Parallélisation du ré-échantillonnage d'images

Mourad Mahboub* - Bernard Philippe**

* Université Abou Bakr Belkaid, Faculté des Sciences, Tlemcen, ALGERIE.

m_mahboub@mail.univ-tlemcen.dz

** ENIT/LAMSIN, Tunis, TUNISIE et IRISA -INRIA Rennes, FRANCE.

philippe@irisa.fr

RÉSUMÉ. Dans le ré-échantillonnage d'une image numérique par la B-spline cubique uniforme, un pixel de sortie est calculé par un filtre appliqué sur 16 pixels voisins de l'image ou plus exactement d'une certaine matrice auxiliaire C. La matrice C est obtenue en un temps de calcul proportionnel au nombre de pixels de l'image à ré-échantillonner et ce temps est nettement inférieur au temps de calcul du filtre des 16 pixels voisins. Nous présentons, dans cet article, une parallélisation du calcul du filtre. Cette approche parallèle nous a permis de réduire le temps global de calcul grâce en particulier à une meilleure gestion de la mémoire.

ABSTRACT. When resampling a digital image by uniform cubic B-splines, an output pixel is computed from a filter applied to 16 neighboring pixels of the original image or more exactly of some auxiliary matrix C. Matrix C is computed with a computational time proportional to the number of pixels of the image to be resampled ; that time is substantially smaller than the computational time of the filtering part. In this paper, we present a parallel computation of the filter. This parallel approach reduces the global computational time partly from a better memory management.

MOTS-CLÉS : Ré-échantillonnage d'images, B-spline cubique uniforme, interpolation, filtre des 16 pixels, accélération, efficacité.

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

1. Introduction

Dans le 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 alors généralement pas avec la grille de l'image d'entrée. Un algorithme de ré-échantillonnage permet d'obtenir la valeur de ces nouveaux pixels. L'interpolation idéale ou de Shannon permet de reconstruire l'image exacte, mais elle est difficile à réaliser [1, 8]. Le ré-échantillonnage par la B-spline cubique uniforme utilise un filtre appliqué sur seize pixels voisins, d'une image (ou matrice) C intermédiaire, de même dimension nxn que l'image d'entrée, pour calculer les pixels de sortie. Nous avons montré [5] que le coût de l'algorithme de calcul de C est en $O(n^2)$. 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.

Nous présentons, dans cet article, un calcul parallèle du filtre de ré-échantillonnage des seize pixels voisins qui réduit le temps de calcul à un temps inférieur à celui du calcul de la matrice C.

2. Ré-échantillonnage par la B-spline cubique uniforme

L'ensemble des étapes est décrit dans [5] et nous ne reprenons ici que les éléments nécessaires à la compréhension de l'organisation des calculs ; sans perte de généralité, nous nous restreignons aux images carrées. La fonction de base de la B-spline cubique uniforme, s(x) est exprimée par:

$$s(x) = x_+^3 - 4(x-\Delta x)_+^3 + 6(x-2\Delta x)_+^3 - 4(x-3\Delta x)_+^3 \quad (1)$$

avec

$$x_+^3 = \begin{cases} x^3 & si\ x > 0, \\ 0 & si\ x \le 0 \end{cases} \quad (2)$$

L'image reconstruite g' peut s'écrire par sous la forme suivante:

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

L'image numérique ré-échantillonnée G' est de dimension MxM, où M=(m+1)n pour m nouveaux pixels calculés dans chaque direction. Elle vérifie l'expression matricielle suivante:

$G' = B\ C\ B^T$, (4)

où la matrice B est une matrice (M,n) de Toeplitz par blocs et la matrice C est une matrice auxiliaire d'ordre (n,n) qui est caractérisée par l'image numérique d'entrée G. On montre dans [5], comment la matrice C peut être calculée en $20\ n^2+O(n)$ opérations.

L'application de la matrice B à la matrice C suivant (4) peut être réalisée grâce au filtre $F_{k,u}$ de ré-échantillonnage des 16 pixels voisins de la B-spline cubique uniforme [6], défini par

$$F_{k,u} = \begin{bmatrix} a_k a_u & a_k b_u & a_k c_u & a_k d_u \\ b_k a_u & b_k b_u & b_k c_u & b_k d_u \\ c_k a_u & c_k b_u & c_k c_u & c_k d_u \\ d_k a_u & d_k b_u & d_k c_u & d_k d_u \end{bmatrix} \quad (5)$$

avec $(k, u) \in [0,\ m]^2$ et où les coefficients sont définis par

$$a_k = (\beta_k + 3)^3 - 4(\beta_k + 2)^3 + 6(\beta_k + 1)^3 - 4\beta_k^3$$
$$b_k = (\beta_k + 2)^3 - 4(\beta_k + 1)^3 + 6\beta_k^3$$
$$c_k = (\beta_k + 1)^3 - 4\beta_k^3$$
$$d_k = \beta_k^3$$
$$\beta_k = kn / M,\ k \in [0, m]$$

Ce filtre permet de calculer m pixels de sortie dans chaque direction. Il est appliqué sur la fenêtre $C_{i,j}$ de dimension 4 x 4 de la matrice C :

$$C_{i,j} = \begin{bmatrix} c_{i-1,j-1} & c_{i-1,j} & c_{i-1,j+1} & c_{i-1,j+2} \\ c_{i,j-1} & c_{i,j} & c_{i,j+1} & c_{i,j+2} \\ c_{i+1,j-1} & c_{i+1,j} & c_{i+1,j+1} & c_{i+1,j+2} \\ c_{i+2,j-1} & c_{i+2,j} & c_{i+2,j+1} & c_{i+2,j+2} \end{bmatrix} \quad (6)$$

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

Cette opération est connue sous le nom de « méthode des stencils » ou méthode de Kernel [4]. La complexité Comp (F) de l'application du filtre des 16 pixels dépend du nombre m des pixels de sortie dans chaque direction ; elle est égale à :

$$\text{Comp}(F_m) = (31\ m^2 + 46\ m)\ n^2 + O(n) \quad (7)$$

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

$$R(m) = \frac{Comp(C_m)}{Comp(C_m) + Comp(F_m)} = \frac{20}{31m^2 + 46m + 20} + O(\frac{1}{n}) \quad (8)$$

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. Ce rapport exprime le rapport des volumes de calcul de chaque partie mais pas nécessairement exactement celui des temps de calcul, car la vitesse des calculs peut varier suivant le type des opérations et l'accès aux données.

3. Comparaison expérimentale des temps de calcul séquentiels

n	8	16	32	64	128	256	512	1024	2048
Construction 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 1 : *Temps de calcul (s) de la construction de C et de plusieurs filtres.*

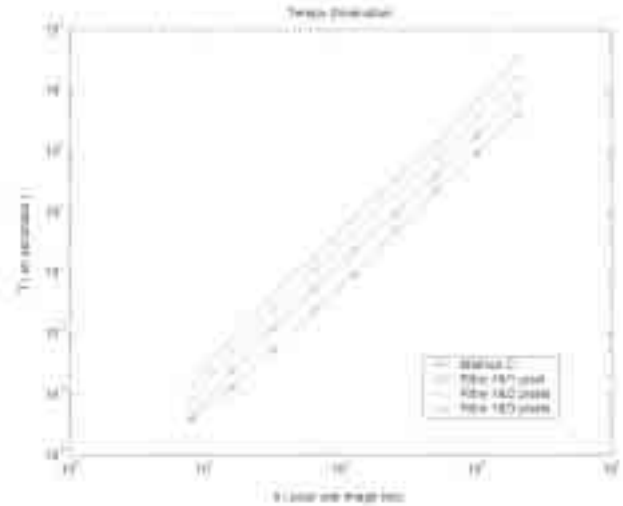

Figure 1. *Temps de calcul (s) de la construction de C et de plusieurs filtres.*

Les tests en monoprocesseur 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 1 et la figure 1).

4. Calcul parallèle du filtre des 16 pixels

Nous considérons maintenant un réseau linéaire 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 (figure 2a). Chaque processeur calcule les coefficients du filtre et en même temps transmet à ses voisins les trois lignes frontière de sa partie de la matrice C (figure 2b). Enfin, chaque processeur applique le filtre sur sa partie de la matrice C.

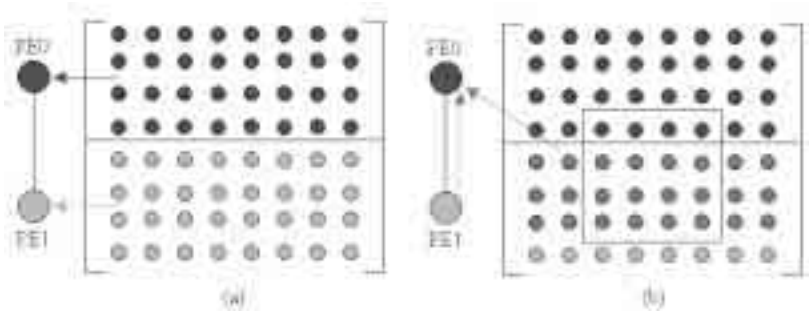

Figure 2. *(a) Distribution de la matrice C à deux processeurs; (b) Communications inter-processeurs du filtre des 16 pixels.*

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) [2, 7] avec le mode SPMD (Single Program Multiple Data) [3].

L'approche parallèle du filtre des 16 pixels nous a permis de réduire et d'obtenir des temps de calcul inférieurs à celui de la construction de la matrice C (voir le tableau 2). La figure 3 illustre l'évolution des temps d'exécution et la figure 4 celle des accélérations et efficacités résultantes. Les courbes de l'accélération S et de l'efficacité E de la figure 4 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.

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

Tableau 2. *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.*

5. Conclusion

Nous avons présenté, dans cet article, 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 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. Pour améliorer l'efficacité, il faudrait maintenant paralléliser cette partie, mais on ne peut en espérer un très grand bénéfice.

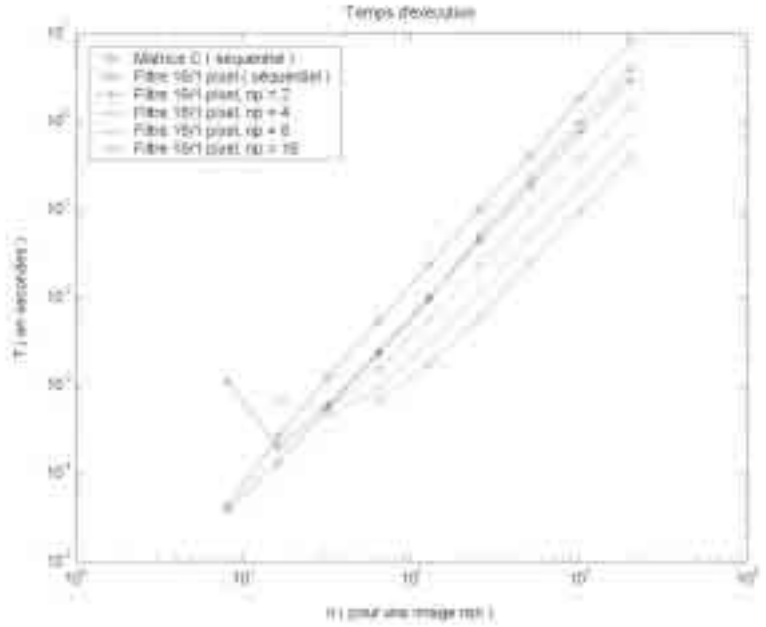

Figure 3. *Temps de calcul de la matrice C en séquentiel et du filtre 16/1 pixel pour np = 1, 2, 4, 8 et 16 processeurs.*

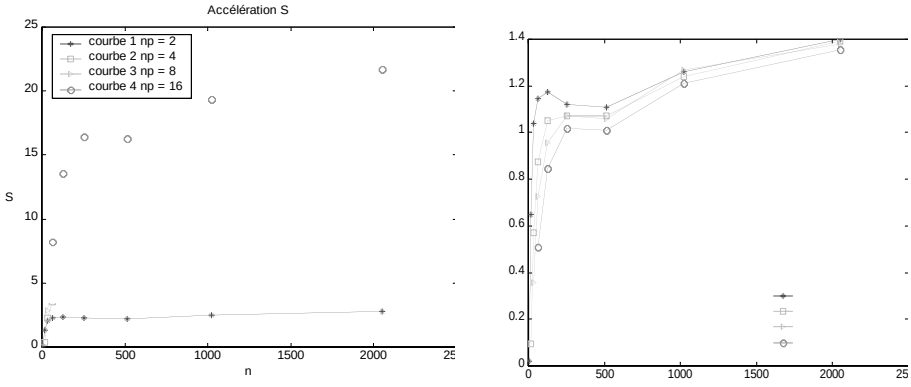

Figure 4. *Accélération S et efficacité E du ré-échantillonnage par la B-spline cubique uniforme pour 1 pixel de sortie et pour np =2, 4, 8 et 16 processeurs courbes.*

6. Références

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

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

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

[4] James H. A., Patten C. J. and Hawick K. A., « Stencil Methods on distributed high performance computers », *Technical note DHPC-010, department of computer science, University of Adelaide,* Australia, June 1997.

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

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

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

[8] Pratt William K., « Digital image processing », *A Wiley Interscience Publication* 1978.