


Fig. 10. Basis restriction error ε [dB] as a function of fraction p of highest-energy coefficients retained for 2-D Markov AR process with $\rho_1=\rho_2=0.9$ for (a) ellipse; and (b) atol.

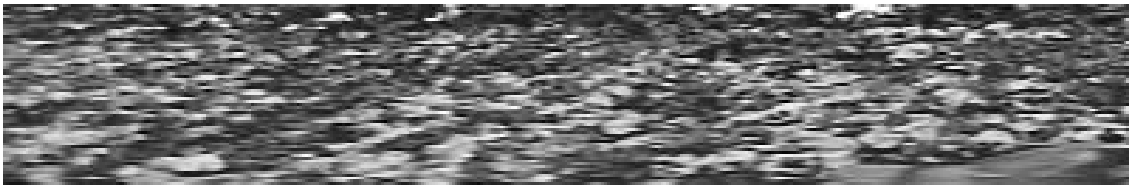


Fig. 11. Flower bed (400×64) from interlaced ITU-R 601 sequence “Flowergarden” (field # 0) used in synthetic shape/natural data experiments. Aspect ratio of the image is not maintained due to the interlace.

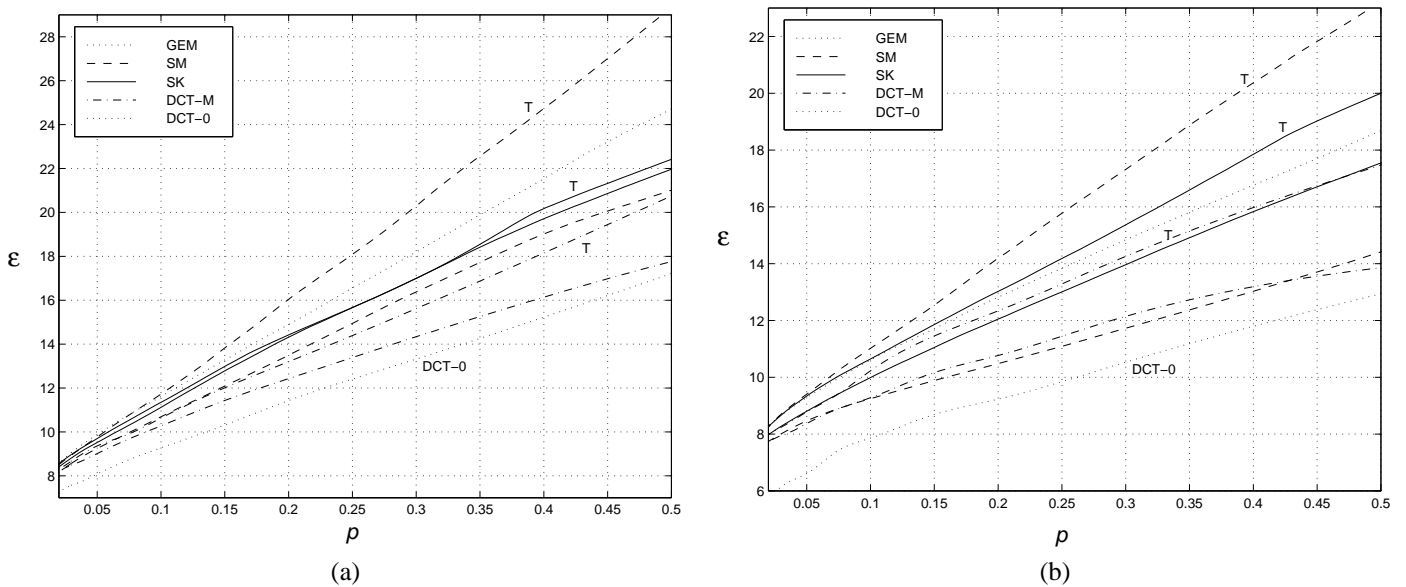


Fig. 12. Basis restriction error ε [dB] as a function of fraction p of highest-energy coefficients retained for the flower bed from “Flowergarden” for (a) ellipse, and (b) atol. Two curves are shown for SM, SK and DCT-M algorithms: one for regular processing and one for transposed processing (curves marked by “T”). Note that the lower dotted curve belongs to the DCT-0 algorithm.

subjectively Fig. 15 shows the reconstructed face segment using the 5 methods after retaining in each case 20%, 10% and 5% of highest-energy coefficients. While the GEM approach gives the best results subjectively, it is computationally impractical. On the other hand DCT-M and especially POCS are too blurred confirming their poor energy compaction properties. As for the SM and SK approaches, in informal subjective viewing on a CRT screen⁸ viewers had difficulty expressing clear preference between the two algorithms; for some p 's they preferred SM while for others they liked SK better. The latter preference was due to sharper edges produced by our algorithm around the mouth, eyes and eyebrows. This can be perhaps explained by the fact that a shift in the first dimension performed in the SM method breaks down the continuity of the orthogonal edges. The resulting jagged edges significantly contribute to the high-frequency content in the transform domain, thus causing a marked distortion upon truncation of the corresponding coefficients. Such a shift is not necessary in the SK method which may explain better edge reproduction.

In the final experiment we have applied the above methods to all segments of "Carphone" while keeping 10% of coefficients in each region (Fig. 16). Note a significant difference in energy compaction, especially for DCT-M, between regular and transposed processing. For the POCS method, the direction of initial mirror-image extrapolation had little impact while DCT-0 is obviously independent of the processing directions. Clearly, SM and SK outperform other approaches⁹ both subjectively and numerically in terms of region-averaged basis restriction error. The transposed processing for both methods seems to be slightly inferior to the regular processing. The SM algorithm has a slight edge in terms of ε and overall quality, although when viewers evaluated images on a CRT screen they could not easily express preference; some viewers again preferred the SK algorithm or judged it equivalent to the SM method due to the improved details at mouth, eye or bow tie boundaries.

As mentioned at the beginning of this section, coefficient indices for the DCT-based shape-adaptive transform do not correspond to the true frequencies, and thus energy ordering of coefficients was used in the experiments. Since an energy-ordered coefficient scan is not practical (coefficient locations must be transmitted), we have also briefly evaluated a zig-zag scan of coefficients similar to that used in the DCT. We have concluded that for natural images the zig-zag scan corresponds very well to the descending order of coefficient amplitudes, however this issue should be explored in depth in the context of rate-distortion performance.

VI. DISCUSSION

Given the impractical computational complexity of the KLT and of the approach proposed by Gilge *et al.* [7], the method developed by Sikora and Makai [19] seems to be currently the most promising approach for shape-adaptive transformation. The method performs very well overall and may even

⁸Due to the reproduction process images shown in this paper are not faithful copies of the originals; proper evaluation should be performed directly on a CRT screen.

⁹We could not run Gilge's algorithm for very large segments (such as the face shown in Fig. 13.b) due to excessive memory requirements of the method.

surpass the performance of a sub-optimal implementation of Gilge's approach. The method has been adapted to MPEG-like compression [17] with interesting results; only boundary blocks (containing more than one region) are processed by Sikora's shape-adaptive transform while other blocks are processed by the standard DCT algorithm. Currently, this transform is being considered for standardization as the texture compression engine in the phase II of MPEG-4 [20], [9]. The method, however, cannot be easily extended to a *true* region-based coding, where each region would be transformed as an entity, since this would require implementation of all 1-D DCTs with lengths from 1 to $\max(K, L)$ for a region with a $K \times L$ surrounding rectangle. For example, a silicon implementation of 100 DCTs with lengths up to 100 would not be practical. Clearly, the method of Sikora and Makai is not a good candidate for a *true* region-based coder.

In contrast, the algorithm proposed in this paper has the complexity of at most two $2^K \times 2^L$ DCTs where $2^K \geq M$ and $2^L \geq N$, and in the particular case of no undetermined samples simplifies to the standard, but sub-optimally implemented, DCT algorithm. For synthetic shapes, the new algorithm's energy compaction performance has been shown to be inferior to that of Sikora's method by about 1-2dB on average. However, for natural shapes and natural data, the performance difference has narrowed to at most 1dB. In fact, subjectively many viewers preferred the new algorithm due to a better detail around clear image structures such as mouth or eyes.

The compaction performance penalty incurred by the new transform is offset to a large extent by its relative insensitivity to the order of processing directions. It may be argued that this order can be adapted to image content as has been proposed for the SM method [3]. The proposed pre-processing, however, does not make the correct decision every time thus allowing a substantial quality drop with every decision error. The new approach does not require such pre-processing, thus simplifying implementation, and will perform more consistently. However, since the pre-processing stage is based primarily on transform's separability, it can be also applied to the new method.

VII. SUMMARY AND CONCLUSIONS

We have presented a new class of fast shape-adaptive orthogonal transforms. We have discussed at length a particular example from this class, namely a shape-adaptive transform derived from a DCT flowgraph. We have compared the computational complexity of the new transform with other well-known shape-adaptive transforms and we have evaluated its energy compaction performance on synthetic and natural data. We have also discussed subjective impressions of informal viewing.

Thus far we have implemented a shape-adaptive algorithm derived from the DCT and we have tested it on a relatively small sample of data. It would be interesting to perform more extensive testing of the algorithm to validate our observations and to study usefulness of the proposed approach for region-based image and video compression. In this context, it would be essential to perform rate-distortion evaluation of the new transform taking into account such issues as coefficient array scanning, coefficient quantization and variable-length coding. Also, it would be interesting to study other algorithms from the

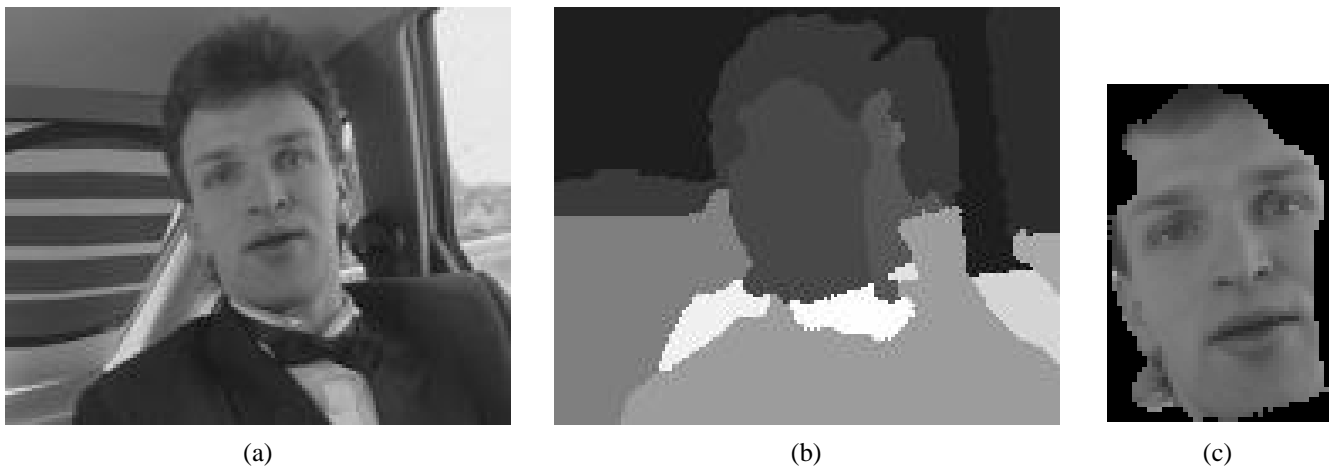


Fig. 13. (a) Image #6 from “Carphone”; (b) its segmentation; and (c) enlarged face segment.

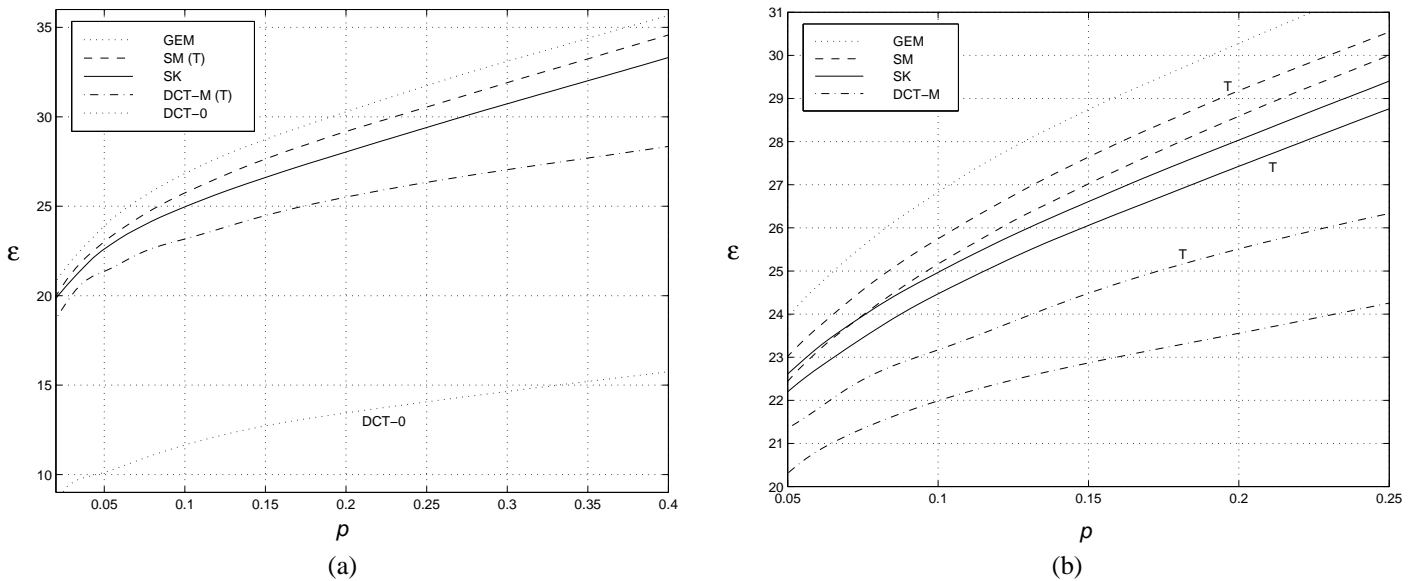


Fig. 14. Basis restriction error ϵ [dB] as a function of fraction p of highest-energy coefficients retained for the face segment extracted from “Carphone”. (a) Only the better curves are shown for the SM, SK and DCT-M algorithms. (b) Both curves are shown (regular processing and transposed processing marked by “T”). Note that the lower dotted curve in (a) belongs to the DCT-0.

same class, e.g., 2-D versions of the DCT (should be insensitive to image anisotropies), DST (some researches advocate its use in image processing), as well as multiplierless transforms (DWHT, DHT) that play an important role for specific classes of images.

ACKNOWLEDGMENTS

We would like to thank Prof. Eric Dubois of INRS-Télécommunications for reading various versions of this manuscript and for valuable comments and discussions on shape-adaptive transforms.

REFERENCES

[1] N. Ahmed and K. Rao, *Orthogonal Transforms for Digital Signal Processing*. Berlin: Springer-Verlag, 1975.
 [2] J. Apostolopoulos and J. Lim, “Coding arbitrarily-shaped regions,” in *Proc. SPIE Visual Communications and Image Process.*, pp. 1713-1726, May 1995.

[3] T. Bayer, W. Büdel, M. Götz, A. Knoll, and P. List, “Modification of the SA-DCT,” ISO/IEC JTC1/SC29/WG11 – MPEG96/0927, July 1996.
 [4] S.-F. Chang and D. Messerschmitt, “Transform coding of arbitrarily-shaped images segments,” in *Proc. ACM Multimedia Conf.*, pp. 83-90, Aug. 1993.
 [5] H. Chen, M. Civanlar, and B. Haskell, “A block transform coder for arbitrarily-shaped image segments,” in *Proc. IEEE Int. Conf. Image Processing*, pp. 85-89, Nov. 1994.
 [6] H. Chen, M. Civanlar, and B. Haskell, “A block transform coder for arbitrarily-shaped image segments,” in *Proc. Int. Workshop on Coding Techniques for Very Low Bit-rate Video*, (Colchester, UK), Apr. 1994.
 [7] M. Gilge, T. Engelhardt, and R. Mehlan, “Coding of arbitrarily shaped image segments based on a generalized orthogonal transform,” *Signal Process., Image Commun.*, vol. 1, pp. 153-180, Oct. 1989.
 [8] G. Golub and C. van Loan, *Matrix Computations*. Johns Hopkins University Press, 1983 (p. 44).
 [9] ISO/IEC JTC1/SC29/WG11, “MPEG-4 video verification model version 8.0.” MPEG97/N1796, July 1997.
 [10] A. Jain, *Fundamentals of Digital Image Processing*. Information and System Sciences Series, New York, NY: Prentice Hall, 1989.
 [11] P. Kauff and K. Schüür, “An extension of shape-adaptive DCT (SA-DCT) towards DC separation and Δ DC correction,” in *1997 Picture Coding*

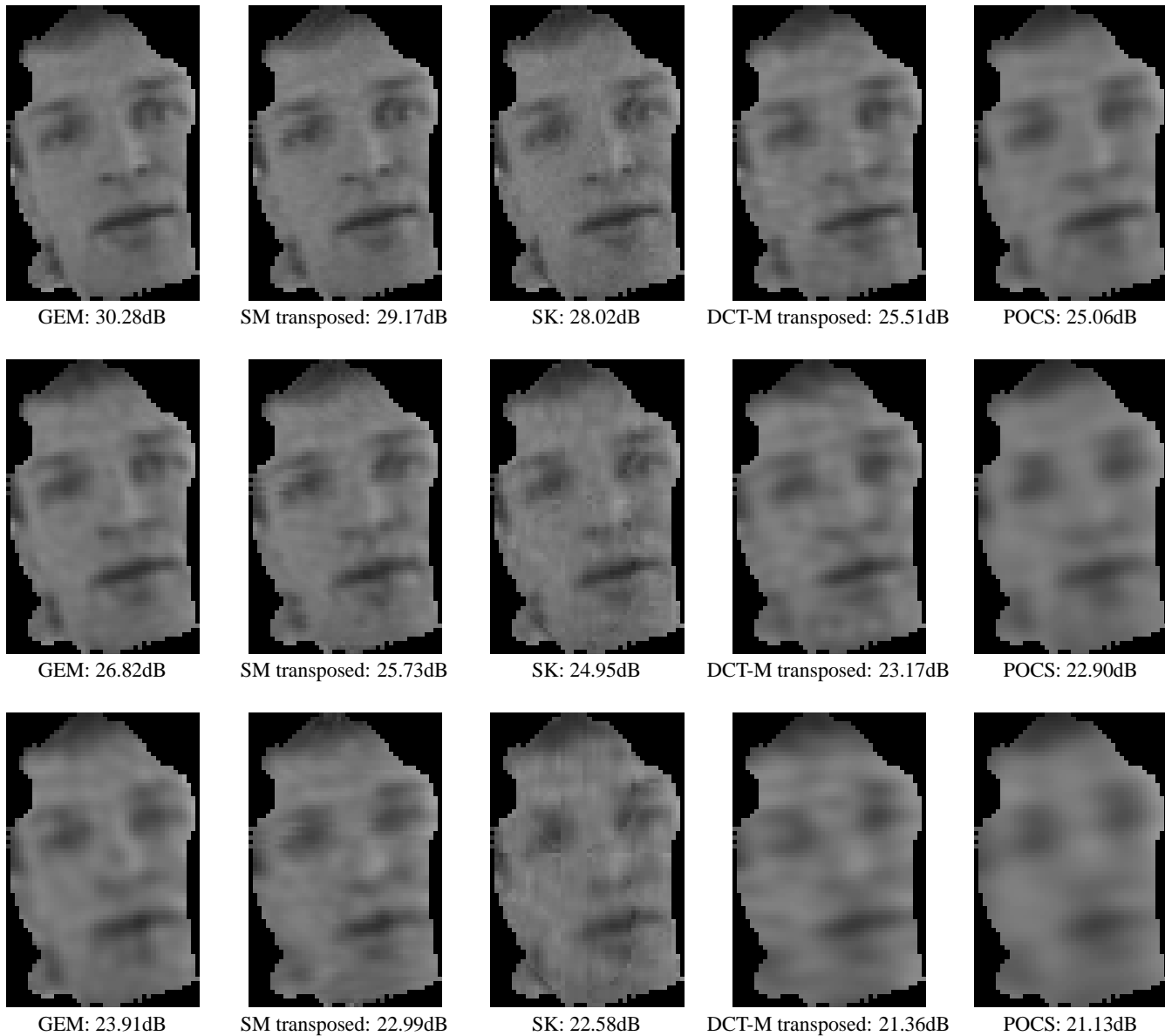


Fig. 15. Face segment after keeping 20% (top), 10% (center) and 5% (bottom) of highest-energy coefficients.

- Symposium*, pp. 647–652, Sept. 1997.
- [12] A. Kaup and T. Aach, “A new approach towards description of arbitrarily shaped image segments,” in *IEEE Int. Workshop on Intell. Sig. Process. and Comm. Systems*, (Taipei, Taiwan), pp. 543–553, Mar. 1992.
- [13] A. Kaup and S. Panis, “On the performance of the shape adaptive DCT in object-based coding of motion compensated difference images,” in *1997 Picture Coding Symposium*, pp. 652–657, Sept. 1997.
- [14] J. Konrad and V.-N. Dang, “Coding-oriented video segmentation inspired by MRF models,” in *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 909–912, Sept. 1996.
- [15] H. G. Musmann, M. Hötter, and J. Ostermann, “Object-oriented analysis-synthesis coding of moving images,” *Signal Process., Image Commun.*, vol. 1, pp. 117–138, Oct. 1989.
- [16] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression*. Plenum Press, 1989.
- [17] T. Sikora, “Low complexity shape-adaptive DCT for coding of arbitrarily shaped image segments,” *Signal Process., Image Commun.*, vol. 7, pp. 381–395, Nov. 1995.
- [18] T. Sikora, S. Bauer, and B. Makai, “Efficiency of shape-adaptive 2-D transforms for coding of arbitrarily shaped image segments,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 254–258, June 1995.
- [19] T. Sikora and B. Makai, “Shape-adaptive DCT for generic coding of video,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 59–62, Feb. 1995.
- [20] Special issue on MPEG-4, *IEEE Trans. Circuits Syst. Video Technol.*, Feb. 1997.
- [21] R. Stasiński, “Alternative algorithms for computing discrete cosine and sine transforms,” in *Int. Symp. on Networks, Systems and Signal Process., ISYNT’89*, (Zagreb, Yugoslavia), pp. 74–77, 1989.
- [22] R. Stasiński, “Optimal DCT algorithms for any data block size,” in *Proc. Nordic Signal Processing Conference NORSIG’95*, (Espoo, Finland), 1995.
- [23] C. Stiller, “Object oriented video coding employing dense motion fields,” in *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing*, pp. V.273–V.276, Apr. 1994.
- [24] C. Stiller and J. Konrad, “Eigentransforms for region-based image processing,” in *Proc. Int. Conf. on Consumer Electronics*, pp. 286–287, June 1995.
- [25] C. Stiller and J. Konrad, “A region-adaptive transform based on a stochastic model,” in *Proc. IEEE Int. Conf. Image Processing*, vol. II, pp. 264–267, Oct. 1995.
- [26] F. Theilheimer, “A matrix version of the fast Fourier transform,” *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 158–161, June 1969.
- [27] P. Venema, “Quantizing the coefficients and internal signals in the syn-



SM: 22.12dB



SM transposed: 22.04dB



SK: 21.59dB



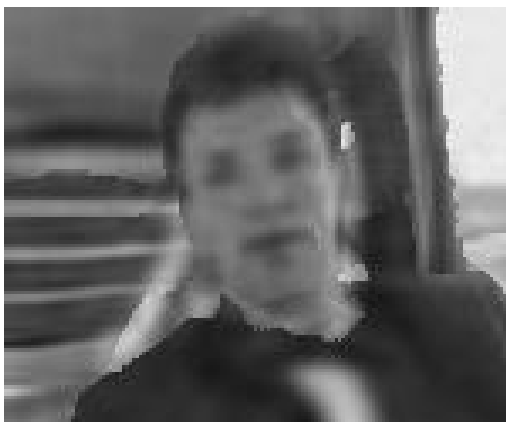
SK transposed: 21.55dB



DCT-M: 18.40dB



DCT-M transposed: 17.58dB



POCS: 18.81dB



DCT-0: 14.34dB

Fig. 16. Combined segments of "Carphone" each with 10% of highest-energy coefficients retained. The basis restriction error ϵ is computed as an average over all regions.

- thesis filterbank of a subband coder for images,” tech. rep., Norwegian Institute of Technology, Aug. 1994.
- [28] M. Vetterli and H. Nussbaumer, “Simple FFT and DCT algorithms with reduced number of operations,” *Signal Process.*, vol. 6, pp. 267–278, 1984.
- [29] Z. Wang, “Fast algorithms for the discrete W transform and for the discrete Fourier transform,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. 32, pp. 803–816, 1984.