REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

# THÈSE

PRESENTÉE

**À L'UNIVERSITÉ ABOUBEKR BELKAID DE TLEMCEN**

**FACULTÉ DES SCIENCES**

**DEPARTEMENT DE PHYSIQUE**

POUR OBTENIR

**LE GRADE DE DOCTEUR D'ETAT ÈS SCIENCES PHYSIQUES**

PAR

**MOURAD MAHBOUB**

# Traitement d'images numériques par les B-splines : Calcul parallèle du filtre de ré-échantillonnage d'images numériques de la bilinéaire et de la B-spline cubique uniforme

Soutenue en Juin 2004 devant la commission d'examen

| | | |
|---|---|---|
| **Président** | M. BOUAMOUD Maamar | Professeur à l'université de Tlemcen |
| **Directeur de thèse** | M. B. BENYOUCEF | Professeur à l'université de Tlemcen |
| **Examinateurs** | MM. B. PHILIPPE | Professeur à l'université de Rennes 1 |
| | T. BENOUAZ | Professeur à l'université de Tlemcen |
| | F. BEREKSI REGUIG | Professeur à l'université de Tlemcen |
| | M. BENABDELLAH | Maître de Conférence à l'université de Tlemcen |

*A la mémoire de ma mère,*

*à ma famille ...*

# *Remerciements*

J'exprime ma profonde reconnaissance à Monsieur Benyoucef Boumédiène, Professeur à l'université Aboubekr Belkaid de Tlemcen pour avoir accepté d'être mon directeur de thèse. Je tiens à lui témoigner ma gratitude pour les encouragements qu'il m'a prodigué sans compter, pour l'encadrement exceptionnel qu'il m'a apporté tout au long de ce travail, et enfin, pour le climat de confiance et d'amitié dans lequel ces années se sont déroulées.

Je remercie Monsieur BOUAMOUD Mammar, Professeur à l'université Aboubekr Belkaid de Tlemcen, qui me fait aujourd'hui l'honneur de participer et de présider le jury de cette thèse.

J'exprime ma sincère reconnaissance à Monsieur Bernard Philippe, Professeur à l'université de Rennes 1 et Directeur de recherches à l'IRISA, pour m'avoir permis d'effectuer mes travaux de recherches dans le projet ALADIN, m'offrant ainsi un environnement très favorable et de m'avoir guidé avec une compétence jamais démentie, me permettant ainsi de mener à bien mes recherches.

Monsieur Benouaz Tayeb, Professeur à l'université Aboubekr Belkaid de Tlemcen, Monsieur Bereksi Reguig Fethi, Professeur à l'université Aboubekr Belkaid de Tlemcen, et Monsieur Benabdellah Mohamed, Maître de Conférence à l'université Aboubekr Belkaid de Tlemcen, ont accepté de juger ce travail et d'en être examinateurs. Qu'ils trouvent ici l'expression de mes plus vifs remerciements.

Je tiens également à remercier Madame Jocelyne Erhel professeur à l'université de Rennes 1 et Directeur de recherche à l'IRISA pour m'avoir accueilli dans le projet ALADIN, Monsieur Hugues Leroy et Monsieur Edouard Canot pour l'aide scientifique et technique, lors de la réalisation de ce travail.

Ma famille et mes amis m'ont beaucoup soutenu. Leur amitié m'a été plus que précieuse et je ne saurais les remercier suffisamment.

# Table des matières

# Chapitre 1

# Introduction générale

Le rôle des systèmes de traitement d'images et de reconnaissance de forme est d'extraire, à partir de données " images ", des informations qui sont présentées sous forme d'images, de graphiques ou de résultats numériques [RK82]. Les données " images " proviennent de capteurs qui font l'analyse bi ou multidimensionnelle de grandeurs physiques. Ces dernières proviennent de l'interaction du rayonnement électromagnétique, acoustique, etc., avec l'objet ou la scène que l'on désire étudier.

Si l'on remonte le spectre des rayonnements vers les grandes longueurs d'onde, on rencontre successivement la gammagraphie, la microscopie électronique, la radiographie, l'astronomie, la télédétection, la microscopie optique, la thermographie et le radar latéral.

Le volume de données à traiter pour résoudre un problème est fonction de l'aspect multitemporel des images, de l'aspect multispectral, de la résolution spatiale et de la dynamique radiométrique. A titre d'exemple, une image LANDSAT couvrant une surface au sol de 185 x 185 $km^2$ est définie par 2 340 lignes, chaque ligne contient 3 240 éléments d'image (ou pixel) ayant quatre valeurs radiométriques codées sur 8 bits. Une telle image est représentée par environ 30 millions de mots de 8 bits.

Dans certaines applications du traitement numérique d'images, il est souvent nécessaire de procéder à des transformations géométriques non linéaires des positions des pixels de la grille de l'image. Les transformées des positions des pixels ne coïncident généralement pas avec la grille de l'image d'entrée. Un algorithme de ré-échantillonnage permet d'obtenir la valeur de ces nouveaux pixels (figure 1).

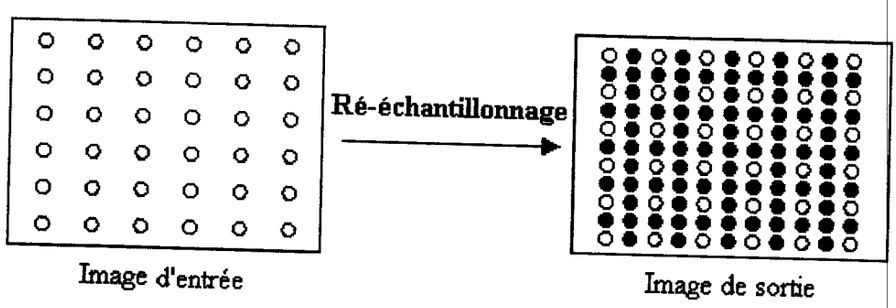



**Figure 1 :** *Ré-échantillonnage d'images numériques (avec 1 pixel de sortie).*

La reconstruction d'images numériques nécessite une image numérique. Dans le chapitre 2, nous rappelons l'échantillonnage d'une image analogique et les méthodes de reconstruction d'images du plus proche voisin, de la bilinéaire et de la B-spline cubique uniforme. L'interpolation idéale ou de Shannon qui nécessite un nombre infini de pixels de l'image d'entrée, permet de reconstruire l'image exacte, mais elle est impossible à réaliser.

Le ré-échantillonnage d'images numériques par les fonctions B-splines d'interpolation [Cox72] est souvent utilisé dans les applications du traitement numérique d'images, notamment la compression d'images, la tomographie, le recalage d'images multitemporelles multispectrales et multisources, l'animation et les zooms d'images utilisés actuellement dans les caméras et les appareils photographiques numériques.

Le ré-échantillonnage d'image du plus proche voisin est très rapide et conserve les valeurs radiométriques d'origines, mais présente un effet d'escalier sur les diagonales qui se traduit par une discontinuité visuelle.

Le ré-échantillonnage par la bilinéaire est utilisé, le plus souvent, dans les applications graphiques. Cette interpolation n'utilise que quatre pixels voisins de la grille de l'image pour chaque pixel de sortie. Cette technique est un peu moins rapide et engendre un léger effet de lissage.

Certaines applications professionnelles utilisent les interpolations cubiques. Le ré-échantillonnage par la B-spline cubique uniforme qui est le plus lent, mais le plus précis utilise un filtre appliqué sur seize pixels voisins, d'une image (ou matrice) C intermédiaire, pour calculer les pixels de sortie. Pour obtenir cette matrice C, nous devons résoudre un système d'équations linéaires. Andrews et Patterson ont introduit en 1976 [AP76] une méthode de calcul par des multiplications de matrices utilisant une inversion explicite très coûteuse en $O(n^3)$ où n est la dimension de la matrice. Depuis, d'autres techniques en $O(n^2)$ ont été introduites pour calculer la matrice C : transformation en z [VV01, UAE91]; factorisation d'une matrice tridiagonale de passage A mentionnée dans [GV96].

Nous présentons dans le chapitre 3, les filtres de ré-échantillonnage des 4 et 16 pixels voisins, les complexités correspondantes et nous proposons un algorithme de résolution du système linéaire par FFT (Fast Fourier Transform) et par une factorisation de Cholesky

adaptée [MP02, GV96 ]. Nous considérons dans nos calculs une matrice A correspondante à une image périodique qui permet de considérer une matrice carrée dans le calcul de C. Bien que la matrice A ne soit plus bidiagonale, la complexité reste en $O(n^2)$. Cet algorithme peut aussi être appliqué à des images non périodiques. Nous comparons les temps de calcul dans le cas du ré-échantillonnage d'images réelles, de la construction de la matrice C et des filtres pour un, deux et trois pixels de sortie. Le calcul des complexités et les essais numériques ont montré que la part de calcul induite par le calcul de la matrice C dans l'ensemble des calculs est faible dès que le nombre de pixels de sortie est différent de un. Les temps de calcul de l'implémentation séquentielle ont permis de calculer la matrice C avec un temps de calcul inférieur à celui du filtre des 16 pixels voisins.

Le temps de calcul du ré-échantillonnage d'images de grandes dimensions ou de séquences d'images s'avère important vu le grand nombre d'opérations flottantes et la complexité des algorithmes. Ce temps de calcul dépend évidemment du nombre de pixels de sortie à déterminer, de la dimension de l'image d'entrée et de la complexité de l'algorithme utilisé. Nous avons développé un calcul parallèle du filtre de ré-échantillonnage. Cette approche parallèle nous a permis de réduire le temps global de calcul grâce en particulier à une meilleure gestion de la mémoire.

La nature des problèmes physiques détermine le choix du langage et des bibliothèques de logiciels à utiliser correspondants aux caractéristiques des machines parallèles [Eti94]. Nous rappelons dans le chapitre 4, les caractéristiques des différents modèles de machines parallèles selon la classification de Flynn et les principales bibliothèques de communication et de calcul utilisées pour les grandes applications scientifiques ainsi que les critères d'évaluation des performances. Plusieurs plat-formes parallèles à hautes performances utilisent actuellement la bibliothèque de communication Message Passing Interface (MPI) [Pac98]. Nous avons choisi, dans notre travail, la bibliothèque MPI avec le mode SPMD (Single Program Multiple Data) pour obtenir une implémentation efficace sur des machines à passage de messages à mémoire distribuée [Hwa93]. Nous avons défini dans ce chapitre les sous programmes de la bibliothèque de communication MPI de l'environnement, de communications point à point, de communications collectives, de types de données dérivés et de topologies. Nous avons utilisé ces principaux sous programmes dans l'implémentation de nos codes parallèles.

Nous proposons dans le chapitre 5, un calcul parallèle des filtres de ré-échantillonnage des quatre et seize pixels voisins. Nous définissons la distribution des données de l'image d'entrée (dans le cas de la bilinéaire) ou de la matrice C (dans le cas de la B-spline cubique uniforme) aux processeurs d'un réseau linéaire ou d'une grille 2D de processeurs. Les valeurs d'entrée sont distribuées aux $n_p$ processeurs à raison d'un bloc de $(n / n_p)$ lignes par processeurs dans le cas d'un réseau linéaire ou d'un bloc de $(n / n_p \times n / n_p)$ dans le cas d'une grille de processeurs. Dans le cas d'un réseau linéaire, chaque processeur calcule les coefficients du filtre et en même temps transmet à ses voisins les lignes (une ou trois selon la méthode) frontières de sa partie des données d'entrée. Dans le cas d'une grille de processeurs chaque processeur transmet les lignes et colonnes frontières ainsi que les données diagonales. Enfin, chaque processeur applique le filtre sur sa partie des données d'entrée . Nous considérons dans ce chapitre, un réseau linéaire et une grille 2D de $n_p$ processeurs communicant par messages. Pour réaliser les tests, le programme décrit a été porté sur une grappe de 8 nœuds bi-processeurs (processeurs Pentium II fonctionnant à 450 Mhz ) couplés par un réseau de 100Mbits/s. La bibliothèque de communication utilisée est la bibliothèque Message Passing Interface (MPI) avec le mode SPMD (Single Program Multiple Data) [Hwa93]. Nous présentons, dans ce chapitre, comment, lors du ré-échantillonnage d'une image numérique, la principale partie des calculs pouvait être facilement et efficacement parallélisée sur un faible nombre de processeurs. Les essais numériques et les calculs de performance ont montré que cette approche parallèle a permis de réduire et d'obtenir des temps de calcul du filtre inférieurs à celui de la partie séquentielle restante. Les valeurs mesurées de l'accélération et de l'efficacité permettent de montrer qu'on commence à obtenir de bonnes accélérations et efficacités pour des images d'entrée de dimensions n supérieur ou égal à 64. En plus de la parallélisation des calculs, on bénéficie de la meilleure gestion de la mémoire (moins de défaut dans la pagination) qui apporte des efficacités supérieures à l'unité.

Nous présentons dans le chapitre 6 les sous programmes séquentiels du calcul de la matrice C par la factorisation de Cholesky adaptée et des filtres des 4 et 16 pixels ainsi que les sous programmes parallèles pour transférer les données frontières et pour calculer les limites d'entrée et de sortie des fenêtres d'images de chaque processeur du réseau linéaire et de la grille de processeurs. Nous avons développé et implémenté ces codes en fortran. Nous définissons dans cette partie leurs paramètres d'entrée-sortie.

Dans le chapitre 7 nous présentons une conclusion générale à notre travail ainsi que nos perspectives pour paralléliser la partie séquentielle restante pour améliorer l'efficacité. L'algorithme de calcul de C par FFT peut facilement être parallélisé en utilisant la FFT 2D parallèle [Bai87, Swa82]. D'autres calculs parallèles peuvent être développés en utilisant la méthode directe de calcul de la matrice C en faisant une décomposition de domaines[DM97]. Ceci fera l'objet de futures recherches.

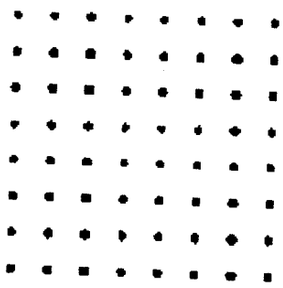
Enfin, nous présentons nos articles publiés, dans l'annexe publications, l'annexe proceedings et l'annexe communications.

# Chapitre 2
# Echantillonnage et reconstruction d'images numériques

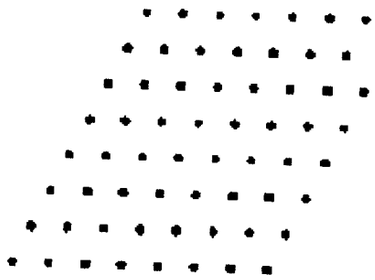
La reconstruction d'images numériques nécessite une image numérique qui est l'échantillonnage d'une image analogique à une certaine fréquence. La reconstruction correspond à un changement de cette fréquence ou encore au ré-échantillonnage de l'image discrète initiale.

## 2.1 Echantillonnage d'images

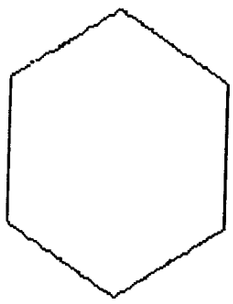
L'échantillonnage d'images est similaire à celui des signaux mono-dimensionnels. La théorie de Shannon s'applique de la même manière. Parmi tous les échantillonnages possibles, on rencontre l'échantillonnage rectangulaire figure 2a, l'échantillonnage parallélogramme figure 2b et l'échantillonnage hexagonal figure 2c [VVPL02].



(a)



(b)



(c)

**Figure 2 :** *(a) Echantillonnage rectangulaire, (b) échantillonnage parallélogramme, (c) échantillonnage hexagonal.*

• L'échantillonnage rectangulaire ou carré est régulier et le plus souvent utilisé. On obtient l'image échantillonnée $g(x, y)$ en faisant le produit de l'image originale $f(x, y)$ par la brosse d'impulsions de Dirac $b(x, y)$.

$$g(x, y) = f(x, y).\ b(x, y) \qquad (2.1)$$

La brosse d'impulsions de Dirac $b(x, y)$ peut être considérée comme le produit de deux peignes étendus, l'un dans la direction des $x$, $p_x(x, y)$ de période $\Delta x$ et l'autre dans la direction des $y$, $p_y(x, y)$ de période $\Delta y$.

$$p_x(x, y) = \sum_{i=-\infty}^{+\infty} \delta(x - i\Delta x) \qquad (2.2)$$

$$p_y(x, y) = \sum_{j=-\infty}^{+\infty} \delta(y - j\Delta y) \qquad (2.3)$$

$$g(x, y) = f(x, y)\ .\ \sum_{i=-\infty}^{+\infty} \delta(x - i\Delta x)\ .\ \sum_{j=-\infty}^{+\infty} \delta(y - j\Delta y) \qquad (2.4)$$

Dans le domaine fréquentiel, la transformée de Fourier du peigne étendu $p_x(x, y)$ de période $\Delta x$ est un peigne d'impulsions de Dirac $p_x(u, v)$ situé sur l'axe $Ou$ et de période $2\pi/\Delta x$. De même, $p_y(u, v)$ est situé sur l'axe $Ov$ et de période $2\pi/\Delta y$.

## 2.2 Reconstruction d'images numériques

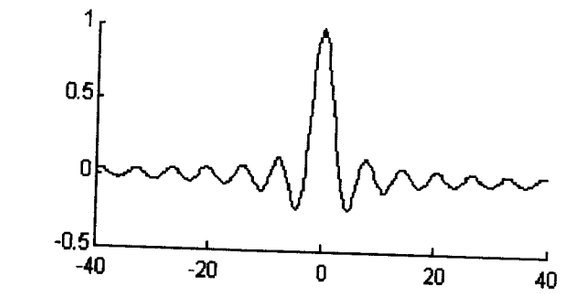
Dans plusieurs applications de traitement numérique d'images, il est souvent nécessaire de faire des transformations géométriques non linéaires des positions des pixels de la grille de l'image comme par exemple le cas du recalage géométrique d'images multitemporelles. Les transformées des positions des pixels ne coïncident généralement pas avec la grille de l'image. Un algorithme de ré-échantillonnage permet d'obtenir la valeur de ces pixels intermédiaires. Le temps de calcul dépend de la complexité de la fonction d'interpolation utilisée. L'interpolation idéale ou de Shannon à deux dimensions (cas des images) est définie, dans le cas d'un échantillonnage régulier et uniforme avec $\Delta x$ et $\Delta y$ les pas d'échantillonnage suivant les directions $x$ et $y$ respectivement, par la relation suivante:

$$F_r(x) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} F(i\Delta x, j\Delta y) R(x - i\Delta x, y - j\Delta y) \qquad (2.5)$$

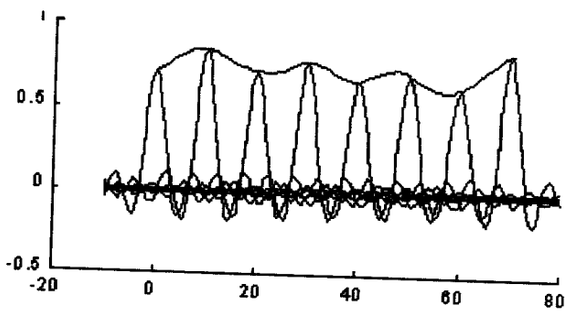
avec:

$$R(x,y)=\frac{\sin\omega_1 x \sin\omega_2 y}{\omega_1 x \quad \omega_2 y} \qquad (2.6)$$

Les pulsations $\omega_1$ et $\omega_2$ sont les pulsations de coupure suivant les directions x et y respectivement, $F(i\Delta x, j\Delta y)$ est la valeur du pixel (i, j) de l'image d'entrée originale à reconstruire et $F_r(x, y)$ est l'image de sortie finale. L'interpolation idéale représentée par la figure 3 dans le cas à une dimension, permet de reconstruire l'image exacte, mais elle est difficile à réaliser. Plusieurs fonctions, dont la réalisation est plus ou moins compliquée, sont utilisées pour approcher la fonction R(x,y). Parmi ces fonctions, on peut citer les fonctions d'interpolation du plus proche voisin (figure 4), linéaire (figure 5) et la B-spline cubique uniforme (figure 6) [HA78, Pra78]. L'interpolation bilinéaire est utilisée, le plus souvent, dans les applications graphiques. Cette interpolation utilise uniquement quatre pixels voisins de la grille de l'image pour chaque pixel de sortie. Par contre, certaines applications professionnelles utilisent les interpolations cubiques. Le ré-échantillonnage par la B-spline cubique uniforme utilise seize pixels voisins, d'une image (ou matrice) C intermédiaire, pour calculer les pixels de sortie. Pour obtenir cette matrice C, nous devons résoudre un système d'équations linéaires.

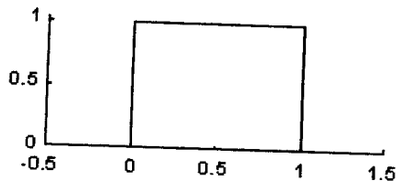



(a)

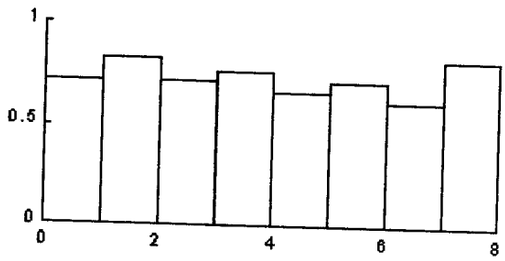



(b)

**Figure 3 :** *Interpolation idéale : (a) Fonction de base, (b) fonction reconstruite.*

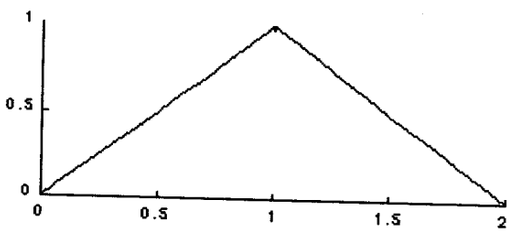



(a)

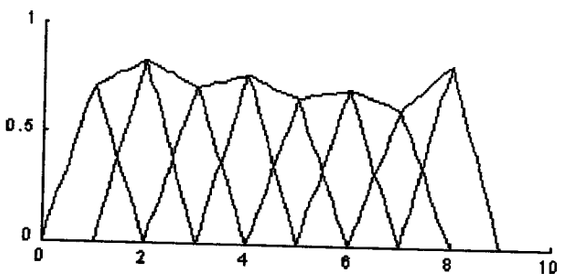



(b)

**Figure 4 :** *Interpolation du plus proche voisin : (a) Fonction de base, (b) fonction reconstruite.*




(a)




(b)

**Figure 5 :** *Interpolation linéaire : (a) Fonction de base et (b) fonction reconstruite.*

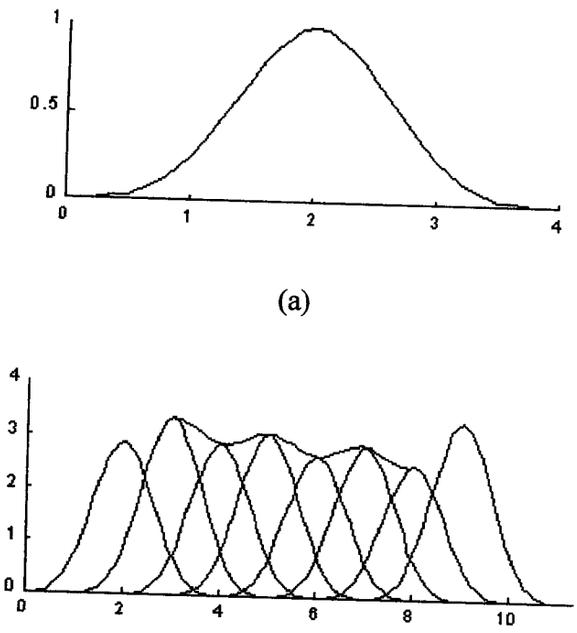



**Figure 6 :** *Interpolation par la B-spline cubique uniforme : (a) Fonction de base, (b) fonction reconstruite.*

# Chapitre 3
# Reconstruction d'images numériques par les B-splines

Nous déterminons, dans ce chapitre, les algorithmes séquentiels de ré-échantillonnage d'images numériques par la méthode du plus proche voisin, de la linéaire et de la B-spline cubique uniforme.

### 3.1 Ré-échantillonnage par les fonctions splines

Si on considère s(x) la spline de base[Boo78], la reconstruction d'un signal continu à une dimension peut être définie par la relation suivante:

$$g(x) = \sum_{i=1}^{n} c_i \, s_i(x) \qquad (3.7)$$

Le signal numérique d'entrée g (n) vérifie l'équation matricielle suivante :

$$g = A\,c \qquad (3.8)$$

La matrice A est une matrice (n x n) dont les coefficients $a_{ij}$ sont égaux à la valeur de la $j^{ème}$ spline au $i^{ème}$ échantillon. Le vecteur des coefficients c est déterminé par inversion de l'équation (3.8) :

$$c = A^{-1} g \qquad (3.9)$$

Le signal ré-échantillonné à une dimension d (M) ( M>n) est donné par l'équation suivante :

$$d = B\,c \qquad (3.10)$$

où la matrice *B* (M, n) est calculée par les splines de bases $s_i(x)$ aux valeurs du signal de sortie.

Dans le cas d'une image, la reconstruction dans le cas continu est définie par l'équation suivante:

$$g(x,y) = \sum_{i=1}^{n} \sum_{j=1}^{n'} c_{ij} \, s_i(x) s_j(x) \qquad (3.11)$$

Si la matrice G (n, n') représente l'image numérique d'entrée, elle vérifie l'équation suivante:

$$G = A_n C A_{n'} \qquad (3.12)$$

Les deux matrices carrées $A_n$ *et* $A_{n'}$ sont calculées respectivement par les splines de bases $s_i(x)$ et $s_j(y)$ aux pixels d'entrée de G. La matrice $C$ est l'inconnue; elle sera calculée par inversion de l'équation (3.12) pour aboutir à:

$$C = A_n^{-1} G A_{n'}^{-1} \qquad (3.13)$$

L'image ré-échantillonnée $G'$ (M, M') avec M>n et M'>n' est déterminée par :

$$G' = B_1 C B_2^T \qquad (3.14)$$

où les deux matrices $B_1$ (M, n) et $B_2^T$ (n', M') sont calculées respectivement par les splines de bases $s_i(x)$ et $s_j(y)$ aux pixels de sortie. Dans la suite nous nous restreignons à des images carrées ce qui entraîne n = n' et M = M'.

### 3.2 Interpolation du plus proche voisin

La fonction de base s($x$) est définie par l'équation suivante :

$$s(x) = \begin{cases} 1 & si \quad 0 < x < \Delta x \\ 0 & ailleurs \end{cases} \qquad (3.15)$$

La matrice $A$ est égale à la matrice identité $I$, on obtient :

$$c = g \qquad (3.16)$$

La matrice $B$ calculée est donnée par :

*3.2 Interpolation du plus proche voisin*

$$B=\begin{bmatrix}
0 & 0 & \dots & 0 & 0 & 1 \\
 & & \vdots & & & \\
0 & 0 & \dots & 0 & 0 & 1 \\
\hline
1 & 0 & \dots & 0 & 0 & 0 \\
 & & \vdots & & & \\
1 & 0 & \dots & 0 & 0 & 0 \\
\hline
 & & \vdots & & & \\
\hline
0 & 0 & \dots & 0 & 1 & 0 \\
 & & \vdots & & & \\
0 & 0 & \dots & 0 & 1 & 0 \\
\hline
0 & 0 & \dots & 0 & 0 & 1 \\
 & & \vdots & & & \\
0 & 0 & \dots & 0 & 0 & 1
\end{bmatrix} \qquad (3.17)$$

Le signal numérique de sortie à une dimension est défini par,

$$g(x)=g(x_i)\ avec\ (2i-1)\frac{\Delta x}{2}<x<(2i+1)\frac{\Delta x}{2} \qquad (3.18)$$

$\Delta x$ : est le pas de l'échantillonnage d'entrée.

L'image de sortie peut être représentée par la matrice $G$ suivante :

$$G=\left[\begin{array}{c|c|c|c|c}
\begin{matrix} G_{n,n}\dots G_{n,n} \\ \vdots \\ G_{n,n}\dots G_{n,n} \end{matrix} & \begin{matrix} G_{n,1}\dots G_{n,1} \\ \vdots \\ G_{n,1}\dots G_{n,1} \end{matrix} & \dots & \begin{matrix} G_{n,n-1}\dots G_{n,n-1} \\ \vdots \\ G_{n,n-1}\dots G_{n,n-1} \end{matrix} & \begin{matrix} G_{n,n}\dots G_{n,n} \\ \vdots \\ G_{n,n}\dots G_{n,n} \end{matrix} \\
\hline
\begin{matrix} G_{1,n}\dots G_{1,n} \\ \vdots \\ G_{1,n}\dots G_{1,n} \end{matrix} & \begin{matrix} G_{1,1}\dots G_{1,1} \\ \vdots \\ G_{1,1}\dots G_{1,1} \end{matrix} & \dots & \begin{matrix} G_{1,n-1}\dots G_{1,n-1} \\ \vdots \\ G_{1,n-1}\dots G_{1,n-1} \end{matrix} & \begin{matrix} G_{1,n}\dots G_{1,n} \\ \vdots \\ G_{1,n}\dots G_{1,n} \end{matrix} \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots \\
\hline
\begin{matrix} G_{n-1,n}\dots G_{n-1,n} \\ \vdots \\ G_{n-1,n}\dots G_{n-1,n} \end{matrix} & \begin{matrix} G_{n-1,1}\dots G_{n-1,1} \\ \vdots \\ G_{n-1,1}\dots G_{n-1,1} \end{matrix} & \dots & \begin{matrix} G_{n-1,n-1}\dots G_{n-1,n-1} \\ \vdots \\ G_{n-1,n-1}\dots G_{n-1,n-1} \end{matrix} & \begin{matrix} G_{n-1,n}\dots G_{n-1,n} \\ \vdots \\ G_{n-1,n}\dots G_{n-1,n} \end{matrix} \\
\hline
\begin{matrix} G_{n,n}\dots G_{n,n} \\ \vdots \\ G_{n,n}\dots G_{n,n} \end{matrix} & \begin{matrix} G_{n,1}\dots G_{n,1} \\ \vdots \\ G_{n,1}\dots G_{n,1} \end{matrix} & \dots & \begin{matrix} G_{n,n-1}\dots G_{n,n-1} \\ \vdots \\ G_{n,n-1}\dots G_{n,n-1} \end{matrix} & \begin{matrix} G_{n,n}\dots G_{n,n} \\ \vdots \\ G_{n,n}\dots G_{n,n} \end{matrix}
\end{array}\right] \qquad (3.19$$

### 3.3 Interpolation linéaire

La fonction de base de l'interpolation linéaire est donnée par l'équation suivante :

$$s(x) = \begin{cases} x & si\ 0 \le x \le \Delta x \\ -(x-2) & si\ \Delta x \le x \le 2\Delta x \\ 0 & ailleurs \end{cases} \qquad (3.20)$$

La matrice $A$ est égale à la matrice identité $I$ ce qui entraîne que la matrice $C$ est égale à la matrice $G$ de l'image numérique d'entrée :

$$C = G \qquad (3.21)$$

La matrice B calculée s'écrit sous la forme suivante:

$$B = \begin{bmatrix} c_1 & 0 & 0 & \dots & 0 & 0 & a_1 \\ & & & \vdots & & & \\ c_{M/n} & 0 & 0 & \dots & 0 & 0 & a_{M/n} \\ \hline a_1 & c_1 & 0 & \dots & 0 & 0 & 0 \\ & & & \vdots & & & \\ a_{M/n} & c_{M/n} & 0 & \dots & 0 & 0 & 0 \\ \hline & & & \vdots & & & \\ \hline 0 & 0 & 0 & \dots & 0 & a_1 & c_1 \\ & & & \vdots & & & \\ 0 & 0 & 0 & \dots & 0 & a_{M/n} & c_{M/n} \end{bmatrix} \qquad (3.22)$$

avec $a_\alpha = 1 - c_\alpha$, $c_\alpha = (\alpha - 1)(M-1)/(n-1)$ et $\alpha \in [1, (M-1)/(n-1)]$ (3.23)

#### 3.3.1 Calcul de la complexité et du filtre de ré-échantillonnage 1D

Les coefficients du filtre de ré-échantillonnage $F_\alpha$ sont déterminés par les équations ( 3.10, 3.22, 3.23 ).

$$F_\alpha = \begin{bmatrix} a_\alpha \\ c_\alpha \end{bmatrix}, \alpha \in [1, (M-1)/(n-1)] \qquad (3.24)$$

Ce filtre est appliqué sur la fenêtre $g_i$ à deux éléments du signal original d'entrée g. Les valeurs du signal de sortie sont obtenues en faisant la somme des produits de chaque

coefficient du filtre par la valeur correspondante de la fenêtre $g_i$. Le filtre est déplacé sur le signal d'entrée g et cette opération est appliquée pour chaque valeur de sortie. Cette opération est connue sous le nom de « méthode des stencils » ou méthode de kernel [JPH97].

L'implémentation de l'algorithme permet de montrer que la complexité est en O(n) de l'algorithme.

### 3.3.2 Calcul de la complexité et du filtre de ré-échantillonnage 2D

En utilisant les équations (3.14, 3.22, 3.23), on peut facilement calculer les coefficients des filtres $F_{\alpha\ \beta}$ de ré-échantillonnage des 4 pixels voisins de la bilinéaire pour calculer m pixels de sortie dans chaque direction:

$$F_{\alpha\beta} = \begin{bmatrix} a_\alpha a_\beta & a_\alpha c_\beta \\ c_\alpha a_\beta & c_\alpha c_\beta \end{bmatrix} \qquad (3.25)$$

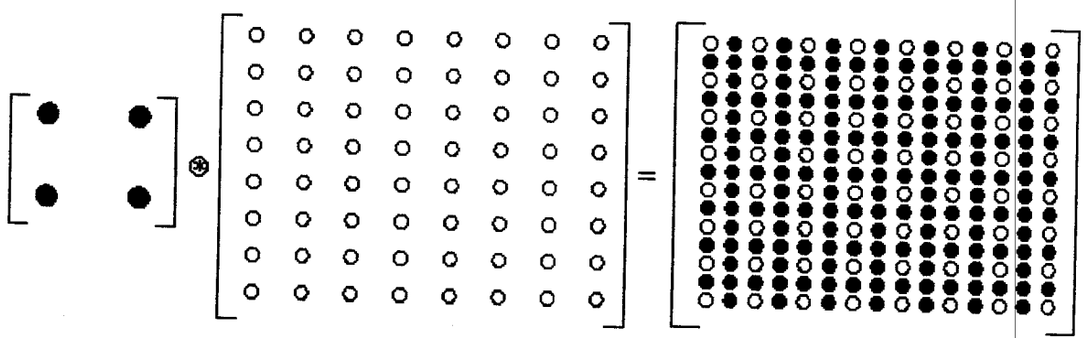
avec $(\alpha, \beta) \in [1, m[^2$.

Le filtre $F_{\alpha\,\beta}$ sera appliqué sur la fenêtre $G_{i\,j}$ de dimensions 2 x 2 de la matrice G de l'image d'entrée.

$$G_{i\,j} = \begin{bmatrix} g_{i\,j} & g_{i\,j+1} \\ g_{i+1\,j} & g_{i+1\,j+1} \end{bmatrix} \qquad (3.26)$$

avec $(i, j) \in [1, n-1]^2$.

La figure 7 représente l'application du masque du filtre 4 pixels voisins sur la matrice *G* pour obtenir une image ré-échantillonnée à 1 pixel de sortie. Les pixels de sortie de couleur bleu n'ont pas forcément la même valeur numérique.



Masque Image d'entrée G Image de sortie G'

**Figure 7 :** Filtre 4 pixels voisins appliqué à une image d'entrée pour obtenir une image ré-échantillonnée à 1 pixel de sortie.

Les coefficients des filtres de ré-échantillonnage de la première ligne et de la première colonne, dépendent des valeurs des coefficients $a_1$ et $c_1$ pour $\alpha = 1$ ou $\beta = 1$. Le calcul de ces coefficients nous donne les valeurs suivantes: $a_1 = 1$ *et* $c_1 = 0$.

Le filtre $F_{1\beta}$ de la première ligne se simplifie, il est égal à:

$$F_{1\beta} = \begin{bmatrix} a_\beta & c_\beta \\ 0 & 0 \end{bmatrix} \qquad (3.27)$$

Le filtre $F_{\alpha 1}$ de la première colonne est égal à:

$$F_{\alpha 1} = \begin{bmatrix} a_\alpha & 0 \\ c_\alpha & 0 \end{bmatrix} \qquad (3.28)$$

La complexité Comp (F) du filtre des 4 pixels voisins dépend du nombre m de pixels de sortie dans chaque direction. En appliquant les filtres des équations (3-25,27,28) sur l'image d'entrée et en dénombrant les opérations, on trouve :

$$\text{Comp } (F_m) = (7m^2 + 6m)\, n^2 + O(n) \qquad (3.29)$$

d'où Comp $(F_1) = 13\, n^2 + O(n)$, Comp $(F_2) = 40\, n^2 + O(n)$ et Comp $(F_3) = 81\, n^2 + O(n)$

### 3.3.3 Comparaison expérimentale des temps de calcul

Pour mener l'expérience qui suit, nous avons choisi un calculateur munie de deux processeurs Pentium II fonctionnant à 450 Mhz. Nous avons mesuré les temps de calcul (figure 8 ) du ré-échantillonnage d'une image pour un pixel de sortie courbe (Filtre 4/1 pixel), pour deux pixels de sortie courbe (Filtre 4/2 pixels) et pour trois pixels de sortie courbe (Filtre 4/3 pixels) en fonction de la dimension n de l'image d'entrée (voir le tableau 1 et la figure 8 ).

| | n | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Filtre 4/1 | | 0.4e-5 | 1.7e-5 | 0.7e-4 | 2.8e-4 | 1.2e-3 | 4.7e-3 | 2.0e-2 | 0.082 | 0.3e0 | 1.3e0 |
| Filtre 4/2 | | 0.7e-5 | 3.2e-5 | 1.4e-4 | 5.7e-4 | 2.4e-3 | 1.0e-2 | 4.1e-2 | 1.7e-1 | 0.7e0 | 2.7e0 |
| Filtre 4/3 | | 1.2e-5 | 5.3e-5 | 2.2e-4 | 9.5e-4 | 3.9e-3 | 1.7e-2 | 6.7e-2 | 2.7e-1 | 1.1e0 | 6.2e0 |

**Tableau 1 :** Temps de calcul (s) de plusieurs filtres.

$$t_n = T\, n^{\alpha}$$

où $t_n$ est le temps d'exécution du calcul considéré. Plus précisément, on a vérifié par une régression linéaire dans le graphique, que l'ordre de n était bien 2 et on a identifié les coefficients T correspondant à chaque courbe.

Régression linéaire pour n = 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048.

| | Ordre estimé α | Coefficient T estimé (s) | Vitesse en mflops |
|---|---|---|---|
| Filtre 4/1 pixel | 2.03 | 0.24e-6 | 54 |
| Filtre 4/2 pixels | 2.03 | 0.49e-6 | 81 |
| Filtre 4/3 pixels | 2.07 | 0.68e-6 | 119 |

**Tableau 2**

L'ordre α estimé est pratiquement constant et égal à 2 (voir tableau 2).

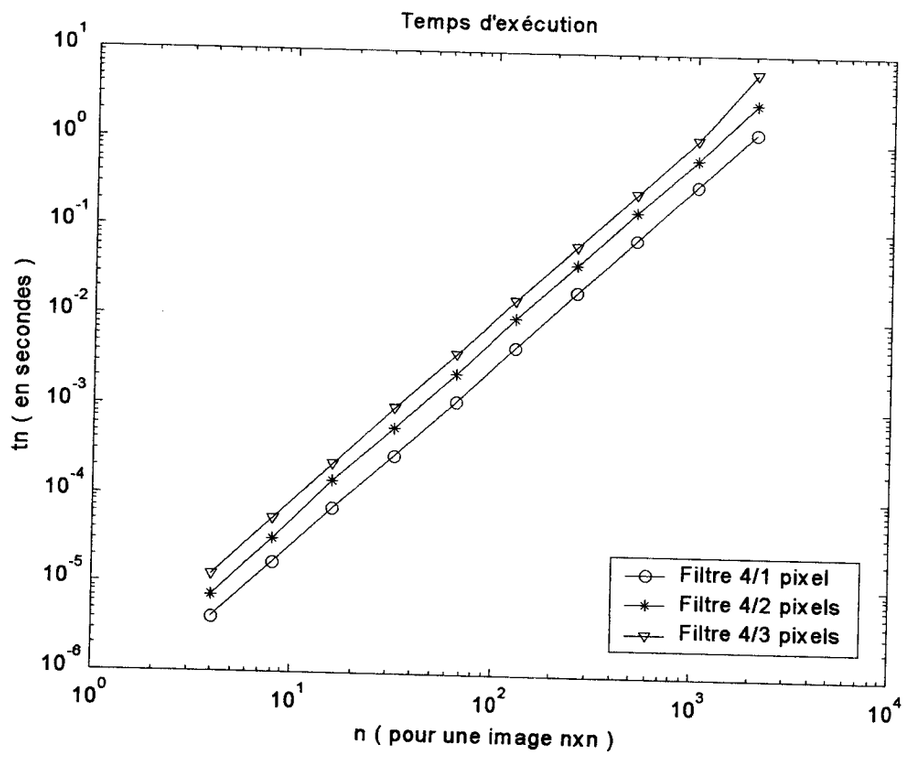



**Figure 8 :** Temps d'exécution : du filtre des 4 pixels voisins pour 1 pixel de sortie courbe (Filtre 4/1 pixel), pour 2 pixels de sortie courbe (Filtre 4/2 pixels) et pour 3 pixels de sortie courbe (Filtre 4/3 pixels).

### 3.4 Cas de la B-spline cubique uniforme

La fonction de base de la B-spline cubique uniforme s(x) est non nulle uniquement sur l'intervalle ]0, 4Δx[, elle peut s'écrire sous la forme suivante:

$$s(x) = x_+^{\,3} - 4(x-\Delta x)_+^{\,3} + 6(x-2\Delta x)_+^{\,3} - 4(x-3\Delta x)_+^{\,3} \qquad (3.30)$$

avec

$$x_+^{\,3} = \begin{cases} x^3 & si\ x > 0, \\ 0 & si\ x \le 0 \end{cases} \qquad (3.31)$$

La matrice $A$ calculée en supposant l'image périodique, est égale à :

$$A = \begin{bmatrix} 4 & 1 & 0 & & & & 0 & 1 \\ 1 & 4 & 1 & 0 & & & & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & & 0 \\ : & & & & : & & & : \\ 0 & & & \dots & & 1 & 4 & 1 & 0 \\ 0 & & & \dots & & 0 & 1 & 4 & 1 \\ 1 & 0 & & \dots & & & 0 & 1 & 4 \end{bmatrix} \qquad (3.32)$$

Le calcul des coefficients de la matrice $B$ est identique à celui de la matrice $A$, mais avec un pas d'échantillonnage plus faible égal à l'inverse du nombre de pixels à déterminer. Le calcul à une dimension montre que cette matrice est formée de n "blocs" dont chacun contient M / n lignes et n colonnes [Mah95] :

$$B = \begin{bmatrix} c_k & d_k & 0 & 0 & \dots & 0 & 0 & a_k & b_k \\ b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \\ a_k & b_k & c_k & d_k & \dots & 0 & 0 & 0 & 0 \\ & & & & : & & & & \\ b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \end{bmatrix} \qquad (3.33)$$

avec,

$$
\begin{aligned}
a_k &= (\beta_k + 3)^3 - 4(\beta_k + 2)^3 + 6(\beta_k + 1)^3 - 4\beta_k{}^3 \\
b_k &= (\beta_k + 2)^3 - 4(\beta_k + 1)^3 + 6\beta_k{}^3 \\
c_k &= (\beta_k + 1)^3 - 4\beta_k{}^3 \\
d_k &= \beta_k{}^3 \\
\beta_k &= kn/M,\ k \in [1, M/n]
\end{aligned}
\qquad (3.34)
$$

Les coefficients du filtre des 16 pixels voisins sont calculés par les équations (3-14,33,34).

### 3.4.1 Calcul de la matrice C par FFT

Pour résoudre le système linéaire de l'équation (3.6) par FFT [CT65, TL89], nous appliquons la matrice FFT Q et la matrice FFT inverse $Q^{-1}$ de la manière suivante :

$$Q.A.Q.Q^{-1}.C.Q^{-1}.Q.A.Q = Q.G.Q$$
$$Y.Q^{-1}.C.Q^{-1}.Y = Z$$
$$Q^{-1}.C.Q^{-1} = Y^{-1}.Z.\ Y^{-1} = B$$
$$C = Q.B.Q$$

Où Y et Z sont respectivement les matrices FFT 2D de la matrice A et de la matrice G de l'image d'entrée. La matrice C est déterminée par FFT 2D de la matrice B ( voir figure 98).

Nous remarquons que la matrice Y est une matrice creuse qui présente une certaine symétrie par rapport à j = n/2 +1. Les équations 3.35 et 3.36 représentent deux exemples de matrice Y pour n = 4 et n = 8.

$$Y = \begin{bmatrix} 24 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 \\ 0 & 0 & 8 & 0 \\ 0 & 16 & 0 & 0 \end{bmatrix} \qquad (3.35)$$

$$Y = \begin{bmatrix} 48.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 43.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 32 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 \\ 0 & 43.3 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (3.36)$$

Un calcul préalable nous permet de stocker les valeurs de $Y^{-1}$ dans des tables pour n = 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096.

Les complexités du calcul de la matrice Z et C sont proportionnelles à $O(n^2 \log_2 n)$. La complexité du calcul de la matrice B est en $O(2n^2)$.

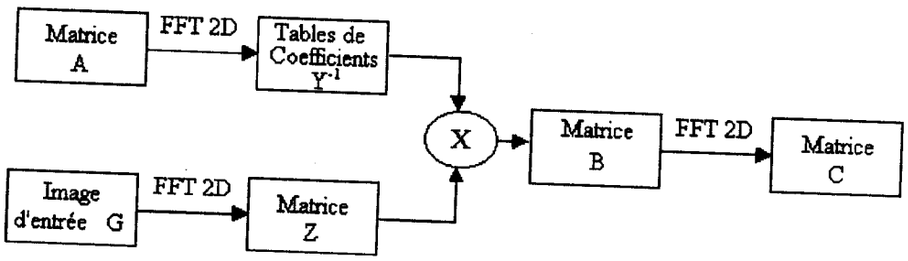



**Figure 9 :** Algorithme de calcul de C par FFT.

### 3.4.2 Factorisation de la matrice *A*

On décompose la matrice *A* d'ordre n de la manière suivante :

$$A = \left[\begin{array}{c|c} A_d & a \\ \hline a^T & 4 \end{array}\right] \qquad (3.37)$$

où $A_d$ est la matrice principale tridiagonale symétrique d'ordre (n -1), et $a$ est le vecteur :

$$a = (1, \ 0, \ \ldots, \ 0, \ 1)^T \in R^{n-1} \qquad (3.38)$$

*3.4 Cas de la B-spline cubique uniforme*

La matrice de factorisation de Cholesky $L$ de la matrice $A$ s'écrit en fonction de $L_d$, facteur de Cholesky de la matrice $A_d$ :

$$A_d = L_d L_d^T \qquad (3.39)$$

On recherche $L$ telle que :

$$A = L L^T \qquad (3.40)$$

$$L = \begin{bmatrix} L_d & 0 \\ l^T & \lambda \end{bmatrix} \qquad (3.41)$$

$L_d$ est une matrice bidiagonale. On doit trouver $L_d$, $l$ et $\lambda$ tels que:

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

Cela peut être réalisé par le calcul suivant:

$L_d$ : = facteur de Cholesky de $A_d$,

Résolution du système triangulaire $L_d l = a$,

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

La factorisation entraîne donc un nombre d'opérations flottantes proportionnel à n.

### 3.4.3 Calcul de la matrice C par factorisation de Cholesky

L'équation (3.12) précédente est équivalente à:

$$G = L L^T C L L^T$$

Elle peut être résolue par la résolution successive des systèmes triangulaires suivants:

$$G = L Y,$$

$$Y = L^T X,$$
$$X = Z L^T,$$
$$Z = C L$$

Chacune des étapes précédentes correspond à la résolution d'un système triangulaire avec $L$ ou $L^T$, à gauche ou à droite, et avec n seconds membres. Pour un second membre, la résolution de l'un de ces systèmes a une complexité égale à 5n + O(1). On en déduit une complexité totale de:

$$\text{Comp (C)} = 20n^2 + \text{O (n)}. \qquad (3.42)$$

### 3.4.4 Calcul des coefficients et de la complexité du filtre des 16 pixels

En utilisant les équations (3-10,33,34), on peut facilement calculer les coefficients des filtres $F_{k,u}$ des 16 pixels voisins de la B-spline cubique uniforme pour calculer m pixels de sortie dans chaque direction [MPB04] :

$$F_{k,u} = \begin{bmatrix} a_k a_u & a_k b_u & a_k c_u & a_k d_u \\ b_k a_u & b_k b_u & b_k c_u & b_k d_u \\ c_k a_u & c_k b_u & c_k c_u & c_k d_u \\ d_k a_u & d_k b_u & d_k c_u & d_k d_u \end{bmatrix} \qquad (3.43)$$

avec (k, u) ∈ [1, m[$^2$.

Le filtre $F_{k,u}$ à deux dimensions est appliqué sur la fenêtre $C_{i,j}$ de dimensions 4 x 4 de la matrice C de l'image originale.

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

avec (i, j) ∈ [2, n−2] $^2$.

Les coefficients du filtre de ré-échantillonnage de la première ligne et de la première colonne, dépendent des valeurs des coefficients $a_1$, $b_1$, $c_1$ et $d_1$ pour k = 1 ou u = 1. Le calcul de ces coefficients nous donne les valeurs suivantes: $a_1 = 1$, $b_1 = 4$, $c_1 = 1$ *et* $d_1 = 0$.

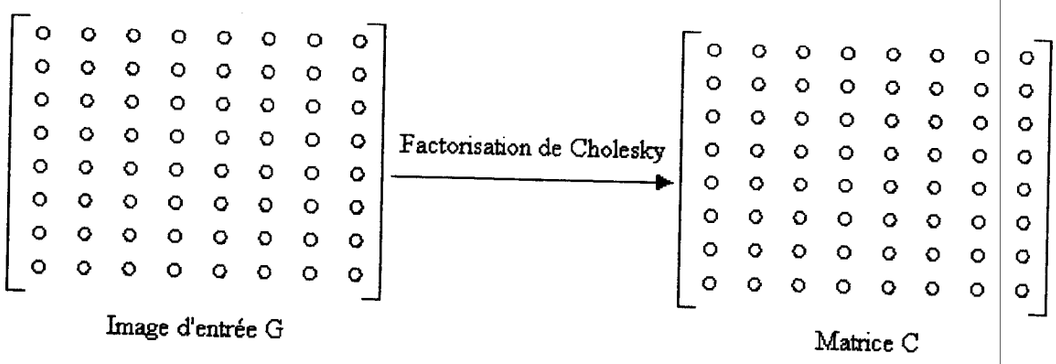
Le filtre $F_{1,u}$ de la première ligne est égal à:

$$F_{1,u} = \begin{bmatrix} a_u & 4a_u & a_u & 0 \\ b_u & 4b_u & b_u & 0 \\ c_u & 4c_u & c_u & 0 \\ d_u & 4d_u & d_u & 0 \end{bmatrix} \qquad (3.44)$$

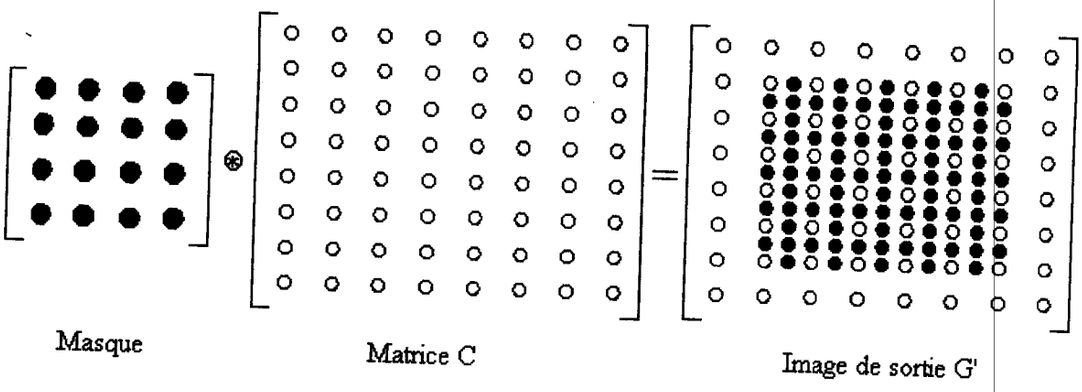
Le filtre $F_{k,1}$ de la première colonne est égal à:

$$F_{k,1} = \begin{bmatrix} a_k & b_k & c_k & d_k \\ 4a_k & 4b_k & 4c_k & 4d_k \\ a_k & b_k & c_k & d_k \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad (3.45)$$

La figure 10 représente les deux étapes de traitement du ré-échantillonnage par la B-spline cubique uniforme pour obtenir une image à 1 pixel de sortie dont les bords ne sont pas définis. Les pixels de sortie de couleur bleu n'ont pas forcément la même valeur numérique.




(a)




(b)

**Figure 10 :** *Etapes de traitement du ré-échantillonnage par la B-spline cubique uniforme : (a) Passage de l'image d'entrée G à la matrice C par factorisation de Cholesky, (b) filtre 16 pixels appliqué à la matrice C pour obtenir une image ré-échantillonnée à 1 pixel de sortie.*

La complexité Comp (F) du filtre des 16 pixels voisins dépend du nombre m des pixels de sortie dans chaque direction. La figure 11 représente le ré-échantillonnage d'une fenêtre d'image contenant quatre pixels pour un, deux et trois pixels de sortie utilisant un ensemble de seize pixels d'entrée voisins.

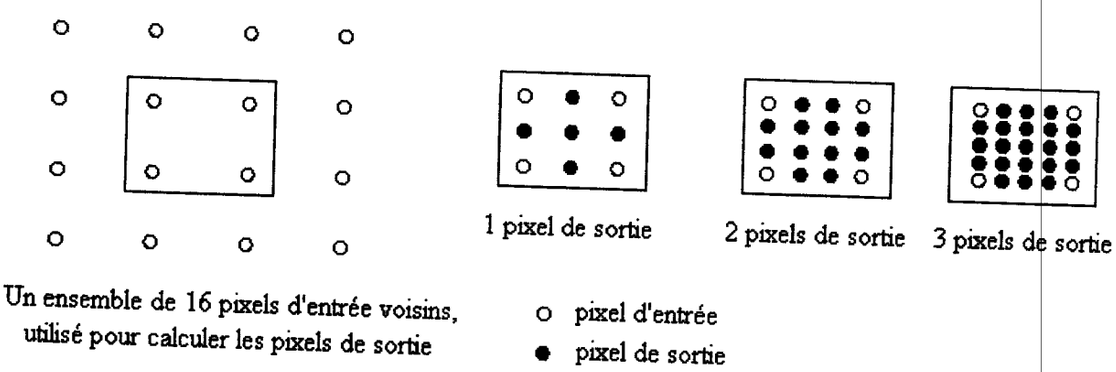



**Figure 11 :** *Ré-échantillonnage d'images avec 1pixel, 2 pixels et 3 pixels de sortie.*

En dénombrant les opérations, on trouve :

$$\text{Comp}(F_m) = (31\,m^2 + 46\,m)\,n^2 + O(n) \qquad (3.46)$$

d'où Comp $(F_1) = 77\,n^2 + O(n)$, Comp $(F_2) = 216\,n^2 + O(n)$ et Comp $(F_3) = 417\,n^2 + O(n)$.

La part de calcul induite par le calcul de la matrice C dans l'ensemble des calculs est déterminée par le rapport :

$$R(m) = \frac{Comp(C_m)}{Comp(C_m) + Comp(F_m)} = \frac{20}{31m^2 + 46m + 20} + O(\frac{1}{n}) \qquad (3.47)$$

ce qui donne R(1) = 21 %, R(2) = 8.5 % , R(3) = 4.6 % et R(4) = 2,9 %. On constate bien que cette part est faible dès que m est différent de un.

### 3.4.5 Comparaison expérimentale des temps de calcul

Les tests qui suivent ont été réalisés sur une station munie de deux processeurs Pentium II fonctionnant à 450 Mhz. Nous avons mesuré les temps de calcul de la construction de la matrice C, du filtre des 16 pixels pour 1, 2 et 3 pixels de sortie (voir le tableau 3 et la figure 12 ).

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Construction de C | 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 |
| Filtre 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 |
| Filtre 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 |
| Filtre 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 |

**Tableau 3 :** Temps de calcul (s) de la construction de C et de plusieurs filtres.

On vérifie bien grâce au graphique log-log que les temps d'exécution peuvent être estimés par des formules du type

$$t_n = T\, n^{\alpha}$$

où $t_n$ est le temps d'exécution du calcul considéré. Plus précisément, on a vérifié par une régression linéaire dans le graphique, que l'ordre de n était bien 2 et on a identifié les coefficients T correspondant à chaque courbe.

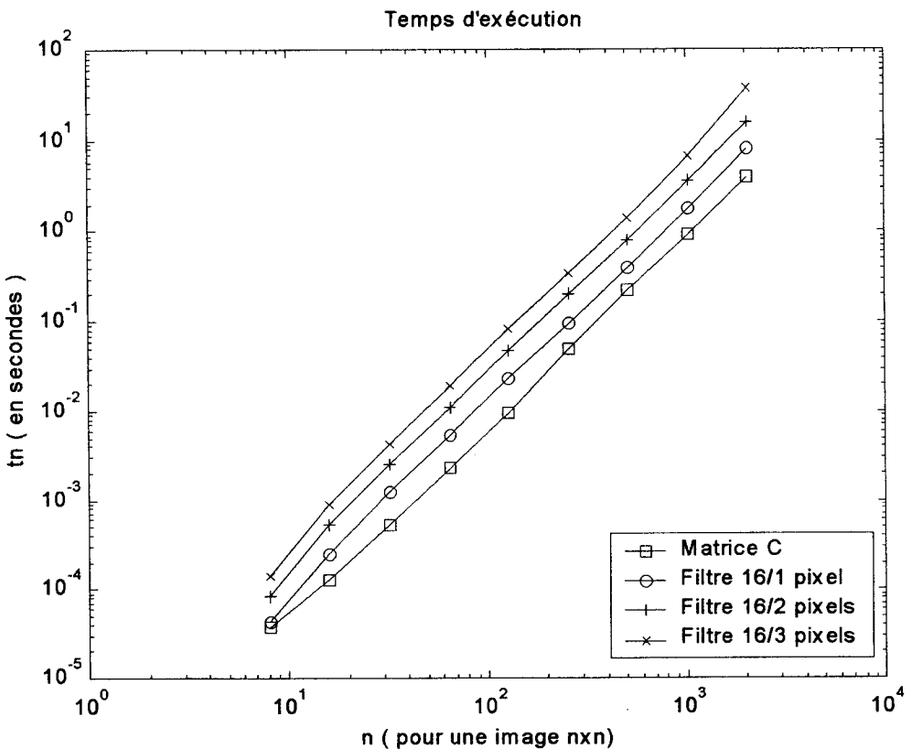



**Figure 12 :** *Temps de calcul (s) de la construction de C et de plusieurs filtres.*

Régression linéaire pour n = 8, 16, 32, 64, 128, 256, 512, 1024, 2048.

| | Ordre estimé $\alpha$ | Coefficient T estimé (s) | Vitesse en Mflops |
|---|---|---|---|
| Calcul de la matrice C | 2.14 | 0.0319e-5 | 63 |
| Filtre 16/1 pixel | 2.12 | 0.0746e-5 | 103 |
| Filtre 16/2 pixels | 2.11 | 0.1594e-5 | 135 |
| Filtre 16/3 pixels | 2.15 | 0.2316e-5 | 180 |

**Tableau 4**

Dans le cas des filtres des 16 pixels voisins, on remarque que l'ordre $\alpha$ estimé est pratiquement constant et égal à 2.1 (tableau 4). Cette différence est due aux termes de complexité inférieure négligés.

Les figures 13, 14 et 15 représentent respectivement le ré-échantillonnage d'une image réelle de 128x128 pixels pour 1, 2 et 3 pixels de sortie.

*3.4 Cas de la B-spline cubique uniforme*

(a) Image d'entrée

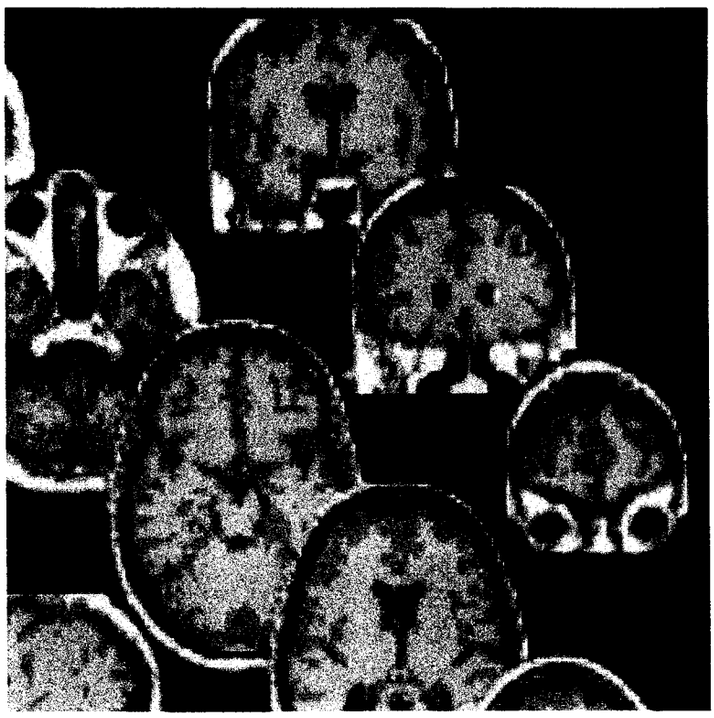
(b) Image de sortie ( ppv )

(b) Image de sortie ( Bilinéaire )

(c) Image de sortie ( B-spline cubique uniforme )

**Figure 13 :** *Images ré-échantillonnées avec 1 pixel de sortie :* ***(a)*** *L'image originale d'entrée de dimensions 128x128 pixels,* ***(b)*** *l'interpolation du plus proche voisin,* ***(c)*** *l'interpolation bilinéaire,* ***(d)*** *l'interpolation B-spline cubique uniforme.*

(a) Image de sortie ( ppv )



(b) Image de sortie ( Bilinéaire )

(c) Image de sortie ( B-spline cubique uniforme )

**Figure 14 :** *Images ré-échantillonnées avec 2 pixels de sortie : **(a)** L'interpolation du plus proche voisin, **(b)** l'interpolation bilinéaire, **(c)** l'interpolation B-spline cubique uniforme.*

(a) Image de sortie ( ppv )

(b) Image de sortie ( Bilinéaire )

(c) Image de sortie ( B-spline cubique uniforme )

**Figure 15 :** *Images ré-échantillonnées avec 3 pixels de sortie :* ***(a)*** *L'interpolation du plus proche voisin,* ***(b)*** *l'interpolation bilinéaire,* ***(c)*** *l'interpolation B-spline cubique uniforme.*

Le tableau 5 représente les temps d'exécution $T_{ex}$ du ré-échantillonnage de l'image réelle (128x128) pixels de la figure 13a correspondant aux trois méthodes pour un pixel, deux pixels et trois pixels de sortie ainsi que leur ordre de continuité.

| | $T_{ex}$ en seconde 1 pixel de sortie | $T_{ex}$ en seconde 2 pixel de sortie | $T_{ex}$ en seconde 3 pixel de sortie | Ordre de continuité |
|---|---|---|---|---|
| ppv | 0.12 e-2 | 0.27 e-2 | 0.47 e-2 | 0 |
| bilinéaire | 0.25e-2 | 0.55e-2 | 1.0e-2 | 1 |
| B-spline cubique uniforme | 1.4 e-2 | 2.6 e-2 | 4.4 e-2 | 2 |

**Tableau 5.** *Temps d'exécution et ordre de continuité.*

Le ré-échantillonnage d'images par la méthode du plus proche voisin est très rapide et conserve les valeurs radiométriques d'origines, mais présente un effet d'escalier sur les diagonales qui se traduit par une discontinuité visuelle (voir figures 13b, 14a et 15a). L'interpolation bilinéaire est un peu moins rapide et engendre un léger effet de lissage (voir figures 13c, 14b et 15b). L'interpolation B-spline cubique uniforme est la plus lente, mais elle est plus précise. On remarque qu'un effet de « flou » apparaît aux contours de l'image de sortie lorsque le nombre de pixels de sortie est important ceci est du au filtrage passe bas qui élimine les hautes fréquences.

Dans le cas de la bilinéaire et de la B-spline cubique uniforme, le calcul des complexités et les essais numériques ont montré que le coût de l'algorithme est $O(n^2)$. Dans le cas des images numériques d'entrée de grandes tailles le temps de calcul s'avère assez important.

Les architectures parallèles sont souvent utilisées et plusieurs algorithmes parallèles ont été développés [WS82, SSF82, CWWZW95, CS92, FLN89]. Nous présentons, dans le chapitre 5 , une approche parallèle en décomposition de domaine 1D et 2D du filtre de ré-échantillonnage des quatre et seize pixels voisins [Pin97a, Pin97b, PAM98]. Cette approche parallèle nous a permis de réduire le temps de calcul. Ce temps de calcul peut être inférieur à celui du calcul de la matrice C dans le cas de la B-spline cubique uniforme. Plusieurs plat-formes parallèles à haute performances utilisent actuellement la bibliothèque de communication Message Passing Interface (MPI). Nous avons choisi, dans notre travail, la bibliothèque MPI avec le mode SPMD (Single Program Multiple Data) pour obtenir une implémentation efficace sur des machines à passage de message à mémoire distribuée.

# Chapitre 4

# Parallélisme

## 4.1 Introduction

Notre travail est basé sur l'utilisation des machines parallèles au ré-échantillonnage d'images. Le domaine du parallélisme est généralement divisé en trois parties (ou espace) : espace des différents problèmes à traiter en parallèle, espace des logiciels ou environnement de programmation et l'espace des architectures parallèles. Les classes et la nature des problèmes déterminent le choix du langage et des bibliothèques de logiciels à utiliser correspondants aux caractéristiques des machines parallèles [Hwa93]. Dans ce chapitre, nous rappelons la classification selon Flynn [Fly79] des architectures parallèles, nous présentons les principales bibliothèques de communication et de calcul utilisées par les grandes applications scientifiques, et quelques langages de programmation en fortran que nous avons utilisé. Les performances des machines parallèles obtenues peuvent être évaluées suivant différents critères : mesures des temps, de l'accélération et de l'efficacité. Ces critères sont définis à la fin de ce chapitre.

## 4.2 Classification des architectures parallèles selon Flynn

Les problèmes de grandes tailles et complexes peuvent être exécutés séquentiellement, ou décomposés en tâches de tailles plus petites pour être exécutées en parallèle par des machines formées de plusieurs processeurs. Les communications entre ces différents processeurs dépendent essentiellement :

- Du contrôle des instructions et des données du programme. Le contrôle peut être centralisé ou distribué.
- De l'architecture des mémoires. On distingue des machines parallèles à mémoire partagée et celles à mémoire distribuée.
- Du type du réseau d'interconnexion.

La classification suivant Flynn (1966) des différentes architectures est basée sur le séquencement du flot d'instructions et des données du programme. Les quatre catégories sont :

- Le mode SISD- Single Instruction, Single Data Stream.
- Le mode SIMD- Single Instruction, Multiple Data Stream.
- Le mode MISD- Multiple Instruction, Single Data Stream (aucun exemple pratique).
- Le mode MIMD- Multiple Instruction, Multiple Data Stream. On peut ajouter une cinquième "pseudo-machine" le modèle SPMD- Single Program Multiple Data.

### 4.2.1 Le modèle d'architecture du mode SISD

Ce sont des ordinateurs scalaires et séquentiels de von Neumann. Le flot d'instructions du programme est séquencé par une horloge unique (figure 16).

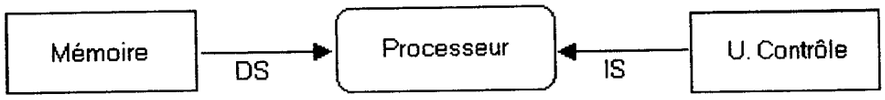



**Figure 16 :** *Calculateur SISD ( DS = Data stream, IS = Instruction Stream).*

### 4.2.2 Le modèle d'architecture du mode MISD

Chaque processeur a sa propre unité de contrôle et se partage une mémoire commune (figure 17). Chaque processeur effectue une opération différente, au même moment sur la même donnée. Il n'existe aucune machine construite sur ce modèle, c'est un modèle théorique sans intérêt.

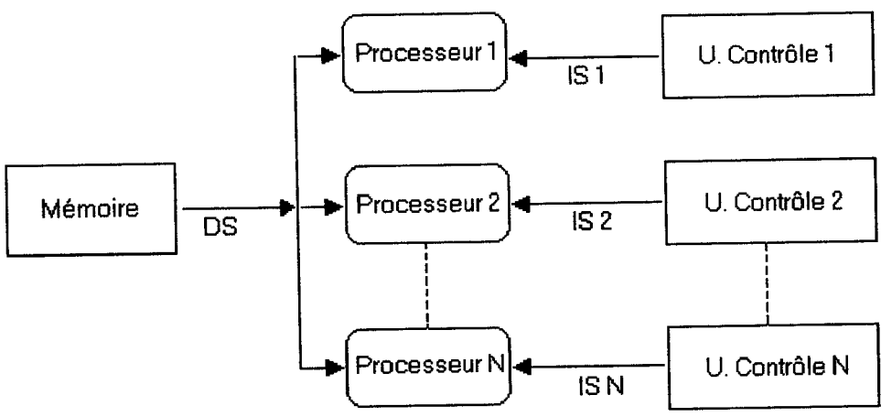



**Figure 17 :** *Calculateur MISD.*

### 4.2.3 Le modèle d'architecture du mode SIMD

Ces machines parallèles sont aussi des machines de von Neumann mais avec une plus forte puissance de calcul par rapport aux machines séquentielles. On distingue deux types majeurs : les machines SIMD vectorielles et les machines SIMD parallèles [SS84]. Les

machines SIMD vectorielles exécutent une seule instruction en plusieurs opérandes (type pipeline). Par contre, les machines SIMD parallèles exécutent les instructions une à une par plusieurs processeurs sur des données multiples. Fonctionnellement, on peut avoir les machines à mémoire commune ou distribuée (figure 18). Les seules machines parallèles SIMD qui existent sont celles à mémoire distribuée. En effet, les calculateurs SIMD possèdent un grand nombre de processeurs élémentaires (> 1000) ce qui rend irréalisable les machines à mémoire commune.

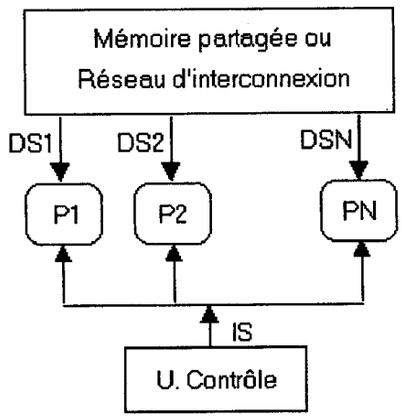



**Figure 18 :** *Calculateur SIMD.*

Dans le modèle d'exécution des machines SIMD à mémoire distribuée, chaque processeur est constitué d'unités arithmétiques et logiques programmables, possède une mémoire de faible capacité et exécute les instructions sous la seule impulsion du contrôleur. L'exécution de chaque instruction du programme par tous les processeurs est synchrone. Les processeurs qui exécutent l'instruction sont des processeurs actifs; les processeurs inactifs attendent la prochaine instruction. La Connexion Machine CM2, CM200 de TMC [CM-87] et la Maspar (Massively Parallel) MP-1 et MP-2 sont des exemples caractéristiques de cette classe de machine. Les machines SIMD sont particulièrement utiles pour traiter les problèmes à structure régulière où la même instruction s'applique à des sous-ensembles de données.

### 4.2.4 Le modèle d'architecture du mode MIMD

Parmi les ordinateurs parallèles rencontrés actuellement, les classes des machines MIMD sont les plus puissantes et les plus générales [CM-93, Cra94, Exe94]. Au contraire des

machines SIMD où chaque PE reçoit ses commandes du contrôleur central, les processeurs des machines MIMD reçoivent leurs commandes d'un programme stocké dans la mémoire principale. Chaque processeur exécute un flot d'instructions généré par sa propre unité de contrôle (i.e. chaque processeur est capable d'exécuter son propre programme sur des données différentes). Donc chaque processeur opère de façon asynchrone. La communication de données ou de résultats entre les processeurs peut se faire à travers une mémoire commune ou un réseau d'interconnexion (figure 19).

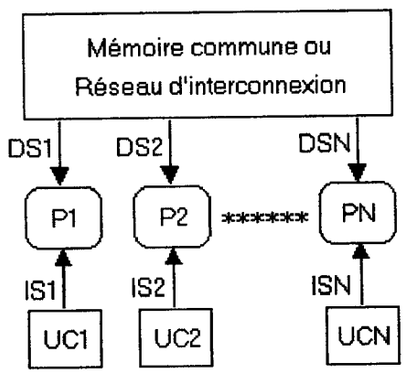



**Figure 19 :** *Calculateur MIMD.*

Les machines à mémoire partagée ont un espace d'adressage global, accessible par tous les processeurs (figure 20) [GGKMRS83]. Les communications de données sont assurées par la mémoire globale à travers un bus qui résout le problème de communication inter-processeur mais qui introduit à son tour le problème de l'accès simultané de la même adresse de la mémoire commune. Souvent, pour résoudre partiellement ce problème, une mémoire cache est associée à chaque processeur. L'accès simultané de plusieurs processeurs à autant de cases mémoire distinctes est possible en une unité de temps. Il y a des règles pour spécifier l'accès concurrent à la même case :

- **CREW** (Concurrent Read Exclusive Write) Ce modèle permet un nombre de lectures simultanées dans une même case; l'accès en écriture n'est possible que par un seul processeur à la fois.

- **CRCW** (Concurrent Read Concurrent Write) Le nombre d'écriture dans une même case mémoire n'est pas borné. Pour déterminer le résultat d'une écriture simultanée on a les modes suivants :

- Mode consistant : tous les processeurs écrivent la même valeur.
- Mode arbitraire : la valeur du dernier processeur qui est écrite.
- Mode fusion : une fonction commutative et associative est appliquée aux différentes valeurs écrites.

C'est le mode le plus puissant.

- **EREW** (Exclusive Read Exclusive Write) C'est le modèle le plus restrictif, et le plus proche des machines réelles.

Les calculateurs MIMD à mémoire commune sont appelés multiprocesseurs ou machines fortement couplées. Par exemple ENCORE, MULTIMAX, SEQUENT et BALANCE. Ce type de calculateur n'est pas massivement "scalable". Le nombre de processeurs ne dépasse pas quelques dizaines.

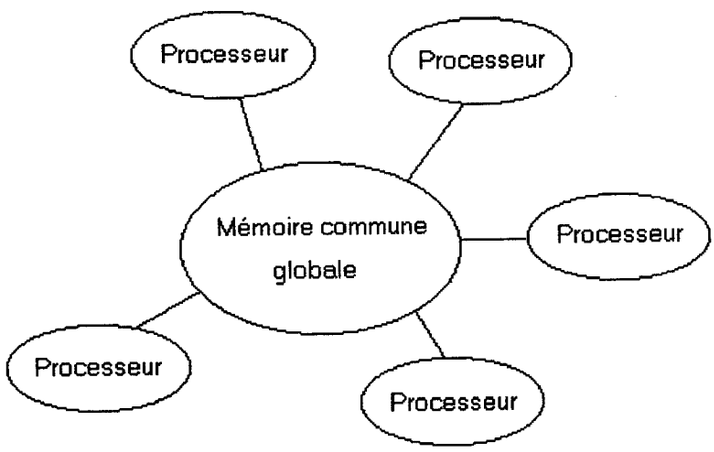



**Figure 20 :** *Modèle mémoire commune globale.*

Dans le cas des machines à mémoire distribuées (figure 21) chaque processeur a sa propre mémoire locale. Les processeurs appelés les "nœuds" sont connectés de telle façon à permettre uniquement la communication de données par le réseau d'interconnexion en utilisant un protocole de communication de type "passage de message". Selon le type de réseau utilisé, on peut avoir des performances différentes. Les calculateurs MIMD avec un réseau d'interconnexion sont appelés "multiordinateurs" ou machines faiblement couplées. Ces machines ont l'avantage d'être fortement "scalables". Les machines INTEL iPSC [Par91],

CUBE/7, IBM SP1 et SP2 [SP193] et réseaux de Transputers sont des exemples de ce type d'architecture.

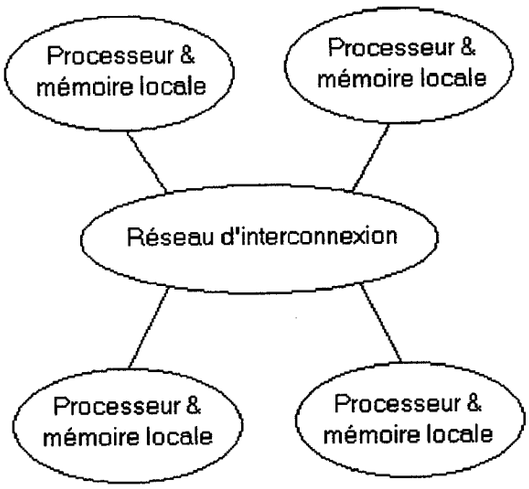



**Figure 21 :** *Mémoire distribuée et réseau d'interconnexion.*

**Mode SPMD**

Le mode SPMD est une version asynchrone associée au mode synchrone SIMD où le même programme est exécuté sur les processeurs d'un calculateur MIMD. Le mode SPMD n'est pas un paradigme hardware, c'est l'équivalent software du SIMD.

## 4.3 Réseaux d'interconnexion

Les processeurs d'une machine à mémoire distribuée sont reliés entre eux par l'intermédiaire d'un réseau d'interconnexion de topologie statique ou dynamique [CS99].

### 4.3.1 Topologies statiques

Le réseau d'interconnexion des topologies statiques est fixé par le constructeur et ne peut être modifié. Les topologies statiques les plus classiques sont : le réseau complet, l'anneau, la grille, le tore et l'hypercube. Les topologies statiques peuvent être décrites en terme de graphe où les sommets sont les processeurs et les arêtes sont les liens de communication. Des exemples de graphes de ces différentes topologies sont schématisés par les figures (22, 23, 24, 25, 26).

- **Le réseau complet ( Fully connected or all-to-all)**

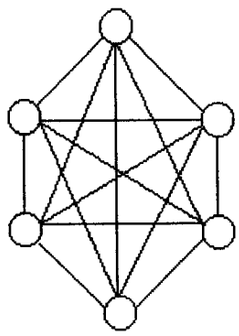


**Figure 22 :** *Réseau complet à 6 noeuds.*

- **La grille ( Mesh)**

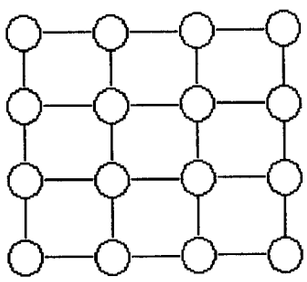


**Figure 23 :** *Grille 2D (4x4) nœuds.*

- **Le Tore ( Torus )**

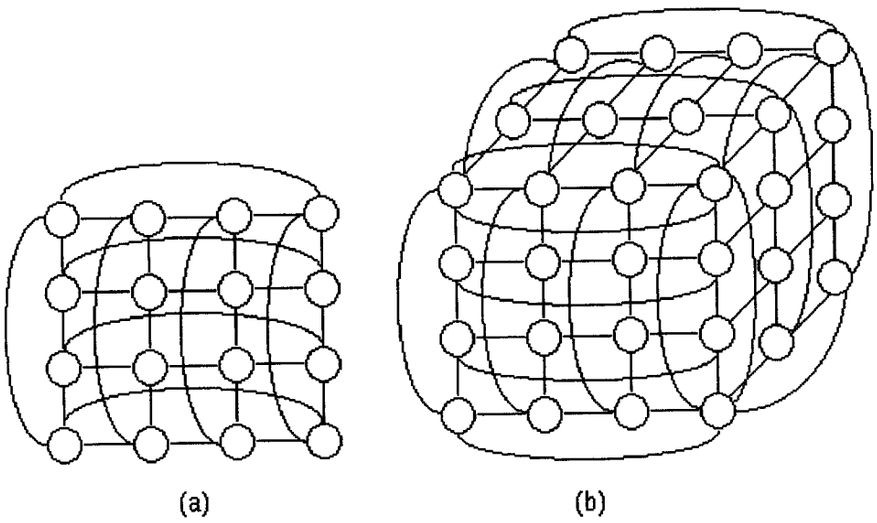



**Figure 24 :** *(a) Tore 2D (4x4) nœuds et (b) Tore 3D (4x4x3) nœuds.*

- **Anneau ( Ring )**

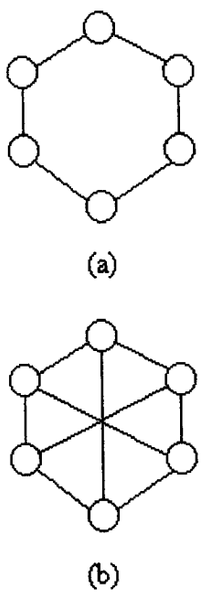



**Figure 25 :** *Anneau à 6 nœuds.*

- **L'hypercube ( Binary n-Cube )**

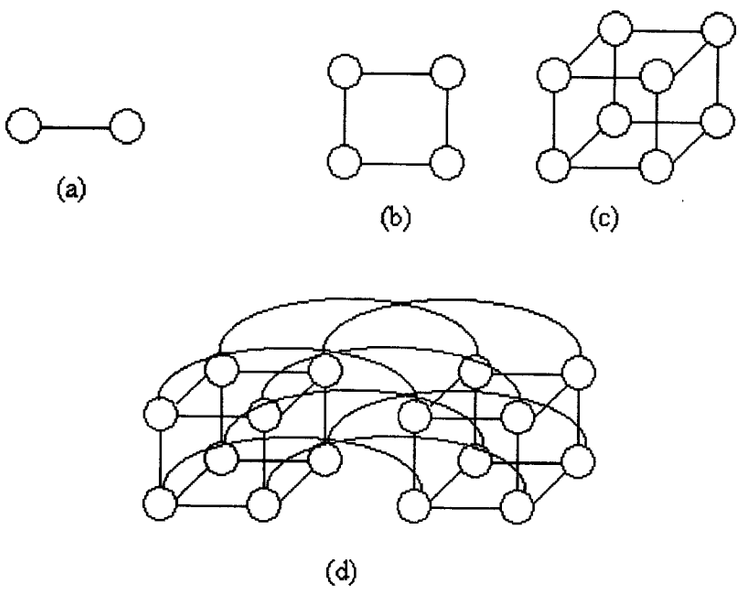



**Figure 26 :** *Hypercubes : (a) dimension 1, (b) dimension 2, (c) dimension 3 et (d) dimension 4.*

La rapidité des machines parallèles est un facteur de performance qui dépend du réseau d'interconnexion dont les caractéristiques sont données par :

- Nombre de nœuds p ou nombre de processeurs de la machine parallèle,
- Le diamètre D du réseau qui définit la plus grande distance entre deux nœuds,
- La connectivité ou le degré k du réseau liée au nombre de liens de communication de chaque processeur aux autres processeurs. Plus le nombre de liens est important, plus les communications peuvent être effectuées en parallèle,
- Le nombre de liens $N_l$ définit le nombre de liens total de la topologie,
- La largeur de bissection $L_B$ représente le minimum du nombre de liens nécessaires pour relier deux moitiés égales d'une topologie entre elles.

Le tableau 6 représente les caractéristiques des topologies les plus usuelles.

| Topologie | Nbre de proc. $P$ | Degré $K$ | Diamètre $D$ | Nbre de liens $N_l$ | Larg. Bisec. $L_B$ |
|---|---|---|---|---|---|
| Réseau complet | $P$ | $P-1$ | 1 | $p(p-1)/2$ | $(p/2)^2$ |
| Anneau | $P$ | 2 | $\lfloor p/2 \rfloor$ | $P$ | 2 |
| Grille 2D | $\sqrt{p}\ \sqrt{p}$ | $2 \rightarrow 4$ | $2(\sqrt{p}-1)$ | $2p-2\sqrt{p}$ | $\sqrt{p}$ |
| Tore 2D | $\sqrt{p}\ \sqrt{p}$ | 4 | $2\lfloor \sqrt{p}/2 \rfloor$ | $2p$ | $2\sqrt{p}$ |
| Hypercube | $p=2^d$ | $d=\log(p)$ | $d$ | $p \log(p)/2$ | $p/2$ |

**Tableau 6.** Caractéristiques usuelles des topologies statiques.

### 4.3.2 Topologies dynamiques

Les liens de communication des topologies dynamiques peuvent être modifiés en cours d'exécution. La figure 27 représente un réseau de Benes pour $p = 8$. Pour connecter $p$ entrées à $p$ sorties, ce réseau utilise $(2\log_2 p - 1)$ étages composés chacun de $p/2$ crossbars de taille 2 x 2 [CS99].

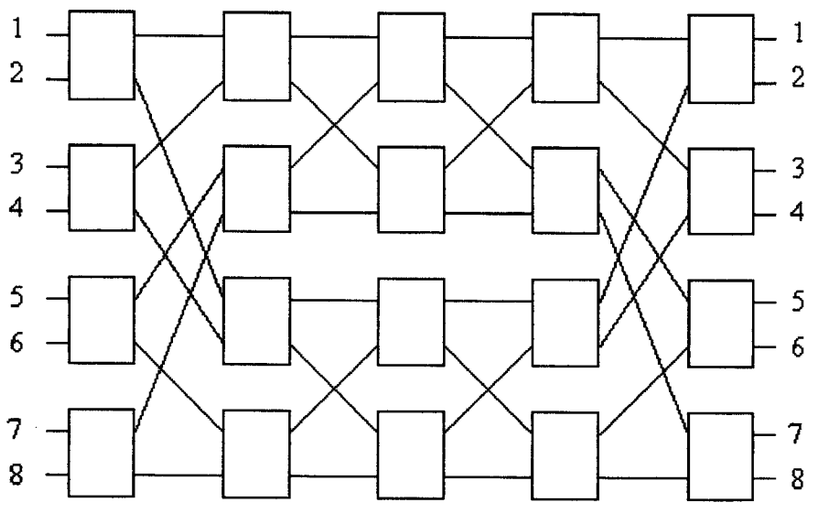



**Figure 27 :** *Réseau de Benes pour p = 8.*

## 4.4 Communications et routage

Les communications entre deux processeurs sont effectuées par un lien. Elles peuvent être unidirectionnelles (mode "half-duplex") ou bidirectionnelles (mode "full-duplex"). Pour communiquer un message, plusieurs techniques utilisent les modèles suivants [Lau96] :

- Modèle "Store-and-Forward" (SF)

  Chaque processeur intermédiaire reçoit et stocke le message avant de le ré-émettre au processeur de réception. Ce mode est également appelé modèle à commutation de message. Les processeurs intermédiaires sont interrompus de leurs tâches au moment des communications. Ce modèle est utilisé sur la première génération des machines parallèles sans co-processeur de communication. Si le message est de taille L, d la distance entre deux processeurs, $\beta$ le temps d'initialisation (ou startup time) et $\tau$ le temps de propagation d'un octet, le coût de cette modélisation est donné par d ($\beta$ + L $\tau$). La technique du pipeline permet de réduire ce coût de communication à :

$$(\sqrt{(d-1)\beta}+\sqrt{L\tau})^2 = L\tau + o(\sqrt{L})$$

- Modèle Cut Through (CT)

  Ce mode de communication utilise un co-processeur de communication et crée un chemin entre le processeur d'émission et celui de la réception. Dans ce modèle on peut distinguer deux protocoles :

  - Circuit-switching (CC)

    Un chemin est créé et le message est acheminé directement entre la source et la destination. Les processeurs intermédiaires sur le chemin sont bloqués pendant toute la période du transfert (immobilisation des liens).

  - Wormhole (WH)

    L'adresse du destinateur est placée dans l'en-tête du message et le routage s'effectue sur chaque processeur intermédiaire. Le message est découpé en flits qui seront stockés dans les registres internes des routeurs des processeurs intermédiaires dans le cas d'un blocage.

## 4.5 Modèles et environnement de programmation

Il y a beaucoup de méthodes de programmation des calculateurs parallèles. Deux modèles des plus communs sont : le modèle données parallèles "data parallel", et le modèle passage de messages "message passing".

### 4.5.1 Modèle données parallèles

Chaque processus fonctionne sur une partie différente de la même structure de données en utilisant uniquement l'approche du mode SPMD. Les données sont distribuées à travers les processeurs par les directives de construction du langage. Le compilateur de ce modèle de programmation convertit le programme de l'utilisateur en un code standard et fait appel à la bibliothèque de passage de message pour distribuer les données à tous les processus. Les communications inter-processeurs et la synchronisation sont invisibles par le programmeur. La programmation est simple; les compilateurs sont très complexes et très difficiles à développer. Dans plusieurs situations les codes optimaux ne sont pas obtenus. Ce modèle de programmation est automatiquement scalable. Le seul langage standard et portable est le HPF (hight performance fortran) [HPF93] qui est basé sur le fortran 90.

### 4.5.2 Modèle passage de messages

Le modèle passage de messages est défini comme un ensemble de processus utilisant la mémoire locale et communiquant par l'envoi et la réception de messages. Les opérations coopératives (send-receive) sont accomplies par chaque processus pour le transfert de données. La programmation à l'aide de passage de messages est faite par linkage et par branchement à des bibliothèques qui permettent l'échange de données entre les processeurs. Les bibliothèques à passage de messages sont disponibles pour tous les langages modernes de programmation. Les bibliothèques les plus utilisées sont : PVM "Parallel Virtual Machine" [GBDWMS94], MPI "Message Passing Interface" [MPI93] et MPL "Message Passing Library". PVM est un environnement de programmation, développé à l'origine en 1989 par des chercheurs de Oak Ridge National Laboratory [BDJMS94]. Cette bibliothèque permet de considérer un ensemble de calculateurs différents comme une simple machine parallèle. Ce système est défini par deux composants principaux : le processus "démon" PVM et la bibliothèque de routines d'interface. Le processus démon permet la gestion des applications hétérogènes et la bibliothèque de routines d'interface permet le contrôle et les communications par passage de messages. MPI est une bibliothèque à passage de messages standard et



portable, utilisée par les calculateurs parallèles à mémoire distribuée. Elle a été développée en 1993 par un groupe de vendeurs de calculateurs parallèles, de programmeurs de logiciels et de scientifiques. Elle est utilisée dans des programmes en fortran ou en C. Le programmeur est responsable du parallélisme des programmes et de l'implémentation des opérations du MPI. On a un parallélisme explicite. MPL est un produit de propriété IBM permettant un ensemble simple d'opérations de coordination et de communication entre processeurs via des routines de passage de messages pour des applications parallèles utilisant des machines à mémoire distribuée.

## 4.6 Environnement de programmation MPI

Une application dans un environnement MPI [GLS94] est définie par un ensemble de processus autonomes exécutant chacun leur propre code et communiquant via des appels à des sous-programmes de la bibliothèque MPI [Nels00]. Le système MPI est complexe et contient 129 fonctions. Ces fonctions peuvent être classées dans les catégories suivantes : environnement, communications point à point, communications collectives, types de données dérivés, topologies.

### 4.6.1 Environnement

L'environnement MPI est défini par des communicateurs. Le communicateur est par défaut MPI_COMM _WORLD ; il est défini par un ensemble de processus actifs pouvant effectuer des opérations de communications. Les fonctions qui permettent de définir l'environnement MPI sont:

- MPI_INIT( )
  Est un sous programme d'initialisation à utiliser avant tout appel à une fonction MPI.

  Paramètre:
  [OUT IERROR] (entier)

  Syntaxe en Fortran:

  ```
  MPI_INIT(IERROR)

  INTEGER IERROR
  ```

- MPI_COMM_SIZE( )

  Ce sous programme permet de connaître le nombre de processus gérés par le communicateur.

  Paramètres:
  [IN COMM] communicateur
  [OUT SIZE] nombre de processus dans le groupe du comm (entier)
  [OUT IERROR] (entier)

  Syntaxe en Fortran:

  ```
  MPI_COMM_SIZE(COMM, SIZE, IERROR)

  INTEGER COMM, SIZE, IERROR
  ```

- MPI_COMM_RANK( )

  Ce sous programme permet d'obtenir le rang d'un processus du communicateur.

  Paramètres:
  [IN COMM] communicateur
  [OUT RANK] nombre de processus appelés dans le groupe du comm (entier)
  [OUT IERROR] (entier)

  Syntaxe en Fortran:

  ```
  MPI_COMM_RANK(COMM, SIZE, IERROR)

  INTEGER COMM, RANK, IERROR
  ```

- MPI_FINALIZE( )

  Est un sous programme de finalisation qui permet de désactiver l'environnement MPI.

  Paramètre:
  [OUT IERROR] (entier)

Syntaxe en Fortran:

```
MPI_FINALIZE(IERROR)

INTEGER IERROR
```

### 4.6.2 Communications point à point

Ce sont des communications entre deux processus, l'un appelé émetteur et l'autre récepteur. Les fonctions de communication qu'on rencontre souvent dans plusieurs applications et que nous avons utilisé dans l'écriture de nos codes parallèles sont :

- MPI_SEND( )

  Ce sous programme permet d'envoyer un message d'un processus émetteur vers un processus cible.

  Paramètres :
  [IN BUF] adresse initiale du tampon d'envoi
  [IN COUNT] nombre d'éléments dans le tampon d'envoi (entier non négatif)
  [IN DATATYPE] type de données de chaque élément du tampon d'envoi
  [IN DEST] rang de la destination (entier)
  [IN TAG] message tag (entier)
  [IN COMM] communicateur
  [OUT IERROR]

  Syntaxe en Fortran:

  ```
  MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)

  <TYPE> BUF(*)

  INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR
  ```

- MPI_RECV( )

  Ce sous programme permet au processus destination de recevoir un message d'un processus émetteur.

Paramètres :

[OUT BUF] adresse initiale du tampon de réception

[IN COUNT] nombre d'éléments dans le tampon de réception (entier)

[IN DATATYPE] type de données de chaque élément du tampon de réception

[IN DEST] rang de la source ou MPI_ANY_SOURCE (entier)

[IN TAG] message tag ou MPI_ANY_TAG (entier)

[IN COMM] communicateur

[OUT STATUS] objet status

[OUT IERROR]

Syntaxe en Fortran:

```
MPI_RECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS,

IERROR)

<TYPE> BUF(*)

INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM

INTEGER STATUS (MPI_STATUS_SIZE), IERROR
```

- MPI_SENDRECV( )

  Ce sous programme permet en même temps d'envoyer et de recevoir le message. Ce type de communication correspond aux communications en anneau.

  Paramètres :

  [IN SENDBUF] adresse initiale du tampon d'envoi

  [IN SENDCOUNT] nombre d'éléments dans le tampon envoyé (entier)

  [IN SENDTYPE] type des éléments dans le tampon envoyé

  [IN DEST] rang de la destination (entier)

  [IN SENDTAG] message tag d'émission (entier)

  [OUT RECVBUF] adresse initiale du tampon de réception

  [IN RECVCOUNT] nombre d'éléments dans le tampon de réception (entier)

  [IN RECVTYPE] type des éléments dans le tampon de réception

  [IN SOURCE] rang de la source (entier)

[IN RECVTAG] message tag de réception (entier)

[IN COMM] communicateur

[OUT STATUS] objet status (Status)

[OUT IERROR]

Syntaxe en Fortran:

```
MPI_SENDRECV(SENDBUF, SENDCOUNT, SENDTYPE, DEST, SENDTAG,

RECVBUF, RECVCOUNT, RECVTYPE, SOURCE, RECVTAG, COMM,

STATUS, IERROR)

<type> SENDBUF(*), RECVBUF(*)

INTEGER SENDCOUNT, SENDTYPE, DEST, SENDTAG, RECVCOUNT,

RECVTYPE, SOURCE, RECV TAG, COMM, STATUS(MPI_STATUS_SIZE),

IERROR
```

### 4.6.3 Communications collectives

Les communications collectives permettent d'effectuer une série de communications point à point entre les processus du communicateur. Il y a trois types de fonctions : les fonctions de synchronisation, de transfert de données et celles permettant de réaliser des opérations sur les données transférées. Les sous programmes utilisés dans nos codes parallèles sont:

- MPI_BARRIER( )

  Est un sous programme de synchronisation. Il est souvent utilisé avant l'appel au sous programme de détermination du temps d'exécution des applications parallèles.

  Paramètres:

  [IN COMM] communicateur

  [OUT IERROR] (entier)

Syntaxe en Fortran:

```
MPI_BARRIER(COMM, IERROR)

INTEGER COMM, IERROR
```

- MPI_WTIME( )

  Ce sous programme permet de déterminer le temps d'exécution.

  Syntaxe en Fortran:

```
MPI_WTIME( )

DOUBLE PRECISION MPI_WTIME( )
```

- MPI_BCAST( )

  Ce sous programme permet de diffuser les données d'un processus émetteur à tous les processus du communicateur; Il est souvent utilisé pour diffuser les données d'entrée.

  Paramètres:

  [INOUT BUF] adresse de commencement du tampon

  [IN COUNT] nombre d'entrées dans le tampon (entier)

  [IN DATATYPE] type de données du tampon

  [IN ROOT] rang de la racine de distribution (entier)

  [IN COMM] communicateur

  [OUT IERROR] (entier)

  Syntaxe en Fortran:

```
MPI_BCAST(BUF, COUNT, DATATYPE, ROOT, COMM, IERROR)

<TYPE> BUF(*)

INTEGER COUNT, DATATYPE, ROOT, COMM, IERROR
```

### 4.6.4 Types de données dérivés

Dans les communications point à point, les données échangées sont typées: MPI_INTEGER, MPI_REAL, MPI_COMPLEX ou MPI_LOGICAL etc. On peut créer des structures de données plus complexes à l'aide des fonctions suivantes:

- MPI_TYPE_CONTIGUOUS( )

  Ce sous programme permet de créer une structure de données à partir d'un ensemble homogène de type prédéfini de données contiguës en mémoire.

  Paramètres :
  [IN COUNT] (entier non négatif)
  [IN OLDTYPE] type de donnée ancienne
  [OUT NEWTYPE] type de donnée nouvelle
  [OUT IERROR] (entier)

  Syntaxe en Fortran:

  ```
  MPI_TYPE_CONTIGUOUS(COUNT, OLDTYPE, NEWTYPE, IERROR)

  INTEGER COUNT, OLDTYPE, NEWTYPE, IERROR
  ```

- MPI_TYPE_VECTEUR( )

  Ce sous programme permet de créer une structure de données à partir d'un ensemble homogène de type prédéfini de données distantes d'un pas constant en mémoire. Le pas est donné en nombre d'éléments.

  Paramètres :
  [IN COUNT] nombre de bloque (entier non négatif)
  [IN BLOCKLENGTH] nombre d'éléments dans chaque bloque (entier non négatif)
  [IN STRIDE] nombre d'éléments entre le début de chaque bloque (entier)
  [IN OLDTYPE] type de donnée ancienne
  [OUT NEWTYPE] type de donnée nouvelle
  [OUT IERROR] (entier)

Syntaxe en Fortran:

```
MPI_TYPE_VECTEUR(COUNT, BLOCKLENGTH, STRIDE, OLDTYPE,
NEWTYPE, IERROR)
INTEGER COUNT, OLDTYPE, NEWTYPE, IERROR
```

- MPI_TYPE_COMMIT( )

  Ce sous programme permet de valider le type de données déjà créés.

  Paramètres :
  [INOUT DATATYPE] type de données commutées
  [OUT IERROR] (entier)

  Syntaxe en Fortran:

```
MPI_ TYPE_COMMIT(DATATYPE, IERROR)
INTEGER DATATYPE, IERROR
```

- MPI_TYPE_FREE( )

  Ce sous programme permet de libérer le type de données. Ce type peut être réutilisé.

  Paramètres :
  [INOUT DATATYPE] type de données libérées
  [OUT IERROR] (entier)

  Syntaxe en Fortran:

```
MPI_TYPE_FREE(DATATYPE, IERROR)
INTEGER DATATYPE, IERROR
```

Il existe dans la bibliothèque MPI d'autres fonctions qui permettent de créer des types de données homogènes à pas constant et à pas variable ainsi que des types de données hétérogènes.



### 4.6.5 Topologies

Les fonctions MPI permettent de définir des topologies virtuelles de type cartésien ou graphe. Ces topologies sont particulièrement utilisées dans les méthodes de décomposition de domaine. Dans la topologie du type cartésien, chaque processus du communicateur est identifié par ses coordonnées dans une grille de processus. Les fonctions permettant de définir cette topologie sont:

- MPI_CART_CREATE( )

  Ce sous programme permet de créer la topologie cartésienne qui peut être périodique ou non.

  Paramètres :

  [IN COMM_OLD] communicateur d'entrée

  [IN NDIMS] nombre de dimensions de la grille cartésienne (entier)

  [IN DIMS] tableau de dimension NDIMS donnant le nombre de processus dans chaque dimension (entier)

  [IN PERIODS] tableau logique de dimension NDIMS spécifiant si la grille est périodique (TRUE) ou non (FALSE) dans chaque dimension

  [IN REORDER] rangement peut être réordonné (TRUE) ou non (FALSE) (logique)

  [OUT COMM_CART] communicateur avec la nouvelle topologie cartésienne

  [OUT IERROR] (entier)

  Syntaxe en Fortran:

  ```
  MPI_CART_CREATE(COMM_OLD, NDIMS, DIMS, PERIODS, REORDER,

  COMM_CART, IERROR)

  INTEGER COMM_OLD, NDIMS, DIMS(*), COMM_CART, IERROR

  LOGICAL PERIODS(*), REORDER
  ```

- MPI_CART_RANK( )

  Cette fonction permet de donner le rang du processus associé aux coordonnées dans la grille dans une topologie cartésienne.

Paramètres :

[IN COMM] communicateur avec une structure cartésienne

[IN COORDS] tableau de dimension NDIMS spécifiant les coordonnées cartésiennes du processus (entier)

[OUT RANK] rang du processus spécifié (entier)

[OUT IERROR] (entier)

Syntaxe en Fortran:

```
MPI_CART_RANK(COMM, COORDS, RANK, IERROR)

INTEGER COMM, COORDS(*), RANK, IERROR
```

- MPI_CART_SHIFT( )

  Dans une topologie cartésienne, cette fonction permet à un processus de connaître le rang de ses voisins dans une direction donnée.

  Paramètres :

  [IN COMM] communicateur avec une structure cartésienne

  [IN DIRECTION] coordonnée de dimension du shift (entier)

  [IN DISP] déplacement (>0 shit ascendant, <0 shift descendant (entier)

  [OUT RANK_SOURCE] rang du processus source (entier)

  [OUT RANK_DEST] rang du processus destination (entier)

  [OUT IERROR] (entier)

  Syntaxe en Fortran:

```
MPI_CART_SHIFT(COMM, DIRECTION, DISP, RANK_SOURCE,

RANK_DEST, IERROR)

INTEGER COMM, DIRECTION, DISP, RANK_SOURCE, RANK_DEST,

IERROR
```

- MPI_DIMS_CREATE( )

  Dans une topologie cartésienne, cette fonction retourne le nombre de processus dans chaque dimension de la grille en fonction du nombre total de processus.

  Paramètres :
  [IN NNODES] nombre de nœuds dans une grille (entier)
  [IN NDIMS] nombre de dimensions cartésienne (entier)
  [INOUT DIMS] tableau de dimensions NDIMS spécifiant le nombre des nœuds dans chaque dimension (entier)
  [OUT IERROR] (entier)

  Syntaxe en Fortran :

  ```
  MPI_DIMS_CREATE(NNODES, NDIMS, DIMS, IERROR)

  INTEGER NNODES, NDIMS, DIMS(*), IERROR
  ```

## 4.7 Bibliothèques de calcul

Parallèlement aux développements des bibliothèques de communication, des bibliothèques de calcul ont été programmées sur un bon nombre de machines. L'objectif de ces bibliothèques est de proposer à l'utilisateur un certain nombre de primitives d'algèbre linéaire parallèle. La première bibliothèque de calcul, LINPACK [DMBS88], contient des programmes de factorisation matricielle : LU, Cholesky, QR et SVD. Cette bibliothèque a été écrite en Fortran. Elle a introduit la programmation modulaire et utilise les notions de BLAS "Basic Linear Algebra Subroutine". Les BLAS sont définies par trois niveaux :

- BLAS 1 : pour les opérations vecteur-vecteur,
- BLAS 2 : pour les opérations matrice-vecteur,
- BLAS 3 : pour les opérations matrice-matrice.

La bibliothèque EISPACK utilise aussi les BLAS et contient des routines pour les problèmes liés aux valeurs propres et aux vecteurs propres. La bibliothèque la plus répandue actuellement est la bibliothèque ScaLAPACK [CDPW92]. Cette dernière utilise les BLACS qui sont la réécriture des routines BLAS sur des machines MIMD.

## 4.8 Evaluation des performances

Les performances sont définies par l'accélération, l'efficacité et par les facteurs qui les limitent.

### 4.8.1 Accélération et efficacité

L'accélération et l'efficacité sont des mesures importantes de la qualité des algorithmes parallèles.

Si $T_S$ est le meilleur temps d'exécution de l'algorithme parallèle sur un seul processeur, et si $T_P$ est celui de l'algorithme parallèle sur N processeurs. L'accélération $S_N$ et l'efficacité $E_N$ sont définies par les équations suivantes:

$$S_N = \frac{T_S}{T_P} \qquad (4.48)$$

$$E_N = \frac{S_N}{N} \qquad (4.49)$$

En pratique, l'accélération est réduite de sa valeur idéale N et l'efficacité est limitée par 1.

### 4.8.2 Loi d'Amdahl

L'accélération est limitée par la composante séquentielle de l'algorithme implémenté sur les ordinateurs parallèles[Gus88].

Si S représente le temps de la part de calcul séquentielle de l'algorithme par un processeur, et si P est le temps de calcul par un processeur de la partie de l'algorithme à paralléliser, alors :

$$T_{seq} = S + P \text{ et } T_{par} = S + \frac{P}{N}$$

où N est le nombre de processeurs. L'accélération $S_N$ peut s'écrire sous la forme :

$$S_N = \frac{S+P}{S+\frac{P}{N}}$$

Si F est la part de calcul séquentielle ( $F = S / T_{seq}$ ), l'accélération $S_N$ peut être exprimée en fonction de F par l'équation suivante :

$$S_N = \frac{1}{F + \frac{1-F}{N}} \qquad (4.50)$$

Les communications inter-processeurs limitent le facteur d'accélération. Les parallélismes à grains fins et à gros grains permettent de réduire leurs influences sur les performances des algorithmes parallèles.

# Chapitre 5

# Calcul parallèle des filtres de ré-échantillonnage d'images

## 5.1 Calcul parallèle de la bilinéaire

Si m est le nombre de pixels de sortie suivant x et y de l'image d'entrée, la part de calcul induite par le calcul des coefficients du filtre des 4 pixels est très faible et est déterminée par le rapport :

$$R1(m) = \frac{4m^2 + 4m + 6}{(7m^2 + 6m)n^2} = O(\frac{1}{n^2}) \qquad (5.51)$$

La figure 28 représente la variation de la part de calcul expérimentale et théorique des coefficients du filtre des 4 pixels en fonction de la dimension n de l'image d'entrée pour m = 1.

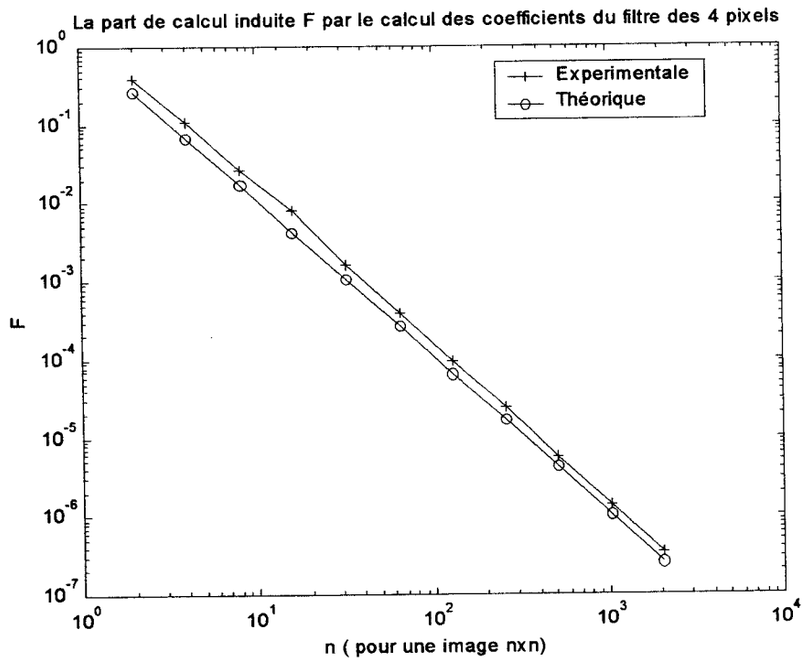
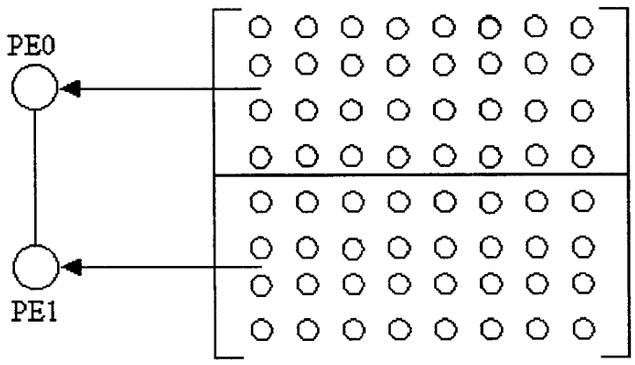



**Figure 28 :** *La part de calcul expérimentale et théorique du calcul des coefficients du filtre des 4 pixels.*

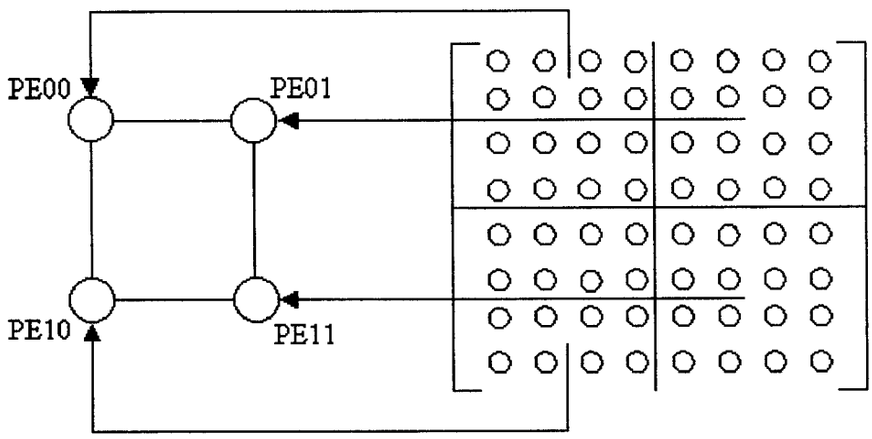
### 5.1.1 Comparaison des temps de calcul

Nous considérons maintenant un réseau de $n_p$ processeurs communicant par messages. Les valeurs de l'image d'entrée G sont distribuées aux $n_p$ processeurs à raison d'un bloc de $(n / n_p)$ lignes par processeurs dans le cas d'un réseau linéaire (figure 29a) et d'un bloc de $(n / n_p \times n / n_p)$ dans le cas d'une grille de processeurs (figure 29b). Chaque processeur

calcule les coefficients du filtre et en même temps transmet à ses voisins la ligne frontière de sa partie de l'image d'entrée G (figure 30). Enfin, chaque processeur applique le filtre sur sa partie de l'image d'entrée G.

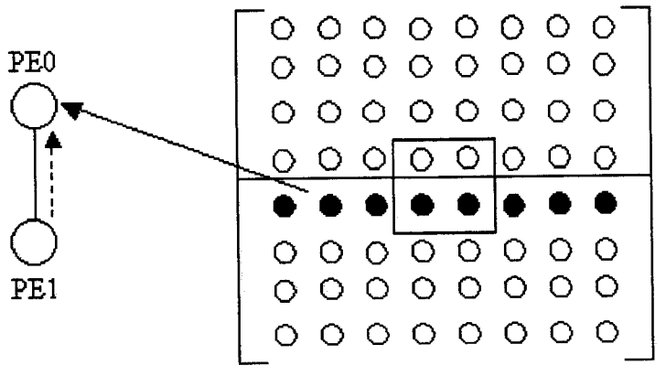



(a)

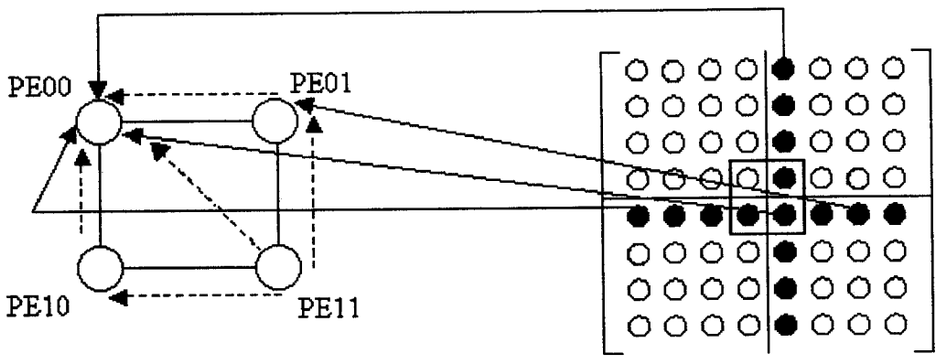



(b)

**Figure 29 :** *Distribution de l'image d'entrée G : (a) à deux processeurs d'un réseau linéaire, (b) à une grille à quatre processeurs.*




(a)




(b)

**Figure 30 :** *Communications inter-processeurs du filtre des 4 pixels : (a) cas d'un réseau linéaire à deux processeurs, (b) cas de la grille à quatre processeurs.*

Pour réaliser les tests, le programme décrit a été porté sur une grappe de 8 nœuds bi-processeurs (processeurs Pentium II fonctionnant à 450 Mhz) couplés par un réseau de 100Mbits/s. La bibliothèque de communication utilisée est la bibliothèque Message Passing Interface (MPI) avec le mode SPMD (Single Program Multiple Data).

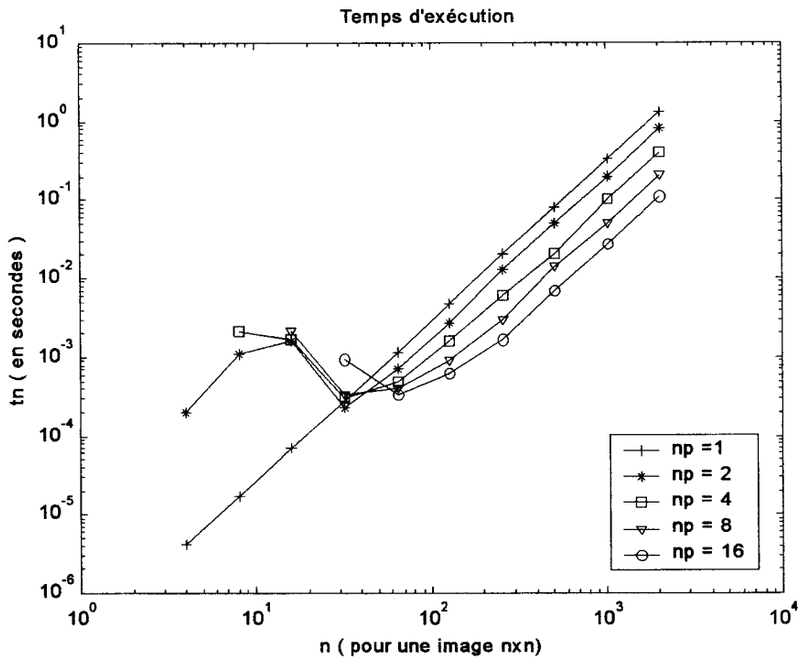
L'approche parallèle du filtre des 4 pixels nous a permis de réduire les temps de calcul (voir les tableaux 7 et 8). La figure 31 illustre respectivement l'évolution des temps d'exécution sur un réseau linéaire et sur une grille de processeurs.

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Filtre 4/1, 1 proc. | 1.7e-5 | 0.7e-4 | 2.8e-4 | 1.2e-3 | 4.7e-3 | 2.0e-2 | 0.82e-1 | 0.33 | 1.33 |
| Filtre 4/1, 2 proc. | 1.1e-3 | 1.6e-3 | 0.2e-3 | 0.7e-3 | 0.3e-2 | 1.3e-2 | 0.05 | 0.2 | **0.8** |
| Filtre 4/1, 4 proc. | 2.1e-3 | 1.7e-3 | 0.3e-3 | 0.5e-3 | 1.6e-3 | 0.6e-2 | 0.02 | 0.1 | 0.4 |
| Filtre 4/1, 8 proc. | | 2.1e-3 | 0.3e-3 | 0.4e-3 | 0.1e-2 | 0.3e-2 | 0.014 | 0.05 | 0.21 |
| Filtre 4/1, 16 proc. | | | 1.0e-3 | 0.3e-3 | 0.6e-3 | 1.7e-3 | 0.007 | 0.027 | 0.11 |

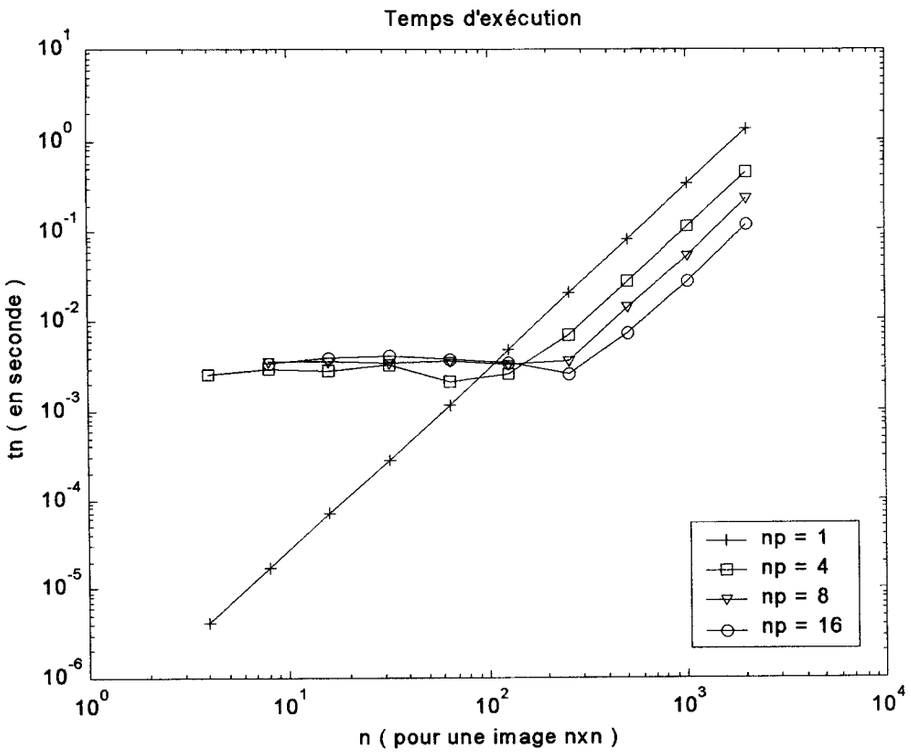
**Tableau 7.** *Temps de calcul (s) du filtre 4/1 pixel pour np = 1, 2, 4, 8 et 16 processeurs (réseau linéaire).*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Filtre 4/1, 1 proc. | 1.7e-5 | 0.7e-4 | 2.8e-4 | 1.2e-3 | 4.7e-3 | 2.0e-2 | 0.82e-1 | 0.33 | 1.33 |
| Filtre 4/1, 4 proc. | 3.0e-3 | 2.8e-3 | 3.3e-3 | 2.1e-3 | 2.6e-3 | 6.8e-3 | 2.7e-2 | 0.1 | **0.45** |
| Filtre 4/1, 8 proc. | 3.5e-3 | 3.5e-3 | 3.3e-3 | 3.6e-3 | 3.3e-3 | 3.6e-3 | 0.14e-1 | 0.054 | **0.225** |
| Filtre 4/1, 16 proc. | 3.5e-3 | 4.0e-3 | 4.1e-3 | 3.7e-3 | 3.4e-3 | 2.6e-3 | 7.4e-3 | **0.028** | **0.116** |

**Tableau 8.** *Temps de calcul (s) du filtre 4/1 pixel pour np = 1, 4, 8 et 16 processeurs (grille 2D).*




(a)




(b)

**Figure 31 :** *Temps de calcul du filtre 4/1 pixel pour : (a) np = 1, 2, 4, 8 et 16 processeurs d'un réseau linéaire , (b) np = 1, 4, 8 et 16 processeurs d'une grille .*

### 5.1.2 Accélération et efficacité

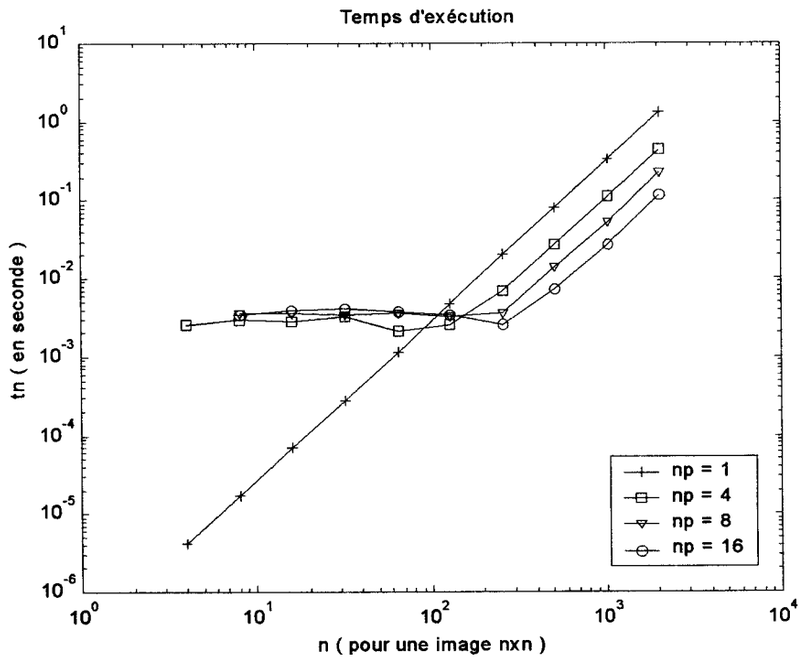
Les figures 32 et 33 représentent les accélérations et efficacités résultantes. Les courbes de l'accélération S et de l'efficacité E permettent de montrer qu'on commence à obtenir de bonnes accélérations et efficacités pour des images d'entrée de dimensions n supérieur ou égal à 64 (voir tableaux 9 et 10).

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Accélération, 2 proc. | 0.015 | 0.044 | 1.22 | 1.67 | 1.74 | 1.58 | 1.64 | 1.65 | 1.66 |
| Accélération, 4 proc. | 0.008 | 0.041 | 0.93 | 2.34 | 2.94 | 3.38 | 4.1 | 3.3 | 3.325 |
| Accélération, 8 proc. | --- | 0.033 | 0.85 | 2.85 | 5.22 | 6.76 | 5.94 | 6.6 | 6.33 |
| Accélération, 16 proc. | --- | --- | 0.3 | 3.44 | 7.58 | 11.94 | 11.71 | 12.22 | 12.1 |

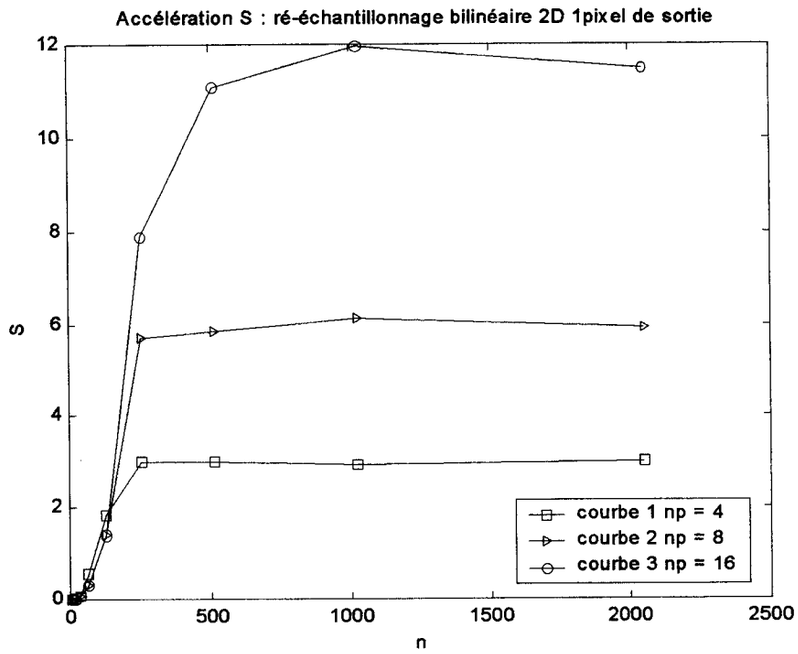
**Tableau 9.** *Accélération du filtre des 4 pixels pour np =2, 4, 8 et 16 processeurs (réseau linéaire).*

| | n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|---|
| Accélération, 4 proc. | | 0.006 | 0.025 | 0.085 | 0.55 | 1.829 | 3.0 | 3.0 | 2.9 | 3.0 |
| Accélération, 8 proc. | | 0.005 | 0.020 | 0.084 | 0.32 | 1.420 | 5.7 | 5.8 | 6.0 | 5.9 |
| Accélération, 16 proc. | | 0.005 | 0.017 | 0.068 | 0.32 | 1.400 | 7.9 | 11.1 | 12.0 | 11.5 |

**Tableau 10.** *Accélération du filtre des 4 pixels pour np = 4, 8 et 16 processeurs (cas d'une grille).*




(a)




(b)

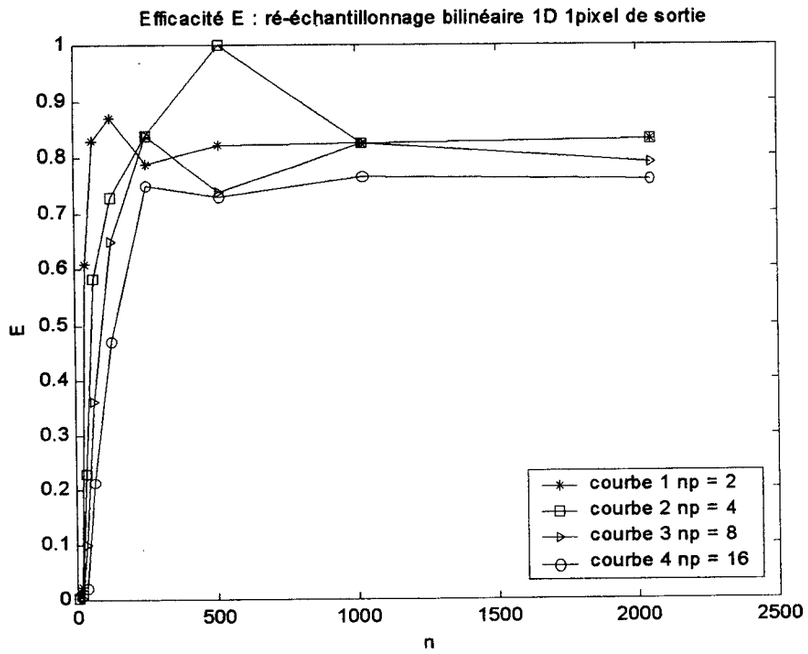
**Figure 32 :** *Accélération S : (a) réseau linéaire pour np = 2, 4, 8 et 16 processeurs, (b) grille de processeurs pour np = 4, 8 et 16 processeurs.*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Efficacité, 2 proc. | 0.008 | 0.022 | 0.61 | 0.83 | 0.87 | 0.79 | 0.82 | 0.83 | 0.83 |
| Efficacité, 4 proc. | 0.002 | 0.01 | 0.23 | 0.59 | 0.73 | 0.84 | 1 | 0.83 | 0.83 |
| Efficacité, 8 proc. | | 0.004 | 0.1 | 0.36 | 0.65 | 0.84 | 0.74 | 0.83 | 0.79 |
| Efficacité, 16 proc. | | | 0.02 | 0.21 | 0.47 | 0.75 | 0.73 | 0.76 | 0.76 |

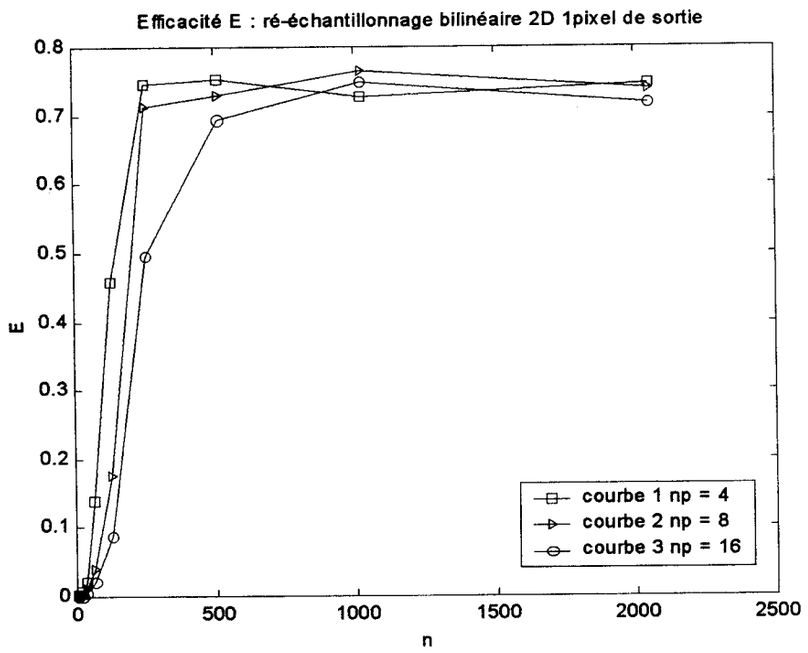
**Tableau 11.** *Efficacité pour np = 2, 4, 8 et 16 processeurs (réseau linéaire).*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Efficacité, 4 proc. | 1.4e-3 | 6.3e-3 | 2.1e-2 | 0.14 | 0.46 | 0.75 | 0.75 | 0.73 | 0.75 |
| Efficacité, 8 proc. | 0.6e-3 | 2.5e-3 | 1.1e-2 | 0.04 | 0.18 | 0.71 | 0.73 | 0.765 | 0.74 |
| Efficacité, 16 proc. | 0.3e-3 | 1.1e-3 | 0.42e-2 | 0.02 | 0.09 | 0.50 | 0.70 | 0.749 | 0.72 |

**Tableau 12.** *Efficacité pour np = 4, 8 et 16 processeurs ( cas d'une grille).*




(a)




(b)

**Figure 33 :** *Efficacité E : (a) réseau linéaire pour np = 2, 4, 8 et 16 processeurs, (b) grille de processeurs pour np = 4, 8 et 16 processeurs.*

## 5.2 Calcul parallèle de la B-spline cubique uniforme

La part de calcul induite par le calcul des coefficients du filtre des 16 pixels est déterminée par le rapport :

$$R2(m) = \frac{16m^2 + 48m + 2}{(31m^2 + 46m)n^2} = O(\frac{1}{n^2}) \qquad (5.52)$$

On constate bien que cette part est très faible. La figure 34 représente la variation de la part de calcul expérimentale et théorique en fonction de la dimension n de l'image d'entrée pour m = 1.

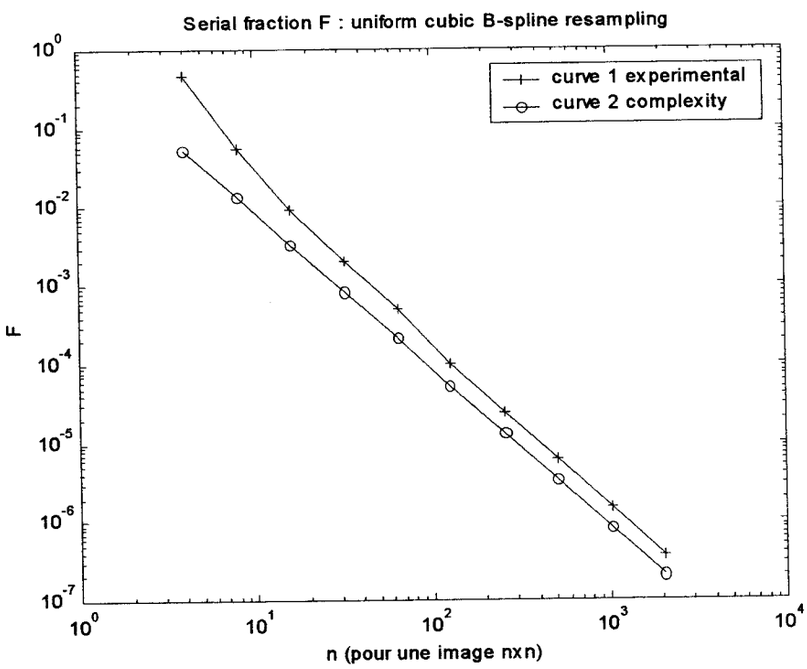



**Figure 34 :** *La part de calcul expérimentale et théorique du calcul des coefficients du filtre des 16 pixels.*

### 5.2.1 Calcul parallèle du filtre des 16 pixels

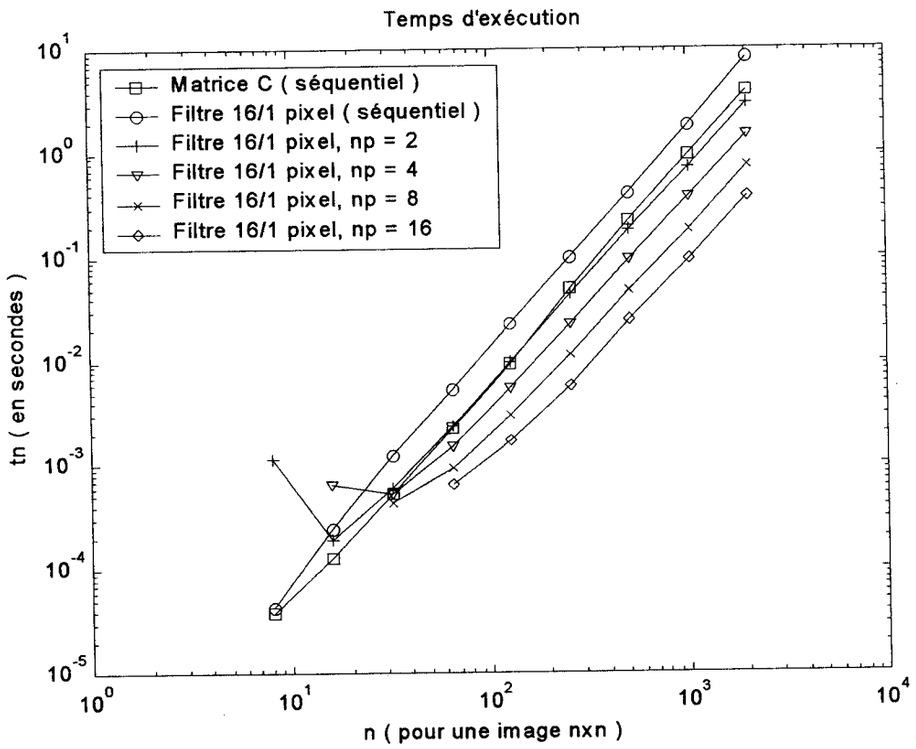
Nous considérons maintenant un réseau de $n_p$ processeurs communicant par messages. Dans une première étape, la matrice C est calculée sur un processeur. Celle-ci est ensuite distribuée aux $n_p$ processeurs à raison d'un bloc de $(n / n_p)$ lignes par processeurs dans le cas d'un réseau linéaire (figure 29a) et d'un bloc de $(n / n_p \times n / n_p)$ dans le cas d'une grille de

| N | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Matrice 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 |
| Filtre 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 |
| Filtre 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 |
| Filtre 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 |
| Filtre 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 |
| Filtre 16/1, 16 proc. | --- | --- | --- | 6.6e-4 | 1.7e-3 | 5.9e-3 | 2.5e-2 | 9.5e-2 | 3.8e-1 |

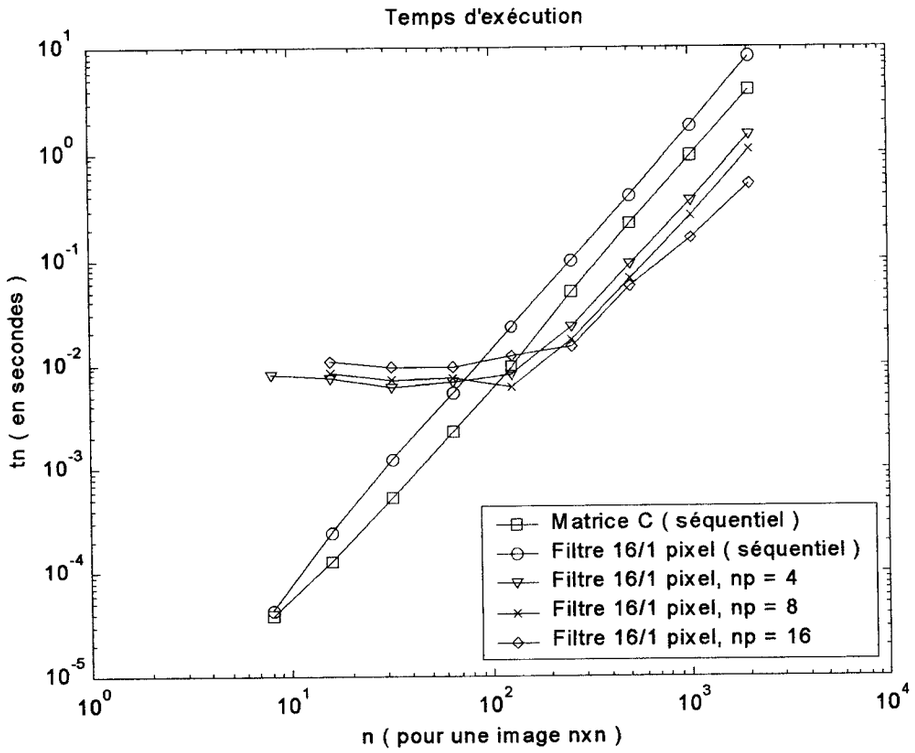
**Tableau 13.** *Temps de calcul (s) de la matrice C en séquentiel et du filtre 16/1 pixel pour np = 1, 2, 4, 8 et 16 processeurs (cas d'un réseau linéaire).*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Matrice 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 |
| Filtre 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 |
| Filtre 16/1, 4 proc. | 7.9e-3 | 7.3e-3 | 6.1e-3 | 6.9e-3 | 7.9e-3 | 2.3e-2 | 9.0e-2 | 3.6e-1 | 1.5e0 |
| Filtre 16/1, 8 proc. | --- | 8.2e-3 | 7.0e-3 | 7.4e-3 | 6.0e-3 | 1.7e-2 | 6.5e-2 | 2.6e-1 | 1.0e0 |
| Filtre 16/1, 16 proc. | --- | 1.0e-2 | 9.5e-3 | 9.5e-3 | 1.2e-2 | 1.5e-2 | 5.5e-2 | 1.6e-1 | 0.53 |

**Tableau 14.** *Temps de calcul (s) de la matrice C en séquentiel et du filtre 16/1 pixel pour np = 1, 4, 8 et 16 processeurs (cas d'une grille).*


Temps d'exécution
Matrice C ( séquentiel )
Filtre 16/1 pixel ( séquentiel )
Filtre 16/1 pixel, np = 2
Filtre 16/1 pixel, np = 4
Filtre 16/1 pixel, np = 8
Filtre 16/1 pixel, np = 16
tn ( en secondes )
n ( pour une image nxn )

(a)




(b)

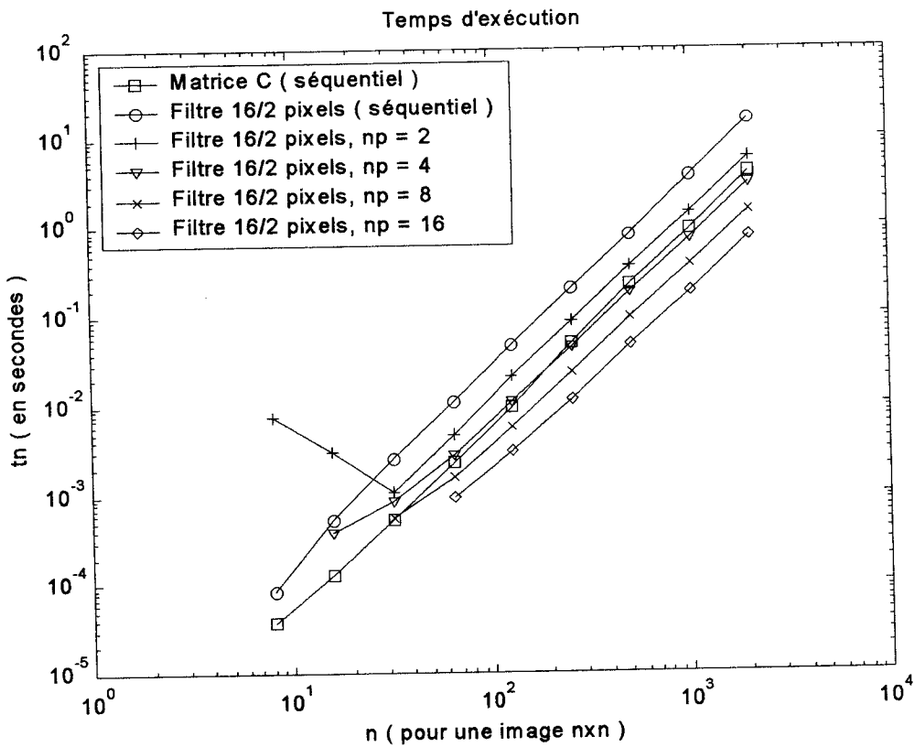
**Figure 36 :** *Temps de calcul de la matrice C en séquentiel et du filtre 16/1 pixel : (a) réseau linéaire pour np = 1, 2, 4, 8 et 16 processeurs, (b) grille pour np = 1, 4, 8 et 16 processeurs.*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Matrice 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 | 4.0 |
| Filtre 16/2, 1 proc. | 8.4e-5 | 5.2e-4 | 2.5e-3 | 1.1e-2 | 4.8e-2 | 2.0e-1 | 8.1e-1 | 3.6e 0 | 16.0 |
| Filtre 16/2, 2 proc. | 7.6e-3 | 3.0e-3 | 1.1e-3 | 4.7e-3 | 2.08e-2 | 8.8e-2 | 3.6e-1 | 1.45 | 6.0 |
| Filtre 16/2, 4 proc. | --- | 0.4e-3 | 0.9e-3 | 2.8e-3 | 1.1e-2 | 4.6e-2 | 0.184 | 0.74 | 3.0 |
| Filtre 16/2, 8 proc. | --- | --- | 0.56e-3 | 1.6e-3 | 5.6e-3 | 2.3e-2 | 0.094 | 0.37 | 1.5 |
| Filtre 16/2, 16 proc. | --- | --- | --- | 1.0e-3 | 3.0e-3 | 1.16e-2 | 0.05 | 0.2 | 0.8 |

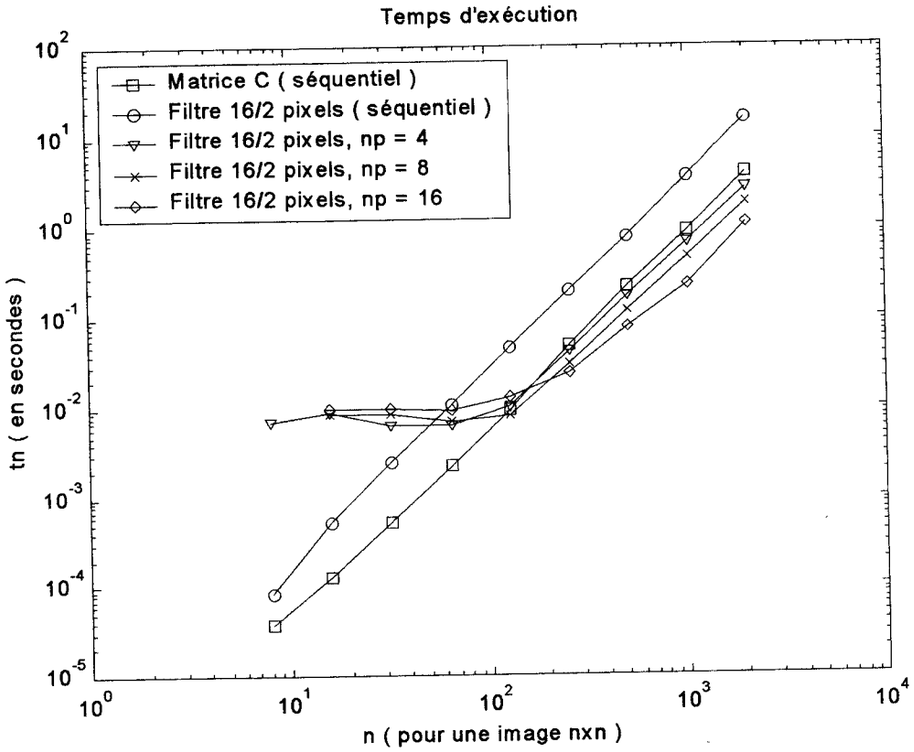
**Tableau 15.** *Temps de calcul (s) de la matrice C en séquentiel et du filtre 16/2 pixels pour np = 1, 2, 4, 8 et 16 processeurs (réseau linéaire).*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Matrice 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 | 4.0 |
| Filtre 16/2, 1 proc. | 8.4e-5 | 5.2e-4 | 2.5e-3 | 1.1e-2 | 4.8e-2 | 2.0e-1 | 8.1e-1 | 3.6 | 16.0 |
| Filtre 16/2, 4 proc. | 7.2e-3 | 8.8e-3 | 6.4e-3 | 6.4e-3 | 1.1e-2 | 4.4e-2 | 1.8e-1 | 0.7 | 2.8 |
| Filtre 16/2, 8 proc. | --- | 8.5e-3 | 8.5e-3 | 7.3e-3 | 8.3e-3 | 3.1e-2 | 1.2e-1 | 0.47 | 1.9 |
| Filtre 16/2, 16 proc. | --- | 1.0e-2 | 9.8e-3 | 9.7e-3 | 1.3e-2 | 2.5e-2 | 7.8e-2 | 0.24 | 1.1 |

**Tableau 16.** *Temps de calcul (s) de la matrice C en séquentiel et du filtre 16/2 pixels pour np = 1, 4, 8 et 16 processeurs (grille de processeurs).*




(a)




(b)

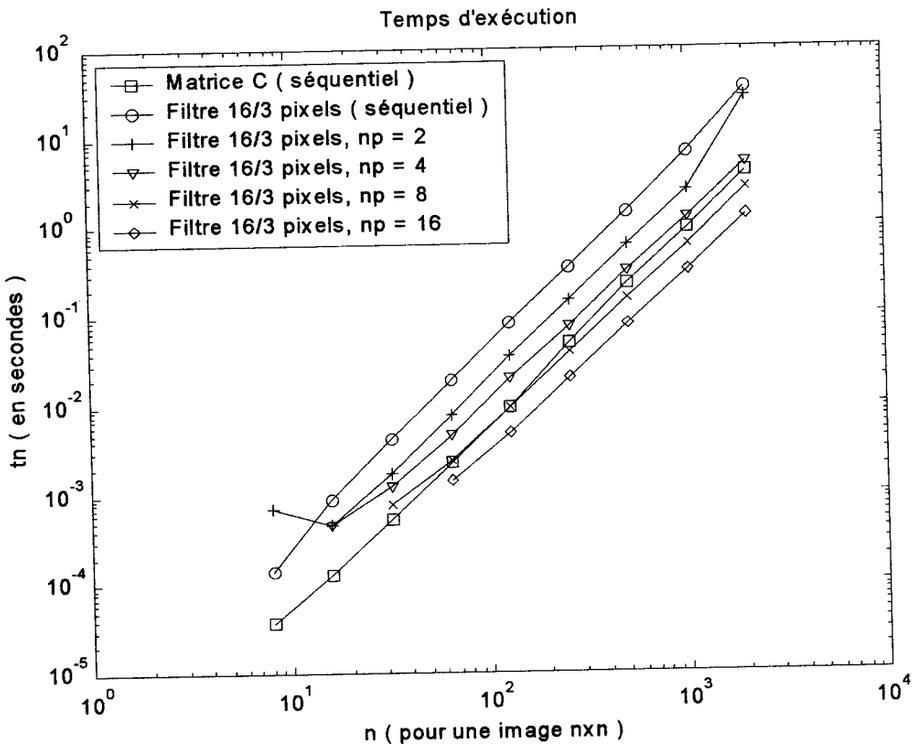
**Figure 37 :** Temps de calcul de la matrice C en séquentiel et du filtre 16/2 pixels : *(a) réseau linéaire pour np = 1, 2, 4, 8 et 16 processeurs, (b) grille pour np = 1, 4, 8 et 16 processeurs.*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Matrice 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 |
| Filtre 16/3, 1 proc. | 1.4e-4 | 8.8e-4 | 4.3e-3 | 1.9e-2 | 8.4e-2 | 3.4e-1 | 1.4e 0 | 6.8e0 | 3.6e1 |
| Filtre 16/3, 2 proc. | 7.0e-4 | 4.5e-4 | 1.75e-3 | 7.8e-3 | 3.6e-2 | 1.5e-1 | 6.1e-1 | 2.5e0 | 2.7e1 |
| Filtre 16/3, 4 proc. | --- | 0.5e-3 | 1.3e-3 | 4.6e-3 | 2.0e-2 | 7.6e-2 | 3.1e-1 | 1.2e0 | 5.1e0 |
| Filtre 16/3, 8 proc. | --- | --- | 0.8e-3 | 2.5e-3 | 9.4e-3 | 3.9e-2 | 1.5e-1 | 6.2e-1 | 2.6e0 |
| Filtre 16/3, 16 proc. | --- | --- | --- | 1.4e-3 | 4.8e-3 | 2.0e-2 | 8.0e-2 | 3.1e-1 | 1.3e0 |

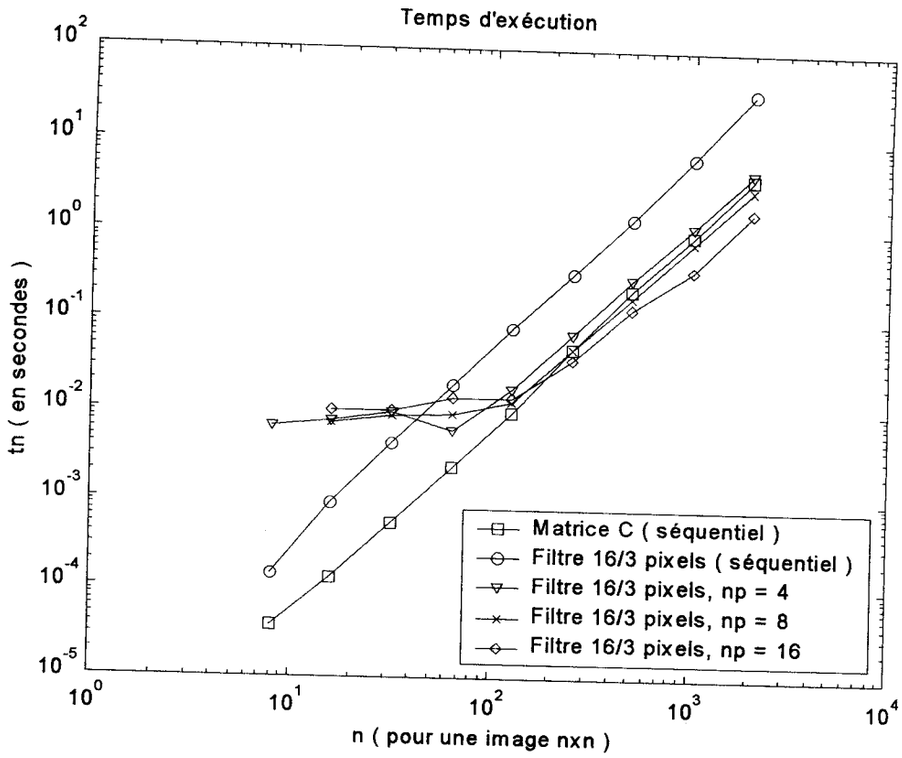
**Tableau 17.** *Temps de calcul (s) de la matrice C en séquentiel et du filtre 16/3 pixels pour np = 1, 2, 4, 8 et 16 processeurs (réseau linéaire).*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Matrice C, 1 proc. | 3.8e-5 | 1.3e-4 | 5.4e-4 | 2.3e-3 | 9.4e-3 | 4.9e-2 | 2.2e-1 | 0.94 | 4.0 |
| Filtre 16/3, 1 proc. | 1.4e-4 | 8.8e-4 | 4.3e-3 | 1.9e-2 | 8.4e-2 | 3.4e-1 | 1.4e0 | 6.8 | 36.0 |
| Filtre 16/3, 4 proc. | 6.4e-3 | 7.4e-3 | 9.4e-3 | 5.9e-3 | 1.8e-2 | 7.3e-2 | 2.9e-1 | 1.2 | 4.7 |
| Filtre 16/3, 8 proc. | --- | 7.2e-3 | 8.6e-3 | 8.9e-3 | 1.3e-2 | 4.9e-2 | 1.9e-1 | 0.76 | 3.0 |
| Filtre 16/3, 16 proc. | --- | 1.0e-2 | 1.0e-2 | 1.4e-2 | 1.4e-2 | 3.8e-2 | 1.4e-1 | 0.4 | 1.7 |

**Tableau 18.** *Temps de calcul (s) de la matrice C en séquentiel et du filtre 16/3 pixels pour np = 1, 4, 8 et 16 processeurs (grille de processeurs).*




(a)




(b)

**Figure 38** : *Temps de calcul de la matrice C en séquentiel et du filtre 16/3 pixels : (a) réseau linéaire pour np = 1, 2, 4, 8 et 16 processeurs, (b) grille pour np = 1, 4, 8 et 16 processeurs.*

Le nombre np de processeurs permettant d'obtenir un temps de calcul du filtre des 16 pixels identique à celui de la matrice C, pour m pixels de sortie peut être estimé par :

$$np = e^{\alpha m} \qquad (5.53)$$

Le tableau 19 représente les valeurs calculées expérimentalement de $\alpha$ et de $e^{\alpha}$ dans le cas d'un réseau linéaire.

| | m | 1 | 2 | 3 |
|---|---|---|---|---|
| $\alpha$ | | 0.67 | 0.69 | 0.65 |
| $e^{\alpha}$ | | 1.95 | 1.99 | 1.92 |

**Tableau 19.** *Valeurs expérimentales du coefficient $\alpha$ et de $e^{\alpha}$ dans le cas d'un réseau linéaire.*

Par conséquent l'égalité des temps de calcul du filtre des 16 pixels et du calcul de la matrice C est vérifiée pour $np \approx 2^m$.

### 5.2.2 Accélération et efficacité

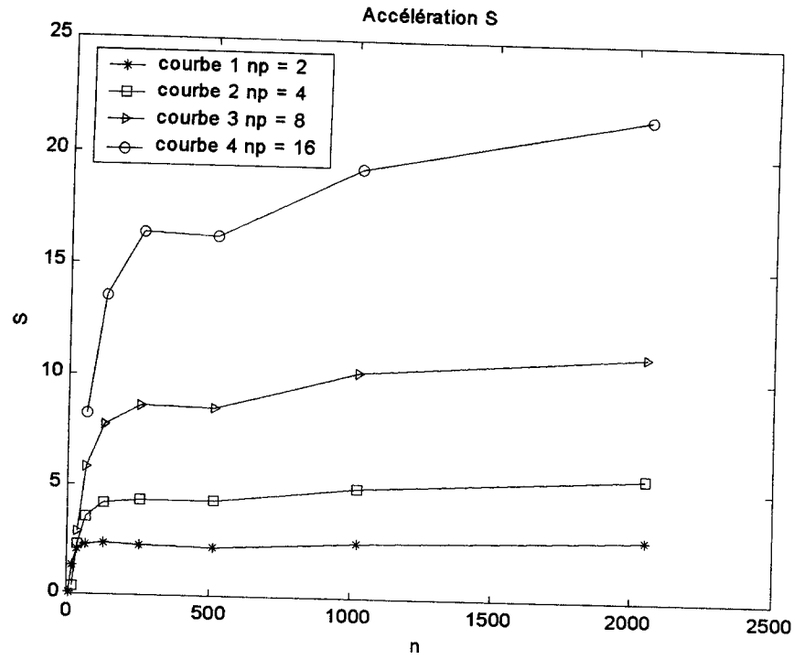
Les figures 39 et 40 représentent les accélérations et efficacités résultantes. Les courbes de l'accélération S et de l'efficacité E permettent de montrer qu'on obtient de bonnes accélérations et efficacités pour des images d'entrée de dimensions n supérieur ou égal à 64. En plus de la parallélisation des calculs, on bénéficie de la meilleure gestion de la mémoire (moins de défaut dans la pagination) qui apporte des efficacités supérieures à l'unité (voir tableaux 20, 21, 22, 23).

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Accélération, 2 proc. | 0.04 | 1.31 | 2.08 | 2.29 | 2.35 | 2.24 | 2.22 | 2.52 | 2.80 |
| Accélération, 4 proc. | --- | 0.38 | 2.28 | 3.5 | 4.2 | 4.29 | 4.3 | 5.0 | 5.56 |
| Accélération, 8 proc. | --- | --- | 2.86 | 5.8 | 7.67 | 8.58 | 8.51 | 10.22 | 11.05 |
| Accélération, 16 proc. | --- | --- | --- | 8.18 | 13.53 | 16.44 | 16.26 | 19.37 | 21.70 |

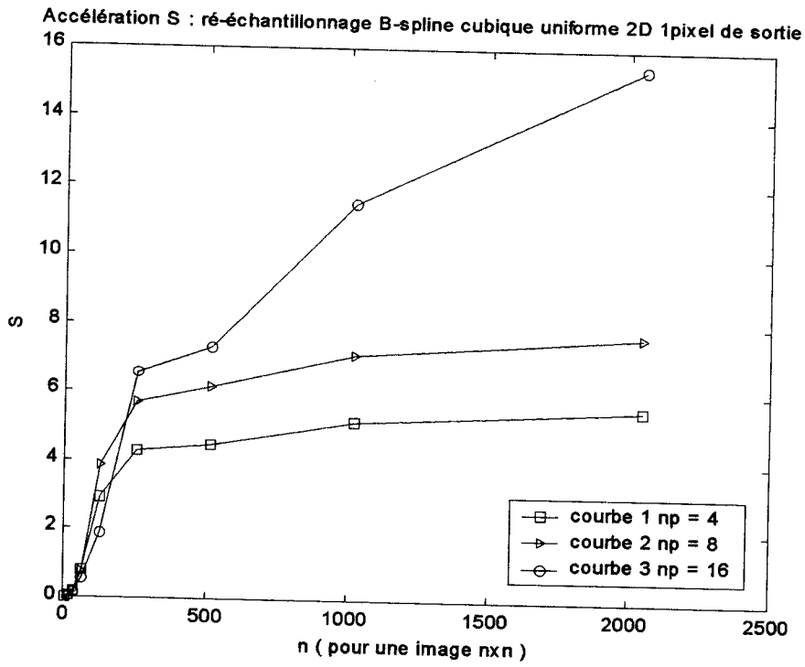
**Tableau 20 .** *Accélération S pour np = 2, 4, 8 et 16 processeurs ( réseau linéaire).*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Accélération, 4 proc. | 0.53e-2 | 0.034 | 0.20 | 0.78 | 2.90 | 4.29 | 4.44 | 5.14 | 5.53 |
| Accélération, 8 proc. | --- | 0.03 | 0.18 | 0.73 | 3.83 | 5.7 | 6.15 | 7.10 | 7.65 |
| Accélération, 16 proc. | --- | 0.024 | 0.13 | 0.57 | 1.90 | 6.51 | 7.27 | 11.5 | 15.43 |

**Tableau 21.** *Accélération S pour np = 4, 8 et 16 processeurs ( grille).*




(a)




(b)

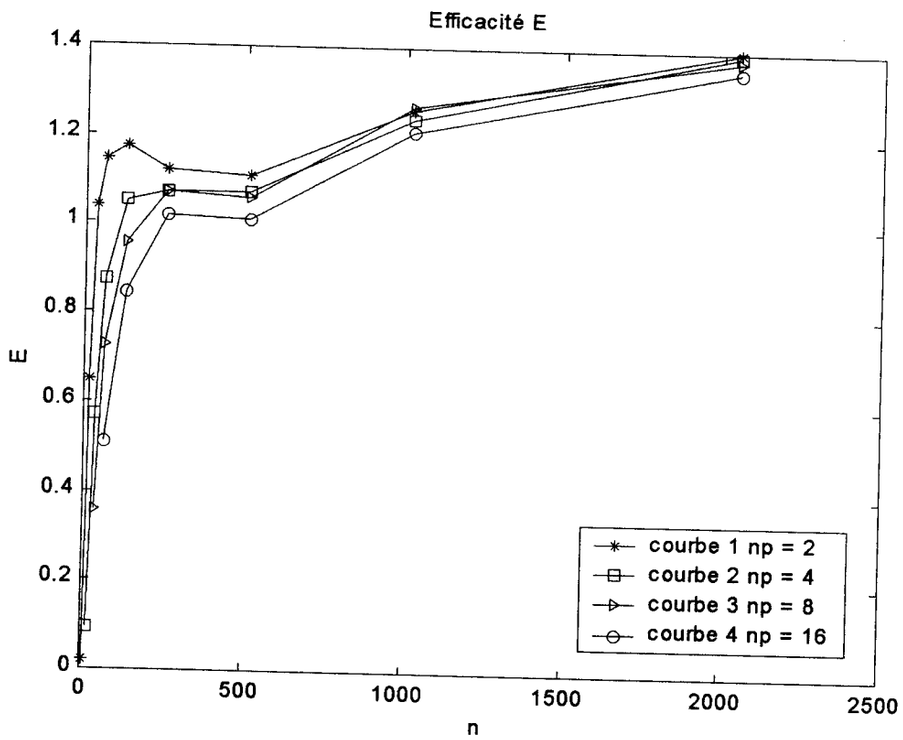
**Figure 39 :** *Accélération S : (a) réseau linéaire pour np = 2, 4, 8 et 16 processeurs, (b) grille de processeurs pour np = 4, 8 et 16 processeurs.*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Efficacité, 2 proc. | 0.02 | 0.65 | 1.04 | 1.15 | 1.17 | 1.12 | 1.11 | 1.26 | 1.40 |
| Efficacité, 4 proc. | --- | 0.01 | 0.57 | 0.88 | 1.05 | 1.07 | 1.07 | 1.24 | 1.39 |
| Efficacité, 8 proc. | --- | --- | 0.36 | 0.73 | 0.96 | 1.07 | 1.06 | 1.27 | 1.38 |
| Efficacité, 16 proc. | --- | --- | --- | 0.51 | 0.85 | 1.02 | 1.01 | 1.21 | 1.36 |

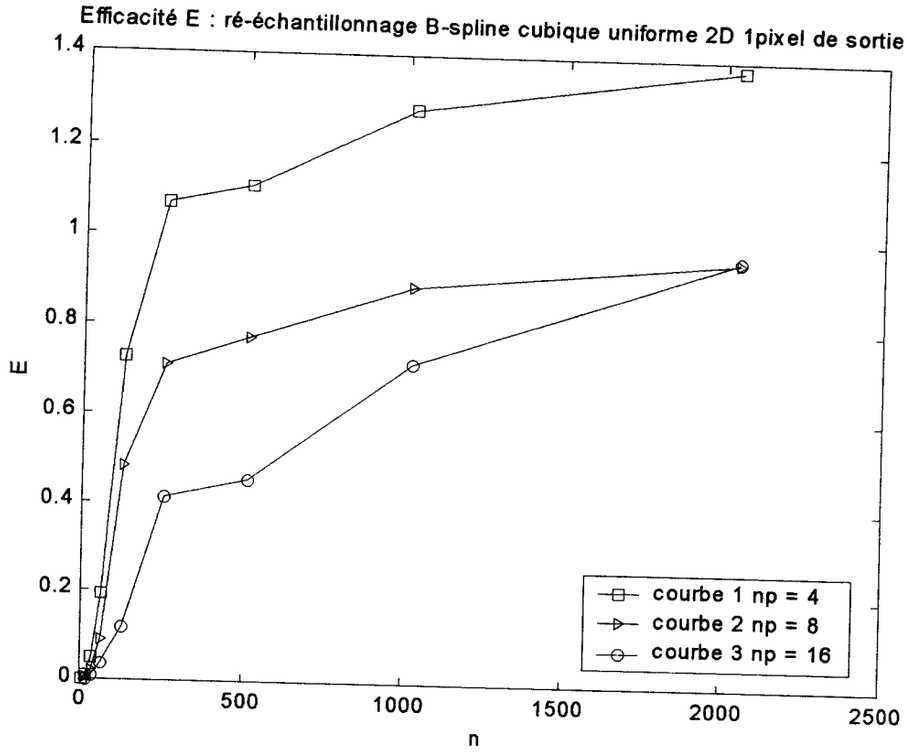
**Tableau 22 .** *Efficacité pour np = 2, 4, 8 et 16 processeurs (réseau linéaire).*

| n | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|---|---|---|---|
| Efficacité, 4 proc. | 0.13e-2 | 0.85e-2 | 0.05 | 0.20 | 0.73 | 1.07 | 1.11 | 1.28 | 1.38 |
| Efficacité, 8 proc. | --- | 0.38e-2 | 0.02 | 0.09 | 0.48 | 0.71 | 0.77 | 0.89 | 0.96 |
| Efficacité, 16 proc. | --- | 0.15e-2 | 0.81e-2 | 0.04 | 0.12 | 0.41 | 0.45 | 0.72 | 0.96 |

**Tableau 23.** *Efficacité pour np = 4, 8 et 16 processeurs (grille).*




(a)




(b)

**Figure 40 :** *Efficacité E : (a) réseau linéaire pour np = 2, 4, 8 et 16 processeurs, (b) grille de processeurs pour np = 4, 8 et 16 processeurs.*

# Chapitre 6
# Description des codes développés et implémentés

## 6.1 Codes de la bilinéaire

### 6.1.1 Codes séquentiels

Nous avons développé les sous programmes COEFF , COEFF1 et COEFF2.

- **COEFF( )**

Est un sous-programme de calcul des coefficients de la fonction de base de la bilinéaire.

**Paramètres :**

[IN N1] paramètre dimension de l'image d'entrée suivant x est N : IN N1 = N-1
[IN M1] dimension de l'image de sortie suivant x est M: IN M1 = M -1
[IN NP1] dimension de l'image d'entrée suivant y est NP : IN NP1 = NP-1
[IN MP1] dimension de l'image de sortie suivant y est MP : IN MP1 = MP -1
[IN J1] nombre de pixels de sortie suivant x est IE : IN J1 = IE+1
[IN JP1] nombre de pixels de sortie suivant x est JE : IN JP1 = JE+1
[OUT A] tableau de coefficients de la fonction de base de dimension ≥ J1
[OUT C] tableau de coefficients de la fonction de base de dimension ≥ J1
[OUT D] tableau de coefficients de la fonction de base de dimension ≥ JP1
[OUT F] tableau de coefficients de la fonction de base de dimension ≥ JP1

**Syntaxe en Fortran:**

```
COEFF(N1, M1, NP1, MP1, J1, JP1, A, C, D, F)
INTEGER N1, M1, NP1, MP1, J1, JP1
DOUBLE PRECISION A, C, D, F
```

**Code Fortran :**

```
SUBROUTINE COEFF(N1, M1, NP1, MP1, J1, JP1, A, C, D, F)
DOUBLE PRECISION A(J1) , C(J1), D(JP1), F(JP1)
XN=N1
YM=M1
XNP=NP1
YMP=MP1
```

```
      XL=XN/YM
      XLP=XNP/YMP
      C(1)=0
      A(1)=0
      DO 10 J=1,J1
      C(J)=(J-1)*XL
      A(J)=1-C(J)
10    CONTINUE
      D(1)=0
      F(1)=0
      DO 15 J=1,JP1
      F(J)=(J-1)*XLP
      D(J)=1-F(J)
15    CONTINUE
      RETURN
      END
```

- **COEFF1( )**

  Est un sous-programme de calcul des coefficients filtre des 4 pixels.

  **Paramètres :**

  [IN J1] nombre de pixels de sortie suivant x est IE : IN J1 = IE+1

  [IN JP1] nombre de pixels de sortie suivant x est JE : IN JP1 = JE+1

  [IN A] tableau de coefficients de la fonction de base de dimension ≥ J1

  [IN C] tableau de coefficients de la fonction de base de dimension ≥ J1

  [IN D] tableau de coefficients de la fonction de base de dimension ≥ JP1

  [IN F] tableau de coefficients de la fonction de base de dimension ≥ JP1

  [OUT H] tableau de coefficients des 4 pixels de dimension 3 : ≥ 4, ≥ J1, ≥ JP1

  **Syntaxe en Fortran:**

```
COEFF1(J1, JP1, A, C, D, H)
INTEGER J1, JP1
DOUBLE PRECISION A, C, D, F, H
```

**Code Fortran :**

```
SUBROUTINE COEFF1(J1, JP1, A, C, D, F, H)
DOUBLE PRECISION A(J1) , C(J1), D(JP1), F(JP1), H(4,J1,JP1)
H(1,1,1)=0
H(2,1,1)=0
H(3,1,1)=0
H(4,1,1)=0
DO 90 IA=2,J1
H(1,IA,1)=0
H(2,IA,1)=0
H(3,IA,1)=0
H(4,IA,1)=0
DO 90 IB=2,JP1
H(1,1,IB)=0
H(2,1,IB)=0
H(3,1,IB)=0
H(4,1,IB)=0
H(1,IA,IB)=A(IA)*D(IB)
H(2,IA,IB)=A(IA)*F(IB)
H(3,IA,IB)=C(IA)*D(IB)
H(4,IA,IB)=C(IA)*F(IB)
90 CONTINUE
RETURN
END
```

- **COEFF2( )**

Ce programme calcule les pixels de sortie.

**Paramètres :**

[IN N] dimension de l'image d'entrée suivant x

[IN N1] paramètre IN N1 = N-1

[IN NP1] dimension de l'image d'entrée suivant y est NP : IN NP1 = NP-1

[IN M] dimension de l'image de sortie suivant x

[IN MP] dimension de l'image de sortie suivant y

[IN J1] nombre de pixels de sortie suivant x est IE : IN J1 = IE+1

[IN JP1] nombre de pixels de sortie suivant x est JE : IN JP1 = JE+1

[IN A] tableau de coefficients de la fonction de base de dimension ≥ J1

[IN C] tableau de coefficients de la fonction de base de dimension ≥ J1

[IN D] tableau de coefficients de la fonction de base de dimension ≥ JP1

[IN F] tableau de coefficients de la fonction de base de dimension ≥ JP1

[IN H] tableau de coefficients des 4 pixels de dimensions 3 : ≥ 4, ≥ J1, ≥ JP1

[IN G] tableau de coefficients de dimensions 2 : ≥ N, ≥ NP

[OUT S] tableau de coefficients de dimensions 2 : ≥ M, ≥ MP

**Syntaxe en Fortran:**

```
COEFF2(N, N1, NP, NP1, M, MP, J1, JP1, A, C, D, F, H, G, S)
INTEGER N1, M1, NP1, MP1, J1, JP1
DOUBLE PRECISION A, C, D, F, H,G, S
```

**Code Fortran :**

```
 SUBROUTINE COEFF2(N, N1, NP, NP1, M, MP, J1, JP1, A, C, D, F, H, G, S)
 DOUBLE PRECISION(M,MP), G(N,NP) , A(J1) , C(J1), D(JP1), F(JP1),
+H(4,J 1,JP1)
 DO 20 JP=1,NP1
 JP2=JP+1
 J=(JP-1)*JP1
 J2=J+1
 DO 25 K=1,N1
 K1=K+1
 I=(K-1)*J1
 I1=I+1
 S(I1,J2)=G(K,JP)
 DO 30 IA=2,J1
 S(I+IA,J2)=A(IA)*G(K,JP)+C(IA)*G(K1,JP)
30 CONTINUE
 DO 40 IB=2,JP1
```

```
      S(I1,J+IB)=D(IB)*G(K,JP)+F(IB)*G(K,JP2)
40    CONTINUE
      DO 45 IB=2,JP1
      JS=J+IB
      DO 45 IA=2,J1
      IS=I+IA
      S(IS,JS)=H(1,IA,IB)*G(K,JP)+H(2,IA,IB)*G(K,JP2)
      S(IS,JS)=S(IS,JS)+H(3,IA,IB)*G(K1,JP)+H(4,IA,IB)*G(K1,JP2)
45    CONTINUE
25    CONTINUE
      S(M, J2)= G(N, JP)
      DO 80 IB=1,JP1
80    S(M, J+IB)=D(IB)*G(N, JP)+F(IB)*G(N,JP2)
20    CONTINUE
      DO 60 K=1,N1
      K1=K+1
      I=(K-1)*J1
      S(I+1,MP)=G(K,NP)
      DO 60 IA=2,J1
      S(I+IA,MP)=A(IA)*G(K,NP)+C(IA)*G(K1,NP)
60    CONTINUE
      S(M,MP)=G(N,NP)
      RETURN
      END
```

### 6.1.2 Codes parallèles du réseau linéaire

Nous avons adapté les sous-programmes MPE_DECOMP1D et EXCHNG1. Le sous programme MPI_CART_SHIFT détermine le réseau linéaire en définissant les processeurs voisins.

- **MPE_DECOMP1D( )**

  Ce sous programme permet à chaque processeur de reconnaître ses limites (inférieur et supérieure) des donnée d'entrée.

**Paramètres :**

[IN N] dimension de l'image d'entrée

[IN NUMPROCS] nombre de processeurs dans le communicateur

[IN MYID] le rang du processeur

[OUT S] limite inférieur de la fenêtre d'entrée suivant x du processeur

[OUT E] limite supérieur de la fenêtre d'entrée suivant x du processeur

[OUT MS] limite inférieur de la fenêtre de sortie suivant x du processeur

[OUT ME] limite supérieur de la fenêtre de sortie suivant x du processeur

[IN IE] nombre de pixels de sortie

**Syntaxe en Fortran:**

```
MPE_DECOMP1D( N, NUMPROCS, MYID, S, E, MS, ME, IE )
INTEGER N, NUMPROCS, MYID, S, E, IE, ME, MS
```

**Code Fortran :**

```
SUBROUTINE MPE_DECOMP1D( N, NUMPROCS, MYID, S, E, MS, ME, IE )
INTEGER N, NUMPROCS, MYID, S, E, IE, ME, MS
INTEGER NLOCAL
INTEGER DEFICIT
NLOCAL = N / NUMPROCS
S      = MYID*NLOCAL + 1
DEFICIT = MOD(N,NUMPROCS)
S      = S + MIN(MYID,DEFICIT)
MS = (IE+1)*(S-1)+1
IF (MYID .LT. DEFICIT) THEN
NLOCAL = NLOCAL + 1
ENDIF
E = S + NLOCAL - 1
ME = MS -1 + (IE+1)*(E-S+1)
IF (E .GT. N .OR. MYID .EQ. NUMPROCS-1) THEN
E = N
ME = MS + (IE+1)*(E-S)
ENDIF
```

```
RETURN
END
```

- **EXCHNG1( )**

  Ce sous programme permet les communications de donnée entre les processeurs voisins du réseau linéaire. Ce sous programme utilise la routine MPI_SENDRECV de la bibliothèque de communication MPI.

  **Paramètres :**

  [IN AA] donnée d'entrée à échanger
  [IN NX] dimension de l'image d'entrée suivant y
  [IN S] limite inférieur des données d'entrée
  [IN E] limite supérieur des donnée d'entrée
  [IN COMM1D] communicateur
  [IN NBRBOTTOM] processeur voisin du bas dans le communicateur
  [IN NBRTOP] processeur voisin du haut dans le communicateur

  **Syntaxe en Fortran:**

```
EXCHNG1( AA, NX, S, E, COMM1D, NBRBOTTOM, NBRTOP )
INTEGER NX, S, E
DOUBLE PRECISION AA
INTEGER COMM1D, NBRBOTTOM, NBRTOP
```

  **Code Fortran :**

```
SUBROUTINE EXCHNG1( AA, NX, S, E, COMM1D, NBRBOTTOM, NBRTOP )
INCLUDE "MPIF.H"
INTEGER NX, S, E
DOUBLE PRECISION AA(1:NX,S:E+1)
INTEGER COMM1D, NBRBOTTOM, NBRTOP
INTEGER STATUS(MPI_STATUS_SIZE), IERR
CALL MPI_SENDRECV( AA(1,S), NX, MPI_DOUBLE_PRECISION,
$ NBRBOTTOM, 1,
$     AA(1,E+1), NX, MPI_DOUBLE_PRECISION,
```

```
$     NBRTOP, 1, COMM1D, STATUS, IERR )
      RETURN
      END
```

### 6.1.3 Codes parallèles de la grille 2D

Nous avons adapté les sous-programmes FND2DNBRS, FND2DDECOMP et EXCHNG2.

- **FND2DNBRS( )**

  Ce sous programme détermine les processeurs voisins de la grille 2D.

  **Paramètres :**

  [IN COMM2D] communicateur
  [OUT NBRLEFT] processeur voisin du gauche dans le communicateur
  [OUT NBRRIGHT] processeur voisin de droite dans le communicateur
  [OUT NBRBOTTOM] processeur voisin du bas dans le communicateur
  [OUT NBRTOP] processeur voisin du haut dans le communicateur

  **Syntaxe en Fortran:**

```
FND2DNBRS( COMM2D, NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM )
INTEGER COMM2D, NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM
```

  **Code Fortran :**

```
SUBROUTINE FND2DNBRS( COMM2D, NBRLEFT, NBRRIGHT, NBRTOP,
$NBRBOTTOM )
INTEGER COMM2D, NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM
INTEGER IERR
CALL MPI_CART_SHIFT( COMM2D, 0, 1, NBRLEFT, NBRRIGHT, IERR )
CALL MPI_CART_SHIFT( COMM2D, 1, 1, NBRBOTTOM, NBRTOP, IERR )
RETURN
END
```

- **FND2DDECOMP( )**

  Ce sous programme permet à chaque processeur de reconnaître ses limites inférieures et supérieures de la fenêtre 2D. Ce sous programme utilise le sous programmes MPE_DECOMP1D et le sous programme MPI_CART_GET de la bibliothèque MPI.

  **Paramètres :**

  [IN COMM2D] nombre de processeurs dans le communicateur
  [IN N] dimension de l'image d'entrée
  [OUT SX] limite inférieur de la fenêtre d'entrée suivant x du processeur
  [OUT EX] limite supérieur de la fenêtre d'entrée suivant x du processeur
  [OUT SY] limite inférieur de la fenêtre d'entrée suivant y du processeur
  [OUT EY] limite supérieur de la fenêtre d'entrée suivant y du processeur
  [OUT MSX] limite inférieur de la fenêtre de sortie suivant x du processeur
  [OUT MEX] limite supérieur de la fenêtre de sortie suivant x du processeur
  [OUT MSY] limite inférieur de la fenêtre de sortie suivant Y du processeur
  [OUT MEY] limite supérieur de la fenêtre de sortie suivant Y du processeur
  [IN IE] nombre de pixels de sortie

  **Syntaxe en Fortran:**

```
FND2DDECOMP( COMM2D, N, SX, EX, SY, EY, MSX, MEX, MSY, MEY, IE)
INTEGER COMM2D, N, SX, EX, SY, EY, MSX, MEX, MSY, MEY, IE
```

  **Code Fortran :**

```
 SUBROUTINE FND2DDECOMP( COMM2D, N, SX, EX, SY, EY,
$                MSX, MEX, MSY, MEY, IE)
 INTEGER COMM2D
 INTEGER N, SX, EX, SY, EY
 INTEGER MSX, MEX, MSY, MEY, IE
 INTEGER DIMS(2), COORDS(2), IERR
 LOGICAL PERIODS(2)
 CALL MPI_CART_GET( COMM2D, 2, DIMS, PERIODS, COORDS,IERR )
 CALL MPE_DECOMP1D( N, DIMS(1), COORDS(1), SX, EX, MSX, MEX, IE)
 CALL MPE_DECOMP1D( N, DIMS(2), COORDS(2), SY, EY, MSY, MEY, IE)
```

```
RETURN
END
```

- **EXCHNG2( )**

  Ce sous programme permet les communications de donnée entre les processeurs voisins de la grille 2D (gauche- droit, bas-haut et suivant la diagonale). Ce sous programme utilise la routine MPI_SENDRECV de la bibliothèque de communication MPI.

  **Paramètres :**

  [IN AA] donnée d'entrée à échanger
  [IN SX] limite inférieur de la fenêtre d'entrée suivant x du processeur
  [IN EX] limite supérieur de la fenêtre d'entrée suivant x du processeur
  [IN SY] limite inférieur de la fenêtre d'entrée suivant y du processeur
  [IN EY] limite supérieur de la fenêtre d'entrée suivant y du processeur
  [IN COMM2D] communicateur
  [IN STRIDETYPE] type vecteur
  [IN NBRLEFT] processeur voisin du gauche dans le communicateur
  [IN NBRRIGHT] processeur voisin de droite dans le communicateur
  [IN NBRBOTTOM] processeur voisin du bas dans le communicateur
  [IN NBRTOP] processeur voisin du haut dans le communicateur
  [IN NBRDIAGLEFT] processeur voisin dans la diagonal à gauche
  [IN NBRDIAGRIGHT] processeur voisin dans la diagonal à droite

  **Syntaxe en Fortran:**

```
EXCHNG2( AA, SX, EX, SY, EY, COMM2D, STRIDETYPE, NBRLEFT,
$              NBRRIGHT, NBRTOP, NBRBOTTOM,
$              NBRDIAGLEFT, NBRDIAGRIGHT )

INTEGER SX, EX, SY, EY, STRIDETYPE
DOUBLE PRECISION AA
INTEGER COMM2D, NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM
INTEGER NBRDIAGLEFT, NBRDIAGRIGHT
```

**Code Fortran :**

```
 SUBROUTINE EXCHNG2( AA, SX, EX, SY, EY,
$                    COMM2D, STRIDETYPE,
$                    NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM,
$                    NBRDIAGLEFT, NBRDIAGRIGHT )
 INCLUDE "MPIF.H"
 INTEGER SX, EX, SY, EY, STRIDETYPE
 DOUBLE PRECISION AA(SX:EX+1,SY:EY+1)
 INTEGER COMM2D, NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM
 INTEGER NBRDIAGLEFT, NBRDIAGRIGHT
 INTEGER STATUS(MPI_STATUS_SIZE), IERR, NNX
 NNX = EX - SX +1
 CALL MPI_SENDRECV( AA(SX,SY), NNX, MPI_DOUBLE_PRECISION,
$                    NBRBOTTOM, 0,
$                    AA(SX,EY+1), NNX, MPI_DOUBLE_PRECISION,
$                    NBRTOP, 0, COMM2D, STATUS, IERR )
 CALL MPI_SENDRECV( AA(SX,SY), 1, STRIDETYPE, NBRLEFT, 1,
$                    AA(EX+1,SY), 1, STRIDETYPE, NBRRIGHT, 1,
$                    COMM2D, STATUS, IERR )
 CALL MPI_SENDRECV( AA(SX,SY), 1, MPI_DOUBLE_PRECISION,
$                    NBRDIAGLEFT, 2,
$                    AA(EX+1,EY+1), 1, MPI_DOUBLE_PRECISION,
$                    NBRDIAGRIGHT, 2,
$                    COMM2D, STATUS, IERR )
 RETURN
 END
```

## 6.2 Codes de la B-spline cubique uniforme

### 6.2.1 Code du calcul de la matrice C par la factorisation de Cholesky

Nous avons développé les sous programmes CHOLA, VECTA, LEMDA, CALY, ALPHAY, CALU, CALX, CALZ, CALC pour calculer la matrice C.

- **CHOLA( )**

  Ce sous programme calcule les vecteurs de la diagonale et de la diagonale inférieure de la matrice A.

  **Paramètres :**

  [IN NCH] dimension d'entrée de A'
  [OUT LD] vecteur de la diagonale
  [OUT LB] vecteur de la diagonale inférieure

  **Syntaxe en Fortran:**

```
CHOLA(NCH, LD, LB)
INTEGER NCH
DOUBLE PRECISION LD, LB
```

  **Code Fortran :**

```
SUBROUTINE CHOLA(NCH, LD, LB)
DOUBLE PRECISION LD(NCH), LB(NCH-1)
INTEGER NCH
LD(1) = 2.0
LB(1) = 0.5
DO I=2,NCH-1
LD(I) = SQRT(4-LB(I-1)**2)
LB(I) = 1/LD(I)
ENDDO
LD(NCH) = SQRT(4-LB(NCH-1)**2)
RETURN
END
```

- **VECTA( )**

  Ce sous programme calcule le vecteur de la factorisation de Cholesky de la matrice A.

**Paramètres :**

[IN LD] vecteur de la diagonale

[IN LB] vecteur de la diagonale inférieure

[IN NCH] dimension d'entrée de A'

[OUT LV] vecteur de la factorisation

**Syntaxe en Fortran:**

```
VECTA(LD, LB, NA, LV)
INTEGER NA
DOUBLE PRECISION LD, LB, LV
```

**Code Fortran :**

```
SUBROUTINE VECTA( LD, LB, NA, LV)
DOUBLE PRECISION LV(NA)
DOUBLE PRECISION LD(NA), LB(NA-1)
INTEGER NA
LV(1) = 1/LD(1)
DO I=2,NA-1
LV(I)=-LB(I-1)*LV(I-1)/LD(I)
ENDDO
LV(NA)= (1.-LB(NA-1)*LV(NA-1))/LD(NA)
RETURN
END
```

- **LEMDA( )**

Calcul du coefficient lembda de la matrice de Cholesky.

**Paramètres :**

[IN LV] vecteur de la factorisation

[IN N] dimension d'entrée de A'

[OUT LEM] coefficient

**Syntaxe en Fortran:**

```
LEMDA( LV, N, LEM)
INTEGER N
DOUBLE PRECISION LV, LEM
```

**Code Fortran :**

```
SUBROUTINE LEMDA( LV, N, LEM)
DOUBLE PRECISION LV(N), SLEM, LEM
INTEGER N
SLEM = 0.
DO I=1,N
SLEM = SLEM + LV(I)**2
ENDDO
LEM = SQRT(4-SLEM)
RETURN
END
```

- **CALY( )**

Ce sous programme Calcule la matrice Y d'ordre n-1 tel que L'Y = G'

**Paramètres :**

[IN G] donnée de l'image d'entrée : (n-1)xn
[IN N] dimension d'entrée de A
[IN LD] vecteur de la diagonale
[IN LB] vecteur de la diagonale inférieure
[OUT Y] valeur de sortie Y

**Syntaxe en Fortran:**

```
CALY( G, N, LD, LB, Y)
INTEGER N
DOUBLE PRECISION G, Y, LD(N-1), LB(N-1)
```

**Code Fortran :**

```
SUBROUTINE CALY( G, N, LD, LB, Y)
DOUBLE PRECISION G(N-1,N), Y(N-1,N)
DOUBLE PRECISION LD(N-1), LB(N-1)
INTEGER N
DO J=1,N
Y(1,J) = G(1,J)/LD(1)
DO I=2,N-1
Y(I,J) = (G(I,J)-Y(I-1,J)*LB(I-1))/LD(I)
ENDDO
ENDDO
RETURN
END
```

- **ALPHAY( )**

  Ce sous programme calcule le vecteur alpha de Y.

**Paramètres :**

[IN N] dimension d'entrée de A
[IN LV] vecteur de la factorisation
[IN Y] valeur d'entrée de Y
[IN BETA] vecteur d'entrée de G
[IN LEM] coefficient lemda
[OUT ALY] vecteur alpha de Y

**Syntaxe en Fortran:**

```
ALPHAY( N, LV, Y, BETA, LEM, ALY)
INTEGER N
DOUBLE PRECISION Y, LV, BETA
DOUBLE PRECISION ALY, LEM
```

**Code Fortran :**

```
SUBROUTINE ALPHAY( N, LV, Y, BETA, LEM, ALY)
DOUBLE PRECISION Y(N-1,N), LV(N-1), BETA(N)
DOUBLE PRECISION ALY(N), LEM, XY
INTEGER N
DO J=1,N
XY = 0.
DO I=1,N-1
XY = XY + LV(I)*Y(I,J)
ENDDO
ALY(J) = (BETA(J)-XY)/LEM
ENDDO
RETURN
END
```

- **CALU( )**

Ce sous programme calcule la matrice U et vecteur de coefficients alpha de X.

**Paramètres :**

[IN N] dimension d'entrée de A
[IN LEM] coefficient lemda
[IN Y] valeur d'entrée de Y
[IN LV] vecteur de la factorisation
[IN ALY] vecteur alpha de Y
[OUT U] matrice de sortie U(N-1,N)
[OUT ALX] vecteur alpha de X

**Syntaxe en Fortran :**

```
CALU( N, LEM, Y, LV, ALY, U, ALX)
INTEGER N
DOUBLE PRECISION Y, U
DOUBLE PRECISION LV, ALY, ALX, LEM
```

**Code Fortran :**

```
SUBROUTINE CALU( N, LEM, Y, LV, ALY, U, ALX)
DOUBLE PRECISION Y(N-1,N), U(N-1,N)
DOUBLE PRECISION LV(N-1), ALY(N), ALX(N), LEM
INTEGER N
DO I=1,N
ALX(I) = ALY(I)/LEM
ENDDO
DO J=1,N
DO I=1,N-1
U(I,J) = Y(I,J) - ALX(J)*LV(I)
ENDDO
ENDDO
RETURN
END
```

- **CALX( )**

Ce sous programme calcule Calcul la matrice X d'ordre n-1 tel que $L^T X = Y$.

**Paramètres :**

[IN U] matrice de sortie U(N-1,N)
[IN N] dimension d'entrée de A
[IN LD] vecteur de la diagonale
[IN LB] vecteur de la diagonale inférieure
[OUT X] matrice de sortie X(N-1,N)

**Syntaxe en Fortran:**

```
CALX( U, N, LD, LB, X)
DOUBLE PRECISION X, U
DOUBLE PRECISION LD, LB
INTEGER N
```

**Code Fortran :**

```
SUBROUTINE CALX( U, N, LD, LB, X)
DOUBLE PRECISION X(N-1,N), U(N-1,N)
DOUBLE PRECISION LD(N-1), LB(N-2)
INTEGER N
DO J=1,N
X(N-1,J) = U(N-1,J)/LD(N-1)
DO I=2,N-1
X(N-I,J) = (U(N-I,J)-X(N-I+1,J)*LB(N-I))/LD(N-I)
ENDDO
ENDDO
RETURN
END
```

- **CALZ( )**

  Ce sous programme calcule la matrice Z.

**Paramètres :**

[IN N] dimension d'entrée de A
[IN X] matrice d'entrée X(N-1,N)
[IN ALX] vecteur alpha de X
[IN LD] vecteur de la diagonale
[IN LB] vecteur de la diagonale inférieure
[IN LV] vecteur de la factorisation
[IN LEM] coefficient lemda
[OUT Z] matrice de sortie Z(N,N)

**Syntaxe en Fortran :**

```
CALZ(N, X, ALX, LD, LB, LV, LEM, Z)
DOUBLE PRECISION X, Z
DOUBLE PRECISION LD, ALX, LB, LV, LEM
INTEGER N
```

**Code Fortran :**

```
SUBROUTINE CALZ(N, X, ALX, LD, LB, LV, LEM, Z)
DOUBLE PRECISION X(N-1,N), Z(N,N)
DOUBLE PRECISION LD(N-1), ALX(N), LB(N-2), LV(N-1), LEM
INTEGER N
DO I=1,N-1
Z(I,1) = X(I,1)/LD(1)
ENDDO
Z(N,1) = ALX(1)/LD(1)
DO J=2,N-1
DO I=1,N-1
Z(I,J) = (X(I,J)-Z(I,J-1)*LB(J-1))/LD(J)
ENDDO
Z(N,J) = (ALX(J)-Z(N,J-1)*LB(J-1))/LD(J)
ENDDO
DO I=1,N-1
Z(I,N)=0.
DO J=1,N-1
Z(I,N)=Z(I,N)+ Z(I,J)*LV(J)
ENDDO
Z(I,N)=(X(I,N)-Z(I,N))/LEM
ENDDO
Z(N,N)=0.
DO J=1,N-1
Z(N,N)=Z(N,N)+Z(N,J)*LV(J)
ENDDO
Z(N,N)=(ALX(N)-Z(N,N))/LEM
RETURN
END
```

- **CALC( )**

  Ce sous programme calcule la matrice finale C.

**Paramètres :**

[IN N] dimension d'entrée de A

[IN Z] matrice d'entrée X(N,N)

[IN LD] vecteur de la diagonale

[IN LB] vecteur de la diagonale inférieure

[IN LV] vecteur de la factorisation

[IN LEM] coefficient lemda

[OUT C] matrice de sortie C(N,N)

**Syntaxe en Fortran :**

```
CALC(N, Z, LD, LB, LV, LEM, C)
DOUBLE PRECISION C, Z
DOUBLE PRECISION LD, LB, LEM, LV
INTEGER N
```

**Code Fortran :**

```
SUBROUTINE CALC(N, Z, LD, LB, LV, LEM, C)
DOUBLE PRECISION C(N,N), Z(N,N)
DOUBLE PRECISION LD(N-1), LB(N-2),LEM, LV(N-1)
INTEGER N
DO I=1,N
C(I,N)=Z(I,N)/LEM
ENDDO
DO I=1,N
C(I,N-1) = ( Z(I,N-1)-C(I,N)*LV(N-1))/LD(N-1)
ENDDO
DO J=2,N-1
DO I=1,N
C(I,N-J) = ( Z(I,N-J)-C(I,N-J+1)*LB(N-J)-C(I,N)*LV(N-J))/LD(N-J)
ENDDO
ENDDO
RETURN
END
```

### 6.2.2 Code séquentiel du filtre des 16 pixels

Nous avons développé les sous-programmes COEFFS , COEFF1S et COEFF2S.

- **COEFFS( )**

Est un sous-programme de calcul des coefficients de la fonction de base de la B-spline cubique uniforme.

**Paramètres :**

[IN NEL1] nombre de pixels de sortie dans chaque direction est IE : NEL1 = IE+1
[OUT COEL] tableau de coefficients suivant x dimension : (4, NEL1)
[OUT COEC] tableau de coefficients suivant y de dimension : (4, NEL1)

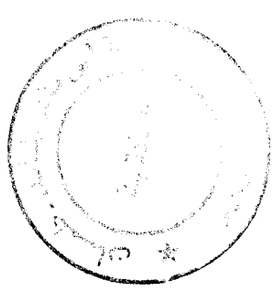
**Syntaxe en Fortran :**

```
COEFFS(NEL1, COEL, COEC)
INTEGER NEL1
DOUBLE PRECISION  COEL, COEC
```

**Code Fortran :**

```
SUBROUTINE COEFFS(NEL1, COEL, COEC)
DOUBLE PRECISION COEL(4,NEL1), COEC(4,NEL1)
DOUBLE PRECISION XPK, YPK, XL, YC, BE, BE1, BE2, BE3
NEC1=NEL1
XPK=NEL1
YPK=NEC1
XL=1.0/XPK
YC=1.0/YPK
COEL(4,1)=0
COEL(3,1)=1
COEL(2,1)=4
COEL(1,1)=1
DO 200 I=2,NEL1
BE=(I-1)*XL
```

```
      BE1=BE+1.
      BE2=BE+2.
      BE3=BE+3.
      BE=BE**3.
      BE1=BE1**3
      BE2=BE2**3
      BE3=BE3**3
      COEL(4,I)=BE
      COEL(3,I)=BE1-4*BE
      COEL(2,I)=BE2-4*BE1+6*BE
      COEL(1,I)=BE3-4*BE2+6*BE1-4*BE
200   CONTINUE
      COEC(4,1)=0.
      COEC(3,1)=1.
      COEC(2,1)=4.
      COEC(1,1)=1.
      DO 210 I=2,NEC1
      BE=(I-1)*YC
      BE1=BE+1.
      BE2=BE+2.
      BE3=BE+3.
      BE=BE**3.
      BE1=BE1**3
      BE2=BE2**3
      BE3=BE3**3
      COEC(4,I)=BE
      COEC(3,I)=BE1-4*BE
      COEC(2,I)=BE2-4*BE1+6*BE
      COEC(1,I)=BE3-4*BE2+6*BE1-4*BE
210   CONTINUE
      RETURN
      END
```



- **COEFF1S( )**

  Ce sous programme calcule les coefficients du filtre des 16 pixels.

  **Paramètres :**

  [IN NEL1] nombre de pixels de sortie dans chaque direction est IE : NEL1 = IE+1
  [IN COEL] tableau de coefficients suivant x dimension : (4, NEL1)
  [IN COEC] tableau de coefficients suivant y de dimension : (4, NEL1)
  [OUT XM] Tableau des coefficients du filtre

  **Syntaxe en Fortran :**

```
COEFF1S(NEL1, COEL, COEC, XM)
INTEGER NEL1
DOUBLE PRECISION COEL, COEC, XM
```

  **Code Fortran :**

```
SUBROUTINE COEFF1S(NEL1, COEL, COEC, XM)
DOUBLE PRECISION COEL(4,40), COEC(4,40)
DOUBLE PRECISION XM(16,40,40)
NEC1=NEL1
XM(1,1,1)=1.
XM(2,1,1)=4.
XM(3,1,1)=1.
XM(4,1,1)=0.
XM(5,1,1)=4.
XM(6,1,1)=16.
XM(7,1,1)=4.
XM(8,1,1)=0.
XM(9,1,1)=1.
XM(10,1,1)=4.
XM(11,1,1)=1.
XM(12,1,1)=0.
XM(13,1,1)=0.
XM(14,1,1)=0.
```

```
      XM(15,1,1)=0.
      XM(16,1,1)=0.
      DO 300 I=2,NEC1
      XM(1,1,I)=COEC(1,I)
      XM(2,1,I)=COEC(2,I)
      XM(3,1,I)=COEC(3,I)
      XM(4,1,I)=COEC(4,I)
      XM(5,1,I)=4.*COEC(1,I)
      XM(6,1,I)=4.*COEC(2,I)
      XM(7,1,I)=4.*COEC(3,I)
      XM(8,1,I)=4.*COEC(4,I)
      XM(9,1,I)=COEC(1,I)
      XM(10,1,I)=COEC(2,I)
      XM(11,1,I)=COEC(3,I)
      XM(12,1,I)=COEC(4,I)
      XM(13,1,I)=0.
      XM(14,1,I)=0.
      XM(15,1,I)=0.
      XM(16,1,I)=0.
300   CONTINUE
      DO 310 I=2,NEL1
      XM(1,I,1)=COEL(1,I)
      XM(2,I,1)=4.*COEL(1,I)
      XM(3,I,1)=COEL(1,I)
      XM(4,I,1)=0.
      XM(5,I,1)=COEL(2,I)
      XM(6,I,1)=4.*COEL(2,I)
      XM(7,I,1)=COEL(2,I)
      XM(8,I,1)=0.
      XM(9,I,1)=COEL(3,I)
      XM(10,I,1)=4.*COEL(3,I)
      XM(11,I,1)=COEL(3,I)
      XM(12,I,1)=0.
      XM(13,I,1)=COEL(4,I)
```

```
      XM(14,I,1)=4.*COEL(4,I)
      XM(15,I,1)=COEL(4,I)
      XM(16,I,1)=0.
310   CONTINUE
      DO 320 J=2,NEC1
      DO 320 I=2,NEL1
      XM(1,I,J)=COEC(1,J)*COEL(1,I)
      XM(2,I,J)=COEC(2,J)*COEL(1,I)
      XM(3,I,J)=COEC(3,J)*COEL(1,I)
      XM(4,I,J)=COEC(4,J)*COEL(1,I)
      XM(5,I,J)=COEC(1,J)*COEL(2,I)
      XM(6,I,J)=COEC(2,J)*COEL(2,I)
      XM(7,I,J)=COEC(3,J)*COEL(2,I)
      XM(8,I,J)=COEC(4,J)*COEL(2,I)
      XM(9,I,J)=COEC(1,J)*COEL(3,I)
      XM(10,I,J)=COEC(2,J)*COEL(3,I)
      XM(11,I,J)=COEC(3,J)*COEL(3,I)
      XM(12,I,J)=COEC(4,J)*COEL(3,I)
      XM(13,I,J)=COEC(1,J)*COEL(4,I)
      XM(14,I,J)=COEC(2,J)*COEL(4,I)
      XM(15,I,J)=COEC(3,J)*COEL(4,I)
      XM(16,I,J)=COEC(4,J)*COEL(4,I)
320   CONTINUE
      RETURN
      END
```

- **COEFF2S( )**

  Ce sous programme calcule les pixels de sortie.

  **Paramètres :**

  [IN N1] dimension de l'image d'entrée

  [IN NEL1] nombre de pixels de sortie dans chaque direction est IE : NEL1 = IE+1

  [IN M1] dimension de l'image de sortie

[IN COEL] tableau de coefficients suivant x dimension : (4, NEL1)

[IN COEC] tableau de coefficients suivant y de dimension : (4, NEL1)

[IN XM] Tableau des coefficients du filtre

[IN G] image d'entrée

[IN C] matrice C

[IN S] image de sortie

**Syntaxe en Fortran:**

```
COEFF2S(NL, NEL1, ML, XM, G, C, S)
INTEGER NL, NEL1
DOUBLE PRECISION G, S, XM
```

**Code Fortran :**

```
SUBROUTINE COEFF2(NL, NEL1, ML, XM, G, C, S)
DOUBLE PRECISION G(NL,NL)
DOUBLE PRECISION C(NL,NL), S(ML,ML)
DOUBLE PRECISION XM(16,NEL1,NEL1)
NEC1=NEL1
NC=NL
NL0=NL-1
NL00=NL-2
NC0=NC-1
NC00=NC-2
ML=(NL-3)*NEL1+1
MC=(NC-3)*NEC1+1
DO 20 JC=2,NC-2
JC00=JC-2
JC0=JC-1
JC1=JC+1
JC2=JC+2
J=JC00*NEC1
J1=J+1
DO 25 IL=2,NL-2
```

```
      IL00=IL-2
      IL0=IL-1
      IL1=IL+1
      IL2=IL+2
      I=IL00*NEL1
      I1=I+1
      S(I1,J1)=G(IL,JC)
      DO 30 IA=2,NEL1
      IIA=I+IA
      S(IIA,J1)=XM(1,IA,1)*C(IL0,JC0)+XM(2,IA,1)*C(IL0,JC)
      S(IIA,J1)=S(IIA,J1)+XM(3,IA,1)*C(IL0,JC1)+XM(5,IA,1)*C(IL,JC0)
      S(IIA,J1)=S(IIA,J1)+XM(6,IA,1)*C(IL,JC)+XM(7,IA,1)*C(IL,JC1)
      S(IIA,J1)=S(IIA,J1)+XM(9,IA,1)*C(IL1,JC0)+XM(10,IA,1)*C(IL1,JC)
      S(IIA,J1)=S(IIA,J1)+XM(11,IA,1)*C(IL1,JC1)+XM(13,IA,1)*C(IL2,JC0)
      S(IIA,J1)=S(IIA,J1)+XM(14,IA,1)*C(IL2,JC)+XM(15,IA,1)*C(IL2,JC1)
30    CONTINUE
      DO 40 IB=2,NEC1
      JIB=J+IB
      S(I1,JIB)=XM(1,1,IB)*C(IL0,JC0)+XM(2,1,IB)*C(IL0,JC)
      S(I1,JIB)=S(I1,JIB)+XM(3,1,IB)*C(IL0,JC1)+XM(4,1,IB)*C(IL0,JC2)
      S(I1,JIB)=S(I1,JIB)+XM(5,1,IB)*C(IL,JC0)+XM(6,1,IB)*C(IL,JC)
      S(I1,JIB)=S(I1,JIB)+XM(7,1,IB)*C(IL,JC1)+XM(8,1,IB)*C(IL,JC2)
      S(I1,JIB)=S(I1,JIB)+XM(9,1,IB)*C(IL1,JC0)+XM(10,1,IB)*C(IL1,JC)
      S(I1,JIB)=S(I1,JIB)+XM(11,1,IB)*C(IL1,JC1)+XM(12,1,IB)*C(IL1,JC2)
40    CONTINUE
      DO 45 IB=2,NEC1
      JS=J+IB
      DO 45 IA=2,NEL1
      IS=I+IA
      S(IS,JS)=XM(1,IA,IB)*C(IL0,JC0)+XM(2,IA,IB)*C(IL0,JC)
      S(IS,JS)=S(IS,JS)+XM(3,IA,IB)*C(IL0,JC1)+XM(4,IA,IB)*C(IL0,JC2)
      S(IS,JS)=S(IS,JS)+XM(5,IA,IB)*C(IL,JC0)+XM(6,IA,IB)*C(IL,JC)
      S(IS,JS)=S(IS,JS)+XM(7,IA,IB)*C(IL,JC1)+XM(8,IA,IB)*C(IL,JC2)
      S(IS,JS)=S(IS,JS)+XM(9,IA,IB)*C(IL1,JC0)+XM(10,IA,IB)*C(IL1,JC)
```

```
      S(IS,JS)=S(IS,JS)+XM(11,IA,IB)*C(IL1,JC1)+XM(12,IA,IB)*C(IL1,JC2)
      S(IS,JS)=S(IS,JS)+XM(13,IA,IB)*C(IL2,JC0)+XM(14,IA,IB)*C(IL2,JC)
      S(IS,JS)=S(IS,JS)+XM(15,IA,IB)*C(IL2,JC1)+XM(16,IA,IB)*C(IL2,JC2)
   45 CONTINUE
   25 CONTINUE
      S(ML,J1)=G(NL0,JC)
      DO 80 IB=2,NEC1
      JIB=J+IB
      S(ML,JIB)=XM(1,1,IB)*C(NL00,JC0)+XM(2,1,IB)*C(NL00,JC)
      S(ML,JIB)=S(ML,JIB)+XM(3,1,IB)*C(NL00,JC1)+XM(4,1,IB)*C(NL00,JC2)
      S(ML,JIB)=S(ML,JIB)+XM(5,1,IB)*C(NL0,JC0)+XM(6,1,IB)*C(NL0,JC)
      S(ML,JIB)=S(ML,JIB)+XM(7,1,IB)*C(NL0,JC1)+XM(8,1,IB)*C(NL0,JC2)
      S(ML,JIB)=S(ML,JIB)+XM(9,1,IB)*C(NL,JC0)+XM(10,1,IB)*C(NL,JC)
      S(ML,JIB)=S(ML,JIB)+XM(11,1,IB)*C(NL,JC1)+XM(12,1,IB)*C(NL,JC2)
   80 CONTINUE
   20 CONTINUE
      DO 60 IL=2,NL00
      IL00=IL-2
      IL0=IL-1
      IL1=IL+1
      IL2=IL+2
      I=IL00*NEL1
      I1=I+1
      S(I1,MC)=G(IL,NC0)
      DO 60 IA=2,NEL1
      IIA=I+IA
      S(IIA,MC)=XM(1,IA,1)*C(IL0,NC00)+XM(2,IA,1)*C(IL0,NC0)
      S(IIA,MC)=S(IIA,MC)+XM(3,IA,1)*C(IL0,NC)+XM(5,IA,1)*C(IL,NC00)
      S(IIA,MC)=S(IIA,MC)+XM(6,IA,1)*C(IL,NC0)+XM(7,IA,1)*C(IL,NC)
      S(IIA,MC)=S(IIA,MC)+XM(9,IA,1)*C(IL1,NC00)+XM(10,IA,1)*C(IL1,NC0)
      S(IIA,MC)=S(IIA,MC)+XM(11,IA,1)*C(IL1,NC)+XM(13,IA,1)*C(IL2,NC00)
      S(IIA,MC)=S(IIA,MC)+XM(14,IA,1)*C(IL2,NC0)+XM(15,IA,1)*C(IL2,NC)
   60 CONTINUE
      S(ML,MC)=G(NL0,NC0)
```

```
RETURN
END
```

### 6.2.3 Codes parallèles du réseau linéaire

Nous avons adapté le sous programme EXCHNG1. Le sous programme MPI_CART_SHIFT détermine le réseau linéaire en définissant les processeurs voisins et le sous programme MPE_DECOMP1D permet à chaque processeur de reconnaître les limites de la fenêtre d'entrée.

- **EXCHNG1( )**

  On a adapté ce programme pour les communications des trois lignes du filtre des 16 pixels.

  **Paramètres :**

  [IN A] donnée d'entrée à échanger
  [IN NX] dimension de l'image d'entrée suivant y
  [IN S] limite inférieur des données d'entrée
  [IN E] limite supérieur des donnée d'entrée
  [IN COMM1D] communicateur
  [IN NBRBOTTOM] processeur voisin du bas dans le communicateur
  [IN NBRTOP] processeur voisin du haut dans le communicateur

  **Syntaxe en Fortran :**

```
EXCHNG1( A, NX, S, E, COMM1D, NBRBOTTOM, NBRTOP )
INTEGER NX, S, E
DOUBLE PRECISION A
INTEGER COMM1D, NBRBOTTOM, NBRTOP
```

  **Code Fortran :**

```
SUBROUTINE EXCHNG1( A, NX, S, E, COMM1D, NBRBOTTOM, NBRTOP )
INCLUDE "MPIF.H"
```

```
      INTEGER NX, S, E
      DOUBLE PRECISION A(1:NX,S:E+3)
      INTEGER COMM1D, NBRBOTTOM, NBRTOP
      INTEGER STATUS(MPI_STATUS_SIZE), IERR
      CALL MPI_SENDRECV( A(1,S), 3*NX, MPI_DOUBLE_PRECISION,
     $           NBRBOTTOM, 1,
     $           A(1,E+1), 3*NX, MPI_DOUBLE_PRECISION,
     $           NBRTOP, 1, COMM1D, STATUS, IERR )
      RETURN
      END
```

### 6.2.4 Codes parallèles de la grille 2D

Nous avons adapté le sous programme EXCHNG2. Le sous programme FND2DNBRS détermine les processeurs voisins de la grille 2D et le sous programme FND2DDECOMP permet à chaque processeur de reconnaître les limites de la fenêtre d'entrée.

- **EXCHNG2( )**

  Nous avons adapté ce code pour les communications des 3 lignes horizontales, des 3 colonnes verticales et des 9 valeurs selon la diagonale du filtre des 16 pixels dans le cas d'une grille 2D.

  **Paramètres :**

  [IN A] donnée d'entrée à échanger
  [IN SX] limite inférieur de la fenêtre d'entrée suivant x du processeur
  [IN EX] limite supérieur de la fenêtre d'entrée suivant x du processeur
  [IN SY] limite inférieur de la fenêtre d'entrée suivant y du processeur
  [IN EY] limite supérieur de la fenêtre d'entrée suivant y du processeur
  [IN COMM2D] communicateur
  [IN STRIDE0YPE] type vecteur
  [IN NBRLEFT] processeur voisin du gauche dans le communicateur
  [IN NBRRIGHT] processeur voisin de droite dans le communicateur
  [IN NBRBOTTOM] processeur voisin du bas dans le communicateur
  [IN NBRTOP] processeur voisin du haut dans le communicateur

[IN NBRDIAGLEFT] processeur voisin dans la diagonal à gauche

[IN NBRDIAGRIGHT] processeur voisin dans la diagonal à droite

**Syntaxe en Fortran :**

```
EXCHNG2( A, SX, EX, SY, EY,
$              COMM2D, STRIDE0TYPE,
$              NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM,
$              NBRDIAGLEFT, NBRDIAGRIGHT )

INTEGER SX, EX, SY, EY, STRIDE0TYPE
DOUBLE PRECISION A(SX:EX+3,SY:EY+3)
INTEGER COMM2D, NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM
INTEGER NBRDIAGLEFT, NBRDIAGRIGHT
```

**Code Fortran :**

```
SUBROUTINE EXCHNG2( A, SX, EX, SY, EY,
$              COMM2D, STRIDE0TYPE,
$              NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM,
$              NBRDIAGLEFT, NBRDIAGRIGHT )
INCLUDE "MPIF.H"
INTEGER SX, EX, SY, EY, STRIDE0TYPE
DOUBLE PRECISION A(SX:EX+3,SY:EY+3)
INTEGER COMM2D, NBRLEFT, NBRRIGHT, NBRTOP, NBRBOTTOM
INTEGER NBRDIAGLEFT, NBRDIAGRIGHT
INTEGER STATUS(MPI_STATUS_SIZE), IERR, NNX
NNX = EX - SX +1
CALL MPI_SENDRECV( A(SX,SY), NNX, MPI_DOUBLE_PRECISION,
$              NBRBOTTOM, 0,
$              A(SX,EY+1), NNX, MPI_DOUBLE_PRECISION,
$              NBRTOP, 0, COMM2D, STATUS, IERR )
CALL MPI_SENDRECV( A(SX,SY+1), NNX, MPI_DOUBLE_PRECISION,
$              NBRBOTTOM, 0,
$              A(SX,EY+2), NNX, MPI_DOUBLE_PRECISION,
```

```
     $          NBRTOP, 0, COMM2D, STATUS, IERR )
      CALL MPI_SENDRECV( A(SX,SY+2), NNX, MPI_DOUBLE_PRECISION,
     $          NBRBOTTOM, 0,
     $          A(SX,EY+3), NNX, MPI_DOUBLE_PRECISION,
     $          NBRTOP, 0, COMM2D, STATUS, IERR )
      CALL MPI_SENDRECV( A(SX,SY), 1, STRIDE0TYPE, NBRLEFT, 1,
     $          A(EX+1,SY), 1, STRIDE0TYPE, NBRRIGHT, 1,
     $          COMM2D, STATUS, IERR )
      CALL MPI_SENDRECV( A(SX,SY), 3, MPI_DOUBLE_PRECISION,
     $          NBRDIAGLEFT, 2,
     $          A(EX+1,EY+1), 3, MPI_DOUBLE_PRECISION,
     $          NBRDIAGRIGHT, 2,
     $          COMM2D, STATUS, IERR )
      CALL MPI_SENDRECV( A(SX,SY+1), 3, MPI_DOUBLE_PRECISION,
     $          NBRDIAGLEFT, 4,
     $          A(EX+1,EY+2), 3, MPI_DOUBLE_PRECISION,
     $          NBRDIAGRIGHT, 4,
     $          COMM2D, STATUS, IERR )
      CALL MPI_SENDRECV( A(SX,SY+2), 3, MPI_DOUBLE_PRECISION,
     $          NBRDIAGLEFT, 5,
     $          A(EX+1,EY+3), 3, MPI_DOUBLE_PRECISION,
     $          NBRDIAGRIGHT, 5,
     $          COMM2D, STATUS, IERR )
      RETURN
      END
```

# Chapitre 7

# Conclusion générale

Cette étude se situe dans le contexte de l'évolution des technologies liées au traitement numériques d'images. Nous avons présenté, dans ce travail, comment, lors du ré-échantillonnage d'une image numérique, la principale partie des calculs pouvait être facilement et efficacement parallélisée sur un faible nombre de processeurs.

Pour en fixer le cadre, nous avons rappelé les fondements de base de l'échantillonnage et les techniques classiques de reconstruction d'images numériques. Nous avons présenté les machines parallèles, les bibliothèques de calcul et de communication les plus utilisées dans les grandes applications scientifiques.

Dans le cas du ré-échantillonnage d'images numériques nous avons proposé une résolution séquentielle d'un système linéaire par FFT 2D et par une factorisation de Cholesky adaptée, pour calculer les coefficients de la matrice C. Nous avons déterminé les filtres des 4 et 16 pixels. Le calcul des complexités et les essais numériques ont montré que le coût de l'algorithme est $O(n^2)$ et que les coefficients sont obtenus avec un temps nettement inférieur à celui du filtre de ré-échantillonnage. La part de calcul induite par le calcul des coefficients dans l'ensemble des calculs est faible dès que le nombre de pixels de sortie est différent de un ($\approx 3$ % pour 4 pixels de sortie).

Dans le cas des images d'entrée de grandes dimensions, nous avons montré que la principale partie des calculs pouvait être facilement et efficacement parallélisée sur un faible nombre de processeurs. Nous avons proposé un calcul parallèle du ré-échantillonnage d'images numériques par les filtres des 4 pixels et 16 pixels de la bilinéaire et de la B-spline cubique uniforme. Nous avons implémenté nos codes sur un réseau linéaire et sur une grille de processeurs. Pour réaliser les tests, nous avons utilisé une grappe de 8 nœuds bi-processeurs (processeurs Pentium II fonctionnant à 450 Mhz) couplés par un réseau de 100Mbits/s. Nous avons utilisé la bibliothèque de communication Message Passing Interface (MPI) avec le mode SPMD (Single Program Multiple Data). Les essais numériques ont montré que dans le cas de la B-spline cubique uniforme, cette approche parallèle a permis de réduire et d'obtenir des temps de calcul du filtre inférieurs à celui de la partie séquentielle restante. En plus de la parallélisation des calculs, on bénéficie de la meilleure gestion de la mémoire qui apporte des efficacités supérieures à l'unité.

Pour améliorer l'efficacité, il faudrait maintenant paralléliser la partie séquentielle. Dans l'algorithme de résolution du système linéaire, la FFT 2D est appliquée à l'image d'entée G pour obtenir la matrice Z et à la matrice B pour déterminer la matrice des coefficients C. Ces deux parties peuvent facilement être parallélisées en utilisant la FFT 2D parallèle [MPI93]. Un calcul parallèle peut être utilisé pour déterminer la matrice B (voir figure 41).

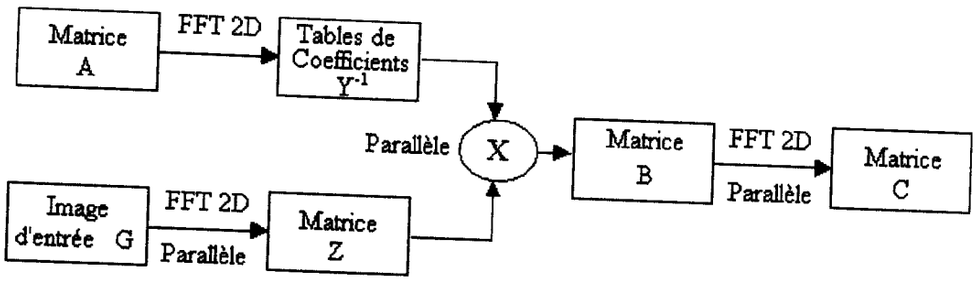



**Figure 41 :** *Calcul parallèle de la matrice C.*

La FFT 2D implémentée en Hardware peut améliorer le temps de calcul des coefficients de la matrice C.

Un calcul parallèle de la factorisation de Cholesky adaptée ainsi qu'une décomposition de domaine du système linéaire peuvent être envisagés.

Pour améliorer le temps de calcul de la partie séquentielle restante, ces trois techniques feront l'objet de futures recherches.

# BIBLIOGRAPHIE

[AP76] H.C Andrews and C.L Patterson II, *Digital interpolation of discrete images*, IEEE Trans. comput., vol. C-25 , pp. 196-202, February 1976 .

[Bai87] D. H. Bailey, *A High-Performance Fast Fourier Transform Algorithm for the Cray-2*, Journal of Supercomputing, vol.1, no. 1, 43-60, July 1987.

[BDJMS94] A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM3 User's Guide and Reference Manual*, Technical report, Oak Ridge National Laboratory, 1994.

[Boo78] C. de Boor, *A Practical Guide to splines*, Springer-Verlag, 1978.

[CM-87] Thinking Machine Corporation, *CM-2, Technical summary*, 1987.

[CM-93] Thinking Machine Corporation, *CM-5, Technical summary*, 1993.

[CDPW92] J. CHOI, J. Dongarra, R. Pozo, and D. Walker. ScaLAPACK: A Scalable Linear Algebra Library for Distributed Memory Concurrent Computers. Technical Report LAPACK WN 55, University of Tennessee, 1992.

[Cox72] M. G. Cox, *The Numerical Evaluation of B-spline*, J. Inst. Maths Applics, 10, pp. 134-149, 1972.

[Cra94] Inc. Cray Research, *Cray T3D, Technical summary*, 1994.

[CS92] K. L. Chung, L. J. Shen, *Vectorized Algorithm for B-Spline Curve Fitting on CRAY X-MP EA/16se*, IEEE Computer Society Press, pp. 166-169, 1992.

[CS99] D. E. Culler and J. P. Singh, Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann, San Francisco, CA, 1999.

[CT65] C. W. Cooley and J. W. Tuckey, *An algorithm for the machine calculation of complex fourier series*, Math. Comput., 19, 297-301,1965.

[CWWZW95] F. Cheng, G. W. Wasilkowski, J. Wang, C Zhang, W. Wang, *Parallel B-Spline Surface Interpolation on a Mesh-Connected Processor Array*, J. of Par. And Distr. Compu., 24, pp. 224-229, 1995.

[Des79] A. M. Despain, *Very Fast Fourier Transform Algorithms Hardware for implementation*, IEEE Transaction on computers, vol. C-28, no. 5, May 1979.

[DM97] M. D'Apuzzo and L. Maddalena, *A Parallel Algorithm for Parametric Cubic B-Spline Curves Interpolation*, Neural, Parallel & Scientific Computations, Dynamic Publishers Inc. USA, ISSN 1061 5396, 5, 1997, pp. 201-220.

[DMBS88] J.J Dongarra, C.B Moler, J.R. Bunch, and G.W. Stewart. Linpack users guide. SIAM Review, 1988.

[Eti94] D. Etiemble, *L'évolution des machines parallèles et massivement parallèles*, La Lettre du Transputer, 6(4),55-60,1994.

[Exe94] Convex Examplar Salable Parallel Processing, *Exemplar, System Overview*, 1994.

[FLN89] Z. Fang, X. Li, and L. M. Ni, *On the communication complexity of generalized 2-D convolution on array processors*, IEEE Trans. Comput., vol. 38, pp. 184-195, February 1989.

[Fly79] M. J. Flynn, *Some computer organization and their effectiveness*, IEEE Transactions on Computer, 948-960, 1979.

[GBDWMS94] A. Geist, A. Beguelin, J. Dongarra, B. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Parallel Computing*, MIT Press, 1994.

[GGKMRS83] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. Mcauliffe, L. Rudolph, and M. Snir, *The NYU Ultracomputer-Designing and MIMD Shared Memory Parallel Computer*, IEEE Transactions on Computers, vol. C-31, no. 2, pp. 175-189, February 1983.

[GLS94] W. Gropp, E. Lusk, and A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, Cambridge, MA, MIT Press, 1994.

[Gus88] J. L. Gustafson, *Reevaluating amdahl's law*, Communication of the ACM, 31, 1988.

[GV96] G.H. Golub et C. F. Van Loan. *Matrix Computations*, 3ème édition, John Hopkins, 1996.

[HA78] H. S. Hou and H. C. Andrews, *Cubic Splines for Image Interpolation and Digital Filtering,*, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 26, pp. 508-517, December 1978.

[HPF93] HPF Forum, High Performance Fortran language specification, version 1.0, 1993.

[Hwa93] K. Hwang, *Advanced Computer Architecture : Parallelism, Scalability, Programmability*, Mc GrawHill, International Edition 1993.

[JPH97] H. A. James, C. J. Patten and K. A. Hawick, Stencil Methods on distributed high performance computers, Technical note DHPC-010, department of computer science, University of Adelaide, Australia.

[KÖD03] A. Kaya, D. Öktem, O. Dönmez, *The Application Areas of Parallel Fast Fourier Transform and Future Trends*, IJCI Proceedings of International Conference on Signal Processing, vol. 1, n°2, September 2003.

[Lau96] C. Laurent, *Adéquation Algorithmiques et Architectures parallèles pour la Reconstruction 3D en tomographie*, Thèse, Université Claude Bernard Lyon 1, Janvier 1996.

[Mah95] M. Mahboub, *Application de la fonction B-spline cubique uniforme au recalage en résolution spatiale d'images satellites (Landsat-Spot)*, MCEA Grenoble, pp. 183-188, Septembre 1995.

[MP02] M. Mahboub, and B. Philippe, *Ré-échantillonnage d'images numériques par la B-spline cubique uniforme*, CARI'02 Yaoundé, pp. 301-308, Octobre 2002.

[MPB04] M. Mahboub, B. Philippe et B. Benyoucef, *Calcul des Coefficients et du Filtre de Ré-échantillonnage d'Images Numériques de la B-spline Cubique Uniforme*, revue Sciences & Technologie B de l'Université de Constantine, no. 21B, Juin 2004.

[MPI93] MPI Forum, *Message Passing Interface, Document for a stadard message-passing interface*, 1993.

[Nels00] T. Nelson, *Introduction to MPI,* NAS Scientific Consulting, (650) 604-4292.

[Pac98] P. S. Pacheco, *A User's Guide to MPI*, Department of Mathematics, University of San Francisco, March 1998.

[Par91] Intel Corporation, *Produit Overview*, 1991.

[Pin97a] C. K. Pink, *Evaluating B-spline Approximations in parallel in One and Two Dimensions,* Technical Report 9705, University of Huddersfield, February 1997 .

[Pin97b] C. K. Pink, *Parallel B-spline Approximation of Large Sets of Uniform Data,* Technical Report 9706, University of Huddersfield, January 1997.

[PAM98] C. K. Pink, I. J. Anderson, and J. C. Mason, *A Parallel Domain Decomposition Method for Spline Approximation*, Ninth International Conference on Domain Decomposition Methods (DDM.org), 1998.

[Pra78] William K. Pratt , Digital image processing, A Wiley interscience publication 1978.

[RK82] A. Rosenfield and A. C. Kak, Digital Picture Processing, Academic Press Inc., 2 ed., 1982.

[SP193] IBM, IBM SP1, product description, 1993.

[SS84] R. R. Seban, and H. J. Siegel, *Shuffling with the Illiac and PM2I SIMD Networks*, IEEE Transaction on Computers, vol. C-33, no. 7, pp. 619-625, July 1984.

[SSF82] L. J. Siegel, H. J. Siegel, and A. E. Feather, *Parallel processing approaches to image correlation,* IEEE Trans. Comput., vol. C-31, pp. 208-218, March 1982.

[Swa82] P. N. Swarztrauber, *Vectorizing theFFTs*, Parallel Computation, 51-83, 1982.

[TL89] R. Tolimieri and C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*, Springer-Verlag, New York, N. Y. 1989.

[UAE91] M. Unser, A. Aldroudi, M. Eden, *Fast B-spline transforms for continus image representation and interpolation*, IEEE Trans. Pattern Anal. Mach. Intell. Vol. 13, pp. 277-285, March 1991.

[VV01] B. Vrcelj, P.P. Vaidyanathan, *Efficient implementation of all-digital interpolation*, EDICS number: 2-INTR, pp. 1-16, July 2001.

[VVPL02] D. Van De Ville, R. Van De Ville,W. Philips, I. Lemahieu, *Image Resampling Between Orthogonal and Hexagonal Lattices,* IEEE ICIP 2002.

[WS82] M. R. Warpenburg, and L. J. Siegel, *SIMD image resampling*, IEEE Transactions on Computers, vol. C-31, pp. 934-942, October 1982.

# Annexe Publications

1. M. Mahboub , B. Philippe et B. Benyoucef, *Calcul des coefficients et du filtre de ré-échantillonnage d'images numériques de la B-spline cubique uniforme,* Sciences & Technologie B de l'Université Mentouri Constantine, 21B, Juin2004.

**Date de réception :** 16/ 02/ 2002

**Date d'acceptation :** 16/ 02/ 2004

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة منتوري - قسنطينة

مديرية النشر والتنشيط العلمي

قسنطينة، في : .....

الرقم : 23/04.....

# ATTESTATION

L'article référencé comme suit:
**N° de code**: ST15/02/E
**Date de soumission:** 16/02/2002
**Date d'acceptation:** 10/02/2004
**Intitulé:** Calcul des coefficients et du filtre de re-echantillonnage d'images numeriques de la b-spline cubique uniforme.
**Auteurs:** M. MAHBOUB[1], B. PHILIPPE[2], B. BENYOUCEF[1]
**Institutions:**
[1]Département de Physique, Faculté des Sciences, Université Abou Bakr Belkaid, Tlemcen Algérie.
[2]IRISA –INRIA, Rennes, France.

a été évalué positivement et est accepté pour paraître dans la revue ***SCIENCES & TECHNOLOGIE*** de l'Université Mentouri de Constantine (N°21B, juin 2004).

Constantine, le 10/04/2004

Le Directeur

20

الدكتور خير الدين م. الهاشمي
مدير النشر و التنشيط العلمي

طريق عين الباي 25000 قسنطينة - الجزائر
الهاتف / الفاكس : 213 (0) 31.81.87.02
E.mail: revuesc@wissal.dz

# Sciences & Technologie

Revue semestrielle de l'université Mentouri Constantine-Algérie

ISSN-1111-5041

N°16-décembre 2001

# COMITE SCIENTIFIQUE

## Section A: MATHEMATIQUES, INFORMATIQUE

D. AISSANI — *Professeur, Institut de Mathématiques Université de Béjaïa (Algérie)*

N.M. BENKAFADAR — *Professeur, Département de Mathématiques Université Mentouri Constantine (Algérie)*

A. BOUCHERIF — *Professeur, Département de Mathématiques Univeristé de Tlemcen (Algérie)*

M. DENCHE — *Professeur, Département de Mathématiques Université Mentouri Constantine (Algérie)*

H. HUDZIK — *Professeur, Faculté de Mathématiques et informatique Université Adam Mickiewicz (Pologne)*

Z. SAHNOUN — *Professeur, Département d'Informatique Université Mentouri Constantine (Algérie)*

M. ZITOUNI — *Professeur,Département de Mathématiques Université de Boumerdès (Algérie)*

F. REBBANI — *Professeur, Département de Mathématiques Université d'Annaba (Algérie)*

## Section B: PHYSIQUE, CHIMIE

M.-S. AIDA — *Professeur, Unité de Recherche Matériaux Université Mentouri Constantine (Algérie)*

V.M. ANDRONOV — *Professeur, Laboratoire de Physique Expérimentale Université de Kharkov (Ukraine)*

B. BENYOUCEF — *Professeur, Département de Physique Université de Tlemcen (Algérie)*

S.-E. BOUAOUD — *Professeur, Institut de Chimie Université Mentouri Constantine (Algérie)*

L. CHETOUANI — *Professeur, Département de Physique Université Mentouri Constantine (Algérie)*

R. HALIMI — *Professeur, Unité de Recherche Matériaux Université Mentouri Constantine (Algérie)*

D. HAMANA — *Professeur, Unité de Recherche Matériaux Université Mentouri Constantine (Algérie)*

N. MEBARKI — *Professeur, Département de Physique Université Mentouri Constantine (Algérie)*

L. OUAHAB — *Professeur, Laboratoire de Chimie du Solide et Inorganique Moléculaire, Université de Rennes 1(France)*

R. PENELLE — *Docteur ès-Sciences Physiques, Directeur du Laboratoire de Métallurgie Structurale, Université Paris-Sud (France)*

R. RAHOUADJ — *Maître de Conférences, UTBM, Laboratoire T2M, Sevenans 90010, Belfort (France)*

S. RHOUATI — *Professeur, Département de Chimie Université Mentouri Constantine (Algérie)*

T. SEHILI — *Professeur, Laboratoire de Photochimie et Environnement Université Mentouri Constantine (Algérie)*

L. ZOUIOUECHE — *Professeur, Laboratoire de Synthèse Asymétrique et Biocatalyse, Université d'Annaba (Algérie)*

## Section C: TECHNOLOGIE : ELECTRONIQUE, ELECTRO-TECHNIQUE, GENIE MECANIQUE, CHIMIE INDUSTRIELLE

M. ABIDAT — *Professeur, Faculté de Génie Mécanique USTBM, Oran. Algérie*

B. AGNEW — *Professor, Department of Mechanical, Materials and Manufacturing Engineering, University of Newcastle (U.K.)*

A. BARHDADI — *Professeur, Laboratoire de Physique des Semiconducteurs et de l'Energie Solaire, E.N.S. de Takaddoum, Rabat (Maroc)*

M. BARKAT — *Professeur, Institut d'Electronique Université Mentouri Constantine (Algérie)*

H. BAUDRAND — *Professeur, Laboratoire d'Electronique ENSEEIHT, I.N.P. Toulouse (France)*

A. DOGHMANE — *Professeur, Institut des Sciences Fondamentales Centre Universitaire de Skikda (Algérie)*

S. LEULMI — *Professeur, Institut d'Electrotechnique Université de Skikda (Algérie)*

A. MEDDOUR — *Professeur, Institut de Génie des Procédés Centre Universitaire de Guelma (Algérie)*

A. MOUHOUB — *Docteur ès-Sciences, Unité Développement de la Technologie du Silicium (U.D.T.S.) - Alger (Algérie)*

B. NECIB — *Professeur, Département de Génie Mécanique Université Mentouri Constantine (Algérie)*

M. NEMAMCHA — *Professeur, Institut de Génie Electrique Centre Universitaire de Guelma (Algérie)*

G. PLUVINAGE — *Directeur du Laboratoire de Fiabilité Mécanique Université de Metz (France)*

A. SAADANE — *Docteur ès-Sciences, IRESTE Université de Nantes (France)*

S. SAHLI — *Professeur, Département d'Electronique Université Mentouri Constantine (Algérie)*

A. SAID — *Professeur, Département d'Electronique Université Mentouri Constantine (Algérie)*

## Section D: BIOLOGIE, INDUSTRIES ALIMENTAIRES, SCIENCES VETERINAIRES, MEDICALES et PHARMACEUTIQUES

M. BENCHEIKH — *Docteur ès-Sciences, Laboratoire de Productions Végétales, Centre Universitaire de Chlef (Algérie)*

A. BENMAKHLOUF — *Docteur ès-Sciences, Département de Vétérinaire Université Mentouri Constantine (Algérie)*

N. BOUGUEDOURA — *Professeur, Institut de Biologie, U.S.T.H.B., Alger (Algérie)*

A. BOULAHROUF — *Docteur ès-Sciences, Département de Biologie Université Mentouri Constantine (Algérie)*

A. DJEKOUN — *Professeur, Laboratoire Amélioration des Plantes Université Mentouri Constantine (Algérie)*

S. EL JAAFARI — *Professeur, Directeur du Laboratoire de Biotechnologie et Amélioration des Plantes, Université Moulay Ismaïl, Meknès (Maroc)*

J.-M. EXBRAYAT — *Professeur à Université Catholique de Lyon (Laboratoire de Biologie Générale) et Directeur d'Etudes à l'Ecole Pratique des Hautes Etude (Laboratoire de Reproduction et Développement des Vertébrés) (France)*

M. KAID-HARCHE — *Professeur, Laboratoire de Biologie Appliquée Université d'Oran (U.S.T.O.) (Algérie)*

J.-P. LARPENT — *Professeur, Laboratoire de Microbiologie 21 Bis Rue Cotepet, 63000 Clermont-Ferrand (France)*

S. MEHENNAOUI — *Docteur ès-Sciences, Institut des Sciences Vétérinaires, Université de Batna (Algérie)*

Z. MERAIHI — *Professeur, Laboratoire de Génie Enzymatique, Université Mentouri Constantine (Algérie)*

B. PASTUSZEWSKA — *Professeur, Institut de Physiologie Animale et Nutrition, Jablonna (Pologne)*

O. RACHED — *Docteur ès-Sciences, Laboratoire d'Ecologie Université Mentouri Constantine (Algérie)*

B. SAMRAOUI — *Professeur, Laboratoire de Recherche Zones Humides, Université d'Annaba (Algérie)*

N. SIDI MANSOUR — *Professeur, Clinique Hématologie C.H.U. de Constantine (Algérie)*

T. LAMAZE — *Professeur, Centre d'Etude Spatiale de la Biosphère (CESBIO) Toulouse (France)*

## Section E: SCIENCES DE LA TERRE, GEOPHYSIQUE, GENIE CIVIL, HYDRAULIQUE, AMENAGEMENT DU TERRITOIRE, ARCHITECTURE ET URBANISME

B. ATTAF — *Docteur ès-Sciences, Département de Génie Civil Université de Blida (Algérie)*

M.T. BENAZZOUZ — *Docteur ès-Sciences, Labotatoire de Géomorphologie Université Mentouri Constantine (Algérie)*

M. BOUMAZA — *Professeur, Département de Génie Civil Université Mentouri Constantine (Algérie)*

S.E. CHERRAD — *Professeur, Département des Sciences de la Terre Université Mentouri Constantine (Algérie)*

P. CLASTRES — *Professeur, INSA de Toulouse, Laboratoire des Matériaux et Durabilité des Constructions (France)*

M. COTE — *Professeur, Institut de Géographie Université de Provence, Aix-Marseille I (France)*

M. DJEDDI — *Professeur, Département des Matériaux de Construction Université de Boumerdes, Algérie*

W. HAGER — *Professeur, Laboratoire d'Hydraulique, d'Hydrologie et de Glaciologie Ecole Polytechnique Fédérale de Zurich (Suisse)*

A. KETTAB — *Professeur, Département d'Hydraulique Ecole Nationale Polytechnique, Alger (Algérie)*

M.-E. LAROUK — *Professeur, Département des Sciences de la Terre Université Mentouri Constantine (Algérie)*

M.-E. SAMAI — *Professeur, Département de Génie Civil Université Mentouri Constantine (Algérie)*

M.-S. ZEROUALA — *Professeur, Département d'Architecture et d'Urbanisme Université Mentouri Constantine (Algérie)*

# INSTRUCTIONS AUX AUTEURS

## I- Généralités

La revue **Sciences & Technologie** publie des articles scientifiques originaux dans trois langues: arabe, français et anglais. Les résumés doivent être fournis dans ces trois langues **et ne doivent pas dépasser 150 mots**. C'est une revue pluridisciplinaire qui comportent cinq sections:

**Section A:** Mathématique, Informatique.
**Section B:** Physique, Chimie.
**Section C:** Technologie: Electronique, Electrotechnique, Génie Mécanique, Chimie Industrielle.
**Section D:** Biologie, Industrie Alimentaire, Sciences Vétérinaires, Médicales et Pharmaceutiques.
**Section E:** Sciences de la Terre, Géophysique, Génie Civil, Hydraulique, Aménagement du Territoire, Architecture et Urbanisme.

## II- Manuscrits

Les articles soumis à la publication (trois exemplaires) ne doivent pas dépasser 20 pages dactylographiées (tableaux, figures, graphiques, bibliographie,... compris) avec une large marge à gauche (3 cm), imprimé sur papier de format 21 x 29,7 cm (A4) avec interligne de bonne lisibilité. Une certaine flexibilité est permise aux auteurs, mais ils doivent organiser le texte clairement en sections telles que: Introduction, Détails expérimentaux, Résultats, Discussion et Conclusion. Les articles plus longs seront publiés par partie dans des numéros successifs, chaque partie étant déterminée par les auteurs. Il est demandé eu outre aux auteurs de bien vouloir accompagner le résumé (dans les trois langues, anglais, français, arabe) de leurs articles de mots clés les plus complets possibles (Pour les articles de mathématiques, indiquer la classification AMS).

Dans le souci de gain de temps et de respect des échéances de publication, il est recommandé aux auteurs de prendre en charge la saisie complète de leur article sur micro-ordinateur, et de le transmettre à la revue, après qu'ils aient été avisés de l'acceptation pour publication, sous forme de fichiers sur disquette, lesquels seront recopiées par les soins du service.

Toutefois, étant donné que la mise en forme finale de l'article est réalisée par P.A.O. (Publication Assistée par Ordinateur), il est demandé aux auteurs d'éviter tout formatage de leur texte. Aussi faudra-t-il éviter de le styliser.

## III- Bibliographie

Les références bibliographiques citées dans le texte doivent ne comporter que le N° de la référence entre crochets (ex.: [5]). Si le nom de l'auteur apparaît dans le texte, il doit être suivi par le N° de la référence. Lorsque la référence comporte plus de deux auteurs, seul le premier est cité, suivi de "et al".

Pour les **articles**, la référence complète comporte les noms des auteurs suivis des initiales de leurs prénoms,, le titre de l'article, le titre du périodique (en se conformant aux abréviations admises), le volume, le N° du périodique, l'année de publication et les pages concernées. *Ex.*: Untel B., "Cadmium toxicity", *Plant Physiol.*, Vol. 10, N°3 (1993), pp. 127-131.

Pour les **ouvrages**, la référence doit comporter les noms des auteurs suivis des initiales de leurs prénoms, le titre complet de l'ouvrage, le volume, le tome, la première et la dernière page se rapportant aux résultats discutés, le numéro de l'édition s'il y en a plusieurs, le nom de l'éditeur, le lieu et l'année d'édition.

Pour les **rencontres scientifiques** (congrès, proceedings,...), la référence comporte les noms des auteurs suivis des initiales de leurs prénoms, le titre de la communication, l'identification de la rencontre, le lieu, la période et les pages concernées.

## IV- Iconographie

Les tableaux, planches, graphiques, cartes, photographies, etc. doivent être fournis à part, en hors-texte. Ils doivent être présentés sur feuilles blanches de format A4, individuellement ou en groupe, et comporter en dessous, la mention "tableau" ou "figure" affectée d'un numéro.

Les illustrations et les figures doivent être claires, faites professionnellement et adéquates pour la reproduction: une réduction éventuelle de 50% doit conduire à une taille et une épaisseur des caractères convenables pour une bonne lisibilité. Par ailleurs, pour les figures réalisées sur ordinateur, afin que le contraste soit maximal, l'usage d'une imprimante laser ou à jet d'encre est indispensable.

Les légendes affectées de leurs numéros doivent être regroupées dans une page à part.

La présentation finale de l'article sera laissée à l'appréciation du comité de rédaction.

# CALCUL DES COEFFICIENTS ET DU FILTRE DE RE-ECHANTILLONNAGE D'IMAGES NUMERIQUES DE LA B-SPLINE CUBIQUE UNIFORME

*Reçu le 16/02/2002 – Accepté le 10/02/2004*

**Résumé**

Dans le cas du ré-échantillonnage d'images numériques par la B-spline cubique uniforme, un pixel de sortie est calculé par un filtre appliqué sur 16 pixels voisins (pv) d'une image ou d'une matrice de coefficients C. Pour obtenir cette matrice C, nous devons résoudre un système d'équations linéaires. Nous résolvons ce système linéaire par une factorisation de Cholesky adaptée. Cette technique nous a permis de calculer la matrice C avec un temps de calcul inférieur à celui du filtre des 16 pixels voisins. Nous présentons, dans cet article, l'algorithme de résolution, le filtre de ré-échantillonnage, les complexités et les temps de calcul pour son implémentation. Le calcul des complexités et les essais numériques ont montré que le coût de l'algorithme est $O(n^2)$.

***Mots clés:*** *Ré-échantillonnage d'images, B-spline cubique uniforme, interpolation, reconstruction, factorisation de Cholesky.*

**Abstract**

For digital image resampling by uniform cubic B-spline, an output pixel is computed from a filter applied to 16 neighboring pixels (np) of the transformed of an original image or a matrix C of coefficients. To obtain this matrix, we must solve a system of linear equations. We propose, in this work, an adapted Cholesky Factorization for solving this linear system. This technique allows to calculate the matrix C with a computational time smaller than the computational time for 16 np filter. In this paper, we present the algorithm and we report computational times obtained from its implementation. The complexities computation and the numerical experiments proved that the algorithm cost is $O(n^2)$.

***Keywords:*** *Image resampling, uniform cubic B-spline, interpolation, reconstruction, Cholesky factorisation.*


**M. MAHBOUB[1]**
**B. PHILIPPE[2]**
**B. BENYOUCEF[1]**

[1] Département de physique
Faculté des Sciences
Université Abou Bakr Belkaid
Tlemcen, Algérie

[2] IRISA –INRIA
Rennes France

**ملخص**

في حال إعادة معايرة الصور العددية بواسطة الدوال من نوع "B-spline" المكعّبة المنتظمة، فإنه يتم حساب أي عنصر صوري في حال الخروج بتطبيق مرشحة على 16 عنصرا صوريا متجاورا في هيئة صورة أو مصفوفة معاملات "C". للحصول على هذه المصفوفة يجب علينا حل جملة معادلات خطية باعتماد تحليل كولاسكي. لقد سمحت لنا هذه التقنية بحساب المصفوفة "C" في زمن حساب أقل من ذلك الذي تتطلبه مرشحات 16 عنصرا صوريا متجاورا. نقدم في هذا العمل خوارزمية الحل إضافة إلى المرشحة لإعادة المعالجة و كذا التعقيدات و أزمنة الحساب من أجل إنشائها. لقد أثبت حساب التعقيدات و التجارب العددية، أن قيمة الخوارزمية هي من النوع "$O(n^2)$".

***الكلمات المفتاحية:*** *إعادة معايرة الصور، "B-spline" مكعبة منتظمة، استكمال، إعادة التشكيل، تحليل كولاسكي.*

Dans plusieurs applications de traitement numérique d'images, il est souvent nécessaire d'effectuer des transformations géométriques non linéaires des positions des pixels de la grille de l'image comme, par exemple, dans le cas du recalage géométrique d'images. Les transformées des positions des pixels ne coïncident généralement pas avec la grille de l'image. Un algorithme de ré-échantillonnage permet d'obtenir la valeur de ces pixels intermédiaires. Le temps de calcul dépend de la complexité de la fonction d'interpolation utilisée. L'interpolation idéale, ou de Shannon, à deux dimensions (cas des images), dans le cas d'un échantillonnage régulier et uniforme avec $\Delta x$ et $\Delta y$ les pas d'échantillonnage, suivant les directions $x$ et $y$ respectivement, est définie par la relation suivante:

$$F_r(x,y)=\sum_{i=-\infty}^{+\infty}\sum_{j=-\infty}^{+\infty}F(i\Delta x,j\Delta y)R(x-i\Delta x,y-j\Delta y) \qquad (1)$$

avec:

$$R(x,y)=\frac{\sin\omega_1 x}{\omega_1 x}\frac{\sin\omega_2 y}{\omega_2 y} \qquad (2)$$

$\omega_1$ et $\omega_2$ sont les pulsations de coupures suivant les directions $x$ et $y$ respectivement, $F(i\Delta x, j\Delta y)$ est la valeur du pixel $(i, j)$ de l'image d'entrée originale à reconstruire et $F_r(x,y)$ est l'image de sortie finale. L'interpolation idéale permet de reconstruire l'image exacte, mais elle est difficile à réaliser. Plusieurs fonctions, dont la réalisation est plus ou

moins compliquée, sont utilisées pour approcher la fonction $R(x,y)$. Parmi elles, on peut citer les fonctions d'interpolation du plus proche voisin, linéaire et la B-spline cubique uniforme [1]. L'interpolation bilinéaire est utilisée, le plus souvent, dans les applications graphiques. Cette interpolation n'utilise que quatre pixels voisins de la grille de l'image pour chaque pixel de sortie. Par contre, certaines applications professionnelles utilisent les interpolations cubiques. Le ré-échantillonnage par la B-spline cubique uniforme utilise seize pixels voisins ($pv$) d'une image (ou matrice) $C$ intermédiaire, pour calculer les pixels de sortie. Pour obtenir cette matrice $C$, nous devons résoudre un système d'équations linéaires. Andrews et Patterson ont introduit en 1976 [2] une méthode de calcul par des multiplications de matrices utilisant une inversion explicite très coûteuse en $O(n^3)$ où $n$ est la dimension de la matrice. Depuis, d'autres techniques en $O(n^2)$ ont été introduites pour calculer la matrice $C$: transformation en $z$ [3, 4]; factorisation d'une matrice tridiagonale de passage $A$ mentionnée dans [4]. Dans cet article, nous considérons dans nos calculs une matrice $A$ correspondant à une image périodique que nous définissons au paragraphe 2 et qui permet de considérer une matrice carrée dans le calcul de $C$. Bien que la matrice $A$ ne soit plus bidiagonale, la complexité reste en $O(n^2)$. Cet algorithme peut aussi être appliqué à des images non périodiques. Nous proposons, dans ce travail, un algorithme de résolution de ce système linéaire par la factorisation de Cholesky [5, 6], nous présentons les filtres de ré-échantillonnage des 16 pixels voisins, les complexités correspondantes ainsi que des exemples de traitement avec des comparaisons des temps de calcul de la matrice $C$ et des filtres pour un, deux et trois pixels de sortie.

## 1. Ré-échantillonnage par les fonctions splines

L'image reconstruite $g'$ par les fonctions splines séparables dans le cas continu est exprimée par:

$$g'(x,y)=\sum_{i=1}^{n}\sum_{j=1}^{n'} c_{ij}\, s(x-i\Delta x)s(y-j\Delta y) \qquad (3)$$

$n$ et $n'$ représentent les dimensions de l'image originale d'entrée, respectivement dans les directions $x$ et $y$. L'expression matricielle dans le cas des images numériques vérifie l'expression suivante :

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

La matrice $G'$ ($M$, $M'$) avec $M>n$ et $M'>n'$ représente l'image ré-échantillonnée. Les deux matrices $B_1$ ($M$, $n$) et $B_2{}^T$ ($n'$, $M'$) effectuent respectivement un ré-échantillonnage suivant les lignes et les colonnes. Elles sont calculées respectivement par les splines de bases aux pixels de sortie. Ces matrices sont données dans le cas de la B-spline cubique uniforme dans le paragraphe 3.

La matrice $G$ ($n$, $n'$) de l'image numérique d'entrée vérifie l'équation (4); on obtient l'équation suivante:

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

Les deux matrices $A_n$ et $A_{n'}$ de l'équation (5) sont calculées respectivement par les splines de bases dans les directions $x$ et $y$ pour obtenir les valeurs des pixels de $G$. La matrice $C$ est l'inconnue; elle sera calculée par inversion de l'équation (5) pour aboutir à:

$$C = A_n{}^{-1}\, G\, A_{n'}{}^{-1} \qquad (6)$$

Dans la suite, nous nous limiterons à des images carrées, ce qui induit $n = n'$ et $M = M'$. Dans le cas d'une image rectangulaire $n \neq n'$ et $M \neq M'$, nous utilisons l'algorithme en $O(n)$ de factorisation de la matrice $A$ que nous développons au paragraphe 3 pour les deux matrices $A_n$ et $A_{n'}$. Le calcul de la matrice $C$, dans ce dernier cas, reste identique, sauf que nous devons remplacer chacune des matrices $A_n$ et $A_{n'}$ par sa propre factorisation dans les étapes correspondantes de la résolution successive des systèmes triangulaires que nous détaillons au paragraphe 4.

## 2. Cas de la B-spline cubique uniforme

La fonction de base de la B-spline cubique uniforme, $s(x)$ peut s'écrire sous la forme suivante:

$$s(x)=x_+^3-4(x-\Delta x)_+^3+6(x-2\Delta x)_+^3-4(x-3\Delta x)_+^3 \qquad (7)$$

avec

$$x_+{}^3=\begin{cases} x^3 & si\ x>0, \\ 0 & si\ x\le 0 \end{cases} \qquad (8)$$

En général, la matrice $A$ est rectangulaire et le système linéaire à résoudre est sous-déterminé. Les techniques de résolution le plus souvent utilisées sont les transformations orthogonales telles que la décomposition QR ou SVD [5].

Pour obtenir une matrice $A$ carrée, nous supposons l'image périodique définie par :

$$g(k\,n+i,\ l\,n+j) = g(i,j)\ avec\ (k,\ l) \in Z^2\ et\ (i,j) \in [1,\ n]^2,$$

Ce qui entraîne :

$$g_{n+1,\,i} = g_{1,\,i},\ g_{i,\,n+1} = g_{i,\,1},\ g_{i,\,0} = g_{i,\,n}$$

et $\quad g_{0,\,i} = g_{n,\,i}$, avec $i \in [1,\ n]$.

La matrice $A$ calculée est égale à:

$$A=\begin{bmatrix} 4 & 1 & 0 & & & \dots & & 0 & 1 \\ 1 & 4 & 1 & 0 & & \dots & & & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & & & 0 \\ : & & & & : & & & & : \\ 0 & & & \dots & & 1 & 4 & 1 & 0 \\ 0 & & & \dots & & 0 & 1 & 4 & 1 \\ 1 & 0 & & \dots & & & 0 & 1 & 4 \end{bmatrix} \qquad (9)$$

Le calcul des coefficients de la matrice $B$ est identique à celui de la matrice $A$, mais avec un pas d'échantillonnage plus faible, égal à l'inverse du nombre de pixels à déterminer. Le calcul à une dimension montre que cette matrice est formée de $n$ "blocs" dont chacun contient $M/n$ lignes et $n$ colonnes [7]:

$$B=\left[\begin{array}{cccccccccc} c_k & d_k & 0 & 0 & \dots & 0 & 0 & a_k & b_k \\ \hline b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \\ \hline a_k & b_k & c_k & d_k & \dots & 0 & 0 & 0 & 0 \\ \hline & & & & : & & & & \\ \hline b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \end{array}\right] \qquad (10)$$

avec:

$$\begin{aligned} a_k &= (\beta_k+3)^3 - 4(\beta_k+2)^3 + 6(\beta_k+1)^3 - 4\beta_k^3 \\ b_k &= (\beta_k+2)^3 - 4(\beta_k+1)^3 + 6\beta_k^3 \\ c_k &= (\beta_k+1)^3 - 4\beta_k^3 \\ d_k &= \beta_k^3 \\ \beta_k &= (k-1)n/M,\ k \in [1, M/n] \end{aligned} \tag{11}$$

Les coefficients du filtre des 16 pixels voisins sont calculés par les équations (6, 10, 11).

## 3. Factorisation de la matrice *A*

On décompose la matrice $A$ d'ordre $n$ de la manière suivante:

$$A = \left[\begin{array}{c|c} A_d & a \\ \hline a^T & 4 \end{array}\right] \tag{12}$$

où $A_d$ est la matrice principale tridiagonale symétrique d'ordre $(n-1)$, et $a$ est le vecteur:

$$a = (1,\ 0,\ \ldots,\ 0,\ 1)^T \in R^{n-1} \tag{13}$$

La matrice de factorisation de Cholesky $L$ de la matrice $A$ s'écrit en fonction de $L_d$, facteur de Cholesky de la matrice $A_d$:

$$A_d = L_d L_d^T \tag{14}$$

On recherche $L$ telle que:

$$A = L L^T \tag{15}$$

$$L = \left[\begin{array}{c|c} L_d & 0 \\ \hline l^T & \lambda \end{array}\right] \tag{16}$$

$L_d$ est une matrice bidiagonale. On doit trouver $L_d$, $l$ et $\lambda$ tels que:

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

Ceci peut être réalisé par le calcul suivant:

$L_d$ : = facteur de Cholesky de $A_d$,

Résolution du système triangulaire $L_d\, l = a$,

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

La factorisation entraîne donc un nombre d'opérations flottantes proportionnel à $n$.

## 4. Calcul de la matrice *C*

L'équation (4) précédente est équivalente à:

$$G = L L^T C L L^T$$

Elle peut être résolue par la résolution successive des systèmes triangulaires suivants:

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

Chacune des étapes précédentes correspond à la résolution d'un système triangulaire avec $L$ ou $L^T$, à gauche ou à droite, et avec n seconds membres. Pour un second membre, la résolution de l'un de ces systèmes a une complexité égale à $5n + O(1)$. On en déduit une complexité totale de:

$$\text{Comp}\,(C) = 20n^2 + O\,(n). \tag{17}$$

## 5. Calcul des coefficients et de la complexité du filtre des 16 pixels voisins

En utilisant les équations (6, 10, 11), on peut facilement calculer les coefficients des filtres $F_{k,u}$ de ré-échantillonnage des 16 pixels voisins de la B-spline cubique uniforme, pour calculer $m$ pixels de sortie dans chaque direction:

$$F_{k,u} = \begin{bmatrix} a'_u \\ b'_u \\ c'_u \\ d'_u \end{bmatrix} * \begin{bmatrix} a_k & b_k & c_k & d_k \end{bmatrix}$$

$$F_{k,u} = \begin{bmatrix} a_k a'_u & a_k b'_u & a_k c'_u & a_k d'_u \\ b_k a'_u & b_k b'_u & b_k c'_u & b_k d'_u \\ c_k a'_u & c_k b'_u & c_k c'_u & c_k d'_u \\ d_k a'_u & d_k b'_u & d_k c'_u & d_k d'_u \end{bmatrix} \tag{18}$$

avec $(k, u) \in [1,\ m+1]^2$.

Le filtre $F_{k,u}$ est appliqué sur la fenêtre $C_{i,j}$ de dimension 4 x 4 de la matrice $C$ de l'image originale.

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

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

Les coefficients des filtres de ré-échantillonnage de la première ligne et de la première colonne, dépendent des valeurs des coefficients $a_1$, $b_1$, $c_1$ et $d_1$ pour $k = 1$ ou $u = 1$. Le calcul de ces coefficients nous donne les valeurs suivantes:

$$a_1 = 1,\ b_1 = 4,\ c_1 = 1 \text{ et } d_1 = 0$$

Le filtre $F_{1,u}$ de la première ligne se simplifie et devient égal à:

$$F_{1,u} = \begin{bmatrix} a'_u \\ b'_u \\ c'_u \\ d'_u \end{bmatrix} * \begin{bmatrix} 1 & 4 & 1 & 0 \end{bmatrix}$$

$$F_{1,u} = \begin{bmatrix} a'_u & 4a'_u & a'_u & 0 \\ b'_u & 4b'_u & b'_u & 0 \\ c'_u & 4c'_u & c'_u & 0 \\ d'_u & 4d'_u & d'_u & 0 \end{bmatrix} \tag{19}$$

Le filtre $F_{k,1}$ de la première colonne est égal à:

$$F_{k,u}=\begin{bmatrix}1\\4\\1\\0\end{bmatrix}*\begin{bmatrix}a_k & b_k & c_k & d_k\end{bmatrix}$$

$$F_{k,1}=\begin{bmatrix}a_k & b_k & c_k & d_k\\ 4a_k & 4b_k & 4c_k & 4d_k\\ a_k & b_k & c_k & d_k\\ 0 & 0 & 0 & 0\end{bmatrix} \qquad (20)$$

La complexité Comp ($F$) du filtre des 16 *pv* dépend du nombre $m$ des pixels de sortie dans chaque direction. En appliquant les filtres des équations (18, 19, 20) sur l'image d'entrée, et en dénombrant les opérations, on trouve:

$$\text{Comp}\,(F_m) = (31\,m^2 + 46\,m)\,n^2 + O\,(n) \qquad (21)$$

d'où :

Comp $(F_1) = 77\,n^2 + O\,(n)$, Comp $(F_2) = 216\,n^2 + O\,(n)$ et Comp $(F_3) = 417\,n^2 + O\,(n)$.

## 6. Comparaison expérimentale des temps de calcul

Pour mener l'expérience qui suit, nous avons choisi une station munie d'un processeur Pentium III fonctionnant à 863.735 MHz. Nous avons mesuré les temps de calcul (Fig. 1) de la construction de la matrice $C$ en fonction de la dimension $n$ de l'image originale d'entrée (courbe (calcul de la matrice $C$)), ainsi que celui des filtres des 16 *pv* pour un pixel de sortie (courbe (Filtre 16/1 pixel), pour deux pixels de sortie (courbe (Filtre 16/2 pixels)) et pour trois pixels de sortie (courbe (Filtre 16/3 pixels)). La figure 2 représente le ré-échantillonnage d'une fenêtre d'image contenant quatre pixels pour un, deux et trois pixels de sortie utilisant un ensemble de seize pixels d'entrée voisins.

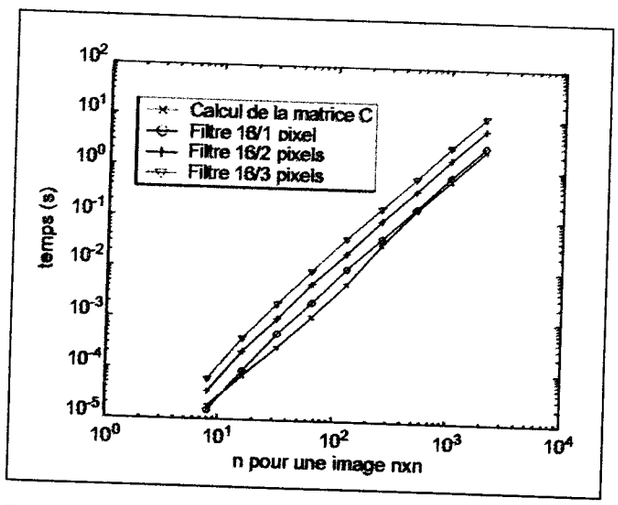



**Figure 1:** Temps d'exécution de la matrice C (—x—), du filtre des 16 pv pour un pixel de sortie (—o—), du filtre des 16 pv pour deux pixels de sortie (—+—) et du filtre des 16 pv pour trois pixels de sortie ( —∀—).

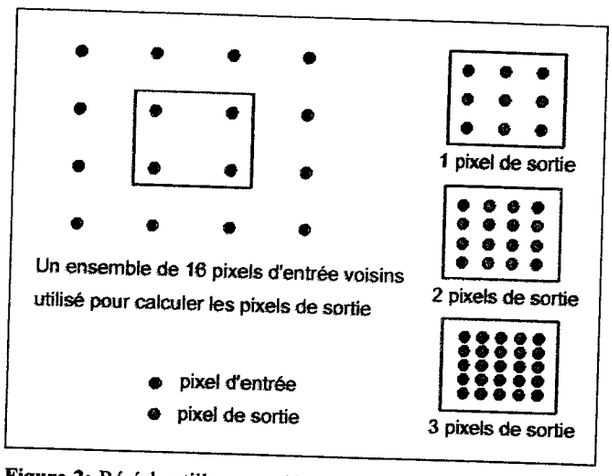



**Figure 2:** Ré-échantillonnage d'images avec 1 pixel, 2 pixels et 3 pixels de sortie.

On vérifie bien grâce au graphique log-log que les temps d'exécution peuvent être estimés par des formules du type $t_n = T\,n^{\alpha}$ où $t_n$ est le temps d'exécution du calcul considéré. Plus précisément, on a vérifié par une régression linéaire dans le graphique, que l'ordre de $n$ était bien 2 et on a identifié les coefficients $T$ correspondant à chaque courbe. Etant donné que $T$ représente le produit du coefficient de $n^2$ dans les formules (17, 21) et du temps d'une opération flottante, on estime donc les vitesses en Megaflops (Tab. 1).

| | Ordre estimé α | Coefficient T estimé | Vitesse en Mflops |
|---|---|---|---|
| Calcul de la matrice C | 2.2 | 1.3826e-007 | 145 |
| Filtre 16/1 pixels | 2.1 | 3.0623e-007 | 251 |
| Filtre 16/2 pixels | 2.1 | 6.9800e-007 | 309 |
| Filtre 16/3 pixels | 2.1 | 1.3756e-007 | 303 |

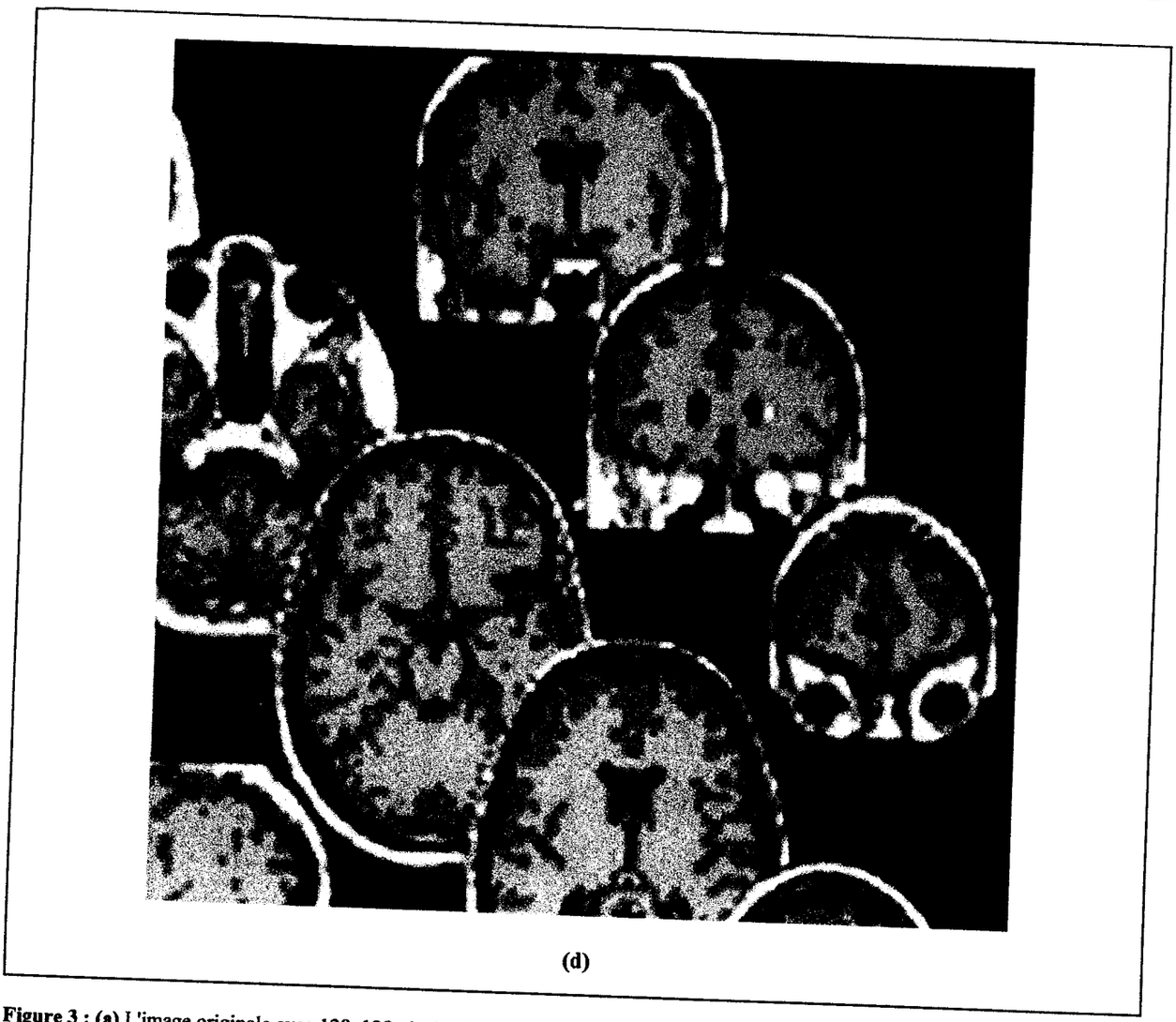
**Tableau 1:** Régression linéaire pour $n$ = 8, 16, 32, 64, 128, 256, 512, 1024, 2048.

On remarque que l'ordre $\alpha$ estimé est pratiquement constant et égal à 2.1 dans le cas des filtres des 16 *pv*. Par contre, dans le cas du calcul de la matrice $C$, l'ordre estimé est égal à 2.2. Cette différence est due aux termes de complexité inférieure négligés.

La figure 3 représente le ré-échantillonnage d'une image réelle de 128x128 pixels pour 1, 2 et 3 pixels de sortie. Nous remarquons un lissage assez important des images de sortie.

Nous avons estimé également les temps d'exécution de ré-échantillonnage de la même image réelle (128x128) de la figure 3 (a), par la méthode du plus proche voisin (*ppv*) et de la bilinéaire. Le tableau 2 représente les temps d'exécution $T_{ex}$ correspondant aux trois méthodes pour un pixel, deux pixels et trois pixels de sortie ainsi que leur ordre de continuité.

Le ré-échantillonnage d'images par la méthode du plus proche voisin est très rapide et conserve les valeurs



(d)

**Figure 3** : **(a)** L'image originale avec 128x128 pixels, **(b)** L'image ré-échantillonnée avec un pixel de sortie, le temps de calcul est estimé à 0.014 s, **(c)** L'image ré-échantillonnée avec deux pixels de sortie, le temps de calcul est estimé à 0.026 s, **(d)** L'image ré-échantillonnée avec trois pixels de sortie, le temps de calcul est estimé à 0.044 s.

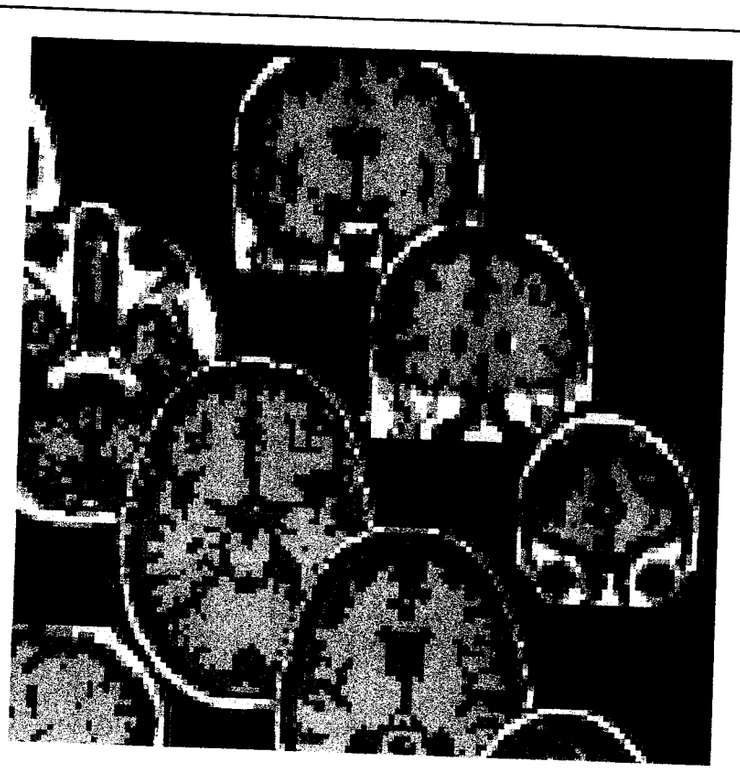
| | $T_{ex}$ en seconde 1 pixel de sortie | $T_{ex}$ en seconde 2 pixel de sortie | $T_{ex}$ en seconde 3 pixel de sortie | Ordre de continuité |
|---|---|---|---|---|
| ppv | 0.12 e-2 | 0.27 e-2 | 0.47 e-2 | 0 |
| bilinéaire | 0.25e-2 | 0.55e-2 | 1.0e-2 | 1 |
| B-spline cubique uniforme | 1.4 e-2 | 2.6 e-2 | 4.4 e-2 | 2 |

**Tableau 2**: Temps d'exécution $T_{ex}$ correspondant aux trois méthodes pour un, deux et trois pixels de sortie.

**(a)**

**(b)**

**(c)**



**(a)**

**(b)**

**Figure 4:** Images ré-échantillonnées : **(a)** l'interpolation du plus proche voisin pour 2 pixels de sortie, **(b)** l'interpolation bilinéaire pour 2 pixels de sortie. L'image d'entrée est celle de la figure 3 (a).

radiométriques d'origine, mais présente un effet d'escalier sur les diagonales qui se traduit par une discontinuité visuelle (Fig. 4a). L'interpolation bilinéaire est un peu moins rapide et engendre un léger effet de lissage (Fig. 4 b). L'interpolation B-spline cubique uniforme est la plus lente, mais elle est plus précise.

## CONCLUSION

Nous avons défini, dans cet article, un algorithme pour ré-échantillonner efficacement des images numériques par la B-spline cubique uniforme. Le calcul des complexités et les essais numériques ont montré que le coût de l'algorithme est $O(n^2)$. Dans le cas des images numériques d'entrée de grandes tailles, le temps de calcul s'avère assez important. Une analyse de cet algorithme montre qu'il sera facilement parallélisable, cet aspect faisant l'objet de travaux ultérieurs.

## REFERENCES


[1]- William K. Pratt, "Digital image processing", A Wiley Interscience Publication, (1978).

[2]- Andrews H.C. and Patterson C.L., "II. Digital interpolation of discrete images", *IEEE Trans. comput.*, vol. C-25, February (1976), pp. 196-202.

[3]- Vrcelj B. and Vaidyanathan P.P., "Efficient implementation of all-digital interpolation", EDICS number: 2-INTR, July (2001), pp. 1-16.

[4]- Unser M., Aldroudi A., Eden M., "Fast B-spline transforms for continues image representation and interpolation", *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 13, March (1991), pp. 277-285.

[5]- Golub G.H. and Van Loan C.F., "Matrix Computations", 3ème édition, John Hopkins, (1996).

[6]- Mahboub M. and Philippe B., "Ré-échantillonnage d'images numériques par la B-spline cubique uniforme", CARI'02 Yaoundé, Octobre (2002), pp. 183-188.

[7]- Mahboub M., "Application de la fonction B-spline cubique uniforme au recalage en résolution spatiale d'images satellites (Landsat-Spot)", MCEA Grenoble, Septembre (1995), pp. 183-188.


□

Constantine, le 15/05/2004


الدكتور خير الدين م. الهاشمي
مدير النشر و التنشيط العلمي

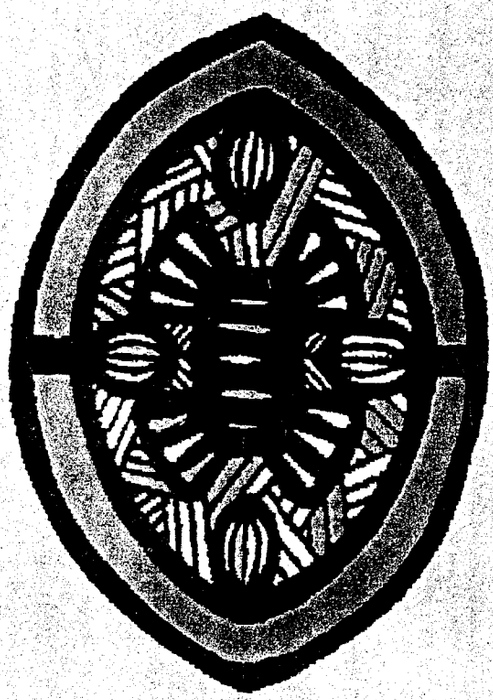
# Annexe Proceedings

1. M. Mahboub, and B. Philippe, *" Ré-échantillonnage d'images numériques par la B-spline cubique uniforme,"* CARI'02 Yaoundé, pp. 183-188, Octobre 2002.

2. M. Mahboub, *Application de la fonction B-spline cubique uniforme au recalage en résolution spatiale d'images satellites (Landsat-Spot)*, MCEA Grenoble, pp. 183-188, Septembre 1995.

# CARI'02

**Actes du 6ème Colloque Africain**
**sur la Recherche en Informatique**
***Proceedings of the 6th African Conference***
***on Research in Computer Science***



**Editeurs scientifiques / *Scientific editors***
**Lala ANDRIAMAMPIANINA / Bernard PHILIPPE /**
**Emmanuel KAMGNIA / Maurice TCHUENTE**

**Avec la contribution de l'Université de Douala**
**et du PNUD**
***With the financial support of the University of Douala***
***and UNDP***

Actes du 6e Colloque Africain
sur la Recherche en Informatique

# CARI'02

*Proceedings of the 6th African Conference*
*on Research in Computer Science*

# Actes du 6e Colloque Africain sur la Recherche en Informatique

# CARI'02

## *Proceedings of the 6th African Conference on Research in Computer Science*

**Editeurs scientifiques / *Scientific editors***
**Lala ANDRIAMAMPIANINA / Bernard PHILIPPE /**
**Emmanuel KAMGNIA / Maurice TCHUENTE**

IMPRIMERIE SAINT PAUL
Yaoundé
Cameroun

INRIA
Institut National de Recherche
en Informatique et en
Automatique
France

## NOTE DE L'EDITEUR

*Chaque communication a été reproduite telle que l'auteur l'a présentée. En conséquence, les lecteurs voudront bien excuser le manque d'homogénéité typographique de ce volume.*

## COMITE DE PROGRAMME / *PROGRAMME COMMITTEE*

**Président/*Chair***

L. Andriamampianina, ESPA, Antananarivo, Madagascar
M. Tchuente, Université de Douala, Cameroun

F. André, IRISA/INRIA Rennes, France
R. André-Obrecht, IRIT, Toulouse, France
J-P. Asselin de Beauville, AUF, Montréal, Québec, Canada
E. Badouel, Ecole Nationale Supérieure Polytechnique, Yaoundé, Cameroun
M-O. Cordier, IRISA/INRIA Rennes, France
R. Deriche, INRIA Sophia Antipolis, France
J-C. Derniame, LORIA/ENSEM, Vandoeuvre les Nancy, France
R. Dieng, INRIA Sophia Antipolis, France
C. Frasson, Université de Montréal, Canada
F. Guerrin, CIRAD, La Réunion
V. Harison, INSCAE, Antananarivo, Madagascar
J-P. Haton, LORIA/INRIA, Vandoeuvre les Nancy, France
J-C. Hochon, IXI, Toulouse, France
M. Jaoua, ENIT-Lamsin, Tunis, Tunisie
B. Kerhervé, Université du Québec, Montréal, Canada
T. Laskri, Université d'Annaba, Algérie
Z. Mammeri, IRIT/Université Paul Sabatier, Toulouse, France
N. Mikou, Université de Bourgogne, Dijon, France
O. Monga, IRD, France
M. T. Niane, Université Gaston Berger, Saint Louis, Sénégal
J-P. Nzali, Université de Yaoundé I, Cameroun
V. Oria, New Jersey Institut of Technologie, Newark, USA
S. Oumtanaga, INPHB, Yamoussoukro, Côte d'Ivoire
C. Pettang, Ecole Nationale Supérieure Polytechnique, Yaoundé, Cameroun
A. Pirotte, Université de Louvain, Belgique
B. Plateau, ENSIMAG, Montbonnot Saint Martin, France
P. Quinton, IRISA/IFSIC, Rennes, France
P. Renaud, UNITAR, Genève, Suisse
P. Rolin, France Télécom/CNET, Issy les Moulineaux, France
H. Sahraoui, Université de Montréal, Canada
I. Sahko, Université de Metz, France
M. Sellami, Université d'Annaba, Algérie
A. Seznec, IRISA/INRIA Rennes, France
K. Tombre, LORIA/INPL, Vandoeuvre les Nancy, France
C. Viho, IRISA/Université de Rennes I, France
F. Voisin, LRI/Université Paris Sud, Orsay, France

## COMITE PERMANENT / *PERMANENT COMMITTEE*

**Représentants des chercheurs africains / *African Researchers Delegates***

K. E. Akoussah, Université du Benin, Lomé, Togo
L. Andriamampianina, ESPA, Antananarivo, Madagascar, **Président / *Chair***
E. M. Daoudi, Université d'Oujda, Maroc
E. Kamgnia, Université de Yaoundé I, Cameroun
G. K. J. Kouacou, INP-HB, Yamoussoukro, Côte d'Ivoire
M. K. Luhandjula, Université de Kinshasa, R. D. Congo
N. R. Razafindrakoto, CNRIT, Antananarivo, Madagascar
R. Shumba-Mazhindy, University of Zimbabwe
Y. Slimani, Université de Tunis, Tunisie
Z. Zemirli, INI, Alger, Algérie

**Représentants des institutions / *Institutions Delegates***

J-P. Asselin de Beauville, AUF, Paris, France
C. Lobry, CIMPA, Nice, France
B. Philippe, INRIA Rennes, France - Secrétaire / Secretary
B. Poniatowski, UNU, Tokyo, Japon
J. Sor, CIRAD, Montpellier, France
J-P. Treuil, IRD, Bondy, France

## COMITE LOCAL D'ORGANISATION / *LOCAL ORGANIZATION COMMITTEE*

**Superviseur / *Supervisor***
Professeur Jean TABI MANGA, Recteur de l'Université de Yaoundé I
**Président / *Chair***
Emmanuel KAMGNIA, Chef de Dpt. d'Informatique, FSC, Université de Yaoundé I
**Vice Président / *Vice-chairman***
Basile LOUKA, Dpt. d'Informatique, FSC, Université de Yaoundé I
**Secrétaire / *Secretary***
Jean Pierre NZALI, Dpt. d'Informatique, FSC, Université de Yaoundé I
**Secrétaire Adjoint / *Deputy secretary***
Eric BADOUEL, Chef de Dpt. Génie Informatique, ENSP, Université de Yaoundé I
**Trésorier / *Treasurer***
Lot TCHEEKO, Dpt. de Génie Informatique, ENSP, Université de Yaoundé I
**Commissaire aux Comptes / *Auditor***
Claude TANGHA, Dpt. de Génie Informatique, ENSP, Université de Yaoundé I
**Relations publiques / *Public Relations***
Pauline Laure FOTSO, Dpt. d'Informatique, FSC, Université de Yaoundé I
**Logistique / *Logistic***
Marcel FOUDA NDJODO, Dpt. d'Informatique, FSC, Université de Yaoundé I

## COORDINATION

M-C. Sance-Plouchart, IRISA/INRIA Rennes, France
V. Verdon, IRISA/INRIA Rennes, France

## HAUT PATRONAGE/UNDER THE PATRONAGE OF

Son Excellence Peter Mafany MUSONGUE, Premier Ministre, Chef du Gouvernement, Cameroun

## COMITE d'HONNEUR/HONORARY COMMITTEE

| | |
|---|---|
| M. François Xavier NGOUBEYOU | Ministre d'Etat chargé des relations extérieures |
| Pr. Maurice TCHUENTE | Ministre de l'Enseignement Supérieur |
| M. Michel MEVA'A MEBOUTOU | Ministre des Finances |
| Dr. Zacharie PEREVET | Ministre de la Recherche Scientifique et Technique |
| Pr. Jacques FAME NDONGO | Ministre de la Communication |
| M. Maximin NKOUE NKONGO | Ministre des Postes et Télécommunications |
| S.E. Jean Paul VEZIANT | Ambassadeur de France au Cameroun |
| S.E. George M. STAPLES | Ambassadeur des Etats-Unis d'Amérique au Cameroun |
| S.E. Michel E. PERRAULT | Haut Commissaire du Canada au Cameroun |
| S.E. Richard WILDASH | British High Commissioner, Cameroun |
| Mme Patricia de MOWBRAY | Coordonnateur Résident de l'ONU, Cameroun |
| Pr. Bernard LARROUTUROU | Président Directeur Général de l'INRIA, France |
| Pr. Philippe LAZAR | Président de l'IRD, France |
| Pr. Gérard MATHERON | Président du CIRAD, France |
| Pr. Hans VAN GINKEL | Recteur de l'UNU, Tokyo |
| Pr. Michèle GENDREAU-MASSALOUX | Recteur de l'AUF, Canada |
| M. Emmanuel NGUIAMBA N. | Directeur Général de CAMTEL |

**SECRETARIAT :**
Mme NONO née SEMOU JAKPOU Edwige

# *Ré-échantillonnage d'images numériques par la B-spline cubique uniforme*

*M. Mahboub* et B. Philippe** [1]*

*Département de physique, Faculté des Sciences,
Université Abou Bakr Belkaid Tlemcen, Algérie.
** IRISA - INRIA Rennes, France.

**Mots clefs**

Rééchantillonnage d'images, B-spline cubique uniforme, interpolation, reconstruction, factorisation de Cholesky.

**Résumé**

*Dans le cas du rééchantillonnage d'images numériques par la B-spline cubique uniforme, un pixel de sortie est calculé par un filtre appliqué sur seize pixels voisins (pv) d'une image ou d'une matrice C. Pour obtenir cette matrice C, nous devons résoudre un système d'équations linéaires. Nous résolvons ce système linéaire par une factorisation de Cholesky adaptée. Cette technique nous a permis de calculer la matrice intermédiaire C avec un temps de calcul inférieur à celui du filtre des seize pv. Nous présentons, dans cet article, l'algorithme de résolution et le temps de calcul pour son implémentation.*

**Abstract**

For digital image resampling by uniform cubic B-spline, an output pixel is computed from a filter applied to 16 neighboring pixels (np) of an image or a matrix C. To obtain this matrix, we must solve a system of linear equations. We propose, in this work, an adapted Cholesky factorization for solving this linear system. This technique allows to calculate the matrix C with a computational time smaller than the computational time for the 16 np filter. In this paper, we present the algorithm and we report computational times obtained from its implementation.

## 1. Introduction

Dans plusieurs applications de traitement numérique d'images, il est souvent nécessaire de faire des transformations géométriques non linéaires des positions des pixels de la grille de l'image comme par exemple le cas du recalage géométrique d'images. Les transformées des positions des pixels ne coïncident généralement pas avec la grille de l'image. Un algorithme de rééchantillonnage permet d'obtenir la valeur de ces pixels intermédiaires. Le temps de calcul dépend de la complexité de la fonction d'interpolation utilisée. L'interpolation idéale ou de Shannon à deux dimensions (cas des images), dans le cas d'un échantillonnage régulier et uniforme avec $\Delta x$ et $\Delta y$ les pas d'échantillonnage suivant les directions $x$ et $y$ respectivement, est définie par la relation suivante:

$$F_r(x,y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} F(i\Delta x, j\Delta y) R(x - i\Delta x, y - j\Delta y) \qquad (1)$$

avec,

$$R(x,y) = \frac{\sin \omega_1 x}{\omega_1 x} \frac{\sin \omega_2 y}{\omega_2 y} \qquad (2)$$

[1] Email : m_mahboub@mail.univ-tlemcen.dz et philippe@irisa.fr

Les pulsations $\omega_1$ et $\omega_2$ sont les pulsations de coupure suivant les directions x et y respectivement, $F(i\Delta x, j\Delta y)$ est la valeur du pixel (i, j) de l'image d'entrée originale à reconstruire et $F_r(x, y)$ est l'image de sortie finale. L'interpolation idéale permet de reconstruire l'image exacte, mais elle est difficile à réaliser. Plusieurs fonctions, dont la réalisation est plus ou moins compliquée, sont utilisées pour approcher la fonction R(x, y). Parmi ces fonctions, on peut citer les fonctions d'interpolation du plus proche voisin, linéaire et la B-spline cubique uniforme [1]. L'interpolation bilinéaire est utilisée, le plus souvent, dans les applications graphiques. Cette interpolation n'utilise que quatre pixels voisins de la grille de l'image pour chaque pixel de sortie. Par contre, certaines applications professionnelles utilisent les interpolations cubiques. Le rééchantillonnage par la B-spline cubique uniforme utilise seize pixels voisins (pv), d'une image (ou matrice) C intermédiaire, pour calculer les pixels de sortie. Pour obtenir cette matrice C, nous devons résoudre un système d'équations linéaires. Andrews et Patterson ont introduit en 1976 [2] une méthode de calcul par des multiplications de matrices utilisant une inversion explicite très coûteuse en $O(n^3)$ où n est la dimension de la matrice. Depuis, d'autres techniques en $O(n^2)$ ont été introduites pour calculer la matrice C: transformation en z [3, 4]; factorisation d'une matrice tridiagonale de passage A mentionnée dans [4]. Dans cet article nous considérons dans nos calculs, une matrice A correspondant à une image périodique qui permet de considérer une matrice carrée dans le calcul de C. Bien que la matrice A ne soit plus bidiagonale, la complexité reste en $O(n^2)$. Cet algorithme peut aussi être appliqué à des images non périodiques. Nous proposons, dans ce travail, un algorithme de résolution de ce système linéaire par la factorisation de Cholesky [5].

**2. Rééchantillonnage par les fonctions splines**

Le rééchantillonnage d'images par les fonctions splines est exprimé par l'équation suivante:

$$g(x,y) = \sum_{i=1}^{n} \sum_{j=1}^{n'} c_{ij}\, s(x - i\Delta x)\, s(y - j\Delta y) \qquad (3)$$

ou par l'expression matricielle dans le cas d'images numériques par

$$G = A_n\, C\, A_{n'} \qquad (4)$$

La matrice $G$ représente l'image originale d'entrée, de dimensions n × n'. Les deux matrices $A_n$ *et* $A_{n'}$ de l'équation (4) sont calculées respectivement par les splines de bases dans les directions x et y pour obtenir les valeurs des pixels de G. La matrice $C$ est l'inconnue; elle sera calculée par inversion de l'équation (4) pour aboutir à:

$$C = A_n^{-1}\, G\, A_{n'}^{-1} \qquad (5)$$

On considère des matrices $B_1$ et $B_2^T$ effectuant respectivement un rééchantillonnage suivant les lignes et les colonnes. L'image $G'$ rééchantillonnée est déterminée par l'équation suivante:

$$G' = B_1\, C\, B_2^T \qquad (6)$$

Dans la suite nous nous restreignons à des images carrées c'est-à-dire à n = n'.

:s directions x et y
..'entrée originale à
ermet de reconstruire
réalisation est plus
ni ces fonctions, on
la B-spline cubique
lans les applications
ille de l'image pour
ielles utilisent les
niforme utilise seize
les pixels de sortie.
néaires. Andrews et
..cations de matrices
ension de la matrice.
ler la matrice C:
sage A mentionnée
correspondant à une
cul de C. Bien que la
hme peut aussi être
, un algorithme de

imé par l'équation

s n × n'. Les deux
plines de bases dans
C est l'inconnue; elle

rééchantillonnage
inée par l'équation

ire à n = n'.

**3. Cas de la B-spline cubique uniforme**

La fonction de base de la B-spline cubique uniforme, s(x) peut s'écrire sous la forme suivante:

$$s(x) = x_+^{\,3} - 4(x-\Delta x)_+^{\,3} + 6(x-2\Delta x)_+^{\,3} - 4(x-3\Delta x)_+^{\,3} \qquad (7)$$

avec

$$x_+^{\,3} = \begin{cases} x^3 & si\, x > 0, \\ 0 & si\, x \le 0 \end{cases} \qquad (8)$$

La matricc $A$ calculée en supposant l'image périodique, est égale à:

$$A = \begin{bmatrix} 4 & 1 & 0 & & & \dots & & 0 & 1 \\ 1 & 4 & 1 & 0 & & \dots & & & 0 \\ 0 & 1 & 4 & 1 & 0 & \dots & & & 0 \\ : & & & & : & & & & : \\ 0 & & & \dots & & 1 & 4 & 1 & 0 \\ 0 & & & \dots & & 0 & 1 & 4 & 1 \\ 1 & 0 & & \dots & & & 0 & 1 & 4 \end{bmatrix} \qquad (9)$$

Le calcul des coefficients de la matrice $B$ est identique à celui de la matrice $A$, mais avec un pas d'échantillonnage plus faible égal à l'inverse du nombre de pixels à déterminer. Le calcul à une dimension montre que cette matrice est formée de n "blocs" dont chacun contient M / n lignes et n colonnes [6]:

$$B = \left[\begin{array}{cccccccccc} c_k & d_k & 0 & 0 & \dots & 0 & 0 & a_k & b_k \\ \hline b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \\ \hline a_k & b_k & c_k & d_k & \dots & 0 & 0 & 0 & 0 \\ \hline & & & & : & & & & \\ \hline b_k & c_k & d_k & 0 & \dots & 0 & 0 & 0 & a_k \end{array}\right] \qquad (10)$$

avec,

$$\begin{aligned} a_k &= (\beta_k + 3)^3 - 4(\beta_k + 2)^3 + 6(\beta_k + 1)^3 - 4\beta_k^{\,3} \\ b_k &= (\beta_k + 2)^3 - 4(\beta_k + 1)^3 + 6\beta_k^{\,3} \\ c_k &= (\beta_k + 1)^3 - 4\beta_k^{\,3} \\ d_k &= \beta_k^{\,3} \\ \beta_k &= kn/M,\ k \in [1, M/n] \end{aligned} \qquad (11)$$

Les coefficients du filtre des 16 pv sont calculés par les équations (6, 10, 11).

**4. Factorisation de la matrice $A$**

On décompose la matrice $A$ d'ordre n de la manière suivante:

$$A = \left[\begin{array}{c|c} A_d & a \\ \hline a^T & 4 \end{array}\right] \qquad (12)$$

où $A_d$ est la matrice principale tridiagonale symétrique d'ordre (n -1), et $a$ est le vecteur:

$$a = (1, \ 0, \ \ldots, \ 0, \ 1)^T \in R^{n-1} \qquad (13)$$

La matrice de factorisation de Cholesky $L$ de la matrice $A$ s'écrit en fonction de $L_d$, facteur de cholesky de la matrice $A_d$:

$$A_d = L_d L_d{}^T \qquad (14)$$

On recherche $L$ telle que:

$$A = L L^T \qquad (15)$$

$$L = \left[\begin{array}{c|c} L_d & 0 \\ \hline l^T & \lambda \end{array}\right] \qquad (16)$$

$L_d$ est une matrice bidiagonale. On doit trouver $L_d$, $l$ et $\lambda$ tels que:

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

Cela peut être réalisé par le calcul suivant:

$L_d$ : = facteur de Cholesky de $A_d$,
Résolution du système triangulaire $L_d \, l = a$,
$\lambda = \sqrt{4 - \|l\|^2}$ .

La factorisation entraîne donc un nombre d'opérations flottantes proportionnel à n.

**5. Calcul de la matrice C**

L'équation (4) précédente est équivalente à:

$$G = L L^T C L L^T$$

Elle peut être résolue par la résolution successive des systèmes triangulaires suivants:

$$G = L\,Y,$$
$$Y = L^T X,$$
$$X = Z\,L^T,$$
$$Z = C\,L.$$

Chacune des étapes précédentes correspond à la résolution d'un système triangulaire avec $L$ ou $L^T$, à gauche ou à droite, et avec n seconds membres. Pour un second membre, la résolution de l'un de ces systèmes a une complexité égale à 5n + O(1). On en déduit une complexité totale de:

$$\text{Comp (C)} = 20n^2 + O(n). \qquad (17)$$

**6. Complexité de l'algorithme du filtre**

La complexité Comp (F) du filtre des 16 pv dépend du nombre m des pixels de sortie de rééchantillonnage. En dénombrant les opérations, on trouve

$$\text{Comp } (F_m) = (\,31\,m^2 + 46\,m\,)\,n^2 + O(n), \qquad (18)$$

d'où Comp $(F_1) = 77\,n^2 + O(n)$, Comp $(F_2) = 216\,n^2 + O(n)$ et Comp $(F_3) = 417\,n^2 + O(n)$

**7. Comparaison expérimentale des temps de calcul**

Pour mener l'expérience qui suit, nous avons choisi une station munie d'un processeur Pentium III fonctionnant à 863.735 Mhz. Nous avons mesuré les temps de calcul ( figure 1 ) de la construction de la matrice C en fonction de la dimension n de l'image d'entrée (courbe (calcul de la matrice C)), ainsi que celui des filtres des 16 pixels voisins pour un pixel de sortie (courbe ( Filtre 16/1 pixel ), pour deux pixels de sortie ( courbe (Filtre 16/2 pixels) ) et pour trois pixels de sortie (courbe ( Filtre 16/3 pixels )).

On vérifie bien grâce au graphique log-log que les temps d'exécution peuvent être estimés par des formules du type

$$t_n = T\,n^{\alpha}$$

où $t_n$ est le temps d'exécution du calcul considéré. Plus précisément, on a vérifié par une régression linéaire dans le graphique, que l'ordre de n était proche de 2 et on a identifié les coefficients T correspondant à chaque courbe.

Régression linéaire pour n = 8, 16, 32, 64, 128, 256, 512, 1024, 2048.

| | Ordre estimé $\alpha$ | Coefficient T estimé (s) |
|---|---|---|
| Calcul de la matrice C | 2.2 | 1.3826e-007 |
| Filtre 16/1 pixels | 2.1 | 3.0623e-007 |
| Filtre 16/2 pixels | 2.1 | 6.9800e-007 |
| Filtre 16/3 pixels | 2.1 | 13.756e-007 |

**Tableau 1**

Dans le cas des filtres des 16 pv, on remarque que l'ordre _ estimé est pratiquement constant et égal à 2.1. Par contre, dans le cas du calcul de la matrice C l'ordre estimé est égal à 2.2. Cette différence est due aux termes de complexité inférieure négligés. La régression de C

est moins bonne car le calcul est plus irrégulier dans le cas de la résolution de systèmes triangulaires.

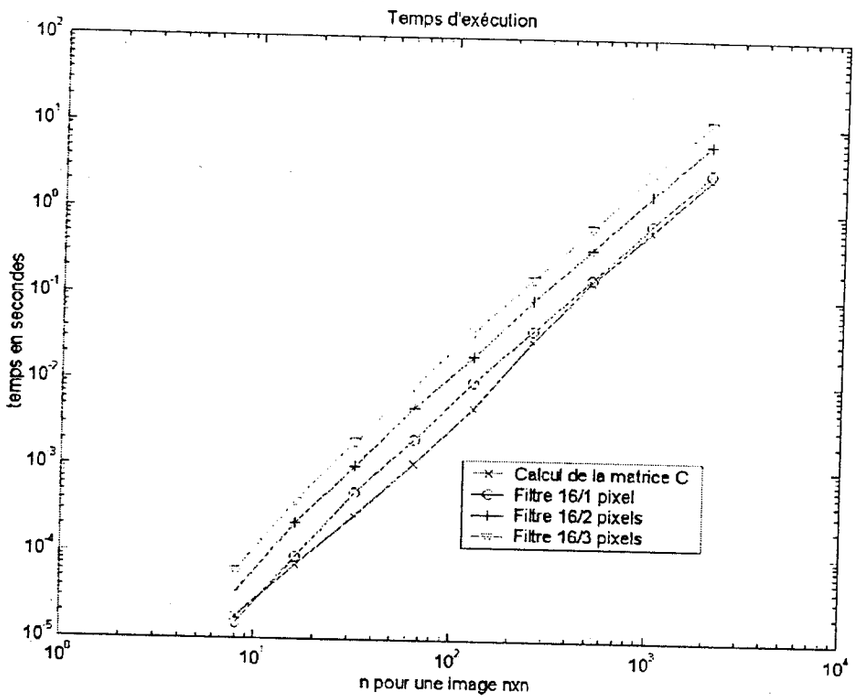
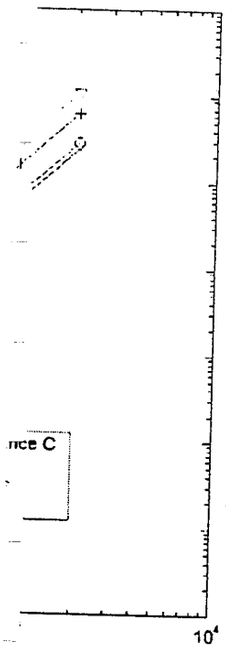



**Figure 1 :** Temps d'exécution: de la matrice C courbe ( calcul de la matrice C ), du filtre des 16 pv pour un pixel de sortie courbe ( Filtre 16/1 pixel ), du filtre des 16 pv pour deux pixels de sortie courbe ( Filtre 16/2 pixels ) et du filtre des 16 pv pour trois pixels de sortie courbe ( Filtre 16/3 pixels ).

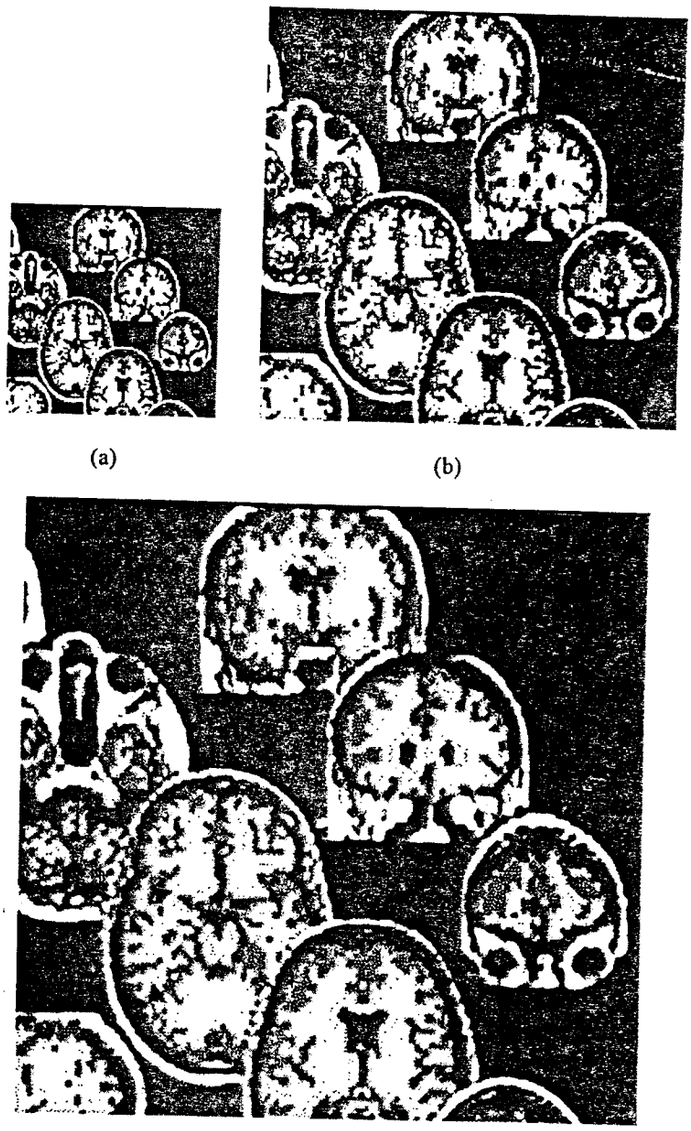
La figure 2 représente le rééchantillonnage d'une image réelle de 128x128 pixels pour 1 et 2 pixels de sortie.

solution de systèmes




a matrice C ), du filtre
filtre des 16 pv pour
16 pv pour trois pixels

128x128 pixels pour 1



(a) (b)

(c)

**Figure 2:** (a) image original 128x128 pixels, (b) image rééchantillonnée avec un pixel de sortie et le temps de calcul estimé à 0.014 s et (c) image rééchantillonnée avec deux pixels de sortie et le temps de calcul estimé à 0.026 s.

## 8. Conclusion

Nous avons défini, dans cet article, un algorithme pour rééchantillonner efficacement des images numériques. Le calcul des complexités et les essais numériques ont montré que le coût de $O(n^2)$ de l'algorithme. Une analyse de cet algorithme montre qu'il sera facilement parallelisable ceci fera l'objet de futures recherches.


## 9. Remerciements

Les auteurs remercient les lecteurs anonymes pour leurs suggestions utiles à l'amélioration de cet article.


## 10. Références


[1] William K. Pratt , *Digital image processing*, A Wiley Interscience Publication 1978.

[2] H.C Andrews and C.L Patterson II, *Digital interpolation of discrete images* ; IEEE Trans. comput., vol. C-25 , pp. 196-202, February 1976.

[3] B. Vrcelj and P.P. Vaidyanathan, *Efficient implementation of all-digital interpolation*, EDICS number: 2-INTR, pp. 1-16, July 2001.

[4] M. Unser, A. Aldroudi, M. Eden, *Fast B-spline transforms for continus image representation and interpolation*, IEEE Trans. Pattern Anal. Mach. Intell. Vol. 13, pp. 277-285, March 1991.

[5] G.H. Golub et C. F. Van Loan. *Matrix Computations*, 3ème édition, John Hopkins, 1996.

[6] M. Mahboub, *Application de la fonction B-spline cubique uniforme au recalage en résolution spatiale d'images satellites (Landsat-Spot)* , MCEA Grenoble, pp. 183-188, Septembre 1995.






Institut de recherche
pour le développement





**INRIA Domaine de Voluceau**
**B.P. 105**
**F-78153 Le Chesnay Cedex**
**ISBN 2-7261-1214-5**

**Imprimerie Saint-Paul**
**Yaoundé**
**Cameroun**

# MEDITERRANEAN CONFERENCE ON ELECTRONICS AND AUTOMATIC CONTROL


# CONFÉRENCE MÉDITERRANÉENNE SUR L'ÉLECTRONIQUE ET L'AUTOMATIQUE

## Tome I

## Traitement du Signal et des Images
## Image and Signal Processing

Grenoble, France
13-15 September 1995 / 13-15 Septembre 1995

Organized by / Organisée par
INPG (National Polytechnic Institute of Grenoble - France)
University of Tizi-Ouzou (Institute of Electronics - Algeria)

École Nationale d'Ingénieurs Électriciens de Grenoble
(ENSIEG - INPG - FRANCE)

# AVANT-PROPOS

La Conférence Méditerranéenne sur l'Électronique et l'Automatique **MCEA «95»** est dans la lignée d'**ICEA-92** tenue à l'initiative de l'Institut d'Electronique de l'Université de Tizi-Ouzou.

La méditerranée, berceau de la civilisation, est le symbole des objectifs de développement et de diffusion des connaissances que se sont fixés les organisateurs de **MCEA «95»**.

L'électronique et l'automatique sont les clés de l'avenir dans un monde de communications et de machines. MCEA «95» donne la parole aux chercheurs et aux développeurs oeuvrant dans ces domaines.

L'ouvrage rassemblant les communications et les affiches présentées à **MCEA «95»** est organisé en trois parties :

- le tome 1, «Traitement du signal et des images» décrit les méthodes nouvelles de traitement et d'interprétation des signaux et des images et les applications aux réseaux neuromimétiques,
- le tome 2, «Automatique» concerne les méthodes de contrôle et de commande, la gestion de la production (productique) et les applications à la commande des moteurs,
- le tome 3 «Électronique» est plus directement axé sur les applications ; il traite des capteurs, de la microélectronique, de l'architecture des systèmes, de l'enseignement assisté par ordinateur et du sondage de l'environnement.

Cet ouvrage est le fruit du travail et de la réflexion des tous les participants à **MCEA «95»**. Que le dialogue, facilité par les nouvelles techniques de communication, et le progrès, issu des nouveaux modes de gestion de l'économie des machines, ouvrent la voie d'une ère nouvelle.

J.L. Lacoume

S. Ameur

# REMERCIEMENTS

MCEA «95» concrétise les relations étroites qui se sont développées entre les chercheurs de l'Institut National Polytechnique de Grenoble et de l'Université de Tizi-Ouzou. Leur participation à l'évaluation des articles proposés et à l'organisation de la conférence a permis sa réalisation.

MCEA «95» n'aurait pu se tenir sans le soutien de l'accord programme entre la France et l'Algérie (94 MDU 260 ) et sans l'aide constante de Monsieur le Professeur J.P. Gélard animateur du comité mixte d'évaluation et de prospective de la coopération inter-universitaire franco-algérienne (CMEP).

Le Groupe d'Étude de Traitement du Signal et des Images (GRETSI) nous a apporté son concours en nous donnant un cadre juridique et financier : cette association poursuit ainsi son œuvre de promotion des recherches dans les sciences de l'information.

L'Institut National Polytechnique de Grenoble (INPG), et particulièrement l'École Nationale Supérieure des Ingénieurs Électriciens de Grenoble (ENSIEG), nous ont efficacement soutenu en recevant le comité d'organisation et en accueillant MCEA «95» sur le site fonctionnel et agréable de l'ENSIEG.

MCEA «95» tenu sur le campus de Gières-Saint Martin d'Hères a bénéficié de son environnement bucolique et de ses moyens pour assurer le confort et l'aménagement matériel de la conférence.

L'édition du programme et des actes de MCEA «95» a été assurée avec brio par F.L. Communication.

Que tous ceux qui, directement ou indirectement, nous ont aidé et encouragé soient ici remerciés pour leur aide efficace.

Le comité d'organisation.

# CONFÉRENCE MÉDITERRANÉENNE SUR L'ÉLECTRONIQUE ET L'AUTOMATIQUE

# MEDITERRANEAN CONFERENCE ON ELECTRONICS AND AUTOMATIC CONTROL

## Présidents / Chairmen

Jean-Louis LACOUME
Professeur à l'Institut National Polytechnique
de Grenoble. France

Soltane AMEUR
Professeur à l'Université
de Tizi-Ouzou. Algérie

## Comité d'Organisation / Organizing commitee

A. ADANE (UTO)
H. ALLA (INPG)
S. BACHA (INPG)
A. CHÉHIKIAN (INPG)
B. GUÉRIN (INPG)
G. JOURDAIN (INPG)
C. JUTTEN (INPG)
Z. SIMEU-ABAZI (INPG)
I. TAS (UTO)

## Comité Scientitique / Scientific Committee

D. ABOUTAJDINE (Morocco)
A. ADANE (Algeria)
M. ALAMIR (France)
H. ALLA (France)
S. AMEUR (Algeria)
L. AUBARD (France)
S. BACHA (France)
O. BADR (Egypt)
A. BARLAUD (France)
A. BENAMARA (France)
B. BESOMBES (France)
J. BIGEON (France)
F. BOULAUT (France)
G. BOUVIER (France)
B. CABON (France)
F. CASTANIE (France)
H. CHAFOUK (Morocco)
Y.A. CHAPUIS (France)
J.P. CHARRAS (France)
A. CHÉHIKIAN (France)
P.Y. COULON (France)
D. DAVID (France)
M. DEPLEY (France)
M. DJEDDI (Algeria)
G. FAVIER (France)
A. FIGUERAS (Spain)
J.M. FLAUS (France)
V. FRISTOT (France)
D. GEORGES (France)
D. GERMAIN (France)
M. GHARBI (France)
B. GUÉRIN (France)
P. GOUNON (France)
S. HADDAB (Algeria)
J. HERAULT (France)
P. HORACEK (France)
G. JOURDAIN (France)
C. JUTTEN (France)
G. KAMARINOS (France)
J.L. LACOUME (France)
M.A. LAGUNAS (Spain)
C. LAUGIER (France)
Y. LECLUSE (France)
G. LEJEUNE (France)
J. LIENARD (France)
F. LUTHON (France)
N. MARTIN (France)
E. MOISAN (France)
M. M'SAAD (France)
M. NAJIM (France)
A. OUAHABI (Algeria)
P. PELLERIN (France)
C. POUPOT (France)
J. ROUDET (France)
P. SAGUET (France)
E. SIMEU (France)
Z. SIMEU-ABAZI (France)
G. TZIRITAS (Greece)
A. WEILL (France)

Application de la fonction B-spline cubique uniforme au recalage en résolution spatiale d'images satellites ( Landsat-Spot ).

Monsieur **MAHBOUB Mourad**

UNIVERSITE DE TLEMCEN
INSTITUT DES SCIENCES EXACTES
B.P 119 DEPARTEMENT DE PHYSIQUE
TLEMCEN

Application of uniform cubic B-spline for image registration of spatial resolution.

Abstract :

We present, in this work, an application of uniform cubic-B spline function for image registration ( Landsat-Spot ). Ideal interpolation($\frac{\sin x}{x}$), which, as we know from the sampling theorem requires an infinite number of terms. We developped an algorithm for image registration, which, use uniform cubic-B spline interpolation function. Resampling a window ( 64 x 85 pixels ) of Landsat image with four pixels in horizontal direction and three pixels in vertical direction, we obtain a result image ( 256 x 255 pixels ) with ( 20 m x 19 m ) spatial resolution, this represent practically Spot's resolution.

## I - I N T R O D U C T I O N .

Dans de nombreux cas nous disposons d'un signal échantillonné et nous désirons reconstruire le signal original,tout en supposant que l'échantillonnage est fait d'une manière que le théorème de Shannon est vérifié. Dans le cas d'une image, nous disposons d'une matrice de points et on désire avoir l'éclairement en un point quelconque. Ceci peut se faire soit par des filtres interpolateurs, qui sont difficilement réalisable et dont la fonction de transfert est à deux dimensions ou par des fonctions d'interpolations à deux dimensions.

L'interpolation idéale ( ou de Shannon ) est définie dans le cas d'un signal à deux dimensions par la relation suivante : [1], [2]

$$F_R(x,y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \Gamma(i\Delta x, j\Delta y)\, R(x-i\Delta x, y-j\Delta y). \qquad (1)$$

$$\text{avec} \quad R(x,y) = \frac{\sin w_1 x}{w_1 x} \cdot \frac{\sin w_2 y}{w_2 y} \qquad (2)$$

$w_1$ et $w_2$ : sont respectivement les pulsations de coupures suivant les directions x et y.

L'interpolation idéale permet de reconstruire l'image exacte, mais elle est difficile à réaliser ( le nombre d'échantillons est infini ). Plusieurs fonctions, dont la réalisation est plus ou moins compliquée, sont utilisées pour approximer la fonction R(x,y). Parmis ces fonctions, on peut citer la fonction rectangle, triangle et la fonction B-spline cubique uniforme [2].

## II - INTERPOLATION PAR LA FONCTION B-SPLINE CUBIQUE UNIFORME :

En général, si la spline de baseest séparable, l'interpolation à deux dimensions peut être exprimée par la relation :

$$g(x,y) = \sum_{i=1}^{N'} \sum_{j=1}^{N} c_{ij} \cdot s_i(x - i\Delta x) \cdot s_j(y - j\Delta y) \quad (3)$$

Les coefficients $c_{i,j}$ sont déterminés par les valeurs des échantillons $g(x_i y_j) i = 1, \ldots, N' ; j = 1, \ldots, N.$

Sous forme matricielle, cette relation peut s'écrire de la manière suivante :

$$N' \downarrow \overrightarrow{[G]}^{N} = N' \downarrow \overrightarrow{[A]}^{N'} \cdot N' \downarrow \overrightarrow{[C]}^{N} \cdot N \downarrow \overrightarrow{[A]}^{N} \quad (4)$$

Et par invension, on obtient la matrice $[C]$ :

$$N' \downarrow \overrightarrow{[C]}^{N} = N' \downarrow \overrightarrow{[A]}^{N'} {}^{-1} \cdot N' \downarrow \overrightarrow{[G]}^{N} \cdot N \downarrow \overrightarrow{[A]}^{N} {}^{-1} \quad (5)$$

Le ré-échantillonnage de la matrice $[G]$ permet d'obtenir une matrice $[G']$ de dimensions M' x M avec M et M' sont respectivement supérieur à N et N'. La matrice $[G']$ peut s'écrire sous la forme suivante :

$$M' \downarrow \overrightarrow{[G']}^{M} = M' \downarrow \overrightarrow{[B]}^{N'} \cdot N' \downarrow \overrightarrow{[C]}^{N} \cdot N \downarrow \overrightarrow{[B]}^{M} {}^{t} \quad (6)$$

Les matrices $[B]$ et $[B]^t$ sont de dimensions N' x M' et N x M. Ces matrices effectuent respectivement un ré-échantillonnage suivant les lignes et les colonnes de la matrice $[C]$. Les coefficients de ces matrices dépendent essentiellement de la fonction de base d'interpolation. La fonction de base de la cubique B-spline uniforme est définie [2] par :

$$s(x) = x_+^3 - 4(x - \Delta x)_+^3 + 6(x - 2\Delta x)_+^3 - 4(x - 3\Delta x)_+^3 \quad (7)$$

Cette fonction de base permet le calcul des coefficients des matrices $[A]$ et $[B]$ . La matrice $[A]$ est une matrice cyclique,

$$[A] = \begin{bmatrix} 4 & 1 & & & & & & 1 \\ 1 & 4 & 1 & & & & & \\ & 1 & 4 & 1 & & & & \\ & & & \cdot & & & & \\ & & & & \cdot & & & \\ & & & & & \cdot & & \\ & & & & & 1 & 4 & 1 \\ 1 & & & & & & 1 & 4 \end{bmatrix} \qquad (8)$$

Le calcul à une dimension des coefficients de la matrice $[B]$ à montrer que cette matrice est formée de N "Blocks" dont chacun contient $\frac{M}{N}$ lignes et N colonnes :

$$[B] = \begin{bmatrix} c_k & d_k & 0 & 0 & \cdots & 0 & 0 & a_k & b_k \\ \hline b_k & c_k & d_k & 0 & \cdots & 0 & 0 & 0 & a_k \\ \hline & & & & \vdots & & & & \\ & & & & \vdots & & & & \\ & & & & \vdots & & & & \\ \hline d_k & 0 & 0 & 0 & \cdots & 0 & a_k & b_k & c_k \end{bmatrix} \qquad (9)$$

avec

$$a_k = (\beta_k + 3)^3 - 4(\beta_k + 2)^3 + 6(\beta_k + 1)^3 - 4\beta_k^3 ,$$

$$b_k = (\beta_k + 2)^3 - 4(\beta_k + 1)^3 + 6\beta_k^3 ,$$

$$c_k = (\beta_k + 1)^3 - 4\beta_k^3 ,$$

$$d_k = \beta_k^3 ,$$

$$\beta_k = \frac{kN}{M} \qquad k = 1, \ldots, \frac{M}{N} . \qquad (10)$$

## III - APPLICATION DE LA B-SPLINE CUBIQUE UNIFORME AU RECALAGE EN RESOLUTION SPATIALE D'IMAGES SATELLITES ( LANDSAT-SPOT ) :

L'image de la figure 1 est une fenêtre d'image de dimensions 85 x 64 pixels ( N' = 85 et N = 64 ) de la ville de Strasbourg prise par le satellite Landsat. La résolution spatiale de chaque pixel est de 79 m x 57m au sol.

L'image de la figure 2 est une image de référence de dimensions 256 x 256 pixels de la ville de Strasbourg prise par le satellite Spot et dont la résolution spatiale du pixel est de 20m x 20m au sol.

Le pas d'échantillonnage de l'image numérique Landsat est de 57m(=3 x 19m) suivant les colonnes et de 79m($\simeq$ 80m) suivant les lignes de la matrice correspondante à cette image. Celui de l'image de référence Spot est de 20m ($\simeq$ 19m) suivant les colonnes et de 20m suivant les lignes.

Pour obtenir l'image finale de la figure 3, on a ré-échantilloné l'image de la figure 1 avec un pas de 19m suivant les colonnes et de 20m suivant les lignes.

Les dimensions de l'image finale sont égales à celles de la matrice $[G']$ :

$$M' = N' \cdot \frac{57}{19} \qquad \text{et} \qquad M = N \cdot \frac{80}{20}$$

$$M' = 255 \qquad \text{et} \qquad M = 256$$

Les coefficients des matrices $[B]$ et $[B]^t$ sont calculés à partir de la relation (10). Dans le cas du ré-échantillonnage suivant les colonnes, les coefficients $a_k$, $b_k$, $c_k$ et $d_k$ ( $k = 1, \ldots, 3$ ) ont les valeurs suivantes :

$a_1 = \frac{8}{27}$ $\quad a_2 = \frac{1}{27}$ $\quad a_3 = 0$ $\quad b_1 = \frac{93}{27}$ $\quad b_2 = \frac{60}{27}$ $\quad b_3 = 1$

$c_1 = \frac{60}{27}$ $\quad c_2 = \frac{83}{27}$ $\quad c_3 = 4$ $\quad d_1 = \frac{1}{27}$ $\quad d_2 = \frac{8}{27}$ $\quad d_3 = 1$

On obtient une matrice $[B]$ de dimension 255 x 85.

$$[B] = \frac{1}{27} \begin{bmatrix} 60 & 1 & 0 & 0 & \ldots & 0 & 0 & 8 & 93 \\ 83 & 8 & 0 & 0 & \ldots & 0 & 0 & 1 & 60 \\ 108 & 27 & 0 & 0 & \ldots & 0 & 0 & 0 & 27 \\ \hline & & & & & & & & \\ & & & & \vdots & & & & \\ & & & & & & & & \\ \hline 1 & 0 & 0 & 0 & \ldots & 0 & 8 & 93 & 60 \\ 8 & 0 & 0 & 0 & \ldots & 0 & 1 & 60 & 83 \\ 27 & 0 & 0 & 0 & \ldots & 0 & 0 & 27 & 108 \end{bmatrix}$$

Figure 1 : Fenêtre d'image Landsat de dimensions (85x64 pixels) de la ville de Strasbourg.

Figure 2 : Image Spot de référence de dimensions (256x256 pixels) de la ville de Strasbourg.

Figure 3 : Image reconstruite de dimensions (255x256), par interpolation (méthode B-spline cubique uniforme) de l'image Landsat de la figure 1.

Par contre, dans le cas du ré-échantillonnage suivant les lignes, les coefficients $a_k$, $b_k$, $c_k$ et $d_k$ ( $k = 1, \ldots, 4$ ) ont des valeurs différentes et la matrice $[B]$ est de dimensions 256 x 64.

$$[B] = \frac{1}{64}\begin{bmatrix}
121 & 1 & 0 & 0 & \ldots & 0 & 0 & 27 & 235 \\
184 & 8 & 0 & 0 & \ldots & 0 & 0 & 8 & 184 \\
235 & 27 & 0 & 0 & \ldots & 0 & 0 & 1 & 121 \\
256 & 64 & 0 & 0 & \ldots & 0 & 0 & 0 & 64 \\
\hline
235 & 121 & 1 & 0 & \ldots & 0 & 0 & 0 & 27 \\
184 & 184 & 8 & 0 & \ldots & 0 & 0 & 0 & 8 \\
121 & 235 & 27 & 0 & \ldots & 0 & 0 & 0 & 1 \\
64 & 256 & 64 & 0 & \ldots & 0 & 0 & 0 & 0 \\
\hline
 & & & & \vdots & & & & \\
\hline
1 & 0 & 0 & 0 & \ldots & 0 & 27 & 235 & 121 \\
8 & 0 & 0 & 0 & \ldots & 0 & 8 & 184 & 184 \\
27 & 0 & 0 & 0 & \ldots & 0 & 1 & 121 & 235 \\
64 & 0 & 0 & 0 & \ldots & 0 & 0 & 64 & 256
\end{bmatrix}$$

Finalement, la relation (6) utilisera $[B]^t$ de cette dernière.

**CONCLUSION / :**

On remarque que l'image finale de la figure 3 a pratiquement la même résolution spatiale que l'image de la figure 2 du satellite Spot.

L'application de la fonction B-spline cubique uniforme donne un résultat satisfaisant pour ce cas de recalage en résolution spatiale d'images.

## IV- REFERENCES :


[1] - MAHBOUB Mourad " Traitement des courbes sur micro-ordinateur, méthode d'approximation et d'interpolation ", Thèse de Magister en Physique Electrique, Université de Tlemcen, Mai 1988.

[2] - WILLIAM K. PRATT, " Digital image processing " A Wiley-Interscience Publication 1978.

# Annexe Communications

1. M. Mahboub, and B. Philippe, " *Ré-échantillonnage d'images numériques par la B-spline cubique uniforme*," CARI'02 Yaoundé, pp. 183-188, Octobre 2002.

**UNIVERSITE DE YAOUNDE I**
THE UNIVERSITY OF YAOUNDE I

Département d'informatique
Department of Computer Science
B.P. 812 Yaoundé
CAMEROUN / **Tel: 7 76 26 34 / 7 80 32 71**


**FACULTE DES SCIENCES**
FACULTY OF SCIENCES

**6eme Colloque Africain sur la Recherche en Informatique CARI'02**

**6th African Conference on Research in Computer Science CARI'02**

# *ATTESTATION DE PARTICIPATION*

Je soussigné Emmanuel KAMGNIA, Président du Comité Local d'Organisation du 6ème Colloque Africain sur la Recherche en Informatique "CARI 02 " certifie que Monsieur **MAHBOUB Mourad** a assisté aux manifestations du 10ème anniversaire du CARI 02 à Yaoundé pendant la période du 10 au 18 octobre 2002, en qualité d'Auteur.

En foi de quoi, la présente attestation lui est délivrée pour servir et valoir ce que de droit.

Fait à Yaoundé le 15 Octobre 2002

**Le Président du Comité Local d'Organisation**

**Emmanuel KAMGNIA**

**Email** : cari_cam@uycdc.uninet.cm

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DES AFFAIRES ÉTRANGÈRES

The United Nations University

INRIA

IRD
Institut de recherche pour le développement

CIRAD

AUF

CIMPA ICPAM

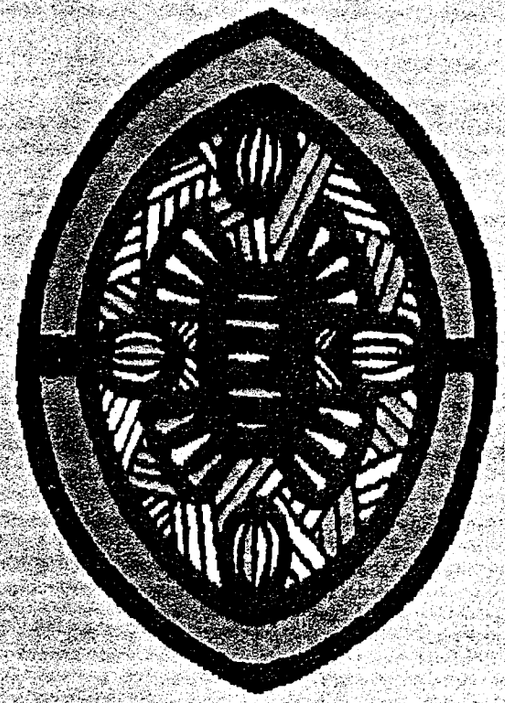
UNIVERSITÉ DE DOUALA

# PROGRAMME

# CARI'02

YAOUNDÉ (Cameroun) / *YAOUNDE (Cameroon)*
14 - 17 octobre 2002
*October 14 - 17, 2002*

**6ème Colloque Africain sur la Recherche en Informatique**
***6th African Conference on Research in Computer Science***



**Journées de formation avancée / *Tutorials***
**10 - 11 et 12 octobre 2002 / *October 10 – 11 and 12, 2002***

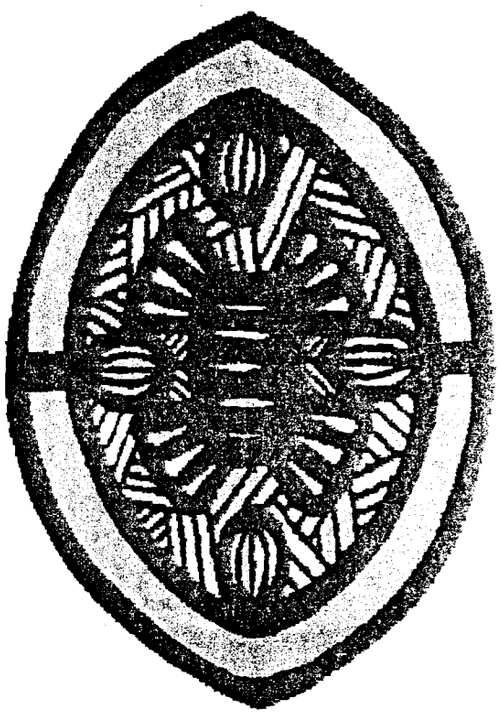
**http://www.cari-info.org**

# PROGRAMME

# CARI'02

YAOUNDÉ (Cameroun) / *YAOUNDE (Cameroon)*
14 - 17 octobre 2002
*October 14 - 17, 2002*

**6ème Colloque Africain sur la Recherche en Informatique**
***6th African Conference on Research in Computer Science***



**Journées de formation avancée / *Tutorials***
**10 - 11 et 12 octobre 2002 / *October 10 – 11 and 12, 2002***

**http://www.cari-info.org**
**http://conference.inria.fr/CARI02**

## Comité de programme / *Programme Committee*

### Présidents / *Chairs*

| | |
|---|---|
| L. ANDRIAMAMPIANINA | ESPA, Antananarivo, Madagascar |
| M. TCHUENTE | Université de Douala, Cameroun |

### Membres / *Members*

| | |
|---|---|
| F. ANDRÉ | IRISA/INRIA Rennes, France |
| R. ANDRÉ-OBRECHT | IRIT, Toulouse, France |
| J-P. ASSELIN DE BEAUVILLE | AUF, Montréal, Québec, Canada |
| E. BADOUEL | École Nationale Supérieure Polytechnique, Yaoundé, Cameroun |
| M-O. CORDIER | IRISA/INRIA Rennes, France |
| R. DERICHE | INRIA Sophia Antipolis, France |
| J-C. DERNIAME | LORIA/ENSEM, Vandoeuvre les Nancy, France |
| R. DIENG | INRIA Sophia Antipolis, France |
| C. FRASSON | Université de Montréal, Québec, Canada |
| F. GUERRIN | CIRAD, La Réunion |
| V. HARISON | INSCAE, Antananarivo, Madagascar |
| J-P. HATON | LORIA/INRIA, Vandoeuvre les Nancy, France |
| J-C. HOCHON | IXI, Toulouse, France |
| M. JAOUA | ENIT-Lamsin, Tunis, Tunisie |
| B. KERHERVÉ | Université du Québec, Montréal, Canada |
| T. LASKRI | Université d'Annaba, Algérie |
| Z. MAMMERI | IRIT/Université Paul Sabatier, Toulouse, France |
| N. MIKOU | Université de Bourgogne, Dijon, France |
| O. MONGA | IRD, France |
| M.T. NIANE | Université Gaston Berger, Saint Louis, Sénégal |
| J-P. NZALI | Université de Yaoundé 1, Cameroun |
| V. ORIA | New Jersey Institut of Technologie, Newark, USA |
| S. OUMTANAGA | INPHB, Yamoussoukro, Côte d'Ivoire |
| C. PETTANG | École Nationale Supérieure Polytechnique, Yaoundé, Cameroun |
| A. PIROTTE | Université de Louvain, Belgique |
| B. PLATEAU | ENSIMAG, Montbonnot Saint Martin, France |
| P. QUINTON | IRISA/IFSIC, Rennes, France |
| P. RENAUD | UNITAR, Genève, Suisse |
| P. ROLIN | France Télécom/CNET, Issy les Moulineaux, France |
| H. SAHRAOUI | Université du Québec, Montréal, Canada |
| I. SAHKO | Université de Metz, France |
| M. SELLAMI | Université d'Annaba, Algérie |
| A. SEZNEC | IRISA/INRIA Rennes, France |
| K. TOMBRE | LORIA/INPL, Vandoeuvre les Nancy, France |
| C. VIHO | IRISA/Université de Rennes 1, France |
| F. VOISIN | LRI/Université Paris Sud, Orsay, France |





## Comité permanent / *Permanent Committee*

### Représentants des chercheurs africains / *African Researchers Delegates*

| | |
|---|---|
| K. E. AKOUSSAH | Université du Benin, Lomé, Togo |
| L. ANDRIAMAMPIANINA | ESPA, Antananarivo, Madagascar, **Président / *Chair*** |
| E. M. DAOUDI | Université d'Oujda, Maroc |
| E. KAMGNIA | Université de Yaoundé 1, Cameroun |
| G. K. J. KOUACOU | INPHB, Yamoussoukro, Côte d'Ivoire |
| M. K. LUHANDJULA | Université de Kinshasa, R. D. Congo |
| N. R. RAZAFINDRAKOTO | CNRIT, Antananarivo, Madagascar |
| R. SHUMBA-MAZHINDY | University of Zimbabwe |
| Y. SLIMANI | Université de Tunis, Tunisie |
| Z. ZEMIRLI | INI, Alger, Algérie |

### Représentants des institutions / *Institutions Delegates*

| | |
|---|---|
| J-P. ASSELIN DE BEAUVILLE | AUF, Paris, France |
| C. LOBRY | CIMPA, Nice, France |
| B. PHILIPPE | INRIA Rennes, France - **Secrétaire / *Secretary*** |
| B. PONIATOWSKI | UNU, Tokyo, Japon |
| J. SOR | CIRAD, Montpellier, France |
| J-P. TREUIL | IRD, Bondy, France |

## Coordination

| | |
|---|---|
| M-C. SANCE-PLOUCHART | IRISA/INRIA Rennes, France |
| V. VERDON | IRISA/INRIA Rennes, France |

## Comité local d'organisation / *Local Organization Committee*

Monsieur le Recteur de l'Université de Yaoundé I, **Superviseur**
M. Emmanuel KAMGNIA, Chef de Dpt. d'Informatique, FSC UYI, **Président**
M. Basile LOUKA, Dpt. d'Informatique, FSC UYI, **Vice Président**
M. Jean Pierre NZALI, Dpt. d'Informatique, FSC UYI, **Secrétaire**
M. Eric BADOUEL, Chef de Dpt. de Génie Informatique, ENSP, UYI, **Secrétaire Adjoint**
M. Lot TCHEEKO, Dpt. de Génie Informatique, ENSP, UYI, **Trésorier**
M. Claude TANGHA, Dpt. de Génie Informatique, ENSP, UYI, **Commissaire aux Comptes**
Mme FOTSO Pauline Laure, Dpt. d'Informatique, FSC UYI, **Relations publiques**
M. FOUDA NDJODO Marcel, Dpt. d'Informatique, FSC UYI, **Logistique**

## Information

**cari02@inria.fr**
**http://www.cari-info.org**
**http://conference.inria.fr/CARI02**

## Session 6A

### Traitement et analyse d'images II / *Image Processing and Analysis II*

17h00 - 17h30 — Détection et suivi des bords de routes basés sur la transformée de Radon
R. NOURINE[1,2], M. ELARBI BOUDIHIR[3], K. SAMIA[4]
([1]Université d'Oran, Algérie / [2]Université Djilalli Liabess, Sidi Bel Abbes, Algérie / [3]M. Ibn Saud University, Riyadh, KSA / [4]King Faisal University, Demmam, KSA)

17h30 - 18h00 — Extraction of linear features in SAR images using localized Radon transform for geographical map updating
V. P. ONANA[1,2,3], E. TROUVE[3], J. P. RUDANT[2], E. TONYE[1]
([1]École Supérieure Polytechnique, Yaoundé, Cameroun / [2]Université de Marne La Vallée, France / [3]ESIA – Université de Savoie, Annecy, France)

18h00 - 18h30 — Estimation du mouvement cardiaque par un système de maillage actif
D.DESSEREE, G. HARMENIL, L. LEGRAND, P.WALKER, A. LALANDE, F. BRUNOTTE
(LPPCE – Faculté de Médecine, Dijon, France)

## Session 6B

### Enseignement Assisté par Ordinateur (EAO) / *Computer Aided Teaching (CAT)*

17h00 - 17h30 — CHELIA : Un environnement coopératif pour l'apprentissage sur Internet
A. ZIDANI[1], M. DJOUDI[2], S. ZIDAT[1], S. TALHI[1]
([1]Université de Batna, Algérie / [2]SIC IRCOM Labs, Futuroscope, France)

17h30 - 18h00 — Un système tuteur utilisant un compagnon perturbateur
T. BOUHADADA, M.T. LASKRI
(Université d'Annaba, Algérie)





## Mercredi 16 octobre 2002 / *Wednesday, October 16, 2002*

08h30 - 09h30 — **Conférence invitée / *Invited lecture***
Modèles individu-centrés, modèles EDP : différences et complémentarité. L'exemple de l'alignement dans un banc de poissons
O. ARINO
(Université de Pau et des Pays de l'Adour, Pau, France)

## SESSIONS PARALLÈLES / *PARALLEL SESSIONS*

## Session 7A

### Traitement et analyse d'images III / *Image Processing and Analysis III*

09h30 - 10h00 — Ré-échantillonnage d'images numériques par la B-spline cubique uniforme
M. MAHBOUB[1], B. PHILIPPE[2]
([1]Université Abou Bakr Belkaid, Tlemcem, Algérie / [2]IRISA-INRIA, Rennes, France)

10h00 - 10h30 — Combinaison de classifieurs. Une approche pour l'amélioration de la classification d'images
S. CHITROUB, A. HOUACINE, B. SANSAL
(USTHB, Alger, Algérie)

10h30 - 11h00 — Pause / *Break*

## Session 7B

### Reconnaissance de texte et de la parole / *Text and Speach Recognition*

09h30 - 10h00 — Deux méthodes morpho-lexicales pour la correction des mots arabes issus des systèmes OCR
T. SARI, M. SELLAMI
(LRI – Université Badji Mokhtar, Annaba, Algérie)

10h00 - 10h30 — Introduction de l'énergie et de la durée dans un modèle de Markov caché
A. YOUSFI, A. MEZIANE
(Université Mohamed Ier, Oujda, Maroc)

10h30 - 11h00 — Pause / *Break*