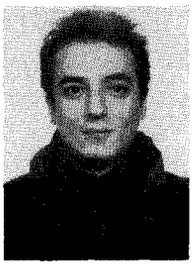
Classification en blocs de taille variable pour codage d'image par quantification vectorielle

Variable size block classification for vector quantization image coding



André DAVIGNON

Laboratoire d'Électronique de Physique Appliquée et de Productique, École Centrale des Arts et Manufactures, Grande Voie des Vignes, 92295 CHATENAY-MALABRY CEDEX.

André DAVIGNON est né en 1965. Il a obtenu le diplôme d'ingénieur de l'École Nationale Supérieure d'Électronique et de Radio-électricité de Bordeaux (ENSERB) en 1987 et prépare actuellement son doctorat. Ses travaux portent sur le traitement et le codage d'images ainsi que sur la vision 3D.

RÉSUMÉ

Cette étude propose de découper les images en blocs dont la taille est variable en fonction de la quantité de détails rencontrée dans la zone traitée. Ce découpage permet d'obtenir des taux de compression plus importants que pour un découpage fixe. Une méthode de classification des blocs de 4×4 pixels basée sur la quantification vectorielle binaire est également présentée.

MOTS CLÉS

Codage d'image, compression, quantification vectorielle, classification de blocs.

SUMMARY

This work proposes to divide images in blocks whose size can be changed according to the amount of details encountered in the processed zone. This division leads to bigger compression ratios than a regular division. We also present a 4×4 *pixels block classification using binary vector quantization.*

KEY WORDS

Image coding, compression, vector quantization, block classification.

1. Introduction

Au cours des années 1980, les travaux de recherche en codage d'image ont montré que la quantification vectorielle est une méthode puissante donnant de très bons résultats et ouvrant un champ d'évolution important. Le procédé le plus simple utilisant la quantification vectorielle (QV) consistait à coder chaque bloc de 4×4 pixels d'une image par un mot de 8 bits en recherchant dans un dictionnaire de taille 256 le vecteur le plus « proche » du bloc à coder, la mesure de distance utilisée étant la distance euclidienne. A partir de cette première méthode sont apparus les principaux problèmes rencontrés avec la QV qui devaient l'amener à évoluer. Ces problèmes sont d'une part l'énorme quantité de calculs à effectuer pour la recherche du plus proche voisin dans les dictionnaires

et d'autre part l'effet de « bloc » visible dans les images après le décodage. Ainsi, de nombreux travaux ont été effectués dans ces deux directions.

S'agissant du seul problème de la masse de calculs à effectuer, des méthodes ont été proposées, soit pour réduire la taille des dictionnaires à parcourir en formant des codes produits (Multistep Vector Quantization, *voir* [1]), soit pour parcourir les dictionnaires de façon arborescente et ainsi obtenir une masse de calcul non plus proportionnelle à la taille du dictionnaire mais au logarithme de cette taille. L'une de ces méthodes utilise les structures d'arbres binaires multidimensionnels (arbres-kd) appelés kd-trees dans la littérature anglo-saxonne [2-3], dont l'utilisation en QV a été introduite par Equitz, et permet de manipuler de très gros dictionnaires. Les arbres-kd organisent les données vecteurs en une structure d'arbre permettant une recherche de plus proche voisin en log(N) (N étant la longueur du dictionnaire). Ces arbres binaires sont formés de nœuds interconnectés et de seaux, dans lesquels se trouvent les données, situés au dernier niveau de l'arbre. Chaque nœud de l'arbre détermine une partition de l'ensemble de vecteurs qui lui est associé. Cette partition est construite en effectuant une comparaison d'une certaine coordonnée des vecteurs (la même pour ceux du même ensemble), à un seuil, ceci jusqu'à obtenir le nombre désiré de vecteurs dans les seaux. Les travaux d'Equitz [4] ont de plus permis de fournir une méthode de construction de dictionnaires demandant moins de calculs que le traditionnel algorithme de Linde, Buzo et Gray [5].

L'autre inconvénient majeur de la QV (effets de bloc) a été étudié principalement par Ramamurthi et Gersho. Leurs premiers travaux [6] ont permis de tenir compte de la perception visuelle en séparant les blocs homogènes et les blocs contenant des contours, et en utilisant pour chaque type de bloc un dictionnaire séparé spécialement construit pour ce type de bloc. De cette façon, non seulement ils appréhendaient le problème des effets de blocs, mais aussi leur méthode permettait de réduire la charge de calculs puisque la recherche du plus proche voisin ne s'effectuait que dans le sous-dictionnaire concerné, de taille bien moins importante que celle du dictionnaire global. Leurs travaux suivants [7-8], ont permis de séparer les blocs de contour en fonction de leur orientation en raison de la sensibilité de l'œil à l'orientation des contours. Dans ce cas le dictionnaire final était la concaténation de 31 sous-dictionnaires dédiés chacun à une classe de blocs (dont 28 classes de blocs de contour) et l'effet de bloc se trouvait encore diminué.

Cependant, dans tous les cas, non seulement la classification demande une quantité importante de calculs, mais d'autre part elle ne s'applique qu'à des blocs dont la taille a été fixée à 4×4 pixels. Ainsi le découpage en blocs s'effectue de la même façon que la zone traitée comporte un grand nombre de détails ou non. C'est pourquoi nous nous sommes intéressés à l'étude d'une classification rapide des blocs de 4×4 pixels ainsi qu'à l'étude d'une classification permettant de découper les zones à coder en blocs dont la taille peut varier en fonction de la quantité de détails ou de contours détectés. La méthode proposée, permettant de mieux tenir compte des propriétés locales des zones à traiter, effectue un codage fin pour les zones de détails et code de façon plus sommaire les zones homogènes. Cette classification permet d'obtenir des taux de compression bien plus importants si le contenu de l'image le permet.

2. Classification en blocs de taille variable

Cette classification s'effectue en deux étapes :

2.1. Classification des blocs de 4×4 pixels

Pour déterminer la taille du bloc qui servira à coder chaque zone de l'image, on commence par diviser l'image obtenue en blocs de 4×4 pixels et par examiner chacun de ces blocs. Il s'agit tout d'abord de savoir si chacun de ces blocs constitue un bloc homogène, un bloc de détails ou un bloc de contour. On définit donc une classe de blocs de zone homogène, une classe de blocs de détails « aléatoires » et huit classes de blocs de contour. Ces huit classes sont divisées en quatre groupes définissant chacun une orientation de contour (verticale, horizontale et les orientations des deux diagonales). Chaque classe à l'intérieur d'un groupe définit une transition de sombre vers clair ou clair vers sombre. Un bloc de zone homogène est un bloc non classé comme bloc de contour et un bloc de détails « aléatoires » un bloc de contour qui ne se range dans aucune des huit classes définies.

Ramamurthi et Gersho [7-8] ont fourni une méthode permettant de détecter les blocs de contour de taille 4×4. Ils proposaient de construire pour chaque bloc à traiter une table de gradients verticaux et une de gradients horizontaux. Ils calculaient les gradients verticaux et horizontaux de pixels adjacents dans la direction correspondante ainsi que leur moyenne. Si le gradient dépassait la valeur du produit de la moyenne par une constante, la table de gradient dans la direction correspondante comportait le signe du gradient sinon la valeur 0. Si une des tables contenait au moins trois valeurs non nulles le bloc était classé comme bloc de contour. Enfin, pour déterminer l'orientation du contour ils effectuaient des manipulations sur les tables de gradients.

La méthode que nous proposons, qui a l'avantage de demander peu de calculs, est la suivante :

Pour chaque bloc de 4×4 pixels on calcule la moyenne μ des pixels du bloc et l'écart-type σ du bloc par rapport à cette moyenne. Si l'écart-type σ est supérieur à un seuil S, le bloc est détecté comme un bloc de contour. S est un seuil choisi expérimentalement dont la valeur est ici de 6. La figure 1 montre, sur l'image de gauche, les blocs de contour obtenus avec cette valeur de seuil dans notre image de test et, sur l'image de droite, les blocs homogènes contenus dans l'image.

Pour déterminer l'orientation du bloc de contour on construit une table binaire de 16 éléments correspondant aux 16 pixels du bloc de l'image. Dans le bloc binaire à la position (i,j) la coordonnée est à 1 si la

Fig. 1. – Gauche : image des blocs de contour de l'image PEPPERS, droite : image des blocs homogènes.

coordonnée à la même position dans le bloc de pixels correspondant est supérieure à la valeur moyenne du bloc. Si la valeur du pixel est inférieure ou égale à la valeur moyenne la coordonnée binaire en position (i, j) est 0. Ainsi cette table binaire constitue la représentation binaire du contour contenu dans le bloc. Disposant de cette table binaire on peut déterminer l'orientation en la comparant à ce que nous appellerons les 28 Blocs Binaires de Contours Fondamentaux (BBCF) qui correspondent aux 28 classes de contours définies dans [8]. Ces BBCF sont définis en fonction des quatre orientations choisies qui sont la direction verticale (*fig.* 2), la direction horizontale (*fig.* 3), les directions des diagonales principales (*fig.* 4 et 5).

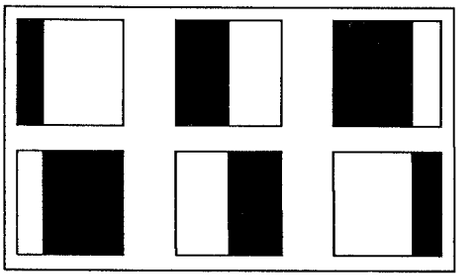


Fig. 2. – BBCF verticaux, classes 1 et 2.

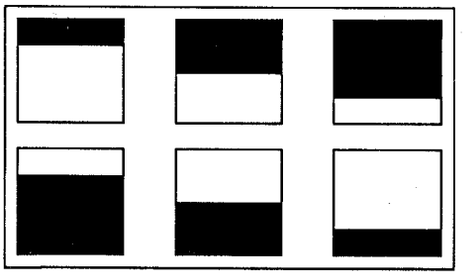


Fig. 3. – BBCF horizontaux, classes 3 et 4.

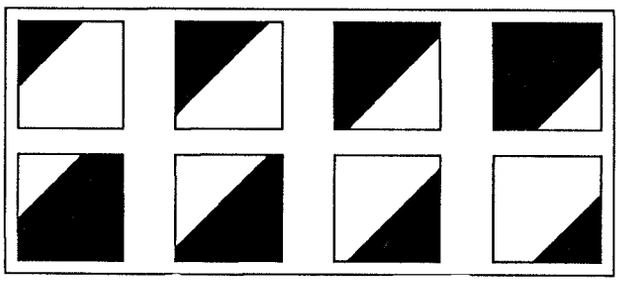


Fig. 4. – BBCF de première diagonale, classes 5 et 6.

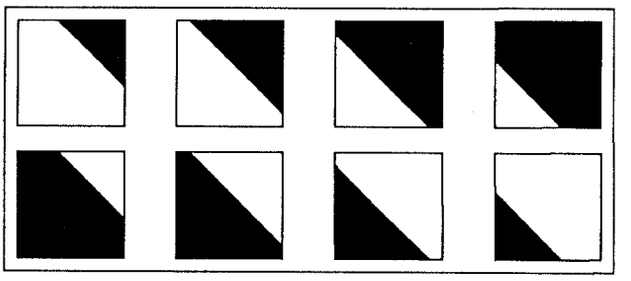


Fig. 5. – BBCF de seconde diagonale, classes 7 et 8.

Mais il est possible de définir d'autre BBCF pour obtenir une autre classification tout en gardant exactement la même méthode de classification. L'orientation du contour dans le bloc est déterminée en utilisant une quantification vectorielle sur des vecteurs binaires.

En effet chaque BBCF et chaque table binaire construite peut être considéré comme un vecteur de dimension 16 à coordonnées binaires. On définit alors une notion de distance entre ces vecteurs :

Pour deux vecteurs **a** et **b**, la distance $d(\mathbf{a}, \mathbf{b})$ entre ces deux vecteurs est définie par la relation :

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=0}^{15} [c_i(\mathbf{a}) \text{ EXOR } c_i(\mathbf{b})]$$

où c_i désigne la fonction qui retourne la valeur de la i-ième coordonnée binaire (ou encore le i-ième bit d'un mot) et EXOR l'opérateur OU exclusif.

Pour chacun des BBCF on calcule donc la distance entre le vecteur binaire obtenu à partir du bloc de contour et le vecteur binaire correspondant au BBCF. Le minimum de distance obtenu après ces calculs détermine le BBCF convenant le mieux à la représentation du contour dans le bloc et la classe du bloc de contour est alors la classe correspondant au BBCF. Si le minimum de distance est supérieur ou égal à 3 le bloc considéré est un bloc détecté comme bloc de contour dans lequel les variations ne déterminent aucune direction de contour. Ce bloc est alors rangé dans la classe des blocs de « détails aléatoires ». Dans notre cas nous n'avons considéré que huit classes de blocs de contours. Pour une transition de sombre vers clair (resp. clair vers sombre) la classe de contours verticaux regroupe les trois BBCF verticaux de même transition, la classe de contours horizontaux les trois BBCF horizontaux de même transition et enfin les classes de contours diagonaux regroupent chacune les quatre BBCF de même direction et de même transition. Il semble (et cette hypothèse a été vérifiée *a posteriori*) que ce nombre de classes est suffisant par rapport à la perception que nous avons des contours. Cependant on peut travailler avec les 28 classes définies dans [8], chaque classe ne comprenant alors qu'un BBCF, pour obtenir une classification encore plus fine ou pour une application autre que la compression des images. Dans ce cas le minimum de distance désignera le BBCF et donc la classe appropriée.

2.2. Regroupement

La seconde partie de la classification est chargée de déterminer quelles zones dans l'image ont besoin d'être codées par blocs de 4×4 pixels. Pour cela, on divise l'image en blocs de 8×8 pixels et on examine chacun de ces blocs pour savoir s'il représente une zone qui doit être codée de manière fine. Utilisons la notation B8 pour désigner les blocs de 8×8 pixels et B4 les blocs de 4×4 pixels.

Si dans un B8 au moins un des quatre sous-blocs B4 a été classifié comme un bloc de contour par la classification précédente, ce B8 représente une zone comportant, soit un (ou plusieurs) contour(s), soit des détails « aléatoires ». Chacun des quatre B4 sera codé séparément par un mot de code et le B8 sera reconstitué par les quatre sous-blocs décodés séparément, étant entendu qu'un ou plusieurs sous-blocs B4 (au plus trois) peut alors être un B4 de zone homogène. Dans ce premier cas, le B8 nécessitera donc pour sa description quatre mots de code. Par contre, nous avons fait l'hypothèse que, si aucun des quatre sous-blocs n'est un bloc de contour ni un bloc de détails (donc si les quatre sous-blocs sont dans la classe de blocs homogènes), alors ce B8 représente une zone homogène de taille 8×8 pixels et sera rangé dans ce que nous appellerons la classe H8 (classe des blocs B8 définissant une zone Homogène). Ce critère nous est apparu comme un compromis entre la simplicité et l'efficacité lors de notre étude. Il est néanmoins possible d'utiliser d'autres critères peut-être moins simples qui permettraient de mieux détecter encore les blocs de classe H8. Ainsi les quatre B4 sont regroupés en un B8 de classe H8 qui sera codé par un seul mot de code représentant le bloc entier. Nous construisons donc un dictionnaire séparé contenant des vecteurs représentant les blocs B8 de classe H8. L'intérêt apparaît immédiatement puisqu'au lieu d'utiliser quatre mots de code pour le B8 on n'en utilise qu'un, ce qui permet de gagner pour ce B8 75 % de l'information de codage. Il suffit donc de connaître pour chaque B8 s'il est ou non un B8 de classe H8. La figure 6 montre, sur l'image de gauche, les zones de l'image devant être décrites par blocs de 4×4 pixels et, sur l'image de droite, les zones classifiées comme blocs H8. Il est clair dans cet exemple que la plus grande partie de l'image est formée de blocs H8.

Ainsi, les images ne contenant que peu de contours et/ou peu de détails présenteront à la classification un grand nombre de B8 de classe H8, ce qui permettra de les comprimer fortement. D'autre part, le gain en compression introduit par le regroupement des B4 en H8 permet d'utiliser pour les B4 de contours (donc non regroupés) des mots de code assez longs sans pénaliser le taux de compression final si le nombre de H8 est assez important. Ainsi nous pourrons utiliser pour ces blocs des dictionnaires de taille assez grande pour permettre une très bonne reconstitution des contours. Nous évaluerons plus loin, dans le paragraphe 5, la compression en fonction de la quantité de H8 contenus dans l'image ainsi que le taux de compression minimal obtenu en l'absence de blocs B8 de classe H8.

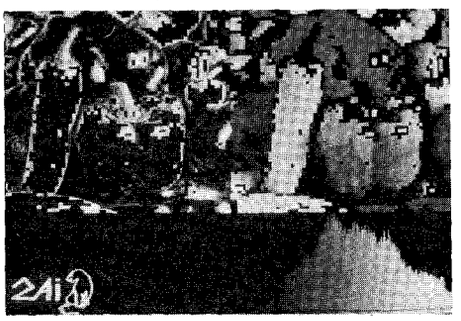


Fig. 6. — Gauche : zones de l'image codées par blocs de 4×4 pixels, droite : zones codées par blocs de 8×8 pixels.

3. Quantification et codage

Chaque bloc B8 se présente donc soit comme un vecteur à coder (s'il est de classe H8), soit comme l'ensemble de quatre vecteurs, représentant chacun un B4, à coder séparément. Dans chacun des cas nous quantifions la valeur moyenne et la partie dynamique du B8 ou des B4 séparément, ce qui permet de bien reconstruire ces valeurs et d'éviter de brusques sauts de valeurs moyennes dans l'image décodée. Pour cela, nous appliquons à chaque B4 ou B8 à quantifier une transformation de Hadamard d'ordre correspondant à la taille du bloc à traiter. Cette transformation est choisie pour sa simplicité et son efficacité semble suffisante pour notre application. En effet dans le domaine transformé une grande partie de l'énergie se trouve dans les coefficients se situant près de l'origine. On peut donc tronquer les blocs transformés et ne retenir pour la quantification qu'un nombre plus faible de coefficients, ce qui permet d'alléger la charge de calculs à effectuer. La troncature a été choisie de telle sorte qu'elle n'introduise pas de dégradation visible après transformation inverse. La figure 7 donne les écarts-types des coefficients transformés des blocs H8 de l'image de test.

2409.51	249.88	107.79	99.30	76.11	76.79	54.35	50.58
215.65	127.13	83.58	70.32	56.62	52.49	43.94	33.66
108.98	74.62	66.42	54.45	42.60	38.07	34.79	25.64
94.54	63.75	55.69	46.06	36.25	31.87	28.63	22.23
67.50	53.93	40.37	35.77	30.57	27.03	22.75	18.04
63.53	51.92	36.81	32.04	28.51	26.26	20.86	16.88
52.52	39.73	32.33	27.81	22.87	21.28	18.74	15.08
48.78	31.42	25.27	21.18	18.07	17.30	14.77	12.54

Fig. 7. — Écarts-types des coefficients des blocs H8 transformés.

On retient pour les B4 et les B8 le même nombre de coefficients dynamiques fixé à 9 (*fig.* 8). Le coefficient en haut à gauche du bloc représente la valeur moyenne du bloc qui sera quantifiée de façon scalaire.

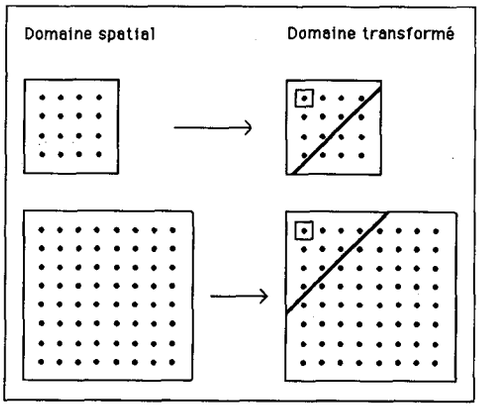


Fig. 8. – Troncatures dans les domaines transformés.

Les coefficients dynamiques retenus dans les blocs H8 transformés correspondent aux coefficients de plus grande variance.

En effet les B8 de classe H8, représentant des zones homogènes de l'image, ne contiennent que peu d'énergie qui est bien décrite par quelques coefficients. Ainsi les vecteurs représentant un B4 ou un B8 de classe H8 sont de même dimension et peuvent être traités par le même quantificateur (*fig.* 9). C'est pourquoi nous avons choisi le même nombre de coefficients dynamiques retenus pour les B4 et les H8. Mais il est clair que les H8 seraient décrits plus fidèlement en introduisant une troncature moins « sévère ».

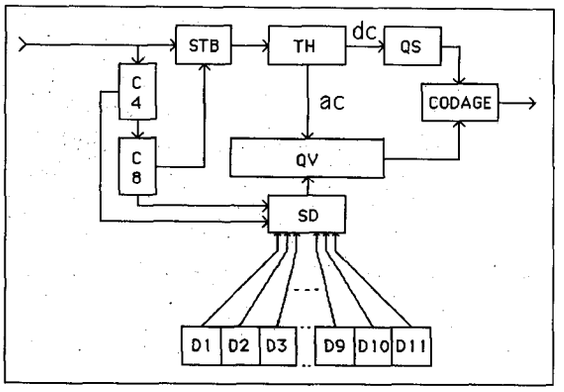


Fig. 9. – Schéma fonctionnel du codeur. STB : sélection de la taille du bloc. C4 : classification des B4. C8 : classification des B8. TH : transformation de Hadamard. QS : quantification scalaire. QV : quantification vectorielle. SD : sélection de dictionnaires. D1...D11 : sous-dictionnaires.

Quelle que soit la dimension du bloc considéré, il est codé par un mot de 16 bits qui est un code produit du code de la valeur moyenne et du code de la partie dynamique. Mais le code de la valeur moyenne n'occupe pas le même nombre de bits dans les deux cas (*fig.* 10 et 11) et donc la taille du dictionnaire global des B4 et des B8 n'est pas la même. Dans le cas des B4 10 bits sont alloués au code de la partie dynamique donc 6 au code de la valeur moyenne. Le dictionnaire global des B4 comporte donc 1 024 vecteurs organisés de la manière suivante :

$512 = 64 \times 8$ vecteurs pour les classes de contour;

256 vecteurs pour la classe de blocs de détails;

256 vecteurs pour la classe de blocs homogènes.

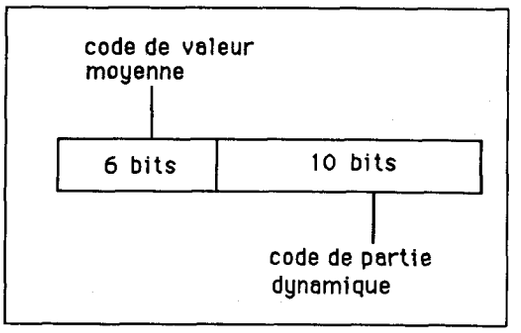


Fig. 10. – Organisation des mots de code des blocs B4.

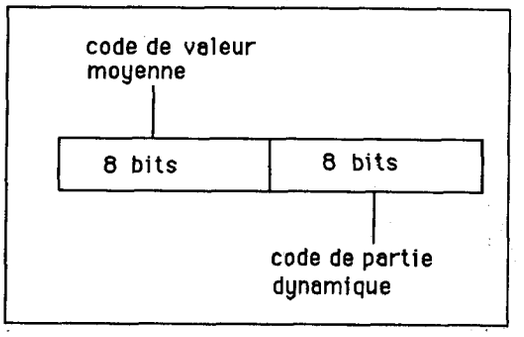


Fig. 11. – Organisation des mots de code des blocs B8.

Remarquons que ces dictionnaires comportent un grand nombre de vecteurs ce qui permet de reproduire d'une façon très fidèle les blocs B4. D'autre part la recherche dans ces dictionnaires étant effectués de manière arborescente en utilisant la structure d'arbres binaires multidimensionnels (arbres-kd) leur taille importante ne pénalise pas le temps de traitement.

Dans le cas des blocs de classe H8 8 bits sont alloués au codage de la valeur moyenne et 8 au codage de la partie dynamique. Le dictionnaire des blocs de classe H8 comporte donc 256 vecteurs organisés également en arbre-kd.

Le codage des valeurs moyennes est effectué dans un cas comme dans l'autre par une quantification uniforme de ces valeurs. Nous avons donc pour les B8 256 niveaux de quantification (8 bits de codage) et pour les B4 64 niveaux de quantification (6 bits de codage).

D'autre part nous devons inclure dans l'en-tête de l'image codée la taille des blocs codés. Pour cela il suffit d'un bit par bloc B8 pour signaler s'il est représenté par un seul mot de code (classe H8) ou par quatre mots de code dans le cas d'un bloc B8 contenant au moins un sous-bloc B4 détecté comme bloc de contour. Pour une image de 512×512 pixels le nombre de B8 est de $64 \times 64 = 4096$. Il sera donc inclus 4 096 bits qui permettront de décoder les B8. Il aurait été possible d'inclure cette information dans les mots de code eux-mêmes pour éviter d'avoir à inclure ce volume supplémentaire d'information mais nous avons préféré réserver ces mots pour la seule description du bloc, utilisant ainsi des dictionnaires plus importants pour une meilleure reconstruction de l'image.

Nous verrons que les informations supplémentaires incluses dans l'image codée et nécessaires pour le décodage (statut de chaque B8) donnent une limite inférieure du nombre de B8 de classe H8 nécessaires pour que la classification en blocs de taille variable soit avantageuse du point de vue du facteur de compression par rapport à une méthode de codage utilisant une simple classification des blocs B4 et un codage de ces blocs.

4. Décodage

La phase de décodage est relativement simple comme c'est souvent le cas pour les dispositifs à QV. L'entête de l'image servant au décodage des blocs est lue pour connaître le statut de chaque B8. Pour chacun d'eux en fonction de ce statut, 1 mot ou 4 mots de 16 bits servent à décoder le B8. Dans chacun des cas, le ou les mots de 16 bits sont séparés en un code de valeur moyenne et un code de composante dynamique. Pour le décodage de la composante dynamique, on dispose au décodeur non pas des mêmes dictionnaires qu'au codeur (c'est-à-dire contenant les vecteurs formés des blocs transformés et tronqués) mais plutôt de dictionnaires contenant ces blocs tronqués qui ont déjà subi la transformation inverse de Hadamard et dans lequel la valeur moyenne a été mise à zéro. Ainsi on évite d'avoir à effectuer cette transformation inverse puisqu'elle est faite une fois pour toutes dans les dictionnaires. Chaque bloc est donc décodé par une lecture de table et on ajoute à chaque coefficient du bloc décodé la valeur moyenne décodée et reconstruite.

5. Résultats et analyse des performances

Considérons un codage en blocs de taille fixe B4 et la classification de ces blocs en blocs de zone homogène, blocs de détails et blocs de contours, et gardons les dictionnaires dont les tailles ont été données précédemment. Chaque B4 se trouve codé par un mot de 16 bits. Ce codage donne alors des résultats de 1 bit par pixel, donc une compression de 8. Ce résultat va constituer la base de l'analyse de performance. Soit E la taille, en nombre de bits, de l'en-tête destinée à signaler le statut de chaque B8 et soient NB8 le nombre total de B8 dans l'image et NH8 le nombre de B8 de classe H8. Pour le premier type de codage en blocs de taille fixe, le volume d'information, en nombre de bits, occupé par une image codée est:

$$I1 = 8 \times 2 \times [4 \times NB8]$$

Introduisons alors le codage en blocs de taille variable. Le volume d'information occupé par une image codée est de:

$$(1)\quad \begin{cases} I2 = E + 8 \times 2 \times [[4 \times (NB8 - NH8)] + [NH8]] \\ I2 = I1 + E - 8 \times 2 \times [3 \times NH8] \end{cases}$$

Le codage en blocs de taille variable devient alors avantageux du point de vue de la compression dès que $I2 < I1$, soit:

$$E < 48 \times NH8$$

Or $E = NB8$ donc la condition devient:

$$NH8 \times 48 > NB8$$

ce qui représente une proportion de 2% de B8 de classe H8. Or la nature des images traitées donne toujours une proportion bien plus importante de B8 de classe H8 et parmi toutes les images sur lesquelles nous avons travaillé aucune ne présentait moins de 2 % de B8 de classe H8.

L'égalité (1) nous montre que pour un nombre total de pixels N le débit D (quantité d'information en bit par pixel) produit par ce codage vaut:

$$D = 8 \times \frac{E + 2[4\,NB8 - 3\,NH8]}{N}$$

or $N = 64\,NB8$ donc:

$$D = 8 \times \frac{E + 2\,NB8\,[4 - 3\,NH8/NB8]}{64\,NB8}$$

en notant $\rho = NH8/NB8$ et puisque $E = NB8$:

$$D = \frac{65}{64} - \frac{3}{4}\rho$$

La figure 12 donne le débit en fonction de la proportion ρ de NH8. On voit que dans le pire des cas ($\rho = 0$) le débit est de 1,02 bit par pixel ce qui est très proche du résultat obtenu par le codage en blocs de taille fixe. Au contraire dans le cas optimal où la proportion serait de 100 % ($NH8 = NB8$ ou $\rho = 1$) le débit serait de 0,27 bit par pixel et correspondrait à un simple codage par transformation dans lequel les blocs transformés sont codés par quantification vectorielle. Mais cette limite inférieure théorique n'est jamais atteinte car une image naturelle comporte toujours une certaine quantité de contours et de détails.

ρ	0	0.2	0.4	0.6	0.8	1
D	1.02	0.87	0.72	0.57	0.42	0.27

Fig. 12. – Débit D en bit par pixel en fonction de la proportion ρ de blocs B8 de classe H8.

Cet algorithme a été simulé sur des images de test et nous rapportons ici les résultats obtenus sur l'image «PEPPERS» (512 × 512 pixels). Sur la photographie (*fig.* 13) l'original se trouve sur la droite et l'image décodée sur la gauche. Le nombre de NH8 dans l'image est de 2 693 (65,7 %) ce qui donne un résultat de 0,52 bit par pixel environ pour un rapport signal/bruit de 31,1 dB. Les dictionnaires ont été construits suivant les méthodes proposées par Linde, Buzo et Gray [5].

Fig. 13. – Gauche : image 0,5 bit par pixel (2 693 blocs homogènes H8), droite : image PEPPERS d'origine.

6. Conclusion

Nous avons proposé d'une part une méthode de classification des blocs basée sur la quantification vectorielle binaire. Cette méthode se présente comme étant plus rapide que celle présentée jusqu'à aujourd'hui et surtout peut servir de base à tous les types de classification de blocs de 4×4 pixels en fonction de leur représentation binaire. Elle permet également de choisir le nombre de classes en fonction de la complexité tolérée. Enfin nous avons décrit une méthode qui permet de traiter différemment les zones de l'image comportant des détails, nécessitant donc un découpage en blocs de petite taille (4×4 pixels), et les zones ne comportant pas de détails ou de contours, qui peuvent être traitées de façon plus sommaire. Cette classification permet de mieux tirer parti des possibilités qu'offre l'image à coder et donc d'obtenir dans la plupart des cas des taux de compression importants pour une reconstruction d'image de bonne qualité, spécialement au niveau des contours.

Ce codage possède cependant une limite inférieure que l'on désirerait faire encore diminuer. Il semble donc souhaitable à l'avenir de poursuivre l'étude de ce découpage de l'image en blocs de taille variable à des blocs de 16×16 pixels ou même 32×32 pixels pour les zones très homogènes. Il reste donc à travailler sur de nouveaux critères permettant de regrouper quatre blocs de 8×8 pixels en un bloc homogène de 16×16 pixels et quatre blocs de 16×16 pixels en un bloc homogène de 32×32 pixels. D'autre part cette méthode peut constituer la base d'un codage qui permettrait une reconstruction progressive de l'image au décodage en effectuant tout d'abord un découpage grossier de l'image et en l'affinant peu à peu en fonction de la quantité de détails et de contours rencontrés.

Cette étude a été financée par la société Automatisme et Avenir Informatique (2AI), 6, rue du Champoreux, 91540 Mennecy.

Manuscrit reçu le 19 *septembre* 1988.

BIBLIOGRAPHIE

[1] R. M. Gray, Vector Quantization, *IEEE ASSP Magazine*, avr. 1984, p. 4-29.

[2] J. L. Bentley, Multidimensionnal Binary Search Trees Used for Associative Searching, *Communication of the ACM*, 18, n° 9, sept. 1975, p. 509-517.

[3] J. H. Friedman, J. L. Bentley et R. A. Finkel, An Algorithm for Finding Best Matches in Logarithmic Expected Time, *ACM Trans. on Mathematical Software*, 3, n° 3, sept. 1977, p. 209-226.

[4] W. Equitz, Fast Algorithms for Vector Quantization Picture Coding, *Proc. Int'l Conf. on Acoustics, Speech and Signal Processing*, 1987.

[5] Linde, Buzo et Gray, An Algorithm for vector Quantizer Design, *IEEE Trans. on Communications*, 28, n° 1, janv. 1980, p. 84-95.

[6] B. Ramamurthi et A. Gersho, Low Rate Image Coding Using Vector Quantization, *IEEE Global Communications Conf. Record*, nov. 1983, p. 184-187.

[7] B. Ramamurthi et A. Gersho, Image Vector Quantization with a Perceptually-Based Cell Classifier, *Proc. Int'l Conf. on Acoustics, Speech and Signal Processing*, avr. 1984.

[8] B. Ramamurthi et A. Gersho, Classified Vector Quantization of Images, *IEEE Trans. on Communications*, 34, n° 11, nov. 1986, p. 1105-1115.