Une technique d'accélération pour la compression fractale d'images

Eric Amram[1] **et Jacques Blanc-Talon**[2]

[1] ENSTA,
32, bd Victor,
75015 Paris, France

[2] CTME/GIP,
16 bis, av. Prieur de la Côte d'Or,
94114 Arcueil, France
blanc@etca.fr

RÉSUMÉ

Les techniques fractales de compression d'images souffrent encore aujourd'hui de temps de codage très importants. Nous proposons ici un nouvel algorithme récursif d'optimisation, couplé à un algorithme de compression de type Jacquin. Un prédicat numérique est utilisé pour déterminer quel bloc candidat peut correspondre à un bloc initial donné, ce qui permet d'éviter des comparaisons coûteuses en temps de calcul. La méthode permet d'obtenir des accélérations considérables pour un prédicat simple comme l'inclusion d'histogrammes des niveaux de gris. Elle est de surcroît compatible avec d'autres méthodes d'accélération.

ABSTRACT

Fractal image compression still suffers from a (sometimes very) high encoding time, depending on the approach being used. Here, we propose a new recursive algorithm based on Jacquin's general scheme. A numerical predicate is used for guessing which domain block may possibly match a given range block. For a simple predicate based on histograms comparison, dramatic accelerations of the encoding process have been measured: the speedup gains prove the interest of the method. An important advantage of our technique is that it is compatible with other acceleration techniques. Results on generic images are shown.

1 Introduction

Depuis les travaux novateurs de Barnsley *et al.* [1] concernant l'introduction de la théorie des IFS en compression d'images, d'importants progrès ont été accomplis permettant d'envisager des applications pratiques. Aujourd'hui, 3 approches principales restent en lice (voir [2, 6, 7, 8] ou [3]) dont la méthode de Jacquin, qui est la plus populaire. De nouvelles classes de transformations ont été introduites afin d'améliorer la qualité des images, comme les isométries de carrés [13] ou des modifications de l'algorithme initial [17]. Toutefois, malgré des réductions importantes des temps de compression, les méthodes restent très lentes en raison de la nécessité de balayer toute l'image.

Il existe deux types de techniques d'accélération selon qu'elles engendrent une perte de qualité de l'image comme dans [10, 14] ou celle présentée ici, ou pas [4, 12, 16].

Dans l'algorithme de Jacquin [9], l'image est divisée en blocs source et cible dont on recherche les meilleurs appariements à une réduction, isométrie et transformation affine de luminance près (fig. 1). L'application successive du *Collage theorem* et du *Contraction mapping theorem* permet de démontrer que l'application itérée de ces transformées à n'importe quelle image initiale non nulle permet de retrouver l'image source à une certaine distance de Hausdorff connue, et donc de remplacer l'image par l'ensemble des transformations.

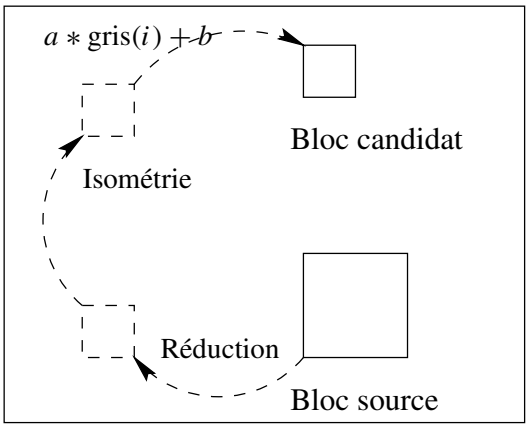

FIG. 1 — Du bloc source au bloc cible

Dans notre application, la recherche est effectuée sur l'image réduite entière, et non seulement dans un voisinage ; nous utilisons des blocs disjoints, mais rien n'empêche de faire le contraire. La décompression est une simple affaire d'itération des transformées jusqu'à une convergence suffisante, ce qui intervient toujours rapidement.

2 L'algorithme QSA

L'idée du Quick-Search est d'effectuer une recherche dichotomique approchée des appariements en utilisant un partitionnement en *quad-tree* de l'image. Pour ce faire, il faut trouver un prédicat d'images qui introduise un ordre partiel. On considère d'abord l'image entière réduite et on demande :

Est-ce qu'elle contient des blocs cible potentiels ?

Si oui, le bloc est décomposé en 4 parties qui correspondent à la position possible en haut à gauche du bloc cible (fig. 2). En

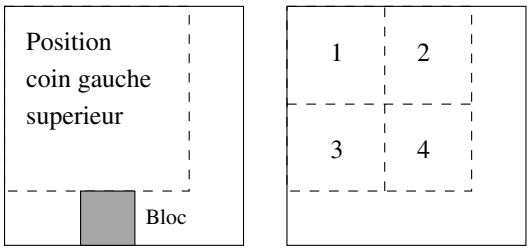

FIG. 2 — Le quadtree des histogrammes

fait, le couple de toute fonction monotone d'un ensemble de valeurs de niveaux de gris dans $\mathbf{R}^+$ et d'un ordre partiel associé peut être un prédicat. Nous utilisons un prédicat basé sur une comparaison d'histogrammes, comme expliqué ci-après.

Considérons une image et son histogramme des niveaux de gris. Pour un nombre de carrés donné, on divise l'ensemble des valeurs de gris en intervalles. Chaque carré représente le nombre de niveaux de gris dans cet intervalle. Comme

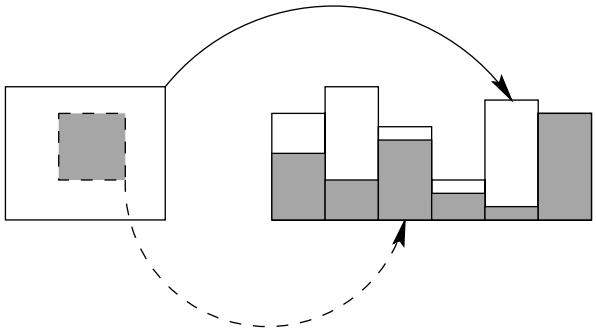

FIG. 3 — Blocs et inclusion d'histogrammes

la transformation des luminances est une application affine, l'ensemble des niveaux de gris d'un bloc doit être normalisé avant le calcul de l'histogramme.

Le prédicat s'en déduit assez simplement : un bloc cible appartient à une partie de l'image réduite si son histogramme est "inclus" dans l'histogramme de cette partie (fig. 3). Si cela est vérifié, l'espace de recherche correspondant au coin supérieur gauche est subdivisé. Ceci bien sûr n'est valide tant que nous n'utilisons que des isométries : aucune approximation ou filtrage du bloc n'est effectuée après réduction de l'image.

En pratique, on précalcule le *quad-tree* des histogrammes des blocs source pour une taille de bloc donnée : la racine est l'image toute entière tandis que les feuilles sont les blocs source réels (fig. 2). L'histogramme de chaque bloc cible est ensuite calculé à chaque passage dans la boucle principale ; il est comparé aux nœuds des *quad-trees* des blocs source de manière récursive.

Cela conduit à un petit nombre de possibilités dont les scores sont calculés en utilisant l'estimateur quadratique habituel. L'algorithme QSA étant ainsi compatible avec d'autres techniques comme [11] et [16], nous pouvons espérer des temps de compression très faibles au moyen d'approches combinées.

Nous attirons l'attention sur le fait que rien n'empêche d'utiliser des blocs source non disjoints. Cela n'a pas été testé ici, mais nous envisageons de le programmer dans la prochaine version afin d'améliorer le SNR (*Signal to Noise Ratio*).

Notre méthode serait absolument sans perte si les appariements étaient corrects. Or en pratique, seuls les petits blocs (3×3) respectent cette condition et une technique de minimisation de l'erreur quadratique doit être appliquée. De manière analogue, il faut donc permettre dans la comparaison des histogrammes une certaine flexibilité ; la possibilité d'avoir des intervalles se recouvrant (fig. 4) nous a semblée être le meilleur compromis. En effet, une trop grande flexibilité amènerait à comparer un grand nombre de blocs et rendrait la technique d'accélération totalement inefficace.

FIG. 4 — Histogrammes recouvrants

La complexité obtenue est de $O(\log X \times \log Y)$ au lieu de $O(X \times Y)$ comparaisons, où $X = \frac{largeurd'image}{tailledubloc}$. Les comparaisons ne sont effectuées qu'entre histogrammes jusqu'à ce que les positions des blocs soient trouvées, ce qui veut dire qu'elles sont faites à temps constant quelles que soient les tailles des blocs.

Cet algorithme est facilement parallélisable, même au niveau de la recherche rapide qui est pourtant récursive.

3 Conclusion et résultats

Dans cette version les blocs source ne se recouvrent pas, ce qui ne permet pas d'atteindre la meilleure qualité possible. Notre but était avant tout de montrer l'intérêt de cette technique d'accélération. Dans la comparaison, la recherche extensive (ou FSA, *Full Search Algorithm*) a été entièrement optimisée afin de mesurer véritablement le gain obtenu.

Les bons appariements restent aussi bons qu'ils étaient. Par contre, les erreurs ont tendance à s'aggraver : cela explique les différences notables survenant à des grandes tailles de blocs cible entre le SNR de FSA et celui de QSA. Un filtre 3×3 passe-bas a été appliqué afin de faire disparaître certains artefacts (quand la taille du bloc est supérieure à 4). Les temps CPU sur les tableaux 1 et 2 ont été mesurés avec la commande

FIG. 5 — Image originale 512 × 512

Unix /usr/5bin/time. Tous les calculs présentés ont été réalisés sur une station SUN Hyper-Sparc mono-processeur.

Cette nouvelle technique a montré qu'il était possible de franchir un ordre de grandeur dans le temps de compression d'une image sans perte globale significative de la qualité. Nous cherchons maintenant de meilleurs prédicats, lesquels, couplés avec des rotations et des intervalles recouvrants, devraient permettre d'augmenter également la qualité d'un ordre de grandeur. Toutefois, afin de rester objectif, il convient de mentionner des études critiques comme [5, 15].

Le principe de cette technique provient d'une idée originale de Seth Teller (M.I.T.) et Michael Werman (Hebrew University) pour trouver la correspondance entre 2 images.

Références

[1] M. Barnsley. *Fractals everywhere*. Academic Press, 1988.

[2] M. F. Barnsley and Hurt L. P. *Fractal Image Compression*. Wellesley, MA : A.K. Peters, 1993.

[3] Jacques Blanc-Talon. Une revue des techniques de compression fractale d'images. Technical report, CTME-GIP, 1997.

[4] Kenneth M. Dawson-Howe. Lossless image compression using a simple prediction method. Technical report, Trinity College, 1996.

[5] F.M. Dekking. Fractal image coding : some mathematical remarks on its limits and its prospects. Technical Report 95-95, TU Delft, 1995.

[6] Y. Fisher (Ed.). *Fractal Image Compression : Theory and Application*. Springer-Verlag, New York, 1995.

[7] Bruno Forte and Edward R. Vrscay. Inverse problem methods for generalized fractal transforms. Technical report, University of Waterloo, Canada, March 1996.

FIG. 6 — Lena - Quick-Search, blocs 8 × 8

FIG. 7 — Lena - Quick-Search, blocs 6 × 6

FIG. 8 — Lena - Quick-Search, blocs 4 × 4

TAB. 1 — Temps CPU et SNR : FSA vs QSA

bloc cible	**Lena** 512×512		
	QSA	FSA	$\frac{t_F}{t_Q}$
4×4	36min 19s 34.5 dB	261min 34.5 dB	**7.2**
6×6	21min 01s 30.7 dB	153min 12s 31.9 dB	**7.3**
8×8	13min 16s 28.6 dB	102min 23s 29.5 dB	**7.7**
12×12	4min 24s 25.3 dB	37min 26.6 dB	**8.4**
16×16	2min 04s 23.2 dB	18min 46s 24.6 dB	**9.1**
20×20	1min 01s 21.6 dB	11min 41s 23.2 dB	**11.5**
24×24	34 sec 20.7 dB	8min 21s 22.1 dB	**14.7**
32×32	20 sec 19.4 dB	5min 28s 20.1 dB	**16.4**

TAB. 2 — Temps CPU et SNR : FSA vs QSA

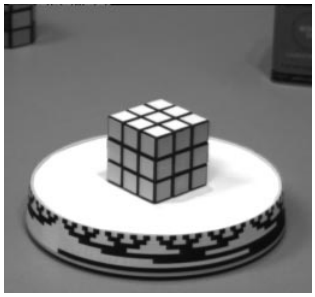

Image Rubic originale 256×240

bloc cible	**Rubic** 256×240		
	QSA	FSA	$\frac{t_F}{t_Q}$
4×4	11.4s 31.8 dB	5min 18s 31.8 dB	**27.9**
6×6	5.3s 26.8 dB	2min 41s 28.4 dB	**30.4**
8×8	3.1s 23.3 dB	1min 41s 23.6 dB	**32.6**
16×16	0.8s 18.2 dB	28.7s 20.0 dB	**35.9**

[8] Bruno Forte and Edward R. Vrscay. Theory of generalized fractal transforms. Technical report, University of Waterloo, Canada, March 1996.

[9] A.E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1) :18–30, January 1992.

[10] H. Lin and A.N. Venetsanopoulos. Fast pyramidal search for perceptually based fractal image compression. In *ICIP*, volume 3, pages 596–599, Washington, USA, October 1995.

[11] H. Lin and A.N. Venetsanopoulos. A pyramid algorithm for fast fractal image compression. In *IEEE International Conference on Image Processing*, volume 3, pages 596–599, Washington, USA, October 1995.

[12] David John Nettleton and Roberto Gari gliano. Reductions in the search space for deriving a fractal set of an arbitrary shape. *Jour. of Mathematical Imaging and Vision*, 1996.

[13] Dan Popescu, Alex Dimca, and Hong Yan. Generalized square isometries - an improvement for fractal image compression. In *8th Int. Image Conf. on Image Analysis and Processing*, San Remo (It), 1995.

[14] Dietmar Saupe. Accelerating fractal image compression by multi-dimensional nearest neighbor search. In *Data Compression Conference*, Freiburg, Germany, March 1995. IEEE.

[15] Dietmar Saupe. The futility of square isometries in fractal image compression. In *International Conference on Image Processing*, Lausanne, September 1996.

[16] Dietmar Saupe and Hannes Hartenstein. Lossless acceleration of fractal image compression by fast convolution. In *International Conference on Image Processing*, Lausanne, September 1996.

[17] Masayuki Tanimoto, Hiroshi Ohyama, and Tadahiko Kimoto. A new fractal image coding scheme employing blocks of variable shapes. In *International Conference on Image Processing*, Lausanne, September 1996.