Un nouvel algorithme de compression exploitant le codage arithmétique en lui introduisant de nouveaux paramètres de codage

Atef MASMOUDI Mohamed Salim BOUHLEL

Unité de Recherche: Sciences et Technologies de l'Image et des Telecommunications
Institut Supérieure de Biotechnologie de Sfax (ISBS)-TUNIS

mas_atef@yahoo.fr medsalim.bouhlel@enis.rnu.tn

Résumé: Le codage arithmétique [01-04] est une technique de compression très puissante. L'intérêt apporté à ce codage trouve ses origines dans son utilisation par les dernières normes de compression d'images et de vidéos tel que JBIG, JBIG2, JPEG2000 [11-13] et H264 [08], ce qui traduit que toute amélioration apportée au codage arithmétique, engendre une amélioration dans ces normes de compression. Le codage arithmétique, comme toute autre technique de compression statistique, dépend uniquement de la distribution probabiliste des symboles à coder, toutefois, grâce à notre approche nous introduisons de nouveaux paramètres de codage qui sont : le découpage de l'image en blocs de pixels en les triant selon l'un de ses caractéristiques, et le sens de codage. Les résultats obtenus ont mis en évidence une réduction du débit de compression. Ainsi, nous avons montré que les débits de compression calculés par notre approche sont en moyenne inférieure à 0.45 bits/pixel par rapport à ceux calculés par le codage arithmétique statique CAS, soit un gain de l'ordre de 6%. Ce papier est développé en cinq parties. D'abord une introduction, ensuite un bref présentation du codage arithmétique. La deuxième section est consacrée à la description de notre approche. Nous avons détaillé les résultats obtenus dans la troisième partie. Enfin on donne la conclusion de ce travail.

Mots clefs : Codage arithmétique, Compression, sans perte, entropie, table de probabilité dynamique, sens de lecture, découpage de l'image, tri des blocs de pixels.

INTRODUCTION

La compression des images devient une nécessité dés lors nous voulons les archiver ou bien les transmettre à travers des systèmes de communication numériques. On parle généralement de la compression avec perte et celle sans perte [14-20]. En effet, la compression avec perte est plus populaire, cependant certaines applications ne tolèrent pas la dégradation causée par ce type de codage. La compression sans perte offre des faibles taux de compression, mais elle permet de conserver les données originales. Dans des domaines tel que l'imagerie médicale, la réduction de la taille des images tout en gardant une bonne qualité est une nécessité. La compression permet également de transférer, dans des temps raisonnables, des images par réseau ce qui offre la possibilité d'utiliser des systèmes de communication pour le télédiagnostic en télémédecine. De plus, la compression avec perte des images satellitaires n'est pas préférée, puisqu'on peut perdre facilement des informations obtenues avec un coût très élevé. Actuellement, il existe plusieurs techniques de compression sans perte. En effet, il y a celles qui travaillent directement sur l'image originale, par contre il y a d'autres qui sont appliquées à l'image après subissions de certaines transformations. Notons que dans le cadre de ce papier nous validons notre approche sur des images originales sans subissions d'aucune transformation. Ainsi, nous proposons à travers ce travail une nouvelle approche de compression sans perte des images tout en exploitant le codage arithmétique. Nous avons validé notre approche sur une base d'images aléatoires et nous avons montré son efficacité notamment pour certaines images médicales.

1. Le codage arithmétique

Le codage arithmétique traite l'ensemble d'une séquence de symboles comme une seule entité, ce qui lui permet de réussir à sortir de la contrainte imposée par les approches VLC (Variable Length Coding) qui attribuent un mot de code court aux symboles probables, et des mots plus longs aux symboles moins fréquents. Ainsi, le codage arithmétique permet de représenter une séquence de symboles par un intervalle de nombres réels compris entre 0 et 1. Il est à signaler que toute valeur appartenant à ce dernier intervalle représentera d'une manière unique la séquence à coder. A mesure que la séquence s'allonge, l'intervalle requis pour le représenter diminue, et le nombre de bits qui servent à préciser cet intervalle s'accroît. Les symboles successifs d'une séquence réduisent cet intervalle en concordance avec la probabilité d'apparition du symbole. Finalement, les données comprimées consistent en la partie fractionnaire du plus court nombre binaire qui se trouve dans l'intervalle final.

L'algorithme du codage arithmétique statique est le suivant:

1. Soit X une séquence de m symboles $X=x_1x_2\ldots x_m$ qui prend ses valeurs à partir d'une source $S= \{s_1, s_2, \ldots, s_n\}$,
2. Calculer la probabilité associée à chaque symbole de la source S, notons p_i=Probabilité($S=s_i$)
3. Associer à chaque symbole s_k un sous intervalle [Ls_k, Hs_k) proportionnel à sa probabilité, avec $Hs_k - Ls_k = p_k$.
4. Initialiser la limite inférieure (L) de l'intervalle de travail à la valeur 0 et la limite supérieure (H) à la valeur 1.
5. Tant qu'il reste un symbole dans la séquence à coder :
   - largeur =H - L
   - L = L + largeur x Ls_k
   - H = L + largeur x Hs_k
6. A la fin, n'importe quelle valeur de l'intervalle [L,H) représente d'une manière unique la séquence d'entrée.

Généralement, on choisi comme représentant de la séquence la limite inférieure (L) de l'intervalle, ou bien on peut choisir la valeur moyenne: $\frac{(L+H)}{2}$. On appelle code(X) la valeur choisie comme représentant de la séquence à coder.

Il est à noter qu'a la fin du processus de codage, le codeur arithmétique génère un fichier composé par un en-tête suivi de la représentation binaire du code choisi. De plus, l'en-tête peut contenir soit la table des probabilités, soit la table des fréquences de tous les symboles.

Après le codage de la séquence X, on peut facilement décoder cette séquence en utilisant la table des probabilités (ou bien la table des fréquences) et en appliquant l'algorithme suivant :

1. D'abord on doit initialiser la limite inférieure (L) de l'intervalle de travail à la valeur 0 et la limite supérieure (H) à la valeur 1,
2. Tant qu'il reste un symbole à décoder :
   - largeur =H – L
   - Chercher le sous intervalle [Ls_k, Hs_k) du symbole à décoder sachant que :

   $Ls_k \leq$ (code(X) – L) /largeur $\leq Hs_k$
   - décoder le symbole s_k
   - L = L + largeur x Ls_k
   - H = L + largeur x Hs_k
3. A la fin, on arrive à décoder la séquence X.

Il est à signaler que le codage arithmétique statique utilise une table des probabilités fixe durant les processus de codage et de décodage. De plus, on peut calculer, à partir de la table des probabilités, la probabilité de la séquence p(X) qui est égale à :

$$P(X) = p_m \times p_{m-1} \ldots p_2 \times p_1 = \prod_{i=1}^{m} p_i \quad \textbf{(1)}$$

Après l'étape de codage, la séquence X sera représenté par un intervalle I= [L,H). L'intervalle I peut s'écrire sous la forme I= [L, L+v) avec v=H-L : la largeur de l'intervalle. Ainsi, d'après l'algorithme de codage, on peut facilement déduire que la largeur de l'intervalle v est égale à la probabilité p(X) de la séquence X. Par conséquent, code(X) peut être représenté uniquement avec un nombre de bits qui est égale à la valeur

$$\lfloor -\log_2(p(X)) \rfloor + 1 \qquad \textbf{(2)}$$

avec $\lfloor X \rfloor$ désigne la partie entière de X.

Ainsi, si la largeur de l'intervalle requis pour représenter une séquence X diminue, alors le nombre de bits qui servent à préciser cet intervalle s'accroît.

Soit, par exemple, à coder la séquence X="aaaaabbbcc". En appliquant l'algorithme du codage arithmétique sur la séquence X, on obtiendra l'intervalle suivant [0.02252875, 0.0225625).

$$p(X)= \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{3}{10} \times \frac{3}{10} \times \frac{3}{10} \times \frac{2}{10} \times \frac{2}{10} = 0{,}00003375$$

De plus, nous avons besoin au minimum de $\lfloor -\log_2(p(X)) \rfloor + 1 = 16$ bits pour coder cette séquence.

Dans ce qui suit, nous allons détailler le principe de notre approche permettant de compresser sans perte des données numériques (texte, image, vidéo, …) avec un débit de compression inférieur à celui calculé par le codage arithmétique statique.

Passons maintenant à présenter notre approche permettant la compression sans perte des données par l'exploitation du codage arithmétique et de la répartition spatiale des symboles à coder.

2. Notre approche

Le codage arithmétique, comme tout autre technique de codage statistique, permet de générer des flux binaires dont la taille ne dépend que des probabilités des symboles. Par contre, grâce à notre approche nous introduisons de nouveaux paramètres de codage. D'une manière générale, l'image est codée en commençant dans le coin en haut à gauche et en continuant de gauche à droite à travers chaque ligne jusqu'au coin inférieur droit. Mais il peut y avoir d'autres manières d'encoder : verticalement, horizontalement, en zigzag, par flots de 8x8 pixels ... Notons que les résultats présentés sont obtenus à partir du codage verticale des images.

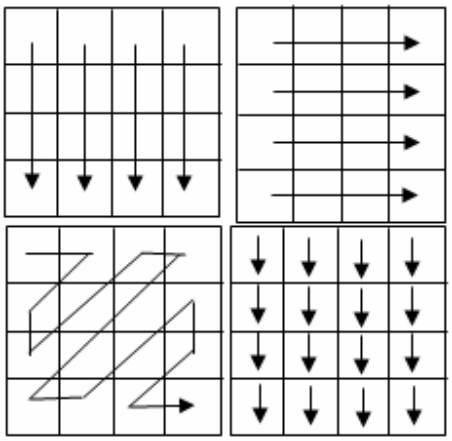

Fig. 1. *Différents types de codage*

De plus, les principales caractéristiques d'une image utiles et facilement mesurable sont la moyenne m, la variance v et l'entropie H. Pour une image de taille NxM codée sur 256 niveaux de gris $(s_i)_{0 \le i \le 255}$, ces caractéristiques sont donnés par :

$$m = \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} n_{i,j} \quad \textbf{(3)}$$

$$v = \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (n_{i,j} - m)^2 \quad \textbf{(4)}$$

$$H = -\sum_{i=0}^{255} p(s_i) \log_2 (p(s_i)) \quad \textbf{(5)}$$

Notre algorithme consiste à appliquer les opérations suivantes :

1. découper de l'image en blocs de pixels
2. trier les blocs selon un paramètre ; la moyenne (Fig. 2), la variance ou l'entropie
3. appliquer un codage arithmétique avec une table de probabilité dynamique, et qui consiste à mettre à jour la table des probabilités après le codage/décodage de la dernière occurrence de chaque symbole.

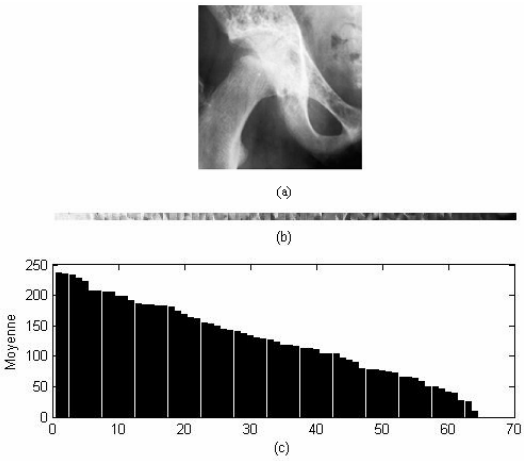

Fig. 2 : *(a) Image originale (b) Arrangement des blocs par moyenne (c) Moyenne relative à chaque bloc*

Il est à noter que l'algorithme proposé peut être intégré dans toute approche de compression exploitant le codage arithmétique statique [05-07]. Ce qui permet de réduire davantage les débits de compression.

Soit par exemple à coder la séquence X tout en appliquant le principe qui consiste à mettre à jour la table des probabilités après le codage/décodage de la dernière occurrence de chaque symbole. Le *Tableau 4* et le *Tableau 5* illustrent l'évolution des processus de codage et de décodage respectivement.

Tableau 1. *Codage de la séquence X avec notre approche*

Symbole	Probabilités a	b	c	L	s
a				0	0.5
a				0	0.25
a	0.5	0.3	0.2	0	0.125
a				0	0.0625
a				0	0.03125
M À J de la table des probabilités					
b				0	0.01875
b	0	0.6	0.4	0	0.01125
b				0	0.00675
M À J de la table des probabilités					
c	0	0	1	0	0.00675
c				0	0.00675

$$p(X) = \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{3}{5} \times \frac{3}{5} \times \frac{3}{5} \times \frac{1}{1} \times \frac{1}{1} = 0{,}00675$$

On obtient à la fin du processus de codage un intervalle I= [0, 0.00675] de largeur 0.00675, et n'importe quelle valeur de cet intervalle représente d'une manière unique la séquence X. Il est à noter que cet intervalle est plus large que celui calculé par le codage arithmétique statique. De plus, nous avons besoin au minimum de $\lfloor -\log_2(p(X)) \rfloor + 1 = 9$ bits pour coder la séquence X, et par conséquent, grâce à notre approche, nous avons réduit le nombre de bits nécessaire pour compresser sans perte la représentation de X.

$$p(X) = \prod_{i=1}^{m} p_{i/1,2,\ldots,i-1} \quad \textbf{(6)}$$

Tableau 2. *Décodage de la séquence X en appliquant notre approche (code(X)=L+s/2)*

Symbole	Intervalle Associé a	b	c	code
a				0.003375
a				0.00675
a	**[0,0.5)**	[0.5, 0.8)	[0.8,1)	0.0135
a				0.027
a				0.054
M À J de la table des probabilités				
b				0.108
b	0	**[0.0, 0.6)**	[0.6,1)	0.18
b				0.3
M À J de la table des probabilités				
c	0	0	**[0,1)**	0.5
c				0.5

Si nous inversons la séquence X, alors nous obtenons la séquence suivante Y=“ccbbbaaaaa”. En compressant cette séquence par le codage arithmétique statique, nous obtenons le même nombre de bits que celui calculé pour la séquence X :

$$p(Y)= \frac{2}{10} \times \frac{2}{10} \times \frac{3}{10} \times \frac{3}{10} \times \frac{3}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} \times \frac{5}{10} =$$

$$0{,}00003375 \text{ et } \lfloor -\log_2(p(Y)) \rfloor + 1 = 16 \text{ bits}$$

Par contre, en appliquant notre approche, nous obtenons un nombre de bits différent :

$$p(Y)= \frac{2}{10} \times \frac{2}{10} \times \frac{3}{8} \times \frac{3}{8} \times \frac{3}{8} \times \frac{1}{1} \times \frac{1}{1} \times \frac{1}{1} \times \frac{1}{1} \times \frac{1}{1} = 0.002109375$$

$$\text{et } \lfloor -\log_2(p(Y)) \rfloor + 1 = 10 \text{ bits}$$

Il faut noter que la présente approche n’est pas limitée à l’utilisation d’un sens de parcours particulier, ainsi il y a beaucoup d’algorithmes qui cherchent à arranger les éléments d’une séquence et par conséquent notre approche est exploitable quelque soit l’algorithme utilisé.

3. Résultats expérimentaux

Dans cette section, nous discutons les résultats des débits de compressions. En effet, pour valider notre approche, nous l’avons comparé avec le codage arithmétique statique, en terme de débit de compression. Les images utilisées pour la validation de notre approche sont en niveaux de gris (8 bits/pixel). La taille de ses images de test est 256x256 octets et 512x512 octets.

Les tests effectués sur une base aléatoires d’images ont monté que les meilleurs résultats sont obtenus en triant les blocs selon la moyenne (Fig. 3). Ainsi, les résultats obtenus, sur des blocs de taille 32x32, ont mis en évidence une réduction moyenne des débits de compression de 0.45 bits/pixel par rapport à ceux calculés par le CAS.

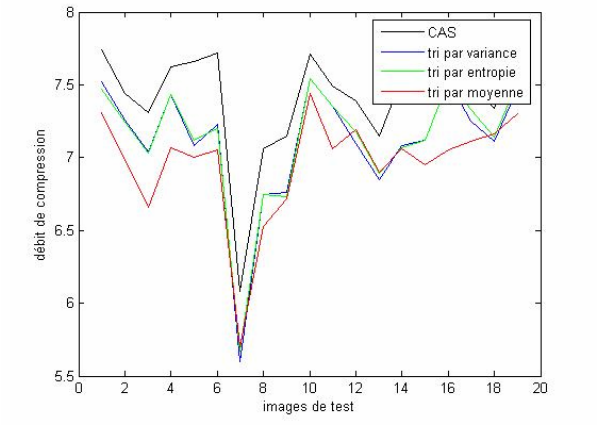

Fig. 3 : *Résultats des débits de compression sur une base aléatoires d’images en niveaux de gris.*

Ce gain est très satisfaisant du fait que nous proposons un schéma de compression sans perte. En plus, nous utiliserons un alphabet de 256 symboles, ce qui est généralement utilisé pour les images numériques. Nous devons donc mettre à jour la table des probabilités au plus 256 fois. Ainsi, du point de vue complexité, notre approche est très rapide et elle ne permet pas de diminuer la vitesse de codage.

4. Conclusions et futurs travaux :

Nous avons présenté dans ce papier une nouvelle approche permettant la compression sans perte des images. Son originalité réside dans le fait qu’elle exploite le codage arithmétique. En fait, contrairement au codage arithmétique statique qui utilise une table de probabilité fixe, nous avons proposé de mettre à jour la table des probabilités après le codage/décodage de la dernière occurrence de chaque symbole.

Nous avons montré, sur une base aléatoire d’images de tests, que nous avons compressé sans perte des images avec un débit de compression qui est en moyenne inférieur à 0.45 bits/pixel par rapport à celui calculé par le codage arithmétique statique.

Vu que l’efficacité de notre approche dépend essentiellement du sens de codage, il est intéressant de rajouter une étape de prétraitement. On a proposé, dans le cadre de ce papier, de découper l’image en blocs de pixels et de les triés selon l’un de ses caractéristiques. Toutefois, on désire tester notre approche tout en exploitant la transformation en ondelette discrète ou bien la transformation en cosinus discrète, ce qui pourra améliorer notre schéma de codage.

Notre approche peut être appliqué dans plusieurs domaines d’activité. Toutefois, on estime qu’il est parfaitement adapté aux secteurs d’archivage et de transfert des images médicales vu qu’il présente l’avantage d’être réversible en effectuant une compression sans perte d’informations.

5. Références :

[01] G. G. Langdon Jr., “An Introduction to Arithmetic Coding”, *IBM Journal of Research and Development*, Vol.28, No.2, March 1984.

[02] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression”, *Communications of the ACM*, 30(6), pp. 520-540, June 1987.

[03] A. Moffat, R. M. Neal, and I. H. Witten, “Arithmetic coding revisited”, *ACM Transactions on Information Systems*, 16(3), pp. 256-294, July 1998.

[04] P. G. Howard and J. S. Vitter, “Arithmetic Coding for Data Compression”, *Proceedings of the IEEE*, 82(6), pp. 857-865, June 1994.

[05] F. Golchin, K.K. Paliwal, “A lossless image coder with context classification, adaptive prediction and adaptive entropy coding”, *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, p. 2545-2548, 1998.

[06] I. Matsuda, N. Shirai and S. Itoh, “Lossless Coding Using Predictors and Arithmetic Code Optimized for Each Image,” *Proc. of Intl. Workshop VLBV03, LNCS*, Vol.2849, p.199–207, Sep. 2003.

[07] N. Kuroki, T. Manabe, and M. Numa, "Adaptive arithmetic coding for image prediction errors",

2004 IEEE international symposium on circuits and systems (ISCAS 2004), vol. III, p. 961-964, May 2004.

[08] D. Marpe, H. Schwarz, et T "Weigand. Context based adaptative binary arithmetic coding in the H.264/AVC video compression standard". *IEEE Trans. on Circuits and Systems for Video Technology*, 13(7) :p. 620–636, 2003.

[09] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, Vol. 27, pp.379–423, 623-656 July 1948.

[10] N. Abramson, "*Information Theory and Coding*", McGraw-Hill Book Company, Inc., New York, NY, 1963.

[11] M.D. Adams, "The JPEG-2000 Still Image Compression Standard (Last Revised: 2002-12-25)", *ISO/IEC JTC 1/SC 29/WG1 N 2412*, December 2002.

[12] C. Christopoulos, A.N. Skodras and T. Ebrahimi, "The JPEG2000 still image coding system: an overview", *IEEE Transactions on Consumer Electronics, vol. 46, No. 4, pp. 1103–1127*, November 2000.

[13] A. Skodras, C. Christopoulos et T. Ebrahimi, "The JPEG2000 Still Image Compression Standard", *IEEE Signal Processing Magazine, vol. 18, pp. 36-58*, September 2001.

[14] M. Burrows and D. J. Wheeler "A block-sorting lossless data compression algorithm". *Tech. Rep. 124. Digital Systems Research Center*, Palo Alto, CA, 1994.

[15] M. Effros, "PPM performance with BWT complexity: A new method for lossless data compression". *In Proceedings of the Data Compression Conference*, 2000.

[16] A. Moffat, T. C. Bell, I. H. Witten, "*Lossless Compression for Text and Images*". University of Melbourne Australia, Dep. Of Comp.Science, Facsimile, Oct.1995.

[17] J. W. Marcelo, G. Seroussi, G. Sapiro, "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm". *In Proceedings of the Data Compression Conference*, 1996.

[18] X. Wu, "An Algorithmic Study on Lossless Image Compression". *In Proceedings of the Data Compression Conference*, 1996.

[19] L. Cappellari , G. A. Mian, "Analysis of joint predictive-transform coding for still image compression", *Signal Processing, v.84 n.11*, p. 2097-2114, November 2004.

[20] Z. Ilcheva, V. Ilchev, "A method for lossless compression of images", *Proceedings of the 4th international conference conference on Computer systems and technologies: e-Learning*, Rousse, Bulgaria , p.278-283, June 19-20, 2003.