Sciences & Technologie **B** – **N°21**, Juin (2004), pp. 63-70.

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 Δx et Δy 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:

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

ω_1 et ω_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]:

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

avec:

$$a_k = (\beta_k + 3)^3 - 4(\beta_k + 2)^3 + 6(\beta_k + 1)^3 - 4\beta_k^3$$
$$b_k = (\beta_k + 2)^3 - 4(\beta_k + 1)^3 + 6\beta_k^3$$
$$c_k = (\beta_k + 1)^3 - 4\beta_k^3 \qquad (11)$$
$$d_k = \beta_k^3$$
$$\beta_k = (k-1)n/M,\ k \in [1, M/n]$$

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:

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

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 λ 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:

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

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:

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

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:

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

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

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:

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

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

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.

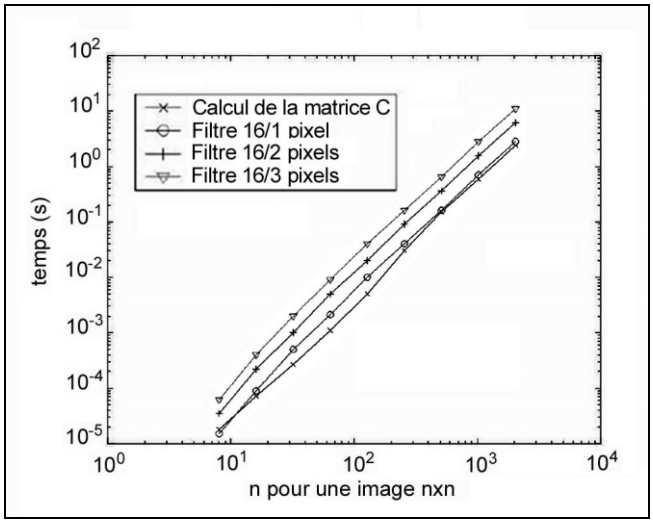


Figure 1: Temps d'exécution de la matrice C (—×—), du filtre des 16 *pv* pour un pixel de sortie (—o—), du filtre des 16 *pv* pour deux pixels de sortie (—+—) et du filtre des 16 *pv* pour trois pixels de sortie (—∀—).

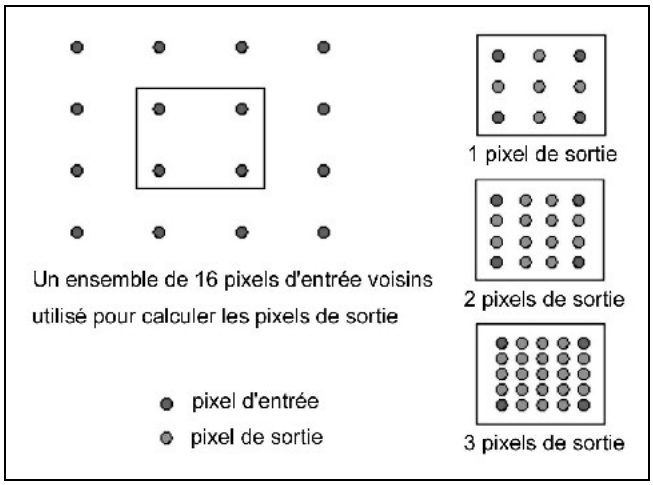


Figure 2: Ré-échantillonnage d'images avec 1 pixel, 2 pixels et 3 pixels de sortie.

On vérifie bien grâce au graphique log-log que les temps d'exécution peuvent être estimés par des formules du type $t_n = Tn^{\alpha}$ 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

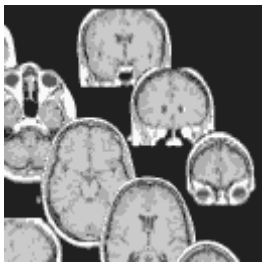
Tableau 1: Régression linéaire pour n = 8, 16, 32, 64, 128, 256, 512, 1024, 2048.

On remarque que l'ordre α 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.

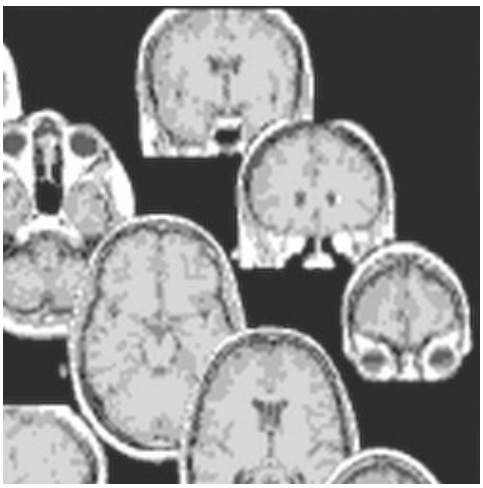
La figure 3 représente le ré-échantillonnage d'une image réelle de 128x128 pixels pour 1, 2 et 3 pixels de sortie. Nous remarquons un lissage assez important des images de sortie.

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.

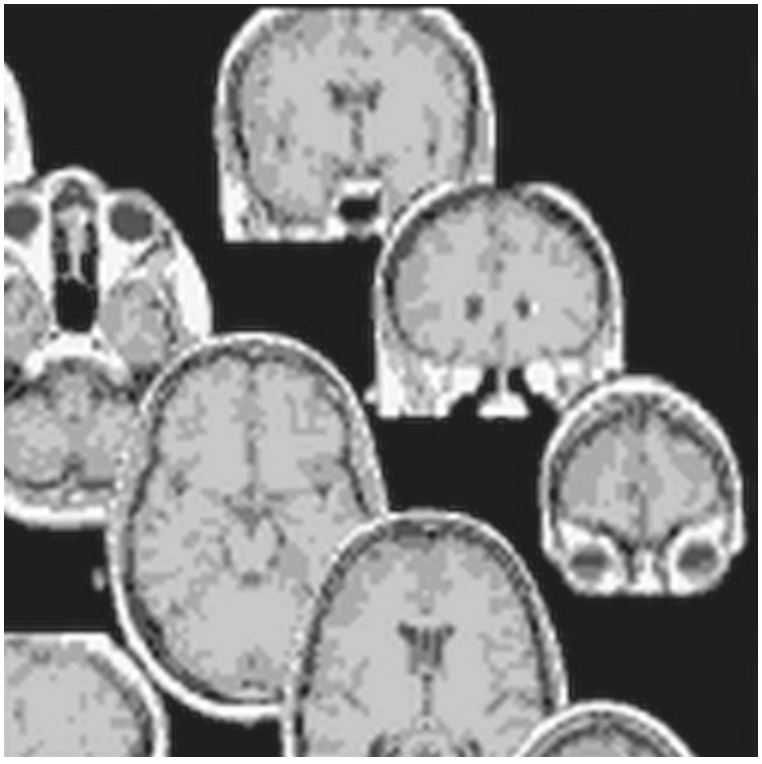
Le ré-échantillonnage d'images par la méthode du plus proche voisin est très rapide et conserve les valeurs



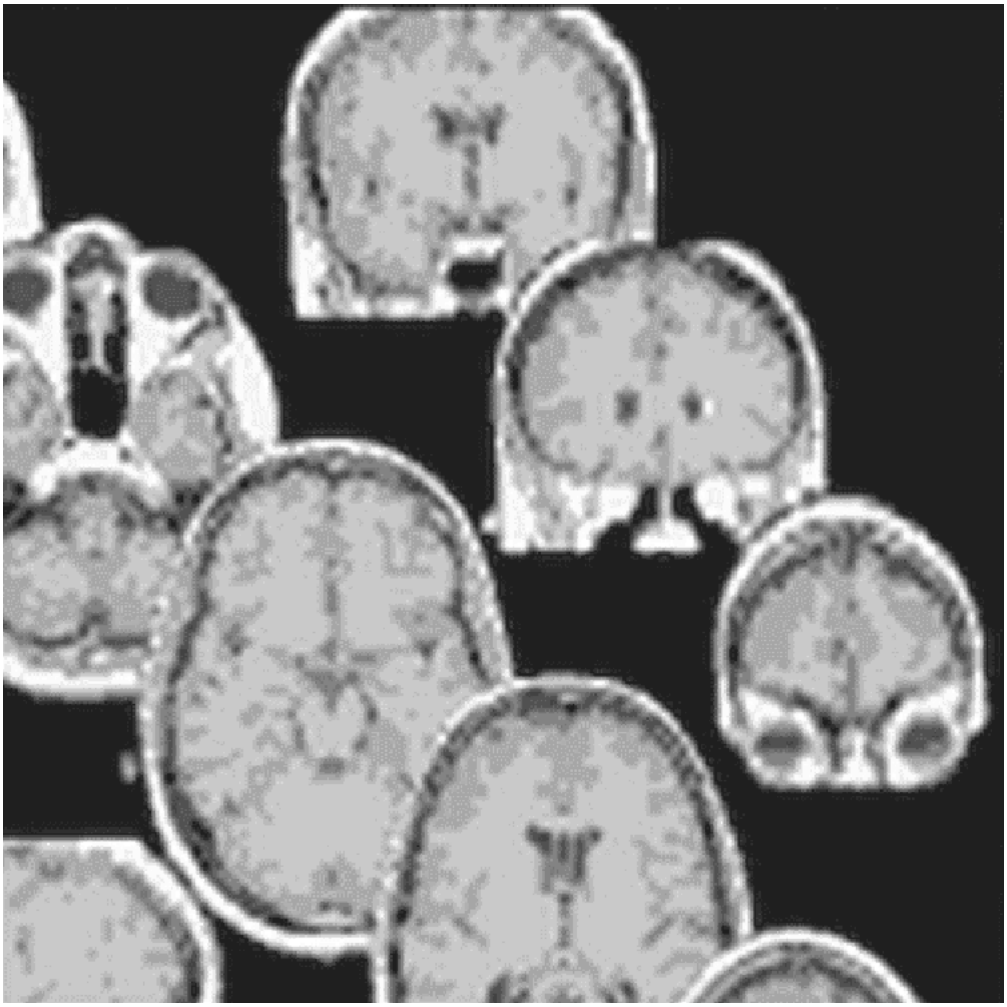
(a)



(b)



(c)

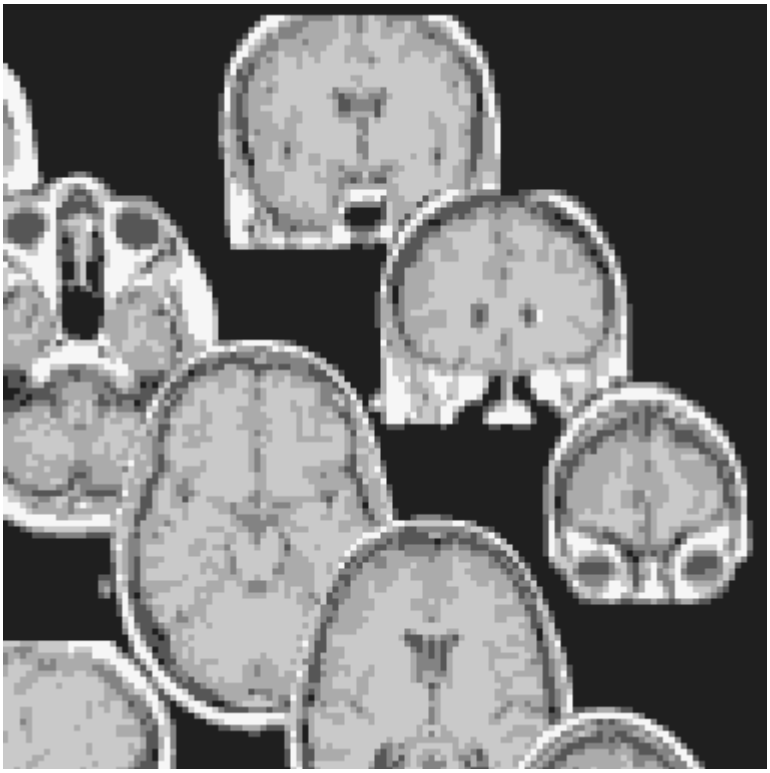


(d)

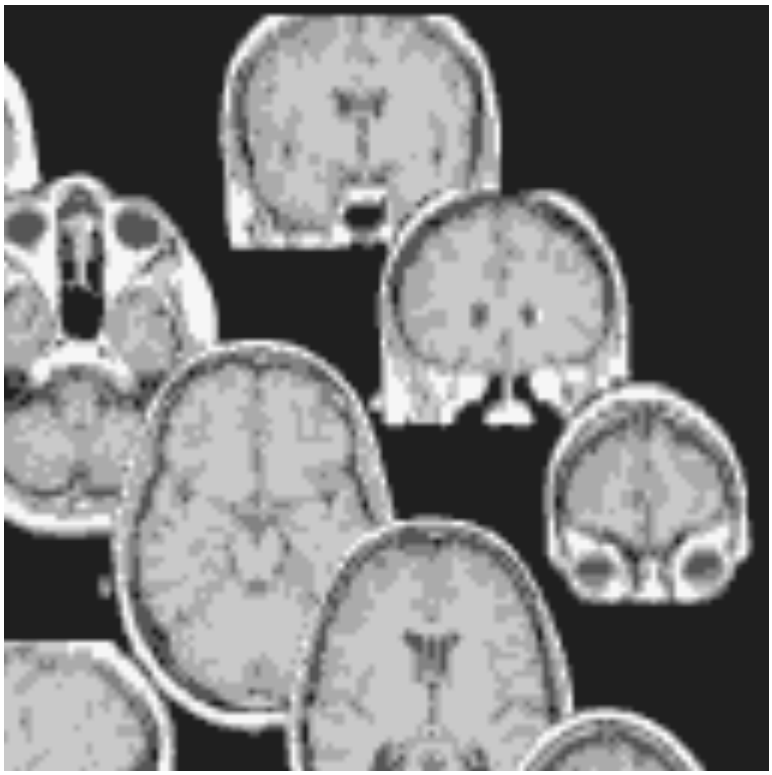
Figure 3 : (a) L'image originale avec 128x128 pixels, **(b)** L'image ré-échantillonnée avec un pixel de sortie, le temps de calcul est estimé à 0.014 s, **(c)** L'image ré-échantillonnée avec deux pixels de sortie, le temps de calcul est estimé à 0.026 s, **(d)** L'image ré-échantillonnée avec trois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

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. □