Parallelization of the Resampling Image Filter

Mourad Mahboub[1] and Djamal Lachachi[2]
[1]Sciences Faculty, Abou Bakr Belkaid University, Algeria
[2] Engineering Sciences Faculty, Abou Bakr Belkaid University, Algeria

***Abstract**: When resampling a digital image by uniform cubic B-splines, an output pixel is computed from a filter applied to 16 neighboring pixels of the original image or more precisely of some auxiliary matrix C. Matrix C is computed with a computational time proportional to the number of pixels of the image to be resampled; that time is substantially smaller than the computational time of the filtering part. In this paper, an adapted Cholesky factorization is presented; it allows to calculate the matrix C, the resampling filter of the 16 neighboring pixels and a parallel computation of the filter. This parallel approach reduces the global computational time partly from a better memory management.*

Keywords: Image resampling, uniform cubic B-spline, interpolation, filter of 16 pixels, speedup, efficiency.

Received March 3, 2006; accepted June 2, 2006

1. Introduction

Nonlinear geometric transformations of the image grid pixel positions are often used in numerical image processing. Generally, the pixel position transforms do not coincide with the input image grid. One resampling algorithm allows to obtain the value of these new pixels. The perfect interpolation (the so-called Shannon interpolation), which allows to reconstruct the exact image, is not possible to perform [1, 9]. To calculate the output pixels, the uniform cubic B-spline resampling requires 16 neighboring pixels of an intermediate image based on a matrix C with the same dimensions (nxn) as for the input image. We proved [7] that the complexity of the algorithm computing the matrix C is O (n^2). Computational times of the matrix C obtained from the sequential implementation is smaller than the time dedicated to the 16 neighboring pixels filter.

We propose, in this paper, an adapted Cholesky Factorization to calculate the matrix C, the resampling filter of the 16 neighboring pixels, and a parallel computation of the filter. This parallel approach reduces the computational time of the filter in such a way that it becomes smaller than the computation of the matrix C [6].

2. The Uniform Cubic B-Spline Resampling

The reconstructed image g' with B-spline functions is expressed by the following transformation:

$$g'(x,y)=\sum_{i=1}^{n}\sum_{j=1}^{n'}c_{ij}\,s(x-i\Delta x)s(y-j\Delta y) \qquad (1)$$

where c_{ij} are the weighting coefficients, and s(x) is the basis functions given by the following equations:

$$s(x)=x_+^3-4(x-\Delta x)_+^3+6(x-2\Delta x)_+^3-4(x-3\Delta x)_+^3 \qquad (2)$$

with

$$x_+{}^3=\begin{cases}x^3 & \text{for } x>0,\\ 0 & \text{for } x\le 0\end{cases} \qquad (3)$$

The numerical resampled image G' is of dimension MxM', where M=(m+1)n, and M'=(m+1)n' for m output pixels calculated in each direction. This matrix is obtained from the following expression:

$$G' = B_1\, C\, B_2^T \qquad (4)$$

The matrices B_1 (M, n) and $B_2{}^T$ (n', M') resample respectively rows and columns of the matrix C. These matrices are block Toeplitz and are calculated by the basis functions at the output pixels. The nxn' matrix C is formed by the weighting coefficients c_{ij} and is characterized by the digital input image G.

The matrix G (n, n') of the digital input image must be consistent with the relation (4); we obtain the following equation:

$$G = A_n\, C\, A_{n'} \qquad (5)$$

The two matrices A_n and $A_{n'}$ are calculated respectively by the spline basis functions in x and y directions to obtain the pixel values of G. Without losing generality, in the following we restrict our study to square images, i. e. n = n' and M = M'.

The matrix A is defined by:

$$A=\begin{bmatrix} 4 & 1 & 0 & & & \dots & 0 & 1\\ 1 & 4 & 1 & 0 & & \dots & & 0\\ 0 & 1 & 4 & 1 & 0 & \dots & & 0\\ \vdots & & & & \vdots & & & \vdots\\ 0 & & & \dots & 1 & 4 & 1 & 0\\ 0 & & & \dots & 0 & 1 & 4 & 1\\ 1 & 0 & & \dots & & 0 & 1 & 4 \end{bmatrix} \qquad (6)$$

and the matrix *B* calculated is equal to:

$$B = \begin{bmatrix} c_k & d_k & 0 & 0 & \dots & 0 & 0 & a_k & b_k \\ \hline b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \\ \hline a_k & b_k & c_k & d_k & \dots & 0 & 0 & 0 & 0 \\ \hline & & & & \vdots & & & & \\ \hline b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \end{bmatrix} \tag{7}$$

with k ∈ [0, m] and where coefficients are defined by:

$$\begin{aligned} a_k &= (\beta_k + 3)^3 - 4(\beta_k + 2)^3 + 6(\beta_k + 1)^3 - 4\beta_k{}^3 \\ b_k &= (\beta_k + 2)^3 - 4(\beta_k + 1)^3 + 6\beta_k{}^3 \\ c_k &= (\beta_k + 1)^3 - 4\beta_k{}^3 \\ d_k &= \beta_k{}^3 \\ \beta_k &= kn/M,\ k \in [0, m] \end{aligned} \tag{8}$$

3. Factorization of Matrix *A*

Let us decompose matrix *A* of order n in the following way:

$$A = \left[\begin{array}{c|c} A_d & a \\ \hline a^T & 4 \end{array}\right] \tag{9}$$

where A_d is the symmetric tridiagonal principal matrix of order (n -1), and *a* is the vector:

$$a = (1,\ 0,\ \dots,\ 0,\ 1)^T \in R^{\,n-1} \tag{10}$$

The Cholesky factorization matrix *L* of the matrix *A* is expressed with the Cholesky factor of L_d of matrix A_d:

$$A_d = L_d\, L_d{}^T \tag{11}$$

We look for an updated triangular factor *L* such that:

$$A = L\, L^T \tag{12}$$

with

$$L = \left[\begin{array}{c|c} L_d & 0 \\ \hline l^T & \lambda \end{array}\right] \tag{13}$$

where L_d is a bidiagonal matrix. We must compute L_d, *l* and λ satisfying:

$$\begin{cases} L_d\, L_d{}^T = A_d \\ L_d\, l = a \\ \lambda^2 + \| l \|^2 = 4 \end{cases}$$

which can be done by the following sequence of steps:

L_d : = Cholesky factor of A_d,

Solve of the triangular system $L_d\, l = a$,

$\lambda = \sqrt{4 - \| l \|^2}$.

The complexity of the factorization is therefore of order O(n).

4. Computation of Matrix C

Equation (5) is equivalent to:

$$G = L\, L^T\, C\, L\, L^T$$

Which can be solved by four successive triangular systems:

$$\begin{gathered} G = L\, Y, \\ Y = L^T X, \\ X = Z\, L^T, \\ Z = C\, L. \end{gathered}$$

Solving each of these systems involves $5n^2$ + O(1) floating point operations. The total number of operations to compute matrix *C* is therefore :

$$\text{Comp (C)} = 20n^2 + \text{O (n)} \tag{14}$$

5. Complexity of the Filter of 16 Neighbor Pixels

The two-sided multiplication of the matrix B with the matrix C, according to the equation (4), can be performed through the resampling uniform cubic B-spline filter $F_{k,\,u}$ of the 16 pixels[5]. The resampling filter is defined by

$$F_{k,u} = \begin{bmatrix} a_k a_u & a_k b_u & a_k c_u & a_k d_u \\ b_k a_u & b_k b_u & b_k c_u & b_k d_u \\ c_k a_u & c_k b_u & c_k c_u & c_k d_u \\ d_k a_u & d_k b_u & d_k c_u & d_k d_u \end{bmatrix} \tag{15}$$

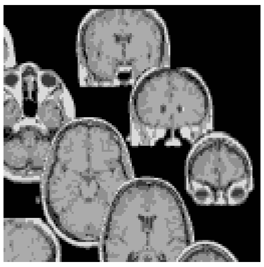
with (k, u) ∈ $[0,\ m]^2$, and where coefficients are defined by equation (8).

This filter allows to calculate m output pixels in each dimension. It is applied over the window $C_{i,\,j}$ (dimension 4x4) of the matrix C.

$$C_{i,j} = \begin{bmatrix} c_{i-1,j-1} & c_{i-1,j} & c_{i-1,j+1} & c_{i-1,j+2} \\ c_{i,j-1} & c_{i,j} & c_{i,j+1} & c_{i,j+2} \\ c_{i+1,j-1} & c_{i+1,j} & c_{i+1,j+1} & c_{i+1,j+2} \\ c_{i+2,j-1} & c_{i+2,j} & c_{i+2,j+1} & c_{i+2,j+2} \end{bmatrix} \tag{16}$$

with $(i, j) \in [2, n-2]^2$.

This operation is known as « the stencil method » or the Kernel method [4]. An example of resampling one real image of 128x128 pixels with 1 output pixel is shown in the figure 1.


(a) Input Image.

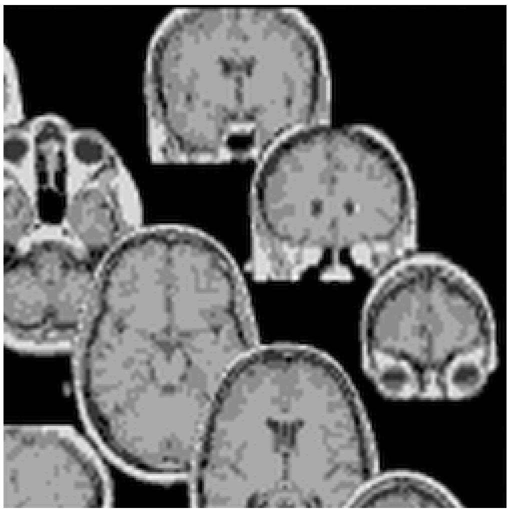

(b) Output Image.

Figure 1. (a) The input original image with 128x128 pixels, (b) the resampled image with one output pixel (m=1).

The complexity Comp (F) of the implementation of this filtering depends on the factor m of the output pixels number. This complexity is given by the following equation:

$$\text{Comp}(F_m) = (31 m^2 + 46 m) n^2 + O(n) \tag{17}$$

The computation part of the matrix C involved in the whole computation is determined by the relation:

$$R(m) = \frac{Comp\,(C_m)}{Comp\,(C_m) + Comp\,(F_m)}$$
$$R(m) = \frac{20}{31m^2 + 46m + 20} + O(\frac{1}{n}) \tag{18}$$

as a result R(1) = 21 %, R(2) = 8.5 % , R(3) = 4.6 % and R(4) = 2,9 %. We remark that this part becomes small when the m value is different of one. This relation expresses the ratio of the number of operations involved in each part but not necessary the computing times because the speed of the computation essentially depends on the type of operations and data access.

6. Experimental Comparison of the Sequential CPU Times

The reported sequential experiments were run on one processor of a machine with two processors Pentium II running at 450 Mhz. We have measured the computation times of the matrix C construction and of the 16 pixels filter for 1, 2 and 3 output pixels (Table 1). The log-log graphs of Figure 2 shows the curves allure of the computation times with respect to the size n of the input image. The experimental computation time of the matrix C is inferior to this one of the 16 pixels filter which confirms the result of the complexities computation.

Table 1 : Computation times (s) of the matrix C construction and of some filters.

n	8	16	32	64	128	256	512	1024	2048
C construction	3.8e-5	1.3e-4	5.4e-4	2.3e-3	9.4e-3	4.9e-2	2.2e-1	9.4e-1	4.0e0
Filter 16/1	4.2e-5	2.5e-4	1.2e-3	5.4e-3	2.3e-2	9.7e-2	4.0e-1	1.8e 0	8.2e0
Filter 16/2	8.4e-5	5.2e-4	2.5e-3	1.1e-2	4.8e-2	2.0e-1	8.1e-1	3.6e 0	1.6e1
Filter 16/3	1.4e-4	8.8e-4	4.3e-3	1.9e-2	8.4e-2	3.4e-1	1.4e 0	6.8e 0	3.6e1

Table 2. Sequential computation times (s) of the matrix C and some 16/1 pixel filter for np = 1, 2, 4, 8 and 16 processors.

n	8	16	32	64	128	256	512	1024	2048
C, 1 proc.	3.8e-5	1.3e-4	5.4e-4	2.3e-3	9.4e-3	4.9e-2	2.2e-1	9.4e-1	4.0e0
16/1, 1 proc.	4.2e-5	2.5e-4	1.2e-3	5.4e-3	2.3e-2	9.7e-2	4.0e-1	1.8e 0	8.2e0
16/1, 2 proc.	1.1e-3	1.9e-4	5.9e-4	2.3e-3	9.8e-3	4.3e-2	1.8e-1	7.3e-1	2.9e0
16/1, 4 proc.	---	6.5e-4	5.4e-4	1.5e-3	5.5e-3	2.3e-2	9.3e-2	3.7e-1	1.5e0
16/1, 8 proc.	---	---	4.3e-4	9.3e-4	3.0e-3	1.1e-2	4.7e-2	1.8e-1	7.4e-1
16/1, 16 proc.	---	---	---	6.6e-4	1.7e-3	5.9e-3	2.5e-2	9.5e-2	3.8e-1

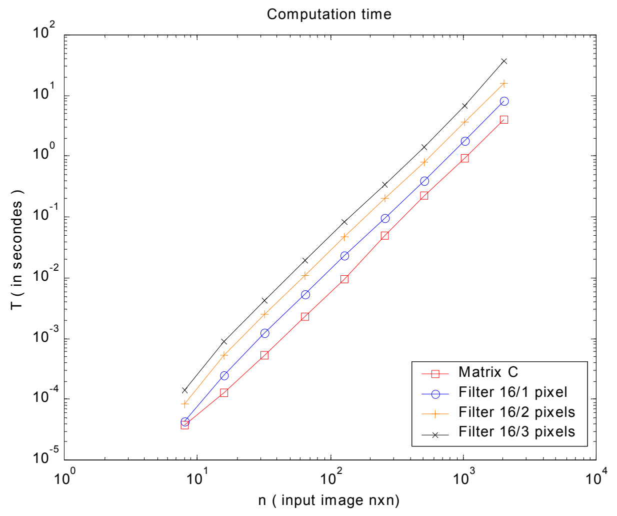


Figure 2. Computation times (s) of the matrix C construction and of some filters.

7. Parallel Computation of the 16 Pixels Filter

We consider now a linear network with n_p processors which communicate by message passing. In the first step, the matrix C is computed by one processor. The matrix C is therefore distributed by blocks of (n / n_p) rows to n_p processors (figure 3a). Each processor calculates the filter coefficients and sends simultaneously the three bordering rows of its part of the matrix C to its neighbor (figure 3b). At last, each processor applies the filter over its part of the matrix C.

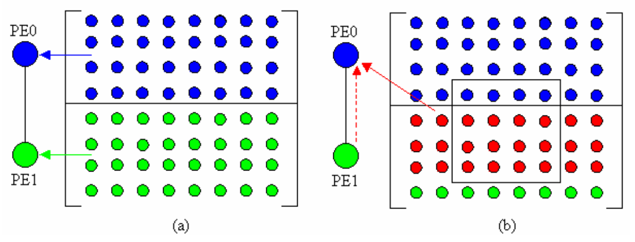


Figure 3. (a) The matrix C distribution to two processors, (b) Inter-processor communications of the 16 pixels filter.

To perform the parallel experiments, the described algorithm is implemented on a network including 8 bi-processor nodes (processors Pentium II running at 450 MHz) coupled by a 100Mbits/s network. The communication library used is the Message Passing Interface (MPI) [2, 8] based on the SPMD model (Single Program Multiple Data) [3].

The parallel approach of the 16 pixels filter has allowed to reduce and to obtain computation times smaller than the time spent into the construction of the matrix C (table 2). The log-log graphs of the figure 4 illustrate the evolution of the computation times. The resulting speed-ups and efficiencies are displayed in figure 5. Speedup S and efficiency E curves in the figure 5 illustrate that for the order $n \geq 64$ of the input images we obtain an acceptable efficiency. We even observe efficiencies superior to one due to a better memory management.

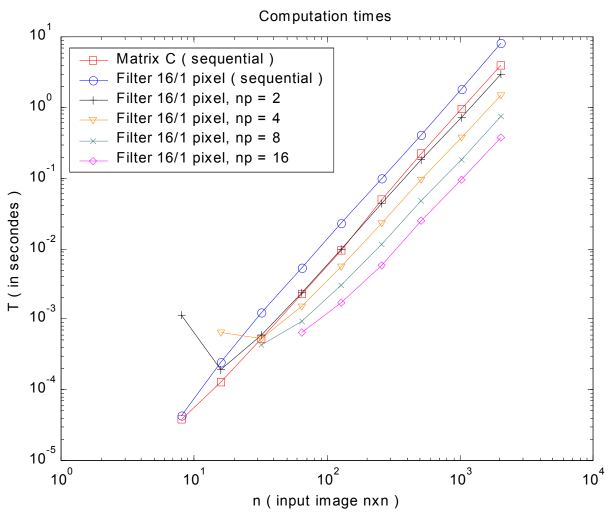


Figure 4. Sequential computation times (s) of the matrix C and some filter 16/1 pixel for np = 1, 2, 4, 8 and 16 processors.

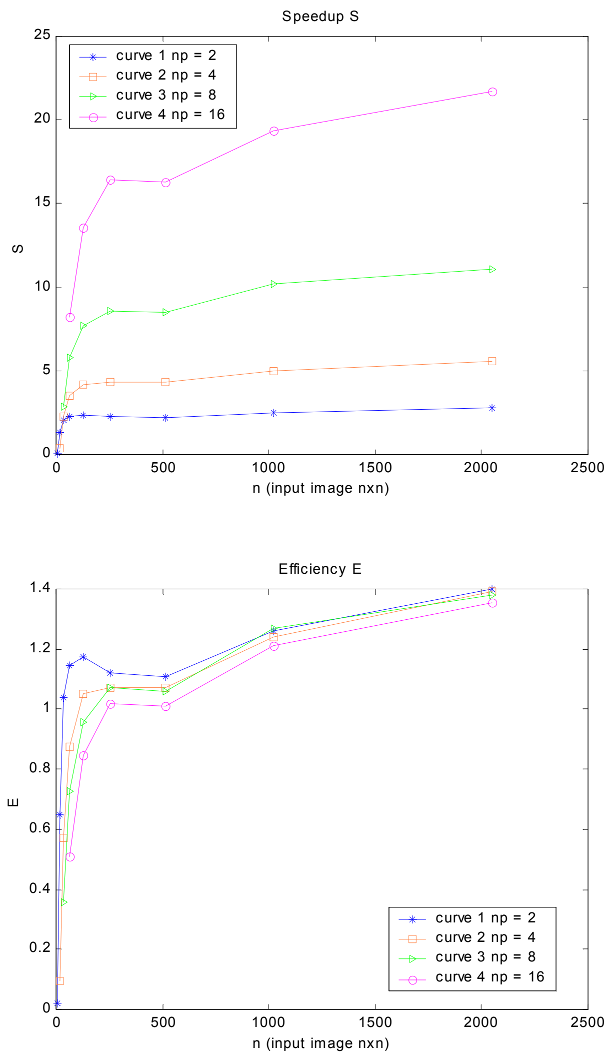


Figure 5. Speedup S and efficiency E of the uniform cubic B-spline resampling filter 16/1 pixel for np = 2, 4, 8 and 16 processors.

8. Conclusion

We have presented, in this paper, how the principal part of the computation, when resampling a digital image could be easily and efficiently parallelized with a small number of processors. The numerical experiments proved that this parallel approach has allowed to reduce the times for the filter application to a time smaller than the remaining sequential part. To improve the efficiency, the remaining part must be parallelized as well. It will be the purpose of a future research.

Acknowledgments

Authors wish to thank the SAGE project group (IRISA-INRIA, Rennes) for technical and scientific help.

References

[1] Andrews H. C. and Patterson II C. L., "Digital Interpolation of Discrete Images," *IEEE Transactions on. computers*, vol. C-25, pp. 196-202, February 1976.

[2] Gropp W., Lusk E., and Skjellum A., *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MA, MIT Press, Cambridge, 1994.

[3] Hwang K., *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw Hill, International Edition, 1993.

[4] James H. A., Patten C. J., and Hawick K. A., "Stencil Methods on Distributed High Performance Computers," *Technical Note DHPC-010,* University of Adelaide, Australia, June, 1997.

[5] Mahboub M., Philippe B., and Benyoucef B., "Calcul des Coefficients et du Filtre de Ré-échantillonnage D'images Numériques de la B-spline Cubique Uniforme," *Sciences & Technologie de l'Université Mentouri Constantine*, vol. 21B, pp. 63-70, June, 2004.

[6] Mahboub M. and Philippe B., "Parallélisation du ré-échantillonnage D'images," *in Proceedings of the 7th African Conference on Research in Computer Science*, vol. 4, pp. 91-98, Hammamet, Tunisia, 2004.

[7] Mahboub M. and Philippe B., "Ré-échantillonnage D'images Numériques par la B-spline Cubique Uniforme," *in Proceedings of the 6th African Conference on Research in Computer Science*, vol. 2, pp. 301-308, Yaounde, Cameroon, 2002.

[8] Pacheco P. S., A *User's Guide to MPI*, University of San Francisco, March, 1998.

[9] Pratt William K., *Digital Image Processing*, A Wiley Interscience Publication, 1978.

Mourad Mahboub received his Master degree in electronics physics in 1988 and his doctorate degree in physics in 2004 from Tlemcen University. Currently, he is an associate professor in physics in the Department of Physics, Tlemcen University. His areas of interest include digital image processing and parallel algorithms.

Djamal Lachachi received his Master degree in electronics physics in 1990. Currently, He is an assistant professor in Electronics Engineering Department at Tlemcen University. His areas of interest include digital image processing and parallel algorithms.