ON COMPUTING THE 2-D EXTENDED LAPPED TRANSFORMS

Dragutin Šević[1] *and Miodrag Popović*[2]

[1] Institute of Physics, Pregrevica 118, 11080 Zemun, Belgrade, Yugoslavia
E-mail: sevic@atom.phy.bg.ac.yu
[2] Faculty of Electrical Engineering, P.O. Box 816, 11001 Belgrade, Yugoslavia
E-mail: pop@el.etf.bg.ac.yu

ABSTRACT

In this paper a new implementation of the two-dimensional Extended Lapped Transform (2-D ELT) is proposed. Compared to the separable solution, proposed by Malvar [1], the new realization of 2-D ELT has reduced arithmetic complexity. Computational savings are achieved because scaling and inverse scaling of butterfly matrices, suggested by Malvar for 1-D case, are, after some modifications of the basic separable algorithm, extended to 2-D case. The new implementation has the same frequency response as Malvar's.

1. INTRODUCTION

For the sake of simplicity and to achieve computational savings, 2-D transforms are often implemented as separable operators, in two steps: all the rows in the block are transformed with a 1-D transform, and then all columns in the transformed block are transformed with the same 1-D transform (or vice versa, result is the same). In this paper we describe a further computational optimization of 2-D separable extended lapped transform (ELT), based on fully optimized 1-D ELT, proposed by Malvar [1]. Results of our numerous image coding simulations, contrary to results presented in [1], fulfilled the expectations based on theory: there are improvements in coding results when overlapping factor K of ELT is increased, or when ELT with $K > 1$ instead of MLT or LOT is used. However, for coding simulations, instead of 256×256 image as in [1], we used 512×512 image. To avoid border effects, we used periodic extension of the image.

The fast algorithms for the ELT are based on FFT algorithm for computation of DCT-IV operator, firstly proposed by Duhamel *et al.* [2]. Duhamel's algorithm includes input and output rotations, with butterfly matrices very similar to window butterfly matrices of ELT's.

This work has been supported by the Ministry of Science and Technology of Republic of Serbia.

2. THE FFT BASED IMPLEMENTATION OF THE ELT

The structures for the ELT analysis filter bank and the ELT synthesis filter bank are shown in Fig. 1. Because of the orthogonality of ELT, the synthesis filter bank is the transpose of the analysis filter bank.

The FFT implementation of ELT is derived in the following way: using the approach of Duhamel *et al.* [2], one can suppose that the window has been applied to the signal, and concentrate on the central part of ELT, the transform itself. Malvar's approach was slightly different: he has concentrated on DCT-IV operator. Since the derivation of the TDAC transform is the straight repetition of the work performed in [2] and [3] it will not be presented here.

The implementation of TDAC transform is shown by a flowgraph in Fig. 1. It should be emphasized that the output rotation by angle $\theta = 0$ uses no real operations, and the rotation by $\theta = \pi/4$ uses only 2 real multiplications and 2 real additions. The FFT is optimally implemented using split-radix algorithm. For the number of bands $M \leq 16$, the FFT could be optimally implemented with a single stage radix-2, 4, or 8 algorithm, which means no indexing. Duhamel *et al.* [2] showed that the TDAC is self-inverse transform, so there is no need to derive the inverse TDAC transform.

Using the angle values from Table D.3 [1], this filter bank has the same frequency response as ELT from [1] (disregarding some irrelevant channel multiplications by -1). The angle values θ_k^i can be read directly from Table D.3 in [1]. However, the values for θ_k^0 should be obtained by following relations:

$$\theta_k^0 = \hat{\theta}_{2k}^0, \quad k = 0, 1, ..., M/4 - 1 \qquad (1)$$

$$\theta_k^0 = \pi/2 - \hat{\theta}_{M-2k-1}^0, \quad k = M/4, ..., M/2 - 1 \qquad (2)$$

where $\hat{\theta}_k^0$ are the angles from Table D.3 [1]. This permutation is essential to achieve the necessary reordering of elements for TDAC transform. On the other hand, in programs for ELT's proposed by Malvar [1], data unshuffling steps were moved outside of recursive

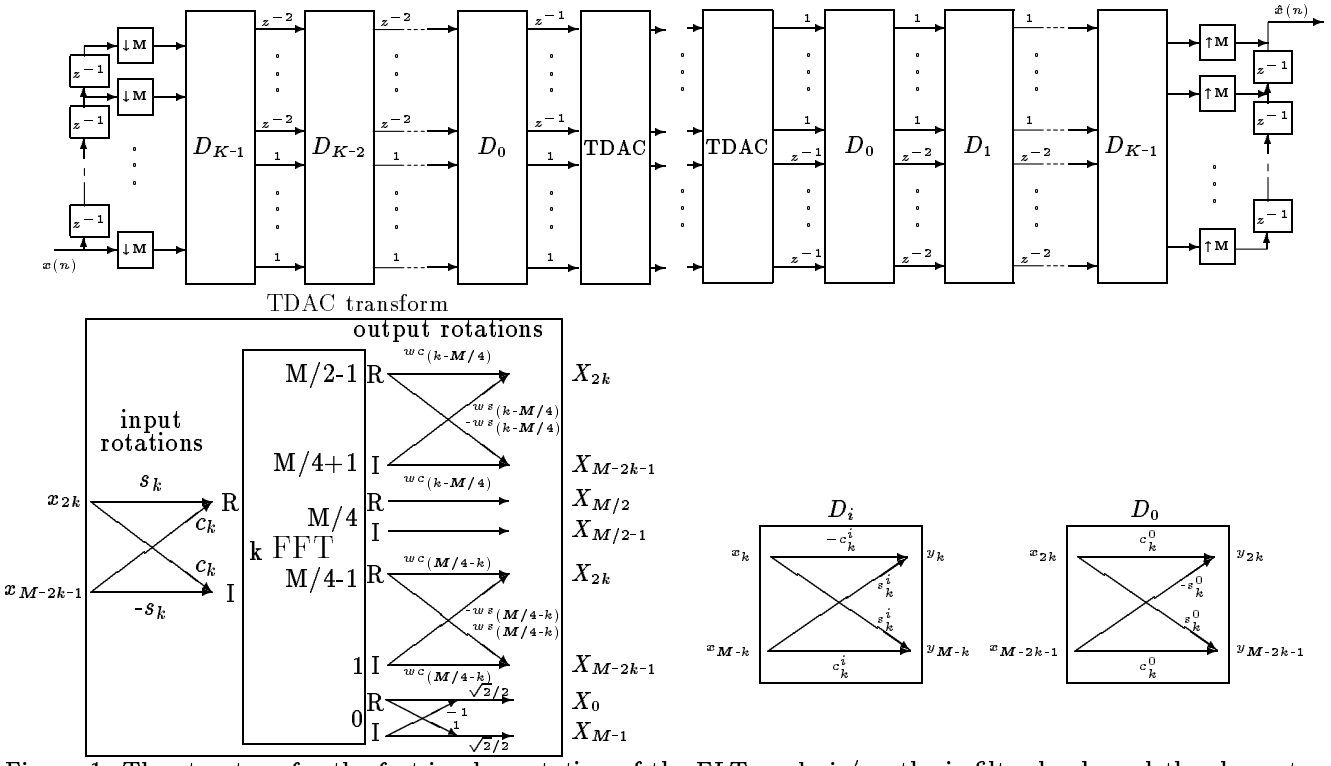

Figure 1: The structure for the fast implementation of the ELT analysis/synthesis filter bank, and the elements of ELT structure: TDAC transform and butterflies. $s_k = \sin\theta_k$, $c_k = \cos\theta_k$, $\theta_k = (4k+1)\pi/(4M)$, $ws_k = \sin(k\pi/M)$, $wc_k = \cos(k\pi/M)$. Note that the butterflies D_i's and D_0 are different.

modules. In our approach, however, it was easier to recognize full possibilities of scaling and inverse scaling of butterflies matrices.

3. SCALING OF BUTTERFLY MATRICES

As proposed by Malvar [1], all the coefficients in the cascade of window butterflies could be scaled, so that diagonal entries would be equal to 1 or -1, and necessary inverse scaling would be applied to the last butterfly in cascade (D_0). Computational complexities of ELT's in Table 5.1 [1] correspond to this way of scaling butterflies. However, looking at the Fig. 1, it is easily perceived that the inverse scaling could be applied to the input rotations of FFT based DCT-IV realization, for all possible numbers of bands, M. This first step in optimization procedure, saves one multiplication per sample. If butterflies are realized as 3/3, then saving is equal to 0.5 multiplications and 0.5 additions per sample. It should be noted here that there is a subtle computational difference between the MLT [1], which uses the sine window, and ELT with overlapping factor $K = 1$, which uses butterfly angles given in [1]. Because of similar frequency responses, MLT is usually considered equivalent to ELT with $K = 1$.

An efficient MLT implementation was proposed by Duhamel *et al.* in [2], and synthesis filter bank algorithm for this MLT implementation was completed by Šević and Popović [3]. If butterflies in MLT are merged, as proposed by Duhamel *et al.* [2], this part of MLT algorithm requires 2 multiplications and 3 additions per sample. However, this kind of merging is not possible for ELT with $K = 1$, where computational savings are achieved by scaling and inverse scaling of butterfly coefficients, so this part of ELT algorithm requires 3 multiplications and 2 additions per sample (if rotations are realized as 3/3, this part of ELT algorithm requires 2.5 multiplications and 2.5 additions per sample).

4. 2-D EXTENSION OF ELT

The basic structure for the 2-D separable ELT analysis filter bank, as proposed by Malvar [1], and based on the use of FFT for DCT-IV realization, is shown in Fig. 2. In the simplest form of implementation, whole rows or columns are fetched from image matrix, and, after processing with 1-D ELT's, returned to matrix. Inverse scaling is applied to input rotations. Although row/column calculations are easy to implement using **FOR** loops, they are not easy to correctly represent in flowgraph.

It is possible to reorder some of row and column

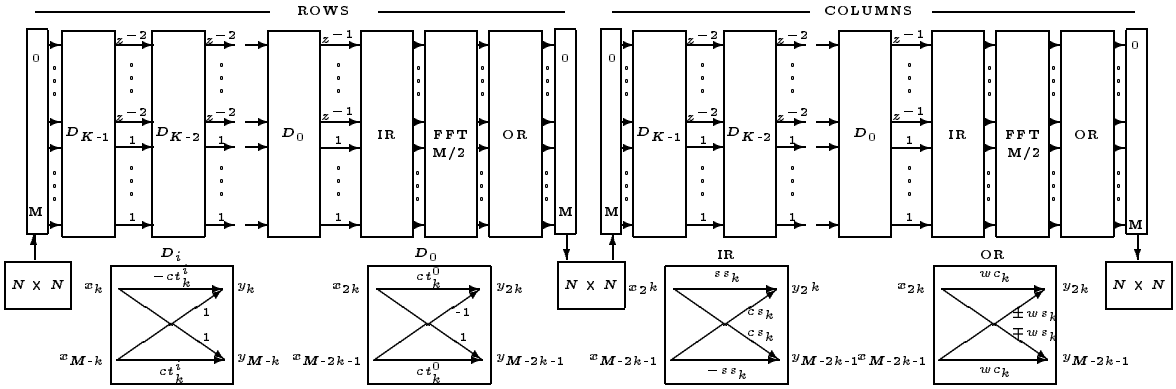

Figure 2: The basic structure for the 2-D separable ELT analysis filter bank. $N \times N$ denotes image matrix, IR and OR are input and output rotations, respectively.

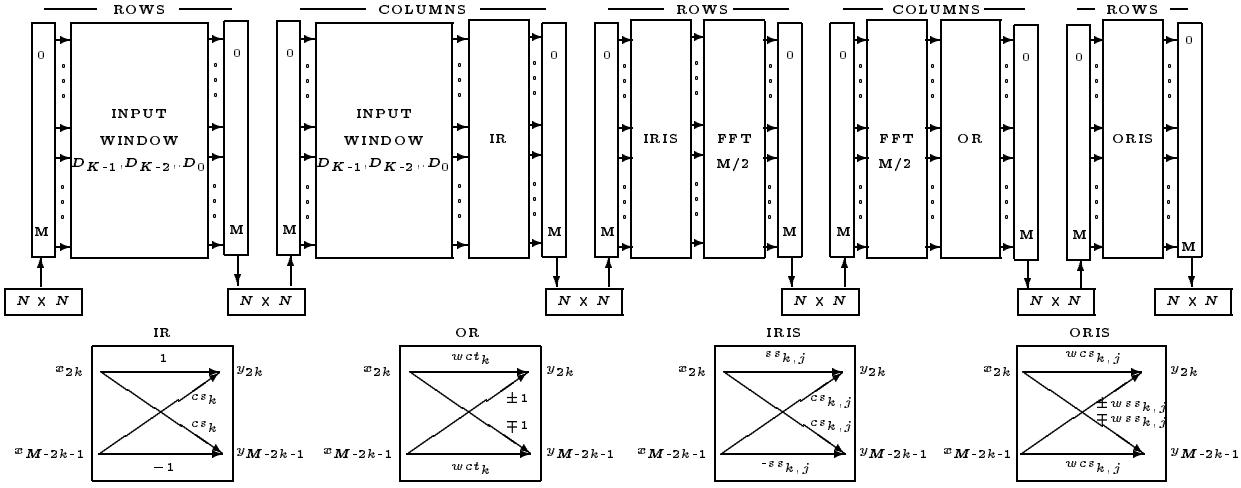

Figure 3: The structure from Fig. 2 with reordered rows/columns computation. Cascade of butterflies and delays is substituted by INPUT WINDOW block. D_i are D_0 are the same as in Fig. 2. IR and OR - scaled input and output rotations. IRIS and ORIS - inverse scaled input and output rotations.

computation, which are independent, without affecting the filter bank output. Reordered flowgraph, shown in Fig. 3. resembles the "true" 2-D implementation. 2-D input window is computed first (row/column), after that 2-D input rotations (column/row, to save on number of row/column fetching), after that 2-D FFT (row/column), and finally, 2-D output rotations (column/row). To save on number of accesses to matrix, it is possible to reorder input window and input rotations row/column computations, as shown in flowgraph in Fig. 4. Implementations shown in Fig 3. and 4. are equivalent.

Reordering, shown in Fig. 3. and 4. makes it possible to achieve computational savings, based on scaling of all butterflies (left or right from FFT computations), and inverse scaling on the last one in the cascade, as shown in Figs. 2. and 3. However, inverse scaling in this last butterfly should compensate for scaling performed both in row and column computations. Inverse scaling in the same dimension is well explained for the 1-D case [1]. Inverse scaling for scaling performed in another dimension is easy to accomplish by using the following rule: if k'th element in row computation is scaled by factor $S[k]$, then subsequent computations in k'th column are to be inverse scaled by the factor $S[k]$. Because of that, inverse scaled butterfly coefficients in Fig. 3. have both row and column indices, k and j. After scaling and inverse scaling, row and column calculations before and after FFT computation are not independent any more, so it is not possible to put scaled operators back in order of Fig. 2. It should be stressed that the new implementation shown in Fig. 3. and 4. has the same frequency response as implementation in [1].

The numbers of operations per sample for the new implementation and for the implementation based on usual rows/columns computations, using 1-D ELT [1] are given in Table I (rotations are counted as 3/3). Number of operations per sample for 2-D ELT proposed in [1] are determined from Table 5.1 [1] by multiplying

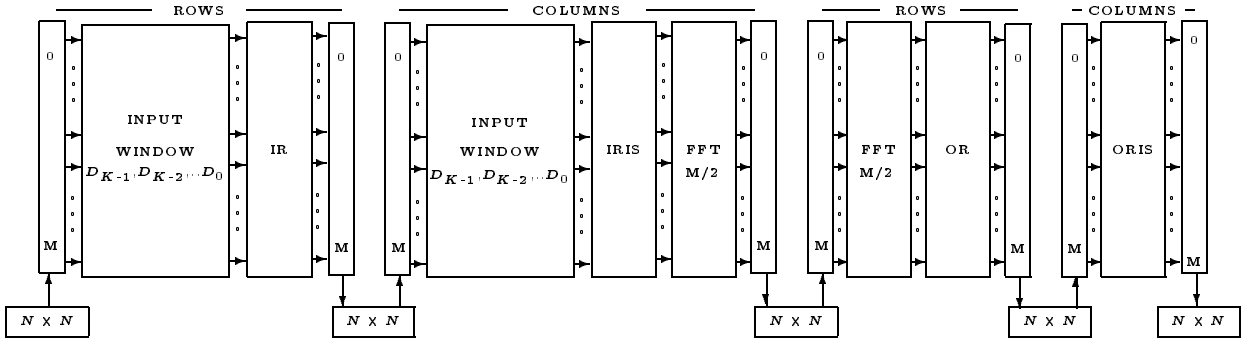

Figure 4: The structure from Fig. 3. with reordered input window and input rotations rows/columns computation. Butterflies are same as in Fig. 3.

Table 1. Computational complexity of the 2-D ELT [1] and the new implementation of the 2-D ELT. M denotes the number of bands in one dimension, K is the overlapping factor, and LT denotes the size of look-up table.

		$K=1$			$K=2$			$K=3$			$K=4$		
	M	Mul/s	Add/s	LT	Mul/s	Add/s	LT	Mul/s	Add/s	LT	Mul/s	Add/s	LT
[1]	2	5.0000	5.0000	4	7.0000	7.0000	5	9.0000	9.0000	6	11.0000	11.0000	7
[1]	4	7.0000	9.0000	8	9.0000	11.0000	10	11.0000	13.0000	12	13.0000	15.0000	14
[1]	8	8.0000	12.0000	18	10.0000	14.0000	22	12.0000	16.0000	26	14.0000	18.0000	30
[1]	16	9.0000	15.0000	44	11.0000	17.0000	52	13.0000	19.0000	60	15.0000	21.0000	68
[1]	32	10.0000	18.0000	90	12.0000	20.0000	106	14.0000	22.0000	122	16.0000	24.0000	138
new	2	4.5000	4.5000	4	6.5000	6.5000	5	8.5000	8.5000	6	10.5000	10.5000	7
new	4	5.5000	7.5000	10	7.5000	9.5000	12	9.5000	11.5000	14	11.5000	13.5000	16
new	8	6.2500	10.2500	40	8.2500	12.2500	44	10.2500	14.2500	48	12.2500	16.2500	52
new	16	7.1250	13.1250	163	9.1250	15.1250	171	11.1250	17.1250	179	13.1250	19.1250	187
new	32	8.0625	16.0625	631	10.0625	18.0625	647	12.0625	20.0625	663	14.0625	22.0625	679

items by 2 (to account for row/column calculations) and by dividing by M, number of input/output elements. LT denotes the size of look-up table necessary for realization of fast algorithm. For rotations counted as 4/2 the size of look-up table would be about 30% less. Savings (Mul + Add) per sample are dependent on number of bands M and vary between (0.5 + 0.5) for $M = 2$ and (1.9375 + 1.9375) for $M = 32$.

The increase of the size of the look-up table for butterfly coefficients is the price to be paid for implementation of the new algorithm. However, memory requirements for look-up table are still negligible compared to memory requirements for whole image processing.

In a simplest form of implementation, whole rows and columns are fetched from and retrieved to image matrix. However, looking at Figs. 2 and 4, it is seen that the new algorithm requires more frequent accesses to image matrix. If realized in this way, the new implementation, with reduced number of real operations, and increased number of real data transfers, wouldn't be much faster compared to the implementation from Fig. 2. on most computers. To take advantage of reduced computational complexity of the new implementation, matrix elements should be directly accessed and processed, using pointers.

Because of orthogonality of the ELT, the synthesis filter bank is the transpose of the analysis filter bank, and has the same number of operations as the analysis filter bank.

5. CONCLUSION

In this paper, a new implementation of the 2-D Extended Lapped Transform is proposed. Compared to the separable solution [1], the new realization of 2-D ELT has reduced arithmetic complexity. Computational savings are achieved because scaling and inverse scaling of butterfly matrices, suggested by Malvar for 1-D case, are, after some modifications of the basic algorithm, extended to 2-D case. The new implementation has the same frequency response as Malvar's.

6. REFERENCES

[1] H.S. Malvar, *Signal Processing with Lapped Transforms*, Artech House, Norwood, MA, 1992.

[2] P. Duhamel, Y. Mahieux, and J.P. Petit, "A fast algorithm for the implementation of filter banks based on "Time domain aliasing cancellation"," in *Proc. ICASSP-91*, Toronto, May 1991, pp. 2209-2212.

[3] D. Šević and M. Popović, "A new efficient implementation of the oddly stacked Princen-Bradley filter bank," *IEEE Signal Processing Letters*, vol. 1, pp. 166-168, Nov. 1994.