IX-th European Signal Processing Conference
EUSIPCO-98, September 8-11, 1998
Island of Rhodes, Greece

FAST QUASI-DCT ALGORITHM FOR SHAPE-ADAPTIVE DCT IMAGE CODING

Ryszard Stasiński[1]
Høgskole i Narvik
Lodve Langes gt. 2
N-8501 Narvik, Norway
rs@tiur.hin.no

Janusz Konrad
INRS-Télécommunications
16 Place du Commerce
Verdun, QC, Canada, H3E 1H6
konrad@inrs-telecom.uquebec.ca

ABSTRACT

In this paper we develop a new variant of the shape-adaptive discrete cosine transform (SA-DCT) recently proposed by Sikora and Makai and currently considered for MPEG-4 as a texture compression engine. We are concerned with the computational complexity of the SA-DCT; although its complexity is acceptable in the context of 8×8 (boundary) blocks as proposed for MPEG-4, it is very high for a *true* region-based coding where complete regions (e.g., 100 by 100 pixels) need to be processed. We adapt the original SA-DCT scheme by replacing the usual DCT with a quasi-DCT for which some basis functions are identical and some similar to those of the DCT. We test the new method and compare it numerically in terms of the basis restriction error as well as subjectively on some natural images. We conclude that the new method's energy compaction performance is slightly inferior to that of the SA-DCT, but its computational complexity is highly reduced.

1 INTRODUCTION

In order to develop efficient region-based image and video compression schemes, several image transforms adapted to shape (other than rectangular) have been recently proposed. One of the promising approaches, due to its simplicity and relatively good performance, is the shape-adaptive discrete cosine transform (SA-DCT) [4]. This scheme is a serious contender as a texture (luminance and chrominance) compression engine in MPEG-4; although not adopted in phase I, it is expected to be included in phase II. SA-DCT performs the computation of 2-D DCT in such a way that only samples inside an irregularly-shaped region are used. Outside samples are omitted thus improving method's performance; intensity transitions at object boundaries, that generate expensive-to-code high frequencies, are excluded from the transformation. For this reason the SA-DCT and other shape-adaptive transforms [7] perform better than methods based on intensity extrapolation to a rectangle followed by a rectangular 2-D DCT [8, 1], such as the method recently adopted in phase I of MPEG-4.

In addition to improved efficiency, an object-oriented approach can be advantageous in applications where an image is treated as a collection of objects (e.g., video compositing, video database querying). MPEG-4 is being currently developed aiming at such applications.

Unfortunately, the SA-DCT has an important drawback when compared with other methods; its asymptotic computational complexity is $0(N^3)$ for an $N \times N$-point data block, in contrast to $0(N^2 \log N)$ asymptotic complexity for the DCT applied to rectangular blocks[2]. In MPEG-4 this is not an issue since SA-DCT is applied only to 8×8 (boundary) blocks. However, SA-DCT would not be practical in a true region-based coder where full regions (corresponding to objects), rather than parts of a block, are treated as an entity. For large regions (e.g., circumscribed by a 100×100 rectangle) the computational complexity of SA-DCT would be prohibitively high. Recently, a computationally-efficient alternative to the SA-DCT has been proposed [7, 8]. The computational complexity of this new DCT-based transform is the same as that of the DCT, i.e., $0(N^2 \log N)$. In informal subjective evaluations on typical videoconferencing material the new technique was judged to give comparable visual quality to that of the SA-DCT, the latter having an edge in energy compaction characteristics [8]. The images processed by the new DCT-based transform are not as smooth as those for SA-DCT, but have less deformed fine patterns; the features are probably due to abrupt changes in amplitudes of DCT-based transform basis functions, and lack of initial image sample shifts characteristic of SA-DCT.

In this paper we propose a new shape-adaptive transform of complexity $0(N^2 \log N)$. This is essentially the same approach as proposed in [4], except that instead of the DCT the so-called quasi-DCT algorithm is used, giving the reduction of computational complexity. Basis functions of the quasi-DCT are in some parts similar while in other parts identical to those of the DCT.

[1] R. Stasiński is on leave from the Institute of Electronics and Telecommunications, Poznań University of Technology, Poznań, Poland.

[2] Although faster SA-DCT realization is possible, its implementation is not very practical.

Hence, they have no abrupt changes in amplitude compared to the basis functions of the DCT-based transform [8, 7]. Additionally, while being orthogonal, the quasi-DCT computes the "true" DC value that is impossible for the original SA-DCT algorithm [2]. Indeed, the presented experimental results show that the SA quasi-DCT method is close to SA-DCT in terms of energy compaction characteristics, and that the difference diminishes when fewer and fewer coefficients are retained.

2 SHAPE-ADAPTIVE DISCRETE COSINE TRANSFORM (SA-DCT)

Let us consider a one-dimensional vector $x(n)$, $n = 1, ..., N$, in which $N_S \leq N$ samples belong to a segment (region), while the remaining samples do not. The one-dimensional shape-adaptive discrete cosine transform (SA-DCT) algorithm can be formulated as follows:

1. shift segment samples to the beginning of the vector in such a way that $x(0), x(1), ..., x(N_S - 1)$ are segment samples,
2. compute the N_S-point DCT for the first N_S samples of the vector - formula (1) below,
3. scale the results; scaling factor $4/N_S$ is used in [4].

Discrete cosine transform algorithms usually compute the non-normalized version of the DCT:

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

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

The method is generalized to two-dimensions in the same way as separable transforms. First, each data row is processed by the 1-D algorithm presented above. Then, the algorithm is applied to each column of the results. The order of computations can be reversed, i.e., the column transformations can precede the row operations, which usually leads to different results [8]. The resulting 2-D transform is non-orthogonal; to make it orthogonal the scaling factor in the last stage of the algorithm should be proportional to $\sqrt{1/N_S}$ [2, 8].

3 QUASI-DCT FORMULATION

The quasi-DCT is defined by an algorithm obtained by a suitable modification of the DCT algorithm proposed in [5]. This is one of the "split-radix" DCT algorithms for 2^n transform sizes that require the smallest known number of arithmetic operations. The generalization of the algorithm to any transform size is done by introducing new substructures to align its flowgraph. The basic "butterfly" of the algorithm is of size 4, hence crucial for program control is the information what is the result of N *modulo* 4 operation (recall that N is the transform size). For N divisible by 4, the first algorithm

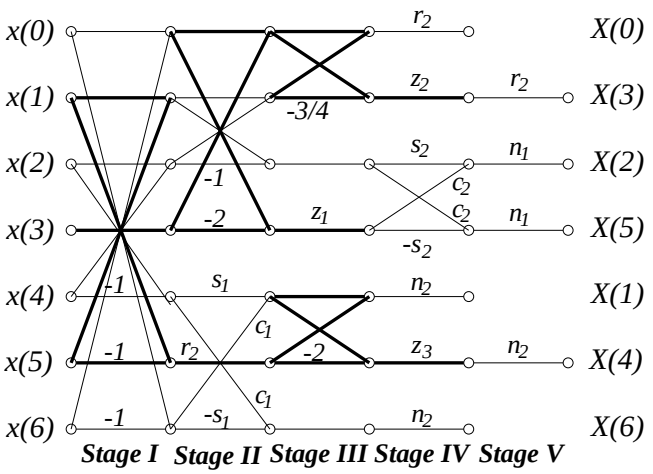

Figure 1: Flowgraph of the 7-point quasi-DCT algorithm. Bold lines indicate the following structures: stage I - "no operation" path for sample $x(3)$, and simplified 2-point butterfly including the multiplication by r_2 in the second stage; stage II - correcting 2-point structure followed by multiplication by z_1 in the third stage; stage III - correcting 2-point structures followed by multiplications by z_2 and z_3 in stage IV. Values of multipliers are: $r_2 = \sqrt{1/2}$, $c_1 = cos(\pi/14)$, $s_1 = sin(\pi/14)$, $c_2 = cos(\pi/8)$, $s_2 = sin(\pi/8)$, $z_1 = \sqrt{2/3}$, $z_2 = \sqrt{4/3}$, $z_3 = \sqrt{1/2}$, $n_1 = \sqrt{7/8}$, $n_2 = \sqrt{7/6}$.

stage is composed entirely of "normal" butterflies (i.e., as defined in [5]). Then, in accordance with the recursive algorithm definition (which is of split-radix type) there are three calls: quasi-DCT subroutine calls itself for $N \leftarrow N/2$, then for $N \leftarrow N/4$, and finally it calls the time-reversed quasi-discrete sine transform subroutine for $N \leftarrow N/4$ [5]. The latter one is obtained from the DST (discrete sine transform) using the same rules as for the quasi-DCT. An example for the most odd case, when N *modulo* $4 = 3$, is shown in Fig. 1 where the current algorithm stage consists of $(N-3)/4$ "normal" butterflies, plus simplified 2-point butterfly (valid also for N *modulo* $4 = 2$), plus "no operation" path for the central data sample (valid when N is odd). The subsequent recursive calls have the following parameters: $N \leftarrow ceiling(N/2)$, $N \leftarrow ceiling(floor(N/2)/2)$, and $N \leftarrow floor(floor(N/2)/2)$; *ceiling* and *floor* are rounding up and rounding down operations. Note that since N is odd, basis functions [1111...] and [1 − 1 − 11...] used for computing $X(0)$, $X(N/2)$ DCT samples for even N are no longer orthogonal. Therefore, two algorithm operations are replaced by special correcting 2-point structures: the bold-marked structure in stage II and the upper bold-marked structure in stage III (Fig. 1). Similarly, the lower correcting structure in stage III is introduced. This operation guarantees that the quasi-DCT is DC-preserving and orthogonal; extensive discussion of this topic can be found in [8]. The coefficients of the operation can be determined from the

N *modulo* 4 value, and from the depth of the recursive procedure call. The range of their values is very limited, hence they can be easily pre-computed and stored in a look-up table. Not shown in Fig. 1 are 3-point DCT and time-reversed DST algorithms necessary for computing quasi-DCT for many values of N, e.g., $N = 3, 5, 6, 9, 10, 11, 12, ...$ They are derived from 3-point real-valued DFT algorithms as outlined in [6].

Similarly to the DCT (1) the sample $X(0)$ of the quasi-DCT always equals the DC component, i.e., it is proportional to the sum of data samples. In general, if $N = q2^s$, where q is an odd number, then 2^s transform basis functions are identical to those of the DCT, with the exception of $q = 3$ when the correct DCT is computed. These are the basis functions for computing $X(0), X(1)$ when $s = 1$, $X(0)$,$X(1)$,$X(2)$,$X(N-1)$ for $s = 2$, then $X(0)$,$X(1)...X(5)$,$X(N-2)$,$X(N-1)$, and so on. Nevertheless, even if N is odd the quasi-DCT basis functions do not differ too much from the DCT ones, at least those for indices close to 0 (and N). For example, the samples of the DCT basis function for computing $X(1)$ when $N = 7$ are $cos(\alpha_n)$ for $n = 1, 2, 3$, where $\alpha_n = \pi/14, 3\pi/14, 5\pi/14, \pi/2$, then $-cos(\alpha_n(6-n))$ for $n = 4, 5, 6$. For the quasi-DCT there is only a slight difference in the definition of angles: $\alpha_n = \pi/14, \pi/4 = 3.5\pi/14, 6\pi/14, \pi/2$, and in the function amplitude (equal to $\sqrt{(7/6)}$). The deviations in α_n values decrease with the growing N.

The basis function for computing $X(2)$ for odd N is similar to that for the DCT, except for a strong "spike" in its center. The spike is due to the multiplication of the central data sample by -2 in the second stage of the algorithm (Fig. 1). Unfortunately, since this multiplication guarantees both the correct computation of the $X(0)$ sample *and* the transform orthogonality [8, 7], it cannot be easily discarded. With the growing index of quasi-DCT, basis functions become more and more deformed; such correcting multipliers produce more spikes in the basis functions. Fortunately, the high-index quasi-DCT samples are of lower importance for image coding.

When constructing a 2-D algorithm from the 1-D version additional advantages of the quasi-DCT become clear. Since the algorithm is given by its flowgraph, we can tailor the 1-D quasi-DCT algorithm computing the sample $X(0,0)$ in such a way that the sample is proportional to the sum of image samples ("true" DC component) [8]. This is related to the use of correcting operations of the type shown in Fig. 1, guaranteeing orthogonality of the transform. In contrast, in the case of the original SA-DCT [2] only one of the two transform features, either DC preservation or orthogonality, can be retained. It should be underlined here that both the correct DC computation and the transform orthogonality have been shown to be important for obtaining high performance algorithms for image coding [8, 7, 2, 3].

4 EXPERIMENTS

The new SA quasi-DCT method is compared with the orthogonal version of the SA-DCT algorithm [2, 3], and with the recently proposed DCT-based shape-adaptive transform [8, 7]. The transformed shape is the "face" segment from image #6 from QCIF sequence "Carphone", used in [8]. The processed "face" segments can be seen in Fig. 3; in each case after the forward transformation only a fraction p of highest-energy coefficients was retained before the inverse transform. To compare the results numerically we use, similarly as in [8, 7], the basis restriction (truncation) error

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

expressed in dB ($10 \log_{10}(\sum_{i=1}^{N_S} x^2(i)/e)$), where $\tilde{\mathbf{x}} = \mathbf{T}^{-1}\tilde{\mathbf{X}}$ ($\mathbf{T}$ is the transform tested and $\mathbf{T}^{-1}$ is its inverse) and $\tilde{\mathbf{X}}$ retains fraction p of the coefficients:

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

under the assumption that transform coefficients in $\mathbf{X}$ are ordered from highest to lowest energy. The basis restriction error curves are shown in Fig. 2. As mentioned before, the results depend, in general, on the order of processing; column transformations followed by row ones, or row transformations followed by column ones. We show both results in Fig. 2. The better (higher) curves for SA-DCT and SA quasi-DCT are for row processing performed first, whereas the better curve for the DCT-based transform is for column processing performed first. As can be seen, for high p values (i.e., few coefficients discarded) the SA-DCT has the best energy compaction performance, with the curves placed 1dB above those for the DCT-based transform and 1.5-2dB above those for the SA quasi-DCT. For low p's, however, all curves are very close to each other.

To compare the results subjectively Fig. 3 shows the reconstructed "face" segment using the 3 methods after retaining in each case 20%, 10% and 5% of highest-energy coefficients. As it was reported in [8], in informal subjective evaluation viewers had difficulty expressing clear preference between the SA-DCT and DCT-based transforms, and the effect varied with the value of p. As for the results reported here, for higher values of p the images for the SA-DCT are slightly smoother than those for the SA quasi-DCT which are more granular (although this may not be clear from images printed here, the effect is obvious on a CRT screen). This is to be expected because of the initial data shift; both methods introduce similar line distortions, while SA quasi-DCT also adds a "salt and pepper"-type noise. The noise can be related to spikes that appear in high-index quasi-DCT basis functions (Section 3). For low values of p, however, the visual differences are very small, which confirms the narrowing gap in Fig. 2.

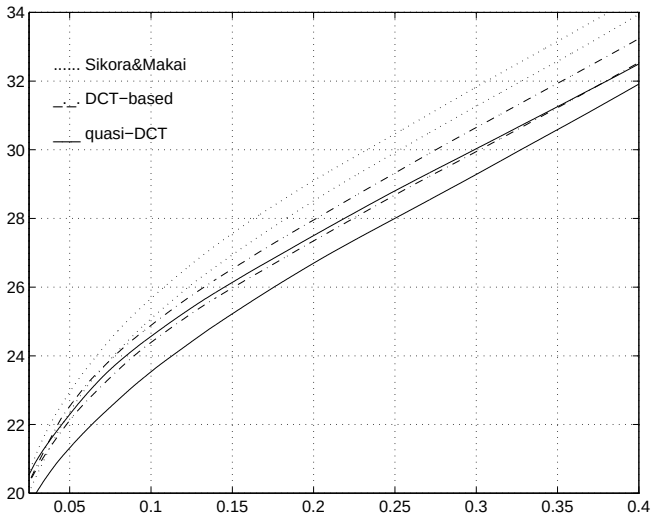

Figure 2: Basis restriction error expressed in dB as a function of the fraction p of highest-energy coefficients retained.

5 CONCLUSIONS

A fast version of the SA-DCT transform proposed by Sikora and Makai and considered for MPEG-4 has been derived and evaluated in the paper. The transform, called SA quasi-DCT, requires $0(N^2 \log N)$ operations for an $N \times N$-point data block, as opposed to $0(N^3)$ operations for SA-DCT. The reduced complexity is obtained by replacing 1-D DCTs in the transform formulation by the quasi-DCTs. Basis functions of the quasi-DCT transform are partly identical and partly similar to those of the DCT. Experimental results show that the energy compaction properties of the new transform are nearly as good as those of the SA-DCT, especially when only few transform coefficients are used for image reconstruction. This has been confirmed both numerically and subjectively. In summary, the SA quasi-DCT transform may be an interesting alternative to the SA-DCT in *true* region-based coding where efficient processing of large regions as an entity is essential.

REFERENCES

[1] S.-F. Chang and D. Messerschmitt, "Transform coding of arbitrarily-shaped images segments," in *Proc. ACM Multimedia Conf.*, pp. 83–90, Aug. 1993.

[2] P. Kauff and K. Schüüer, "An extension of shape-adaptive DCT (SA-DCT) towards DC separation and ΔDC correction," in *1997 Picture Coding Symposium*, pp. 647–652, Sept. 1997.

[3] 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.

[4] 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.

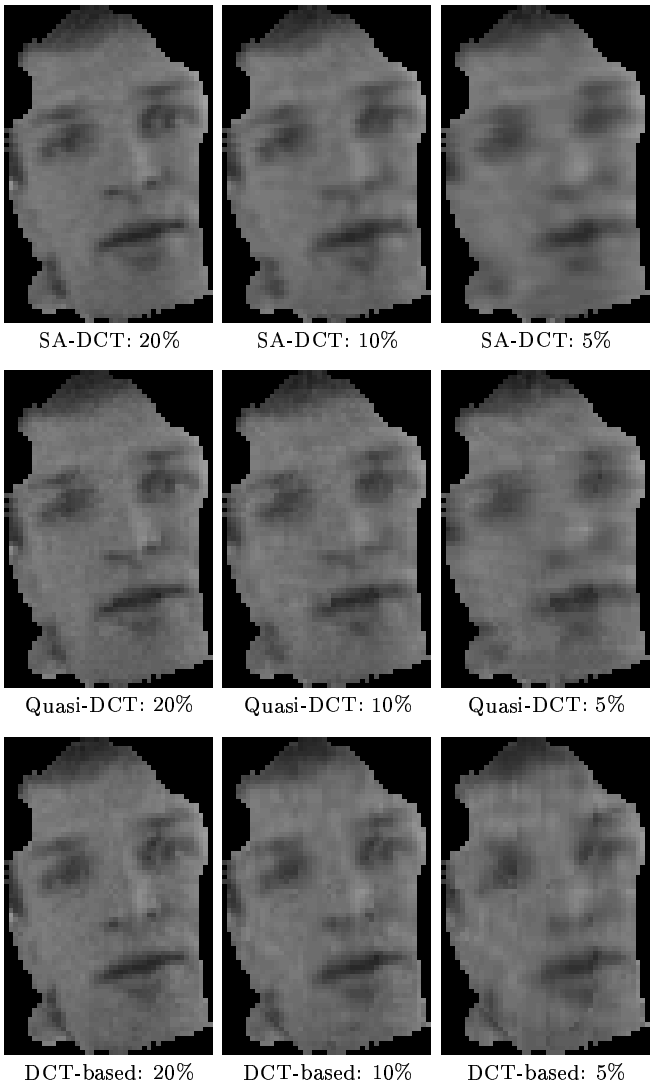

Figure 3: Face segment after processing by SA-DCT, quasi-DCT and DCT-based transforms and keeping 20%, 10% and 5% of highest-energy coefficients.

[5] 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.

[6] R. Stasiński, "Optimal DCT algorithms for any data block size," in *Proc. Nordic Signal Processing Conference NORSIG'95*, (Espoo, Finland), 1995.

[7] R. Stasiński and J. Konrad, "DCT-based shape-adaptive transform for region-oriented image compression and manipulation," in *Workshop on Image Analysis for Multimedia Interactive Services*, (Louvain, Belgium), June 1997.

[8] R. Stasiński and J. Konrad, "A new class of fast shape-adaptive orthogonal transforms and their application to region-based image compression," *IEEE Trans. Circuits Syst. Video Technol.*, submitted Oct. 1997.