Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'97, June 24-25, 1997, Louvain, Belgium

DCT-BASED SHAPE-ADAPTIVE TRANSFORM FOR REGION-ORIENTED IMAGE COMPRESSION AND MANIPULATION

Ryszard Stasiński

Poznań University of Technology
Inst. of Electronics and Communiations
ul. Piotrowo 3a
60-965 Poznań, Poland
rstasins@et.put.poznan.pl
tel: +48 61 782-631, fax: +48 61 782-572

Janusz Konrad *

INRS-Télécommunications
Inst. Nat. de la Recherche Scientifique
16 Pl. du Commerce
Verdun, QC, H3E 1H6, Canada
konrad@inrs-telecom.uquebec.ca
tel: +1 514 765-7768, fax: +1 514 761-8501

Abstract. In the paper a new DCT-based shape-adaptive transform algorithm is presented. The transform is derived from the DCT algorithm flowgraph by substitution of operations in such a way that region and background samples are not mixed together. The computational complexity of the algorithm is of the same rank as that of the DCT and significantly lower than that of the state-of-the-art shape-adaptive transforms. Preliminary experiments show that the new algorithm performs better than the direct DCT (with extrapolation) and is only slightly inferior to the approaches of Gilge *et al.* and of Sikora and Makai.

1 INTRODUCTION

The widely-used uniform partitioning of image data into rectangular blocks is not well-suited for emerging applications in image compression and processing. Although simple and easily implementable, such partitioning results in visible block boundaries when bit rate is severely limited and also does not lend itself naturally to object-wise manipulation (extraction, insertion, repositioning of objects). As an alternative, non-uniform (arbitrarily-shaped) image partitioning has been actively studied over the last few years. Such a partitioning is expected to improve compression quality for a given bit rate (error is uniformly distributed over the whole region), but perhaps more importantly it is also expected to assure additional functionalities, e.g., object-by-object progressive transmission, object-based manipulation.

In this paper we present a new approach to devising a shape-adaptive transform. Derivation of the transform starts with a DCT algorithm flowgraph. This flowgraph is subsequently modified so that data samples within a region are not mixed with those outside (background). The resulting shape-adaptive algorithm has the same computational complexity as the original DCT algorithm. At the same time, the new shape-adaptive transform has good data compression capabilities, much better than the standard DCT with zeroed background samples [1], and almost as good as shape-adaptive transforms proposed by Gilge *et al.* [2] and Sikora and Makai [3].

* This work was partially supported by the Fonds pour la formation de chercheurs et l'aide à la recherche, Québec, Canada under research grant 96-ER-1577, and research grant BW/44/486, Poznań University of Technology.

2 ALGORITHM DESCRIPTION

Our goal is to redefine a DCT algorithm flowgraph in such a manner that data samples in the region and outside are not mixed. It is reasonable to add an additional requirement that background samples be not processed at all; they are irrelevant for transformation results. Let us start with the observation that fast DCT algorithm flowgraphs for data vectors of size 2^k consist of two types of structures (butterflies) only:

$$\begin{aligned} y_a &= x_a + x_b \\ y_b &= x_a - x_b, \end{aligned} \tag{1}$$

where x_a, x_b, y_a, y_b are input, intermediate or output data samples, respectively, and

$$\begin{aligned} y_a &= x_a \cdot \sin(\phi) + x_b \cdot \cos(\phi) \\ y_b &= x_a \cdot \cos(\phi) - x_b \cdot \sin(\phi) \end{aligned} \tag{2}$$

where ϕ is a rotation angle, plus some scalar multiplications (Fig. 1.a). If either x_a or x_b belongs to the background, we can replace any of the above structures by one of the following "permuting" butterflies:

$$\begin{aligned} y_a &= x_a \\ y_b &= x_b, \end{aligned} \quad \text{or} \quad \begin{aligned} y_a &= x_b \\ y_b &= x_a. \end{aligned} \tag{3}$$

Clearly, our objective of not mixing region and background samples has been achieved. Extension of the above approach to the whole DCT flowgraph is immediate. As a result we obtain all background samples repeated at the output of the shape-adaptive transform. The remaining transform samples describe the content of the region; they result from combinations of region samples computed by unchanged substructures of the "parent" DCT flowgraph.

Unfortunately, such a simple approach to shape-adaptive transform generation causes that the transform is not orthogonal. This is due to the fact that while butterflies (2,3) are orthogonal, the butterfly (1) is not. However, this butterfly can be represented as an orthogonal butterfly

$$\begin{aligned} y_a &= x_a \cdot \sin(\pi/4) + x_b \cdot \cos(\pi/4) \\ y_b &= x_a \cdot \cos(\pi/4) - x_b \cdot \sin(\pi/4) \end{aligned} \tag{4}$$

followed by multiplications by $\sqrt{2}$. Therefore, we can replace the orthogonal butterfly (4) by one of the "permuting" butterflies (3), but we should not remove the following multiplications by $\sqrt{2}$. The background samples need not be multiplied as their values at the transform output are irrelevant.

The introduction of multiplications by $\sqrt{2}$, however, prevents the algorithm from computing the "true" DC component of the processed region. This is an important drawback as in natural images the DC component is much larger than other spectral components. We will analyze the computation of the DC component for FFT; in the case of DCT it is performed in the same way, but on differently ordered data. In FFT the DC component, i.e. sample $X(0)$, is computed jointly with sample $X(N/2)$ in a butterfly (1), where $X(0) = y_a$, $X(N/2) = y_b$. Samples x_a and x_b are equal to:

$$x_a = \sum_{n \text{ even}} x(n), \quad \text{and} \quad x_b = \sum_{n \text{ odd}} x(n),$$

where $x(n)$ are data samples. If some data samples belong to the background, the above formula changes as follows

$$x_a = \sum_{n \text{ even}} \delta(n) \cdot x(n), \quad \text{and} \quad x_b = \sum_{n \text{ odd}} \delta(n) \cdot x(n), \tag{5}$$

where $\delta(n)$ excludes background samples:

$$\delta(n) = \begin{cases} 1 & \text{if } x(n) \text{ belongs to object,} \\ 0 & \text{if } x(n) \text{ belongs to background.} \end{cases} \tag{6}$$

Therefore, the transform basis functions used for computing samples $X(0)$ and $X(N/2)$ may be described as vectors: $[\delta(0), \delta(1), \delta(2), \ldots, \delta(N-1)]$, and $[\delta(0), -\delta(1), \delta(2), \ldots, -\delta(N-1)]$, respectively; N is the transform size. Inner product of these vectors is equal to $\alpha(a) - \alpha(b)$, where $\alpha(a), \alpha(b)$ are numbers of object samples added in (5) for computing variables x_a and x_b, respectively. However, if we correct the butterfly (1) as follows:

$$\begin{aligned} y_a &= x_a + x_b \\ y_b &= x_a - w \cdot x_b \end{aligned} \tag{7}$$

with $w = \alpha(a)/\alpha(b)$, the inner product becomes zero, i.e., the two basis functions turn into mutually orthogonal ones.

In the preceding stage of the transform flowgraph samples x_a and x_b are calculated in the same manner as sample $X(0)$. Namely, x_a is obtained as a sum of

$$\sum_{n \bmod 4=0} \delta(n) \cdot x(n), \quad \text{and} \quad \sum_{n \bmod 4=2} \delta(n) \cdot x(n), \tag{8}$$

while x_b is a sum of

$$\sum_{n \bmod 4=1} \delta(n) \cdot x(n), \quad \text{and} \quad \sum_{n \bmod 4=3} \delta(n) \cdot x(n) \tag{9}$$

computed in two different butterflies of the form (1). In both butterflies differences between the above sums are computed as well in analogy to $X(N/2)$. Thus, we may talk about basis functions for computing x_a and its companion sample in the butterfly. Basis functions associated with those two samples are: $[\delta(0), \delta(2), \delta(4), \ldots, \delta(N-2)]$, and $[\delta(0), -\delta(2), \delta(4), \ldots, -\delta(N-2)]$. Their inner product is equal to $\alpha(a) - \alpha(b)$, where a and b are input nodes of the butterfly. We may orthogonalize these functions by replacing butterfly of the form (1) by the butterfly of the form (7). The same concerns the second butterfly in which sample x_b is computed.

Then, the sums in (8) and (9) are computed by four butterflies in the preceding stage from eight sub-sums in which data samples are summed up for indices $n \bmod 8 = 0, 1, 2, \ldots, 7$. The same reasoning as above lead us to the conclusion that butterflies should be corrected as in (7). The induction stops at the first stage of the algorithm.

Notice that because some $\delta(n)$ are equal to zero, the basis functions described by vectors $[\delta(i), \pm\delta(K+i), \delta(2K+i), \ldots, \pm\delta(N-K+i)]$ have lost part of their energy in comparison with those of the non-adaptive algorithm (i is the index of a butterfly in a stage and $K = 2^j$ where j is the stage number counted backwards, i.e., j=0 (K=1)

for the last stage of the algorithm). A correction for this loss can be acomplished by multiplying y_b in (7) by:

$$z = \sqrt{\frac{2^k}{\alpha_a + w^2 \cdot \alpha_b}} = \sqrt{\frac{2^k}{\alpha_a \cdot (1+w)}} = \sqrt{\frac{2^k}{(\alpha_a + \alpha_b) \cdot w}} \tag{10}$$

where k is the stage number in which the butterfly is performed ($k = 1, \ldots, \log_2 N$), and by multiplying $x(0)$ by $v = \sqrt{N/2 \cdot N_S}$, where N_S is the total number of object samples. After these corrections the flowgraph substructure involved in the computation of $X(0)$ may replace the analogous part of the non-adaptive DCT algorithm.

Fig. 1 shows flowgraph of the 8-point DCT algorithm from [4], and its shape-adaptive counterpart. As can be seen, outside the substructure involved in the computation of $X(0)$, orthogonalization of the algorithm is done by the use of $\sqrt{2}$ multipliers. It is desirable that the background samples be moved to output positions for which indices are as high as possible. The strategy consists of using second permuting butterfly (3) whenever x_a belongs to background. For $N > 8$ the strategy is not always optimal, but exceptions are few, at least for small and moderate N values.

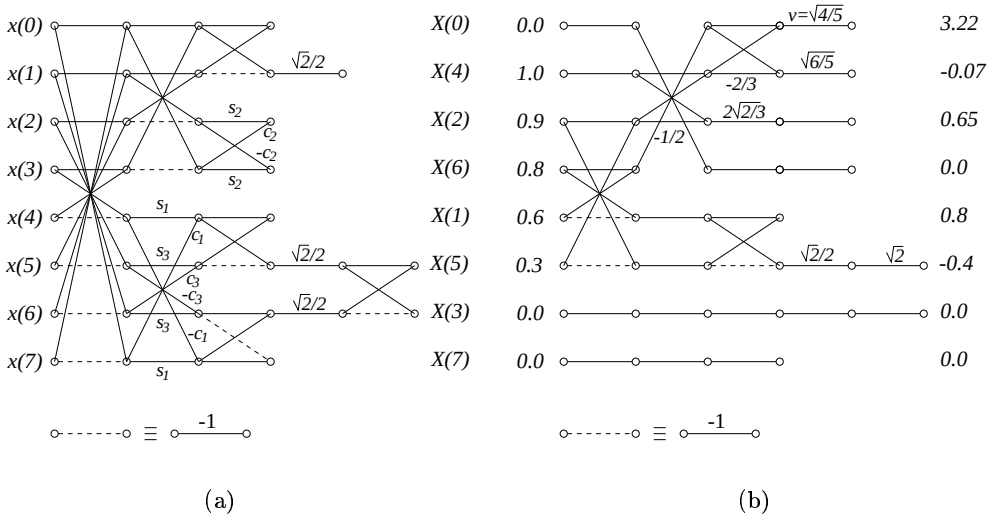

Fig. 1. Flowgraphs of the (a) 8-point DCT algorithm ($s_i = \sin \pi i/16$, $c_i = \cos \pi_i/16$); and (b) 8-point shape-adaptive DCT-based algorithm developed for an input signal in which background samples are set to zero. For easier comparison redundant flowgraph structures in (b) are not removed.

Notice that the computational complexity of the new algorithm is of the same rank as that of the DCT. There are $0(N \log N)$ additional logical operations associated the with switching between algorithm structures. Our conservative estimate is that the new algorithm requires no more than twice the number of operations used by the DCT algorithm.

3 EXPERIMENTAL RESULTS

The 2-D version of the shape-adaptive DCT-based algorithm has been simulated in software and compared with other extrapolative and shape-adaptive techniques. The experiment consisted of picking 32×32-pixel blocks from an image, windowing them by an artificial C-shaped mask, and transforming regions generated in this way. The image data have been taken from the flower bed from field #0 of ITU-R 601 sequence "Flowergarden". The methods have been compared on the basis of averaged basis restriction error, i.e., the error of a signal representation given by the largest transform coefficients.

In addition to the proposed DCT-like shape-adaptive algorithm we have simulated in software(Matlab and C) and tested 4 approaches: Gilge's approach [2], Sikora's approach [3], DCT with extrapolation based on mirror-image extension and zero padding. All separable algorithms have been applied either in vertical-horizontal or horizontal-vertical mode; the latter is called subsequently "transposed algorithm". In Gilge's approach Matlab's Gram-Schmidt orthogonalization was used and subsequent basis functions for orthogonalization were selected by zig-zag scanning of transform coefficients. 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.

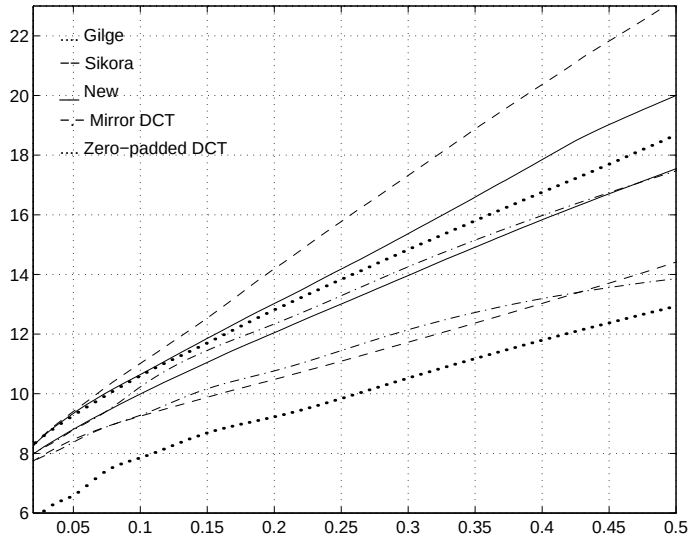

Fig. 2. Basis restriction error expressed in dB as a function of the fraction of highest-energy coefficients retained for the flower bed from "Flowergarden" for a C-shaped region. The Sikora's, new and mirror-DCT algorithms are shown for two different orders of directions (vertical followed by horizontal and transposed, i.e., horizontal followed by vertical). The lower dotted curve belongs to the DCT algorithm with zero padding.

Results of our experiments are shown in Fig. 2. For Sikora's, new and mirror-DCT approaches two curves are shown: one when vertical direction of processing is selected first and the other when the horizontal direction is used first (transposed); Gilge's approach is non-separable, and zero-padding DCT does not depend on the direction. Note the high sensitivity of Sikora's algorithm to the direction of processing; differences of up 8dB can be observed. Smaller but also very significant sensitivity can be observed for the mirror-DCT algorithm. Our algorithm, however, performs very similarly in both direction modes having a much smaller gap between curves than the other two algorithms. Overall our algorithm outperforms mirror-DCT significantly and performs much better than Sikora's algorithm in the worse direction. Interestingly, the better versions of Sikora's and our algorithm outperform the Gilge's approach. This may be due, however, to the fact that for this particular image the selection of basis functions during orthogonalization in Gilge's approach was far from optimal.

It should be pointed out here that computational complexity of Gilge's method is much greater than that of other methods compared in Fig. 2. It is of the order of $O(N^3)$ for one-dimensional vectors, which gives $O(N^6)$ for $N \times N$-point ones. In comparison, the complexity of Sikora's method computed from the DCT definition for an $N \times N$ block is of the order of $O(N^3)$. On the other hand, this is much more than for the DCT and the new method, for which the complexity is of the rank $O(N^2 \log_2 N)$. Nevertheless, the Sikora's method is usually applied to block sizes not greater than 8×8 for which differences between algorithm complexities are not critical.

4 CONCLUSION

A new DCT-based shape-adaptive transform intended for region-oriented image compression and manipuation has been presented in the paper. The transform is derived from the DCT algorithm flowgraph by redefining its substructures in such a way that object and background samples are processed separately. Special care has been taken to maintain transform orthogonality and proper calculation of the object's DC component. The algorithm has been tested on natural images from which artificial C-shaped regions have been extracted. Those initial results show only slightly inferior performance of our algorithm when compared with the approaches of Gilge *et al.* [2] and of Sikora and Makai [3], however at much lower complexity. Note that although Sikora and Makai's method has low complexity for small regions, the method becomes complex for large regions treated as one entity.

References

1. S.-F. Chang and D. Messerschmitt, "Transform coding of arbitrarily-shaped images segments," in *Proc. ACM Multimedia Conf.*, pp. 83–90, 1993.
2. 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.
3. 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.
4. 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.