Applying Matrix Decomposition Techniques to Edge Detection Operators

Leonel Sousa[1], José Salvado[2]

[1] *Instituto Superior Técnico e Instituto de Engenharia de Sistemas e Computadores*
R. Alves Redol 9, 1000 Lisboa , PORTUGAL
email: las@molly.inesc.pt

[2] *Escola Superior de Tecnologia, Instituto Politécnico de Castelo Branco*
Av. do Empresário, 6000 Castelo Branco, PORTUGAL

Abstract: *In this paper decomposition techniques are applied to derivative operators, used for image edge detection. It is shown that the application of decomposition techniques to common edge detectors can result in substantial savings in computing time. For a 25x25 Laplacian of Gaussian, mask, an improvement of six times less arithmetic operations is achieved when decomposition techniques are applied.We also show that these techniques are advantageous for hardware realization of the filters. The memory required to a 2-D* (nxn)*-th order FIR filter direct realization with distributed arithmetic is* $O(2^{(n+1)^2})$ *while the worst case for the decomposed filter is* $O(n \times 2^n)$.

Keywords*: 2-D filters, edge detection, matrix decomposition, computing efficiency.*

1. INTRODUCTION

2-D digital linear filtering is often used in image and video processing, for noise removal, edge enhancement and edge detection purposes. The most common filters are finite impulse response (**FIR**) filters, whose transfer function may be represented as:

$$H(z_1, z_2) = \mathbf{Z}_1^T \mathbf{F}\, \mathbf{Z}_2 \ , \qquad (1)$$

where

$$\mathbf{Z}_1 = \begin{bmatrix} 1 & z_1^{-1} & \dots & z_1^{-(m-1)} \end{bmatrix}^T$$

$$\mathbf{Z}_2 = \begin{bmatrix} 1 & z_2^{-1} & \dots & z_2^{-(n-1)} \end{bmatrix}^T .$$

The matrix $\mathbf{F}_{(mxn)}$ is usually squared and composed by a large number of coefficients. The application of these operators to large arrays of pixels is computationally intensive. It requires huge amount of time when implemented in general purpose computers and great quantity of hardware for dedicated realizations.

The problem of reducing all the computacional requirements associated with 2-D FIR filters realization can be attacked in several ways. A possible method involves the approximation of the given response function by an 2-D recursive filter [1]. The approach we are interested on is based on the expansion of the given coefficient matrix into a finite and converging sum of separable matrices; this means that each matrix in the expansion can be expressed as the product of a column vector by a row vector. Several methods have been proposed to the decomposition of the filters transfer function [2,3].

In this paper, we study the decomposition techniques application to derivative operators, used for image edge detection. The classic gradient or Laplacian operators usually have low performance when applied to noisy images. To obtain performance improvement, some type of averaging is made before the application of derivative operators. The resulting coefficient matrices are larger, which allows performance improvement at the expense of computational complexity. The computing efficiency gain achieved with the application of decomposition techniques to those operators are analysed in this paper.

In the next section, a short description of the used matrix decomposition methods is presented. The subject of their advantages is also discussed. In section 3, experimental results obtained by applying the decomposition methods to typical edge detection operators are presented and analysed. In section 4, we discuss the implications of the decomposition in hardware filters' realizations, namely for systolic structures and distributed arithmetic computation. Finally, conclusions are drawn in in section 5.

2. 2-D FIR FILTERS DECOMPOSITION

By a 2-D filter coefficient matrix decomposition, the matrix **F** can be represented as:

$$\mathbf{F} = \sum_{i=1}^{p} \mathbf{F}_i \ , \qquad (2)$$

where p is the rank of the original matrix and $\mathbf{F}_i$ represents separable matrices. Since each matrix $\mathbf{F}_i$ is separable, it can be split into the product of a column vector by a row vector. Therefore, $H(z_1,z_2)$ can be expressed as a sum of products of 1-D polynomials, each one of which is function of one variable only:

$$H(z_1,z_2) = \sum_{i=1}^{p} \alpha_i(z_1) \times \beta_i(z_2) \,. \qquad (3)$$

Two methods for decomposing the matrix of weighting coefficients ($\mathbf{F}_i$) are now briefly presented: *i*) Singular Value (**SV**) decomposition, and *ii*) Lower Upper triangular (**LU**) decomposition. Thses two methods are compared regarding both the error introduced and the computional efficiency gain achieved. Without loss of generality, in order to facilitate the discussion, a square coefficient matrix $\mathbf{F}_{(nxn)}$ (rank $p \le n$) is considered.

2.1. Singular-Value (SV) Decomposition

With SV decomposition, the matrix **F** can be represented by [2]:

$$\begin{aligned} &\mathbf{F} = \mathbf{c}_1\mathbf{r}_1 + \mathbf{c}_2\mathbf{r}_2 + \cdots + \mathbf{c}_p\mathbf{r}_p \\ &\mathbf{c_i} = \sqrt{\lambda_i}\,\mathbf{k}_i \end{aligned} \quad , \qquad (4)$$

where:

- $\lambda_1 > \lambda_2 > \ldots > \lambda_p$ - are the positive eigenvalues of the symmetric matrix $\mathbf{F}^\mathbf{T}\mathbf{F}$;
- $\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_p$ - are the normalized column eigenvectors associated with the corresponding eigenvalues of the matrix $\mathbf{F}\mathbf{F}^\mathbf{T}$;
- $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_p$ are the normalized row eigenvectors associated with the corresponding eigenvalues of the matrix $\mathbf{F}^\mathbf{T}\mathbf{F}$.

Eq. 1 can then be written as:

$$H(z_1,z_2) = \sum_{i=1}^{p} (\mathbf{Z}_1^T\mathbf{c}_i).(\mathbf{r}_i\,\mathbf{Z}_2) = \sum_{i=1}^{p} \alpha_i(z_1)\beta_i(z_2) \,.$$

The rank of matrix **F** determines the number of the expansion terms. Thus far, all the p terms are included in the expansion, which leads to an exact representation of **F**. However, the p terms of Eq. 4 can not contribute equally to the entire expansion. If only the first d dominant terms are included, Eq. 4 assumes the form:

$$\mathbf{F} = \sum_{i=1}^{d} \mathbf{c}_i\mathbf{r}_i + \mathbf{E}_d \quad , \qquad (5)$$

where the (nxn) array $\mathbf{E}_d$ represents the truncation error matrix. For a mean-square error (MSE) approximation, we can use the energy in the truncation error matrix (ε_d) as a figure of merit:

$$\varepsilon_d = \frac{J_d}{J_0} \quad , \qquad (6)$$

where J_d is the sum of the squares of the truncation error matrix elements and J_0 is the sum of the squares of matrix **F** elements. Treitel and Shanks [2] have proved that:

$$\varepsilon_d = \frac{\sum_{i=d+1}^{p} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \quad , \qquad (7)$$

where the bounds of ε_d are 0 for $d=p$ and 1 for $d=0$.

In the SV decomposition, the truncation error is easily computed and controlled. However, as we will see in sub-section 2.3, the computing efficiency gain achieved with this method is only significant when the number of dominant terms of the expansion is small.

2.2. Lower Upper (LU) Triangular Decomposition

To apply the LU decomposition method to matrix **F**, one has to guarantee that its first p successive principle minors are nonzero. This is achieved by making the suitable permutations of rows and columns of matrix **F** (**F**'):

$$H(z_1,z_2) = \hat{\mathbf{Z}}_1^T\mathbf{F}'\hat{\mathbf{Z}}_2 \,. \qquad (8)$$

Applying the LU decomposition method to the matrix **F**' we obtain:

$$\mathbf{F}' = \sum_{i=1}^{p} d_i\mathbf{L}_i\mathbf{U}_i^T \quad , \qquad (9)$$

where

$$\begin{aligned} \mathbf{L_i} &= \begin{bmatrix} 0 & \cdots & 0 & 1 & l_{i+1,i} & \cdots & l_{n,i} \end{bmatrix}_{n\times 1}^T \\ \mathbf{U_i} &= \begin{bmatrix} 0 & \cdots & 0 & 1 & u_{i,i+1} & \cdots & l_{i,n} \end{bmatrix}_{n\times 1}^T \end{aligned} \,. \qquad (10)$$

Therefore, the transfer function is:

$$H(z_1,z_2) = \sum_{i=1}^{p} d_i(\hat{\mathbf{Z}}_1^T\mathbf{L_i}).(\mathbf{U_i}\,\hat{\mathbf{Z}}_2) = \sum_{i=1}^{p} \alpha_i(z_1)\beta_i(z_2) \,.$$

As in the SV decomposition, in the LU decomposition is also possibly to retain only part of the expansion terms for a non-exact representation of matrix **F**.

In the following sub-section we discuss the advantages of these two decomposition methods and the possibility of their conjunction .

2.3. Computing Efficiency Gain

The computing efficiency gain achieved with the decomposition techniques can be evaluated by determining the number of "multiply and add" operations (MADs) required for filters realization. n^2 MADS are required for the direct realization of a $(n-1)\times(n-1)$-th order filter (Eq. 1). $2\times n\times d$ MADs are required to realize the filter if SV decomposition is applied and d terms are retained. In the same conditions, the number of MADs for the LU decomposition is:

$$2 \times \sum_{i=1}^{d} (n - i + 1) = 2 \times n \times d + d - d^2$$

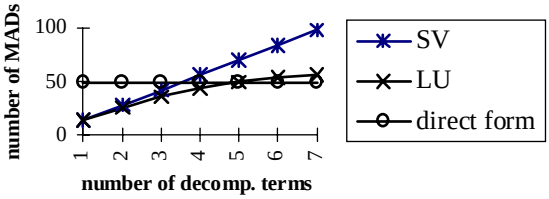


Figure 1 - Number of MADs versus the number of expansion terms for a 7x7 matrix.

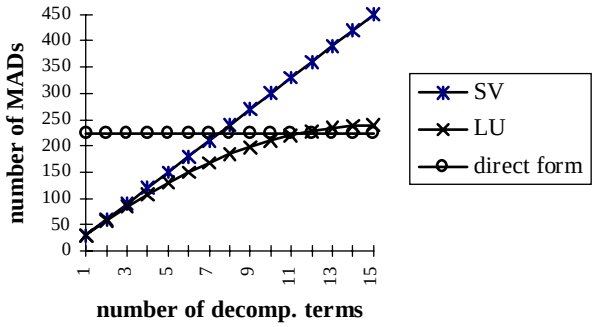


Figure 2. Number of MADs versus the number of expansion terms for a 15x15 matrix.

Figures 1 and 2 graphically present the number of MADs versus the number of the expansion terms for, respectively, 7x7 and 15x15 matrices. The computing efficiency gain is quite similar for both decomposition methods if a relative small number of terms are considered (less than about 30% of the total number). However, for the SV decomposition the number of MADs grows linearly with the number of terms, while this growing rate is much smaller for the LU decomposition. Moreover, the LU decomposition only requires a greater number of MADs than the direct form when almost the total number of terms is included in the expansion. On the other hand, experimental studies have shown that the SV decomposition requires a less number of expansion terms to achieve a given MSE [3]. This is also confirmed by the results presented in the next section.

We can take advantage of the SV and LU properties in the following way:

- by applying the SV decomposition on the matrix **F**, and by retaining only the *d* expansion terms required for a given MSE;
- by computing a new matrix **G** based on only the *d* expansion terms

$$\mathbf{G} = \sum_{i=1}^{d} \mathbf{c}_i \mathbf{r}_i \quad ;$$

- by performing the LU decomposition on **G**.

The final decomposition requires a minimum number of stages and a minimum number of MADs for a given MSE.

3. Applying SV and LU Decomposition to Edge Detectors

In a simplified approach, an edge can be detected as a sharp discontinuity in the grey-level profile. However, the task of detecting edges in real images is more complex, due to the presence of noise and image resolution. Well-known edge detectors based on first and second order derivative operators are *gradient* and *Laplacian* edge detectors [5]. Pixels with high gradient magnitude or with *Laplacian* zero-crossings are labeled as edges. The weighting coefficient matrices corresponding to these linear operators have small size (typically 3x3 or 5x5). In general, these detectors have low performance poorly on noisy images. To improve their performance, some of the noise is usually removed by previous filtering.

Generalized *Laplacian* operators, with large size coefficient arrays, can be used for noise reduction and edge detection. Marr and Hildreth suggest a family of generalized *Laplacian* edge operators, which uses *Gaussian* low-pass filters to remove noise [5]. The matrix corresponding to the *Laplacian* of the *Gaussian* (LoG) must have at least $(6\sqrt{2}\sigma \times 6\sqrt{2}\sigma)$ elements, where σ is the space constant of the *Gaussian* — for example, a 17x17 matrix should be used for σ=2.

Software tools for the decomposition of 2-D filters were developed by the authors [6], based on the MATLAB[1] mathematics routines. These tools were used to decompose the edge detectors just referred. The results are presented in the next sub-sections.

3.1. Gradient Edge Detector

The application of SV decomposition is illustrated, by using the basic gradient edge detector. The two Sobel masks to compute the *gradient* are:

$$\mathbf{F}_{\Delta x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \mathbf{F}_{\Delta y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

The rows and columns of the matrices are linearly dependent, so they have a unitary rank.

By performing the SV decomposition:

[1] MATLAB is a registered trademark of MathWorks Inc..

$$\mathbf{F}_{\Delta x}^{T}\mathbf{F}_{\Delta x} = \begin{bmatrix} 6 & 0 & -6 \\ 0 & 0 & 0 \\ -6 & 0 & 6 \end{bmatrix} \quad \mathbf{F}_{\Delta y}^{T}\mathbf{F}_{\Delta y} = \begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

$$\lambda_{\Delta x} = \lambda_{\Delta y} = 12$$

$$\mathbf{F}_{\Delta x} = \begin{bmatrix} +1..4142 \\ +2.8284 \\ +1.4142 \end{bmatrix} \times \begin{bmatrix} -0.7071 & 0 & +0.7071 \end{bmatrix}$$

$$\mathbf{F}_{\Delta y} = \begin{bmatrix} +2.4494 \\ 0 \\ -2.4494 \end{bmatrix} \times \begin{bmatrix} 0.4082 & 0.8165 & 0.4082 \end{bmatrix}.$$

By performing the LU decomposition:

$$\mathbf{F}_{\Delta x} = -1 \times \begin{bmatrix} +1. \\ +2 \\ +1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{F}_{\Delta y} = 1 \times \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} \times \begin{bmatrix} +1 & +2 & +1 \end{bmatrix}.$$

For this simple but illustrative example, the small filter arrays are separable, but decomposition leads to poor computing efficiency gain.

3.2. Zero-Crossing Edge Detectors

Laplacian and generalized *Laplacian* are isotropic edge detectors. The weighting coefficients arrays associated to these operators exhibit a low rank. The weighting coefficients matrix upper left quadrant for a generalized *Laplacian* filter may be:

$$\begin{vmatrix} -34 & -25 & -18 & -12 & -8 & -7 \\ -25 & -15 & -7 & -1 & 3 & 4 \\ -18 & -7 & 2 & 8 & 12 & 14 \\ -12 & -1 & 8 & 15 & 20 & 21 \\ -8 & 3 & 12 & 20 & 24 & 26 \\ -7 & 4 & 14 & 21 & 26 & 27 \end{vmatrix}$$

which correspond to the following eigenvalues:

$$\lambda = \begin{bmatrix} 16058 & 13512 & 4 & 2 \end{bmatrix}.$$

The first two expansion terms are clearly dominant. By applying the SV and LU decomposition we obtain the results presented in Table 1.

SV			LU	
d	ε_d	MADs	ε_d	MADs
1	$4.57\text{x}10^{-2}$	22	$7.35\text{x}10^{-1}$	22
2	$2.14\text{x}10^{-4}$	44	$2.57\text{x}10^{-3}$	42
3	$7.90\text{x}10^{-5}$	66	$1.67\text{x}10^{-3}$	60

Table 1 - Error and number of MADs obtained for the generalized Laplacian filter decomposition.

A good approximation of the filter is obtained by retaining the first two expansion terms. In these conditions, only 60 MADs are required to realize the filter, against the 121 required when we directly use the transfer function. As referred in the previous section, the SV and LU decompositions can be applied in conjunction to achieve a better computing efficiency gain with a minimum error.

The coefficient matrices associated to the *Laplacian* of *Gaussian* (LoG) zero-crossing edge detector is given by:

$$\nabla^2 G(x,y) = k\left(2 - \frac{x^2 + y^2}{\sigma^2}\right) \times e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}, \quad (11)$$

$$k = \frac{1}{2\Pi\sigma^4}$$

where σ is the space constant of the *Gaussian*. As Huertas and others observed [4], it is possible to decompose the LoG into a sum of two separable filters:

$$\nabla^2 G(x,y) = h_1(x)h_2(y) + h_2(x)h_1(y)$$

$$h_1(\xi) = \sqrt{k}\left(1 - \frac{\xi^2}{\sigma^2}\right) \times e^{-\frac{\xi^2}{2\sigma^2}} \quad . \quad (12)$$

$$h_2(\xi) = \sqrt{k} \times e^{-\frac{\xi^2}{2\sigma^2}}$$

If a floating point numbering system is used, only two expansion terms are required for the exact representation of $\nabla^2(x,y)$. However, the coefficients of the matrix are usually quantized to integers [5]. When the SV and LU decomposition methods are applied to $(6\sqrt{2}\sigma \times 6\sqrt{2}\sigma)$ LoG 2-D matrices and the coefficients are quantized to 16 bit integers, the results in Table 2 are obtained.

		SV		LU	
σ	d	ε_d	MADs	ε_d	MADs
0.5	1	$3.08\text{x}10^{-2}$	10	$2.19\text{x}10^{-1}$	10
	2	$1.21\text{x}10^{-11}$	20	$5.94\text{x}10^{-6}$	18
	3	0	30	0	24
1	1	$6.78\text{x}10^{-2}$	18	$4.62\text{x}10^{-1}$	18
	2	$1.87\text{x}10^{-9}$	36	$3.34\text{x}10^{-3}$	34
	3	$1.37\text{x}10^{-10}$	54	$9.90\text{x}10^{-7}$	48
1.5	1	$6.70\text{x}10^{-2}$	26	$4.57\text{x}10^{-1}$	26
	2	$4.70\text{x}10^{-9}$	52	$8.40\text{x}10^{-2}$	50
	3	$1.60\text{x}10^{-9}$	78	$8.35\text{x}10^{-5}$	72

2	1	$6.71\text{x}10^{-2}$	34	$4.60\text{x}10^{-1}$	34
	2	$4.26\text{x}10^{-9}$	68	$1.94\text{x}10^{-2}$	66
	3	$1.33\text{x}10^{-9}$	102	$1.63\text{x}10^{-3}$	96
2.5	1	$6.71\text{x}10^{-2}$	42	$4.52\text{x}10^{-1}$	42
	2	$4.47\text{x}10^{-9}$	84	$7.99\text{x}10^{-2}$	82
	3	$2.26\text{x}10^{-9}$	126	$6.47\text{x}10^{-3}$	120
3	1	$6.71\text{x}10^{-2}$	50	$4.57\text{x}10^{-1}$	50
	2	$3.67\text{x}10^{-9}$	100	$1.10\text{x}10^{-1}$	98
	3	$1.97\text{x}10^{-9}$	150	$1.23\text{x}10^{-2}$	144

Table 2 - Error and number of MADs for the LoG filter decompostion.

For the same number of expansion terms, the MSE is considerable lower for the SV decomposition than for the LU decomposition. For the SV decomposition the MSE is negligible when only two expansion terms are retained. Although the original LoG filter can be decomposed into a sum of two separable filters, results in Table 2 are concerned to an integer quantized filter approach. Using two expansion terms, the computing efficiency gain, compared to the direct form, varies between 1.25 to 6.25, for arrays size between 5x5 and 25x25.

The MSE for the LU decomposition is significant when two expansion terms are retained. As it was referred in the previous section, the LU decomposition can be useful when applied in conjunction with the SV decomposition, in order to increase the computing efficiency gain.

4. HARDWARE FILTER REALIZATION

The block diagram of Fig. 3 shows the realization of the FIR filters described by Eq. 3.

Figure 3.- Decomposition realization for 2-D filters.

The decomposition realization exhibits parallelism, because the *d* terms can be computed in parallel. Moreover, each of the *d* stages corresponds to two 1-D filters. Array processors [7] and distributed arithmetic computation [9] are two approaches used for the hardware implementation of the filters. As we will see in the next sub-sections, filter decomposition can be advantageous for both approaches.

4.1. Array Processors

A systolic array is a computing network in which the data are rhythmically computed and passed through the network. Several systolic array processors have been proposed for 1-D and 2-D filters [7, 8]. The structure of such arrays is directly derived from the filter signal flow graph. Retiming procedures are applied to the graphs in order to achieve completely pipelined computation. Array processor elements basically perform MAD operations. Processing elements are interconnected trough delay elements corresponding to the sampling period of a pixel or of a row of pixels.

A systolic processor directly obtained from the transfer function of a (*nxn*)-th order 2-D FIR filter implies $(n+1)^2$ processor elements. The number of processor elements required to realize filters with decomposition is equal to the number of MADs. Therefore, the cost of systolic array processors depends directly on the number of expansion terms.

4.2. Distributed Arithmetic Computation

To implement a filter equation with distributed arithmetic it is assumed that: *i*) signals are bounded by ±1; *ii*) signals are coded in two's complement with *B* bits of accuracy. The general difference equation for a *n*-th order 1-D FIR filter can be written in the form:

$$y_i = \sum_{s=1}^{B-1}\left(\sum_{k=0}^{n} 2^{-s} a_k x_{i-k}^{s}\right) - \sum_{k=0}^{n} a_k x_{i-k}^{0} \quad , \qquad (13)$$

where x_{i-k}^{s} are binary variables. Through the definition of a proper function, $\varphi(x_i, \cdots, x_{i-n})$, Eq. 13 can be re-written as:

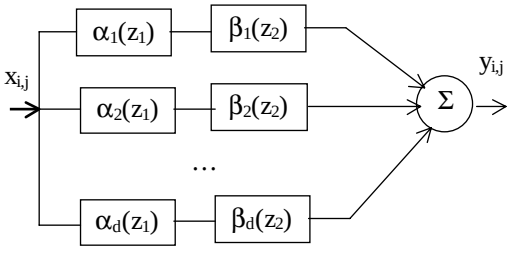
$$y_i = \sum_{s=1}^{B-1} \varphi^{s}(.) 2^{-s} - \varphi^{0}(.) \quad , \qquad (14)$$

and the filter can be realized through look-up-tables, for realizing the φ(.) function, mask bit shifters and adders. The total memory requirement to realize a 1-D *n*-th order filter is then:

$$B \times t \times 2^{n+1} \quad ,$$

where *t* is the coefficient precision (in bits).

Distributed arithmetic can also be applied to 2-D FIR



filters, by writing the corresponding difference equation in the form:

$$y_{i,j} = \sum_{s=1}^{B-1}\left(\sum_{k=0}^{n}\sum_{l=0}^{n} 2^{-s} a_{k,l} x_{i-k,j-l}^{s}\right) - \sum_{k=0}^{n}\sum_{l=0}^{n} a_{k,l} x_{i-k,j-l}^{0} \qquad (15)$$

Through the definition of a proper function:

$$\phi(x_{i,j}, \cdots, x_{i,j-n}, x_{i-1,j}, \cdots, x_{i-n,j-n}) \quad ,$$

Eq. 15 can be re-written as:

$$y_{i,j} = \sum_{s=1}^{B-1} \phi^s(.)2^{-s} - \phi^0(.) \ . \qquad (16)$$

The total memory requirement to realize a (*nxn*)-th order 2-D FIR filter is then:

$$B \times t \times 2^{(n+1)^2} \ .$$

Distributed arithmetic computation is easily applied to the filter decomposition realization depicted in Fig. 3. For each of the *d* stages, we have two 1-D filters. The total memory requirement to realize a 2-D (*nxn*)-th order filter with decomposition in *d* stages is:

$$d \times B \times t \times 2^{n+2} \ .$$

Therefore, an high reduction on the memory requirement is achieved when decomposition is applied. For the exact representation of the filters (all expansion terms are retained), the memory requirement order is:

$$O(n \times 2^n) \ ,$$

while for the direct form realization it rises to:

$$O(2^{(n+1)^2}) \ .$$

5. CONCLUSIONS

In this paper the application of decomposition techniques to derivative edge detectors is discussed. It is shown that significant computing efficiency gain is achieved by applying these techniques to zero-crossing edge detectors. For a 25x25 LoG mask (σ=3), an improvement of six times less MADs is achieved when compared with the direct form realization. It is also shown that the application of decomposition techniques allow to significantly reduce the hardware filter realization cost. The memory required to a 2-D (*nxn*)-th order FIR filter direct realization with distributed arithmetic is $O(2^{(n+1)^2})$ while the worst case for the decomposed filter is $O(n \times 2^n)$.

Research work is being done in order to quantify the decomposition errors impact in edge detection performance in real images.

6. REFERENCES

[1] J. S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall, 1990.

[2] S. Treitel, J. Shanks, "The design of Multistage Separable Filters", *IEEE Trans. On Geocs. Elect.*, vol.9, nº1, Jan. 1971.

[3] C. L. Nikias, *et al.*, "The LU Decomposition Theorem and its Implications to the Realization of Two-Dimensional Digital filters", *IEEE Trans. On Ac., Speech, and Sig. Proc.*, vol.33, nº3, Jun. 1985.

[4] A. Huertas, G. Medioni., "Dectection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks", *IEEE Trans. Pattern Analysis and Mach. Intell,*, vol.8, nº5, Jun. 1986.

[5] R. Haralick, L. Shapiro, *Computer and Robot Vision*, vol I, Addison-Wesley, 1992.

[6] J. Salvado, L. Sousa, "FIDEC- Software Tools for Decomposition of 2-D Filters, *INESC internal report*, 1997 (available only in portuguese language).

[7] S. Y. Kung, *Array Processors*, Information an System Sciences Series, Prentice-Hall, 1988.

[8] N. Petkov, *Systolic Parallel Processing*, Advance in Parallel Computing Series, Elsevier Science Publishers, 1993.

[9] A. Peled, B. Liu, "A New Hardware Realization of Digital Filters", *IEEE Trans. On Ac., Speech, and Sig. Proc.*, vol.22, nº6, Dec. 1974.

[10] J. Jaggernauth, *et al..*, "Real-Time Image Processing by Distributed Arithmetic Implementation of Two-Dimensional Digital Filters", *IEEE Trans. On Ac., Speech, and Sig. Proc.*, vol.33, nº6, Dec. 1985.