Compression d'images par fractales basée sur la triangulation de Delaunay

Franck Davoine

THESE

présentée par

Franck Davoine

pour obtenir le titre de
DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(Arrêté ministériel du 30.3.1992)

Spécialité : Signal Image Parole

COMPRESSION D'IMAGES PAR FRACTALES BASÉE SUR LA TRIANGULATION DE DELAUNAY

Date de soutenance : 20 **Décembre** 1995

Composition du Jury :

Alain Chéhikian	PRESIDENT
Dominique Barba	RAPPORTEUR
Michel Barlaud	RAPPORTEUR
Jean-Marc Chassery	EXAMINATEUR
Claude Labit	EXAMINATEUR
Jacques Lévy-Véhel	EXAMINATEUR
Gilles Privat	EXAMINATEUR

Thèse préparée au sein du Laboratoire **TIMC - Institut IMAG**

à mes parents

Remerciements

Ce travail a été réalisé au sein de l'équipe INFODIS du laboratoire TIMC-IMAG de Grenoble.

Je tiens à remercier Monsieur Alain CHÉHIKIAN, Professeur à l'Institut National Polytechnique de Grenoble, pour l'honneur qu'il m'a fait en présidant mon jury de thèse.

Je remercie Monsieur Dominique BARBA, Professeur à l'IRESTE de Nantes, pour s'être intéressé à mon travail et pour avoir accepté la lourde charge d'être rapporteur de ma thèse.

Je remercie également Monsieur Michel BARLAUD, Professeur à l'Université de Nice-Sophia Antipolis, pour l'intérêt qu'il a porté à mes recherches tout au long de mes trois années de thèse, et pour m'avoir permis de collaborer avec différents chercheurs de son équipe, notamment Marc ANTONINI. Je lui sais gré d'avoir accepté d'être rapporteur de ma thèse.

Cette thèse a été menée sous la direction de Monsieur Jean-Marc CHASSERY, Directeur de Recherches au CNRS. Je tiens à lui exprimer ma profonde reconnaissance pour m'avoir proposé ce sujet de recherche, et pour m'avoir constamment soutenu au cours de ces trois années.

Monsieur Claude LABIT, Directeur de Recherches à l'INRIA, m'a fait le grand plaisir de juger mon travail et d'être membre de mon jury. Je l'en remercie très sincèrement.

Merci à Monsieur Jacques LÉVY-VÉHEL, Directeur de Recherches à l'INRIA, pour avoir accepté d'être membre de mon jury. Sa compétence sur l'analyse des signaux par fractales fait que sa présence dans mon jury est un honneur pour moi.

J'exprime également mes remerciements à Monsieur Gilles PRIVAT, Chef de Projets au CNET de Grenoble, pour avoir accepté de participer à mon jury.

Merci à Annick MONTANVERT, Professeur à l'ENS de LYON, pour son enthousiasme, ses conseils et sa confiance.

Je tiens enfin à exprimer ma sympathie à toutes les personnes que j'ai pu cotoyer au sein du CERMO puis de l'Institut Albert Bonniot durant mes années de thèse, et qui m'ont permis de travailler dans les meilleures conditions,

à Christophe pour sa gentillesse,

à Paulette, Nicole et Guy pour leur disponibilité permanente.

Pensée particulière à Christine.

Résumé

Ce mémoire traite de la compression des images fixes par fractales, fondée sur la théorie des systèmes de fonctions itérées (IFS). Après quelques rappels sur les principales méthodes de codage entropique et de compression réversible et irréversible des images nous introduisons les notions nécessaires à la compréhension de la théorie des IFS. Nous détaillons ensuite les principaux algorithmes de compression des images naturelles selon l'approche fractale. Ces derniers consistent à approximer chacun des éléments d'une partition à l'aide d'une transformation locale contractante appliquée sur une autre partie de l'image. Ceci nous conduit à présenter les modèles de partitionnement utilisés pour coder les similarités locales des images. La partie suivante constitue la contribution majeure du travail. Nous présentons un algorithme de codage par fractales fondé sur la triangulation de Delaunay. La souplesse de ce modèle nous permet d'utiliser diverses triangulations adaptées au contenu de l'image à compresser. Nous proposons ensuite différentes solutions ayant pour but d'améliorer le schéma de codage-décodage. La première vise à réduire la complexité de la phase de codage en diminuant le nombre de comparaisons inter-blocs, par un algorithme de quantification vectorielle de l'espace de recherche. La seconde vise à réduire le nombre de blocs traités tout en améliorant les résultats visuels, pour des taux de compression élevés. Ceci est fait en introduisant des quadrilatères dans la triangulation de l'image. Nous concluons le mémoire en commentant différents résultats de décompression obtenus à partir des partitionnements étudiés, puis comparons ces résultats à ceux obtenus à partir de méthodes hybrides liant le codage par fractales à une décomposition multirésolution de l'image ou à la transformée en cosinus discrète.

Abstract

This thesis deals with fractal compression of still images, based on the theory of iterated function systems (IFS). After an overview of the main lossy and lossless still image compression methods, we introduce the IFS theory. Then, we detail the main fractal compression algorithms proposed in the literature. A coder involves comparisons between two sets of blocks. A block in the partition of the original image must be approximated by a collage block, with a contractive function. We therefore present different partitioning schemes used for the modelisation of the local-similarities in the images. Thereafter, our algorithm based on the Delaunay triangulation is explained, which constitutes the major contribution of our work. This flexible scheme allows to construct different triangulations, in a image content dependant way. In order to improve the coder-decoder, we propose different solutions. The first method allows to reduce the encoding complexity by the use of a scheme for quantizing the blocks. The second method makes use of a mixed partition composed of triangles and quadrilaterals in order to improve the visual decoding quality at low bit rates. To conclude this work, we compare decoding results computed on different partitioning schemes and we do a comparison between the block based fractal methods and recent hybrid methods merging fractal image compression and wavelet, or Fourier transform methods.

Notations

partition R	partition de l'image composée de blocs destination;
partition D	partition de l'image composée de blocs source;
$\mathbf{r}_n$	bloc destination numéro n. En anglais : *range block*;
$\mathbf{d}_{\alpha(n)}$	bloc source. En anglais : *domain block*;
$\mathbf{b}_2$	bloc source décimé, superposé au bloc destination;
$\mathbf{b}_1$	bloc constant, dont la luminance des pixels est égale à un;
collage	transformation ramenant un bloc source sur un bloc destination : transformation spatiale + transformation dans l'espace des luminances de manière à ce que le bloc transformé $\hat{\mathbf{r}}_n$ approxime le bloc destination $\mathbf{r}_n$;
transformation fractale	transformation d'une image A à partir de transformations élémentaires ω_n composant l'opérateur W finalement contractant, de manière à ce que l'image $W(A)$ approxime au mieux l'image A. Une transformation ω_n opère le collage du bloc source $\mathbf{d}_{\alpha(n)}$ sur le bloc destination $\mathbf{r}_n$.

Table des matières

Chapitre 1

Introduction générale

La société actuelle produit un nombre croissant de données qui doivent être traitées, transmises et/ou stockées. Celles-ci sont principalement des sons, des images ou des textes et proviennent de différents secteurs tels que par exemple la physique, la médecine, la biologie, l'industrie, la culture, le tourisme ou la finance. La représentation de ces informations sous forme numérique fiabilise leur transmission au travers des réseaux informatiques et facilite leur manipulation. La numérisation présente cependant un inconvénient : elle requiert que les dispositifs de stockage ainsi que les largeurs des bandes passantes des lignes de transmission soient suffisamment importants. Ceci n'est pas toujours possible et il faut dans ce cas faire appel à des algorithmes de compression des données.

Nous nous intéresserons dans le cadre de cette thèse à la compression des images numériques fixes en niveaux de gris, tout en faisant remarquer que les techniques étudiées peuvent être généralisées dans des schémas de compression d'images multicanaux ou de séquences vidéo (codage intra-image).
L'idée de base de la compression des images est de réduire le nombre moyen de bits par pixel nécessaire à leur représentation. Il est possible dans une certaine limite de réduire ce nombre sans perdre d'information. Au delà, il est nécessaire d'élaborer des algorithmes de compression (irréversibles) induisant une distorsion pas ou peu visible dans les conditions normales d'observation des images.
Différentes méthodes de compression ont été étudiées dans la littérature. Celles-ci peuvent se diviser en deux classes dont la frontière est de plus en plus difficile à déterminer : les méthodes par transformation et les méthodes spatiales.

Les méthodes par transformation consistent à décomposer l'image sur une base de fonctions orthogonales puis à quantifier de manière scalaire ou vectorielle les coefficients (spectraux) décorrélés issus de la transformation. Il est à noter que le fait de quantifier les coefficients induit une perte d'information et rend ainsi la compression irréversible. La transformée en cosinus discrète [134] fait partie des méthodes les plus utilisées. Elle doit être calculée sur des blocs de petite taille de manière à adapter la quantification aux propriétés statistiques locales de l'image. La transformée en ondelettes orthogonales ou bi-orthogonales est à la fois bien localisée en fréquence et dans l'espace, et elle admet la non-stationarité du signal. Pour cette dernière raison elle peut être calculée sur l'image entière. Elle permet de décrire l'évolution spatiale de l'image à différentes échelles d'observation. La décomposition de l'image en sous-bandes est très proche de la transformée en ondelettes puisque toutes les deux sont calculées en pratique à l'aide de filtres numériques suivant un algorithme pyramidal et fournissent une représentation multi-résolution de l'image. La quantification des coefficients des transformations permet la compression de l'information. Elle est dans ces deux cas réalisée de manière vectorielle [68].

Les méthodes spatiales sont généralement basées sur un partitionnement géométrique du support de l'image et opèrent directement sur des blocs de pixels. Les méthodes

de compression dites de seconde génération [97] basées sur des algorithmes de segmentation régions-contours, ainsi que celles qui opérent des transformations morphologiques sur l'image peuvent également être affectées à cette classe de méthodes. En marge de celle-ci, les méthodes prédictives [87] éliminent l'information redondante entre les pixels voisins de l'image en ne codant que la différence entre la valeur d'un pixel courant et sa valeur prédite localement.

Nous présentons dans le cadre de cette thèse une méthode de compression des images selon une approche fractale [111] [56]. La méthode opère sur une partition de l'image et cherche à exploiter les redondances entre des blocs de pixels à diverses résolutions. On parle dans ce cas de transformation (géométrique) fractale, basée sur un opérateur finalement contractant. Nous faisons ci-après un bref rappel sur l'historique de la compression par fractales puis donnons le plan de la thèse.

Suite aux travaux de Hutchinson [81] en 1981, Barnsley, Demko et d'autres chercheurs du "Georgia Institut of Technology" d'Atlanta ont démontré dans une série d'articles entre 1985 et 1988 ([9] [11] [14] [10] [13]) l'intérêt d'utiliser la théorie des fractales pour coder les images numériques. La méthode, basée sur le *théorème du collage* [8], montre qu'il est possible d'approximer un objet fractal binaire défini dans le plan à l'aide de quelques transformations contractantes définissant un *système de fonctions itérées (IFS)*. L'objet est dit aussi *auto-similaire* dans le sens où il est composé de l'union de transformations contractantes de lui-même. L'approximation d'un objet donné constitue un problème inverse difficile à résoudre de manière automatique et Barnsley proposait à cette époque une solution "manuelle". L'idée fut ensuite généralisée aux objets en niveaux de gris en les approximant, toujours de façon manuelle, à l'aide de mesures invariantes normalisées définies sur un support fractal du plan [8]. L'approximation des images naturelles peut être faite de la même manière en les considérant comme étant formées de l'union d'objets auto-similaires, chacun étant approximé indépendamment par une mesure invariante. Les images initialement présentées par M. Barnsley étaient obtenues de cette manière, mais constituaient des approximations très grossières des images réelles [168] [8].

Sur la base de ces travaux, Jacquin a proposé en 1989 [84] une approche fractale ne nécessitant pas d'intervention humaine et permettant de coder une image naturelle. La méthode est basée sur une série de transformations affines contractantes et "locales" définissant un opérateur contractant. Elle a l'avantage d'être automatique mais ne résout cependant pas directement le problème inverse décrit par Barnsley puisque l'image n'est pas considérée comme une union de transformations d'elle-même mais comme une union de sous-parties transformées d'elle-même. Dès 1989, de nombreuses autres recherches ont débuté [60] [148] [86] visant toutes à accélérer la phase de calcul des transformations locales, et/ou à répondre au compromis taux de compression-distorsion. Une formulation algébrique de l'opérateur contractant a permis de définir d'autres types de transformations locales plus optimales tout en contrôlant sa propriété de contraction. Fisher a proposé de réduire le nombre

de transformations en adaptant le partitionnement à l'image. Il a pour cela utilisé un partitionnement en quadtree puis un partitionnement rectangulaire [60]. En parallèle à ces travaux visant à définir un opérateur optimal agissant dans l'espace des niveaux de gris, des chercheurs utilisent actuellement la théorie des IFS ainsi que l'extension proposée par Jacquin dans des schémas hybrides basés sur la transformée en ondelettes ou en cosinus discrète.

Notre approche vise à montrer l'intérêt d'utiliser un modèle de partitionnement géométrique beaucoup plus souple que ceux proposés jusqu'à présent [44] [45] [42] et à apporter différentes améliorations au schéma de compression développé. Cette caractéristique de souplesse permet d'envisager ce travail dans une perspective de développement sur les mailles actives qui font l'objet actuellement de nombreux travaux en vue du codage des séquences d'images.

L'organisation générale du mémoire est décrite ci-dessous.

Le chapitre 2 est consacré à la présentation des principales méthodes de codage entropique et de compression réversibles et irréversibles des images fixes en niveaux de gris. Ces méthodes seront évoquées de nouveau au terme de ce mémoire dans un contexte de comparaisons avec l'approche développée.

Le chapitre 3 introduit les notions nécessaires à la compréhension de la théorie des systèmes de fonctions itérées (IFS) et présente les principaux algorithmes de compression des images naturelles selon l'approche fractale sur des blocs de pixels.

Le chapitre 4 est consacré à l'étude des partitionnements déjà utilisés en compression d'images par fractales et présente notre approche fondée sur le modèle de partitionnement triangulaire de Delaunay. La souplesse de ce partitionnement nous permet de proposer trois triangulations adaptées au contenu de l'image à compresser. La première est implantée dans un contexte algorithmique de type division-fusion en considérant la variance des niveaux de gris à l'intérieur de chacun des triangles. La deuxième partition est implantée dans un contexte de type division seule à partir d'un ensemble dense de points disposés sur le support de l'image. La troisième partition est contrainte par les contours de l'image et permet de se rapprocher des méthodes basées régions-contours mentionnées auparavant.

Dans le chapitre 5, nous montrons comment calculer la transformation fractale d'une image naturelle à l'aide des triangulations décrites dans le chapitre 3. Nous détaillons l'algorithme de compression-décompression et effectuons une étude comparative des différents partitionnements triangulaires proposés.

Le chapitre 6 présente des améliorations du schéma de compression-décompression (hybrides ou non) proposées par différents chercheurs. Dans ce contexte, nous montrons comment optimiser notre propre algorithme. La première solution que nous

proposons [41] vise à réduire la complexité de la phase de codage en diminuant le nombre de comparaisons inter-blocs. Elle est fondée sur la quantification des vecteurs histogrammes du contenu des triangles. La seconde solution [43] [47] vise à réduire le nombre de blocs traités et par conséquent le nombre de transformations locales de façon à améliorer les résultats visuels pour des taux de compression élevés. Ceci est fait en introduisant des quadrilatères au sein de la triangulation de l'image. Nous concluons ce chapitre en commentant différents résultats de décodages fractals obtenus à partir des modèles de partitionnement étudiés. Nous comparons aussi ces résultats à ceux obtenus à partir de méthodes hybrides basées sur une repésentation en sous-bandes de l'image et sur la transformée en cosinus discrète.

Enfin nous concluons ce travail dans le chapitre 7 en dégageant ses points importants, en soulignant ses limites et en discutant les perspectives de recherche sur la compression des images par fractales.

Chapitre 2

Compression des images numériques fixes

2.1 Introduction

Les méthodes de compression et de codage réduisent le nombre moyen de bits par pixel à stocker ou à transmettre, en exploitant la redondance informationnelle de l'image. Les techniques de compression se différencient par le fait qu'elles permettent ou non de compresser sans perte d'information, c'est-à-dire de manière réversible. Nous présentons dans ce chapitre les principales méthodes de codage et de compression des images fixes. Parmi les méthodes de compression non réversibles, nous présentons celles qui prennent en compte directement les pixels de l'image (méthodes spatiales), les méthodes par transformations, ainsi que des méthodes hybrides.

2.2 Besoins en compression

- une image couleur représentée dans l'espace Rouge-Vert-Bleu, de taille 512×512 pixels[1], dont chacune des composantes est codée sur 8 bits par pixel représente 786 Kilo-octets;
- un film négatif 24×36 mm numérisé à 12 μm par point retourne une image de taille 3000×2000 pixels par couleur, 8 bpp, 3 couleurs, ce qui représente 18 Méga-octets;
- une image LANDSAT : 6000×6000 pixels par bande spectrale, 8 bpp, 6 bandes, représente 216 Méga-octets;
- la transmission d'une séquence vidéo 512×512, 8 bpp, 3 couleurs sur une ligne téléphonique avec un modem à 9600 bauds nécessite 11 minutes par image;
- la transmission d'une séquence d'images couleur au format QCIF échantillonnée à 30 Hertz représente un débit de 9,12 Méga-bits par seconde, et de 36,40 Mégabits par seconde au format CIF[2].

2.3 Nature des images

Les images à compresser peuvent être de différentes natures :

- images photographiques : ce sont généralement des images de scènes naturelles dans lesquelles l'intensité lumineuse varie de manière relativement continue;
- les images "modales" dans lesquelles l'intensité lumineuse est très changeante localement (dessins manuels par exemple). Leur histogramme est multimodal.

La numérisation des images peut aussi se faire de différentes manières, retournant :

- les images binaires, qui contiennent seulement deux niveaux de gris différents;

1. pixel signifie en anglais picture element

2. Le format QCIF (Quart de CIF) est défini par 176 pixels sur 144 lignes pour la luminance et 88×72 pixels pour les chrominances. Le format CIF (Common Intermediate Format) est défini par 352 pixels sur 288 lignes pour la luminance et 176×144 pixels pour les chrominances.

- les images multi-niveaux numérisées sur plus d'un bit par pixel;
- les images multi-canaux dont l'exemple classique est celui des images couleurs.

2.4 Classification des méthodes

Les méthodes de compression visent à enlever la redondance présente dans l'image de manière à diminuer le nombre de bits nécessaires à sa représentation. Plusieurs types de redondance en terme de corrélation peuvent être considérés :

- la redondance spatiale entre pixels ou blocs voisins dans l'image;
- la redondance spectrale entre plans de couleur ou bandes spectrales;
- la redondance temporelle entre images successives dans une séquence vidéo.

Les méthodes de compression peuvent se regrouper en deux classes :

- les méthodes sans perte d'information (réversibles) : le taux de compression est limité par l'entropie de l'image.

- les méthodes avec perte d'information (irréversibles) : le taux de compression est sensiblement supérieur à l'entropie de l'image.

La technologie actuelle est capable de compresser une image couleur à $0,25$ bpp ($T_c = \frac{8}{0.25} = 32$) sans la dégrader, en considérant que la personne regardant l'image se trouve à une distance de 6 fois la hauteur de l'image.

2.5 Eléments de théorie de l'information

A la fin des années 1940, les premières idées sur la compression des données émergeaient, avec le développement de la théorie de l'information. Les chercheurs parlaient de concepts tels que l'entropie, la quantité d'information, la redondance, sous l'impulsion de C. Shannon [151]. Une opinion voulait que si la probabilité d'un symbole était connue, il devait exister une manière de le coder efficacement.

L'idée principale de la théorie de l'information [90] est de dire que si il n'y a pas d'incertitude vis à vis du message émis par la source, il n'y a pas d'information à la réception du message. Considérons une *source* S définie par deux quantités :

- un ensemble fini de N symboles $[S]_N = [s_1, s_2, \ldots, s_N]$ appelé alphabet.

- un mécanisme d'émission de suites de tels symboles, suivant une loi de probabilité donnée $[P]_N = [p(s_1), p(s_2), \ldots, p(s_N)]$, avec $\sum p(s_i) = 1$ et $\bigcup s_i =$ événement certain.

Différents types de sources

Une source est dite *simple* ou *sans mémoire* si :

- Les symboles successifs émis par la source sont indépendants (variables aléatoires indépendantes et de même loi).
- La probabilité d'une suite de symboles $S_n = s_{t_1} s_{t_2} s_{t_3} \dots s_{t_n}$ émis à des instants successifs t_j est donnée selon l'égalité (1). La source simple est équivalente à une suite de variables aléatoires indépendantes à valeurs dans $[S]_N$.

$$p(s_{t_1}, s_{t_2}, s_{t_3}, \dots, s_{t_n}) = p(s_{t_1}) p(s_{t_2}) \dots p(s_{t_n}) \quad (1)$$

Une source est dite de *Markov à l'ordre r* si l'apparition du symbole s_t est conditionné par seulement r symboles précédents :

$$p(s_t | s_{t-1}, s_{t-2}, s_{t-3}, \dots) = p(s_t | s_{t-1}, s_{t-2}, s_{t-3}, \dots, s_{t-r}), \forall t$$

Par exemple, la parole ne peut pas être modélisée correctement par un modèle de source simple. On préfère utiliser un modèle de Markov d'ordre un ou deux pour prendre en compte les effets de liaison entre syllabes, et de coarticulation.

Dans ce qui suit, nous ne considérerons que des sources simples.

Information associée au symbole d'une source

L'information $\imath(s_i)$ associée au résultat s_i est fonction de la probabilité d'apparition de s_i : $\imath(s_i) = F(p(s_i))$. Trois conditions permettent de déterminer la fonction F :

- si la source ne délivre qu'un seul message, l'information associée au message est nulle : $F(1) = 0$.
- soit $s_i = s_{i1} \cup s_{i2}$ égal à la réunion de deux événements indépendants. L'information $\imath(s_i)$ doit être égale à la somme des informations associées à s_{i1} et s_{i2} : $F(p(s_i)) = F(p(s_{i1})) + F(p(s_{i2}))$. Or, de par l'indépendance, la probabilité $p(s_i)$ est égale à $p(s_{i1}).p(s_{i2})$. Donc $F(p(s_{i1}).p(s_{i2})) = F(p(s_{i1})) + F(p(s_{i2}))$. La fonction F est additive.
- F est continue, monotone et positive.

La fonction F vérifiant ces 3 conditions est $-\lambda \log(.)$. La quantité d'information associée au message s_i est donc donnée par $\imath(s_i) = -\lambda \log(p(s_i))$. Le coefficient λ est choisi de manière à rendre égale à 1 la quantité d'information associée aux symboles d'une source pouvant générer r symboles différents, équiprobables. Dans ce cas $-\lambda \log(\frac{1}{r}) = 1$ et $\imath(s_i) = -\log_r(p(s_i))$ unités. L'unité exprimant la quantité d'information $\imath(s_i)$ dépend du coefficient de la base du logarithme. Le cas le plus classique revient à considérer la source comme une source binaire équiprobable ($r = 2$). L'unité est dans ce cas appelée *bit*, et la quantité d'information est donnée par :

$$\imath(s_i) = -\log_2(p(s_i)) \quad \text{bits.}$$

Entropie d'une source simple

L'entropie $H(S)$ d'une source simple $[S]_N$ associée à une loi de probabilité $[P]_N$ est définie selon la formule suivante :

$$H(S) = -\sum_{i=1}^{N} p(s_i) \log_2(p(s_i)) \quad \text{bits.}$$

$H(S)$ est une appréciation numérique globale attachée à une réalisation de la source. Elle définit l'information moyenne de chaque message (symbole) de la source.

Propriétés de $H(S)$

- $H(S)$ est maximale si tous les symboles $[s_1, s_2, \ldots, s_N]$ de $[S]_N$ sont équiprobables. On a alors $p(s_i) = \frac{1}{N}$ et $H(S) = \log_2 N$ bits. Dans ce cas, l'entropie de la source est égale à l'information associée à chaque message pris individuellement.
  Exemple : au jeu de pile ou face (source binaire) on associe deux messages équiproblables. La probabilité $p(s_i)$ est égale à $\frac{1}{2}$ et l'entropie $H(S)$ à 1 bit.

- La composition des événements fait décroître l'entropie.
  Soit $([S]_N, [P]_N)$ une source composée de N symboles. La composition consiste à regrouper les symboles s_i de $[S]_N$ de façon à générer une nouvelle source $[R] = [Y, Z]$ composée de deux groupes Y et Z avec
  $P_Y = \sum_{i=1}^{k} p(s_i)$ et $P_Z = \sum_{i=k+1}^{N} p(s_i)$. Dans ce cas, $H(S) > H(R)$.
  EXEMPLE : regroupement des valeurs positives et négatives de $[S]_N$.

- La scission des événements accroît l'entropie (extension à l'ordre N).
  EXEMPLE : Soit S une source simple avec un alphabet de taille n. Regroupons les symboles de la source en blocs de N symboles. Chaque bloc peut être vu comme un symbole généré par une source simple S^N avec un alphabet de sortie de taille n^N. On montre dans ce cas que l'entropie de la nouvelle source S^N est égale à N fois l'entropie de la source S. Une telle opération de regroupement en blocs de taille N réalise une *extension de la source à l'ordre N*.

2.5.1 Codage et décodage.

L'alphabet du codeur est formé de r symboles différents. A chaque message s_i de $[S]_N$ (séquence de symboles appartenant à l'alphabet de la source), le codeur fait correspondre un message (mot code) u_i de $[U]_N$ formé de l_i symboles.

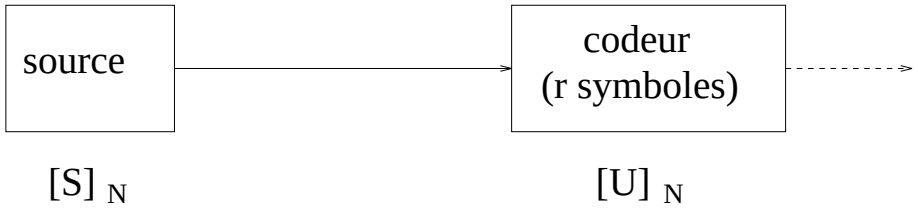


FIG. 1 - Schéma d'un codeur.

Définitions

1. La *longueur moyenne* des mots codes de $[U]_N$ est égale à $\overline{n} = \sum_{i=1}^{N} l_i.p(s_i)$.

2. L'ensemble des N mots codes de $[U]_N$ constitue le *code* de la source $[S]_N$.

3. Un code est dit *séparable* ou *déchiffrable* lorsqu'il peut être lu sans ambiguîté (on peut aussi utiliser des codes de même longueur ou un signe auxiliaire de séparation).

4. Un code séparable est dit *irréductible* s'il n'existe aucun couple $u_i u_j$ pour lequel le mot code u_i soit le début du mot code u_j : aucun u_i n'est le préfixe de u_j.

Inégalité de Kraft

Un code composé de N mots code de longueur l_i ($i = 1$ à N) peut être irréductible si et seulement si :

$$\sum_{i=1}^{N} \frac{1}{r^{l_i}} \leq 1$$

où r est la taille de l'alphabet du codeur (codeur binaire : $r = 2$).

Condition sur la longueur moyenne $\overline{n}$ des mots code d'une source S

La longueur moyenne $\overline{n}$ possède une limite inférieure, égale à l'entropie H de la source :

$$\overline{n} \geq -\sum_{i=1}^{N} p(s_i).\log_2 p(s_i).$$

On peut aussi montrer qu'une source d'entropie $H(S)$ peut toujours être codée de façon irréductible avec un code de longueur moyenne $\overline{n}$ telle que :

$$H(S) \leq \overline{n} < H(S) + 1.$$

Théorème du codage exact de Shannon

En codant des extensions d'ordre k[3] de plus en plus élevé, la longueur moyenne $\overline{n}$ des mots code tend vers une limite inférieure donnée par l'expression suivante :

$$\lim_{k\to\infty} \overline{n} = H(S). \tag{2}$$

Il est possible de coder sans erreur une source d'entropie égale à H bits à l'aide de mots de longueur moyenne $\overline{n} = H + \epsilon$ bits/symbole où ϵ est arbitrairement petit.

Démonstration :
Soit une source simple $[S]_N, [P]_N$ vérifiant : $H(S) \leq \overline{n} < H(S) + 1$.
La source $[S^2], [P^2]$ étendue à l'ordre 2 vérifie : $H(S^2) \leq \overline{n_2} < H(S^2) + 1$.
Son entropie est augmentée d'un facteur deux : $H(S^2) = 2.H(S)$.
L'extension de la source à l'ordre k donne : $k.H(S) \leq \overline{n_k} < k.H(S) + 1$.
en remarquant que $\overline{n_k} = k.\overline{n}$, nous avons $H(S) \leq \overline{n} < H(S) + \frac{1}{k}$ et donc $\lim_{k\to\infty} \overline{n} = H(S)$.

2.6 Compression réversible des données

2.6.1 Modélisation des données

Les méthodes de codage classiques élaborent une table de probabilités statique avant de construire le code. La table est calculée lors du codage puis transmise au décodeur. Elle peut aussi être calculée une fois pour toutes, servant ainsi au codage de plusieurs réalisations d'une source (images). Mais l'utilisation d'un tel modèle statique est dangereux car le flot d'entrée peut ne pas correspondre aux statistiques précédemment calculées. Le taux de compression est diminué. Une meilleure compression peut être atteinte en augmentant l'ordre de la modélisation (extension d'ordre plus élevé de la source). Le gain en compression réalisé est dans ce cas annulé par la taille exponentiellement croissante de la table de probabilités. Une image contenant 256 niveaux de gris, vue comme une réalisation d'une source simple (ordre 0) nécessite la construction d'une table de 256 probabilités. Une table d'ordre 1 contient quant à elle 65536 probabilités. Pour ces raisons, on utilise aujourd'hui des modèles adaptatifs. Les statistiques sont continuellement modifiées au cours de la lecture des symboles à coder. L'avantage de ces méthodes par rapport à celles basées sur des modèles statiques est leur capacité d'adaptation aux conditions locales.
Remarque : Deux opérations très étroitement liées composent les méthodes de compressions de données : la modélisation et le codage. Il existe un grand nombre de manières de modéliser des données. Un même processus de codage prenant ces données en entrée permettra des taux de compression très différents suivant les cas.

3. On rappelle que l'extension à l'ordre k d'une source S de n symboles est une nouvelle source S^k composée de blocs de k symboles. La nouvelle source a un alphabet de n^k symboles.

2.6.2 Définition du codage entropique

Considérons une image dont chaque pixel est quantifié sur B bits[4]. L'image (monochrome) possède 2^B niveaux de gris différents notés s_i ($i = 1 \ldots 2^B$). Si les niveaux de gris sont indépendants et équiprobables (probabilité du pixel $i = p_i = 2^{-B}$)), l'entropie H de l'image est égale à $\log_2 2^B = B$ bits. Chaque pixel porte la même quantité d'information, l'image est "incompressible". Si les niveaux de gris ne sont pas équiprobables (densité de probabilité non uniforme), l'entropie H est inférieure à B. Le but du codage entropique est de coder les pixels à l'aide de mots code de longueurs variables, égales à $-\log_2 p_i$ bits, de manière à ce que le nombre moyen de bits par pixel soit égal à l'entropie $H = -\sum_{i=1}^{2^B} p_i \log_2(p_i)$ bits ($H \leq B$). Le codage est dans le cas général sous-optimal car la quantité $(-\log_2 p_i)$ est rarement entière. La limite H peut être approchée en codant des extensions de la source (en regroupant les pixels).

2.6.3 Codage de Shannon-Fano

C. Shannon du laboratoire Bells et R.M. Fano du MIT ont développé à peu près en même temps une méthode de codage basée sur la simple connaissance de la probabilité d'occurrence de chaque symbole dans un message.

Algorithme de construction de la table de codes irréductibles

1. Les probabilités d'occurrence de chaque message sont placées dans une liste dans un ordre décroissant. La liste constitue la racine d'un arbre qui, pour l'instant, est une feuille.

2. Couper la liste en deux groupes de symboles S_0 et S_1, dont les probabilités totales sont aussi voisines que possible ($\simeq \frac{1}{2}$).

3. Le groupe S_0 est codé par un "0", le groupe S_1 par un "1".

4. Si un groupe S_i n'a qu'un seul élément, il est appelé "feuille terminale" et est inchangé. Sinon, la procédure reprend à l'étape 2 sur le groupe S_i.

La procédure de codage construit un arbre dont les suites de bits 1 ou 0 partant de la racine vers chacune des feuilles constituent les mots code du code.

2.6.4 Codage de Huffman

Le codage de Huffman crée des codes à longueur variable sur un nombre entier de bits [75]. L'algorithme considère chaque message à coder comme étant une feuille d'un arbre qu'il reste à construire. L'idée est d'attribuer aux deux messages de plus faible probabilité, les mots codes les plus longs. Ces deux mots codes ne se

4. Il est à noter qu'une compression importante a déjà été réalisée à ce niveau puisqu'une quantité infinie "d'éléments visuels" par unité de surface est approximée par l'intensité d'un pixel.

différencient que par leur dernier bit. Contrairement au codage de Shannon-Fano qui part de la racine d'un arbre et évolue par divisions successives, le codage de Huffman part des feuilles de l'arbre et, par fusions successives, redescend vers la racine [124].

Procédure de codage

1. Les probabilités d'occurrence de chaque message sont placées dans une liste dans un ordre décroissant. Nous dirons que la liste est composée d'enfants.

2. Les deux probabilités les plus faibles sont identifiées en fin de liste.

3. La somme des deux probabilités est placée à sa place dans la liste triée. Elle constitue un nœud parent. Les deux enfants sont retirés de la liste.

4. Le chemin "enfant de plus faible probabilité, parent" est codé par un 1, l'autre par un 0.

5. La procédure reprend à l'étape 2 jusqu'à ce qu'il ne reste plus qu'une probabilité dans la liste.

Bilan : Chaque message est codé par la suite de 0 et de 1 rencontrés sur le chemin le menant à la racine de l'arbre. Cette suite inversée constitue le code réel du message. On remarque qu'il existe plusieurs codes de Huffman possibles pour une distribution de probabilités donnée, mais que la longueur totale des codes sera toujours la même. L'algorithme retourne un dictionnaire ("codebook") composé de mots code de différentes longueurs. La probabilité de chacun des messages sert d'index (de point d'entrée dans le dictionnaire) pour retrouver les mots codes.

> EXEMPLE : Considérons une source $[S]_5, [P]_5 = [s_1, s_2, s_3, s_4, s_5], [0.4, 0.2, 0.15, 0.15, 0.1]$. L'entropie de la source est égale à 2.146. Le codage de Huffman retourne les mots codes de longueur variable suivants : [1, 000, 001, 010, 011]. La longueur moyenne des mots code est dans ce cas égale à 2.2 alors que les messages initiaux étaient codés sur 3 bits. Il serait de plus possible de se rapprocher de l'entropie en codant des extensions à l'ordre n supérieur à 1 de la source.

Reconstruction

Le décodeur a besoin de l'arbre de Huffman (dictionnaire) créé lors du codage. Celui-ci est transmis au décodeur, en en-tête du fichier contenant le flot de mots code. La reconstruction de chacun des messages codés se fait, après lecture du mot code, en partant de la racine de l'arbre de Huffman. La suite des bits du mot code indique le chemin menant au message lui correspondant. Le lecteur trouvera des exemples de codage-décodage dans diverses références concernant le codage des données telles que la référence [124].

Codage de Huffman modifié

L'algorithme de Huffman retourne un dictionnaire composé de mots codes de différentes longueurs. Les "points d'entrée" dans le dictionnaire dépendent de la probabilité d'occurrence des messages à coder. Généralement, lorsqu'un fichier à coder est de grande taille, il contient beaucoup de symboles (messages) ayant une faible probabilité d'apparition. Les mots codes codant ces symboles sont de grande taille (taille $\leq$ nombre de symboles différents dans le fichier), puisque cette dernière dépend de l'information associée au symbole.
Un algorithme de Huffman plus fréquemment utilisé consiste à réunir les symboles de plus faible probabilité dans un "symbole" plus probable appelé AUTRE et à calculer le dictionnaire contenant l'ensemble réduit de symboles. Un bit est ensuite ajouté au mot code du symbole AUTRE pour identifier le sous-symbole.
Une solution similaire est utilisée dans le standard de compression des facsimilés.

2.6.5 Codage arithmétique

Le codage de Huffman n'est pas optimal puisque la taille théorique du mot code d'un symbole s_i, donnée par $-\log_2 p(s_i)$, n'est pas entière. Considérons par exemple un symbole, avec une probabilité d'apparition égale à 0.9. La taille optimale du code est égale à 0.15, alors que la procédure de Huffman affecte un code sur 1 bit au symbole. Le codage de Huffman n'est optimal que si les probabilités des différents symboles de la source sont des puissances négatives de deux.

Le codeur arithmétique [124] consiste quant à lui à coder une chaîne de symboles par un nombre appartenant à l'intervalle [0,1[. Une étude complète du codeur est donnée dans la référence [74], et différents solutions pratiques ont été proposées [74] de manière à faciliter la compression des fichiers de grande taille sans être limité par la précision des calculs.

Nous décrivons brièvement ci-dessous la procédure de codage arithmétique dans le but d'en illustrer le principe, sachant que le décodage opère de manière inverse.

- Calculer la probabilité associée à chaque symbole dans la chaîne à coder.
- Associer à chaque symbole un sous-intervalle proportionnel à sa probabilité, dans l'intervalle [0,1[(l'ordre de rangement des intervalles sera mémorisé car nécessaire au décodeur).
- Initialiser la limite inférieure de l'intervalle de travail à la valeur 0 et la limite supérieure à la valeur 1.
- Tant qu'il reste un symbole dans la chaîne à coder :
  - largeur = limite supérieure - limite inférieure
  - limite inférieure = limite inférieure + largeur $\times$ (limite basse du sous-intervalle du symbole)

  - limite supérieure = limite inférieure + largeur × (limite haute du sous-intervalle du symbole)

- La limite inférieure code la chaîne de manière unique.

On remarque que le premier symbole de la chaîne fixe le premier chiffre après la virgule du code final.

2.6.6 Codage à base de dictionnaires

Les méthodes telles que celle de Huffman, Shannon-Fano sont basées sur des modèles statistiques plus ou moins complexes (ordres 0,1,2, ou modèles adaptatifs). Les méthodes de compression par dictionnaire ne traitent quant à elles pas directement une chaîne de bits codant un symbole mais codent un index pointant sur la chaîne dans un dictionnaire de symboles.

Les *méthodes statiques* utilisent un dictionnaire pré-défini pour coder les symboles d'un fichier. Le dictionnaire est construit avant que la compression ne commence, en rassemblant un échantillon de symboles représentatifs. Par exemple, si on veut coder un fichier texte écrit en langage C, le dictionnaire doit se concentrer sur des mots comme "read", "while", "printf". Le dictionnaire est utile au codeur et au décodeur car ceux-ci respectivement génèrent et recoivent un index pointant sur un mot dans le dictionnaire.

Les *méthodes adaptatives*: La compression commence sans dictionnaire, ou avec un dictionnaire minimum par défaut. Lors de l'évolution de l'algorithme, des nouvelles phrases, ou de nouveaux mots sont ajoutés, utilisés plus tard pour coder d'autres mots.

Compression avec fenêtre coulissante : l'algorithme LZ77

L'algorithme proposé par Ziv et Lempel est basé sur l'utilisation d'une fenêtre de texte, divisée en deux [166]. La première partie, de grande taille contient le texte déjà codé. La deuxième, composée d'une dizaine ou d'une centaine de caractères est un tampon de pré-lecture. Si le début du texte dans le tampon est déjà dans la première partie, le code correspondant est composé de la position du bloc dans la première partie, la longueur du bloc, ainsi que le premier symbole différent. Le codeur émet le code puis fait entrer "longueur + 1" symboles supplémentaires dans le tampon. Le processus se répète ainsi jusqu'à la fin du fichier à compresser. Si aucune correspondance n'est trouvée entre le symbole dans le tampon et le symbole dans la première partie de la fenêtre, le symbole est codé par 0,0 suivi de lui-même. La décompression ne nécessite pas de recherche ou de comparaison entre les différents blocs. L'algorithme lit le premier code, émet le bloc de symboles indiqué, suivi du symbole supplémentaire, décale et recommence.
Un inconvénient majeur de l'algorithme LZ77 est le temps de calcul important pour coder un fichier. Ceci est dû aux nombreuses comparaisons des blocs entre le tampon de pré-lecture et ceux de la première partie de la fenêtre. Le décodage est par contre

très rapide. Aujourd'hui, différents archiveurs (arj, lha, zip, zoo) sont basés sur des algorithmes très proches du LZ77.

Extension : l'algorithme LZ78

L'algorithme LZ77 utilise une fenêtre de taille fixe, divisée en deux sous-parties. Celle-ci ne permet pas d'exploiter correctement l'aspect récent du texte, car elle oublie les mots lorsque ceux-ci sont sortis de la fenêtre coulissante. De plus, la taille restreinte du tampon de pré-lecture limite la longueur des chaînes équivalentes.
Une solution à ces deux problèmes serait d'augmenter la taille de la fenêtre glissante, mais cela augmenterait la taille des mots pour coder les positions dans la fenêtre ainsi que pour coder les longueurs des blocs. Ziv et Lempel ont proposé en 1978 un algorithme de codage connu sous le nom LZ78 [167] que nous rappelons brièvement ci-dessous :

- chaîne courante = vide
- dictionnaire = chaîne courante
- répéter
  - lire le caractère et l'ajouter à la chaîne courante
  - si chaîne courante est dans l'arbre constituant le dictionnaire alors continuer
  - sinon
    - ajouter la chaîne courante dans le dictionnaire, et étiquetter la chaîne
    - émettre l'étiquette de la chaîne courante sans le dernier caractère
    - émettre le dernier caractère
    - chaîne courante = vide
- jusqu'à ce qu'il n'y ait plus de caractère à coder.

Une variante de cet algoritme connue sous le nom LZC est utilisée dans le programme COMPRESS compilé pour s'exécuter sous le système d'exploitation UNIX.

2.6.7 Compression réversible des images

La représentation d'une image directement en terme de pixel est inefficace puisque la corrélation inter-pixels est trop importante. L'entropie de l'image LENA codée sur 8 bits par pixel est estimée à 7.57 bits, en supposant que les valeurs des pixels sont indépendantes. Cette valeur fixant le nombre moyen minimum de bits par pixel pour coder l'image de manière exacte ne permet pas de compresser sensiblement l'image. Le but est donc de trouver une représentation de l'image de manière à décorréler les pixels, et donc à réduire son entropie. Différentes techniques ont été proposées, et peuvent se regrouper en deux classes : les techniques prédictives, et les techniques par transformées. Celles-ci sont non-réversibles si le décodeur ne reconstruit pas exactement l'image originale et réversibles dans le cas contraire.

Nous présentons dans cette section trois méthodes de compression réversibles. Il en existe bien sur d'autres que nous ne détaillerons pas dans le cadre de ce mémoire.

Le codage des images par longueur de séquence

Cette méthode [88], connue sous le terme anglais de RLE (Run Length Encoding) code une séquence de pixels identiques sur une ligne d'image à l'aide de 3 paramètres : la position du premier pixel de la séquence dans l'image, la longueur de la séquence ainsi que la valeur du premier pixel. La méthode RLE est utilisée pour compresser des images ayant un nombre de niveaux de gris très limité. Elle est très comparable aux méthodes de compression à base de dictionnaires.

Le codage des images par plans de bits

Une image codée sur 2^n niveaux de gris (n bits par pixel) peut être considérée comme une superposition de n plans, chacun de hauteur 1 bit, en isolant à chaque fois un bit de même poids pour chaque pixel. Chaque plan de bits (image binaire) peut être codé séparément en utilisant la méthode RLE. Le code de Gray est dans ce cas utilisé pour augmenter la cohérence au sein des différents plans de bits.
Le codage réversible par plans de bits permet des taux de compression compris entre 1.5 et 2, mais présente l'inconvénient d'être sensible vis à vis des erreurs de transmission. Les plans composés des bits de poids fort contiennent la majeure partie de l'information visuelle de l'image.

Le codage prédictif MICD sans perte

L'idée du codage prédictif (Modulation par Impulsions Codées Différentielles) est de supprimer la redondance entre pixels voisins et de ne coder que la différence entre la valeur du pixel courant et sa valeur prédite à partir des pixels voisins [88].
Considérons un échantillon (pixel) $u(n)$. La quantité $\overline{u}(n)$, valeur prédite de $u(n)$, est calculée à partir des échantillons d'entrée précédents :
$\overline{u}(n) = \phi(u(n-1), u(n-2), \dots)$ où ϕ est appelée règle de prédiction. L'erreur de prédiction $e(n) \stackrel{\text{def}}{=} u(n) - \overline{u}(n)$ est ensuite codée puis transmise ou stockée.
Cette méthode de codage appliquée aux images opère sur des échantillons d'entrée à valeurs entières. En imposant au prédicteur de ne retourner que des valeurs entières, le codage est dit "sans perte". L'image de différence composée des échantillons $e(n)$ est beaucoup moins corrélée que l'image originale et a une variance très inférieure. L'entropie de celle-ci est donc inférieure à celle de l'image originale composée des échantillons $u(n)$. L'information est dans ce cas compressée à l'aide d'un codeur entropique (Huffman ou arithmétique).

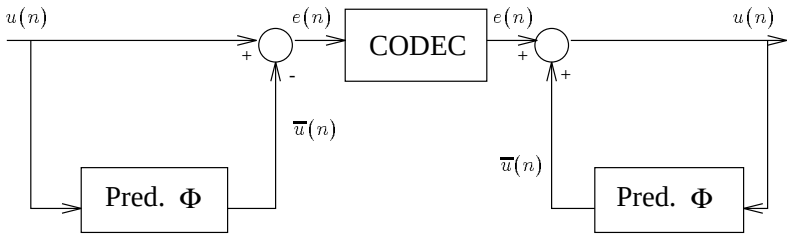


FIG. 2 - Codage réversible par prédiction

2.7 Compression non réversible des images

2.7.1 Mesure de la qualité visuelle de l'image reconstruite

La mesure de la performance d'un codeur en terme de qualité visuelle de l'image reconstruite se fait généralement en mesurant le rapport signal à bruit ou le rapport signal à bruit de crête dont les expressions sont rappelées ci-dessous :
Le rapport signal à bruit, noté SNR, entre l'image originale composée de pixels $x(m,n)$ et l'image décodée composée de pixels $\hat{x}(m,n)$ est donné par :

$$\begin{aligned} SNR &= 10\log_{10}\frac{E[x(m,n)^2]}{E[(x(m,n)-\hat{x}(m,n))^2]} \\ &= 10\log_{10}\frac{\sum_m\sum_n (x(m,n))^2}{\sum_m\sum_n (x(m,n)-\hat{x}(m,n))^2} \\ &= 10\log_{10}\frac{E[x(m,n)^2]}{MSE} \end{aligned}$$

Le rapport signal à bruit de crête, noté PSNR, (plus souvent utilisé en compression) est donné par :

$$PSNR = 10\log_{10}\frac{255^2}{E[(x(m,n)-\hat{x}(m,n))^2]}$$

Ces mesures sont très utilisées car très simples à calculer. Ce ne sont cependant pas des mesures fiables de ressemblance visuelle entre deux images [37]. Un contre exemple consiste à prendre deux images composées d'un bruit blanc. L'erreur MSE est très grande entre les deux images alors qu'elles sont très semblables visuellement.

2.7.2 Quantification scalaire

Quantification scalaire uniforme

Un quantificateur scalaire est un opérateur qui associe à une variable continue u une variable discrète u' pouvant prendre un nombre plus faible, et fini de valeurs [5].

5. Il est à noter que les méthodes de codage utilisant un quantificateur ne sont jamais réversibles parce que l'étape de quantification introduit inévitablement une distorsion.

Pour un nombre de niveaux de quantification fixé L, on peut choisir les régions de décision $\{t_k, k = 1 \dots L+1\}$ ainsi que les seuils de décision $\{r_1 \dots r_L\}$ de façon à minimiser la distorsion entre l'entrée et la sortie. Si u se trouve dans la région $[t_k, t_{k+1})$, u' aura pour valeur r_k.

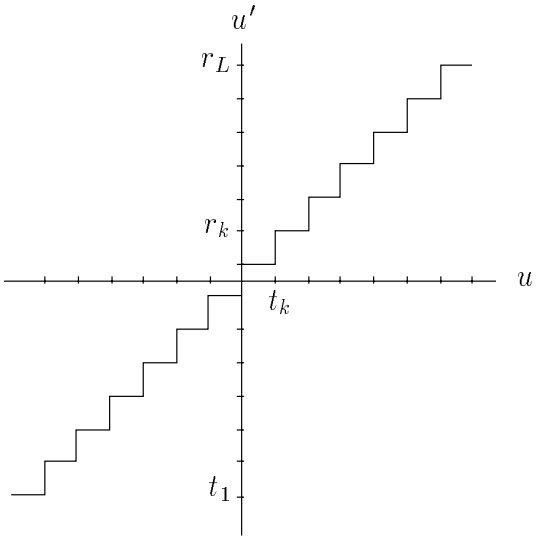


FIG. 3 - Quantification scalaire uniforme

Quantificateur scalaire optimal de Lloyd-Max

Le quantificateur de Lloyd-Max minimise l'erreur quadratrique moyenne entre l'entrée et la sortie, pour un nombre de niveaux de quantification donné L. Cette méthode a été indépendamment proposée par Lloyd en 1957 [107] et Max en 1960 [113]. La minimisation de l'erreur quadratique de quantification donnée par l'expression :

$$E[(u-u')^2] = \int_{t_1}^{t_{L+1}} (u-u')^2 p_u(u) du$$

conduit aux valeurs optimales suivantes :

$$t_k = \frac{r_k + r_{k-1}}{2}$$

$$r_k = \frac{\int_{t_k}^{t_{k+1}} u \; p_u(u) du}{\int_{t_k}^{t_{k+1}} p_u(u) du}$$

où $p_u(u)$ est la densité de probabilité continue de la variable scalaire u (approximée en pratique par l'histogramme de l'image). Si la densité de probabilité du signal d'entrée u est uniforme, le quantificateur optimal de Lloyd-Max est un quantificateur uniforme (appelé aussi quantificateur linéaire) avec $q = \frac{t_{L+1}}{L}$, $t_k = t_{k-1} + q$, $r_k = t_k + \frac{q}{2}$.

L'erreur quadratique de quantification est dans ce cas donnée par :

$$E[(u-u')^2] = \frac{1}{q}\int_{-\frac{q}{2}}^{\frac{q}{2}} u^2 du = \frac{q^2}{12}.$$

L'inconvénient est qu'en général, la densité de probabilité des signaux à quantifier n'est pas uniforme. Le calcul des seuils et des régions de quantification est alors complexe. Deux lois de probabilité sont souvent utilisées en fonction de la nature de la source :

- la loi Gaussienne : $p_u(u) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-(u-\mu)^2}{2\sigma^2})$
- la loi Laplacienne : $p_u(u) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{2}{\sigma^2}|u-\mu|)$.

où σ^2 et μ désignent la variance et la moyenne de u. Des tables donnent les seuils pour différentes valeurs de L, dans le cas des lois de Gauss, Laplace ou Rayleigh.

Quantification visuelle

Lorsqu'une image est quantifiée avec un nombre trop faible de niveaux, des faux contours apparaissent. Les pixels voisins de niveaux de gris similaires se regroupent pour former des zones homogènes dont les bords forment des contours.
Généralement, les images sont uniformément quantifiées. Chaque pixel codé sur 8 bits peut prendre une parmi 256 valeurs de niveaux de gris. Les faux contours apparaissent lorsque les pixels sont codés sur 6 bits.

- *La quantification de contraste* : l'image u est transformée à l'aide d'une fonction non linéaire en une image de contraste, puis quantifiée. Une transformation inverse donne l'image résultat u'.
- *La quantification pseudo-aléatoire* : cette méthode ajoute du bruit uniforme dans l'image à quantifier de manière à réduire la taille des zones de pixels à valeurs identiques. Cela a pour effet d'éviter l'apparition de faux contours lorsque l'image est quantifiée avec trop peu de niveaux de quantification.

2.7.3 Codage prédictif MICD (DPCM)

La principale différence entre le codage MICD sans perte d'information (section 2.6.7) et le codage MICD non réversible est due à l'utilisation ou non d'un quantificateur.

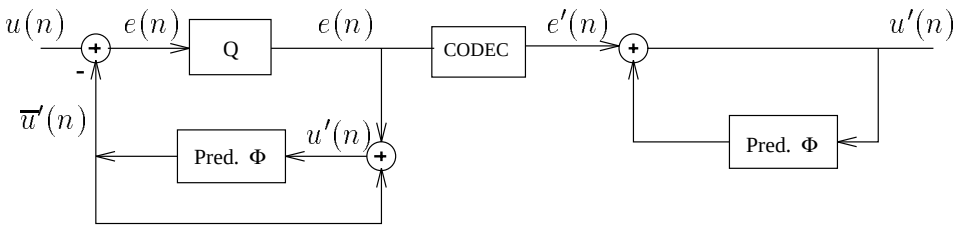


FIG. 4 - Codeur MICD.

La boucle de prédiction dans le codeur n'est autre que celle constituant le décodeur. La quantité $\overline{u}'(n)$, valeur prédite, est calculée à partir des échantillons précédemment décodés $u'(n-1), u'(n-2), \ldots$. Elle est donnée par :

$$\overline{u}'(n) = \phi(u'(n-1), u'(n-2), \ldots).$$

La différence entre la valeur originale $u(n)$ et la valeur décodée $u'(n)$ représente l'erreur de reconstruction notée $q(n)$. La valeur $q(n)$ est aussi égale à $e(n) - e'(n)$ et donc à l'erreur de quantification de $e(n)$.

Le prédicteur MICD reçoit en entrée des échantillons quantifiés. Il prend ainsi en compte les erreurs de quantification au cours du codage d'une séquence de pixels et évite une accumulation d'erreurs dans le signal reconstruit $\overline{u}(n)$.

Avantages du codeur MICD par rapport au codeur MIC

Le codeur MIC est composé d'un simple quantificateur recevant directement les échantillons du signal d'entrée $u(n)$. Aucune prédiction et rétroaction n'est mise en jeu. Il est montré [88] que dans ce cas, le nombre moyen minimal de bits par pixel, noté R_{MIC}, nécessaire à la quantification d'une variable aléatoire gaussienne $u(n)$, tout en tolérant une distorsion $\sigma_q^2 = E[(u-u')^2]$ fixée est donné par :

$$R_{MIC} = \frac{1}{2}\log_2(\frac{\sigma_u^2}{\sigma_q^2}) \quad \text{bits par échantillon,}$$

où u' est l'échantillon quantifié, et σ_u^2 est la variance du signal d'entrée ((σ_q^2 est considéré inférieur à σ_u^2)). Dans le cas du codeur prédictif différentiel MICD, la quantité R_{MICD} est donnée par :

$$R_{MICD} = \frac{1}{2}\log_2(\frac{\sigma_e^2}{\sigma_q^2}) \quad \text{bits par échantillon,}$$

où σ_u^2 est la variance du signal $u(n)$, $q(n)$ l'erreur de quantification, $e(n)$ l'erreur de prédiction, $\sigma_q^2 = E[q(n)^2]$ et $\sigma_e^2 = E[e(n)^2]$.

Le gain réalisé en utilisant un codeur MICD est alors donné par :

$$R_{MIC} - R_{MICD} = \frac{1}{2}\log_2(\frac{\sigma_u^2}{\sigma_e^2}) \quad \text{bits par échantillon.}$$

La compression autorisée dépend du facteur $\frac{\sigma_u^2}{\sigma_e^2}$, et donc de la qualité de la prédiction des valeurs d'entrée $u(n)$, celle-ci étant directement liée à la redondance inter-pixels. Un simple prédicteur sous-optimal retournera toujours un signal prédit $e(n)$ de variance inférieure à celle de l'entrée $u(n)$, et donc le codage du vecteur $e(n)$ nécessitera un nombre de bits inférieur à celui nécessaire au codage du vecteur $u(n)$.

Ecriture de la règle de prédiction ϕ

La règle de prédiction la plus souvent utilisée est une combinaison linéaire de pixels voisins. Les coefficients du prédicteur peuvent varier en fonction des zones de l'image à coder (prédiction adaptative). Le prédicteur prend en compte les pixels précédents sur la même ligne (cas 1D) ou considère aussi les pixels voisins, sur la ou les lignes précédentes (cas 2D). Le nombre de pixels considérés pour prédire la valeur du pixel courant est appelé l'ordre du prédicteur. Il a été montré que la qualité visuelle de l'image reconstruite n'augmentait sensiblement plus à partir d'un ordre supérieur à 3.

Une solution largement utilisée pour calculer les coefficients optimaux de ϕ est de minimiser l'erreur quadratique moyenne de prédiction :

$$\sigma_e^2 = E[(x_m - \sum_{i=0}^{m-1} \alpha_i x_i)^2]$$

Cela revient à rendre l'erreur de prédiction orthogonale aux m données du prédicteur en vérifiant l'expression suivante :

$$E[(x_m - \sum_{i=0}^{m-1} \alpha_i x_i) x_i] = 0, \quad \forall i = 0 \ldots m-1.$$

Les m produits scalaires doivent être nuls. La résolution de ces m équations donnant les m coefficients fait intervenir des termes intermédiaires d'autocorrélation. La résolution du système de m équations à m inconnues demande trop de calculs pour que la prédiction se fasse pour chaque image en temps réel. Une solution est de calculer un prédicteur global qui puisse être utilisé pour un maximum d'images différentes.
Une fonction d'autocorrélation séparable fréquemment utilisée en traitement d'images est donnée (pour des images à valeur moyenne nulle) par :

$$R_{k,l} = \sigma^2 \rho_v^{|k|} \rho_h^{|l|}$$

où σ^2 est la variance de l'image, k et l sont les déplacements horizontaux et verticaux, et ρ_h (resp. ρ_v) est le coefficient de corrélation dans la direction horizontale et (resp. verticale).

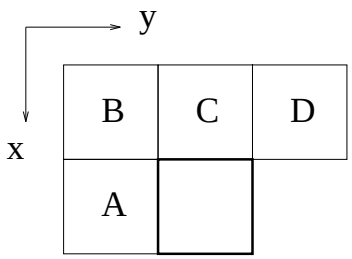


FIG. 5 - Prédiction à partir des pixels précédents.

Par exemple, un prédicteur d'ordre 4 dont les pixels intervenant dans le calcul des coefficients sont ceux de la figure 5 est donné par $x_m = \rho_h A - \rho_v \rho_h B + \rho_v C$ (cette formule est correcte pour des images à valeur moyenne nulle).

Nous donnons ci-après quelques exemples de prédicteurs d'ordre 1, 2 ou 3 utilisés pour le codage d'images quelconques :
$x_m = 0.97A$
$x_m = 0.5A + 0.5C$
$x_m = 0.9A - 0.81B + 0.9C$
$x_m = 0.75A - 0.5B + 0.75C$
$x_m = A - B + C$

2.7.4 Quantification vectorielle

Les techniques de compression d'images exploitent généralement la redondance statistique présente dans l'image. Les différentes méthodes vues jusqu'ici effectuent une quantification scalaire sur des échantillons (pixels individuels de l'image). La compression par transformée (DCT ...) quantifie les échantillons issus de blocs transformés d'images. Le codage prédictif quantifie une erreur (différence) entre l'échantillon courant et sa prédiction. Mais ces valeurs ne sont jamais totalement décorrélées, ou indépendantes. Shannon a montré qu'il était toujours possible d'améliorer la compression de données en codant des vecteurs plutôt que des scalaires, cela étant vrai même si la source est sans mémoire (simple). La quantification vectorielle, développée par Gersho et Gray (1980) fait aujourd'hui l'objet de nombreuses publications dans le domaine de la compression des images numériques [68].

Principe de la quantification vectorielle

Un quantificateur peut être vu comme une application Q associant à chaque vecteur d'entrée $X_i = (x_j, j = 1 \ldots k)$ un vecteur $Y_i = (y_j, j = 1 \ldots k) = Q(X_i)$ choisi parmi un dictionnaire de taille finie, $C = (\hat{X}_l, l = 1 \ldots N_c)$. C peut être vu comme un catalogue de formes.

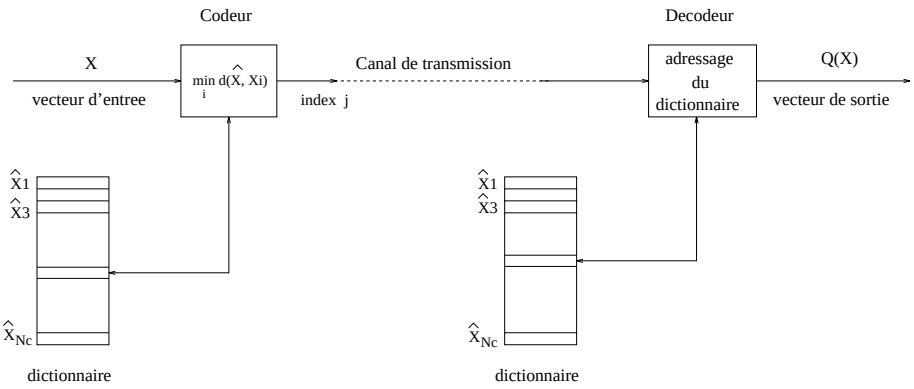


FIG. 6 - Principe du quantificateur vectoriel.

Le quantificateur est complètement décrit par :

- le dictionnaire (codebook) C.
- le partitionnement $S = (S_i, i = 1 \dots N_c)$ qui divise l'espace d'entrée en N_c vecteurs X_i, et qui leur fait correspondre un vecteur (reconstruit) $Q(X_i) = \hat{X}_i$.

La quantification vectorielle peut aussi être vue comme une combinaison de deux fonctions :

- un codeur qui prend en entrée un vecteur X_i et qui recherche dans un dictionnaire l'adresse du vecteur lui ressemblant le plus.
- un décodeur qui reçoit l'adresse et génère le vecteur $\hat{X}_i$ correspondant du dictionnaire. Ce vecteur est une approximation du vecteur X_i à coder.

Calcul du taux de compression

Le débit est donné par $R = \frac{\log_2 N_c}{k}$ bits par pixel. Il est lié à la dimension k des vecteurs à coder de la séquence d'apprentissage et à la taille du dictionnaire.

> EXEMPLE: Pour un débit fixé à un bit par pixel et des vecteurs composés de blocs d'image de taille 2×2, le dictionnaire doit contenir 16 vecteurs (16 vecteurs représentant 256^4 vecteurs possibles). Si la taille des vecteurs de l'image à coder est de 4×4, le dictionnaire contient 2^{16} vecteurs (représentant 256^{16} vecteurs possibles).

Plusieurs questions se posent lors de l'élaboration d'un quantificateur vectoriel :

- Le choix de l'entité vectorielle : un vecteur peut être formé des niveaux de gris d'un ensemble de pixels voisins (QV spatiale), d'une suite de coefficients issus de la transformation d'un bloc d'image (QV à transformée), des 3 composantes couleurs d'un pixel, ou autres.
- Le choix d'une mesure de distorsion permettant la construction du dictionnaire ainsi que la mesure de ressemblance entre vecteurs.
- La génération du dictionnaire optimal en partant d'un dictionnaire initial à construire.
- L'organisation du dictionnaire, facilitant la recherche des vecteurs lors de la quantification, ou plus généralement, la structure du quantificateur vectoriel.

Mesure de la distorsion

Différentes mesures de distorsion ont été proposées dans la littérature. La mesure idéale doit évaluer la qualité subjective de l'image reconstituée.

Faute de modèle fiable de qualité subjective, l'erreur quadratique moyenne est la mesure la plus souvent utilisée en raison de sa simplicité. Elle s'exprime par :

$$d(X,Y) = \sum_{i=1}^{k} |x_i - y_i|^2.$$

Une mesure plus générale est l'erreur quadratique moyenne pondérée, donnée par : $d(X,Y) = (X-Y)P(X-Y)^t = \sum_{i=1}^{k}\sum_{j=1}^{k} P_{i,j}(x_i - y_i)(x_j - y_j)$, où P est une matrice carrée définie positive, symétrique.

Génération du dictionnaire

Le but est de construire un dictionnaire optimal composé de N_c vecteurs, minimisant une distorsion moyenne donnée par :

$$D(X, Q(X)) = \frac{1}{N_c} \sum_{i=1}^{N_c} d(X_i, \hat{X}_i).$$

Un algorithme de classification connu sous le nom de LBG, présenté en 1980 par Linde, Buzo et Gray [106], retourne un dictionnaire localement optimal. Le dictionnaire peut également être obtenu à l'aide de cartes de Kohonen [93], ou de techniques d'optimisation stochastique telles que le recuit simmulé, ce dernier permettant d'atteindre le minimum global au prix d'un accroissement du coût de calculs [68].

Algorithme de Linde-Buzo-Gray

Cet algorithme connu sous le nom de LBG est une généralisation de la méthode non variationnelle proposée par Lloyd [107] en 1960, pour la quantification scalaire. Il a pour but de générer une partition sur une image (séquence d'apprentissage), partant d'un dictionnaire initial composé de vecteurs les plus "éloignés" possible. Ces vecteurs doivent être représentatifs des vecteurs rencontrés parmi les images à coder. L'algorithme itératif converge vers un dictionnaire localement optimal.

Un quantificateur vectoriel $\breve{Q}$ est dit (globalement) optimal s'il minimise une distorsion D donnée. $\breve{Q}$ est optimal si pour tout autre quantificateur Q composé comme $\breve{Q}$ d'un dictionnaire à N_c vecteurs, on a $D(\breve{Q}) \leq D(Q)$.
Q est localement optimal si $D(Q)$ est un minimum local : une faible perturbation sur Q augmente la distorsion $D(Q)$.

Détail de l'algorithme LBG

1. On se donne un dictionnaire initial C^0 composé de N_c vecteurs $\hat{X}_i (i = 1 \ldots N_c)$, une mesure de distorsion d, un seuil $\epsilon \geq 0$, un compteur d'itération $l = 0$, une distorsion moyenne D^{l-1} initialisée avec une très grande valeur et une séquence d'apprentissage composée de n vecteurs $(X_j, j = 1 \ldots n)$.

2. Partant du dictionnaire $C^l = (\hat{X}_i, i = 1 \ldots N_c)$, trouver la partition $S^l = (S_i, i = 1 \ldots N_c)$ de la séquence d'apprentissage minimisant la distorsion :

$X_j \in S_i$ si $d(X_j, \hat{X}_i) \leq d(X_j, \hat{X}_m), \forall m$. Autrement dit :
- Pour tous les vecteurs X_j de la séquence d'apprentissage ($j = 1 \ldots n$)
- Pour tous les vecteurs $\hat{X}_i$ du dictionnaire ($i = 1 \ldots N_c$)
- Si $d(X_j, \hat{X}_i) \leq d(X_j, \hat{X}_l), \forall l$ alors $X_j \in S_i$.

La recherche de la distorsion minimale définit une région de décision S_i pour chaque vecteur $\hat{X}_i$ du dictionnaire. Chaque vecteur X_j de la séquence d'apprentissage inclus dans la région de décision S_i est approximé par le vecteur $\hat{X}_i$ associé.

3. Calculer la distorsion moyenne: $D^l = D[C^l, S^l] = \frac{1}{n} \sum_{j=1}^{n} \min_{\hat{X} \in C^l} d(X_j, \hat{X})$.
   Cette expression permet de calculer la moyenne des distorsions minimales entre les n vecteurs X_j de la séquence d'apprentissage et les vecteurs d'approximation $\hat{X}$ correspondant (à ce niveau d'itération de l'algorithme).

4. Si $\frac{D^{l-1} - D^l}{D^l} \leq (\epsilon = 0.001)$, le dictionnaire C^l est conservé. La procédure s'arrête. Sinon, continuer.

5. Rechercher l'ensemble optimal des vecteurs d'approximation $\hat{X}(S^l) = \hat{X}(S_i)$, $(i = 1 \ldots N_c)$.
   $\hat{X}(S_i) = \frac{1}{||S_i||} \sum_{j : X_j \in S_i} X_j$ est le barycentre (centroïde) de la partition S_i: Le nouveau vecteur d'approximation de chaque partition est le barycentre de tous les vecteurs de la séquence d'apprentissage appartenant à la partition. $||S_i||$ est le nombre de vecteurs d'apprentissage X_j inclus dans la cellule S_i. L'utilisation de l'erreur quadratique moyenne (RMS) pour le calcul des vecteurs $\hat{X}(S_i)$ reviendrait à calculer la moyenne des vecteurs de la séquence d'apprentissage à l'intérieur des régions S_i.

6. Actualiser le dictionnaire: $C^{l+1} = \hat{X}(S^l)$, incrémenter l et aller à l'étape (2).

Choix du dictionnaire initial

Différentes solutions permettent la génération du dictionnaire initial nécessaire à l'algorithme LBG. Nous n'en citerons que deux :

- Méthodes aléatoires.
  Les N_c vecteurs sont regroupés aléatoirement pour former le dictionnaire initial. Une solution est de prendre N_c vecteurs dans la séquence d'apprentissage, ou dans différentes images. Les vecteurs doivent être le plus "éloignés" possible. Ce choix de vecteurs amène souvent l'algorithme dans un minimum local loin du minimum global.

- Méthode des divisions successives.
  Linde, Buzo et Gray ont proposé de partir de la séquence d'apprentissage. Le centroïde de celle-ci (moyenne de tous les vecteurs de la séquence) est calculé.

Le vecteur résultant est divisé en deux (la division est réalisée en ajoutant une quantité $+\zeta$ et $-\zeta$ au vecteur). L'algorithme LBG est ensuite exécuté pour retourner un dictionnaire optimal de taille 2. Le processus est itéré pour retourner un dictionnaire optimal de taille $2^2, 2^3, \ldots$, et enfin N_c.

Organisation du dictionnaire

L'organisation du dictionnaire est très importante car elle facilite la mise en correspondance du vecteur à quantifier avec son approximation dans le dictionnaire. Le critère utilisé pour la mise en correspondance est la distance quadratique (MSE).

Quantification à recherche exhaustive.

Pour chaque vecteur d'entrée, le vecteur d'approximation est recherché dans tout le dictionnaire. Le coût de calcul évolue en $O(n.N_c)$ où n est le nombre de vecteurs à quantifier et N_c le nombre de vecteurs dans le dictionnaire.

Quantification à recherche arborescente.

Buzo et *al.* [28] ont proposé une recherche dans un arbre (TSVQ = Tree Search VQ), dans le but d'accélérer la quantification. Le dictionnaire est construit en appliquant successivement l'algorithme LBG sur des sous-espaces indépendants de taille décroissante. Si l'arbre est de dimension 2, la séquence à quantifier est au départ divisée en deux (premier niveau de l'arbre binaire). Le deuxième niveau de l'arbre est ensuite composé de 4 nœuds. Le processus s'arrête lorsque le nombre de vecteurs, ou de sous-espaces est suffisant.

EXEMPLE: Considérons un arbre binaire contenant 8 vecteurs d'approximation. La taille de l'arbre est dans ce cas de 14 car il faut mémoriser les vecteurs intermédiaires. La recherche d'un vecteur particulier nécessite 6 calculs de distance au lieu de 8.

Avantage de la TSVQ : la recherche dans le dictionnaire est accélérée car une partie seulement de l'arbre est parcourue.
Inconvénients :

- La taille du dictionnaire est plus importante
- Au cours de la quantification, une fois qu'une branche est sélectionnée, le vecteur recherché ne pourra plus être comparé avec les vecteurs d'une branche voisine.

Quantification à codes produits

Lorsqu'un vecteur peut être décomposé en sous-vecteurs ayant des propriétés différentes et indépendantes, plusieurs quantificateurs peuvent être développés pour coder chacun des sous-vecteurs [143]. Ce type de quantification est en général sous

optimal lorsqu'on la compare à la quantification vectorielle à recherche exhaustive basée sur un seul dictionnaire de même taille effective.
Taille effective: Considérons deux dictionnaires de tailles N et M, permettant la quantification de l'orientation et de l'amplitude de vecteurs. La taille effective du dictionnaire global est $N_C = N \times M$ (N orientations différentes pour chaque amplitude) alors que l'on ne mémorise qu'un dictionnaire de taille $N + M$. L'avantage de cette méthode est sa robustesse vis à vis du nombre important d'images différentes à quantifier.

Quantification vectorielle spatiale

La quantification vectorielle spatiale [68] opère directement sur l'image. Celle-ci est partitionnée en blocs de pixels formant des vecteurs de niveaux de gris. Différentes approches ont été proposées dans le but de faciliter la recherche des vecteurs dans le dictionnaire, et d'améliorer la qualité de la quantification.

Quantification vectorielle à moyenne et/ou gain séparés

Murakami et *al.* [121] ont proposé un quantificateur vectoriel codant des vecteurs (blocs d'image) à moyenne nulle et variance unitaire. La moyenne et la variance sont quantifiées de façon scalaire, ou par un quantificateur vectoriel de dimension 2.
Des variantes de cette méthode sont la quantification vectorielle à moyenne séparée (M/SVQ = Mean/Shape VQ) [5] et la quantification vectorielle résiduelle à moyenne séparée (M/RVQ) [4]. Dans la première, la moyenne est quantifiée scalairement et le vecteur à moyenne nulle est quantifié vectoriellement. La deuxième quantifie de façon vectorielle le vecteur d'entrée auquel on soustrait au préalable la moyenne du signal d'entrée, quantifiée de façon scalaire. De cette manière, les erreurs de quantification scalaire et les "effets de blocs" trop visibles sont réduits.

Quantification vectorielle à classification

Une classification est réalisée sur chaque vecteur image de manière à utiliser différents dictionnaires pour chaque classe (textures, frontières, homogènes ...) [136] [135]. Cette méthode permet de quantifier plus efficacement les vecteurs incluant les frontières de l'image.

Beaucoup d'autres approches ont été présentées dans la littérature. Nous citerons la quantification vectorielle hiérarchique, la quantification vectorielle à dictionnaire rafraîchi, la quantification vectorielle prédictive [123].

Quantification vectorielle à transformées

L'association de la quantification vectorielle avec une technique de compression indirecte (opérant dans un espace transformé) permet des taux de compression importants. Différentes études ont montré qu'il était judicieux de quantifier de manière vectorielle les coefficients issus d'une représentation multirésolution de l'image [2]

ou d'une transformation en cosinus discrète [57].

2.7.5 Quantification vectorielle algébrique

La quantification vectorielle algébrique (QVA) ou "Lattice Vector Quantization" (LVQ) est une méthode permettant de réduire sensiblement le coût des calculs par rapport aux quantificateurs utilisant des méthodes d'apprentissage. Ceci est dû au fait que le dictionnaire est défini *a priori* comme un ensemble de vecteurs régulièrement répartis dans l'espace. Les vecteurs appartiennent à un réseau régulier infini de points. La définition du dictionnaire consiste à ne conserver qu'un nombre fini de points dans le réseau, à l'intérieur d'une pyramide multidimensionnelle ou d'une hypersphère, centrée sur l'origine. De plus, la structure régulière facilite la définition d'algorithmes simples et rapides de quantification au sein du réseau [7] [138].

Réseaux réguliers de points

Un réseau régulier Λ de points dans $\mathbb{R}^n$ est défini par [36] :

$$\lambda_n = \{Y \in \mathbb{R}^n / \exists (u_1, \ldots, u_n)^t \in \mathbb{Z}^n, X = \sum_{i=1}^{n} u_i \mathbf{a}_i^t\}$$

où les vecteurs linéairement indépendants $\mathbf{a}_i$ $(i = 1 \ldots n)$ appartiennent à l'espace $\mathbb{R}^m$ avec $m \geq n$. A chacun des points Y du réseau est associée une région de Voronoï, comprenant l'ensemble des points les plus proches de $Y \in \Lambda_n$ que de n'importe quel autre point du réseau. Parmi les réseaux réguliers couramment utilisés, on peut citer :

Le réseau $\mathbb{Z}^n$
Ce réseau regroupe l'ensemble des points de $\mathbb{R}^n$ à coordonnées entières.

$$\mathbb{Z}^n = \{Y = (y_1, \ldots, y_n) \in \mathbb{R}^n / y_i \in \mathbb{Z}\}$$

Si $n = 2$, les régions de Voronoï sont des carrés.

Le réseau $\mathbb{D}_n$
Ce réseau regroupe l'ensemble des points de $\mathbb{Z}^n$ dont la somme des coordonnées est paire.

$$\mathbb{D}_n = \{Y = (y_1, \ldots, y_n) \in \mathbb{Z}^n / \sum_{i=1}^{n} y_i = 0 \bmod(2)\}$$

Si $n = 2$, les régions de Voronoï sont des losanges. Ce réseau est utile pour la définition des réseaux réguliers E_n et Λ_{16} (voir [36]).

Quantification rapide

La quantification consiste à associer un vecteur d'entrée X à l'un des points du réseau régulier. Par exemple, dans le cas simple du réseau $\mathbb{Z}^n$ le point Y du réseau

représentant un vecteur d'entrée X réel est :

$$Y = f(x) = [f(x_1), \ldots, f(x_n)], \quad X \in \mathbb{R}^n \quad \text{et} \quad Y \in \mathbb{Z}^n$$

où $f(x)$ est l'entier le plus proche de x.
D'autres algorithmes de quantification rapide dans les réseaux $\mathbb{D}_n$, E_n et Λ_{16} ont été développés par Conway et Sloane [36].

Conception du dictionnaire

La conception du dictionnaire consiste à tronquer le réseau régulier, de manière à ne conserver qu'un nombre fini de points, inclus dans une hypersurface. La forme de l'enveloppe de l'hypersurface dépend directement de la statistique de la source à coder. Dans [7], les auteurs considèrent des lois Gaussiennes ou Laplaciennes qui approximent bien les statistiques des sous-images de coefficients d'ondelettes ou des images d'erreur de prédiction dans les algorithmes de quantification.

L'enveloppe de l'hypersurface est telle que les vecteurs du réseau ont même probabilité. Pour une loi gaussienne multidimensionnelle donnée par

$$f_x(x) = \frac{1}{\left(\sigma\sqrt{2\pi}\right)^n} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n} x_i^2\right),$$

les points X qui ont la même probabilité d'apparition définissent une surface (hypersphère) d'énergie constante m^2 au sens de la norme L_2 :

$$\sum_{i=1}^{n} x_i^2 = \|X\|_2^2 = m^2$$

Pour une loi Laplacienne multidimensionnelle donnée par

$$f_x(x) = \frac{1}{\left(\sigma\sqrt{2}\right)^n} \exp\left(-\frac{\sqrt{2}}{\sigma}\sum_{i=1}^{n} |x_i|\right),$$

les points X équiprobables définissent une surface (pyramide) d'énergie constante m au sens de la norme L_1 :

$$\sum_{i=1}^{n} |x_i| = \|X\|_1 = m$$

Dans chacun des cas, la troncature du réseau se fait sur l'hypersphère choisie, en fonction de la statistique de la source, et de l'énergie de troncature.

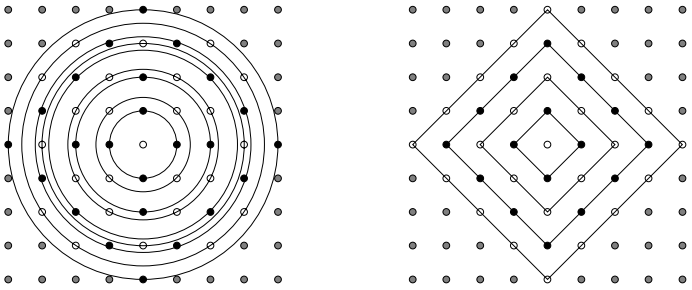


FIG. 7 - Troncature du réseau $\mathbb{Z}^2$ par une pyramide (à droite) et une sphère (à gauche) de rayon 4 et dénombrement des points du dictionnaire (points blanc et points noirs)

La phase suivante consiste à dénombrer les points inclus dans le dictionnaire. Ces points sont répartis sur des surfaces imbriquées à la facon des poupées Russes. La taille du dictionnaire est donc calculée en comptant le nombre de points sur chacune des surfaces puis en en faisant la somme. L'approche classique est d'utiliser les séries génératrices *théta* [36] pour les sphères et les séries *Nu* pour les pyramides.

Principe de quantification d'une source

Un algorithme global de quantification d'une source vectorielle est donné dans [7]. Dans le cas idéal où le dictionnaire contient tous les vecteurs de la source, un vecteur réel X est codé par l'index du point Y le plus proche dans le dictionnaire. Dans le cas général le dictionnaire ne contient pas tous les points de la source. Une opération de normalisation de la source par un facteur d'échelle r est donc nécessaire pour l'adapter au dictionnaire. De plus, différentes techniques d'indexage des vecteurs dans le réseau dilaté ont été étudiées [36] [7], car contrairement au cas des codeurs vectoriels usuels, l'index d'un vecteur dans le dictionnaire n'est pas une adresse dans un tableau.

2.7.6 Codage par transformée

Les méthodes de codage qui invoquent des transformations d'images sont causales, contrairement au codage prédictif qui considère les échantillons précédemment émis. Le codage par transformée tel que nous le verrons dans la suite de cette section opère sur les blocs d'une image. Une transformation unitaire a pour but de décorréler les "pixels image" (données d'entrée) de manière à compacter l'énergie de l'image transformée sur un faible nombre de composantes (coefficients).

Transformation bidimensionnelle unitaire séparable
Nous donnons ci-après les expressions d'une transformation bidimensionnelle directe (3) et inverse (4). L'image originale F est composée de pixels notés $f(i,j)$. Les éléments $g(u,v)$ sont les coefficients de la transformation et l'ensemble $\{g(u,v)\}$

constitue l'image transformée G.

$$g(u,v) = \sum_{i,j=0}^{N-1} a_u(i).f(i,j).a_v(j) \longleftrightarrow G = AFA^T \tag{3}$$

$$f(i,j) = \sum_{u,v=0}^{N-1} a_u^*(i).g(u,v).a_v^*(j) \longleftrightarrow F = A^{*T}GA^* \tag{4}$$

Les ensembles $\{a_u(i), u = 0 \ldots N-1\}$ et $\{a_v(j), v = 0 \ldots N-1\}$ constituent deux bases complètes et orthogonales de vecteurs monodimensionnels.

Images de base
Soit a_u^* le u^{ieme} vecteur de A^{*T}. La matrice $A_{u,v}^*$ est définie par $A_{u,v}^* = a_u^* a_v^{*T}$. L'image F peut s'écrire sous la forme $F = \sum_{u,v=0}^{N-1} g(u,v).A_{u,v}^*$
et par conséquent $g(u,v) = < F, A_{u,v}^* >$.

L'image transformée G s'exprime par une combinaison linéaire de matrices carrées $(N \times N)$, formant une "base d'images". Le coefficient $g(u,v)$ est issu du produit scalaire entre l'image (u,v) de la base et l'image F à transformer: $g(u,v)$ est la projection de l'image F sur l'image (u,v) de la base.

Propriétés de la transformation unitaire

- Conservation de l'énergie: $\sum_{i,j=0}^{N-1} |f(i,j)|^2 = \sum_{i,j=0}^{N-1} |g(u,v)|^2$.
  Une transformation unitaire peut être vue comme la rotation d'un vecteur F dans un espace vectoriel de dimension N (la transformation exécute une rotation des vecteurs de base et les coefficients $g(u,v)$ sont les projections de F sur cette nouvelle base).

- Concentration de l'énergie.
  L'énergie moyenne $E[|g(u,v)|^2]$ est inégalement distribuée. Une grande partie est concentrée sur un faible nombre de coefficients.

- Décorrélation.
  Lorsque les pixels de l'image d'entrée F sont fortement corrélés, les coefficients transformés tendent à être décorrélés. Les éléments en dehors de la diagonale de la matrice de covariance $(R_g = E[(G-\mu_g)(G-\mu_g)^{*T}])$ tendent à devenir faibles par rapport aux éléments de la diagonale.
  Remarque: Une matrice de covariance R_g dont seuls les éléments sur la diagonale sont différents de zéro est la matrice d'une transformation optimale, en terme de décorrélation des coefficients.

Remarque : la transformation unitaire décorrèle les pixels voisins, compacte l'énergie sur un plus petit nombre de composantes mais ne fait pas de compression. La compression est introduite dans la phase de quantification des coefficients.

Étude de différentes transformations d'images

1 - Transformation de Karhunen Loeve (KLT)

La KLT est optimale en terme de compactage de l'énergie. Les principaux inconvénients de cette transformation sont dus au fait que les images de base sont dépendantes de l'image à transformer.

Rappel: Les composantes $y(l)$ et $y(k)$ sont orthogonales si $E[y(l)y(k)] = 0, \ \forall k \neq l$.
Elles sont décorrélées si $E[y(l)y(k)] = E[y(l)]E[y(k)] \ \forall k \neq l$.
De plus, si $y(l)$ et $y(k)$ sont décorrélées et à valeur moyenne nulle, elles sont orthogonales.

Transformée de Karhunen Loeve monodimensionnelle:

Soit deux vecteurs aléatoires $Y = \{y(0), \ldots, y(N)\}$ et $X = \{x(0), \ldots, x(N)\}$ dont les composantes sont des variables aléatoires. Les composantes du vecteur d'entrée X sont en général corrélées entre elles. Le but de la KLT est de construire le vecteur de sortie $Y = U^T X$ tel que ses composantes soient orthogonales. La transformée inverse est égale à $X = UY$.

Soit $E[XX^T] = R_x$ la matrice d'autocorrélation à $N \times N$ éléments, engendrée par le vecteur X. La matrice de Karhunen Loeve réduisant R_x à sa forme diagonale ∇ s'exprime par $U = [u_1 \ldots u_k]$. Les vecteurs u_i sont les vecteurs propres normalisés de la matrice symétrique R_x, rangés dans un ordre décroissant ($\lambda_1 \geq \lambda_2 \ldots \geq \lambda_N \geq 0$), et λ_i sont les valeurs propres associées. La forme diagonale de X est donnée par:

$$\nabla = E[YY^T] = E[U^T X X^T U] = U^T E[XX^T]U = U^T R_x U = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_N \end{bmatrix}$$

avec λ_i égales aux variances des composantes transformées.
Les composantes du vecteur transformé Y sont orthogonales car la matrice $E[YY^T]$ se réduit à une matrice diagonale d'éléments λ_i ($E[y(k)y(l)] = \lambda_k \delta(k-l)$).
Si la matrice R_x était la matrice de covariance de X, alors les composantes de Y seraient décorrélées.

2 - Transformation de Fourier Discrète: DFT

La DFT décompose l'image en ses composantes spectrales: $f(i,j) \longleftrightarrow f(u,v)$, où (u,v) sont les coordonnées de l'image dans le domaine des fréquences spatiales. La transformation bidimensionnelle est donnée par [88]:

$$g(u,v) = \sum_{i,j=0}^{N-1} f(i,j).a_u(i).a_v(j), \qquad 0 \leq u,v \leq N-1$$

où

$$a_u(i).a_v(j) = \exp\left[-\imath 2\pi\left(\frac{ui}{N} + \frac{vj}{N}\right)\right]$$

Avantages :

- Le noyau 2D de la transformation est séparable en deux transformations 1D
- Un calcul de transformation rapide existe (FFT). Le coût de calcul est en $O(N \log_2 N)$ pour une image de taille $N \times N$.

Inconvénients :

- Les coefficients générés par la DFT sont complexes, au nombre de $2N^2$. A cause des propriétés de symétrie, près de la moitiée des coefficients sont redondants.
- Des composantes spectrales de repliement apparaissent, dues à la périodicité implicite de l'image : à bas débits, les effets de bloc sont très visibles.

3 - Transformation en Cosinus Discrète : DCT

Lorsque les images présentent une forte corrélation entre pixels voisins, les images de base de la DCT tendent à être égales à celles de la transformation de Karhunen Loeve (ces dernières dépendant de l'image à traiter) [88].
La périodicité implicite de la DCT, pour transformer une séquence 1D de longueur n, est de $2n$. La séquence est étendue sur $2n$ points par réflexion par rapport à l'axe vertical d'origine. Il n'y a pas dans ce cas de discontinuités de frontières, contrairement à la DFT qui a une périodocité implicite, pour la même séquence, de longueur N. Lorsqu'un bloc d'image est transformé par DFT, les erreurs apparaissent aux bords du bloc, et sont beaucoup moins visibles lorsque le bloc est transformé par DCT. Pour cette raison, et puisque les calculs des coefficients de la DCT se font dans le domaine réel, la transformation DCT est très utilisée en codage d'images.

2.7.7 Codage hiérarchique

Cette classe de méthodes de codage permet la reconstruction progressive, ou l'accès aux images de qualité et de résolution différentes. L'exemple souvent cité est celui de la recherche d'images compressées dans une base de données. Le codage hiérarchique est aussi utile lorsque l'image compressée est utilisée par différents équipements de visualisation ou de transmission : écran de télévision haute définition (HDTV) avec incrustation d'écran de moindre qualité, image sortie sur une imprimante de très haute résolution, ou transmise sur une ligne téléphonique à basse résolution, etc ...

Deux types de hiérarchies sont à distinguer :

- La hiérarchie à résolution fixe : l'image reconstruite est de même taille que l'image originale mais chaque pixel est affiné au fur et à mesure du décodage.

Nous citerons le codage par plan de bits, la quantification vectorielle à recherche arborescente, le codage par transformées (codage zig-zag dans la norme JPEG).

- La hiérarchie à résolution variable : la résolution spatiale de l'image reconstruite évolue. Les méthodes de compression sont dans ce cas basées sur une structure pyramidale régulière ou irrégulière. La pyramide la plus simple est constituée d'une suite d'images de tailles décroissantes jusqu'à une taille égale à celle du pixel (sommet de la pyramide = apex). Le passage du niveau 0 (image originale) au niveau 1 se fait par simple moyennage des blocs de 4 pixels en blocs de 1 pixel. Nous citerons dans cette classe les pyramides obtenues par sous-échantillonnage régulier ou irrégulier, les pyramides obtenues par moyennage, les pyramides Gaussiennes et Laplaciennes.

Codage par pyramide Laplacienne

Nous présentons dans cette section l'algorithme de Burt et Adelson [27] permettant de calculer la pyramide Laplacienne et montrons l'intérêt d'une telle représentation pour la compression des images.

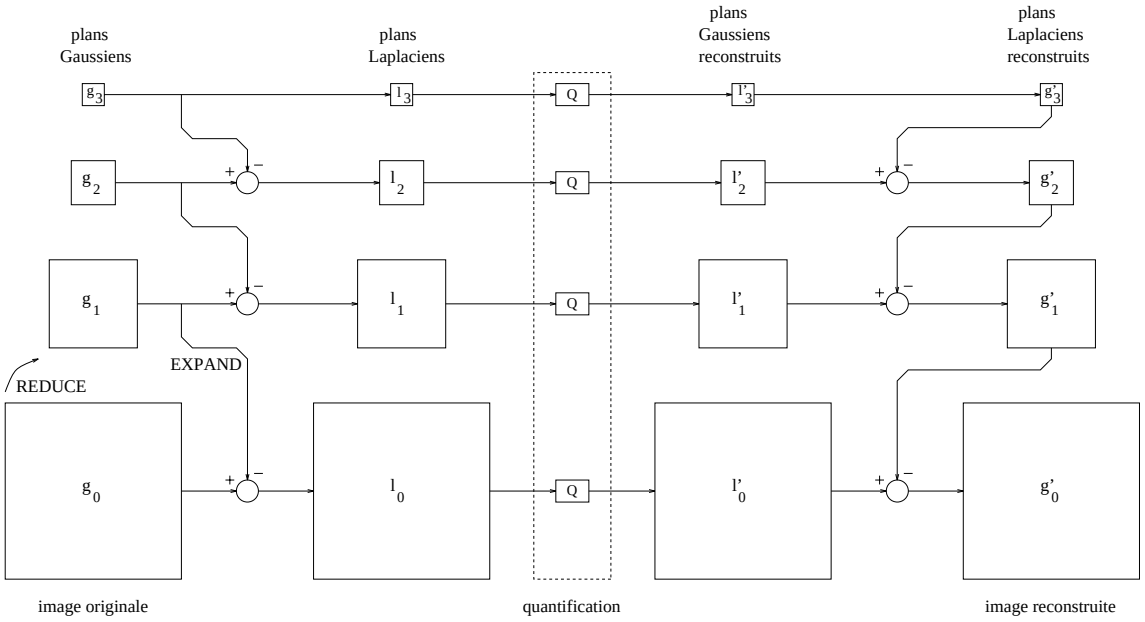


FIG. 8 - Illustration de l'algorithme de Burt et Adelson.

L'image originale $g_0(x, y)$ est dans un premier temps filtrée passe-bas pour retourner une image de prédiction $g_1(x, y)$. L'image $g_1(x, y)$ est calculée en moyennant localement les pixels à l'aide d'un filtre à réponse impulsionnelle proche d'une courbe Gaussienne et peut donc être sous-échantillonnée en vertu du théorème de Shannon (le filtre utilisé par Burt et Adelson permet un sous-échantillonnage d'un facteur 2:1). La soustraction des deux images précédantes avant sous-échantillonnage de $g_1(x, y)$

retourne une image d'erreur de prédiction $l_0(x,y)$ d'entropie réduite, s'exprimant de la manière suivante :

$$l_0(x,y) = g_0(x,y) - g_1(x,y).$$

La procédure ainsi décrite est itérée n fois pour retourner deux ensembles $g_i(x,y)$ $i = 0 \ldots n$ et $l_i(x,y)$ $i = 0 \ldots n$ constitués d'images sous-échantillonnées à chaque niveau d'un facteur 2:1 et donc de taille de plus en plus petite. Les deux ensembles forment respectivement une *pyramide Gaussienne* et une *pyramide Laplacienne*. Les images $l_n(x,y)$ et $g_n(x,y)$ au sommet des deux pyramides sont égales et $g_n(x,y)$ est issue de n filtrages passe-bas successifs de l'image originale (voir figure 8).

Un algorithme rapide a été proposé par Burt et Adelson [27] en 1983 pour calculer la pyramide Laplacienne. Les images $g_1(x,y)$ et $l_0(x,y)$ sont obtenues par :

$$g_1(x,y) = \text{REDUCE}(g_0(x,y)) \qquad \text{et}$$

$$l_0(x,y) = g_0(x,y) - \text{EXPAND}(g_1(x,y))$$

où la fonction REDUCE comprend un noyau gaussien 5×5 qui n'est recalculé qu'un échantillon sur deux. Elle retourne une image de taille deux fois plus petite, filtrée passe bas. La fonction EXPAND est l'inverse de la fonction REDUCE. Elle réalise une interpolation de manière à compenser la décimation causée par la fonction REDUCE. La rapiditité en termes de coûts de calculs de cet algorithme vient du fait que les fonctions REDUCE et EXPAND opèrent sur des images de taille de plus en plus petite au fur et à mesure de la construction de la pyramide.
De plus, le noyau gaussien peut être utilisé pour déterminer la fonction de pondération équivalente $h_i(x,y)$ à chaque niveau de la pyramide [27]. $h_i(x,y)$ permet par convolution avec l'image originale d'obtenir directement l'image du niveau i de la pyramide Gaussienne. La différence entre les deux convolutions de $h_i(x,y)$ et $h_{i+1}(x,y)$ avec l'image originale retourne directement l'image i de la pyramide Laplacienne.

L'image originale $g_0(x,y)$ est complètement codée par la pyramide Laplacienne. En n'utilisant pas de quantificateur, l'image originale est reconstruite de manière exacte en partant de l'image basse résolution $g_n(x,y)$. En redescendant dans la pyramide, les images $g_i(x,y)$ pour i allant de $n-1$ à 0 sont données par (figure 8) :

$$g_i(x,y) = l_i(x,y) + \text{EXPAND}(g_{i+1}(x,y)).$$

La compression dans ce schéma vient du fait que l'entropie et la variance des images d'erreur de prédiction $l_i(x,y)$ (du type passe-haut) est réduite (l'histogramme de ces images est très concentré sur zéro). Il est donc possible de les coder efficacement à l'aide de codes à longueur variable. L'image basse résolution $g_n(x,y)$ est elle directement quantifiée puisqu'elle est de taille très inférieure à celle de $g_0(x,y)$.

2.7.8 Codage sous-bandes

Le codage sous-bandes a été initialement proposé pour la compression du signal de parole [38] en 1976. Il consiste à décomposer le signal ou l'image en différentes bandes

de fréquences spatiales et à coder chacune d'entre elles de manière indépendante ou non. Dans les schémas classiques, la bande spectrale du signal original est d'abord divisée en deux parties à l'aide de deux filtres numériques (passe-bas et passe-haut). La procédure est ensuite répétée dans chacune des sous-bandes fréquentielles. Le résultat constitue une structure arborescente, symétrique ou non, composée d'un nombre entier de sous-bandes. Le signal est reconstruit par recombinaison des sous-bandes à l'aide de filtres d'interpolation. La figure 9 illustre une décomposition en quatre sous-bandes d'une image. L'image *centre BF* est une sous-image de basse résolution. Les quatre sous-images *centre BF*, *horizontal HF*, *vertical HF* et *diagonal HF* correspondent à quatre sous-bandes fréquentielles directionnelles.

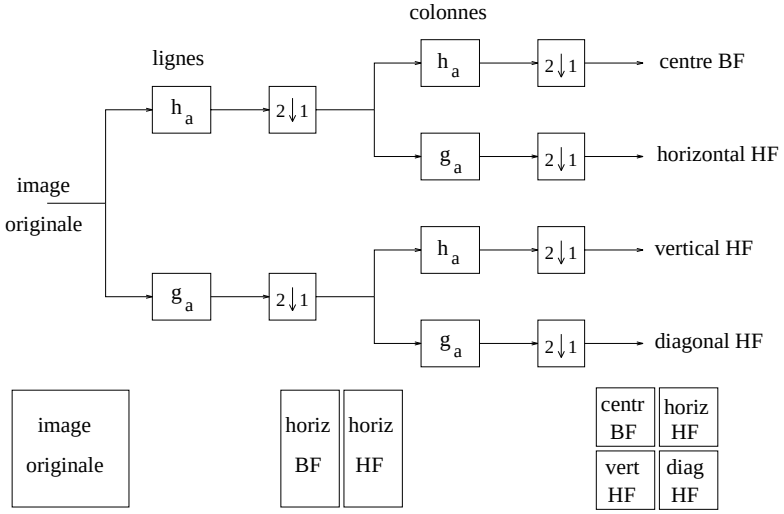


FIG. 9 - Décomposition d'une image en sous-bandes.

Analyse-synthèse

Soit un signal d'entrée unidimensionnel x_n, dont la transformée en z est notée $S(z)$. Le signal est décomposé en deux signaux à largeur de bande spectrale limitée, à l'aide d'un filtre passe-bas $H_a(z)$ et d'un filtre passe-haut $G_a(z)$. Les signaux sont ensuite sous-échantillonnés d'un facteur 2 puis codés. Le sous-échantillonnage permet de conserver le même nombre total d'échantillons que dans l'image originale. La reconstruction se fait par sur-échantillonnage d'un facteur 2 en intercalant dans chacune des sous-bandes des zéros entre les échantillons. Les signaux sont ensuite filtrés pour interpolation, par respectivement un filtre passe-bas $H_r(z)$ et un filtre passe-haut $G_r(z)$, puis recombinés pour retourner le signal reconstruit $\hat{x}_n$.

La transformée en z de $\hat{x}_n$ est donnée par :

$$\hat{S}(z) = \frac{1}{2}[H_r(z).H_a(z) + G_r(z).G_a(z)]S(z) + \frac{1}{2}[H_r(z).H_a(-z) + G_r(z).G_a(-z)]S(-z)$$

Le premier terme représente une version pondérée du signal original, alors que le deuxième est dû à un repliement spectral.

La reconstruction est exacte si $\hat{S}(z) = S(z)$ et donc si

$$H_r(z).H_a(z) + G_r(z).G_a(z) = 2$$

$$H_r(z).H_a(-z) + G_r(z).G_a(-z) = 0$$

Plusieurs solutions ont été proposées pour résoudre ce système de 2 équations à 4 inconnues. Dans ce cadre, on peut citer les Filtres Miroirs en Quadrature (QMF), les Filtres Conjugués en Quadrature (CQF), et les Filtres à Noyau Court.

Les filtres QMF proposés par Esteban et Galand [55] sont composés d'une paire de filtres à gains symétriques par rapport au quart de la fréquence de Shannon-Nyquist (effet miroir) et à phases décalées de $\frac{\pi}{2}$ (quadrature). Ils sont synthétisés par des filtres à réponse impulsionnelle à durée limitée (FIR), à phase linéaire, avec un nombre pair de coefficients. Ils doivent respecter les conditions suivantes :

$$\begin{aligned} G_a(Z) &= H_a(-z) \\ H_r(z) &= H_a(z) \\ G_r(z) &= -G_a(z) \end{aligned}$$

La transformée en z de $\hat{S}(z)$ est donnée par

$$\hat{S}(z) = [H_a^2(z) - G_a^2(z)]S(z).$$

Le terme entre crochets n'introduit pas de distorsion de phase mais son module n'est pas facile à rendre égal à l'unité. Le filtre satisfaisant cette dernière condition est de longueur 2. En pratique, on tolère une certaine distorsion d'amplitude en utilisant des filtres assez longs pour avoir une meilleure sélectivité fréquentielle.

Une solution visant à éliminer les 3 types de distorsion (repliement de spectre, distorsion de phase et d'amplitude) est proposée par Smith et Barnwell [154] par l'utilisation de filtres CQF, autorisant un retard R sur le signal reconstruit : $\hat{s}_{n+R} = s_n$. Les filtres doivent vérifier (avec R égal à l'ordre des filtres) :

$$\begin{aligned} H_r(Z) &= G_a(-z) \\ G_r(z) &= -H_a(-z) \\ G_a(z) &= -H_a(-z^{-1})z^{-R} \end{aligned}$$

Dans ce cas, la transformée en z de $\hat{S}(z)$ est donnée par :

$$\hat{S}(z) = [H_a(z).H_a(z^{-1}) - H_a(-z).H_a(-z^{-1})]z^{-R}S(z).$$

Le filtre $H_a(z)$ est approximativement à phase linéaire. L'inconvénient est que les filtres générés sont asymétriques.

Le Gall et Tabatabai ont proposé [67] des filtres à noyau court, à phase linéaire,

et permettant une reconstruction exacte du signal original. L'avantage est le faible coût de calcul dû à la courte longueur des filtres. L'inconvénient est la faible capacité de séparation des bandes fréquentielles des filtres.

Nussbaumer et Vetterli [127] ont quant à eux proposé l'utilisation de filtres Pseudo-QMF autorisant une décomposition non exacte mais directe en N sous-bandes par modulation d'un seul filtre passe-bas prototype. La décomposition ne s'effectue pas de manière itérative comme dans les schémas classiques, ce qui implique des largeurs de bandes constantes. L'avantage de la méthode est qu'elle permet une décomposition du signal en un nombre élevé de sous-bandes, contrairement aux méthodes itératives pour lesquelles les erreurs se propagent d'un étage à l'autre.

Des travaux permettant une décomposition en sous-bandes en une seule étape ont aussi été proposés par Diab dans [49]. Les auteurs décomposent le signal en B sous-bandes fréquentielles dans le domaine de la Transformée en Cosinus Discrète, à l'aide de filtres idéaux. Par transformée en Cosinus Discrète inverse, les B sous-images obtenues sont ensuite quantifiées et codées (dans le domaine spatial).

Codage-décodage

La quantification et le codage des sous-bandes sont en général précédés d'une phase d'allocation de bits. Le nombre de bits alloué à chacune des sous bandes est variable et est calculé de manière à réduire l'erreur quadratique moyenne de quantification. Cette phase peut elle-même être précédée d'une phase de pondération psychovisuelle de la variance des sous-bandes, basée sur la sensibilité de l'œil humain à des images de fréquences spatiales différentes [142] [89].
La quantification des sous-bandes peut être scalaire (uniforme ou optimale) ou bien vectorielle. Elle se fait de manière indépendante sur chacune des images des sous-bandes.
Le codage peut lui se faire en considérant plusieurs images du même type (orientations horizontales, verticales ou diagonales) dans différentes sous-bandes. Le schéma classique utilise un codeur de Huffman pour chaque sous-bande. Une étude détaillée sur l'utilisation de codeurs prédictifs différentiels (MICD) adaptés aux caractéristiques des images dans les différentes sous-bandes est donnée dans la référence [98].

2.7.9 Codage par ondelettes

S. Mallat [110] a proposé en 1989 un opérateur permettant :

- d'approximer une fonction à une résolution[6] 2^m;
- de retrouver la différence entre l'approximation à la résolution 2^{m-1} et la résolution 2^m, en décomposant la fonction sur une base d'ondelettes orthogonales.

6. la résolution de l'image originale est notée 2^0, et les résolutions inférieures sont notées 2^m avec $m > 0$.

L'ensemble de ces deux opérations donne lieu à une représentation multirésolution complète et orthogonale appelée représentation en ondelettes, que nous introduisons dans les sections suivantes.

Définition des ondelettes

Les ondelettes sont des fonctions générées à partir d'une fonction mère Ψ, par dilatations et translations. Dans le cas monodimensionnel, la fonction Ψ s'écrit :

$$\Psi_{a,b}(x) = \frac{1}{\sqrt{|a|}}\Psi(\frac{x-b}{a})$$

où l'indice a représente un facteur d'échelle et l'indice b est un facteur de translation. L'ondelette mère Ψ doit être de carré intégrable et doit satisfaire la condition $\int |\hat{\Psi}(\omega)|^2|\omega|^{-1}d\omega < \infty$.
L'idée de base de la transformée en ondelettes est de représenter l'information contenue dans une fonction $f(x)$ de carré intégrable à une échelle a et une position b. La fonction $f(x)$ est vue comme une superposition d'ondelettes.
Dans le cas discret, les coefficients a et b sont donnés par :

$$a = a_0^m \quad \text{et} \quad b = nb_0a_0^m \qquad \text{avec } m, n \in Z,\ a_0 > 1,\ \text{et } b_0 > 0.$$

De plus, dans le cas où $a_0 = 2$ et $b_0 = 1$, il est montré [114] qu'il existe des fonctions Ψ telles que l'ensemble $\{\Psi_{m,n}\}_{m,n\in Z^2}$ constitue une base orthogonale de fonctions de carré intégrable. L'ondelette est alors définie par la relation suivante :

$$\Psi_{m,n}(x) = 2^{\frac{-m}{2}}\Psi(2^{-m}x - n)$$

et la décomposition en ondelettes d'une fonction $f(x)$ s'écrit :

$$f = \sum_{m,n} D_{m,n}(f)\ \Psi_{m,n}$$

où $D_{m,n}$ est appelé coefficient d'ondelette, représentant une mesure des variations locales du signal. Sa valeur est importante si l'ondelette Ψ à l'échelle m et à la position n est proche de la structure locale de f.
Les coefficients d'ondelette $D_{m,n}$ sont donnés par la relation suivante :

$$D_{m,n}(f) = \langle \Psi_{m,n}, f\rangle = \int \Psi_{m,n}(x)f(x)dx.$$

De nombreuses bases correspondant à diverses ondelettes ont été proposées [39] [114] [19]. La plus simple de celles-ci est la base de Haar pour laquelle la fonction $\Psi(x)$ vaut 1 sur l'intervalle $[0, \frac{1}{2}]$, -1 sur $[\frac{1}{2}, 1]$ et 0 ailleurs.

Analyse multirésolution

Le concept d'analyse multirésolution fait intervenir deux fonctions :

- l'ondelette mère Ψ et

- une fonction d'échelle ϕ définie par $\phi_{m,n}(x) = 2^{\frac{-m}{2}}\phi(2^{-m}x - n)$.

L'ensemble $\{\phi_{m,n}(x)\}_{n\in Z}$ constitue une base orthonormale de l'espace vectoriel V_m (V_m représente l'ensemble de toutes les approximations possibles à la résolution 2^m des fonctions dans $L^2(\mathbb{R})$).
L'approximation du signal à la résolution 2^m contient toute l'information nécessaire pour calculer le même signal à une résolution inférieure 2^{m+1}, $(\forall m \in Z, V_{m+1} \subset V_m)$.
Pour une résolution m fixée, l'ondelette $\Psi_{m,n}$ est définie dans un espace vectoriel O_m qui est le complément orthogonal de V_m dans V_{m-1} : O_m est orthogonal à V_m, $O_m \oplus V_m = V_{m-1}$.
Dans ce cas, les coefficients $\langle \Psi_{m,n}, f\rangle$ (projection orthogonale du signal original sur O_m) représentent l'information perdue en passant de l'approximation de f à la résolution 2^{m-1} à une résolution plus grossière 2^m.
Remarque : l'approximation à la résolution 2^m du signal f est égale à sa projection orthogonale $\langle f, \phi_{m,n}\rangle$ sur V_m.

Tout ce qui précède peut être transcrit dans un algorithme proposé par S. Mallat [110], basé sur des filtres à réponse impulsionnelle finie donnés par :

$$\begin{aligned} D_{m,n}(f) &= \sum_k g_{2n-k}\ A_{m-1,k}(f) = \langle \Psi_{m,n}, f\rangle \qquad \text{et} \\ A_{m,n}(f) &= \sum_k h_{2n-k}\ A_{m-1,k}(f) = \langle \phi_{m,n}, f\rangle, \end{aligned}$$

où :

$$g_l = (-1)^l\ h_{1-l}$$

$$h_n = 2^{\frac{1}{2}} \int \phi(x-n)\ \phi(2x)dx$$

sont les réponses impulsionnelles de filtres discrets notés H et G.
Ces deux équations définissent une décomposition en sous-bande du signal, à partir d'un filtre passe-bas H et un filtre passe-haut G.
La reconstruction exacte (due à l'orthonormalité entre les bases d'ondelettes) à la résolution supérieure est donnée par :

$$A_{m-1,k}(f) = \sum_n [h_{2n-k} A_{m,n}(f) + g_{2n-k} Dm, n(f)].$$

S. Mallat définit quelques contraintes supplémentaires sur la construction des filtres h et g, dont nous parlerons pas ici. Le lecteur interessé se référera à l'article de Mallat[110].

La décomposition du signal sur une base d'ondelettes orthogonales relativement douces permet une reconstruction exacte. Cependant, en compression numérique, il est nécessaire d'imposer des contraintes sur l'élaboration des filtres. Ces derniers doivent être :

- courts (à support compact) pour minimiser les temps de calcul

- symétriques (à phase linéaire)

Des travaux antérieurs [39] ont montré que les conditions énoncées ne peuvent pas être satisfaites simultanément et qu'une solution est de restreindre la contrainte d'orthogonalité en utilisant des bases d'ondelettes bi-orthogonales [35]. L'approximation $A_{m,n}(f)$ et la différence d'information $D_{m,n}(f)$ dans la phase d'analyse s'écrivent dans le cas bi-orthogonal de la même manière que dans le cas orthogonal. La reconstruction (phase de synthèse) se fait quant à elle à l'aide de deux filtres $\tilde{h}$ et $\tilde{g}$ supplémentaires :

$$A_{m-1,k}(f) = \sum_n [\tilde{h}_{2n-k} A_{m,n}(f) + \tilde{g}_{2n-k} Dm, n(f)],$$

où h, g, $\tilde{h}$ et $\tilde{g}$ respectent :

$$\tilde{g}_n = (-1)^n h_{1-n},$$

$$g_n = (-1)^n \tilde{h}_{1-n},$$

$$\sum_n h_n \tilde{h}_{n+2k} = \delta_{k,0}.$$

Application à l'image

S. Mallat a défini une fonction d'échelle 2D séparable s'écrivant $\phi(x,y) = \phi_{m,n_x}(x)\phi_{m,n_y}(y) = \phi(x)\phi(y)$, où $\phi(x)$ est une fonction d'échelle monodimensionnelle. L'information de différence entre deux approximations de l'image originale se calcule dans ce cas à partir de trois ondelettes bidimensionnelles définies par :

$$\Psi^H(x,y) = \phi(x)\Psi(y),$$

$$\Psi^V(x,y) = \Psi(x)\phi(y),$$

$$\Psi^D(x,y) = \Psi(x)\Psi(y).$$

Une telle représentation multirésolution de l'image fournit à chaque échelle les quatre sous-images décorrélées suivantes :

- une image de faible résolution;
- une image de détails horizontaux;
- une image de détails verticaux;
- une image de détails diagonaux.

Cette décomposition, lorsqu'elle est réalisée à l'aide de filtres 1D est bien sûr à rapprocher du codage en sous-bandes décrit dans la section 2.7.8.

Application de la transformée en ondelettes à la compression des images

Barlaud et *al.* proposent de quantifier vectoriellement les coefficients d'ondelettes orthogonales [2] puis bi-orthogonales [7]. Leur algorithme consiste à construire un dictionnaire "multirésolution". Chaque sous-image de la décomposition est codée séparément. Le dictionnaire est ainsi constitué de plusieurs sous-dictionnaires de petite taille, optimisés pour le codage de chacune des résolutions. Il en résulte un gain de temps au niveau de la recherche des vecteurs de codage, ainsi qu'une meilleure restitution des hautes fréquences, qui étaient lissées par l'algorithme LBG appliqué directement sur l'image originale. La quantification est faite de manière plus ou moins grossière en fonction des résolutions, en tenant compte de la sensibilité de l'œil humain.

2.7.10 Méthodes dites de seconde génération

M. Kunt [97] a utilisé le terme de seconde génération pour décrire une classe de méthodes de compression d'images permettant d'atteindre des taux de compression supérieurs à 10:1 ou 20:1. Les algorithmes utilisés ne sont plus fondés sur des prédicteurs ou des transformations linéaires réversibles, mais cherchent à décrire et à coder l'image en terme de régions texturées et de contours, tels que les régions correspondent le plus fidèlement possible aux objets de l'image. Nous ne citerons dans cette partie que deux méthodes appartenant à cette classe :

- le codage basé sur la croissance de régions. L'algorithme de segmentation retourne des régions dont les points intérieurs partagent la même propriété (proches d'une même moyenne de niveaux de gris, même énergie, ...). La croissance des régions est précédée d'une phase d'atténuation du bruit dans l'image tout en préservant les contours, de manière à limiter l'apparition de faux contours ainsi que le nombre de régions.

- le codage par division et fusion (en anglais *split and merge*). Nous montrerons dans le chapitre 4 comment utiliser un tel algorithme de division-fusion pour construire une triangulation de Delaunay adaptée au contenu de l'image, en vue du codage par fractales.

La *morphologie mathématique* est un exemple de méthode qui offre des outils pour le codage de seconde génération. Elle permet de prendre en compte les caractéristiques de l'image en niveaux de gris telles que la taille, la forme, ou le contraste des objets qui la composent. Le lecteur interessé trouvera dans [150] les définitions mathématiques des différents opérateurs de base que sont la dilatation, l'érosion, l'ouverture et la fermeture.
L'ouverture et la fermeture peuvent de plus être combinées de manière à former des filtres morphologiques plus complexes, tels que l'ouverture et la fermeture par reconstruction, adaptés au type de simplification désirée de l'image. Dans ces deux cas, les opérations de base sont itérées plusieurs fois jusqu'à vérification d'un critère fixé, de forme ou de surface des zones arasées ou comblées.

La compression par morphologie mathématique est réalisée en deux étapes : La première étape consiste à simplifier l'image en faisant apparaître des zones homogènes, tout en préservant les contours originaux. Ceci est fait en utilisant les principaux opérateurs morphologiques listés précédemment. La segmentation proprement dite de l'image est réalisée à l'aide de l'algorithme de partage des eaux (*watershead*) consistant à définir plus précisément les contours des régions. La seconde étape consiste à coder les contours détectés ainsi que le contenu des régions homogènes ou texturées. Différents résultats de compression et de segmentation selon cette approche sont donnés dans [144] [115].

2.8 Norme de compression JPEG

Le "Joint Photographic Experts Group" a défini un standard pour compresser les images fixes, couleurs ou monochromes, de scènes réelles. Le programme est efficace pour le codage des images naturelles présentant une continuité spatiale, mais ne l'est pas pour les images telles que les bandes dessinées, les lettres manuscrites ou dactylographiées. La norme JPEG a été définie pour compresser des images appréciées par l'œil humain, et non analysées par un ordinateur qui détecterait des erreurs non perceptibles visuellement.

Différentes contraintes ont été imposées au standard :

- La norme JPEG doit être très proche des techniques nouvelles de compression, en terme de taux de compression, de qualité de restitution et de temps de calculs;
- JPEG doit pouvoir compresser n'importe quel type d'images réelles (images de tailles différentes, multi-composantes etc ...)
- L'algorithme doit être implémentable sans trop de problèmes sur une grande gamme de CPUs, et sur des cartes spécialisées;
- Le codage doit pouvoir être séquentiel, progressif, sans pertes et/ou hiérarchique.

En 1988, le groupe JPEG a retenu l'utilisation de l'"ADCT" (Adaptive Discret Cosinus Transform) basée sur la transformation DCT sur des blocs d'image de taille 8×8.

Détails de l'algorithme de compression JPEG non-réversible :

- L'image est décomposée en blocs de taille 8×8. Les pixels de chacun des sous-blocs sont translatés dans l'intervalle $[-2^{p-1}, 2^{p-1} - 1]$

- Les blocs sont individuellement transformés par DCT. Les $64 \times n$ coefficients ainsi obtenus caractérisent l'image de manière unique.

- Chaque coefficient est pondéré (normalisé). La matrice de pondération est la même pour tous les blocs. Les coefficients normalisés sont ensuite uniformément quantifiés par arrondi à l'entier le plus proche. La pondération suivie de la quantification sont la cause d'une perte d'information rendant la compression non réversible.
  La norme prévoit 4 tables de pondération différentes. Celles-ci sont construites de manière à rendre les dégradations "juste perceptibles" visuellement. Chaque coefficient issu de la DCT code une information fréquentielle. Des expériences psychovisuelles ont permis d'établir des tables tenant compte de l'importance perceptuelle de chaque coefficient.

- Un traitement particulier est porté au premier coefficient intitulé coefficient DC. Il correspond à la valeur moyenne des pixels images du bloc considéré. Les coefficients DC de chacun de blocs transformés contiennent une grande partie de l'énergie totale de l'image. De plus, les coefficients associés à des blocs voisins sont corrélés. Pour ces raisons ils sont codés par une méthode du type DPCM sans perte.

- Les 63 autres coefficients de chacun des blocs sont réordonnés de manière à former un vecteur unidimensionnel. Le réordonnancement est obtenu par un parcours en "zig-zag" de la matrice des coefficients de la transformation DCT. Les coefficients qui pondérent les basses (resp. hautes) fréquences se trouvent au début (resp. à la fin) du vecteur. Après les étapes de pondération et de quantification, le vecteur contient un grand nombre de zéros associés aux hautes fréquences spatiales.

- Une compression entropique des coefficients AC au sein d'un même vecteur termine la chaîne de traitement. JPEG propose deux méthodes : le codage de Huffman, et le codage arithmétique.

Algorithme JPEG sans perte d'information

La transformation DCT n'est dans ce cas pas utilisée. JPEG préconise l'utilisation simple d'une méthode prédictive suivie d'un codage entropique (Huffman ou arithmétique). Le traitement se fait sur l'image entière, et non sur des blocs. Ce type de codage permet des taux de compression en moyenne égaux à 2:1 pour des images couleurs naturelles.

Codage des images à plusieurs composantes

Un en-tête est inséré dans le fichier compressé JPEG. Son rôle est de coder les informations caractérisant l'image compressée. L'en-tête est ainsi constituée des éléments suivants :

- le nombre de composantes de l'image, pouvant aller de 1 (image en niveaux de gris) à 255 (imges multi-spectrales) en passant par 3 (images couleurs codées dans l'espace RGB, YUV, CIELUV, CIELAB ...);

- le nombre B de bits codant l'échantillon d'une composante. Ce nombre est égal à 8 ou 12 pour les images nécessitant une grande précision (images médicales). Dans le cas des codeurs JPEG sans perte, B est compris entre 2 et 16;

- des informations sur les dimensions spatiales de chacune des composantes, celles-ci pouvant être échantillonnées différemment (codage hiérarchique);

- la manière dont sont disposées les informations dans le fichier compressé. L'image multi-composantes peut être reconstruite de manière séquentielle (composante après composante) ou parallèle. Chaque donnée unité (bloc dans une composante, de taille 8×8 pixels) peut être disposée sur des composantes entrelassées. Une partie seulement des composantes peut être entrelassée, à l'intérieur de la même image compressée ;

- la ou les tables de pondération (JPEG en autorise 4 différentes ainsi que une à quatre tables pour le codage entropique). Dans l'espace YUV codant des images couleurs, la chrominance (information couleur) codée par les composantes U et V utilise une table de Huffman, et la luminance (composante Y) utilise une deuxième table de Huffman.

2.9 Conclusion

Le but de ce chapitre était d'introduire différentes méthodes de codage entropique et de compression réversible ou irréversible des images. Certaines d'entre elles sont incorporées dans des schémas hybrides basés sur la transformation fractale. Nous serons ainsi amenés à faire appel, dans la suite de ce mémoire, à la transformation en cosinus discrète, à la quantification vectorielle, au codage par ondelettes ainsi qu'au codage en sous-bandes.

Chapitre 3

IFS et transformation fractale

3.1 Introduction

Nous détaillerons dans la section 3.3.2 la méthode originale de compression des images naturelles par une approche fractale, proposée par Jacquin en 1989. La section 3.3.3 présente une formulation algébrique de la transformation fractale. Celle-ci a été largement utilisée dans la littérature, suite aux travaux de Lundheim sur la compression des signaux monodimensionnels [108]. Nous continuons par une description succincte de la méthode de Fisher, celle-ci étant plus longuement discutée dans le chapitre 4. Nous terminons en détaillant la méthode de Dudbridge qui diffère des approches classiques puisqu'elle ne nécessite pas de mises en correspondance de blocs à l'intérieur de l'image, et donc de recherches coûteuses en temps de calculs.

3.2 Théorie des IFS

Nous rappelons dans cette section les principaux aspects de la théorie des IFS permettant le codage et la synthèse d'images fractales binaires et en niveaux de gris.

3.2.1 Transformation Lipschitzienne

Soit $\omega : \mathbb{R}^2 \to \mathbb{R}^2$ une transformation définie sur l'espace métrique $(\mathbb{R}^2, d)$. d désigne la distance entre deux points de $\mathbb{R}^2$. La transformation ω est dite Lipschitzienne, avec pour facteur de Lipschitz le réel s strictement positif si

$$d(\omega(x), \omega(y)) \leq s.d(x, y) \quad \forall x, y \in \mathbb{R}^2. \tag{5}$$

3.2.2 Transformation contractante

Soit $\omega : \mathbb{R}^2 \to \mathbb{R}^2$ une transformation définie sur l'espace métrique $(\mathbb{R}^2, d)$. La transformation ω est dite contractante, avec pour facteur de contraction le réel s, $0 < s < 1$ si

$$d(\omega(x), \omega(y)) \leq s.d(x, y) \quad \forall x, y \in \mathbb{R}^2. \tag{6}$$

3.2.3 Point fixe

Une transformation contractante ω possède un unique point fixe $x_f \in \mathbb{R}^2$, tel que $\omega(x_f) = x_f$. Pour tout point x élément de $\mathbb{R}^2$, la séquence $\{\omega^{on}(x) : n = 0, 1, 2, \dots\}$ converge vers x_f :

$$\lim_{n \to \infty} \omega^{on}(x) = x_f \quad \forall x \in \mathbb{R}^2. \tag{7}$$

3.2.4 Distance de Hausdorff

Considérons l'espace métrique $(\mathbb{R}^2, d)$. $H(\mathbb{R}^2)$ désigne l'espace dont les éléments sont les sous-ensembles compacts non vides de $\mathbb{R}^2$, différents de l'ensemble vide.

La distance du point x élément de $\mathbb{R}^2$ à l'ensemble B élément de $H(\mathbb{R}^2)$, notée $d(x, B)$, est définie par :

$$d(x, B) = \min\{d(x, y) : y \in B\}.$$

La distance de l'ensemble A élément de $H(\mathbb{R}^2)$ à l'ensemble B élément de $H(\mathbb{R}^2)$, notée $d(A, B)$, est définie par :

$$d(A, B) = \max\{d(x, B) : x \in A\}.$$

La distance de Hausdorff entre deux ensembles A et B éléments de $H(\mathbb{R}^2)$, notée $h_d(A, B)$, est définie par :

$$h_d(A, B) = \max\{d(A, B), d(B, A)\}. \tag{8}$$

3.2.5 Transformation contractante sur l'espace $H(\mathbb{R}^2)$

Soit une transformation contractante $\omega : \mathbb{R}^2 \to \mathbb{R}^2$ définie sur l'espace métrique $(\mathbb{R}^2, d)$ avec le réel s pour facteur de contraction. La transformation $\omega : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ définie par :

$$\omega(B) = \{\omega(x) : x \in B\}, \quad \forall B \in H(\mathbb{R}^2) \tag{9}$$

est contractante sur $(H(\mathbb{R}^2), h_d)$, avec s pour facteur de contraction. h_d désigne la distance de Hausdorff.

3.2.6 Système de transformations itérées

Un IFS défini dans l'espace métrique complet $(\mathbb{R}^2, d)$ est composé d'un ensemble de N transformations $\omega_i : \mathbb{R}^2 \to \mathbb{R}^2$ $(i = 1, \ldots, N)$, à chacune desquelles est associé un facteur de Lipschitz s_i.
Dans la suite de cette section nous considérerons que les N transformations sont contractantes : le système de transformations est dans ce cas appelé IFS *hyperbolique*. Le facteur de contraction de l'IFS hyperbolique, noté s est égal à $\max\{s_i : i = 1, \ldots, N\}$.

3.2.7 Attracteur d'un IFS

Soit un IFS $\{\mathbb{R}^2; \omega_i, i = 1, \ldots, N\}$. Il a été démontré [12] que l'opérateur $W : H(\mathbb{R}^2) \to H(\mathbb{R}^2)$ défini par :

$$W(B) = \bigcup_{i=1}^{N} \omega_i(B), \quad \forall B \in H(\mathbb{R}^2) \tag{10}$$

est contractant et a pour facteur de contraction celui de l'IFS. L'opérateur W possède un unique *point fixe* $\mathcal{A}_t$ donné par :

$$\mathcal{A}_t = W(\mathcal{A}_t) = \lim_{n \to \infty} W^{on}(X), \quad \forall X \in H(\mathbb{R}^2) \tag{11}$$

L'objet $\mathcal{A}_t$ est aussi appelé *attracteur* de l'IFS. Il est invariant sous la transformation W et est égal à l'union de N copies de lui-même transformées par $\omega_1, \ldots, \omega_N$. L'objet invariant est dit auto-similaire, ou "auto-affine" lorsque les transformations élémentaires ω_i sont affines.

Exemple a.: Soit l'IFS $\{\mathbb{R}^2; \omega_i, i = 1, \ldots, 3\}$ composé des transformations affines suivantes :

$$\begin{aligned}
\omega_1 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{y_0}{2} \end{pmatrix} \\
\omega_2 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{-x_0}{2} \\ \frac{-y_0}{2} \end{pmatrix} \\
\omega_3 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{x_0}{2} \\ \frac{-y_0}{2} \end{pmatrix}
\end{aligned} \tag{12}$$

Son facteur de contraction est égal à 0.25.
L'attracteur codé par l'IFS, appelé *triangle de Sierpinski*, est illustré par la figure 10.

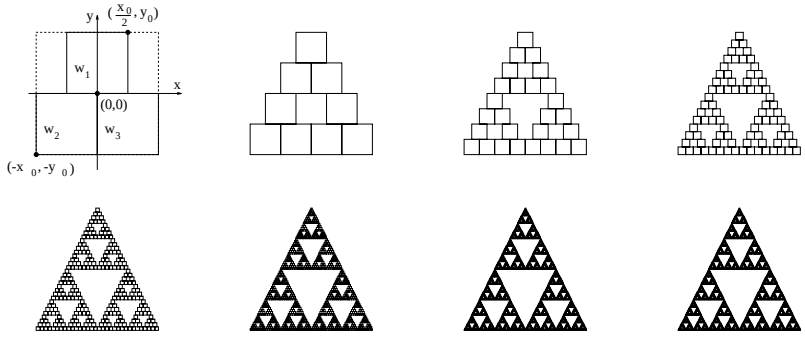


FIG. 10 - Attracteur de l'IFS de l'exemple a. Le carré initial centré à l'origine et de cotés de longueur $2x_0, 2y_0$ est transformé en trois carrés homothétiques par les transformations contractantes ω_1, ω_2 et ω_3. Ce processus est ensuite itéré.

3.2.8 Théorème du collage

Ce théorème démontré dans [12] fournit une borne supérieure à la distance de Hausdorff h_d entre un point A inclus dans $H(\mathbb{R}^2)$ et l'attracteur $\mathcal{A}_t$ d'un IFS.

Théorème :
Soit l'espace métrique complet $(\mathbb{R}^2, d)$. Soit un point A appartenant à $H(\mathbb{R}^2)$,
et un IFS $\{\mathbb{R}^2; \omega_1, \omega_2, \ldots, \omega_n\}$ avec le réel $0 \leq s < 1$ pour facteur de contraction.

On vérifie la relation :

$$h_d(A, \mathcal{A}_t) \leq \frac{1}{1-s}\, h_d(A, \bigcup_{i=1}^{N} \omega_i(A)) \tag{13}$$

Le théorème indique que s'il est possible de transformer un objet A de manière à vérifier $A \simeq W(A)$ tout en s'assurant que W est contractant, alors le point fixe $\mathcal{A}_t$ de l'opérateur W est proche de A. L'opérateur W, défini dans la section 3.2.7, caractérise pleinement dans ce cas l'approximation[1] $\mathcal{A}_t$ de l'objet A, et code ce dernier de manière exacte si $A = W(A)$ [11].

Exemple b. : Considérons un carré noté A, transformé à l'aide de quatre transformations affines contractantes de manière à obtenir quatre sous-carrés. Si ces derniers recouvrent exactement le carré A de départ, le théorème du collage est vérifié. L'attracteur de l'IFS est alors un carré identique au carré A. L'IFS $\{\mathbb{R}^2; \omega_i, i = 1, \ldots, 4\}$ est composé des transformations affines suivantes :

$$\begin{aligned}
\omega_1 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{-x_0}{2} \\ \frac{y_0}{2} \end{pmatrix} \\
\omega_2 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{x_0}{2} \\ \frac{y_0}{2} \end{pmatrix} \\
\omega_3 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{-x_0}{2} \\ \frac{-y_0}{2} \end{pmatrix} \\
\omega_4 \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{x_0}{2} \\ \frac{-y_0}{2} \end{pmatrix}
\end{aligned} \tag{14}$$

L'attracteur codé par cet IFS est illustré par la figure 11. Le processus itératif initialisé sur un cercle converge vers le carré A. On aurait le même résultat en initialisant le processus sur une autre forme qu'un cercle.

1. le codage sera d'autant plus efficace que l'objet A est auto-similaire

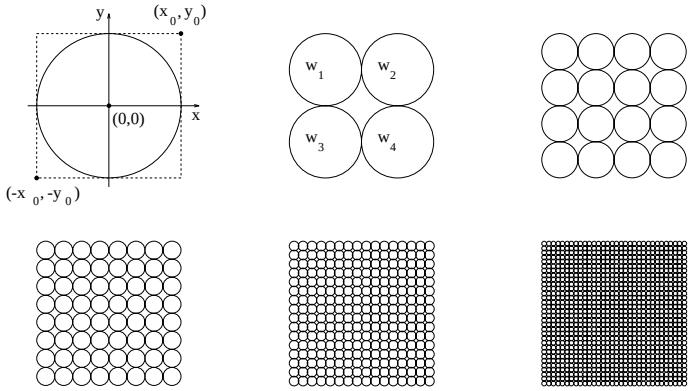


FIG. 11 - Attracteur de l'IFS de l'exemple b.

3.2.9 Transformation finalement contractante

Soit une transformation Lipschitzienne ω. Si il existe un entier n tel que la transformation ω^{on} est contractante, alors ω est dite finalement contractante. L'entier n est appelé exposant de contraction.
L'opérateur W défini par l'équation (10) peut être finalement contractant si un nombre limité de transformations ω_i ne sont pas contractantes.

3.2.10 Théorème du collage généralisé

Considérons l'opérateur W finalement contractant, avec pour exposant de contraction l'entier n; il existe alors un unique point fixe $x_f \in \mathbb{R}^2$ tel que :

$$x_f = W(x_f) = \lim_{k \to \infty} W^{ok}(x) \quad \forall x \in \mathbb{R}^2$$

Dans ce cas,

$$h_d(A, \mathcal{A}_t) \leq \frac{1}{1-s} \frac{1-\sigma^n}{1-\sigma} h_d(A, W(A)) \tag{15}$$

où σ est le facteur de Lipschitz de W et s est le facteur de contraction de W^{on} [62] [108].

3.2.11 Attracteurs et mesures invariantes

Nous nous contentons seulement d'introduire dans ce paragraphe les principaux concepts concernant la génération des objets fractals en niveaux de gris. Le lecteur intéressé trouvera plus d'information dans le chapitre 9 de la référence [8].

La génération d'un objet fractal en niveaux de gris est rendue possible en associant une probabilité p_i à chacune des N transformations ω_i d'un IFS :

$$\forall i, p_i \geq 0 \qquad \text{et} \qquad \sum_{i=1:N} p_i = 1$$

L'objet fractal induit une mesure[2] μ sur son support, associée à un opérateur de Markov $\mathcal{M}$ donné par l'expression suivante :

$$\mu_n(B) = \mathcal{M}\mu_{n-1}(B) = \sum_{i=1:N} p_i \mu_{n-1}\left(\omega_i^{-1}(B)\right) \tag{16}$$

où B est un sous-ensemble de Borel de $\mathbb{R}^2$ et $\mu_n(B)$ est la probabilité de B à l'itération n. Il est montré qu'un tel opérateur $\mathcal{M}$ est contractant [8] (au regard de la métrique de Hutchinson dans l'espace des mesures), et donc qu'il existe une unique mesure μ appelée *mesure invariante de l'IFS*, donnée par :

$$\mathcal{M}\mu = \mu = \lim_{k\to\infty} \mathcal{M}^{ok}(\mu_0), \qquad \forall \mu_0. \tag{17}$$

De plus, la mesure invariante μ a pour support l'attracteur de l'IFS.

Considérons maintenant le cas pratique dans lequel l'objet fractal est une image numérique. La valeur normalisée d'un pixel B de l'image correspond à la probabilité du sous-ensemble de Borel B de $\mathbb{R}^2$. D'après (16), la valeur d'un pixel B dans l'image μ_n est égale à la somme des valeurs des pixels $\omega_i^{-1}(B)$ dans μ_{n-1}, multipliées par les probabilités p_i.

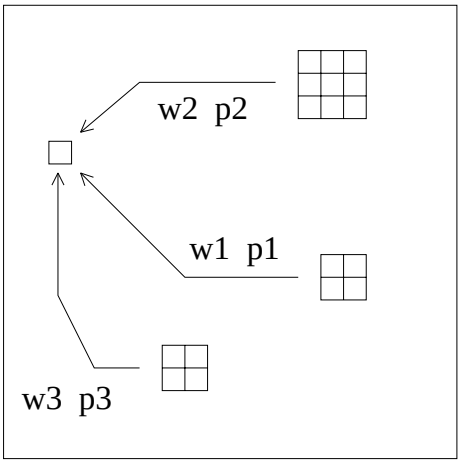


FIG. 12 - Illustration du calcul de la mesure μ_n.

La mesure invariante associée à l'attracteur d'un IFS peut aussi s'obtenir en itérant un très grand nombre de fois les trois opérations suivantes, initialisées sur un point quelconque de $\mathbb{R}^2$ noté x_0 :

- choisir une transformation ω_i avec la probabilité p_i,
- calculer $x_1 = \omega_i(x_0)$,

2. Nous rappelons qu'une mesure est au sens physique du terme une grandeur mesurable (intensité lumineuse par exemple) qui permet d'associer des poids relatifs aux différentes points de son support (le support de la mesure est l'ensemble des points sur lesquels elle est définie).

- remplacer x_0 par x_1

Lorsque le nombre d'itérations est suffisamment élevé, les points se répartissent sur un ensemble compact de $\mathbb{R}^2$ définissant l'attracteur de l'IFS. La densité (fréquence des visites) en chacun des pixels de l'attracteur définit la mesure invariante μ (voir [54] et le "jeu du chaos" dans [8]).

3.2.12 Problème inverse

La définition du problème inverse est la suivante :

> Etant donné un objet A appartenant à $H(\mathbb{R}^2)$ et une mesure μ sur A : comment trouver l'IFS et l'ensemble des probabilités p_i pour lesquels l'objet A est l'attracteur et μ est la mesure invariante?

Différents travaux ont tenté de résoudre ce problème d'optimisation sous contraintes qui est difficile de par sa grande dimensionnalité et l'irrégularité de la fonction à minimiser. Les solutions envisagées utilisent des techniques diverses basées sur les algorithmes génétiques [109], sur les ondelettes [140], et autres [11] [102] [112] [29] [161] [95].

3.3 Compression des images naturelles par fractales

3.3.1 Introduction

Le but de cette section est d'introduire les méthodes de base permettant d'associer à une image naturelle une transformation finalement contractante W ayant pour attracteur une approximation de l'image elle-même. Si ce problème est résolu on peut alors parler de compression d'images puisque la mémorisation des coefficients de W nécessite "moins d'information" que la mémorisation de l'image originale. On parle aussi de compression avec pertes du fait que l'attracteur ne constitue qu'une approximation de l'image originale.
Il est à noter que le problème est différent du problème inverse introduit dans la section 3.2.12 car les transformations spatiales mises en jeu ne sont pas appliquées à l'image entière mais à des sous-parties de l'image puisque celle-ci n'est pas fractale. Nous verrons que parmi les méthodes présentées, la méthode de Dudbridge est celle qui s'en rapproche le plus.

La compression d'une image par fractales repose sur une transformation que nous appelons *transformation fractale* et qui consiste à transformer l'image à l'aide d'un opérateur finalement contractant, de manière à ce que son aspect visuel reste quasiment inchangé. Pour cela, la transformation de l'image est composée de N sous-transformations élémentaires, chacune opérant sur un bloc de l'image, de la manière

suivante (voir fig. 13) :

L'image A est partitionnée en N blocs $\mathbf{r}_n$ appelés *blocs destination*. Elle s'écrit :

$$A = \bigcup_{n=1}^{N} \mathbf{r}_n \tag{18}$$

Cette partition du support de l'image en blocs destination est appelée partition R.

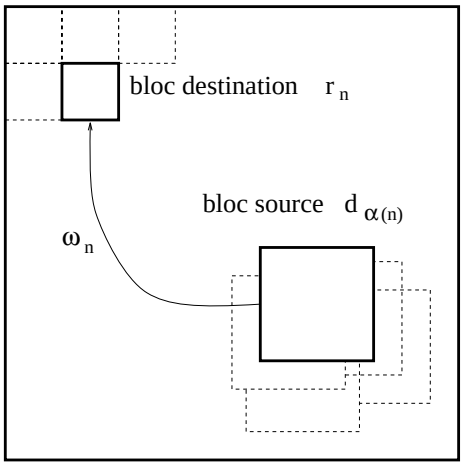


FIG. 13 - Blocs destination $\mathbf{r}_n$ et blocs source $\mathbf{d}_{\alpha(n)}$.

Chaque bloc destination est ensuite mis en correspondance avec un autre bloc transformé $\omega_n(\mathbf{d}_{\alpha(n)})$ lui "ressemblant" au sens d'une mesure d'erreur sur les niveaux de gris. Le bloc $\mathbf{d}_{\alpha(n)}$, appelé *bloc source* est recherché au travers d'une *librairie* composée de Q blocs appartenant à l'image : $\alpha(n)$ est donc une application de $[1 \ldots N]$ vers $[1 \ldots Q]$. Les Q blocs ne forment pas nécessairement une partition de l'image mais sont représentatifs de toute l'image. La transformation W de l'image A est formulée à l'aide de l'équation suivante :

$$W(A) = \bigcup_{n=1}^{N} \omega_n(\mathbf{d}_{\alpha_n}) = \bigcup_{n=1}^{N} \hat{\mathbf{r}}_n \tag{19}$$

où $\hat{\mathbf{r}}_n$ est l'approximation du bloc destination $\mathbf{r}_n$ obtenue en transformant le bloc source $\mathbf{d}_{\alpha_n}$ par ω_n (l'opération permettant d'obtenir le bloc $\hat{\mathbf{r}}_n$ à partir du bloc $\mathbf{d}_{\alpha_n}$ est appelée *opération de "collage"*).

3.3.2 Méthode de A. Jacquin

L'approche de A. Jacquin [85] est fondée sur une partition R régulière à géométrie carrée. L'image est partitionnée en blocs destination carrés[3] de taille fixe égale à B^2 pixels ($B = 8$). L'algorithme recherche, pour chacun des blocs destination $\mathbf{r}_n$,

3. Dans la formulation de Jacquin, ces blocs portent le nom de *blocs parents*

le bloc source $\mathbf{d}_{\alpha(n)}$ de taille D^2 $(D = 2B)$ qui minimise l'erreur $d(\mathbf{r}_n, \hat{\mathbf{r}}_n)$ où $\hat{\mathbf{r}}_n$ est l'approximation de $\mathbf{r}_n$ calculée à partir du bloc source $\mathbf{d}_{\alpha(n)}$. La mesure d'erreur d est donnée par :

$$d(\mathbf{r}_n, \hat{\mathbf{r}}_n) = \sum_{j=1}^{B^2} \left(r_{n_j} - \hat{r}_{n_j} \right)^2 \tag{20}$$

où r_{n_j} et $\hat{r}_{n_j}$ sont respectivement les valeurs des pixels d'indice j à l'intérieur du bloc original $\mathbf{r}_n$ et du bloc collé $\hat{\mathbf{r}}_n$. L'opération de collage, appelée *collage parent*, est détaillée dans le paragraphe suivant.

Collage d'un bloc source sur un bloc destination

L'opération de collage d'un bloc source $\mathbf{d}_{\alpha(n)}$ sur un bloc destination $\mathbf{r}_n$, réalisée par la transformation ω_n, se décompose en deux parties :

- une transformation spatiale déforme le support du bloc $\mathbf{d}_{\alpha(n)}$;
- une transformation "massique" agit sur la luminance des pixels du bloc $\mathbf{d}_{\alpha(n)}$ déformé.

Ces deux points sont détaillés dans la suite de ce paragraphe.

La *transformation spatiale* ramène le bloc source $\mathbf{d}_{\alpha(n)}$ de taille D^2 à l'échelle et au-dessus du bloc destination $\mathbf{r}_n$ de taille B^2. Le bloc ainsi transformé, noté $\mathbf{b}_2^{(n)}$, est obtenu par décimation des pixels du bloc source : un pixel de coordonnées (x_i, y_j) dans $\mathbf{b}_2^{(n)}$ est donné par l'équation suivante :

$$b_2^{(n)}(x_i, y_j) = \frac{1}{4} \left[d_{\alpha(n)}(x_k, y_l) + d_{\alpha(n)}(x_k, y_{l+1}) + d_{\alpha(n)}(x_{k+1}, y_l) + d_{\alpha(n)}(x_{k+1}, y_{l+1}) \right] \tag{21}$$

dans laquelle (x_k, y_l) sont les coordonnées d'un pixel de niveau de gris noté $d_{\alpha(n)}$ à l'intérieur du bloc $\mathbf{d}_{\alpha(n)}$.

La *transformation massique* agit sur le bloc $\mathbf{b}_2^{(n)}$ pour approximer le bloc destination $\mathbf{r}_n$. La complexité de cette transformation dépend de la nature du bloc $\mathbf{r}_n$ considéré. Jacquin propose pour cela de classifier les blocs carrés à l'aide de la méthode développée par Ramamurthi et Gersho [136] : trois classes regroupent les blocs homogènes, les blocs texturés et les blocs avec contours (simples et divisés) de l'image. Selon la classe à laquelle appartient le bloc destination $\mathbf{r}_n$, une transformation massique plus ou moins complexe lui est associée. Celle-ci dépend du bloc décimé $\mathbf{b}_2^{(n)}$ et/ou d'un bloc constant $\mathbf{b}_1^{(n)}$ formé de pixels tous égaux à un. Au bloc $\mathbf{b}_2^{(n)}$ sera associé un coefficient d'*échelle* noté $\beta_2^{(n)}$ et au bloc $\mathbf{b}_1^{(n)}$ un coefficient de

décalage noté $\beta_1^{(n)}$. Le choix du type de transformation est fonction de la procédure suivante :

si le bloc $\mathbf{r}_n$ est homogène : absorption des niveaux de gris de $\mathbf{r}_n$. Aucune recherche de blocs source $\mathbf{d}_{\alpha(n)}$ n'est effectuée. La transformation de $\mathbf{r}_n$, codée sur I_s bits, est donnée par :

$$\hat{\mathbf{r}}_n = \beta_1^{(n)} \mathbf{b}_1^{(n)}$$

où l'entier $\beta_1^{(n)}$ est compris entre 0 et 255.

si le bloc $\mathbf{r}_n$ est texturé : recherche du bloc source $\mathbf{d}_{\alpha(n)}$, puis modification de contraste et décalage. La transformation de $\mathbf{d}_{\alpha(n)}$, codée sur I_m bits, est donnée par :

$$\hat{\mathbf{r}}_n = \beta_2^{(n)} \mathbf{b}_2^{(n)} + \beta_1^{(n)} \mathbf{b}_1^{(n)}$$

où $\beta_2^{(n)}$ appartient à l'ensemble $\{0.7, 0.8, 0.9, 1.0\}$ et l'entier $\beta_1^{(n)}$ est compris entre -255 et 255.

si le bloc $\mathbf{r}_n$ contient des contours : recherche du bloc source $\mathbf{d}_{\alpha(n)}$, puis modification de contraste, décalage et isométrie discrète $\imath_n$ (rotations de 0, +90, -90 et +180 degrés, réflexions suivant les axes de symétrie verticaux et horizontaux, et réflexions suivant les deux axes diagonaux). La transformation de $\mathbf{d}_{\alpha(n)}$, codée sur I_e bits, est donnée par :

$$\hat{\mathbf{r}}_n = \imath_n \left(\beta_2^{(n)} \mathbf{b}_2^{(n)} + \beta_1^{(n)} \mathbf{b}_1^{(n)} \right)$$

où $\beta_2^{(n)}$ appartient à l'ensemble $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ et $\beta_1^{(n)}$ est compris entre -255 et 255.

Lorsque le bloc destination est texturé ou recouvre des contours, le coefficient d'échelle β_2 est calculé de manière à rendre égaux les écarts types des deux blocs $\mathbf{b}_2^{(n)}$ et $\mathbf{r}_n$. Il est ensuite approximé par le coefficient appartenant à un ensemble de valeurs prédéfinies réelles positives, et inférieures à un. Le coefficient de décalage β_1 est calculé de manière à ce que les moyennes des pixels des deux blocs $\mathbf{b}_2^{(n)}$ et $\mathbf{r}_n$ soient égales. Il n'est pas quantifié.

La recherche exhaustive du bloc source $\mathbf{d}_{\alpha(n)}$ est effectuée en déplaçant sur le support de l'image un bloc carré d'un pas de $\delta_h = \delta_v = 4$ pixels dans les directions horizontales et verticales. Lorsque deux blocs sont comparés, les huit isométries discrètes sont considérées. Pour une image de taille 256×256, une telle recherche est ainsi effectuée au travers d'une librairie composée de Q blocs source où :

$$Q = 8 \left(\frac{256 - 16}{4} + 1 \right)^2 = 29768$$

Dans le cas d'une image de taille 512×512 pixels, le nombre Q de blocs source s'élève à 125000.

Optimisation

Jacquin propose dans un deuxième temps de rediviser les blocs parents collés $\hat{\mathbf{r}}_n$ en quatre sous-blocs destination de taille 4×4 pixels appelés *blocs enfants* (voir figure 14). Les blocs obtenus sont comparés à leurs correspondants dans l'image originale, au sens de la mesure d'erreur donnée par la formule (20), avec $B = 4$. Si l'erreur est supérieure à un seuil donné, ils sont codés séparément en recherchant dans l'image le meilleur bloc source de taille 8×8. Le processus de collage est dans ce cas appelé *collage enfant*.
Si, pour un bloc parent, trois ou quatre collages enfant sont nécessaires, seuls sont codés les 4 collages enfant. Si un ou deux collages enfant sont nécessaires, le bloc parent est codé par le collage parent complété des collages enfants.

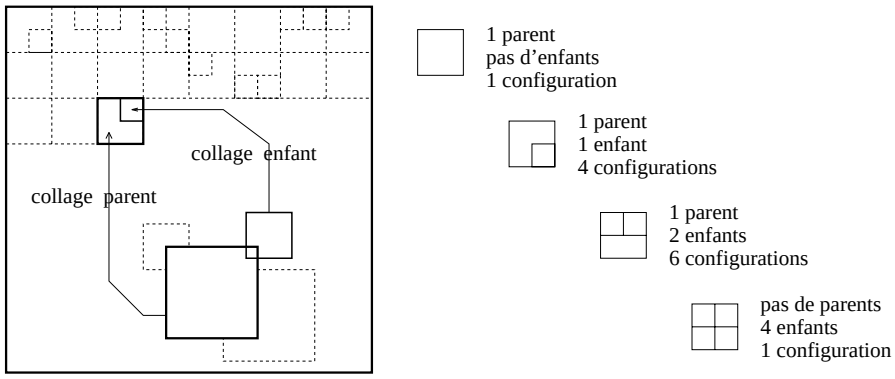


FIG. 14 - Partitionnement formé de blocs parents et enfants.

Codage de l'opération de collage sur un bloc destination

La mémorisation du collage d'un bloc source (parent ou enfant) $\mathbf{d}_{\alpha(n)}$ sur un bloc destination (parent ou enfant) $\mathbf{r}_n$ comprend :

- l'indice du bloc source $\mathbf{d}_{\alpha(n)}$ retenu parmi les Q blocs de la librairie à condition que ceux-ci soient rangés dans une liste de blocs et que leur organisation sur le support de l'image soit connue. Sinon, il est nécessaire de mémoriser les coordonnées (x_k, y_l) d'un pixel de référence dans le bloc $\mathbf{d}_{\alpha(n)}$ (par exemple le coin supérieur gauche dans le cas d'un bloc carré).

- l'isométrie utilisée lors du collage (une parmi huit)

- les coefficients β_1 et β_2 de la transformation massique

Cette information est associée à chacun des N blocs destination de la partition R. Elle est codée sur un nombre variable de bits puisque la mémorisation de l'ensemble des trois points n'est pas toujours nécessaire. Elle dépend de la transformation massique utilisée.

Calcul du débit

Le débit en bits par pixels est donné par l'expression suivante :

$$\frac{N_p I_c + N_s I_s + N_m I_m + N_e I_e}{N_p B^2}$$

où N_p désigne le nombre total de blocs parents destination et N_s, N_m et N_e dénotent respectivement les nombres de blocs (parents et enfants) homogènes, texturés et avec contours. Un bloc parent peut ne pas être divisé ou être divisé en un, deux, ou quatre blocs enfants. Les 12 configurations possibles, illustrées sur la figure 14, sont codées sur $I_c = 4$ bits.

Contrôle de la contraction de la transformation fractale

Le contrôle de la contraction de la transformation fractale est un problème difficile. Jacquin montre que le facteur de contraction d'une transformation massique dépend du facteur d'échelle $\beta_2^{(n)}$ et est égal à $(\beta_2^{(n)})^2$. Le facteur de contraction des autres transformations massiques (décalage, absorption) est égal à l'unité. Il montre aussi que le facteur de contraction de la transformation spatiale qui ramène un bloc source sur un bloc destination en moyennant les pixels est égal à un. L'auteur assure ainsi la contraction de la transformation fractale en imposant la condition $(\beta_2^{(n)})^2 < 1$ quel que soit n dans $[1, \ldots, N_s + N_m + N_e]$. Cette contrainte trop stricte limite la qualité des collages. Nous verrons dans la section 3.3.4 qu'il est possible de relâcher un peu la contrainte tout en préservant la contraction finale de la transformation fractale.

3.3.3 Formulation algébrique de la transformation fractale

Suite au travail de Jacquin, L. Lundheim a très vite proposé une formulation algébrique visant à faciliter la compréhension des différents problèmes théoriques et pratiques soulevés par l'extension de la théorie des IFS au codage des images naturelles [108]. Sa formulation appliquée au signal monodimensionnel a ensuite été étendue au cas des signaux bidimensionnels par Øien [128], et Lepsøy [100].
La formulation que nous donnons ici est utilisée dans le cas simple d'une transformation fractale opérant sur des blocs source et destination de taille fixe et de géométrie simple (carrée, rectangulaire ou triangulaire). Les blocs source ne se recouvrent pas. Un bloc est vu comme un vecteur en supposant que les pixels qui le composent sont connexes à l'intérieur de l'image originale.

Considérons une image comme un vecteur colonne $\mathbf{x}$ composé de M^2 pixels.
La transformation fractale T de l'image $\mathbf{x}$, composée d'un terme linéaire $\mathbf{L}$ et d'un vecteur de translation $\mathbf{t}$ s'exprime de la manière suivante :

$$T\mathbf{x} = \mathbf{L}\mathbf{x} + \mathbf{t}. \tag{22}$$

En spécifiant cette transformation au niveau de chacun des blocs destination de la partition R on peut écrire l'équation (22) sous la forme suivante :

$$T\mathbf{x} = \left(\sum_{n=1}^{N} \mathbf{L}_n\right)\mathbf{x} + \sum_{i=1}^{N} \mathbf{t}_n. \tag{23}$$

Nous détaillons ci-dessous les transformations élémentaires associées à chacune des matrices $\mathbf{L}_n$ et chacun des vecteurs $\mathbf{t}_n$.

La transformation associée à la matrice élémentaire $\mathbf{L}_n : \mathbb{R}^{M^2} \rightarrow \mathbb{R}^{M^2}$ opère sur le vecteur source $\mathbf{d}_{\alpha(n)}$ de façon à le coller sur le vecteur destination $\mathbf{r}_n$. Les vecteurs source et destination appartiennent respectivement à $\mathbb{R}^{D^2}$ et $\mathbb{R}^{B^2}$, où $D > B$.
$\hookrightarrow$ La matrice $\mathbf{L}_n$ s'écrit :

$$\mathbf{L}_n = \beta_2^{(n)} \mathbf{P_n} \underbrace{\mathbf{D}\mathbf{F}_{\alpha(\mathbf{n})}}_{\mathbf{b}_2^{(n)}} \tag{24}$$

où

1. $\mathbf{F}_{\alpha(\mathbf{n})} : \mathbb{R}^{M^2} \rightarrow \mathbb{R}^{D^2}$ sélectionne un bloc source $\mathbf{d}_{\alpha(n)}$ de taille D^2 pixels de l'image. Des isométries peuvent être effectuées à l'intérieur du bloc.

2. $\mathbf{D} : \mathbb{R}^{D^2} \rightarrow \mathbb{R}^{B^2}$ ramène le bloc source sélectionné à la taille d'un bloc destination par sous-échantillonnage ou en moyennant les pixels. Le bloc ainsi obtenu est appelé *bloc* $\mathbf{b}_2^{(n)}$. Il est à rapprocher du bloc source décimé décrit par Jacquin.

3. $\mathbf{P_n} : \mathbb{R}^{B^2} \rightarrow \mathbb{R}^{M^2}$ positionne le bloc source décimé $\mathbf{b}_2^{(n)}$ sur le bloc destination $\mathbf{r}_n$ et annule les autres pixels de l'image.

Un vecteur colonne $\mathbf{L}_n\mathbf{x}$ est essentiellement composé de zéros, à l'exception de la partie correspondant au bloc destination considéré, d'indice n. La matrice $\mathbf{L}$ donnée par la formule (25) est ainsi composée de sous-matrices associées à chacun des blocs $\mathbf{r}_n$ de l'image, et de zéros ailleurs.

$$\mathbf{L} = \begin{bmatrix} \cdots & \boxed{\beta_2^{(1)}\mathbf{D}} & \cdots \\ \cdots & \cdots & \boxed{\beta_2^{(n)}\mathbf{D}} \\ \boxed{\beta_2^{(N)}\mathbf{D}} & \cdots & \cdots \end{bmatrix} \tag{25}$$

La position verticale d'une sous-matrice $\beta_2^{(n)}\mathbf{D}$ dans la matrice $\mathbf{L}$ correspond à l'index n du bloc destination considéré. La position horizontale correspond à la position du bloc source qui lui est associé.
$\hookrightarrow$ Le vecteur élémentaire de translation $\mathbf{t}_n$ s'écrit :

$$\mathbf{t}_n = \beta_1^{(n)} \mathbf{P_n} \mathbf{b}_1^{(n)} \tag{26}$$

où $\beta_1^{(n)}$ est un coefficient réel. Le bloc constant $\mathbf{b}_1^{(n)}$, non issu de l'image $\mathbf{x}$, de taille B^2 pixels est composé de pixels égaux à un ($\mathbf{b}_1^{(n)} = [1, 1, \ldots, 1]^T$).
Le vecteur $\mathbf{t}$ opérant sur l'image entière s'écrit ainsi :

$$\mathbf{t} = [\beta_1^{(1)}, \beta_1^{(1)}, \ldots, \beta_1^{(1)},\ \beta_1^{(2)}, \beta_1^{(2)}, \ldots, \beta_1^{(2)},\ \beta_1^{(N)}, \beta_1^{(N)}, \ldots, \beta_1^{(N)}]^T \tag{27}$$

En résumé, la transformation fractale T de l'image $\mathbf{x}$ est donnée par l'expression suivante :

$$T\mathbf{x} = \left(\sum_{n=1}^{N} \beta_2^{(n)} \mathbf{P_n} \mathbf{b}_2^{(n)} \right) \mathbf{x} + \sum_{n=1}^{N} \beta_1^{(n)} \mathbf{P_n} \mathbf{b}_1^{(n)} \tag{28}$$

Formulation de la transformation massique

Nous omettrons dans la suite l'opérateur $\mathbf{P_n}$ ainsi que l'indice n dans les expressions de la transformation fractale T, dans le but de simplifier les notations.

Moyennant cette simplification d'écriture, l'approximation du bloc $\mathbf{r}$, notée $\hat{\mathbf{r}}$, est donnée par une combinaison linéaire de deux blocs, parmi lesquels le bloc $\mathbf{d}$ est extrait de l'image elle-même. Elle s'écrit :

$$\hat{\mathbf{r}} = \beta_2 \mathbf{b}_2 + \beta_1 \mathbf{b}_1 \tag{29}$$

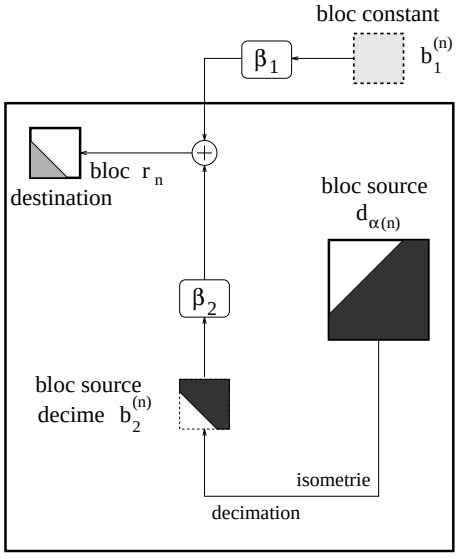


FIG. 15 - Transformation fractale de l'image.

La transformation ainsi réalisée est celle initialement proposée par A. Jacquin en 1989. Il est bien sûr possible de compliquer l'expression (29) dans le but d'améliorer

l'approximation du bloc destination. De manière plus générale, l'approximation de $\mathbf{r}$ s'exprime de la manière suivante :

$$\hat{\mathbf{r}} = \beta_2 \mathbf{b}_2 + \beta_1 \mathbf{b}_1 + \sum_{i=3}^{K} \beta_i \mathbf{b}_i \tag{30}$$

Les blocs $\mathbf{b}_i$ sont constants et connus du codeur et du décodeur. Ce type d'expression est notamment utilisé dans [120] et [69].

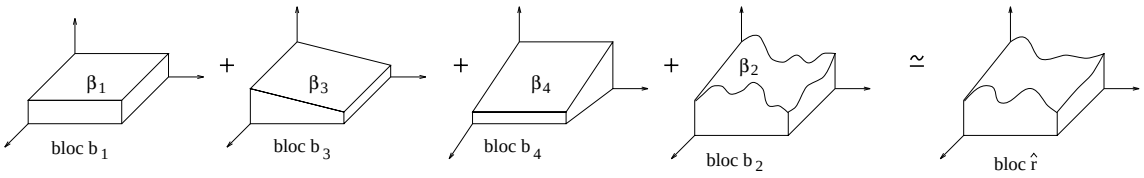


FIG. 16 - Ajout de vecteurs fixes pour l'approximation d'un bloc destination.

Vines [160] propose un autre schéma en travaillant sur des blocs carrés destination et source décimés, de taille égale à 8×8 pixels. Il construit une base orthonormale composée des 3 vecteurs fixes illustrés sur la figure 16, et de 61 autres vecteurs obtenus à partir des vecteurs source décimés de l'image. Un bloc destination est ensuite approximé par une combinaison linéaire de quelques vecteurs de la base. Le nombre de vecteurs considérés dépend de la complexité du bloc destination.

Contrôle de la contraction de la transformation fractale

Le facteur de Lipschitz s de l'opérateur affine T (équation 28) est égal à la norme de la matrice $\mathbf{L}$, et donc à la racine carrée de la plus grande valeur propre de $\mathbf{L}^T\mathbf{L}$ si l'on considère la norme L_2. Partant de cette remarque, Lundheim définit les conditions suffisantes qui assurent la contraction de l'opérateur T [108], en considérant que les blocs source ne se recouvrent pas :
Si les collages se font par sous-échantillonnage des blocs carrés source, alors s est donné par :

$$s = \sqrt{\max_{l=1:Q} \sum_{\alpha(n)=l} \left(\beta_2^{(n)}\right)^2} \tag{31}$$

où Q est le nombre de blocs source utilisés. La somme considère les coefficients d'échelle $\beta_2^{(n)}$ associés à l'ensemble des blocs destination $\mathbf{r}_n$ qui dépendent du bloc source $\mathbf{d}_{\alpha(n)}$. Si les collages se font en moyennant les pixels des blocs source, s est donné par :

$$s = \sqrt{\frac{B}{D} \max_{l=1:Q} \sum_{\alpha(n)=l} \left(\beta_2^{(n)}\right)^2} \tag{32}$$

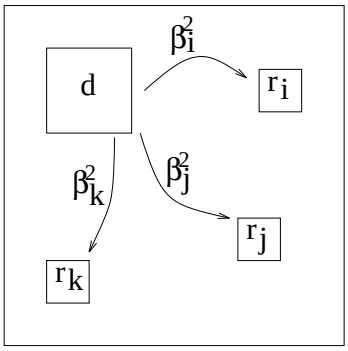


FIG. 17 - Illustration des équations précédentes : pour chacun des blocs source $\mathbf{d}_{\alpha(n)}$ on calcule la somme des coefficients d'échelle $\beta_2^{(n)}$ associés aux blocs destination $\mathbf{r}_n$ qui dépendent de $\mathbf{d}_{\alpha(n)}$.

L'équation (32) montre que le facteur de Lipschitz s de l'opérateur T est réduit d'un facteur $\frac{B}{D}$ lorsque les pixels des blocs source sont moyennés. Il dépend des coefficients d'échelle $\beta_2^{(n)}$ mais aussi de la différence de taille des blocs comparés. La "contraction spatiale" des blocs influe donc dans ce cas sur le facteur de contraction de T.

3.3.4 Méthode de Y. Fisher

Y. Fisher [60] décrit l'opération de collage d'un bloc source sur un bloc destination en utilisant une formule unique donnée par :

$$\omega_n \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_n & b_n & 0 \\ c_n & d_n & 0 \\ 0 & 0 & \beta_2^{(n)} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_n \\ f_n \\ \beta_1^{(n)} \end{pmatrix}. \tag{33}$$

où (x, y) sont les coordonnées d'un pixel intérieur au bloc source $\mathbf{d}_{\alpha(n)}$, et z est le niveau de gris du pixel. a_n, b_n, c_n, d_n, e_n et f_n sont les coefficients de la transformation spatiale affine ramenant les pixels du bloc source $\mathbf{d}_{\alpha(n)}$ à l'intérieur du bloc destination $\mathbf{r}_n$. $\beta_1^{(n)}$ et $\beta_2^{(n)}$ sont les coefficients de la transformation du niveau de gris des pixels.
Alors que les coefficients de la transformation massique utilisée par Jacquin sont choisis dans des ensembles prédéfinis de valeurs, Y. Fisher utilise un quantificateur scalaire uniforme. Jacobs et *al.* ont montré que pour ce type de quantificateur, la quantification des coefficients de translation et d'échelle, respectivement sur 7 et 5 bits, est optimale en terme de qualité visuelle des images reconstruites [83].

Calcul des coefficients optimaux de la transformation massique

Pour un bloc destination $\mathbf{r}$, l'approximation $\hat{\mathbf{r}}$ est donnée par :

$$\hat{\mathbf{r}} = \beta_2 \mathbf{b}_2 + \beta_1 \mathbf{b}_1$$

Le calcul des coefficients de la transformation "massique" est un problème d'optimisation dans un sous-espace linéaire X de l'espace vectoriel $\mathbb{R}^{B^2}$.
Le but est de trouver pour un bloc destination $\mathbf{r}$ et un bloc source décimé $\mathbf{b}_2$ les coefficients β_1 et β_2 optimaux minimisant la distance d au sens des moindres carrés entre $\mathbf{r}$ et son collage $\hat{\mathbf{r}}$.

> *Théorème de projection*: L'approximation optimale au sens de la norme L_2 d'un vecteur $\mathbf{r}$ élément de $\mathbb{R}^{B^2}$, dans le sous-espace linéaire X est le vecteur $\hat{\mathbf{r}}$ élément de X rendant le vecteur résiduel $\mathbf{r} - \hat{\mathbf{r}}$ orthogonal à tous les vecteurs engendrant le sous-espace X.

Soit le produit scalaire de deux vecteurs $\mathbf{x}$ et $\mathbf{y}$ sur l'espace $\mathbb{R}^{B^2}$ défini par l'expression suivante :

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{j=1}^{B^2} (x_j - y_j)^2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^{B^2}$$

Déterminer les 2 coefficients β_1 et β_2 optimaux permettant de trouver le vecteur $\hat{\mathbf{r}}$ approximant au mieux le vecteur $\mathbf{r}$ dans la base $\mathbf{b}_1, \mathbf{b}_2$ revient à annuler les deux produits scalaires :

$$\begin{cases} \langle \mathbf{r} - \beta_1 \mathbf{b}_1 - \beta_2 \mathbf{b}_2, \mathbf{b}_1 \rangle = 0 \\ \langle \mathbf{r} - \beta_1 \mathbf{b}_1 - \beta_2 \mathbf{b}_2, \mathbf{b}_2 \rangle = 0 \end{cases}$$

Les coefficients β_1 et β_2 optimaux ainsi calculés ne sont pas contraints, et la contraction de l'opérateur de collage n'est pas assurée. Jacobs et *al.* montrent que le fait de fixer un seuil égal à 1.5 sur le module du coefficient β_2 assure la convergence finale de la transformation fractale. Nous verrons plus précisément dans la section 6.2 les précautions à prendre lors du calcul des coefficients β_2 des opérateurs de collage.
Hürtgen propose une étude détaillée du contrôle de la contraction de la transformation fractale en considérant des cas particuliers fondés sur des partitionnements carrés [77] [79], et le rayon spectral associé au terme linéaire $\mathbf{L}$ introduit par Lundheim (eq (22)).

3.3.5 Méthode de F. Dudbridge

Dudbridge a proposé en 1992 [52] une méthode rapide de compression des images par fractales basée sur un partitionnement carré régulier. La rapidité de l'algorithme de compression est due au fait qu'aucune recherche de similarité inter-blocs n'est faite. L'image est partitionnée en un ensemble de blocs carrés, de taille fixe, puis chacun des blocs est codé individuellement par une transformation fractale. Au dire de l'auteur, la méthode donne de moins bons résultats que par exemple la méthode de Jacquin. En fin de section, nous en expliciterons les raisons.

> *Lorsque nous utiliserons dans la suite de cette section le terme d'image, cela désignera un bloc carré issu du partitionnement régulier de l'image originale.*

Codage

Une image est codée à l'aide d'un ensemble de transformations spatiales contractantes $\{\omega_1, \ldots, \omega_N\}$ définies sur $\mathbb{R}^2$ (IFS) auquel est associée une transformation contractante G agissant sur la luminance des pixels.
Le support carré de l'image transformée par l'IFS à la résolution m est donné par :

$$A = \bigcup_{k=1}^{N} \omega_k(A) = \bigcup_{k_1=1}^{N} \ldots \bigcup_{k_m=1}^{N} \underbrace{\omega_{k_1} \circ \ldots \circ \omega_{k_m}(A)}_{A_{k_1 \ldots k_m}} \tag{34}$$

Nous remarquons ici que la transformation spatiale est appliquée à A et non à une sous-partie de A comme c'est le cas dans l'approche classique de codage définie par Jacquin. $p = A_{k_1 \ldots k_m}$ dénote un "élément" du support de l'image à la résolution m, pouvant contenir plusieurs pixels de l'image originale. Lorsque la résolution m est maximale, la taille de l'élément p est égale à celle d'un pixel de l'image. L'ensemble $P_m = \{A_{k_1 \ldots k_m}; k_1, \ldots, k_m = 1, \ldots, N\}$ contient tous les éléments de l'image à la résolution m.

Nous considérerons dans la suite de cette section que l'IFS est composé de $N = 4$ transformations affines données par les équations (14). L'équation (34) est dans ce cas illustrée sur la figure 18.

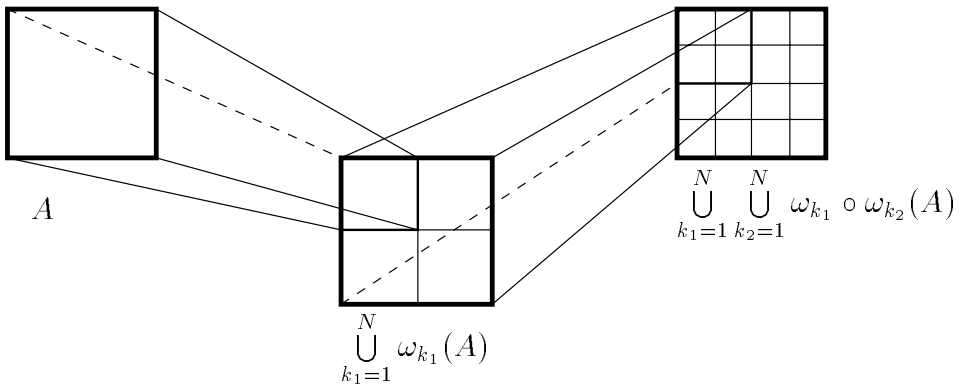


FIG. 18 - L'image carrée A est divisée en quatre éléments carrés par quatre transformations affines contractantes ω_{k_1} $(k_1 = 1 \ldots 4)$. Au centre, $A_{k_1} = \omega_{k_1}(A)$ correspond à l'un des quatre éléments de l'image à la résolution 1. A droite, $A_{k_1 k_2} = \omega_{k_1} \circ \omega_{k_2}(A)$ correspond à l'un des seize éléments de l'image à la résolution 2.

La transformation G est donnée par l'équation suivante [52] :

$$Gf(p) = \int_p (a_{k_1} x + b_{k_1} y + t_{k_1})\, dxdy + s_{k_1} v(p) \tag{35}$$

dans laquelle la fonction $f : P_m \to \mathbb{R}$ retourne le niveau de gris de l'élément p, et $v(p)$ est la somme des niveaux de gris des éléments inclus dans le bloc $\omega_{k_1}^{-1}(p)$

(figure 19) :

$$v(p) = \sum_{i=1}^{N} f(A_{k_2 \ldots k_m i})$$

Contrairement à la transformation massique de Jacquin qui ne contient qu'un facteur d'échelle et un facteur de décalage sur les niveaux de gris, l'équation (35) contient deux coefficients supplémentaires, liés à la position (x, y) dans l'image de l'élément à approximer.

Dudbridge montre que la transformation G est finalement contractante à toutes les résolutions m si $\left| \sum_{k_1=1}^{N} s_{k_1} \right|$ est inférieur à 1, en considérant la distance Euclidienne.

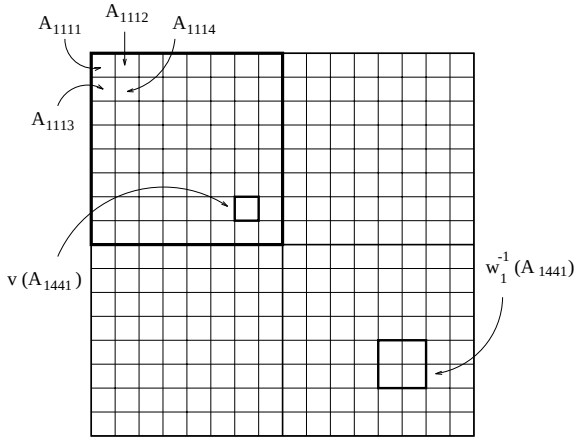


FIG. 19 - Exemple pour $k_1 = 1$ (quadrant supérieur gauche), $m = 4$ et $N = 4$: $v(A_{1441}) = \sum_{i=1}^{4} f(A_{441i})$. Dans ce cas particulier, l'élément $p = A_{1441}$ atteint la taille d'un pixel de l'image.

Le calcul des coefficients a_{k_1}, b_{k_1}, t_{k_1} et s_{k_1} pour tout k_1 dans l'ensemble $[1 \ldots N]$ est fait de manière à minimiser l'erreur au sens des moindres carrés à la résolution m entre l'image originale f et sa transformée par G. De cette façon, le théorème du collage peut être vérifié. Il suffit pour cela de minimiser la fonction suivante, pour tout k_1 dans l'ensemble $[1 \ldots N]$:

$$\sum_{p \in \omega_{k_1}(P_{m-1})} \left[\int_p (a_{k_1} x + b_{k_1} y + t_{k_1}) \, dxdy + s_{k_1} v(p) - f(p) \right]^2 \qquad (36)$$

Les N sommations sont faites sur le sous-bloc k_1 de l'image originale, noté $\omega_{k_1}(P_{m-1})$ (nous rappelons que celui-ci est l'un des quatre quadrants de l'image originale). La somme $v(p)$ dépend de la résolution m à laquelle est approximée la luminance de l'élément p.

La minimisation de la fonction (36) consiste à résoudre le système d'équations suivant [52] :

$$\begin{bmatrix} \sum_p \left(\int_p x\right)^2 & \sum_p \int_p x \int_p y & \sum_p \int_p x \int_p 1 & \sum_p v(p) \int_p x \\ \sum_p \int_p x \int_p y & \sum_p \left(\int_p y\right)^2 & \sum_p \int_p y \int_p 1 & \sum_p v(p) \int_p y \\ \sum_p \int_p x \int_p 1 & \sum_p \int_p y \int_p 1 & \sum_p \left(\int_p 1\right)^2 & \sum_p v(p) \int_p 1 \\ \sum_p v(p) \int_p x & \sum_p v(p) \int_p y & \sum_p v(p) \int_p 1 & \sum_p (v(p))^2 \end{bmatrix} \begin{bmatrix} a_{k_1} \\ b_{k_1} \\ t_{k_1} \\ s_{k_1} \end{bmatrix} = \begin{bmatrix} \sum_p f(p) \int_p x \\ \sum_p f(p) \int_p y \\ \sum_p f(p) \int_p 1 \\ \sum_p f(p) v(p) \end{bmatrix}$$

Une image (un bloc carré de la partition de l'image originale) est ainsi codée par une suite de 4×4 coefficients réels.

Décodage

L'algorithme de décodage reconstruit la fonction invariante g associée à l'opérateur G de manière très rapide, et non itérative, connaissant les coefficients a_{k_1}, b_{k_1}, t_{k_1} et s_{k_1} associés à chacune des N transformations spatiales ω_{k_1}.

F. Dudbridge montre que la somme, notée g_{k_1}, des niveaux de gris des sous-éléments inclus dans l'élément A_{k_1} est donnée par l'équation suivante [52] [118]:

$$g_{k_1} = a_{k_1} \int_{A_{k_1}} x \, dxdy \; + \; b_{k_1} \int_{A_{k_1}} y \, dxdy \; + \; t_{k_1} \int_{A_{k_1}} 1 \, dxdy \; + \; s_{k_1} \sum_{k=1}^{N} g_k \tag{37}$$

et que par conséquent, la somme des niveaux de gris des N éléments A_{k_1} est donnée par :

$$\sum_{k=1}^{N} g_k = \frac{\sum_{k=1}^{N} \left(a_k \int_{A_k} x + b_k \int_{A_k} y + t_k \int_{A_k} 1\right)}{1 - \sum_{k=1}^{N} s_k} \tag{38}$$

De la même manière, $g_{k_1k_2}$ désigne la somme des niveaux de gris des sous-éléments de $A_{k_1k_2}$ et est donné par :

$$g_{k_1k_2} = a_{k_1} \int_{A_{k_1k_2}} x \, dxdy \; + \; b_{k_2} \int_{A_{k_1k_2}} y \, dxdy \; + \; t_{k_2} \int_{A_{k_1k_2}} 1 \, dxdy \; + \; s_{k_2} \sum_{k=1}^{N} g_{k_2k} \tag{39}$$

$$\text{avec} \qquad \sum_{k=1}^{N} g_{k_2k} = g_{k_2}$$

La procédure de décodage se résume ainsi :

- $\sum_{k=1}^{N} g_k$ est directement calculé à partir des coefficients de G (formule 38). Le résultat est égal à la somme des niveaux de gris des pixels de l'image originale.

- d'après (37), g_{k_1} $(k_1 = 1 \ldots N)$ est fonction des variables a_{k_1}, b_{k_1}, t_{k_1}, s_{k_1} et de la valeur $\sum_{k=1}^{N} g_k$ précédemment calculée

- d'après (39), $g_{k_1 k_2}$ est fonction des variables a_{k_2}, b_{k_2}, t_{k_2}, s_{k_2} et de g_{k_2} précédemment calculé.

- etc ...

Chacun des éléments de la fonction invariante g à la résolution m peut de cette manière être calculé récursivement. La procédure de reconstruction n'est pas itérative, contrairement à la plupart des algorithmes de décodage par fractales.

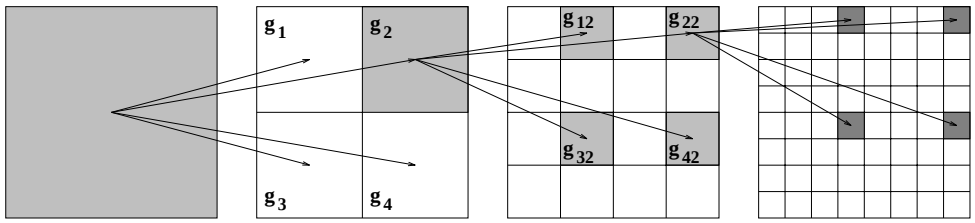


FIG. 20 - Décodage à la résolution $m = 3$ d'une image 8×8 selon l'algorithme de Dudbridge. Les quatre valeurs $g_{k_1 k_2}$ $(k_1 = 1 \ldots 4)$ à la résolution 2 dépendent de la valeur g_{k_2} à la résolution 1 et de leur position dans l'image.

Discussions sur la méthode de Dudbridge

La méthode présentée dans cette section consiste à approximer la fonction de luminance f dans chacun des blocs carrés d'une partition de l'image originale. Un bloc est approximé par une fonction invariante g de manière complètement indépendante du reste de l'image. L'approximation, à une résolution donnée m, se fait au sens des moindres carrés à l'aide d'un IFS associé à une transformation finalement contractante G dans l'espace des luminances. L'expression de G (équation (35)) est comparable à celle de la transformation massique proposée par Jacquin puisqu'elle est également composée d'un facteur d'échelle s_{k_1} et d'un facteur de décalage t_{k_1}. Elle contient deux coefficients supplémentaires a_{k_1} et b_{k_1}. Ces derniers agissent sur les coordonnées des éléments approximés à l'intérieur du bloc : l'approximation est dans ce cas faite à l'ordre un. L'équation (35) est à rapprocher de l'équation (30) : les coefficients a_{k_1} et b_{k_1} pondèrent deux plans inclinés dans l'espace des luminances.

La raison de "l'infériorité" de cette méthode par rapport au schéma de base de Jacquin vient du fait qu'un bloc n'est approximé qu'à partir de lui-même et non pas à partir d'un autre bloc de l'image. La transformation G doit dans cas être suffisamment complexe pour permettre une bonne approximation du bloc. Dudbridge a pour cela rajouté deux paramètres à l'expression de la transformation massique. La mémorisation de ces derniers a pour effet de réduire le taux de compression.

La méthode a cependant l'avantage d'être très rapide. L'algorithme de codage-décodage est de plus symétrique au niveau des temps de calculs.

Dudbridge [50] avait initialement présenté cette méthode de codage dans sa thèse en 1992. L'approche est actuellement généralisée sur des blocs carrés issus d'un partitionnement quadtree [51].
L'ajout de coefficients supplémentaires dans l'expression de G pondérant des termes en x^2, y^2, x^3 et y^3 est étudié par Monro et *al.* [116] [117] [120] [165]. Les auteurs généralisent aussi cette approche à la compression des séquences vidéos [164].

3.4 Extensions

Le travail initial de Jacquin, basé sur l'utilisation d'un système de fonctions itérées *locales* et contractantes, a donné lieu au démarrage de nombreuses autres recherches sur la compression des signaux réels 1D, 2D et 3D par fractales [148] [86] [60]. Nous avons décrit dans ce chapitre quelques méthodes de base. Quelques méthodes supplémentaires de compression des images naturelles par fractales seront détaillées dans le chapitre 6.
Celles-ci concernent principalement :

- la construction d'une partition optimale pour calculer la transformation fractale [61] [64] [47] [80] [137] [126] [156];
- l'accélération de l'étape de codage [100] [50] [41];
- l'introduction de vecteurs fixes pour approximer les blocs destination à partir des blocs source [160] [69];
- l'utilisation de fonctions élémentaires ω_n non-affines permettant de coder la redondance spatiale de l'image [105];
- l'accélération du décodage : itératif, non-itératif, hiérarchique [128] [6] [129];
- l'étude théorique de la convergence du décodeur [108] [77] [79];
- l'extension de la méthode au codage des images vidéo [20] [76] [65] [78] [24] [18] [99] [119] [164];
- l'utilisation des fractales dans les schémas de codage-décodage hybrides [17] [16] [15] [153] [96] [40] [139] [158] [159].

3.5 Conclusion

Nous avons introduit dans ce chapitre les bases mathématiques nécessaires à la compréhension de la théorie des systèmes de fonctions itérées. Nous avons dans une

deuxième partie présenté les principaux algorithmes de compression des images naturelles selon l'approche fractale fondée sur des blocs de pixels.

Peu de recherches ont été faites jusqu'à aujourd'hui en ce qui concerne le modèle de partitionnement à utiliser pour la compression des images par fractales. Les auteurs optimisent le calcul de la transformation fractale sur un partitionnement carré régulier, ou en arbre quaternaire (quadtree). La qualité visuelle de l'image reconstruite a été sensiblement améliorée par Fisher en proposant un partitionnement rectangulaire adapté au contenu de l'image. Nous verrons dans le prochain chapitre que ce dernier a l'avantage par rapport aux deux autres partitionnements carrés de contenir moins de blocs.
C'est précisément sur ce problème du choix du modèle de partitionnement que se situe notre travail et par conséquent l'apport personnel. Dans le cadre de cette thèse une nouvelle approche basée sur un partitionnement triangulaire irrégulier est proposée. De la même manière que Fisher, notre but est de minimiser le nombre de blocs au sein de la partition en l'adaptant au contenu de l'image. Le principal avantage de la triangulation par rapport au partitionnenent rectangulaire est de pouvoir fournir des blocs d'orientation quelconque au dessus des contours et des régions texturées, ou le long des contours. Nous montrerons aussi que notre méthode de codage diffère du schéma classique dans le sens où les recherches de similarités se font au travers d'un seul partitionnement régulier. Cette solution pratique est préférée à l'autre solution qui consiste à faire une recherche exhaustive parmi des blocs source disposés sur un réseau régulier et dense de points.

Chapitre 4

Partitionnements de l'image

4.1 Introduction

Ce chapitre a pour objectif de proposer différents modèles géométriques de partitionnement d'images pour la compression par fractales. Nous distinguerons trois classes de partitionnements :

1. les partitionnements rigides (Section 4.2);
2. les partitionnements semi-rigides (Section 4.3);
3. les partitionnements souples (Section 4.4).

4.1.1 Utilité du partitionnement géométrique des images

Les modèles de partitionnement sont des outils qui peuvent être utiles en analyse et en traitement d'images, notamment en segmentation, en analyse quantitative, en codage, en modélisation et en restauration. Nous présentons ci-après quelques utilisations particulières.

Segmentation

Sur la base d'une représentation en termes de pixels de l'image, la segmentation a pour objectif de fournir une partition de l'image sous forme de régions ou de contours. Nous rappelons dans cette partie les principales techniques de segmentation d'images, qui retournent des régions de forme quelconque, directement calculées à partir de la surface des niveaux de gris de l'image. On peut également envisager que les régions soient de forme contrainte par le modèle de partitionnement géométrique considéré. Nous verrons que ce type de régions peut être obtenu selon une approche par division et fusion.

La segmentation d'une image A est un traitement bas niveau s'appuyant sur une phase de détection et une phase de mise en correspondance. Il existe principalement deux approches pour segmenter une image, qui sont respectivement les approches régions et les approches contours.

Les **approches contours** regroupent un grand nombre de méthodes. Leur but est de détecter les discontinuités dans la surface des niveaux de gris de l'image, de manière à former des contours fermés sur lesquels s'appuient les régions de l'image. Il existe diverses méthodes de détection de contours qui sont la plupart du temps basées sur des opérateurs de filtrage différentiel. Elles sont appelées *méthodes dérivatives*. Les opérateurs recherchent les extremas locaux de la norme du gradient de la fonction des niveaux de gris, les passages par zéro de la dérivée seconde, ou bien encore les zones présentant une forte corrélation avec un profil prédéterminé. L'inconvénient du calcul du gradient en un point de l'image est qu'il fait appel à la première dérivée partielle du signal de luminance. Il est donc sensible au bruit présent dans l'image. Pour cette raison, la plupart des méthodes de détection de contours passent par

une étape de prétraitement consistant à diminuer le bruit par une opération de filtrage [25].
Canny [30] a défini un opérateur de filtrage optimal, satisfaisant les trois contraintes suivantes, pour une entrée en échelon :

- bonne détection des contours;
- localisation précise des contours;
- faible multiplicité des réponses de l'opérateur, dues au bruit.

Deriche [48] part également d'un opérateur optimal tendant à vérifier les trois contraintes définies par Canny et calcule le gradient en chaque point de l'image à partir des dérivées selon les directions m et n de l'image originale lissée. Le filtre séparable bidimensionnel de lissage, $f(m,n)$, est donné par l'expression suivante :

$$f(m,n) = k(\alpha|m|+1)e^{-\alpha|m|}k(\alpha|n|+1)e^{-\alpha|n|},$$

où k est une constante. Les dérivées directionnelles suivant les directions x et y de l'image lissée sont calculées en convoluant directement l'image originale $A(m,n)$ par les filtres directionnels suivants :

$$f_x(m,n) = k'me^{-\alpha|m|}k(\alpha|n|+1)e^{-\alpha|n|}, \tag{40}$$

$$f_y(m,n) = k(\alpha|m|+1)e^{-\alpha|m|}k'ne^{-\alpha|n|}, \tag{41}$$

où k et k' sont des constantes de normalisation qui dépendent de α. Il est à noter que le paramètre α intervient aussi bien sur la phase de lissage (quantité de bruit supprimé) que sur la phase de dérivation. L'opération de convolution de l'image $A(m,n)$ avec l'un des deux filtres directionnels (40) et (41) est mise en œuvre de manière récursive, pour limiter le nombre de calculs par pixel. L'algorithme est entièrement détaillé dans [48]. Après calcul de la norme du gradient partant des images de dérivées directionnelles $A_x(m,n)$ et $A_y(m,n)$, l'extraction des contours se fait par la recherche des extrémas locaux de la norme du gradient, dans la direction du gradient. Cette étape est suivie d'un seuillage par hystérésis pour éliminer les artefacts provoqués par les contours entrecoupés.
Nous serons amenés dans la section 4.4.3 (page 97) à faire appel à cette méthode d'extraction d'attributs de type contours pour contraindre le positionnement de blocs triangulaires le long des frontières de l'image.

Les **approches régions** sont basées sur la définition formelle de la segmentation d'une image relative à un prédicat d'homogénéité [73] [169] :
Soit un prédicat d'homogénéité, appliqué sur un ensemble connexe de pixels composant une région R_i d'une image A. Alors une segmentation de A en régions R_i, relativement au prédicat P vérifie :

1. $\forall i, R_i \neq \emptyset$;

2. $A = \cup R_i$

3. $\forall i, j$ et $i \neq j, R_i \cap R_j = \emptyset$;

4. $\forall i, P(R_i) = vrai$;

5. $\forall i \neq j,$ et R_i; adjacent à $R_j, P(R_i \cup R_j) = faux$.

Le prédicat P est fonction d'un attribut scalaire ou vectoriel associé à une région R_i. Les attributs considérés peuvent être de natures très différentes. En voici quelques exemples :

- la moyenne, la variance des niveaux de gris, le contraste;
- les dimensions fractales et fractales généralisées [111] [103] [8], la lacunarité [101];
- des attributs géométriques tels que l'aire, le périmètre, la compacité, l'orientation, ou d'autres;
- des attributs de surface, en termes de plateau, bosse, creux, vallée, selle, ou crête.

Nous présentons ici rapidement certaines méthodes de segmentation par région. Le lecteur interessé pourra se référer à l'ouvrage collectif impulsé par le Groupe De Recherche 134 en Traitement Du Signal et Images du CNRS [34].

Segmenter une image selon une *approche par croissance de régions* consiste à initialiser une partition de l'image en régions de base et à regrouper itérativement celles qui possèdent des attributs scalaires ou vectoriels semblables, jusqu'à ce qu'il n'y ait plus de regroupements possibles. Les régions de départ peuvent être réduites à un seul pixel. La procédure itérative de regroupement est modélisée au moyen d'un graphe d'adjacence des régions. Un graphe d'adjacence est composé de nœuds et d'arêtes. Chaque arête connecte les nœuds associés à des régions adjacentes dans l'image. A chaque arête est associée une mesure de dissimilarité entre les deux nœuds qu'il relie. Le principe de regroupement de deux régions, au travers du graphe, est de rechercher l'arête minimale du graphe, de réunir les deux nœuds (régions) que ce lien connecte, et de recalculer les liens qui connectent le nouveau nœud au reste du graphe.
Une autre technique consiste, à partir d'un ensemencement de quelques pixels (régions) sur le support de l'image, à faire croître chaque région pixel par pixel selon un critère d'homonénéité fixé. Avec cette dernière approche, on peut ne pas avoir de partitionnement complet du support de l'image. L'ensemencement de départ peut se faire par exemple au niveau d'un objet recherché. Les points non affectés à des régions appartiennent alors à un ensemble appelé classe "fond".

L'*approche par division et fusion* connue sous le terme anglais de *split and merge* est

associée à un modèle de partitionnement de l'image. La segmentation d'une image selon cette approche se décompose en deux phases successives, appelées phase de division et phase de fusion, opérant sur une structuration en blocs de l'image. La phase de *division* est généralement itérative. Un bloc R_i est divisé selon une mesure de non homogénéité $E(R_i)$ comparée à un seuil T_1 comme l'indique la relation suivante :

$$P(R_i) = \begin{cases} vrai \text{ si } E(R_i) < T_1 \\ faux \text{ sinon} \end{cases}$$

Si le bloc R_i est reconnu homogène, il est laissé inchangé. Sinon, il est divisé de manière à restituer des sous-blocs plus homogènes. La *fusion*, ou le rassemblement des régions, pallie au point faible de la phase de division qui ne réunit pas les régions adjacentes identiques. Le but de la phase de fusion est de regrouper ces régions à condition que la mesure d'inhomogénéité provoquée par le regroupement reste au-dessous d'un seuil T_2. Nous présentons dans la section 4.2.1 le partitionnement en quadtree donnant une représentation pyramidale de l'image. Ce modèle simple et classique de partitionnement géométrique est bien adapté aux méthodes de segmentation par division et fusion. Nous montrerons dans la suite de la thèse comment l'utiliser pour calculer la transformation fractale d'une image. L'algorithme de division et fusion sera aussi appliqué au cas de la triangulation de Delaunay afin d'initialiser la phase de compression par fractales.

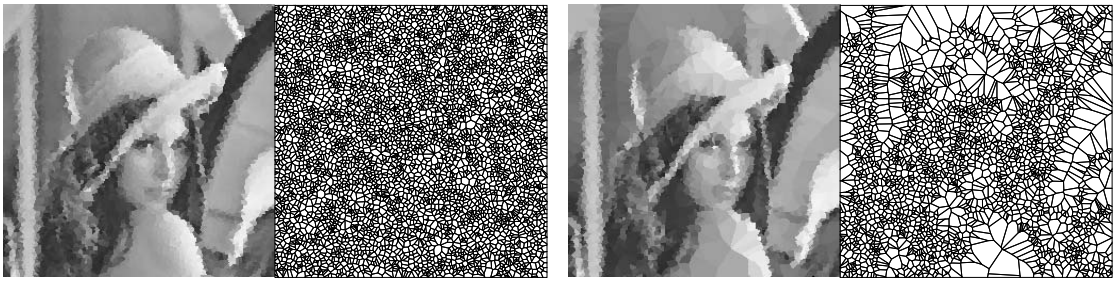
Codage

Le codage d'une image à l'aide de partitionnements consiste à approcher la surface des niveaux de gris de l'image dans chacun des blocs de la partition au moyen de fonctions bidimensionnelles. L'objectif est dans ce cas de construire une partition dont chacun des blocs vérifie un prédicat donné.
Lorsque le but est de compresser l'image, un compromis doit être trouvé entre, d'une part la complexité du modèle de partitionnement et d'autre part celle du codage. La complexité du partitionnement est liée à la géométrie des blocs et aux relations de voisinages entre les blocs. La complexité du codage est liée aux fonctions d'approximation utilisées pour représenter le contenu des éléments de la partition.

Une solution classique consiste à calculer les paramètres de fonctions quadratiques de la forme $\hat{f}(x, y) = a + bx + cy + dxy$, approximant la surface des niveaux de gris de l'image dans chacun des blocs d'un quadtree [22]. Un bloc est déclaré homogène si la surface de celui-ci peut être approximée par une fonction quadratique, au sens de la distance L_2. Une fois que le quadtree est calculé et codé, la réduction de l'information résulte du fait que chaque bloc de pixels est codé par les quatre coefficients réels a_i, b_i, c_i et d_i.

Une autre solution consiste à calculer la partition de Voronoï d'un ensemble dense de points positionnés aléatoirement sur le support de l'image selon un processus de

Poisson initialisé sur un point quelconque [1]. A l'issue du partitionnement de Voronoï, Ahuja et al. obtiennent des blocs de petite taille et distribués uniformément sur le support de l'image. Les blocs homogènes ayant des voisins directs également homogènes et de même valeur moyenne sont dans une deuxième phase déclarés inutiles et leurs germes sont effacés. La nouvelle partition est ainsi adaptée au contenu de l'image (figure 21). Elle est telle que la taille des blocs est localement maximale : des blocs de grande taille recouvrent les zones homogènes de l'image, et des blocs de petite taille recouvrent les contours. La compression de l'image est réalisée en ne mémorisant que la règle de construction de la partition et le niveau de gris moyen de chaque bloc. Le décodeur restitue la partition en ne connaissant que l'intensité λ et le point d'initialisation du processus de Poisson ainsi que le drapeau indiquant si les germes générés sont à conserver ou non. L'inconvénient d'une telle approche de codage est que les contours contrastés de l'image sont mal restitués puisqu'ils sont approximés par une suite de polygones homogènes et compacts.



(a) Partition d'un ensemble de 5000 points aléatoires

(b) Fusion

FIG. 21 - Partitionnements de Voronoï et approximation du contenu des éléments par leur valeur moyenne.

Nous verrons dans le chapitre 5 que le codage par fractales calculé sur un partitionnement triangulaire (adapté de façon similaire au contenu de l'image et composé du même nombre de blocs) permet la reconstruction de contours beaucoup plus nets puisque le contenu des blocs n'est dans ce cas pas homogène.

4.1.2 Rôle du partitionnement pour la compression par fractales

Le but de la compression par fractales est de "saisir" la redondance visuelle *locale* à l'intérieur de l'image à l'aide de transformations contractantes. La transformation fractale est directement calculée sur une partition R de l'image, dont les propriétés recherchées sont les suivantes :

- la partition R doit être adaptée à l'image, de façon à minimiser le nombre de blocs;

- la localisation et la "manipulation informatique" des blocs dans la partition doit être facile;

- l'information nécessaire à son codage doit être minimale.

Chacun des blocs destination $\mathbf{r}_i$ de la partition R est mis en correspondance avec une région source $\mathbf{d}_{\alpha(i)}$ de l'image, lui étant proche au sens des moindres carrés dans l'espace des niveaux de gris (ceci est détaillé dans la section 3.3).
La géometrie de la région $\mathbf{d}_{\alpha(i)}$ *doit être proche de celle du bloc* $\mathbf{r}_i$, de façon à limiter les temps de calcul lors des comparaisons inter-blocs, puisque ceux-ci sont liés à la complexité des transformations spatiales utilisées. Par exemple, le contenu d'un bloc destination carré ne sera pas comparé à celui d'une région source polygonale ou ayant des frontières arrondies puisque dans ce cas la transformation spatiale mettant en correspondance les pixels de chacun des deux blocs est trop complexe. Dans le schéma de base défini par Jacquin, les deux formes (carrées) sont liées par une transformation affine: un bloc destination de la partition R est mis en correspondance avec un bloc source inclus dans une librairie que l'on appellera librairie D. Nous utilisons dans la section 5 des blocs source et destination triangulaires qui peuvent de la même manière être déformés à l'aide de transformations affines. Nous montrons dans la section 6.4 (page 141) que la combinaison de triangles et de quadrilatères au sein d'une même partition complique les transformations spatiales utilisées pour comparer le contenu des blocs, mais permet cependant de conserver une qualité de reconstruction convenable à forts taux de compression.
Une contrainte supplémentaire impose que les *blocs source soient en moyenne de surface supérieure à celle des blocs destination*. De cette manière, un certain nombre de transformations spatiales contractantes sont mises en jeu et assurent la contraction finale de la transformation fractale.

Pour que le codage par fractales soit efficace, il doit y avoir suffisamment de *similarités locales à diverses échelles* entre les blocs de la partition R et les blocs source $\mathbf{d}_{\alpha(i)}$. Lorsque les blocs $\mathbf{d}_{\alpha(i)}$ sont recherchés dans toute l'image, la condition est généralement vérifiée, au moins dans le cas des images naturelles (figure 22). Si la recherche se fait dans une partition contenant un nombre restreint de blocs, le codage n'est plus optimal, mais plus rapide. Le but est de trouver le meilleur compromis entre le nombre de blocs $\mathbf{d}_{\alpha(i)}$ disponibles pour les recherches de similarités, et la ressemblance entre les blocs de la partition R et les blocs $\mathbf{d}_{\alpha(i)}$.

Dans la suite de ce chapitre, nous présenterons des partitionnements adaptatifs rigides (quadtree) et semi-rigides (horizontal-vertical) ainsi que leur utilisation pour la compression des images par fractales. Les partitionnements souples (Voronoï et Delaunay) sont présentés dans la section 4.4. Leur application à la compression par fractales sera décrite dans le chapitre 5.

FIG. 22 - A gauche : image présentant des similarités locales à diverses échelles. A droite : image auto-similaire, ne pouvant cependant pas être compressée par transformation fractale puisqu'aucune zone de l'image ne ressemble à une autre zone de taille supérieure.

4.2 Partitionnement rigide

Un partitionnement est qualifié de rigide s'il ne s'adapte pas à moindre coût aux formes des objets présents dans l'image. Au travers de tels partitionnements, tous les blocs ont la même géométrie à un facteur d'échelle près. Par rapport à un partitionnement fondé sur des blocs identiques (carrés de taille 8×8 par exemple), la notion d'échelle permet l'adaptivité au contenu de l'image : c'est le cas du quadtree.

4.2.1 Quadtree

Le quadtree ou arbre quaternaire est une structure initialement utilisée pour représenter des images binaires [145] [92]. La représentation est exacte lorsque le processus récursif de sous-division des blocs carrés descend jusqu'à la taille du pixel. Chaque bloc est alors composé de valeurs toutes égales à 1, ou toutes égales à 0.
Le quadtree peut aussi être utilisé pour la représentation des images en niveaux de gris. C'est ce dernier cas que nous considérerons dans la suite de cette section. Des études détaillées du quadtree sont données dans [146] [141].

Phase de division

Considérons une image initiale de taille $2^m \times 2^m$, que l'on note ${}^0A$. La construction du haut vers le bas (top-down) du quadtree consiste à diviser récursivement tout bloc ${}^lA_i$ non homogène selon un prédicat donné, en considérant le bloc ${}^0A$ comme une seule région de départ. l est le niveau de la représentation pyramidale sous-jacente au codage du quadtree. L'indice i ($i = 1, \ldots, 4$) dénote le numéro du sous-bloc. La

division d'un bloc ${}^{l}A_i$ de taille $2^{m-l} \times 2^{m-l}$ crée quatre sous-blocs carrés ${}^{l+1}A_j$ ($j = 1, \ldots, 4$) de dimension 2^{m-l-1}. A chaque nouvelle division, les attributs des quatre blocs créés sont recalculés pour être à nouveau soumis au prédicat d'homogénéité.

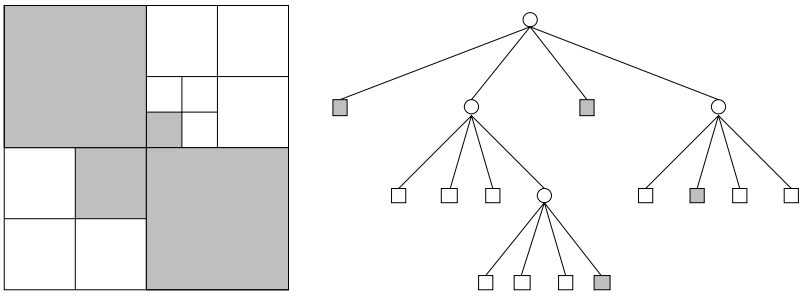


FIG. 23 - A gauche : principe de la division récursive d'une image binaire. A droite : representation arborescente du quadtree calculé sur une image de taille 8×8 pixels. Les cercles sont appelés *sommets* de l'arbre quaternaire et les carrés gris et blancs sont appelés *feuilles*.

Phase de fusion

La phase de fusion est utilisée en segmentation d'images. Elle consiste à regrouper les blocs adjacents égaux selon le prédicat d'homogénéité considéré. Il est à noter qu'à l'issue de celle-ci, la structure de l'arbre quaternaire est perdue. L'intérêt de cette étape est de réduire le nombre total de régions. Les bords des régions sont aussi beaucoup plus proches des contours réels de l'image.
L'étape de fusion n'est cependant pas utilisée dans le cas de la compression par fractales, car les blocs obtenus sont de formes trop complexes (voir section 4.1.2).

Implémentation du Quadtree

La construction récursive du quadtree n'est pas une solution optimale en terme de temps de calculs puisque chaque pixel est visité un nombre de fois égal à sa profondeur finale dans l'arborescence. Une méthode plus optimale consiste à utiliser la courbe de Peano en $\mathbf{Z}$ (ordre de Morton).
Considérons une image de taille $2^m \times 2^m$ dont les pixels sont numérotés dans un ordre croissant, en base quatre, selon la courbe de Peano (figure 24). Un pixel de l'image est numéroté sur m chiffres. Le chiffre de poids le plus significatif code un des quatre quadrants du niveau 1 dans l'arborescence du quadtree. La suite des chiffres d'un même numéro permet d'atteindre directement un pixel en passant par chacun des quadrants dont il est issu. Par exemple, sur la figure 23 le pixel gris dans le dernier niveau (niveau trois) de l'arborescence porte le numéro 122. Remarquons aussi que tous les pixels d'un même quadrant dans l'arborescence sont numérotés consécutivement. Cette dernière propriété est importante pour calculer rapidement les feuilles d'un quadtree. Il suffit de parcourir une seule fois les pixels dans l'ordre de Morton pour extraire les quadrants vérifiant le critère d'homogénéité fixé : le quadtree est construit à partir de la base, par fusion des quadtrees partiels

construits lors du parcours des pixels.

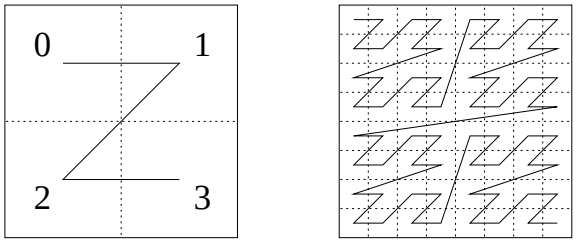


FIG. 24 - A gauche : courbe de Peano parcourant 4 blocs. A droite : courbe parcourant tous les pixels d'une image de taille 8×8.

Propriétés du Quadtree

La représentation d'une image à l'aide d'un quadtree est à résolution variable dans le sens où la partition exhibe des blocs de tailles différentes. Le partitionnement obtenu est rigide puisqu'il est guidé par le processus de découpage récursif en blocs carrés. Il n'est pas adapté aux formes des objets présents dans l'image, même s'il s'adapte au contenu de celle-ci. Le partitionnement contient un nombre important de blocs, si on le compare aux partitionnements de Voronoï et de Delaunay (voir section 5.6).
Il n'est pas robuste en rotation et en translation. Ceci peut être montré par l'exemple suivant : considérons un bloc carré, noir, de taille 2×2 pixels, centré sur un fond blanc de taille $2^m \times 2^m$. La représentation en quadtree d'une telle image contient $16m - 11$ feuilles et sommets. Le fait de décaler le carré noir central de un pixel horizontalement modifie sensiblement la structure du quadtree qui ne contient plus que $8m - 3$ points, soit de l'ordre de deux fois moins de points.

4.2.2 Utilisation du quadtree pour la compression par fractales

Le quadtree est largement utilisé dans la littérature [61] [21] [130] [80] pour la compression des images selon une approche fractale.
La transformation fractale est dans ce cas calculée au fur et à mesure de la construction du partitionnement. Nous détaillerons dans ce paragraphe la procédure de codage proposée par Y. Fisher [61].

Procédure de codage :
La profondeur du quadtree est fixée à l'avance, ce qui impose la taille minimale B_{min} des blocs destination. La taille maximale B_{max} est aussi imposée. Pour chacun des N blocs destination $\mathbf{r}_n$, de taille B^2, l'algorithme recherche un bloc source $\mathbf{d}_{\alpha(n)}$ de taille D^2 avec $D = 2B$, centré sur l'un des points d'un réseau régulier. Fisher définit trois types de réseaux à utiliser en fonction de l'image à compresser, chacun étant

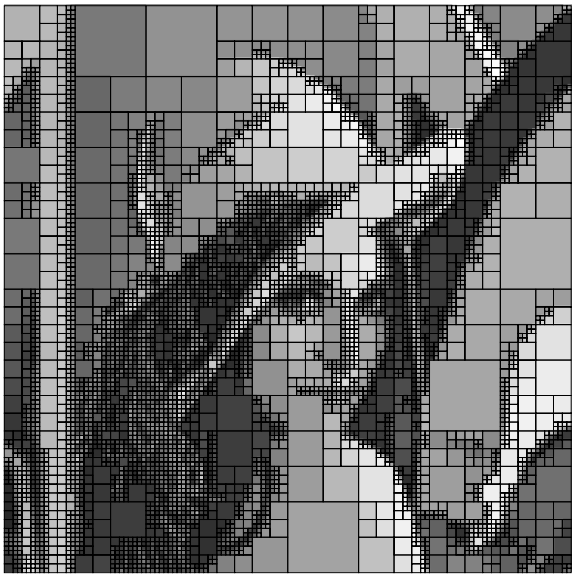


FIG. 25 - Illustration d'un partitionnement quadtree sur l'image Lena 512×512, composé de 5704 carrés. Sur cette figure, le contenu d'un bloc est constant : il est égal à la valeur moyenne des niveaux de gris du bloc de l'image originale (le taux de compression est dans ce cas égal à 46:1, sans compter le codage du quadtree et sans quantification ni codage entropique des luminances).

caractérisé par son pas d'échantillonnage p (voir figure 26) :

1. le réseau RES_1 pour lequel le pas p est constant. Le dictionnaire complet issu de ce réseau contient en moyenne autant de petits blocs que de grands blocs. Nous parlons ici de dictionnaire complet parce qu'il regroupe l'ensemble des blocs de taille $2B_{min} \times 2B_{min}$ à $2B_{max} \times 2B_{max}$.

2. le réseau RES_2 pour lequel $p = D$. Le dictionnaire complet contient dans ce cas plus de petits blocs que de grands.

3. le réseau RES_3 pour lequel p est fonction de B_{min} et de B_{max}. Le contenu du dictionnaire complet est inversé par rapport à celui calculé sur le réseau RES_2. Il contient plus de grands blocs que de petits.

Le réseau RES_3 est préféré lorsque l'image contient de larges zones homogènes et peu de textures fines. Il pourrait par exemple être utilisé pour compresser une image de nuages sur un fond homogène.

Le codage d'un bloc destination $\mathbf{r}$ se fait par recherche du bloc source $\mathbf{d}_{\alpha(n)}$ décimé qui permet de minimiser l'erreur entre $\mathbf{r}$ et l'approximation $\hat{\mathbf{r}}$ donnée par :

$$\min_{\beta_1,\beta_2} d(\mathbf{r}, \hat{\mathbf{r}}) = \min_{\beta_1,\beta_2} d(\mathbf{r} - \beta_1 \mathbf{b_1} - \beta_2 \mathbf{b_2}) \tag{42}$$

où β_1 et β_2 sont des coefficients réels. Si la distance minimale demeure supérieure à un seuil prédéfini, et si le niveau de $\mathbf{r}$ dans le quadtree demeure inférieur à la profondeur maximale de ce dernier, alors le bloc destination est redivisé et la procédure

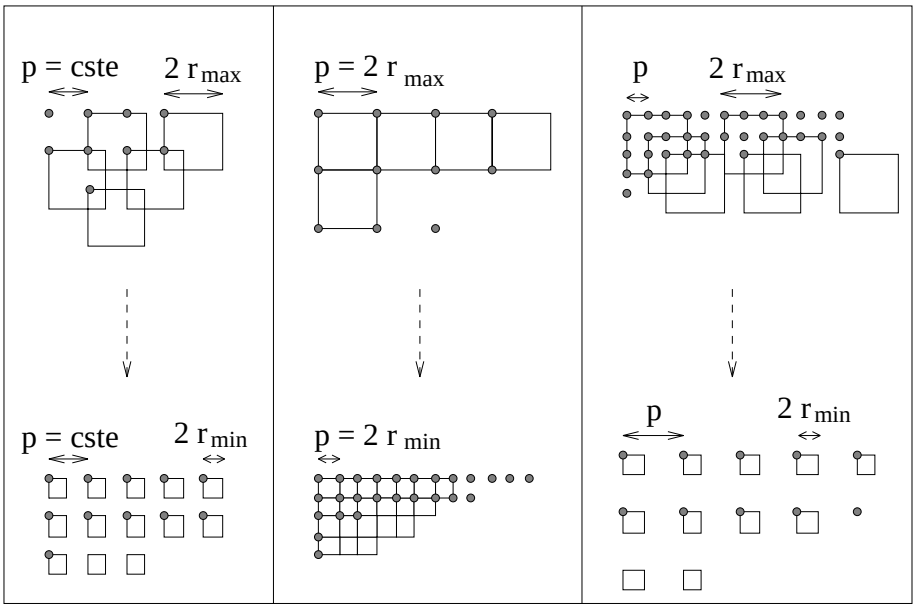
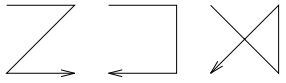


FIG. 26 - De gauche à droite : schémas des réseaux RES_1, RES_2 et RES_3 retournant les dictionnaires de blocs source.

de codage est relancée sur chacun des quatres sous-blocs créés. Si la distance minimale est inférieure au seuil fixé, le bloc $\mathbf{r}$ est codé par les coefficients β_1 et β_2 de la transformation massique retournant $\hat{\mathbf{r}}$, et par la position dans le dictionnaire du bloc source $\mathbf{d}$ associé. Ces informations codent la transformation élémentaire ω.
Jacobs et *al.* proposent de minimiser le nombre de blocs source à comparer avec un bloc destination en les classifiant [82]. Jacquin [85] a également proposé ce type d'optimisation en regroupant les blocs dans trois classes constituées de blocs homogènes, de blocs texturés, et de blocs incluant des frontières (voir section 3.3.2). Dans [82], les auteurs considèrent un bloc carré divisé en quatre quadrants ordonnés selon la courbe de Peano (fig. 24). Les quatre valeurs A_i représentent les valeurs moyennes des luminances de chacun des quatre quadrants ordonnés. La méthode de classification repose sur le fait qu'il est toujours possible d'orienter le bloc de manière à ce que la suite des valeurs A_i vérifie l'une des trois inégalités suivantes :

1. $A_1 \geq A_2 \geq A_3 \geq A_4$
2. $A_1 \geq A_2 \geq A_4 \geq A_3$
3. $A_1 \geq A_4 \geq A_2 \geq A_3$



Tout bloc carré peut donc être associé à l'une de ces trois classes. En calculant les variances V_i des valeurs de luminance, chacune des trois classes peut en outre être subdivisée en $4! = 24$ sous-classes.

4.3 Partitionnement semi-rigide

4.3.1 Partitionnement horizontal - vertical

Fisher et Menlove proposent un partitionnement rectangulaire couramment appelé *partition H-V* pour le codage des images selon une approche fractale [64] (figure 27).

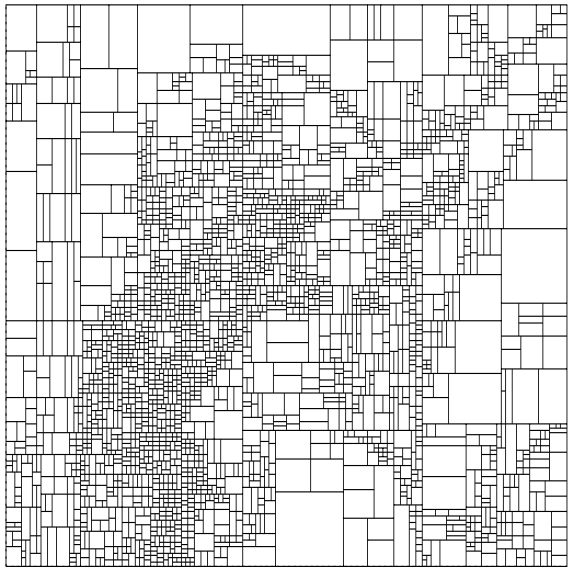


FIG. 27 - Extrait du livre de Y. Fisher : partitionnement H-V calculé sur l'image Lena et composé de 2910 rectangles.

Un processus récursif de subdivision des blocs en deux sous-rectangles conduit à une partition adaptée au contenu de l'image. La partition n'est pas rigide comme le quadtree puisque la division d'un bloc, qui tient compte de sa texture, ne crée pas nécessairement deux blocs d'égales surfaces. La partition est semi-rigide dans le sens où les arêtes des blocs demeurent obligatoirement soit horizontales, soit verticales : la géométrie des blocs est à orientation définie.

Intérêt du partitionnement pour la compression par fractales

La règle de construction de la partition H-V est proche de celle utilisée pour le calcul du quadtree dans le sens où la subdivision horizontale ou verticale d'un rectangle qui ne vérifie pas le critère d'homogénéité crée de nouveaux rectangles (en l'occurrence deux). Fisher propose deux méthodes de division qu'il utilise en fonction de la nature du rectangle inhomogène (figure 28) :

1. lorsque celui-ci est parcouru par une frontière oblique de l'image, la droite de séparation est choisie de manière à ce que l'un des sous-blocs soit parcouru diagonalement par la frontière, et que l'autre, s'il existe, reste homogène [63] [59]. Lorsqu'un rectangle de grande taille est déjà traversé par une frontière oblique,

il est redivisé de façon à ce que deux sous-rectangles disjoints soient traversés diagonalement par la frontière;

2. lorsque le rectangle inclut une frontière horizontale ou verticale, la droite de séparation est choisie de manière à créer deux sous-blocs homogènes : la droite est placée sur la frontière [60].

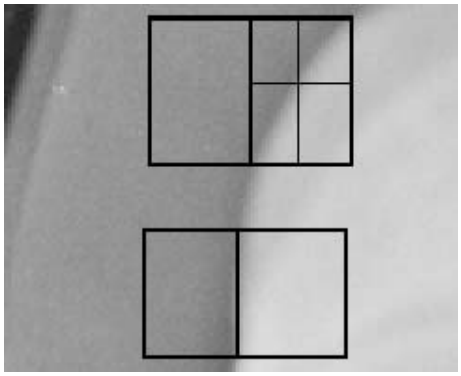


FIG. 28 - Partitionnement H-V: deux solutions possibles. Illustrations de la méthode 1 (en haut) et de la méthode 2 (en bas).

La partition H-V retourne les blocs destination à partir desquels est calculée la transformation fractale. Les blocs source peuvent être recherchés au travers de la même partition. Pour cela, la technique de division des rectangles selon les diagonales (méthode 2) est intéressante puisque l'union de quatre blocs destination peut dans certains cas constituer un bloc source à frontière diagonale. Un exemple est donné en haut de la figure 28 : l'union des quatre sous-blocs incluant la frontière oblique ressemble aux deux sous-blocs (haut droit et bas gauche) qui recouvrent une partie de la même frontière. La sous-optimalité due au fait que les blocs source ne sont pas recherchés dans l'image entière devrait en partie être compensée par le fait que la partition est construite de manière à ce qu'elle fournisse des blocs ou une union de blocs "auto-similaires à différentes échelles".
Fisher montre cependant dans [64] que ce n'est pas le cas. Il préconise plutôt de rechercher le bloc source parmi les blocs centrés sur un réseau régulier (et dense) de points du type de ceux représentés sur la figure 26. La hauteur (ou la largeur) du bloc source est contrainte à être supérieure à celle du bloc destination d'un facteur deux ou trois.
La comparaison ou le collage se fait par sous-échantillonnage spatial ou en moyennant des ensembles de 2×2 pixels à l'intérieur des rectangles source. Fisher propose différents niveaux d'optimisation qui se résument ainsi, dans un ordre de complexité décroissante :

- Les blocs source sont recherchés dans toute l'image (réseau régulier de pas égal à un pixel dans les directions horizontales et verticales). Il est évident que dans ce cas le temps de codage peut être très élevé;

- Lors des comparaisons inter-blocs, les quatre rotations du rectangle source sont ou ne sont pas considérées. Le fait de les prendre en compte revient à multiplier par quatre le nombre de blocs sources disponibles. Il est à noter que contrairement au carré sur lequel on opère huit isométries discrètes (4 rotations et 4 symétries), le rectangle n'en permet que six (4 rotations et 2 symétries horizontale et verticale);

- Le facteur d'échelle β_2 de la transformation affine des luminances, normalement positif ou négatif, peut être contraint à n'être que positif;

- Différents facteurs de taille sur les hauteurs et les largeurs des blocs sont ou ne sont pas testés (facteurs 2×2, 2×3, 3×2, et/ou 3×3);

- Une classification des rectangles selon une technique proche de celle mise en œuvre pour classifier les blocs carrés au sein d'un quadtree (section 4.2.2) peut être utilisée, dans le but d'accélérer le codage au détriment de la qualité de reconstruction.

Lorqu'un partitionnement quadtree et un partitionnement rectangulaire sont adaptés de la même manière au contenu de l'image (en utilisant le même critère d'homogénéité des blocs), le partitionnement rectangulaire fournit moins de bloc que le partitionnement en quadtree. Cette propriété vient principalement du fait qu'un rectangle non homogène peut être divisé en quatre ou deux parties de tailles différentes. Il apparaît dans ce cas moins de blocs "inutiles" dans la partition. Le contenu de chacun des blocs (homogène ou à forte variance) est en outre choisi explicitement au cours du calcul de la partition. Ces différentes propriétés rendent la construction de la partition très souple par rapport à celle du quadtree. Les arêtes des blocs sont cependant contraintes à être soit horizontales, soit verticales.

4.4 Partitionnement souple

Nous décrivons dans cette section le diagramme de Voronoï ainsi que le graphe dual de Delaunay, chacun retournant une partition du support de l'image. L'intérêt de ces partitionnements est qu'ils sont souples puisqu'ils sont calculés sur un ensemble de points pouvant être positionnés à peu près n'importe où sur le support de l'image. Nous rappelons dans la section 4.4.1 les principales définitions et propriétés de ces deux modèles de partitionnement du plan, nécessaires à la compréhension de la suite du chapitre. Nous présentons dans la section 4.4.2 différents algorithmes de construction des deux diagrammes, en insistant sur la méthode incrémentale. Cette dernière, de par son aspect dynamique, est utilisée pour construire une partition triangulaire adaptée au contenu de l'image. Le calcul de la partition est détaillé dans la section 4.4.3.

4.4.1 Triangulation de Delaunay et diagramme de Voronoï

Définitions et propriétés

Points, sites, germes
On désigne par P un ensemble composé de n *points* de l'espace $\mathbb{R}^2$ appelés aussi *sites* ou *germes*:

$$P = \{p_i \in \mathbb{R}^2, i = 1, \ldots, n\}$$

De manière à éviter les ambiguïtés sur la topologie des triangles de Delaunay, nous supposerons dans toute la suite qu'aucun sous-ensemble de P n'est formé de quatre points co-circulaires, et qu'aucun sous-ensemble n'est formé de trois points alignés.

Région de Voronoï
Soit p un élément de P. La *région de Voronoï* notée $Vor_P(p)$ associée à p est l'ensemble des points de $\mathbb{R}^2$ plus proches, au sens de la distance Euclidienne d, de p que de tous les autres points de P :

$$Vor_P(p) = \{x \in \mathbb{R}^2, d(x,p) \leq d(x,q), \forall q \in P - p\}$$

Il est montré dans ce cas que chaque région de Voronoï est polygonale et convexe. Les *polygones de Voronoï* sont soit bornés, soit non bornés (les polygones ouverts sur l'infini sont associés aux points p_i appartenant à la frontière de l'enveloppe convexe de P).
Le sommet d'un polygone de Voronoï est appelé *sommet de Voronoï*.
Remarque : la distance d peut ne pas être Euclidienne. Nous ne parlons pas dans ce cas de polygones mais simplement de régions de Voronoï. Les approches discrètes par propagation, basées sur les distances discrètes d_4, d_8, ou de chanfrein, retournent ce type de partitionnement. Nous en reparlerons dans la section 4.4.2.

Diagramme de Voronoï
Le *diagramme de Voronoï* appelé aussi partition de Voronoï d'un ensemble P de n points est l'ensemble de tous les polygones de Voronoï de P :

$$VOR_n(P) = \bigcup_{p \in P} Vor_P(p)$$

Graphe de Delaunay
Le graphe dual du diagramme de Voronoï d'un ensemble P de n points est le *graphe de Delaunay*. Deux points de P, p et q, créent une arête dans le graphe de Delaunay (i.e. p et q sont voisins) si et seulement si, $Vor_P(p)$ et $Vor_P(q)$ sont adjacents dans le Diagramme de Voronoï :

$$DEL(P) = \langle P, E = \{(p,q) \in P^2, Vor_P(p) \cap Vor_P(q) \neq \emptyset\}\rangle$$

Voisinage
Le *voisinage* au sens de Delaunay d'un point p de P peut aussi être défini par :

$$N_s(p) = \{q \in P \text{ tels que } (p,q) \in E\}$$

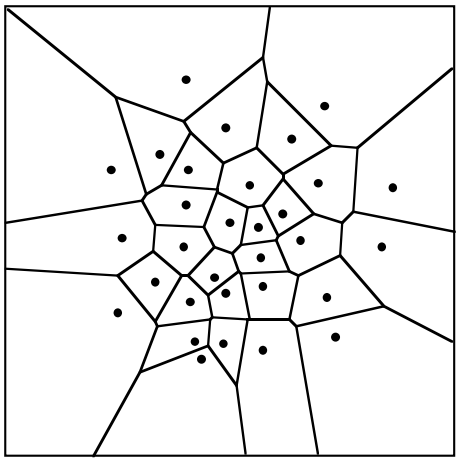


FIG. 29 - Partition de Voronoï d'un ensemble de points de P.

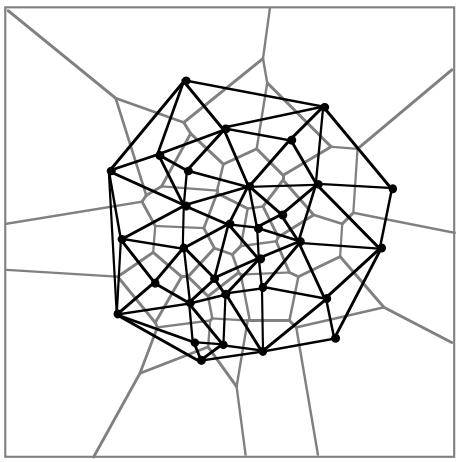


FIG. 30 - Diagramme de Voronoï et graphe de Delaunay.

où E est l'ensemble des arêtes du graphe de Delaunay.

Diagramme de Delaunay
Le graphe de Delaunay de P est considéré comme l'unique triangulation de l'enveloppe convexe de P telle que l'intérieur des cercles circonscrits aux triangles (p_i, p_j, p_k) de P^3 ne contienne aucun autre point de P :

$$DEL(P) = \{(p_i, p_j, p_k) \in P^3 \text{ tel que } B(p_i, p_j, p_k) \cap (P - p_i - p_j - p_k) = \emptyset\}$$

Une telle propriété nous permet de considérer le graphe de Delaunay comme une partition triangulaire de l'enveloppe convexe d'un ensemble de points dans $\mathbb{R}^2$. En disposant des points le long du bord d'une image, il est ainsi possible de construire une partition du support de cette image.

Triangle de Delaunay

Un triangle $T = (p_i, p_j, p_k)$ où p_i, p_j, p_k sont dans P est un *triangle de Delaunay* si et seulement si, l'intérieur du cercle circonscrit à T ne contient aucun point de P :

$$B(p_i, p_j, p_k) \cap (P - p_i - p_j - p_k) = \emptyset$$

Une propriété intéressante des triangles de Delaunay est qu'ils sont tous bornés.

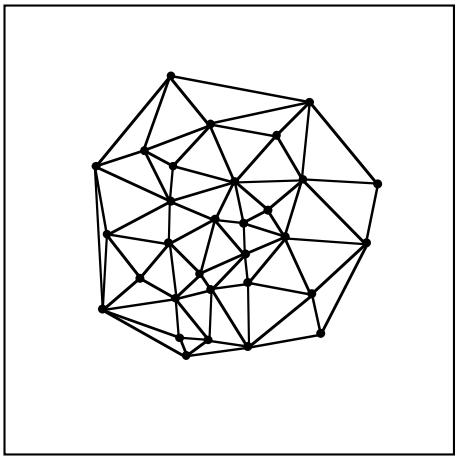


FIG. 31 - Partition de Delaunay d'un ensemble de points de P.

4.4.2 Calcul des partitions

Il existe une multitude d'algorithmes pour construire un diagramme de Voronoï ou de Delaunay dans le plan ou dans l'espace [71] [131]. Les méthodes peuvent se décomposer principalement en deux classes :

- les approches globales qui calculent le partitionnement de l'ensemble P prédéfini de points.
- les approches incrémentales fonctionnant de manière dynamique. Le partitionnement est ajusté après chaque ajout d'un nouveau point dans l'ensemble P.

Méthodes globales

La méthode la plus connue est la méthode dénommée *divide and conquer*, détaillée dans [133] et dans [32]. Le principal avantage de cet algorithme récursif est que sa complexité est réduite, en $O(n \log n)$ dans le pire des cas, où n désigne le nombre de points de l'ensemble P. Son défaut est qu'il ne permet pas l'insertion ou la suppression dynamique de points dans le diagramme de Voronoï, ni même leur déplacement. Il nécessiterait une réévaluation complète du diagramme.

Une autre approche globale est spécifique du caractère discret des données. Dans ce cas, P est un sous-ensemble de $\mathbb{Z}^2$. La construction du diagramme s'effectue par propagation autour de chacun des points de l'ensemble P. Les arêtes de Voronoï,

composées de points de $\mathbb{Z}^2$, sont situées à égale distance entre deux points voisins p_i et p_j de P. Pour cela, différentes *distances discrètes* sont utilisées, retournant des régions de Voronoï : la distance aux quatre plus proches voisins notée d_4, la distance aux huit plus proches voisins notée d_8 [32], et les distances du chanfrein permettant de mieux approximer la distance euclidienne [155] [94]. Des algorithmes rapides de calcul des régions de Voronoï existent [32], mais le défaut majeur de l'approche discrète est qu'elle n'est pas structurée dans un environnement de graphe contenant l'information de voisinage entre les polygones, et permettant le passage direct au diagramme dual de Delaunay.

Méthodes incrémentales

Les méthodes incrémentales sont intéressantes de par leur aspect dynamique. Le principe est d'évaluer le diagramme $VOR_{n+1}(P)$ de l'ensemble P composé des points p_1 à p_{n+1}, par modification locale du diagramme $VOR_n(P - p_{n+1})$, après insertion du nouveau point p_{n+1}.

Structure de données :
Bowyer a remarqué [26] que les polygones de Voronoï ont certaines propriétés pouvant être exploitées pour la construction d'une structure de données efficace en terme de manipulation informatique du diagramme de Voronoï et de parcours dans le graphe de delaunay :

- Chaque sommet de Voronoï provient de l'intersection d'exactement trois polygones de Voronoï (toujours sous la condition de la non cocircularité).

- Tout sommet s de Voronoï est équidistant de 3 points générateurs appartenant à P. Ces points générateurs sont les sommets du triangle de Delaunay associé à s et s est le centre du *cercle de Delaunay* circonscrit au triangle, vide de tout autre point. Le triangle est encore appelé *triangle dual* du sommet s. Il est à noter que s ne se trouve pas nécessairement à l'intérieur du triangle.

Partant de la remarque que l'information est bornée, Bowyer a défini une *structure de données* associant à chacun des sommets s de Voronoï des pointeurs sur :

1. ses 3 sommets voisins s_1, s_2 et s_3 (éventuellement rejetés vers l'infini dans le cas des bords de l'enveloppe convexe de P)

2. les 3 points p_1, p_2 et p_3 de P, générateurs de s.

On utilise en outre une numérotation intelligente des pointeurs, codant la topologie des triangles adjacents. La numérotation permet par exemple de retrouver directement l'arête séparant un sommet s courant de son sommet voisin s_i. Celle-ci est portée par les points p_j p_k avec $j, k \neq i$ (voir figure 32).

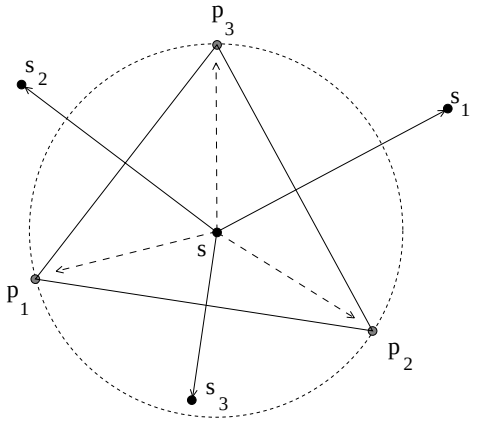


FIG. 32 - Structure de données.

Nous résumons ci-après la méthode que nous utilisons pour insérer un point p_{n+1} dans le diagramme de Voronoï $VOR_n(P)$. L'algorithme est le suivant :

1. chercher un premier sommet de Voronoï s à supprimer. La localisation se fait dans le graphe de Voronoï et consiste à rechercher le sommet s dont le triangle dual contient le point p_{n+1}. Cette recherche est initialisée sur un sommet quelconque de Voronoï. Elle est globale, et tend à suivre la "ligne de plus grande pente", le long des arêtes des polygones de Voronoï liant le sommet d'initialisation au sommet s (cette recherche est donc faite par un algorithme de type descente en gradient).

2. chercher tous les autres sommets de Voronoï à supprimer. De tels sommets sont plus proches du nouveau point p_{n+1} que de leurs points générateurs. Ceci est équivalent à trouver tous les centres des triangles pour lesquels le cercle de Delaunay contient le point p_{n+1}. Les triangles associés à tous les sommets à supprimer forment un *polygone étoilé*. La recherche est locale.

3. créer tous les nouveaux sommets de Voronoï, ainsi que leurs relations de voisinage (mise à jour des pointeurs). Cette étape reconstruit la nouvelle triangulation de Delaunay intérieure au polygone étoilé. Les sommets de ce dernier sont voisins du point p_{n+1} au sens de Delaunay.

Dans [70], les auteurs programment une descente en gradient le long des arêtes du graphe de Delaunay initialisée sur un point quelconque, dans le but de trouver le point p le plus proche du nouveau point p_{n+1}. Bowyer recherche le point le plus proche du nouveau point p_{n+1} au sein du graphe de Delaunay [26]. Partant de ce point, la recherche du sommet à supprimer est facile, sachant qu'il fait partie des sommets plus proches de p_{n+1} que de leurs points générateurs. Cette recherche locale nécessite cependant de stocker une liste de points contigus associée à chacun des points de P.
Bertin apporte une amélioration à l'algorithme de Bowyer, en effectuant la recherche directement dans le graphe de Voronoï, sans avoir à mémoriser de listes

de points supplémentaires [23]. Il utilise pour cela un algorithme initialement donné par Schmitt et Borouchaki dans [149]. L'algorithme consiste à traverser une arête du triangle courant lorsque le point p_{n+1} se trouve de l'autre côté de la droite contenant cette arête. Il se résume de la manière suivante :

Soit p_1, p_2 et p_3 les points générateurs d'un sommet s quelconque du diagramme de Voronoï.
- si pour toute arête (p_i, p_j) $(i \neq j,\ i,j \in \{1,2,3\})$ du triangle associé au sommet s courant, le produit scalaire $\langle ss', p_i p_{n+1} \rangle$ est négatif, où s' est le sommet voisin de s sur l'arête de Voronoï perpendiculaire à (p_i, p_j), alors s est le sommet cherché. p_{n+1} est à l'intérieur du triangle associé au sommet s, l'algorithme a convergé. Cette condition est illustrée sur la figure 33.
- sinon prendre s' parmi les sommets voisins de s, tel que $\langle ss', p_i p_{n+1} \rangle \geq 0$, puis poser $s = s'$ et retourner à l'étape précédente. Le point s est ainsi déplacé dans le graphe, en direction du point p_{n+1}.

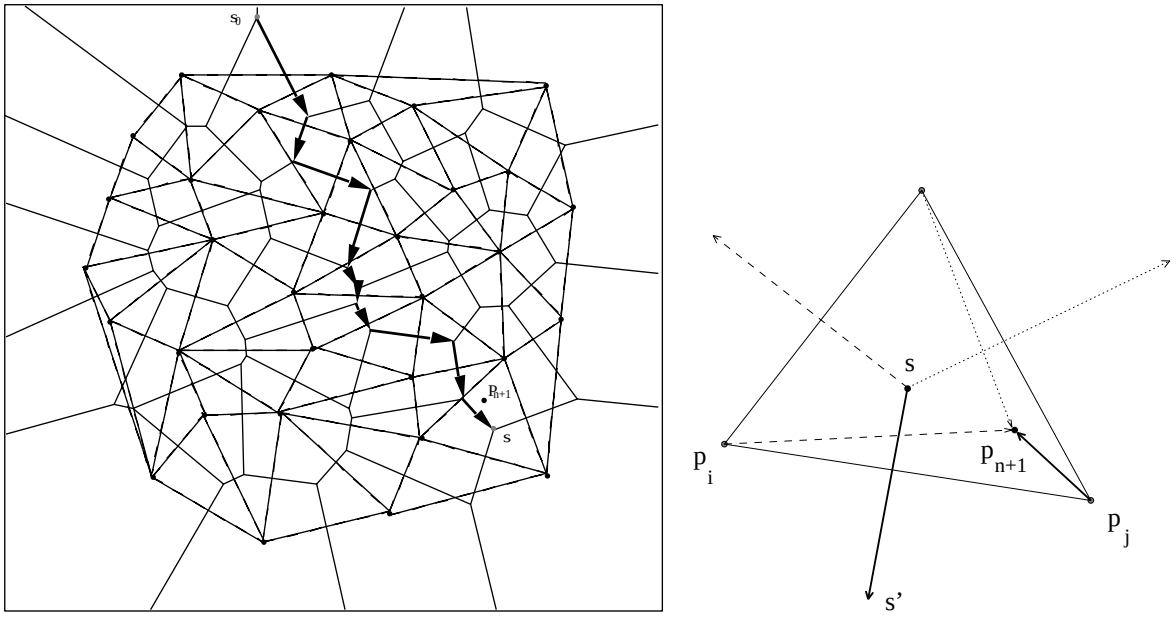
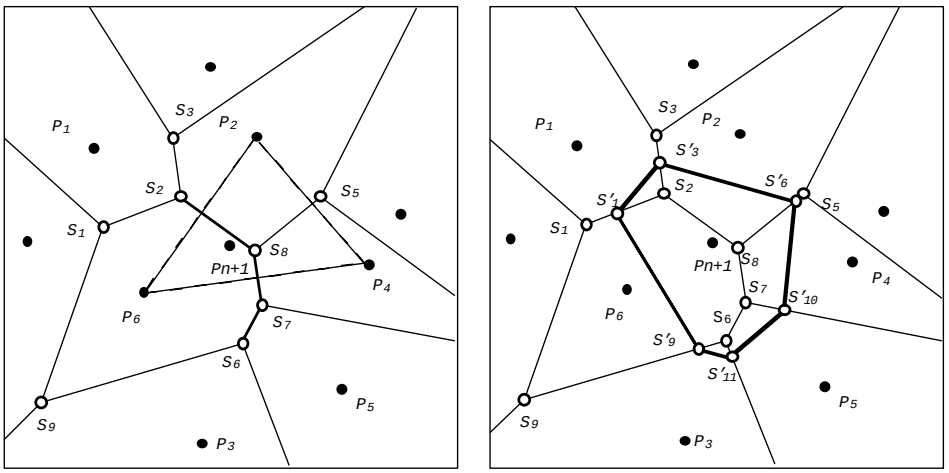


FIG. 33 - Illustration de l'étape 1 de l'algorithme d'insertion.

L'étape 3 consiste à parcourir tous les sommets à supprimer, et pour chacun d'eux à effectuer le traitement suivant :

Soit s un des sommets à supprimer, s_1, s_2, s_3 ses trois sommets voisins, et p_1, p_2, p_3 ses trois points générateurs. Pour tout $i = 1$ à 3,
- si s_i n'est pas un sommet à supprimer, alors créer un nouveau sommet s' centre du cercle circonscrit au nouveau triangle défini par les deux points p_i et p_j créant

l'arête de Voronoï (s_i, s) et par le nouveau point p_{n+1}. On montre que s' est situé sur l'arête (s_i, s). La relation de voisinage entre le nouveau triangle et le triangle connexe, extérieur au polygone étoilé reste inchangée.
- sinon, ne rien faire.



(a) étape 2 (b) étape 3

FIG. 34 - Illustration des étapes 2 et 3 de l'algorithme d'insertion.

Il reste ensuite à mettre à jour les relations de voisinage entre les nouveaux triangles inclus dans le polygone étoilé.

4.4.3 Partitionnement triangulaire adapté au contenu de l'image

Nous détaillons dans cette partie l'algorithme de partitionnement en triangles de Delaunay du support d'une image en niveaux de gris. Cette méthode est qualifiée de souple puisqu'elle retoune la triangulation de l'enveloppe convexe d'un ensemble quelconque de points, distribués sur le support de l'image.
Au cours de la construction de la partition, et lorsque celle-ci est entièrement calculée, chacun des éléments triangulaire est caractérisé par la valeur moyenne et l'écart-type des niveaux de gris qu'il englobe. Ces paramètres peuvent ensuite être utilisés directement au cours du codage par fractales.

Algorithme de division et fusion

L'algorithme de division et fusion que nous proposons est une généralisation de la méthode de *split and merge* sur l'arbre quaternaire, proposée dans [73] et rappelée dans la section 4.2.1, page 80. Il retourne une partition adaptée au contenu de l'image, servant à initialiser l'étape suivante de compression par fractales. Le détail

de l'algorithme est donné ci-après :

1. disposer un ensemble régulier de points sur le support de l'image, formant un maillage triangulaire comme indiqué sur la figure 35. Nous montrerons dans la section 5.3.2 l'utilité d'une telle initialisation pour la compression par fractales.

2. *divisions* : tant qu'il existe des triangles non homogènes :

   - calculer la triangulation de l'ensemble des points.
   - calculer les attributs de chaque triangle (écart-type des niveaux de gris des points image contenus dans le triangle et surface du triangle).
   - insérer un nouveau point sur le barycentre de chaque triangle ne vérifiant pas le prédicat d'homogénéité, et dont la surface demeure supérieure à un seuil T_0. Un bloc est déclaré homogène si la variance des niveaux de gris de celui-ci est inférieure à un seuil T_1 (cette étape a pour effet de diviser les triangles non homogènes).
   - réitérer cette phase de division tant qu'il y a des points à ajouter et donc des triangles non homogènes à surface supérieure à T_0.

3. *fusion* : supprimer tous les points inutiles de la triangulation. Ces points sont les points centraux des polygones étoilés formés de l'union de triangles homogènes connexes et de même amplitude. Les blocs inclus dans un même polygone ont la même amplitude si la différence maximale des valeurs moyennes des triangles demeure inférieure à un seuil T_2.

Remarques :
1 - : L'ajout d'un nouveau point sur le barycentre du triangle à diviser est préféré à une solution adaptative puisque nous voulons minimiser la quantité d'information nécessaire au codage de la partition. Dans [33] les auteurs proposent une méthode adaptative qui consiste à positionner le nouveau point à l'endroit où le module de l'erreur entre les niveaux de gris de l'image et le plan passant par les niveaux de gris des trois sommets du triangle à diviser est maximale. Cette solution ralentit l'étape de division des triangles, et implique la mémorisation des coordonnées de chacun des points ajoutés. Dans notre cas, nous pouvons coder efficacement la triangulation de Delaunay issue des étapes de division et fusion, sans avoir à mémoriser les coordonnées des sommets des triangles. La méthode, proche de celle utilisée pour le codage d'un quadtree, est donnée en section 5.3.2.
2 - : L'étape itérative de division rend une image sur-segmentée, composée d'un grand nombre de blocs connexes similaires. Dans [33] les auteurs regroupent itérativement deux à deux les triangles similaires au sein de la partition, en considérant leur moyenne de niveaux de gris. Leur objectif final est de retourner une segmentation de l'image en larges régions homogènes. Chaque région, formée d'un ensemble de triangles connexes, est de forme polygonale.
Notre approche vise plutôt à supprimer directement de la partition un grand nombre

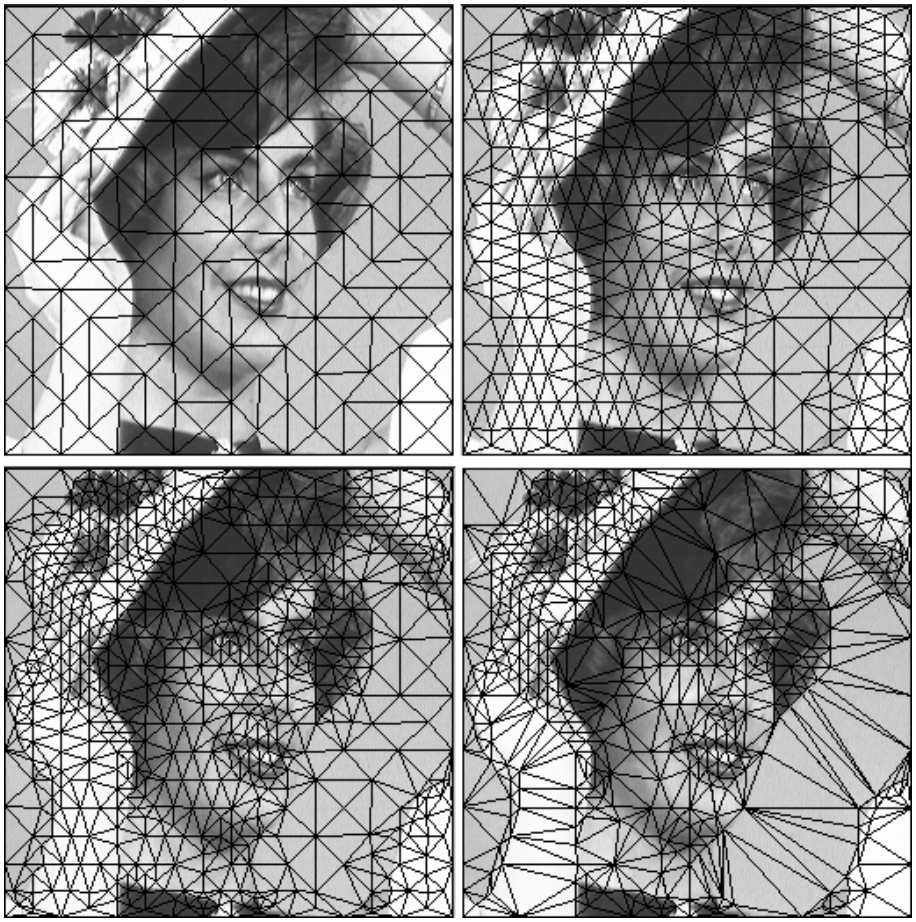


FIG. 35 - Initialisation (en haut à gauche) suivie des divisions et de la fusion (en bas à droite).

de triangles connexes similaires, tout en conservant des blocs triangulaires[1]. Le partitionnement ainsi simplifié se compose de triangles allongés au-dessus des zones homogènes, prenant appui sur des petits triangles situés au-dessus des zones texturées et des contours de l'image originale. Ce type de partitionnement est issu de l'étape de fusion.

Algorithme de fusion simple : une variante

Une solution très rapide pour calculer une partition de Delaunay adaptée au contenu de l'image consiste à n'opérer qu'une seule étape de fusion sur la triangulation d'un réseau dense et régulier de points disposés sur le support de l'image. La figure 36 montre un résultat de l'algorithme. L'inconvénient d'une telle méthode vient du fait que la mauvaise localisation des triangles de départ conduit à supprimer un trop grand nombre de triangles au cours de la phase de fusion, en laissant le reste

1. L'algorithme de fusion préservant la forme des blocs de la partition originale n'est pas réalisable dans le cas du quadtree

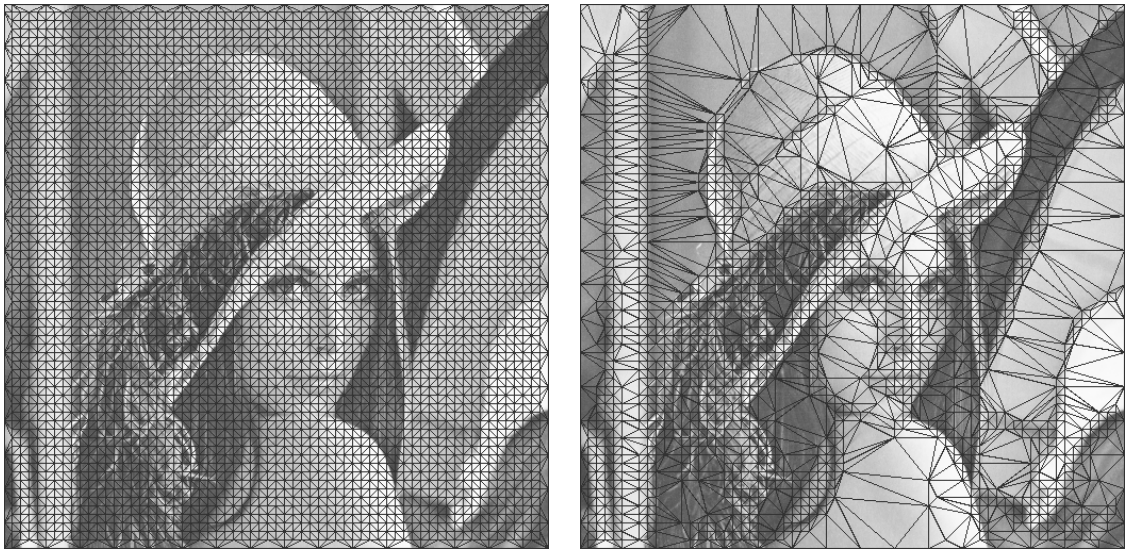


FIG. 36 - Fusion sur une triangulation fine.

des blocs inchangés. Au terme des phases de division de l'algorithme de "Split and Merge" la partition est adaptée au contenu de l'image. L'étape de fusion supprime dans ce cas moins de triangles, et la disposition des autres triangles tient compte de la texture de l'image.

Triangulation contrainte par les contours

Les approches classiques de partitionnement utilisées pour initialiser la phase de compression par fractales retournent un nombre élevé de blocs carrés ou triangulaires, de petite taille, recouvrant les contours de l'image. Les blocs ne sont cependant pas tous traversés par les contours de la même manière, comme le montre la figure 37.

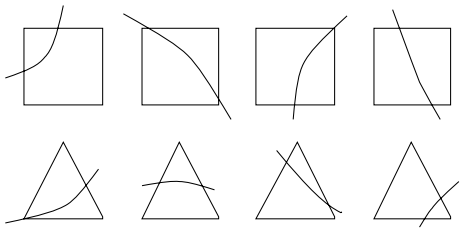


FIG. 37 - Blocs carrés et triangulaires incluant un contour simple.

Ceci peut être un inconvénient puisque le codage par fractales consiste à rechercher, pour un bloc destination, un bloc source de même nature, de surface plus grande au sein de la même image. Le bloc destination avec frontière ne peut donc

être correctement codé que si l'image contient une grande diversité de blocs source incluant une frontière. Une solution à ce problème, suggérée par Fisher [64] consiste à contraindre le passage des contours à l'intérieur des blocs rectangulaires de la partition Horizontale-Verticale (section 4.3.1) : les contours sont principalement, soit contre les bords des rectangles, soit diagonaux à l'intérieur des rectangles.
Nous présentons dans cette section un autre type de partition contrainte, tirant pleinement profit de la souplesse du modèle de partitionnement de Delaunay. La partition est telle qu'en moyenne, les bords des triangles s'appuyent sur les frontières d'orientation quelconque de l'image. La plupart des blocs ainsi générés le long d'un même contour se ressemblent et sont homogènes. Le partitionnement est adapté à la forme des objets de l'image.

La triangulation contrainte est calculée sur un ensemble P de points obtenus selon l'algorithme détaillé ci-dessous en quatre points :

1. détecter les principaux contours de l'image.

2. échantillonner les contours en plaçant des points aux endroits de forte courbure.

3. ajouter des points régulièrement espacés sur les contours, entre chacun des points précédemment détectés. L'écart entre les points ajoutés est noté d_1.

4. placer des points supplémentaires de chaque côté des contours, perpendiculairement aux segments. L'ajout d'un de ces points se fait dans un contexte de contrôle de proximité vis à vis des autres points déjà insérés.

L'étape de recherche des contours de l'image s'effectue à l'aide de l'opérateur de Deriche [48], présenté dans la section 4.1.1, page 74. Nous vérifions expérimentalement que la délocalisation des contours détectés est d'autant plus perceptible que le paramètre α des filtres directionnels est petit (formules 40 et 41).
Ce dernier est fixé dans notre application à $\alpha = 0.2$ puisque ce choix permet de détecter les principaux contours de l'image, tout en limitant le nombre de petits contours inutiles.

L'étape 2 consiste à rechercher les points de forte courbure du contour. De nombreuses études ont été faites sur ce sujet [58]. Nous utilisons dans notre cas une méthode d'approximation itérative [162], mise en cascade avec un processus de suivi de contour. L'algorithme traite les pixels au fur et à mesure de leur arrivée, jusqu'à ce qu'un critère, remis à jour après chaque ajout, ne soit plus vérifié.
La figure 38 illustre l'algorithme de découpage recevant en entrée la liste des points c_0 à c_N d'un contour de l'image. L'ensemble des points de rupture est noté P.

La méthode de recherche des points de forte courbure est détaillée dans [32]. Nous en rappelons ci-après l'algorithme :

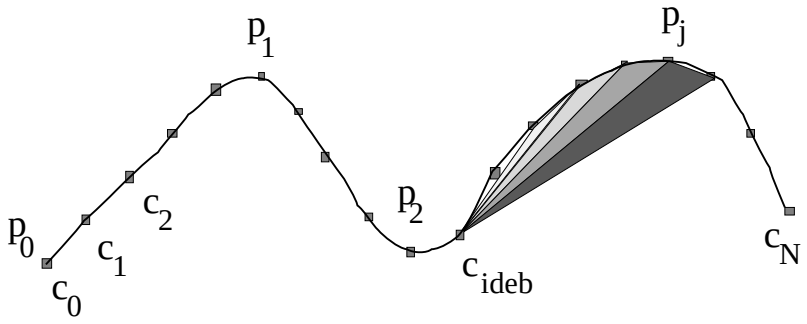


FIG. 38 - Approximation itérative d'un contour.

$i_{deb} = 0$, $err = 0$,

$c_{i_{deb}}$ a pour coordonnées $(x_{i_{deb}}, y_{i_{deb}})$, c_j a pour coordonnées (x_j, y_j).

pour $j = 1$ à N **faire**

$err = err + (x_j - x_{i_{deb}})(y_j - y_{j-1}) - (y_j - y_{i_{deb}})(x_j - x_{j-1})$

si $|err| \geq \epsilon \sqrt{(x_j - x_{i_{deb}})^2 + (y_j - y_{i_{deb}})^2}$ **alors**

le critère d'erreur n'est plus vérifié.

c_{j-1} est l'un des points recherchés, il est ajouté à l'ensemble des points de rupture P.

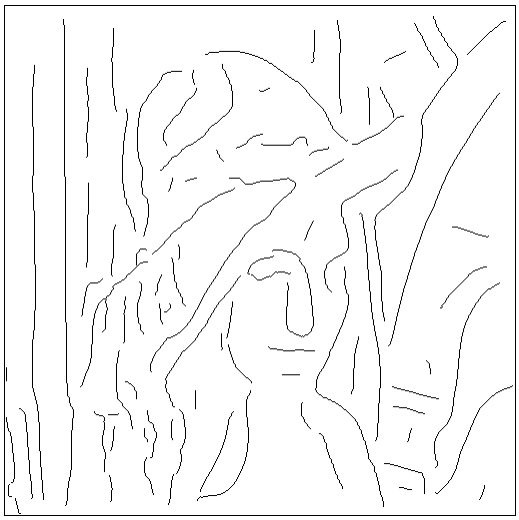
$i_{deb} = j$, $err = 0$.

fin_si

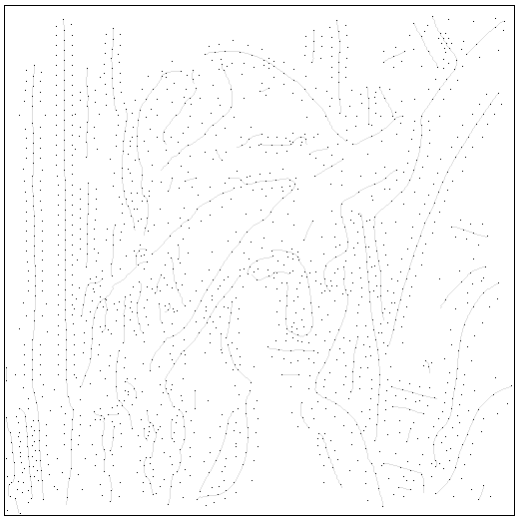
fin_pour

La valeur ϵ représente le maximum d'erreur cumulée admise par unité de surface du segment approximant $[c_{i_{deb}}, c_j]$. L'erreur err correspond à la somme signée des surfaces définies par les triangles $(c_{i_{deb}} c_{j-1} c_j)$.

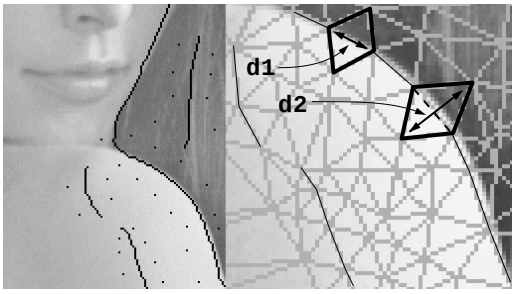
A l'issue de l'étape 4, les triangles s'appuient contre les contours de l'image, à condition que la distance d_1 soit inférieure à la distance d_2 (figure 39.c). Dans le cas contraire, les triangles ont un sommet sur un contour et leur base coupe perpendiculairement le contour. Ces deux solutions nous permettent de contrôler avec précision le contenu des triangles proches des contours. La partition obtenue peut ainsi être comparée à la partition H-V dans le sens où les blocs sont soit homogènes, soit traversés tous de la même manière par un contour. Les régions texturées de l'image sont quant à elle partitionnées de manière quelconque à l'aide de triangles de petite taille.



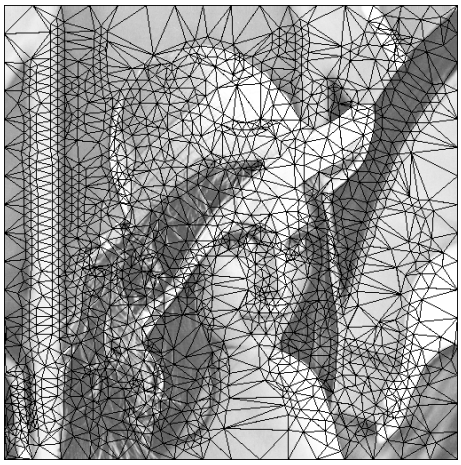
(a) détection des contours



(b) positionnement des points de P



(c) positionnement des points de P



(d) Triangulation de l'ensemble P

FIG. 39 - Triangulation contrainte par les contours.

Autres partitionnements triangulaires

Fisher suggère dans [60] l'utilisation d'une triangulation adaptée au contenu de l'image, sans l'utiliser pour la compression par fractales. Il propose de diviser diagonalement le support rectangulaire de l'image, puis de subdiviser récursivement chacun des deux triangles parent en quatre sous-triangles enfants. La division est réalisée en joignant trois points judicieusement choisis sur les arêtes du triangle père. Ce partitionnement, qui est plus flexible que le partitionnement H-V, a l'inconvénient d'être difficile à implanter et surtout à coder.
L'algorithme de calcul récursif du type de celui utilisé pour la construction du quadtree peut aussi être implanté sur des blocs triangulaires en partant du support de

l'image divisé diagonalement en deux triangles. La division peut être faite en ajoutant un nouveau sommet sur le barycentre du triangle non homogène, et en joignant les trois sommets du triangle à ce point [2]. Cette solution crée très vite des triangles très étirés, difficiles à traiter dans le cas discret. Une meilleure solution consiste à opérer une subdivision barycentrique considérant l'ensemble des six triangles qui ont pour sommet commun le barycentre du triangle original. Ce type de division récursive est illustré sur la figure 40.

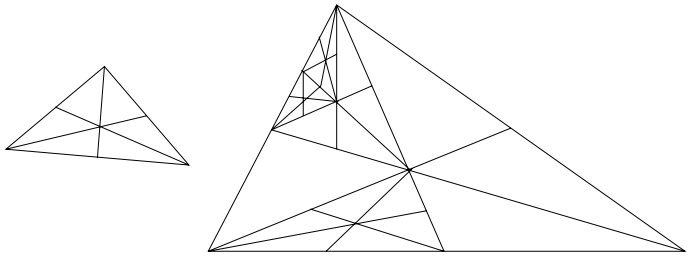


FIG. 40 - Subdivision barycentrique récurrente d'un triangle.

La triangulation obtenue à partir de cet algorithme est adaptative mais non flexible. La taille et la forme des blocs est contrainte par le processus de division récursif, tout comme dans le cas du quadtree. Pour ces raisons, nous n'avons pas testé ce partitionnement triangulaire pour la compression des images par fractales.

4.5 Conclusion

Nous avons introduit dans ce chapitre différents modèles géométriques de partitionnement pour la compression par fractales. Le quadtree génère une partition en blocs carrés, adaptée au contenu de l'image. La règle de construction du partitionnement est rigide mais a l'avantage de pouvoir être codée efficacement. Nous avons décrit l'algorithme de compression proposé par Fisher et utilisant ce partitionnement ainsi que son extention au partitionnement rectangulaire. Ce dernier est plus souple que le quadtree ; il est par conséquent plus difficile à coder. La troisième partie de ce chapitre était consacrée à la triangulation de Delaunay adaptée de façon très souple au contenu de l'image. Nous avons décrit trois algorithmes permettant de générer de telles partitions. Celles-ci seront utilisées pour la compression par fractales dans le chapitre 5.

2. Il est à noter que la triangulation adaptative ainsi obtenue ne vérifie pas la propriété du cercle vide de la triangulation de Delaunay.

Chapitre 5

Compression fractale et triangulation

5.1 Introduction

Le codage d'une image A par fractales consiste à calculer un opérateur finalement contractant W tel que $W(A)$ soit une très bonne approximation de A. Cette méthode s'appuie sur les travaux de Barnsley concernant la théorie des IFS.
Nous avons vu dans le chapitre 3 que l'approximation de l'image A à partir d'elle-même n'est pas facile à réaliser lorsque A n'est pas auto-similaire. La solution à ce problème consiste à approximer chacune des régions de l'image à partir d'autres régions de la même image, en utilisant des transformations locales. L'image est pour cela partitionnée en N blocs $\mathbf{r}_n$, formant une partition R. Chacun des blocs est ensuite mis en correspondance avec un autre bloc $\mathbf{d}_n$ de l'image, à partir duquel il est possible d'approximer, par une transformation élémentaire ω_n, la fonction des niveaux de gris de $\mathbf{r}_n$.
L'opérateur W définissant une transformation fractale de l'image est composé de l'ensemble des N transformations ω_n. Il est finalement contractant à condition qu'un nombre suffisant de transformations ω_n soient contractantes.

Le codage de la transformation fractale nécessite de mémoriser les coefficients des N transformations ω_n. Il est donc d'autant plus efficace que la partition R contient un nombre réduit de blocs.
Les premiers travaux de Jacquin exposés dans le chapitre 3 ont démontré l'intérêt de s'inspirer de la théorie des IFS pour coder des images naturelles mais n'ont cependant pas permis d'atteindre des taux de compression élevés. Cette limitation est due au fait que la partition R proposée est carrée régulière et donc composée d'un nombre trop important de blocs $\mathbf{r}_n$. Nous avons vu dans le chapitre 4 que ce probème a pu être contourné en calculant la transformation fractale sur un partitionnement carré ou rectangulaire adapté au contenu de l'image (quadtree, H-V).

Nous décrivons dans ce chapitre 5 notre approche qui consiste à calculer la transformation fractale de l'image sur un partitionnement triangulaire adapté au contenu de l'image. Divers algorithmes [31] [46] permettant de construire une telle triangulation ont déjà été présentés dans le chapitre 4. Nous verrons que notre méthode de codage diffère du schéma de base dans le sens où les blocs $\mathbf{d}_n$ ne sont pas recherchés n'importe où dans l'image mais au travers d'une deuxième triangulation. Les contraintes imposées sur les tailles des blocs sources et destination sont en outre plus souples que celles imposées dans les schémas classiques puisque les formes respectives des triangles peuvent être quelconques.

5.2 Transformation fractale

La compression d'une image A par fractales consiste à calculer la transformation finalement contractante W qui permet de vérifier la relation suivante[1] :

$$A = \bigcup_{n=1}^{N} \mathbf{r}_n \simeq W(A) = \bigcup_{n=1}^{N} \omega_n(\mathbf{d}_{\alpha(n)}) \tag{43}$$

où les N blocs $\mathbf{r}_n$ sont appelés blocs destination, et les blocs $\mathbf{d}_{\alpha(n)}$ sont appelés blocs source. Nous rappelons qu'un bloc source $\mathbf{d}_{\alpha(n)}$ peut être associé à plusieurs blocs destination, et que par conséquent α est une application de $[1 \dots N]$ vers $[1 \dots Q]$ où Q est le nombre de blocs source utilisés.

5.2.1 Algorithme de codage

Nous détaillerons par la suite chacune des étapes de l'algorithme de codage. Celui-ci est résumé dans la table a, et illustré sur la figure 41.

```
calculer la triangulation R
calculer la triangulation D
pour i allant de 1 à N (i = indice du triangle r_i de R)
     {
     erreur_min = float_maximum
     pour j allant de 1 à Q (j = indice du triangle d_j de D)
          {
          erreur = d(r_i, w(d_j))
          si erreur < erreur_min alors mémoriser
              {
              la combinaison du collage de d_j sur r_i
              le coefficient d'échelle beta_2(j) optimal
              le coefficient de décalage beta_1(j) optimal
              j_min = j
              erreur_min = erreur
              }
          }
     stocker  {
              l'indice j_min
              la combinaison du collage de d_j_min sur r_i
              le coefficient d'échelle beta_2(j_min) optimal
              le coefficient de décalage beta_1(j_min) optimal
              }
     }
```

TABLE a - Algorithme de codage.

1. cette notation est indroduite dans le chapitre 3

Nous supposons que la triangulation R (adaptée au contenu de l'image) est composée de N blocs destination $\mathbf{r}_i$, et que la triangulation (régulière) D contient Q blocs source $\mathbf{d}_j$. L'algorithme consiste à associer à chacun des blocs $\mathbf{r}_i$ le bloc $\mathbf{d}_j$ qui minimise l'erreur d entre la fonction des niveaux de gris du bloc $\mathbf{r}_i$ et celle du bloc $\mathbf{d}_j$ transformé par ω.

On peut dès à présent noter que la transformation massique qui réalise le collage du bloc source $\mathbf{d}_j$ sur le bloc destination $\mathbf{r}_i$ est la même que celle utilisée par Jacquin (section 3.3.2) et Fisher (section 3.3.4). Elle n'est composée que de deux coefficients : le coefficient de décalage $\beta_1^{(i)}$ et le coefficient d'échelle $\beta_2^{(i)}$.

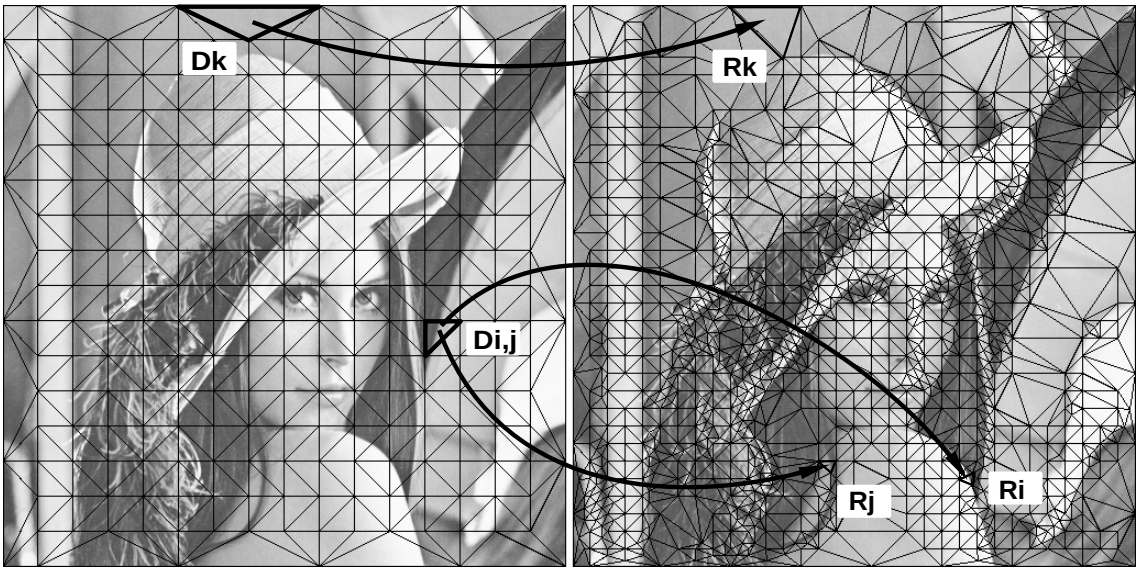


FIG. 41 - Calcul de la transformation fractale. La partition de gauche (D) contient les blocs source $\mathbf{r}$ et la partition de droite (R) contient les blocs destination $\mathbf{r}$.

5.2.2 Initialisation : calcul des triangulations

Nous avons vu que le schéma de compression classique fondé sur des blocs rectangulaires recherche, pour un bloc $\mathbf{r}$ de la partition R, l'élément $\mathbf{d}$ au travers d'un ensemble de blocs disposés sur un ou plusieurs réseaux réguliers de points (voir figure 26, page 84). Les blocs sources sont aussi contraints à être de taille supérieure à celle des blocs destination d'un facteur 2 ou 3 [64].

La qualité du codage est d'autant meilleure que le nombre de blocs source considérés au cours du codage est élevé. Il est en effet raisonnable de penser que si l'on compare un bloc destination avec un très grand nombre de blocs source, l'image transformée $W(A)$ peut être très proche de l'image originale A. Cette amélioration est cependant faite au détriment du temps de calcul de la transformation fractale, et la rend

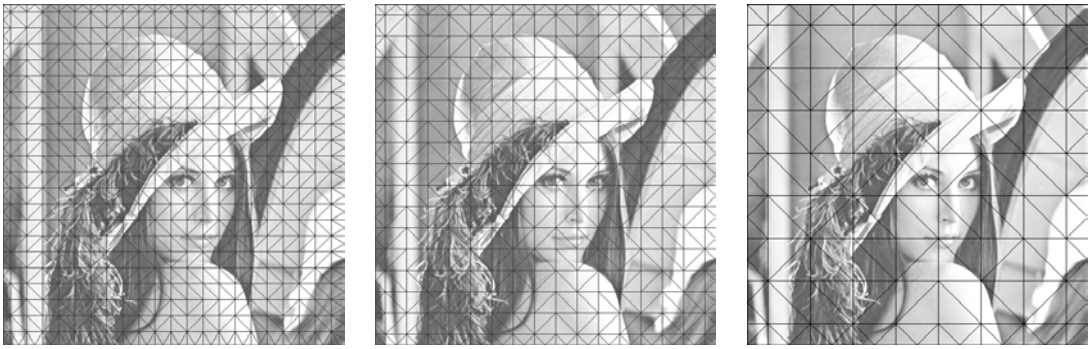
en pratique impossible (selon la qualité désirée, le temps de codage d'une image 512×512 sur une machine séquentielle peut varier de quelques secondes jusqu'à plus de 48 heures !).

Notre méthode diffère de ce schéma principalement pour les deux raisons suivantes :

1. les blocs destination et source sont de formes triangulaires
2. les triangles source sont recherchés parmi une triangulation régulière de l'image, appelée triangulation D.

Le point 2 permet de réduire sensiblement le nombre de triangles $\mathbf{d}$ considérés. Ceux-ci sont de taille constante et ne se recouvrent pas (figure 42). Le choix d'une telle partition D permet de réduire le temps de codage en limitant le nombre de comparaisons inter-blocs mais ne permet pas de minimiser l'erreur $d(A, W(A))$. Nous montrerons que la qualité du codage est cependant acceptable car une partition D régulière recouvre une variété suffisamment importante de régions différentes. Il serait possible d'utiliser une deuxième partition D' de manière à doubler le nombre de blocs sources disponibles. Cette solution a été testée mais s'est révélée être peu efficace si l'on compare les temps de calculs par rapport à la qualité de l'image décodée.

La triangulation R est quant à elle adaptée au mieux au contenu de l'image puisque le nombre de triangles $\mathbf{r}_i$ fixe le taux de compression. Nous présenterons et comparerons dans les paragraphes qui suivent divers résultats de compression obtenus à partir des triangulations de Delaunay décrites dans le chapitre 4.



(a) Pas = 24 (b) Pas = 32 (c) Pas = 64

FIG. 42 - Partitions régulières D.

5.2.3 Utilisation du théorème du collage

Supposons que l'opérateur W est contractant. D'après le théorème du collage [8], l'erreur entre l'image A et l'attracteur $\mathcal{A}_t$ de l'opérateur W est bornée de la façon

suivante :

$$d(A, \mathcal{A}_t) \leq \frac{1}{1-s}\, d(A, W(A)) \tag{44}$$

Le fait de minimiser $d(A, W(A))$ ne diminue qu'une partie de la borne supérieure. L'autre terme dépend du facteur de contraction s de W, qui est fonction des coefficients d'échelle $\beta_2^{(n)}$ de chacune des transformations élémentaires ω_n de W. Ce facteur est souvent mal maîtrisé. Il doit cependant ne pas être trop proche de la valeur 1 pour que la borne supérieure ne soit pas trop grande.

Supposons maintenant que l'opérateur W est finalement contractant et a pour exposant de contraction l'entier n. Cette propriété est souvent autorisée en pratique puisqu'elle permet d'améliorer l'approximation des blocs destination $\mathbf{r}_n$ en imposant moins de contraintes sur les coefficients d'échelle $\beta_2^{(n)}$. L'erreur $d(A, \mathcal{A}_t)$ est dans ce cas bornée par l'expression suivante :

$$d(A, \mathcal{A}_t) \leq \frac{1}{1-s}\, d(A, W^{on}(A)) \tag{45}$$

où s est le facteur de contraction de W^{on}. Il est montré [60] que $d(A, W^{on}(A))$ est elle-même bornée selon l'expression suivante :

$$d(A, W^{on}(A)) \leq \frac{1-\sigma^n}{1-\sigma}\, d(A, W(A)) \tag{46}$$

où σ est le facteur de Lipschitz de W.
L'équation (45) peut être utilisée si le facteur s et l'entier n sont connus. En pratique, ces deux valeurs ne sont pas maîtrisées. De plus il est impossible de minimiser l'erreur $d(A, W^{on}(A))$ puisque cela nécessiterait d'itérer un grand nombre de fois l'opérateur W qui lui-même est en cours de calcul. La solution qui consiste à minimiser l'erreur $d(A, W(A))$ dans l'équation (46) est donc sous-optimale. Malgré toutes ces suppositions, il est vérifié expérimentalement que l'erreur finale entre l'image originale A et l'attracteur $\mathcal{A}_t$ est faible lorsque l'erreur $d(A, W(A))$ est faible.

5.2.4 Calcul de la transformation fractale

La transformation élémentaire ω transformant un bloc source $\mathbf{d}$ en un bloc destination $\mathbf{r}$ s'écrit :

$$\begin{aligned} \omega(\mathbf{d}) &\equiv \omega(x_i, y_i, f(x_i, y_i)) \\ &= (x'_i, y'_i, f(x'_i, y'_i)) \\ &= (v(x_i, y_i), \beta_2\ f(x_i, y_i) + \beta_1), \end{aligned}$$

où $f(x_i, y_i)$ est la luminance du pixel de coordonnées (x_i, y_i) à l'intérieur du bloc source $\mathbf{d}$, $f(x'_i, y'_i)$ est la luminance du pixel de coordonnées (x'_i, y'_i) à l'intérieur du bloc destination $\mathbf{r}$, et v dénote la transformation spatiale d'un triangle $\mathbf{d}$ sur un triangle $\mathbf{r}$. Les coefficients β_2 et β_1 contrôlent respectivement le contraste et la luminosité de la fonction des niveaux de gris.

5.2.5 Comparaison des triangles : transformation spatiale

Le schéma classique de compression basé sur des blocs destination carrés $\mathbf{r}$ de taille B^2 et des blocs source carrés $\mathbf{d}$ de taille $4B^2$ est le plus simple qui soit : un pixel $r_{i,j}$ du bloc $\mathbf{r}$ est comparé à la valeur $\hat{f}(x_i, y_i)$ obtenue en moyennant 2×2 pixels connexes dans le bloc source $\mathbf{d}$. La position de l'ensemble des quatre pixels à l'intérieur du carré est fonction de l'isométrie discrète considérée. Lorsqu'aucune rotation et symétrie par rapport aux axes centrés verticaux, horizontaux et diagonaux n'est réalisée, $\hat{f}(x_i, y_j)$ est donnée par l'expression suivante [85] :

$$\hat{f}(x_i, y_j) = \frac{\beta_2}{4}\left(f(x_k, y_l) + f(x_k, y_{l+1}) + f(x_{k+1}, y_l) + f(x_{k+1}, y_{l+1})\right) + \beta_1$$

où (x_k, y_l) sont les coordonnées du pixel $d_{k,l}$ dans le bloc $\mathbf{d}$. La transformation affine n'est pas calculée explicitement puisque les formes et les tailles respectives des blocs sont connues. L'approche est un peu plus compliquée lorsque les blocs sont rectangulaires, mais le calcul explicite des transformations spatiales n'est également pas nécessaire.

Notre approche, basée sur des blocs source et destination triangulaires, sous-entend l'utilisation de transformations spatiales pour comparer le contenu des triangles puisque ceux-ci sont *a priori* de forme et d'orientation quelconques sur le support de l'image. Pour cela, nous rappelons ci-après les principales propriétés de la transformation affine 2D.

Transformation affine

Une transformation affine $v : \mathbb{R}^2 \to \mathbb{R}^2$ s'écrit de la manière suivante :

$$v\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = \mathbf{L}\mathbf{x} + \mathbf{t} \qquad (47)$$

où a, b, c, d, e, et f sont des réels. La transformation v opère sur le vecteur $\mathbf{x}$ une transformation linéaire $\mathbf{L}$ suivie d'une translation spécifiée par le vecteur $\mathbf{t}$. Une telle transformation permet de déformer un système de coordonnées initial en un autre système de coordonnées. Elle possède six degrés de liberté.

L'espace transformé a pour vecteurs de base les vecteurs (a, c) et (b, d) et pour origine le vecteur (e, f). La figure 43 montre que la transformation affine induit des changements d'échelle, des translations, des rotations, et des "cisaillements". Elle préserve les lignes parallèles et les distances relatives entre les points. Elle peut donc par exemple transformer un triangle en un autre triangle ou un rectangle en un parallélogramme. Il est à noter que des déformations plus complexes telles que la transformation d'un carré en un quadrilatère quelconque peuvent se faire à l'aide de transformations projectives ou bilinéaires. Nous verrons dans le chapitre 6 l'utilisation de ces transformations pour la compression par fractales utilisant des

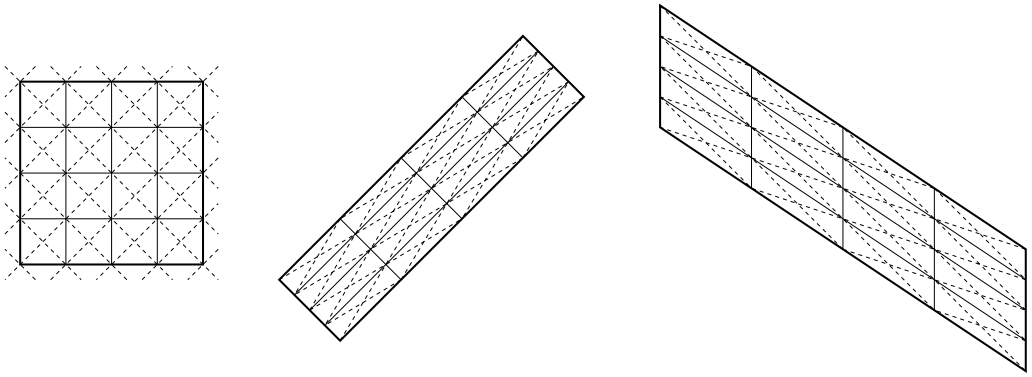


FIG. 43 - Transformations affines 2D du bloc carré.

quadrilatères.

Le calcul des six coefficients de la transformation affine est effectué en considérant trois points dans l'espace de départ et trois points dans l'espace d'arrivée. La déformation d'un triangle source peut donc être calculée à partir de ses trois sommets et des trois sommets du triangle destination. On obtient de cette façon six équations à six inconnues données sous forme matricielle par l'expression suivante :

$$\begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \end{bmatrix} \begin{pmatrix} a \\ b \\ e \\ c \\ d \\ f \end{pmatrix} = \begin{pmatrix} x'_0 \\ x'_1 \\ x'_2 \\ y'_0 \\ y'_1 \\ y'_2 \end{pmatrix}. \tag{48}$$

Nous rappelons que le rapport de la surface du triangle source (transformé par v) sur la surface du triangle destination est égal au module du déterminant de la matrice L (équation 47).

Échantillonnage du bloc source

Le calcul de l'erreur $d(\mathbf{r}, \omega(\mathbf{d}))$ considère l'ensemble des pixels inclus dans le triangle $\mathbf{r}$ et l'échantillonnage du triangle $\mathbf{d}$ par la transformation spatiale v^{-1} (chaque pixel r_i de $\mathbf{r}$ est comparé au pixel le plus proche de son antécédent par la transformation spatiale v). Si le bloc $\mathbf{r}$ contient M pixels, l'erreur d est donnée par :

$$d(\mathbf{r}, \omega(\mathbf{d})) = \sum_{i=1}^{M} \Big[f(x'_i, y'_i) - \beta_2 \, f(v^{-1}(x'_i, y'_i)) - \beta_1 \Big]^2 \tag{49}$$

où (x'_i, y'_i) sont les coordonnées d'un pixel d'indice i dans le bloc destination $\mathbf{r}$.

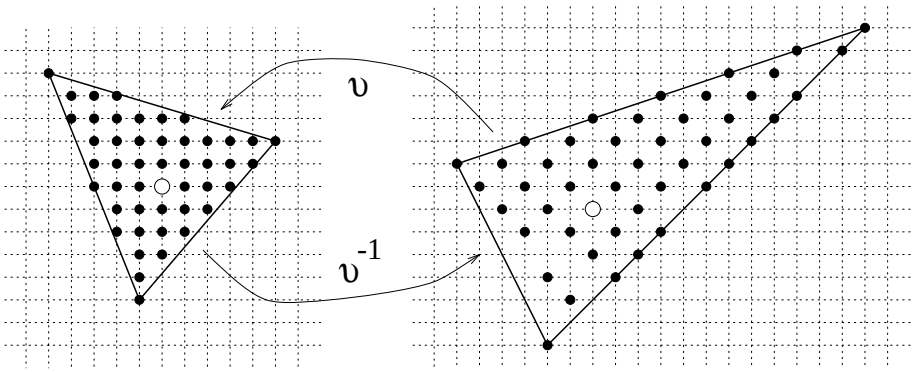


FIG. 44 - Comparaison d'un triangle destination et d'un triangle source.

Cette méthode simple de comparaison des blocs triangulaires par échantillonnage spatial donne de bons résultats à condition que le rapport des surfaces des deux blocs soit inférieur à une valeur limite que nous fixons en pratique égale à 10. La figure 45 montre quelques exemples d'associations de triangles **r** et **d** après calcul de la transformation fractale. Le parcours des pixels dans un bloc destination se fait ligne après ligne. Les lignes associées sont visibles sur le triangle source lorsqu'il est plus grand que le triangle destination : il est dans ce cas sous-échantillonné. Dans le cas contraire, il est sur-échantillonné (triangles noirs sur la figure 45).

FIG. 45 - Relations affines entre des triangles destination et leurs triangles source associés : illustration de l'échantillonnage des blocs source.

5.2.6 Contraction de la transformation fractale

Le facteur de contraction s de l'opérateur W dépend des coefficients d'échelle s_i $(i = 1 \ldots N)$ associés aux N transformations ω_i. Jacobs et *al.* montrent que si $s = \sup\{s_i\}$ est strictement inférieur à 1 alors la transformation fractale est finalement contractante, ceci étant vrai en considérant l'erreur quadratique comme mesure de similarité entre les blocs. Nous vérifions expérimentalement, tout comme beaucoup d'autres chercheurs [60], que la transformation fractale reste finalement contractante si la valeur maximale des coefficients $\{s_i\}$ est inférieure à 1.6.

5.2.7 Étude de la transformation massique

Nous rappelons que l'erreur $d(\mathbf{r}, \omega(\mathbf{d}))$ entre le bloc destination $\mathbf{r}$ et son approximation $\hat{\mathbf{r}}$ calculée à partir du bloc source $\mathbf{d}$ est donnée par :

$$d(\mathbf{r}, \omega(\mathbf{d})) = \sum_{i=1}^{M} \Big[f(x'_i, y'_i) - \beta_2 \, f(v^{-1}(x'_i, y'_i)) - \beta_1 \Big]^2 \qquad (50)$$

où (x'_i, y'_i) sont les coordonnées d'un pixel d'indice i dans le bloc destination $\mathbf{r}$. Les coefficients d'échelle β_2 et de décalage β_1 optimaux sont calculés de façon à annuler les deux dérivées partielles de l'erreur d par rapport à β_2 et β_1. Cette méthode est équivalente à celle proposée par Fisher et Lundheim qui considèrent la transformation massique comme une combinaison linéaire de deux vecteurs et qui annulent deux produits scalaires (voir section 3.3.4).

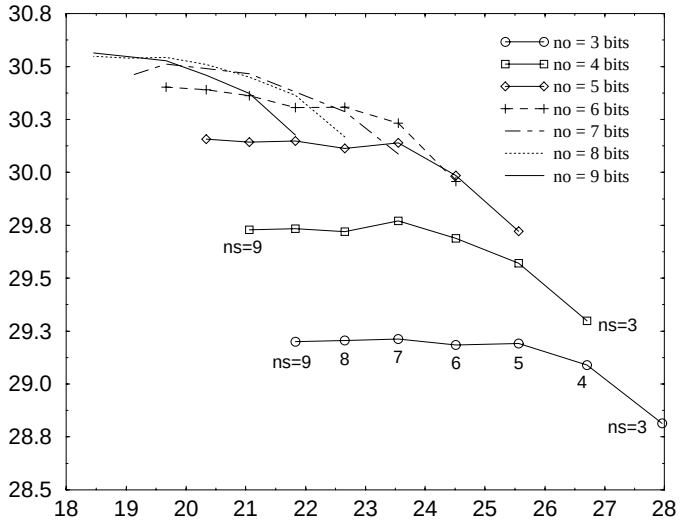


FIG. 46 - Rapport signal à bruit fonction du taux de compression. Les coefficients d'échelle et de décalage sont respectivement quantifiés sur "ns" et "no" bits. L'image Lena 512×512 est compressée sur une partition composée d'approximativement 3500 triangles.

Les coefficients sont ensuite quantifiés séparément à l'aide d'un quantificateur scalaire uniforme, avant d'être réintroduits dans l'expression (50). Ceci permet de prendre en compte les erreurs de quantification lors du calcul des similarités entre blocs. La figure 46 présente différents résultats de décompression obtenus en quantifiant les coefficients de décalage et d'échelle respectivement sur n_o et n_s bits. La configuration optimale qui donne un taux de compression ainsi qu'un rapport signal à bruit maximal se situe en haut à droite du plan. Elle est égale à $n_s = 5$ et $n_o = 6$ bits pour ce cas particulier. Nous vérifions expérimentalement que la combinaison $n_s = n_o = 6$ bits est dans la plupart des cas l'optimale.

Analyse des coefficients d'échelle et de décalage

La figure 47 montre les suites de coefficients β_2 et β_1 associés à chacun des blocs destination après codage de l'image Lena 512×512. On note dès à présent que le coefficient d'échelle β_2 est centré sur la valeur zéro alors que la moyenne des coefficients de décalage β_1 est positive. Nous discuterons de cette remarque dans la section 6.2. Le coefficient β_2 quantifié uniformément sur n_s bits appartient à un ensemble fini de 2^{n_s} valeurs. Le coefficient β_1 est quant à lui quantifié sur n_o bits de façon à vérifier la relation $0 \leq \overline{\beta}_2 f + \beta_1 \leq 255$ dans laquelle f représente l'amplitude d'un pixel comprise entre 0 et 255.

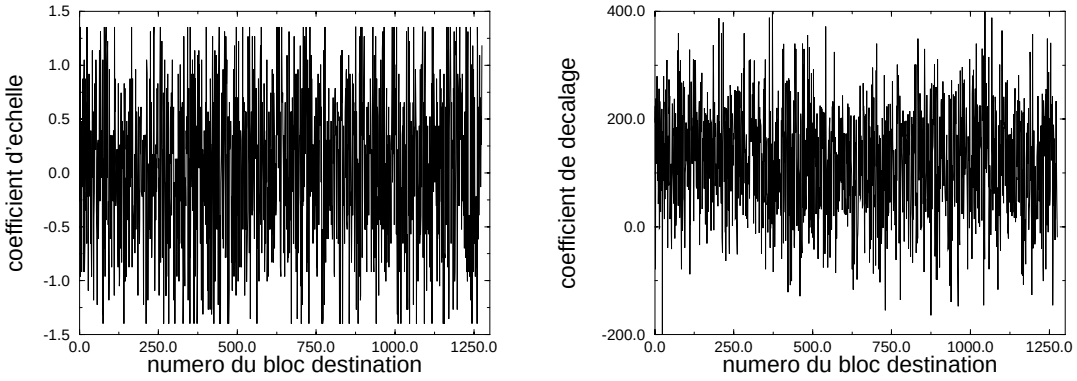
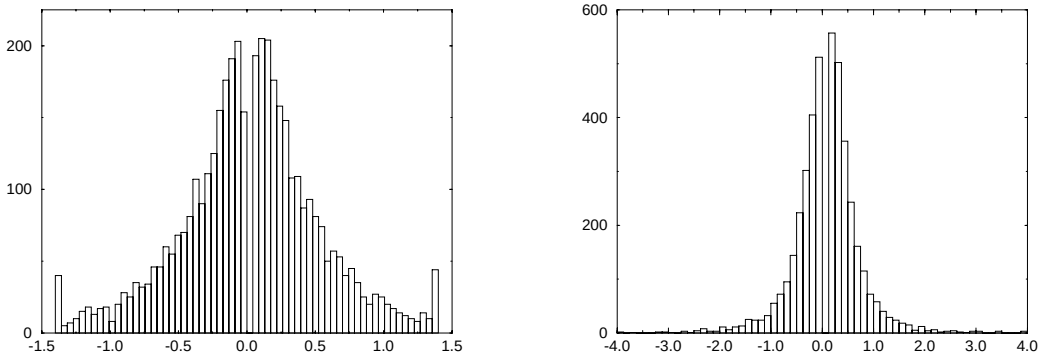


FIG. 47 - Coefficients d'échelle β_2 à gauche et de décalage β_1 à droite associés après codage à 1276 triangles de l'image Lena 512×512.

La figure 48 montre les histogrammes des coefficients d'échelle β_2 quantifiés sur 6 bits, dont le module est borné par la valeur 1.4 (48a) et 4.0 (48b) lors du codage de l'image Lena. L'histogramme (non représenté) est plus plat lorsque la valeur maximum de β_2 est égale à 1 puisque les nombreux coefficients écrêtées (pics sur les valeurs 1 et -1) ont pour effet d'"écraser" l'histogramme. On remarque aussi que l'histogramme est plus pointu lorsque le module de β_2 est borné par la valeur 4.0. Øien [128] utilise dans ce cas un quantificateur optimal de Lloyd-Max (voir section 2.7.2) sur 5 ou 6 bits mais indique qu'il n'améliore pas sensiblement ses



(a) $\max \beta_2 = 1.4$, entropie $= 5.37$ (b) $\max \beta_2 = 4.0$, entropie $= 4.12$

FIG. 48 - Histogramme des coefficients d'échelle calculés sur l'image Lena et quantifiés sur 6 bits.

résultats par rapport à ceux obtenus avec un quantificateur uniforme.
La figure 49 montre que les coefficients d'échelle et de décalage sont corrélés. Il serait dans ce cas intéressant d'utiliser un quantificateur vectoriel (cette étude n'a pas été réalisée dans le cadre de cette thèse). Nous expliquerons dans la section 6.2 la raison de la dépendance entre les deux coefficients β_1 et β_2.

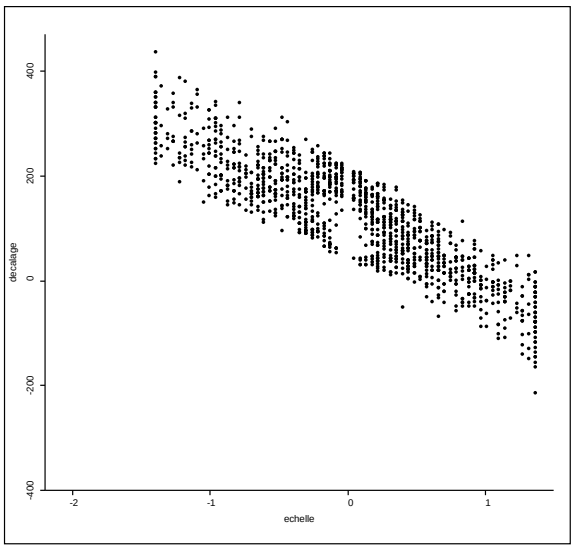


FIG. 49 - Coefficients de décalage en fonction des coefficients de d'échelle.

En anticipant sur la section 5.4 qui traite du décodage, le tableau 1 montre que le fait de borner le module des coefficients d'échelle β_2 par une valeur comprise entre 1.4 et 2 améliore la qualité de l'image reconstruite à taux de compression constant. La qualité est inférieure lorsque le module de β_2 n'est pas contraint, et on vérifie

	Rapport signal à bruit (PSNR)
$\max \beta_2 = 1.0$	31.90 dB
$\max \beta_2 = 1.4$	32.19 dB
$\max \beta_2 = 2.0$	32.24 dB
$\max \beta_2 = 4.0$	32.15 dB

TAB. 1 - Rapports signal à bruit entre l'image originale Lena 512×512 et l'image décodée en limitant le module du coefficient d'échelle à différentes valeurs lors du codage.

expérimentalement que dans ce cas le décodage itératif est plus lent. Pour ces raisons, nous choisissons dans notre application de limiter le module du coefficient d'échelle β_2 à la valeur 1.4. Ce dernier est quantifié uniformément sur 6 bits.

Erreur entre l'image originale et son collage

La différence entre l'image originale et son collage, centrée sur le niveau de gris 128 et réhaussée à l'aide d'un égalisateur d'histogramme montre que les erreurs de collage sont concentrées sur les frontières et les structures principales de l'image (figure 50).

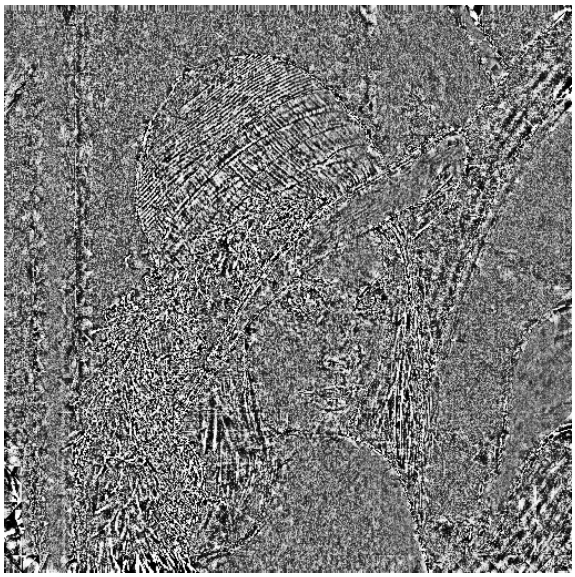


FIG. 50 - Image d'erreur entre l'image originale Lena 512×512 et l'image originale transformée par la transformation fractale. Le rapport signal à bruit entre l'image originale et l'image transformée est égal à 33.07 dB.

Ceci s'explique par le fait que les similarités inter-blocs ne sont pas exactes. Un bloc destination incluant un contour est généralement mis en correspondance avec un bloc source traversé par un contour mais de manière différente (l'orientation

du contour dans les deux triangles n'est pas rigoureusement la même). Une région texturée a de la même manière très peu de chance d'être associée à une autre région texturée identique [152].

Temps de calcul

Les temps de calculs sont importants pour le codage d'une image. Ils dépendent directement du nombre N de triangles dans la partition R ainsi que du nombre Q de blocs considérés dans la partition D, puisque ND comparaisons inter-blocs sont réalisées. Selon la qualité de reconstruction désirée, le codage d'une image de taille 512×512 sur un ordinateur Silicon Graphics INDIGO varie de 5 minutes à 1 heure CPU. La version actuelle de notre algorithme de compression est plus coûteuse en temps de calcul que la même version basée sur un partitionnement carré ou rectangulaire [60]. Ceci s'explique par le fait que la comparaison des triangles implique le calcul d'une transformation affine pour chaque pixel du triangle destination. Une version améliorée de l'algorithme consisterait à mémoriser à l'avance les associations entre pixels de chaque triangle de façon à les retrouver rapidement lors des comparaisons.

5.3 Compression de l'image

Le codage de la transformation fractale consiste à :

- mémoriser la partition R adaptée au contenu de l'image originale;
- mémoriser l'information nécessaire au codage des N transformations ω composant l'opérateur W.

Il est important de trouver le bon compromis entre la quantité d'information codant la partition R et celle associée aux N transformations locales. De manière générale, lorsque la partition est adaptée au contenu de l'image, le nombre de blocs, et par conséquent le nombre de transformations à mémoriser, diminue. Lorsque la partition est trop complexe, sa mémorisation annule le gain réalisé par la diminution du nombre de transformations. Ceci a été remarqué par Fisher [59] et par Reusens [137] qui proposaient respectivement une partition triangulaire et une partition polygonale.

5.3.1 Codage des transformations locales

Le codage d'une transformation ω_n comprend :

- l'indice du bloc $\mathbf{d}_{\alpha(n)}$ à transformer (associé au bloc $\mathbf{r}_n$). Lorsque le nombre de triangles source dans la partition régulière D est inférieur à 1024, l'indice est codé sur $n_i = 10$ bits;

- la manière de coller un bloc sur un autre bloc (il existe 6 solutions pour coller un triangle **d** sur un triangle **r**. La combinaison est codée sur $n_c = 3$ bits;
- le facteur d'échelle $\beta_2^{(n)}$ quantifié uniformément sur $n_s = 6$ bits;
- le facteur de translation $\beta_1^{(n)}$ quantifié uniformément sur $n_o = 6$ bits;

Les coefficients a_n, b_n, c_n, d_n, e_n et f_n des transformations ω_n ne sont pas mémorisés puisqu'ils peuvent être calculés par le décodeur qui connaît la forme des blocs source et destination. En pratique, une transformation ω_n est donc codée sur $10+3+6+6 = 25$ bits.

5.3.2 Codage des partitions

Le codage de la partition triangulaire R dépend de l'algorithme utilisé pour sa construction. Lorsque le partitionnement est contraint par les contours de l'image (les triangles s'appuient contre les contours) il est nécessaire de mémoriser les positions horizontales et verticales des sommets de chacun des blocs. Considérons une image de taille 512×512 partitionnée en N triangles de Delaunay (le nombre de sommets est approximativement égal à $\frac{N}{2}$). Si $N = 5000$, l'ensemble des transformations ω_n est codé sur $25 * 5000 = 125000$ bits. La partition est quant à elle codée sur $2 * 9 * 2500 = 45000$ bits, et représente 26% de l'information totale mémorisée.

Lorsque la triangulation est calculée à l'aide des procédures de division-fusion ou de division seule, il est possible de la coder très efficacement. La procédure consiste dans ce cas à ne mémoriser que les étapes de division et/ou de fusion, en partant d'un partitionnement régulier connu du codeur et du décodeur. Lors des étapes de division, l'ajout d'un point sur le barycentre d'un triangle non-homogène est codé sur un bit. Lors de l'étape de fusion, la suppression d'un sommet est de la même manière codée sur un bit. A titre d'exemple, nous détaillons ci-après le codage du partitionnement de l'image Femme de taille 256×256 illustré sur la figure 35. La triangulation régulière contient 255 éléments. La triangulation issue de la première étape de division contient 623 triangles : elle est codée sur 255 bits. La triangulation issue de la seconde étape de division, composée de 1012 triangles, est codée sur $255 + 623 = 878$ bits. Elle contient 492 sommets. La triangulation finale obtenue au terme de l'étape de fusion (composée de 893 blocs) est codée par une suite de $878 + 492 = 1370$ bits qu'il est encore possible de compacter à l'aide d'un codeur par longueur de séquence (section 2.6.7). Sans codage entropique, l'information liée à la partition représente 5.78% de l'information totale mémorisée.

5.4 Décodage de l'image

L'algorithme de décodage est résumé dans la table b. Il consiste à itérer l'opérateur W, après reconstruction des partitions R et D, en partant d'une image quelconque

f_0. Le niveau de gris $f_k(x'_i, y'_i)$ du pixel inclus dans le bloc $\mathbf{r}$, à la k^e itération de l'opérateur W est donné par :

$$f_k(x'_i, y'_i) = \beta_2 \; f_{k-1}(v^{-1}(x'_i, y'_i)) + \beta_1 \qquad \forall \; (x'_i, y'_i) \in \mathbf{r}. \tag{51}$$

Le résultat converge en pratique au terme de 5 à 10 itérations vers l'image reconstruite, attracteur de la transformation fractale (figure 51). Le nombre d'itérations dépend essentiellement du rapport de surface entre les blocs de la partition D et ceux de la partition R. Au cours des itérations, le collage d'un bloc $\mathbf{d}_{\alpha(n)}$ (recouvrant plusieurs blocs $\mathbf{r}$) sur le bloc $\mathbf{r}_n$ lui correspondant a pour effet de diminuer la taille des détails à l'intérieur des blocs de la partition R. Plus la différence de taille entre les blocs $\mathbf{d}$ et les blocs $\mathbf{r}$ est importante, plus la convergence est rapide. Un compromis doit être fait car dans ce cas les blocs sont moins semblables, et le point fixe de l'opérateur W (l'image reconstruite) peut être trop éloigné de l'image originale. L'étude théorique d'un cas particulier (partitionnement carré) est donnée dans [130].

```
reconstruire la triangulation R
reconstruire la triangulation D
pour itération allant de 1 à n_itérations
    {
    pour i allant de 1 à N (i = indice du triangle r_i de R)
        {
        lire
           {
           l'indice j du triangle d_j associé à r_i
           la combinaison du collage de d_j sur r_i
           le coefficient d'échelle beta_2(j) optimal
           le coefficient de décalage beta_1(j) optimal
           }
        calculer w(d_j)
        }
    }
```

TABLE b - Algorithme de décodage.

La figure 52 illustre le fait qu'un pixel reconstruit après 15 itérations de la transformation fractale est relié à 15 autres pixels et que ces derniers sont distribués sur tout le support de l'image.

(a) itération 1 (b) itération 2

(c) itération 3 (d) itération 4

(e) itération 5 (f) itération 10

(g) itération 15 (h) image Lena 512 × 512

FIG. 51 - Décodage de l'image Lena. itération 15 : $T_c = 11.2 : 1$, PSNR = 32.29 dB

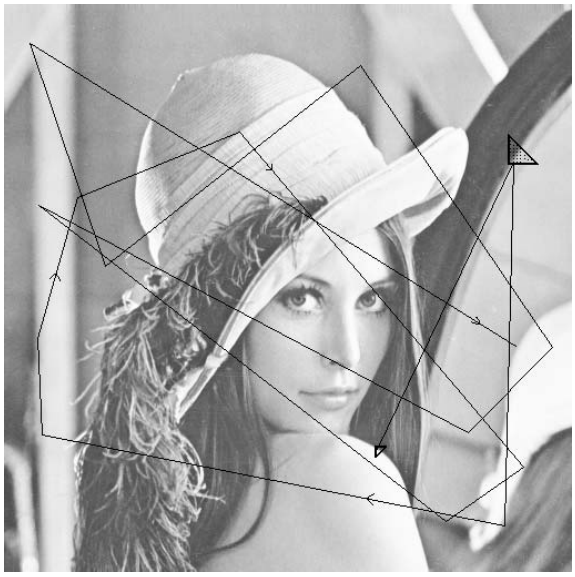


FIG. 52 - Suivi de la reconstruction d'un pixel au cours du décodage (15 itérations). Le pixel r à l'intérieur du petit triangle destination est issu de la transformation du pixel d inclu dans le triangle source de plus grande taille. Le pixel d est lui-même à l'intérieur d'un autre triangle destination. Il est associé à un autre pixel d'un autre bloc source. En continuant ce raisonnement, nous remontons jusqu'au quinzième pixel.

Zoom fractal

Nous citerons en guise d'introduction le cas d'un logiciel qui parcourt un ensemble d'images en les affichant au fur et à mesure sur un écran d'ordinateur. Les images affichées sont généralement de petite taille pour permettre un parcours rapide. L'intérêt du zoom est dans ce cas évident et peut être fait de différentes façons. L'approche la plus simple consiste à dupliquer quatre fois chaque pixel de l'image de manière à quadrupler sa surface. L'image résultant d'un tel sur-échantillonnage présente des effets "d'escalier" qui peuvent devenir gênants lorsque le facteur de zoom augmente. D'autres méthodes classiques approximent la valeur de points dans l'image, de coordonnées non entières, à partir d'un ensemble de pixels voisins. L'approximation est faite dans ce cas par interpolation bilinéaire ou bicubique [91], en considérant que la fonction des niveaux de gris de l'image est continue.

Le codage par fractales est quant à lui dit indépendant de la résolution de l'image originale. Il est fondé sur l'hypothèse que deux zones similaires de l'image à une résolution donnée restent similaires à une résolution supérieure ou inférieure. Une image codée peut donc théoriquement être reconstruite à n'importe quelle résolution [132]. Nous avons vu que l'algorithme de codage ne mémorise pas explicitement les transformations spatiales associées à chacun des blocs destination puisque celles-ci sont retrouvées par le décodeur qui connaît les deux partitions R et D et donc la forme géométrique des blocs source et destination. Il est ainsi possible de reconstruire

une image carrée de surface n fois supérieure à celle de l'image originale en multipliant par $\frac{n}{2}$ les dimensions horizontales et verticales des deux partitions puis en itérant la transformation fractale sur la nouvelle partition R. L'exemple de la figure (53) illustre le zoom d'un facteur 2 sur l'image Lena : la transformation fractale a été calculée sur l'image originale de taille 512×512 permettant un taux de compression de 12.3:1. La figure montre une partie extraite de l'image reconstruite sur une partition R de taille 1024×1024 à partir de la même transformation fractale. On peut considérer que la nouvelle image est compressée d'un facteur égal à $4 \times 12.3 : 1 = 49.2 : 1$.

FIG. 53 - Image Lena 512×512 décodée avec un facteur d'échelle égal à 2 : illustration d'une partie de l'image reconstruite 1024×1024.

Cette propriété n'est intéressante qu'à condition que le codage soit de bonne qualité (au détriment du taux de compression et du temps de codage). Dans le cas contraire, les défauts visibles sur l'image reconstruite sont amplifiés par le zoom et deviennent très vite gênants.

L'indépendance du code fractal vis à vis de la résolution permet en outre, lorsque l'image originale est carrée, de la reconstruire sur un support rectangulaire. Il suffit pour cela d'appliquer des facteurs d'échelle différents sur les coordonnées horizontales et verticales des sommets des triangles, de construire les triangulations R et D puis de calculer la transformation fractale sur les nouveaux blocs.

5.5 Résultats

Dans cette section, nous présentons différents résultats obtenus à partir d'une triangulation de Delaunay adaptée au mieux à l'image. La partition est calculée en

utilisant l'algorithme de division-fusion détaillé dans la section 4.4.3. Les taux de compression sont calculés selon la procédure décrite dans la section 5.3 sans utiliser de codeur entropique.

Histogrammes des images au cours du décodage.

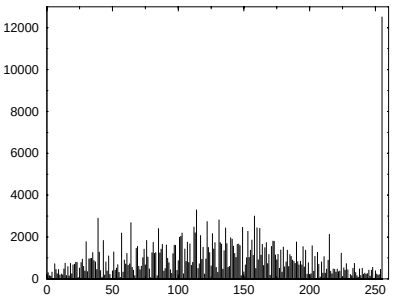
La figure 54 montre l'évolution des histogrammes calculés lors d'une reconstruction itérative de l'image Lena (le décodage est initialisé sur une image noire). L'histogramme de la première image reconstruite présente un pic sur la valeur 255 puisqu'un nombre important de triangles sont "saturés". Ceci s'explique par le fait que beaucoup de coefficients de décalage $\beta_1^{(n)}$ sont supérieurs à 255 (voir figure 47) et qu'ils sont les seuls à intervenir dans l'équation de décodage (51) lorsque $k = 1$ et $f_0(x, y) = 0$ quel que soit (x, y). L'histogramme (b) montre que le nombre de pixels saturés diminue. Ce dernie s'annule au bout de la quatrième itération.

Évolution de deux profils au cours du décodage

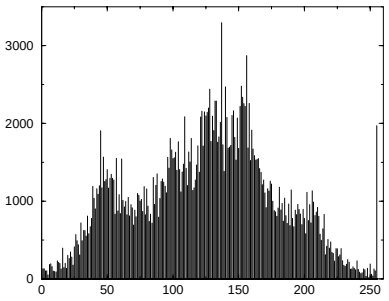
Les images reconstruites par fractales selon l'approche développée dans cette thèse sont peu bruitées (il en est de même lorsque le codeur utilise des blocs carrés ou rectangulaires). Cette remarque se confirme en comparant les profils (g) des images décodées avec les profils de l'image Lena, illustrés sur les figures 55 et 56.

Rapport signal à bruit en fonction du taux de compression

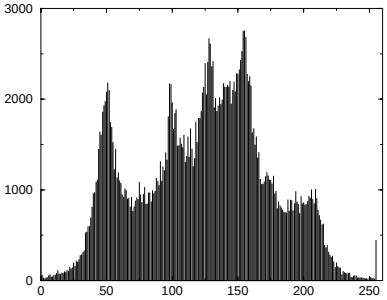
La figure 57 illustre des résultats de décodage des images Lena et Cornouaille à différents taux de compression. Le fond de l'image Lena est visuellement homogène. Une triangulation grossière de cette partie permet d'augmenter le taux de compression tout en préservant une qualité de reconstruction acceptable. Ceci n'est pas possible sur l'image Cornouaille qui contient beaucoup plus de détails (câbles, écritures, textures fines) et qui ne peuvent pas être reconstruits lorsque le taux de compression désiré est trop élevé.
Le résultat "$T_c = 11.7 : 1$, PSNR $= 32.73dB$" de la figure 57 obtenu après reconstruction de l'image Lena est à rapprocher de celui de la figure 50 montrant l'erreur entre l'image originale et la même image transformée par W. Le rapport signal à bruit entre ces deux images était de 33.07 dB.



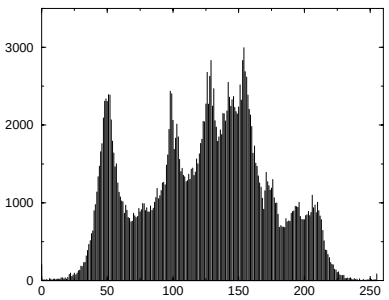
(a) Histogramme itération 1



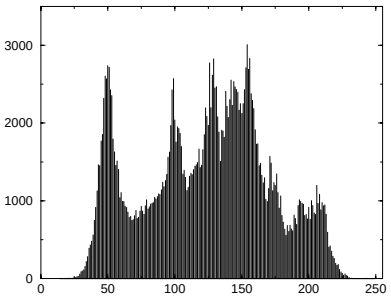
(b) Histogramme itération 2



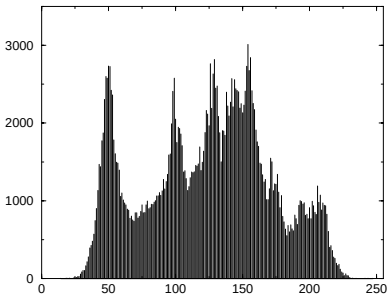
(c) Histogramme itération 3



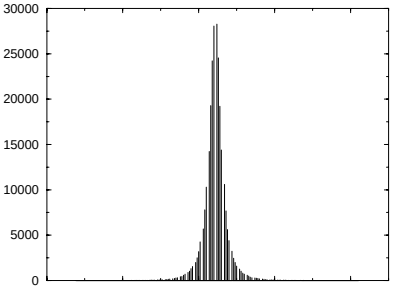
(d) Histogramme itération 4



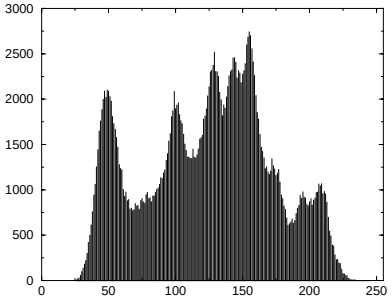
(e) Histogramme itération 10



(f) Histogramme itération 15



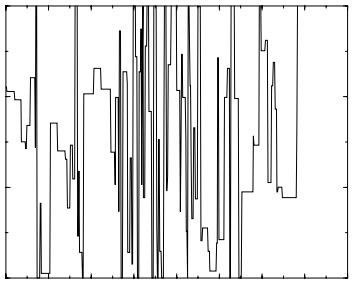
(g) Histogramme de la différence entre l'image Lena et l'image reconstruite



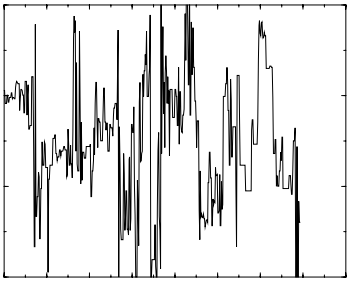
(h) Histogramme de l'image Lena

FIG. 54 - Histogrammes des images reconstruites au cours des itérations du décodage.

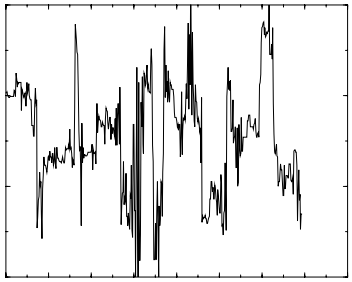
(a) Profil



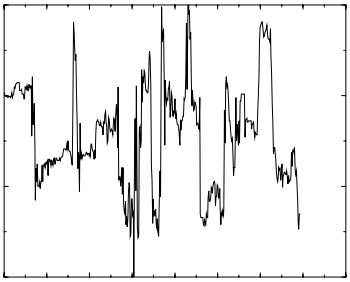
(b) Profil itération 1



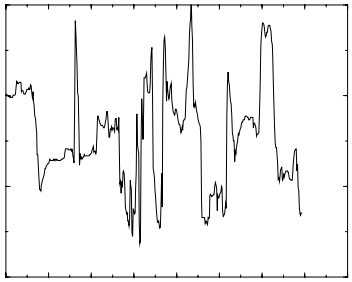
(c) Profil itération 2



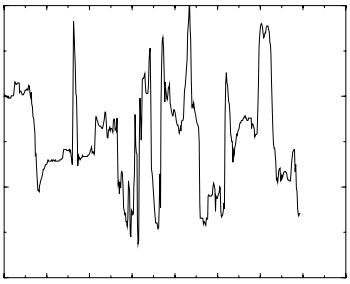
(d) Profil itération 3



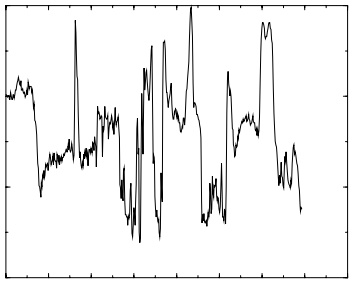
(e) Profil itération 4



(f) Profil itération 10



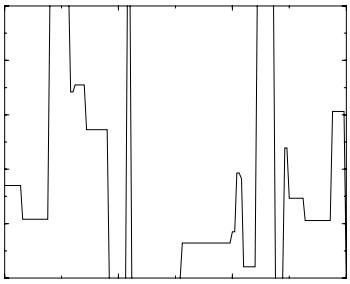
(g) Profil itération 15



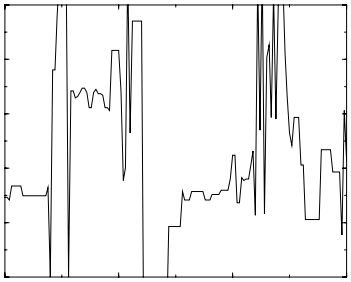
(h) Profil sur l'image Lena

FIG. 55 - Évolution d'un profil au fur et à mesure du décodage itératif.

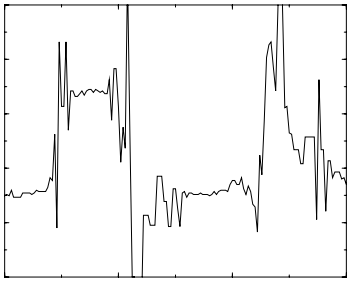
(a) Profil



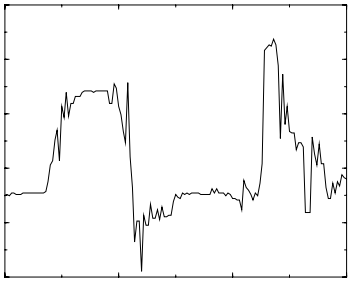
(b) Profil itération 1



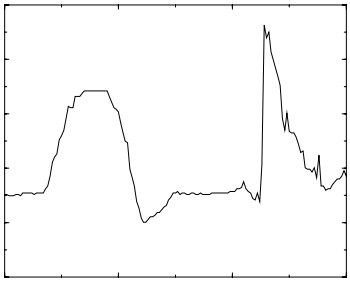
(c) Profil itération 2



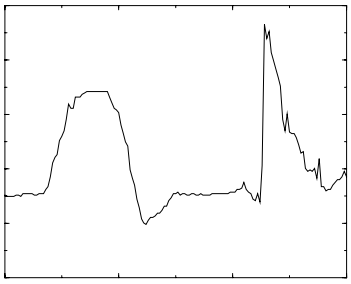
(d) Profil itération 3



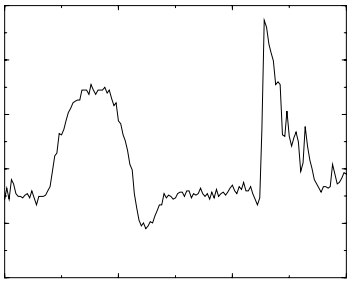
(e) Profil itération 4



(f) Profil itération 10



(g) Profil itération 15



(h) Profil sur l'image Lena

FIG. 56 - Évolution d'un profil au fur et à mesure du décodage itératif.

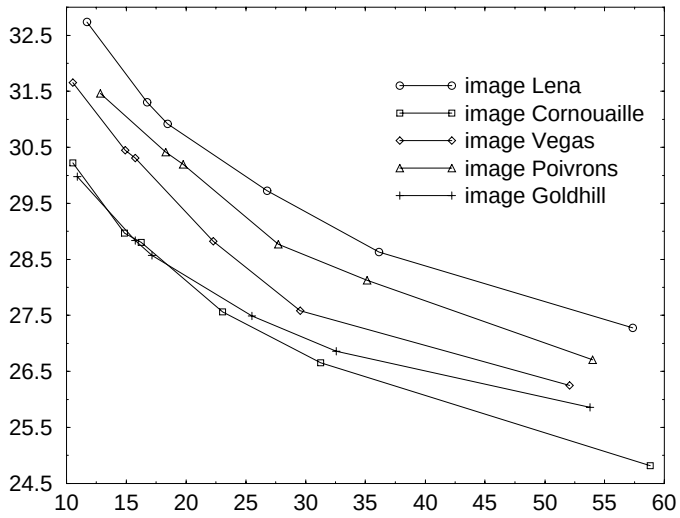


FIG. 57 - Rapport signal à bruit fonction du taux de compression. Décodage des images Lena et Cornouaille 512×512.

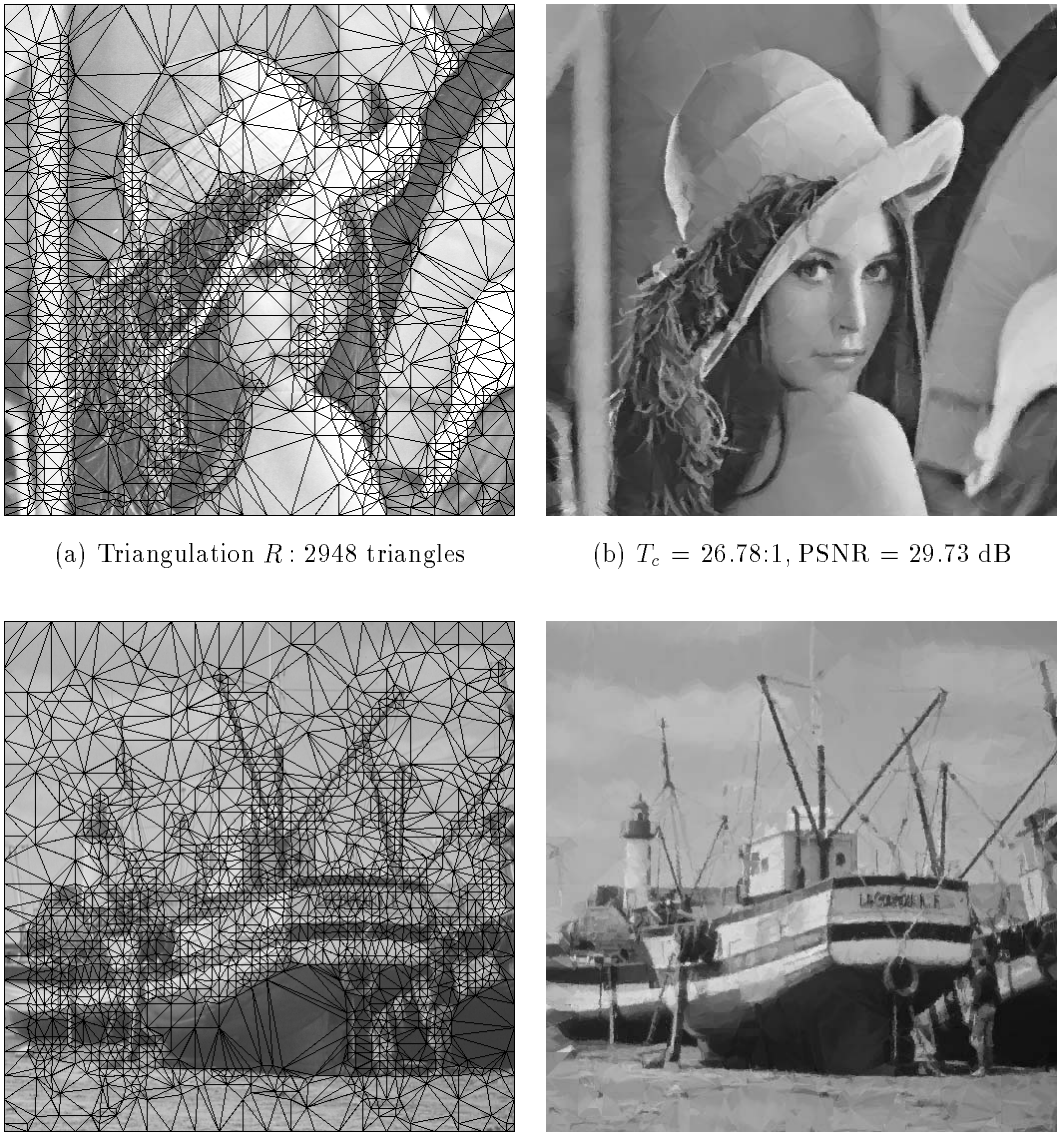
La figure 58 montre des exemples de reconstruction des images Lena et Cornouaille ainsi que leur triangulation associée. Les détails fin inclus dans des triangles de trop grande taille sont perdus.

Influence du nombre et de la taille des blocs source

	Rapport signal à bruit	Taux de compression
$D = 908$	23.1 dB	52.0
$D = 468$	27.4 dB	54.5
$D = 214$	27.0 dB	56.5

TAB. 2 - Influence du nombre de blocs source sur la qualité du codage : la partition R contient 1468 blocs.

Nous montrons à l'aide du tableau 2 que le nombre de blocs source considérés lors du codage influe sur la qualité du décodage. La partition D étant régulière, le nombre de blocs source impose leur taille. Nous vérifions expérimentalement que la taille des blocs source doit être supérieure en moyenne d'un facteur 5 à celle des blocs destination pour que l'image reconstruite soit de bonne qualité. Dans le cas particulier du tableau 2, les 908 blocs sources sont trop petits par rapport aux blocs 1468 blocs destination. Lorsque la partition D contient 214 blocs, ceux-ci sont trop peu nombreux pour permettre une bonne approximation des blocs destination.
On remarque aussi que le taux de compression augmente lorsque le nombre de blocs



(a) Triangulation R : 2948 triangles

(b) $T_c = 26.78{:}1$, PSNR $= 29.73$ dB

(c) Triangulation R : 3434 triangles

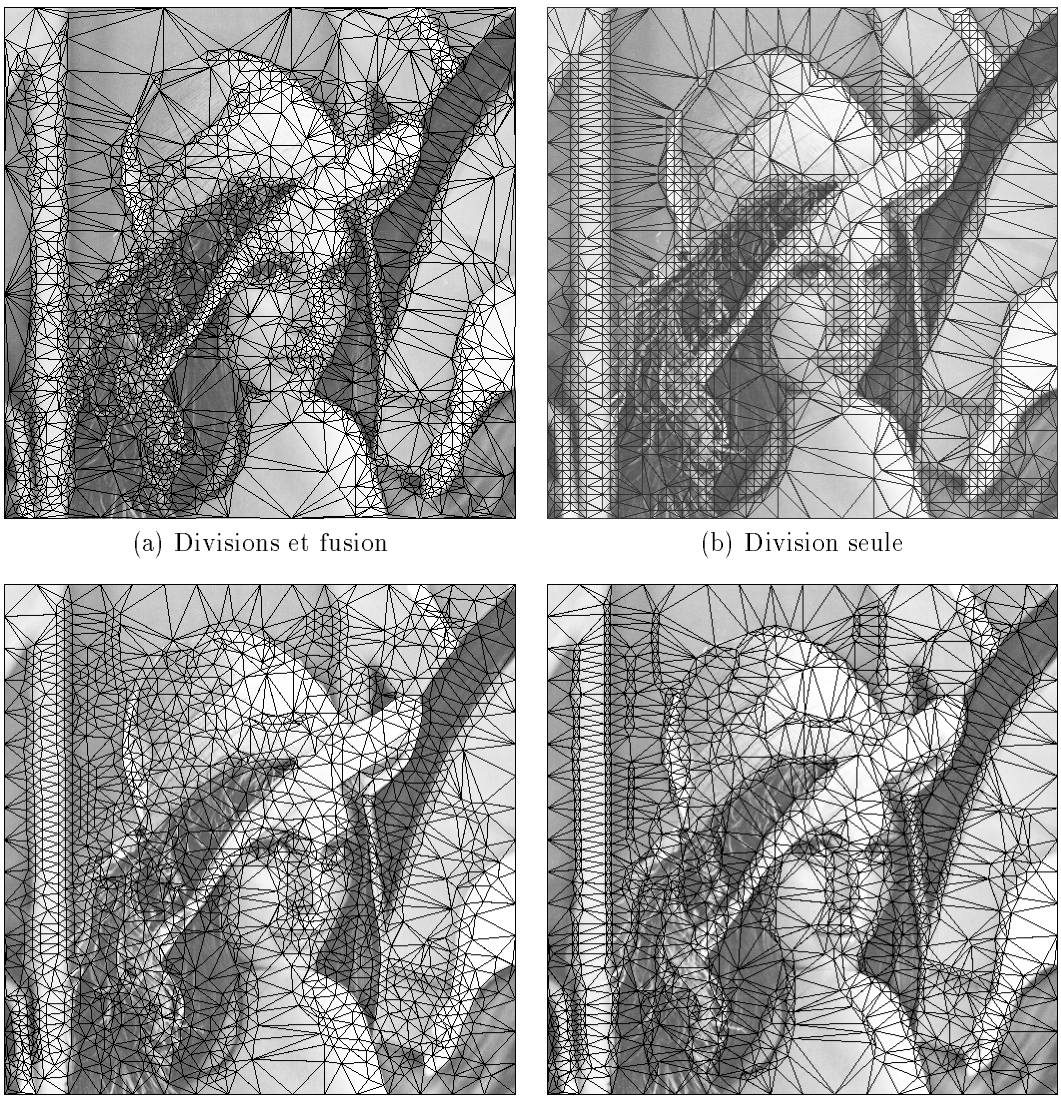
(d) $T_c = 23.074$, PSNR $= 27.56$ dB

FIG. 58 - Reconstruction des images Lena et Cornouaille 512×512.

source diminue. Ceci vient du fait que l'indice d'un bloc source est codé sur un nombre décroissant de bits.

5.6 Comparaisons des triangulations

La figure 59 résume les différentes triangulations étudiées dans le chapitre 4. La



(a) Divisions et fusion

(b) Division seule

(c) Triangulation contrainte : triangles contre les contours

(d) Triangulation contrainte : triangles "à cheval" sur les contours

FIG. 59 - Résumé des differents partitionnements de Delaunay étudiés.

différence entre les deux triangulations contraintes dépend uniquement du rapport des distances d_1 et d_2 indiquées sur la figure 39. Dans le cas 59(c), d_2 est supérieure à d_1. Dans le cas 59(d), d_2 est inférieure à d_1 de façon à ce que les triangles recouvrent les contours.

Nous nous proposons d'étudier les avantages et les inconvénients des triangulations en les regroupant dans les deux classes suivantes :

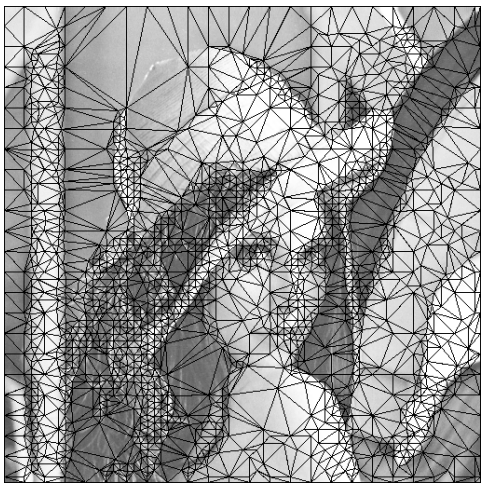
- les triangulations 59(a), (b) et (d) appartiennent à la même classe. Dans chacune des trois, les triangles de petite taille recouvrent les frontières de l'image

ainsi que les textures;

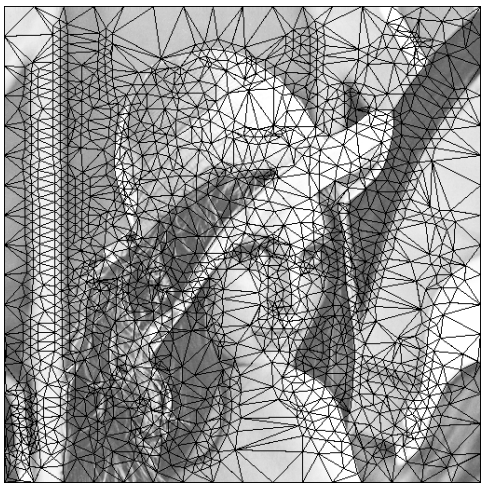
- la triangulation contrainte 59(c) contient un plus grand nombre de blocs homogènes.

La figure 60 montre un triangle source accompagné des triangles destination auxquels il est associé, au terme de la phase de codage. Dans ce cas particulier, le triangle source est au bord d'une frontière de l'image. La figure 60(c) montre que le bloc source est mis en correspondance avec plusieurs blocs destination incluant une frontière du même type. Elle montre également qu'un nombre important d'autres blocs plus homogènes lui sont associés. On vérifie expérimentalement que dans ce dernier cas, les modules des coefficients d'échelle des transformations massiques sont très faibles et ont pour effet "d'aplatir" le contenu des triangles source. La triangulation contrainte réduit le nombre de blocs pour lesquels le collage peut sembler visuellement faux. Les mises en correspondances dans la figure 60(d) sont plus vraisemblables dans le sens où le bloc source ressemble visuellement aux blocs destinations auxquels il est associé.
Les erreurs de reconstruction dans l'image 61(a) sont très visibles sur le chapeau de Lena puisque l'algorithme de division-fusion considère un critère d'homogénéité sur les niveaux de gris des triangles. La surface trop homogène du chapeau est ainsi recouverte par des triangles de grande taille. Mises à part ces erreurs, la qualité visuelle moyenne de l'image 61(a) est meilleure que celle de l'image 61(b) reconstruite sur la triangulation containte : la première est plus contrastée. Ceci peut s'expliquer en analysant les histogrammes des coefficients d'échelle (figures 60(e) et (f)) associés à l'ensemble des triangles destination. Lorsque la triangulation contient un grand nombre de blocs homogènes (triangulation contrainte) les modules des coefficients d'échelle sont plus faibles et réduisent ainsi sensiblement la variance du contenu des blocs source décimés. La supériorité en terme de qualité visuelle des images reconstruites sur une triangulation obtenue par division-fusion est confirmée par la mesure du rapport signal à bruit (figure 61). Ceci est aussi vérifié expérimentalement sur d'autres triangulations plus fines, et d'autres images [45]).

En terme de compression, il est important de noter que le codage de la triangulation contrainte obtenue après détection des contours est moins efficace que le codage des étapes de division-fusion. Ceci implique que pour un taux de compression constant, la triangulation contrainte possède moins de blocs.



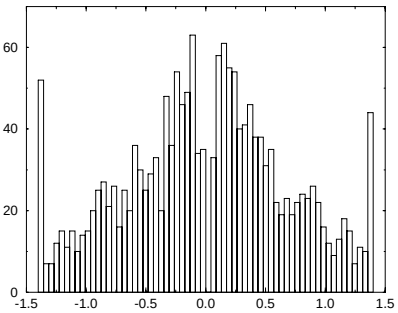
(a) 3424 triangles destination



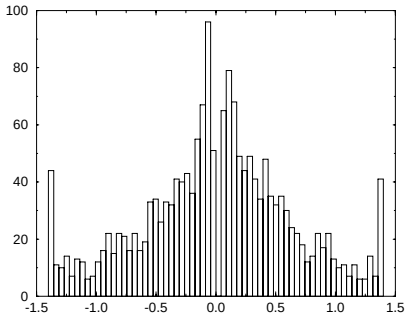
(b) 3490 triangles destination

(c) Source (en pointillé)-destinations

(d) Source (en pointillé)-destinations



(e) Histogramme du coefficient d'échelle



(f) Histogramme du coefficient d'échelle

FIG. 60 - Comparaison des deux types de triangulation. À gauche : division-fusion, à droite : contrainte

(a) Triangulation obtenue par division-fusion : 3424 triangles, PSNR = 30.23 dB

(b) Triangulation contrainte : 3490 triangles, PSNR = 29.52 dB

FIG. 61 - Reconstruction de l'image Lena à partir de deux triangulations différentes.

5.7 Conclusion

Nous avons décrit dans ce chapitre notre approche de compression fractale fondée sur la triangulation de Delaunay. Le partitionnement, calculé sur un ensemble initial de points, est adapté au contenu de l'image. Nous avons montré comment coder efficacement une telle triangulation, et comment l'utiliser pour le calcul de la transformation fractale. Cette dernière diffère des schémas classiques pour la principale raison suivante : le rapport de taille entre les blocs source et les blocs destination n'est pas constant (il peut de plus être supérieur ou inférieur à 1). Les triangles source sont sous-échantillonnés ou sur-échantillonnés lors des comparaisons. Nous montrerons dans le chapitre 6 que le nombre de triangles source réellement utilisés pour calculer la transformation fractale peut être très faible. Nous comparerons aussi nos résultats à ceux obtenus à partir des autres partitionnements géométriques introduits dans le chapitre 4.

Chapitre 6

Améliorations et schémas hybrides

6.1 Introduction

Ce chapitre présente diverses améliorations apportées au schéma de compression-décompression (hybrides ou non) par différents chercheurs. Nous proposons aussi deux méthodes visant à optimiser notre propre algorithme fondé sur un partitionnement triangulaire. Nous montrons dans la section 6.4 comment augmenter le taux de compression pour une qualité de reconstruction désirée. Nous montrons aussi dans la section 6.5 comment accélérer la phase de codage d'une image en réduisant le nombre de triangles source à prendre en considération lors du calcul de la transformation fractale. Nous concluons en présentant différents résultats de décodage obtenus à partir des principales méthodes décrites dans ce chapitre.

6.2 Accélération du décodage : orthogonalisation

Nous considérons ici que le vecteur $\hat{\mathbf{r}}$, égal à l'approximation du vecteur destination $\mathbf{r}$ est donné par :

$$\hat{\mathbf{r}} = \beta_2 \mathbf{b}_2 + \beta_1 \mathbf{b}_1. \tag{52}$$

$\hat{\mathbf{r}}$ appartient au sous-espace vectoriel engendré par les vecteurs $\mathbf{b}_1$ et $\mathbf{b}_2$, où $\mathbf{b}_1$ est un vecteur de base constant ($\mathbf{b}_1 = [1, 1, \ldots, 1]^T$), et $\mathbf{b}_2$ est un vecteur extrait de l'image à coder. Chacun des vecteurs $\hat{\mathbf{r}}$, $\mathbf{r}$, $\mathbf{b}_1$ et $\mathbf{b}_2$ est de dimension B^2.

Le calcul des coefficients β_2 et β_1 soulève des problèmes théoriques et pratiques délicats concernant :

1. l'assurance de la convergence du décodeur vers un point proche du collage de l'image originale,
2. et le maintient des valeurs de luminance des images reconstruites au fur et à mesure des itérations de décodage à l'intérieur de la plage :
   $[\mathrm{lum}_{min} = 0, \ldots, \mathrm{lum}_{max} = 255]$.

Les coefficients β_2 et β_1 optimaux sont classiquement calculés de manière à obtenir la meilleure approximation $\hat{\mathbf{r}}$ du bloc $\mathbf{r}$, au sens de la norme L_2. Celle-ci s'écrit :

$$\min_{\beta_1, \beta_2} \|\mathbf{r} - \beta_1 \mathbf{b_1} - \beta_2 \mathbf{b_2}\|^2. \tag{53}$$

Lorsque le calcul n'est pas contraint, le coefficient d'échelle β_2 peut atteindre des valeurs élevées largement supérieures à 1, pouvant entraîner une divergence lors du décodage itératif. Pour pallier à ce problème, les auteurs s'imposent expérimentalement de borner tous les modules des coefficients d'échelle β_2 par une valeur $\beta_{2_{max}}$ égale à 1.6 ([83]). Cette limite supérieure, tout en étant nécessaire, rend les approximations sous-optimales, et influe sur la rapidité de convergence du décodeur. Si $\beta_{2_{max}}$ est trop petite (inférieure à 0.5), le décodage est rapide, mais ne converge pas vers le

point fixe désiré puisque les collages sont de mauvaise qualité. Si au contraire $\beta_{2_{max}}$ est supérieure à 1.6, la convergence n'est plus assurée.

Le deuxième point cité ci-dessus vise à réduire la propagation des erreurs au cours du décodage itératif. Il est important de noter que la reconstruction à l'itération n d'un pixel met généralement à contribution n autres pixels répartis sur l'image entière[1]. L'erreur de reconstruction d'une partie de l'image influe donc sur le reste de l'image.
Les coefficients β_2 et β_1 sont calculés pendant la phase de codage pour approximer un bloc dont les valeurs de luminance sont incluses dans $[0, \ldots, 255]$, et sont dépendants l'un de l'autre : par exemple, lorsque le coefficient d'échelle est égal à -1, le coefficient de décalage peut être de forte valeur pour ramener la luminance des pixels intérieurs au bloc dans la plage autorisée. Lors du décodage, les mêmes coefficients

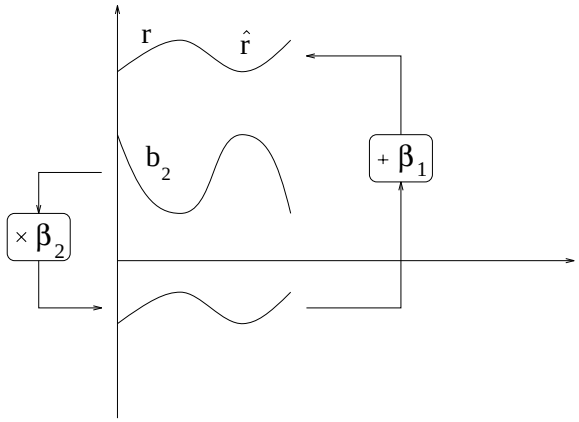


FIG. 62 - Approximation du bloc $\mathbf{r}$ à partir du bloc $\mathbf{b}_2$: illustration sur un profil.

β_2 et β_1 sont appliqués sur les pixels d'une image quelconque servant à initialiser le décodeur. Ils peuvent donc retourner à la première itération des valeurs de luminance sortant largement de l'intervalle permis. Ces erreurs sont en pratique ramenées dans l'intervalle $[0, \ldots, 255]$ par troncature, mais elles se propagent à la deuxième itération sur des zones quelconques de l'image. Elles tendent ensuite à s'estomper puisque le coefficient d'échelle β_2 est en module inférieur à un. Un bloc erroné $\hat{\mathbf{r}}$ (dont les luminances sortent largement de l'intervalle $[0, \ldots, 255]$), de grande taille, reste cependant longtemps visible au cours des itérations, et impose à l'utilisateur d'itérer plus longtemps la transformation fractale pour obtenir un résultat correct. Ceci sera illustré dans la section 6.2.3 lorsque nous appliquerons les résultats de la section 6.2.2 aux triangles de Delaunay.

6.2.1 Décorrélation des coefficients de la transformation massique

Barthel propose dans [17] une solution permettant de restreindre les valeurs de luminance de l'image reconstruite à la deuxième itération dans l'intervalle $[0, \ldots, 255]$.

1. Ceci est montré sur l'exemple de la figure 52 du chapitre 5

Il utilise pour cela la transformation massique suivante :

$$\hat{\mathbf{r}} = \gamma_2(\mathbf{b}_2 - \mu_2) + \gamma_0\mu_2 + \gamma_1\mathbf{b}_1, \qquad \gamma_0 = 0.5 \tag{54}$$

dans laquelle μ_2 est la valeur moyenne du bloc source décimé $\mathbf{b}_2$. Le coefficient d'échelle γ_2 agit seulement sur la dynamique du bloc $\mathbf{b}_2$. Le coefficient γ_0 est fixé à 0.5, de manière à ce que la valeur $\gamma_0\mu_2$ puisse être considérée comme une prédiction grossière de la valeur moyenne du bloc destination $\mathbf{r}$. Le coefficient de décalage γ_1 est de cette façon décorrélé du coefficient γ_2, et de variance plus réduite que celle du coefficient β_1 précédent. L'intérêt de la solution présentée est de limiter les erreurs de dépassement apparaissant à l'issue d'une première itération de la transformation fractale sur une image quelconque.

6.2.2 Orthogonalisation de l'espace de collage

Øien propose dans [128] une solution élégante généralisant la méthode de Barthel. Elle vise à accélérer la phase de décompression par orthogonalisation des vecteurs de base $\mathbf{b}_1$ et $\mathbf{b}_2$ engrendrant le sous-espace vectoriel de collage.
Le deuxième avantage de son approche est de ne pas avoir à imposer de contraintes aux coefficients d'échelle de la transformation fractale.

Orthogonalisation des vecteurs de base

Dans les applications de traitement du signal telles que la compression de données, les vecteurs manipulés sont généralement représentés par une combinaison linéaire de fonctions orthogonales. L'exemple le plus classique est celui de la transformation de Fourier où les fonctions sont des exponentielles complexes.

L'orthogonalisation du bloc décimé $\mathbf{b}_2$ par rapport au vecteur de base constant $\mathbf{b}_1$ du sous-espace vectoriel peut être faite à l'aide de la procédure de Gram-Schmidt. La procédure revient à multiplier le vecteur $\mathbf{b}_2$ par la matrice orthogonalisante $O = \mathbf{I} - \mathbf{b}_1{\mathbf{b}_1}^T$, où $\mathbf{I}$ est la matrice identité de dimension $B^2 \times B^2$. Le vecteur orthogonalisé $\tilde{\mathbf{b}}_2$ est donné par

$$\tilde{\mathbf{b}}_2 = O\mathbf{b}_2.$$

Ceci revient en pratique à enlever la composante de $\mathbf{b}_2$ appartenant au sous-espace engendré par le vecteur $\mathbf{b}_1$, et donc à annuler la valeur moyenne du bloc $\mathbf{b}_2$.
Le nouveau collage, maintenant réalisé dans le sous-espace vectoriel engendré par les vecteurs orthogonaux $\mathbf{b}_1$ et $\tilde{\mathbf{b}}_2$ est donné par :

$$\hat{\mathbf{r}_o} = \alpha_2\tilde{\mathbf{b}}_2 + \alpha_1\mathbf{b}_1.$$

Dans ce cas, les coefficients α_i sont indépendants les uns des autres. G. Øien et S. Lepsøy montrent dans [130] qu'il n'est pas nécessaire d'orthogonaliser explicitement les vecteurs pour calculer les coefficients α_1 et α_2. Ces derniers s'expriment par les

relations :

$$\alpha_1 = \langle \mathbf{r}, \mathbf{b}_1 \rangle = \frac{1}{B^2} \sum_{j=1}^{B^2} r_j,$$

$$\alpha_2 = \frac{\langle \mathbf{r}, \mathbf{b}_2 \rangle - \alpha_1 \langle \mathbf{b}_1, \mathbf{b}_2 \rangle}{\|\mathbf{b}_2\|^2 - \langle \mathbf{b}_1, \mathbf{b}_2 \rangle^2} = \frac{B^2 \sum_{j=1}^{B^2} r_j b_j - \sum_{j=1}^{B^2} r_j \sum_{j=1}^{B^2} b_j}{B^2 \sum_{j=1}^{B^2} {b_j}^2 - \left(\sum_{j=1}^{B^2} b_j \right)^2} \tag{55}$$

dans lesquelles r_j (resp. b_j) sont les pixels intérieurs au bloc $\mathbf{r}$ (resp. $\mathbf{b}_2$). Les coefficients α_1 et α_2 sont liés aux anciens coefficients β_1 et β_2 par les relations :

$$\beta_2 = \alpha_2,$$

$$\beta_1 = \alpha_1 - \langle \mathbf{b}_1, \mathbf{b}_2 \rangle \beta_2.$$

Convergence rapide du décodeur

Øien et Lepsøy montrent qu'en faisant précéder la phase de codage par une orthogonalisation des vecteurs $\mathbf{b}_1$ et $\mathbf{b}_2$, il est possible d'assurer la convergence exacte du décodeur en un nombre fini d'itérations. L'image ainsi reconstruite est identique à celle calculée sans orthogonalisation. Leur démonstration donnée dans [130] est faite sur un cas particulier, pour lequel :

- le partitionnement R est construit sur un quadtree ;
- la décimation d'un bloc source est faite par moyenne des pixels ;
- et chaque bloc source utilisé recouvre un nombre entier de blocs destination.

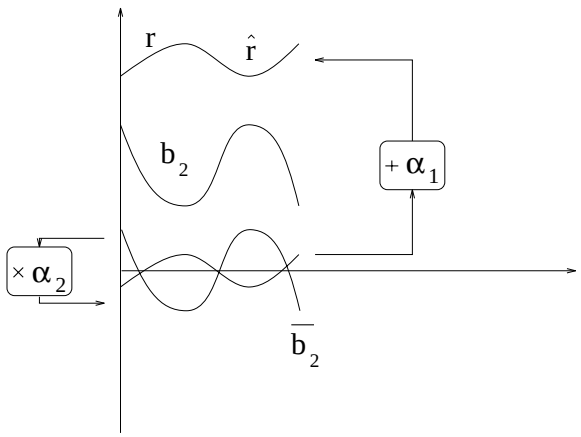


FIG. 63 - Approximation du bloc $\mathbf{r}$ à partir du bloc $\mathbf{b}_2$, avec orthogonalisation de l'espace de collage : illustration sur un profil.

Nous avons vu qu'après orthogonalisation, le coefficient de décalage α_1 (éq. (55)) est égal à la valeur moyenne du bloc destination auquel il est associé. Le code de la transformation contient ainsi une information directement liée au contenu de l'image originale. Ceci est un point important de l'approche puisqu'il peut faciliter les traitements avals visant à reconnaître ou à manipuler le contenu de l'image, directement à partir du code compressé.
L'approche restitue, après la première itération de décodage, une image composée de blocs de valeur moyenne égale à celle des niveaux de gris qu'ils englobent dans l'image originale. L'image ainsi obtenue est donc très proche du point fixe recherché. Chacune des itérations suivantes ajoute de nouveaux détails à l'intérieur des blocs de la partition, tout en laissant la valeur moyenne de ces derniers inchangée.

6.2.3 Orthogonalisation sur la triangulation de Delaunay

Nous appliquons dans ce paragraphe le processus d'orthogonalisation sur les triangles de Delaunay et vérifions ainsi expérimentalement la convergence rapide du décodeur ainsi que la décorrélation des coefficients d'échelle et de décalage.

La démonstration de convergence donnée par Øien et Lepsøy [130] est dans notre cas plus difficile puisque les triangles source recouvrent un nombre non entier de triangles destination. L'orthogonalisation est cependant faite de la même manière et génère un code fractal dans lequel les coefficients de décalage sont égaux aux valeurs moyennes de chacun des triangles de la partition R.

FIG. 64 - Décodage itératif de l'image Lena (avec orthogonalisation).

FIG. 65 - Décodage itératif de l'image Lena (sans orthogonalisation).

On vérifie sur la figure 64 que la première itération de décodage, initialisée sur une image dans laquelle tous les pixels sont de valeur nulle, est composée de blocs homogènes. La figure 67 (à comparer avec la figure 66) confirme qu'après orthogonalisation, les coefficients d'échelle et de décalage sont décorrélés.

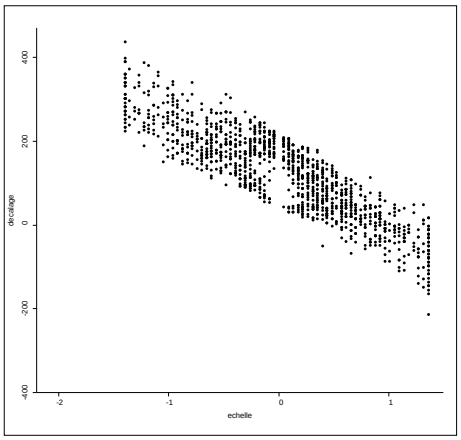


FIG. 66 - Coefficients de décalage en fonction des coefficients d'échelle calculés sans orthogonalisation.

La figure 68 montre que l'ortogonalisation des vecteurs $\mathbf{b}_1$ et $\mathbf{b}_2$ n'améliore pas la qualité des images reconstruites mais accélère la convergence du décodeur. Signès [152] explique géométriquement sur un cas simple la raison pour laquelle l'orthogonalisation n'améliore pas la qualité des images reconstruites.

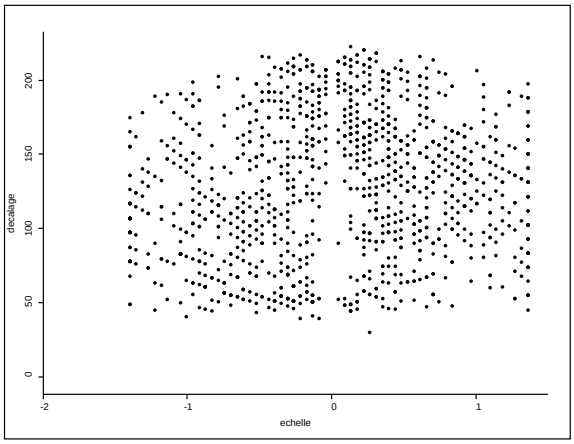


FIG. 67 - Coefficients de décalage en fonction des coefficients d'échelle calculés avec orthogonalisation.

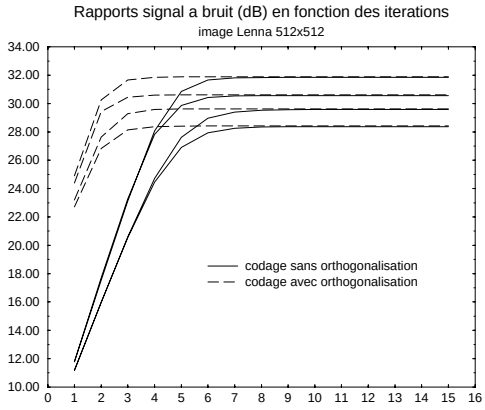


FIG. 68 - Evolution du rapport signal à bruit au cours du décodage itératif de l'image Lena pour quatre taux de compression différents.

6.3 Accélération du codage : recherche du plus proche voisin

D. Saupe propose dans [147] une procédure de codage par fractales de complexité en $O(\log Q)$, où Q est le nombre de blocs source, contrairement aux schémas traditionnels de complexité en $O(Q)$. La procédure classique utilisant une transformation affine dans l'espace des luminances, recherche pour un bloc destination $\mathbf{r} \in \mathbb{R}^{B^2}$, un bloc source $\mathbf{b_2} \in \mathbb{R}^{B^2}$ parmi Q, qui minimise

$$E(\hat{\mathbf{r}}, \mathbf{r}) = \min_{\beta_1, \beta_2} \|\mathbf{r} - \beta_1 \mathbf{b_1} - \beta_2 \mathbf{b_2}\|^2.$$

Le calcul des coefficients optimaux β_1, β_2 ainsi que de l'erreur est coûteux en temps de calculs.

Saupe considère la base orthonormale du sous-espace vectoriel de $\mathbb{R}^{B^2}$, engendrée par les vecteurs normalisés $\mathbf{b_1}$ ($b_1 = \frac{1}{B}(1, \ldots, 1)$), et $\phi(\mathbf{b_2})$. ϕ est l'opérateur de projection rendant $\phi(\mathbf{b_2})$ orthonormal à $\mathbf{b_1}$. Il est montré dans ce cas [147] que l'erreur $E(\hat{\mathbf{r}}, \mathbf{r})$ est proportionnelle à une fonction monotone croissante de la distance D donnée par :

$$D(\hat{\mathbf{r}}, \mathbf{r}) = \min(d(\phi(\mathbf{b_2}), \phi(\mathbf{r})), d(-\phi(\mathbf{b_2}), \phi(\mathbf{r}))).$$

Minimiser $E(\hat{\mathbf{r}}, \mathbf{r})$ revient ainsi à minimiser $D(\hat{\mathbf{r}}, \mathbf{r})$, et donc à *rechercher le plus proche voisin* de $\phi(\mathbf{r})$ parmi les $2Q$ vecteurs $\pm\phi(\mathbf{b_2})$. Différents algorithmes rapides de recherche du voisin le plus proche sont proposés dans la littérature. Dans [66], les auteurs construisent un arbre de dimension B^2 et définissent une méthode de recherche de complexité en $O(\log Q)$. Les résultats donnés dans [147] sont obtenus en fusionnant

- le programme de compression basé sur le quadtree (intégrant une classification des blocs) de Fisher [60] (section 4.2.2),
- le programme de recherche d'une approximation du plus proche voisin de Arya et al. [3], basé sur un arbre de dimension B^2 proposé dans [66].

Ils montrent qu'il est possible de gagner un facteur 1.3 à 11.5 sur le temps de compression, sans trop dégrader la qualité de l'image reconstruite. Le gain dépend bien sûr de la nature de l'image, et du nombre de blocs source considérés.

6.4 Compression sur des triangles et des quadrilatères

Notre approche de compression par fractales repose sur l'exploitation d'une triangulation de Delaunay, adaptée au contenu de l'image. Nous avons vu dans le chapitre 5 que la triangulation adaptée à l'image recouvre plus régulièrement les contours que ne le font les blocs carrés du quadtree (figure 25). Cependant, la construction par la méthode de division et fusion fait apparaître un nombre important de triangles allongés au-dessus des régions homogènes de l'image. Deux triangles ayant en commun deux côtés de grande longueur sont très souvent homogènes et de même valeur moyenne. Nous proposons dans cette section de les rassembler sous forme d'un quadrilatère convexe, de manière à les coder à l'aide d'une seule transformation élémentaire ω_i. Nous montrons dans la suite de ce paragraphe d'une part comment extraire de la triangulation des quadrilatères homogènes et convexes, et d'autre part comment décrire des transformations spatiales non affines, nécessaires à leur codage.

6.4.1 Extraction des quadrilatères convexes

Etant donné deux triangles adjacents semblables (de même niveau de gris moyen), nous nous proposons de les fusionner dans un quadrilatère. Différentes situations peuvent se présenter et nous ne retenons que celles qui forment des quadrilatères convexes en comparant les signes des deux produits vectoriels $v_1 \wedge v_3$ et $v_2 \wedge v_3$ (voir figure 69).

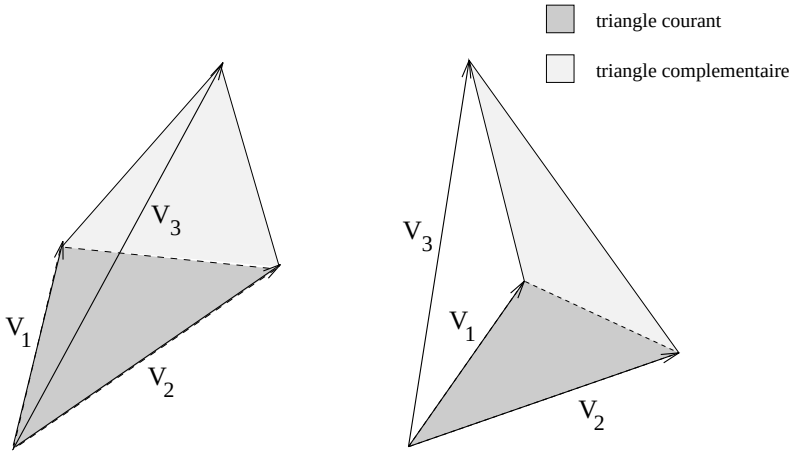


FIG. 69 - Détection des quadrilatères convexes par comparaison des signes des deux produits vectoriels $v_1 \wedge v_3$ et $v_2 \wedge v_3$.

La figure 70 illustre une partition formée de quadrilatères et de triangles qui sera réutilisée dans la section 6.4.3 lorsque nous présenterons nos résultats de décodage à partir d'un partitionnement mixte. Le choix du seuil sur la différence des moyennes de niveaux de gris permet de contrôler le nombre de quadrilatères créés.

6.4.2 Transformation spatiale des quadrilatères

Le codage par fractales implique que nous puissions comparer le contenu de deux quadrilatères source et destination respectivement notés **d** et **r** (figure 71). Pour cela, il nous faut définir une transformation spatiale mettant en correspondance deux à deux les pixels de chacun des quadrilatères de forme quelconque. Ceci ne peut être fait par une transformation affine car celle-ci n'a pas suffisamment de degrés de liberté. Nous présentons dans les deux paragraphes suivants les transformations projectives et bilinéaires 2D souvent utilisées dans les applications nécessitant la déformation de blocs carrés en quadilatères [72]. Nous montrons ensuite les avantages et les inconvénients de chacune d'elles.

Transformation projective

Notre but est ici de déterminer les coefficients d'une transformation projective permettant de déformer un quadrilatère de la partition D de l'image originale en un autre quadrilatère de la partition R calculée sur la même image. Nous présentons

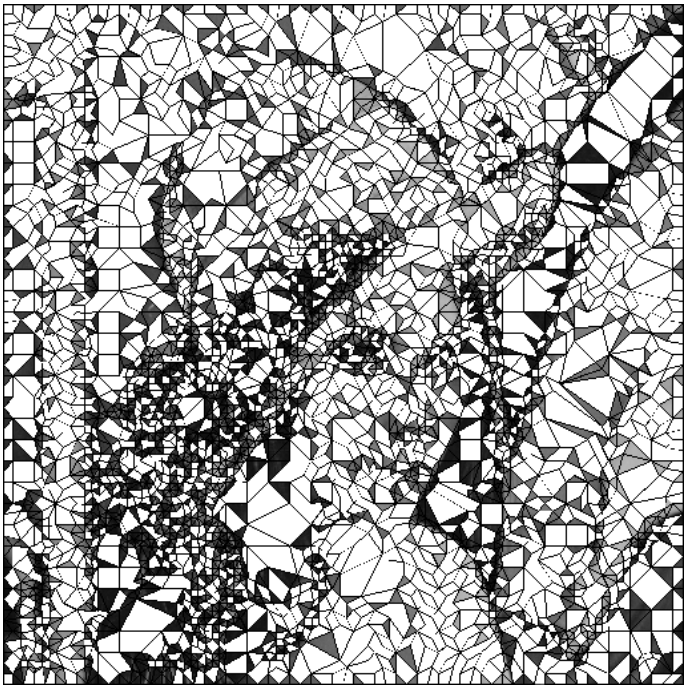


FIG. 70 - Partition formée de 2988 triangles et de 2176 quadrilataires.

FIG. 71 - Deux quadrilatères à comparer.

dans cette partie la géométrie du problème avant de montrer comment déterminer la transformation recherchée. Nous terminons en mettant en exergue les problèmes induits par ce type de transformation.

Considérons un repère tridimensionnel (X, Y, Z) associé à un objet, dont le centre est noté O. Le repère est appelé repère objet. Considérons un autre repère bidimensionnel (x, y) associé à l'image originale, appelé repère image. Ce repère image est centré dans l'image, sur le point O_i. La droite passant par les points O_i et O est perpendiculaire à l'image, et la distance entre O et O_i est égale à F (voir figure 72).

Supposons aussi que le point O est placé derrière le plan image par rapport à l'objet. Les coordonnées de la projection d'un point P de l'objet au temps t_1 dans

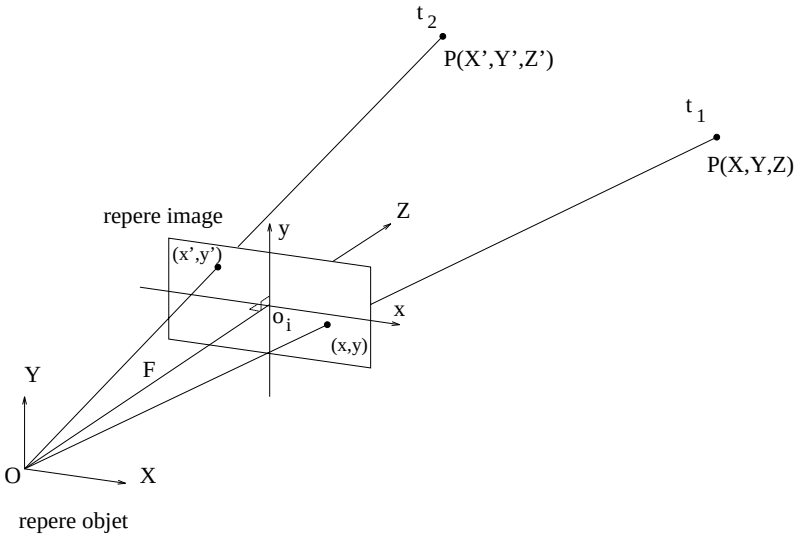


FIG. 72 - Principe de la transformation projective.

le repère image sont données par les équations suivantes :

$$x = F\frac{X}{Z} \tag{56}$$
$$y = F\frac{Y}{Z}$$

En supposant que l'objet tridimensionnel est soumis à une rotation, une translation et une déformation linéaire entre un temps t_1 et un temps t_2, les nouvelles coordonnées du point P au temps t_2 dans le repère objet sont données par :

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = S \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{pmatrix} \tag{57}$$

où $(\Delta X, \Delta Y, \Delta Z)$ codent la translation du point, S est une matrice de déformation linéaire et $(R + I)$ est une matrice de rotation (I = matrice unitaire 3×3). L'équation 57 peut aussi s'écrire sous une formulation affine de la manière suivante :

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{pmatrix}. \tag{58}$$

En considérant maintenant que le point P se déplace sur un plan d'équation :

$$a_p X + b_p Y + c_p Z = 1, \tag{59}$$

on remarque d'après (56) et (59) que :

$$z = \frac{F}{a_p x + b_p y + c_p F}. \tag{60}$$

D'après (56), (58) et (60), les coordonnées de la projection du point P au temps t_2 dans l'image sont données par :

$$\begin{cases} x' = \dfrac{ax + by + e}{gx + hy + 1} \\ \\ y' = \dfrac{cx + dy + f}{gx + hy + 1} \end{cases} \tag{61}$$

Le lecteur intéressé trouvera dans l'article [157] la formulation des huit coefficients a, b, c, d, e, f, g et h en fonction des paramètres de rotation et de translation de l'objet plan ainsi que des paramètres a_p, b_p et c_p.
L'équation (61) définit, pour un jeu de huit coefficients a à h fixés, la projection du repère image (x, y) au temps t_1 sur un autre repère (x', y') dans la même image au temps t_2. La transformation a huit degrés de liberté.
Ce type de transformation peut donc être utilisé pour caractériser la déformation dans l'image d'un quadrilatère en un autre quadrilatère. Chaque sommet de coordonnées (x, y) du quadrilatère à transformer se projetant sur un sommet de coordonnées (x', y') du deuxième quadrilatère fournit deux équations. Ces équations sont linéaires par rapport aux huit paramètres a, b, c, d, e, f, g et h. Pour calculer ces derniers, il suffit donc de résoudre le système de huit équations à huit inconnues associées aux quatre sommets du quadrilatère à transformer. Ces dernières s'écrivent sous la forme matricielle suivante :

$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ u_4 & v_4 & 1 & 0 & 0 & 0 & -u_4x_4 & -v_4x_4 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1x_1 & -v_1x_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2x_2 & -v_2x_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_4 & v_4 & 1 & -u_4x_4 & -v_4x_4 \end{bmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}. \tag{62}$$

Nous proposons maintenant d'utiliser la transformation projective définie par les équations (61) dans notre algorithme de compression par fractales. Notre but est de comparer un quadrilatère destination $\mathbf{r}$ à un quadrilatère source $\mathbf{d}$, tous les deux appartenant respectivement aux partitions R et D du support de l'image. Après avoir identifié la transformation projective v associée aux deux quadrilatères à l'aide de l'équation (62), chacun des pixels du quadrilatère $\mathbf{r}$ est comparé à son antécédent par la transformation spatiale v (voir la figure 73). Ceci revient de manière automatique à sous-échantillonner le quadrilatère source si celui-ci est plus grand que le quadrilatère destination, et à le sur-échantillonner dans le cas contraire.

La figure 74 illustre la déformation d'un bloc carré en deux quadrilatères quelconques par une transformation projective qui préserve les lignes droites mais ne préserve pas les distances relatives entre les points.

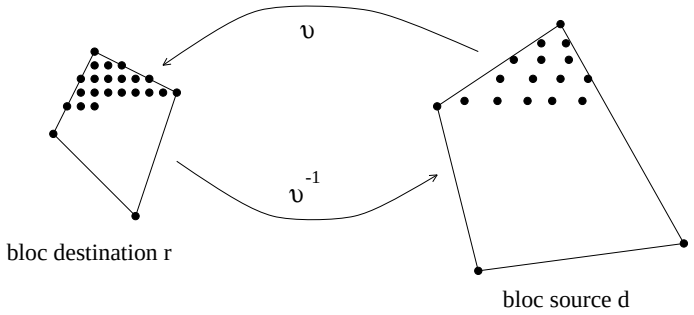


FIG. 73 - Comparaison de deux quadrilatères destination et source.

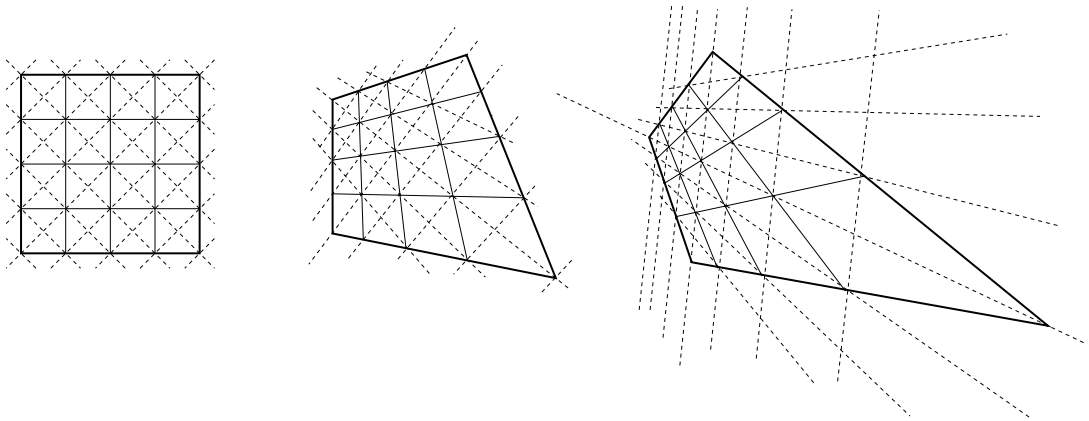


FIG. 74 - Transformation projective d'un bloc carré en un quadrilatère quelconque.

Le sous-échantillonnage est ainsi non régulier lorsque les deux quadrilatères comparés sont de formes très différentes (voir figure 75). Ceci s'explique par le fait que, d'après les explications données en début de paragraphe, le plan objet intervenant dans le calcul de la transformation projective n'est pas forcément parallèle au plan image. Selon l'inclinaison du plan objet par rapport au plan image, une accumulation de points peut apparaître sur un sommet ou un côté du quadrilatère source. Cet échantillonnage non régulier fausse par conséquent la comparaison des deux blocs. Pour pallier à ce defaut, nous étudions dans le paragraphe suivant une autre transformation spatiale souvent utilisée dans la littérature pour modéliser la déformation de blocs carrés en quadrilatères [163].

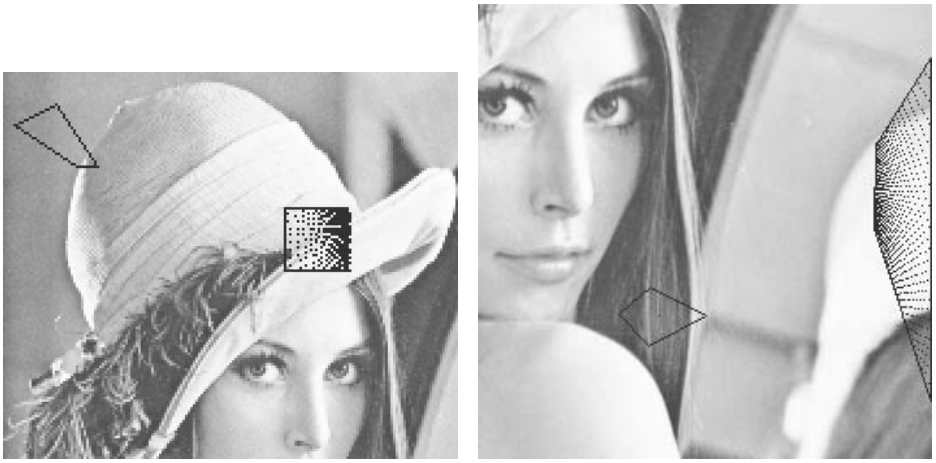


FIG. 75 - Deux exemples de transformations projectives ω_i d'un bloc source sur un bloc destination. Illustration du sous-échantillonnage du bloc source en transformant chaque pixel du bloc destination par ω_i^{-1}.

Transformation bilinéaire

Considérons deux quadrilatères destination et source, de forme quelconque, notés respectivement **r** et **d**. Un pixel de coordonnées (x', y') (resp. (x, y)), inclus dans le quadrilatère **r** (resp. **d**) est noté r (resp. d). Nous proposons dans cette section de calculer la transformation v illustrée sur la figure 76 permettant de déterminer le pixel r, image du pixel d.

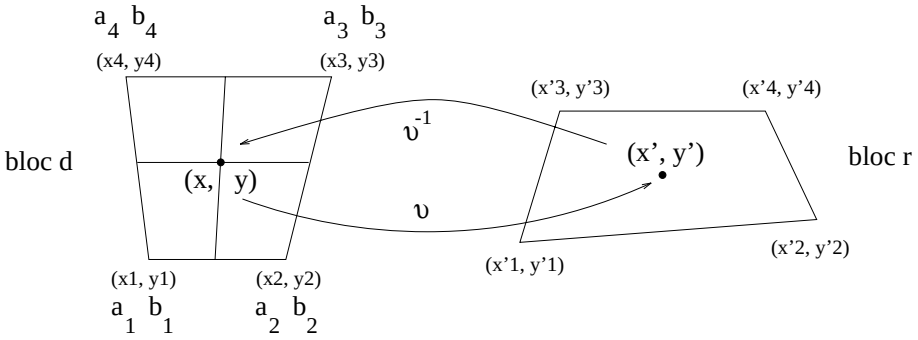


FIG. 76 - Comparaison par interpolations bilinéaires.

La transformation utilisée revient à appliquer un facteur d'échelle sur chacune des

coordonnées du pixel d. Elle s'écrit de la manière suivante[2] :

$$\begin{cases} x' = ax \\ y' = by \end{cases} \tag{63}$$

Dans la suite de cette partie, nous montrons comment évaluer le paramètre a de la transformation linéaire donnée par l'équation précédente, sachant que l'approche serait la même pour évaluer le paramètre b.

Le facteur a est vu comme étant une valeur prise par la fonction $a(x, y)$ passant par les quatre valeurs associées aux sommets du quadrilatère source **d**. Notre but est donc de déterminer la fonction $a(x, y)$ en chacun des pixels intérieurs au quadrilatère **d**. L'illustration d'une telle fonction est donnée sur la figure 77, et montre que $a(x, y)$ peut être approchée par la surface d'un paraboloïde hyperbolique.

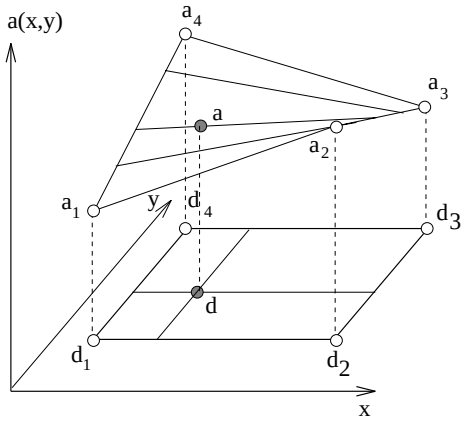


FIG. 77 - Interpolation bilinéaire d'un point.

Les valeurs que prend la fonction $a(x, y)$ en chacun des quatre sommets du quadrilatère sont notées a_1 à a_4 et b_1 à b_4. Elles sont déterminées à partir des huit équations suivantes :

$$\begin{cases} x'_k = a_k x_k \\ y'_k = b_k y_k \end{cases} \qquad \forall k = 1 \ldots 4 \tag{64}$$

Une approche systématique du problème consiste à exprimer $a(x, y)$ comme une combinaison linéaire de quatre fonctions linéairement indépendantes qui sont xy, x, y et 1 :

$$a(x, y) = \varphi_1 + \varphi_2 x + \varphi_3 y + \varphi_4 xy \tag{65}$$

2. En pratique nous opérons un changement de repère de manière à ne pas nous préoccuper des effets de bord en $x = 0$ et/ou $y = 0$.

Les coefficients φ_1 à φ_4 sont déterminés en appliquant (65) sur les quatre valeurs connues a_1 à a_4. Ils sont ensuite utilisés pour transformer l'ensemble des points intérieurs au quadrilatère. Ceci s'écrit sous la forme matricielle suivante [125] :

$$a(x,y) \;=\; [1 \;\; x \;\; y \;\; xy]\;\phi^{-1}\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}, \qquad \text{où} \qquad \phi = \begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \\ 1 & x_4 & y_4 & x_4y_4 \end{bmatrix}.$$

La difficulté de cette méthode vient du fait que l'inversion de la matrice ϕ peut être difficile lorsqu'elle est calculée sur un quadrilatère quelconque. L'inversion est facile dans le cas classique où ϕ est calculée sur un carré de référence.

Une autre formulation du problème consiste à exprimer a en fonction de deux coefficients réels α et β compris entre 0 et 1 représentés sur la figure 78. a est donné par l'équation suivante :

$$a = (1-\alpha)(1-\beta)\,a_1 + \alpha(1-\beta)\,a_2 + \alpha\beta\,a_3 + (1-\alpha)\beta\,a_4 \qquad (66)$$

L'équation (66) réalise une interpolation bilinéaire de la valeur a à partir des quatre valeurs a_1, a_2, a_3, et a_4. Le calcul exact des paramètres α et β est instantanné lorsque le bloc est de forme carrée. La solution est cependant moins simple à obtenir lorsque le quadrilatère est quelconque.

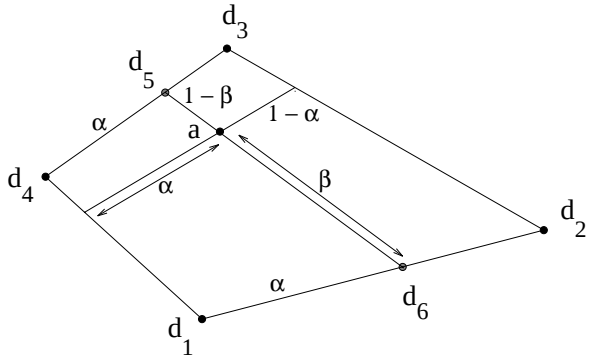


FIG. 78 - Interpolation bilinéaire du coefficient a.

a : Évaluation des coefficients α et β.

Pour calculer la valeur exacte du coefficient α il suffit de considérer le fait que les trois points d_5, d et d_6 sont alignés sur la base du quadrilatère (ces points indiqués sur la figure 77 sont associés aux valeurs a_5, a et a_6). Le couple $(\alpha, 1-\alpha)$ définit le système de coordonnées barycentriques de d_5 (resp. d_6) par rapport aux points d_3 et d_4 (resp. d_1 et d_2). Par conséquent, les deux valeurs d_5 et d_6 s'expriment à l'aide

des équations suivantes :

$$\begin{aligned} d_5 = d_4(1-\alpha) + d_3\alpha &= \alpha(d_3 - d_4) + d_4 \\ d_6 = d_1(1-\alpha) + d_2\alpha &= \alpha(d_2 - d_1) + d_1 \end{aligned} \tag{67}$$

La condition pour que les trois points soient alignés est qu'il existe u, v et w tels que :

$$\begin{aligned} ux + vy + w &= 0 \\ ux_5 + vy_5 + w &= 0 \\ ux_6 + vy_6 + w &= 0 \end{aligned} \tag{68}$$

et donc que le déterminant suivant soit nul :

$$\begin{vmatrix} x & y & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \end{vmatrix} = 0 \tag{69}$$

L'annulation du déterminant implique :

$$x(y_5 - y_6) - y(x_5 - x_6) + x_5y_6 - x_6y_5 = 0 \tag{70}$$

On aboutit ainsi, en utilisant les équations (67) et (70), à une équation du second degré en α du type $A\alpha^2 + B\alpha + C = 0$ à partir de laquelle seule la racine α comprise entre 0 et 1 est conservée.
Le coefficient β est ensuite calculé de façon analogue.

b : Approximation des coefficients α et β : solution retenue.

Dans le but de simplifier le calcul des coefficients de la transformation bilinéaire, nous proposons ici d'approximer les paramètres α et β. La méthode n'est pas exacte puisqu'elle considère à tort que les deux droites à partir desquelles sont calculés les coefficients α et β passent par les points d'intersection des droites portées par les côtés opposés du quadrilatère (figure 79). L'erreur ainsi introduite dans le calcul des coefficients peut cependant être négligée en pratique.

L'algorithme est le suivant :
Soit la valeur a à déterminer, associée au pixel d de coordonnées (x, y). Soit d_9 le point d'intersection des droites d_1d_4 et d_2d_3. La droite d_9d coupe la droite d_1d_2 au point d_6, et la droite d_4d_3 au point d_5. Soit aussi d_{10} le point d'intersection des droites d_4d_3 et d_1d_2. La droite $d_{10}d$ coupe la droite d_4d_1 au point d_8, et la droite d_3d_2 au point d_7. Les paramètres α et β sont obtenus après détermination des coordonnées des points d_5, d_6, d_7 et d_8 à l'aide des deux équations suivantes :

$$\alpha = \frac{x - x_8}{x_7 - x_8} \qquad \text{et} \qquad \beta = \frac{y - y_6}{y_5 - y_6}. \tag{71}$$

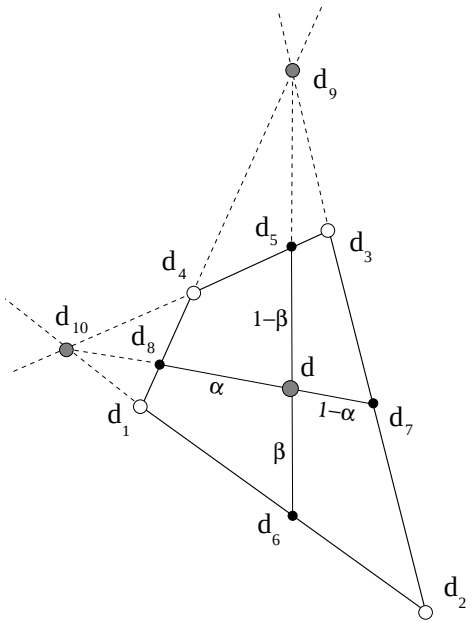


FIG. 79 - Calcul des paramètres α et β.

La figure 80, à comparer à la figure 74, montre la déformation d'un bloc carré en deux quadrilatères quelconques par une transformation bilinéaire. Nous remarquons que la transformation préserve les droites horizontales et verticales du carré de départ. Ceci s'explique par le fait que l'interpolation est linéaire dans ces deux directions. Les droites diagonales ne sont quant à elles pas préservées.

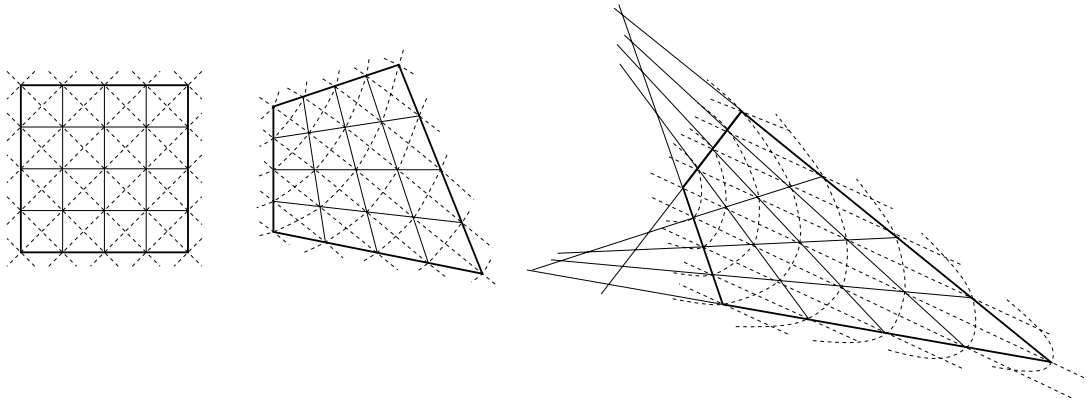


FIG. 80 - Transformation bilinéaire d'un bloc carré en un quadrilatère quelconque : les droites diagonales ne sont pas conservées.

La figure 81 illustre le sous-échantillonnage du même quadrilatère que celui de la figure 75, à l'aide de la transformation bilinéaire. Le sous-échantillonnage du support du quadrilatère est toujours irrégulier mais les points sont dans ce cas mieux répartis que sur les exemples de la figure 75.

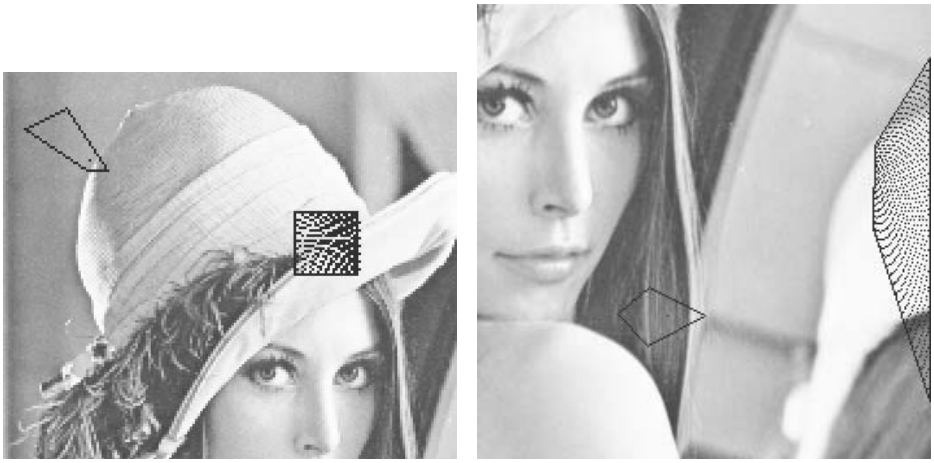


FIG. 81 - Deux exemples de transformations bilinéaires ω_i d'un bloc source sur un bloc destination. Illustration du sous-échantillonnage du bloc source en transformant chaque pixel du bloc destination par ω_i^{-1}.

6.4.3 Résultats

Nous présentons dans ce paragraphe différents résultats de décodage de l'image Lena (figure 82). Les partitions sont composées de triangles et de quadrilatères.

Afin de montrer l'influence de la transformation spatiale sur la qualité du décodage, nous utilisons les deux transformations étudiées dans les paragraphes précédents. Les coefficients de la transformation projective sont calculés à l'aide de la méthode *b.* décrite précédemment.

	projectif	bilinéaire
$T_c = 17.2$:1	30.83 dB	30.98 dB
$T_c = 37.9$:1	28.50 dB	28.75 dB
$T_c = 69.7$:1	26.70 dB	26.72 dB

TAB. 3 - Résultats de décodage de l'image Lena en utilisant des transformations spatiales projectives et bilinéaires: rapport signal à bruit (PSNR) pour différents taux de compression (T_c).

La table 3 confirme que la transformation spatiale utilisée pour comparer le contenu de deux blocs au cours du codage influe sur la qualité de l'image reconstruite. La transformation bilinéaire améliore les résultats puisque nous avons vu que dans ce cas l'échantillonage des blocs source est plus uniforme. On peut noter cependant que le gain est très faible (inférieur à 0.25 dB).

FIG. 82 - Partition R composée de 2988 triangles et de 2176 quadrilataires, partition D composée de 1052 triangles et de 407 quadrilaterals. Itérations 1, 2 puis 3. L'image reconstruite est en bas à droite (taux de compression = 15:1, PSNR = 32.2 dB).

La figure 83 montre qu'une partition composée de triangles et de quadrilatères améliore les résultats lorsque le taux de compression est supérieur à 30:1 et les dégrade lorsqu'il est inférieur à 15:1. Quand le but est de compresser une image sans la dégrader visuellement, les triangles sur les textures fines sont très petits (de taille inférieure à 30 pixels) et il est important dans ce cas de ne pas les regrouper deux à deux. Dans l'autre sens, l'amélioration est constatée visuellement mais reste limitée,

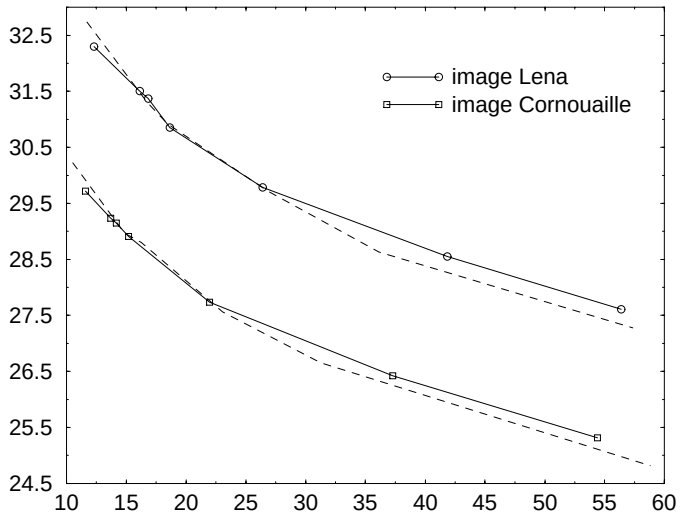


FIG. 83 - Rapport signal à bruit fonction du taux de compression. Décodage des images Lena et Cornouaille 512×512 partitionnées en triangles et quadrilatères (courbes continues). Les courbes en pointillé sont obtenues à partir de partitions triangulaires uniquement.

en terme de rapport signal à bruit, à 0.5 dB.

6.5 Accélération du codage : quantification des blocs

Nous avons vu que la durée importante de la phase de codage par fractales est essentiellement due au grand nombre de comparaisons entre les blocs destination de la partition R et les blocs source de la partition D. Notre but est ici d'accélérer le codage en réduisant le nombre de comparaisons. Pour cela, nous proposons de quantifier les triangles source à l'aide d'un quantificateur vectoriel de façon à ne conserver dans la partition D qu'un nombre réduits de triangles "représentatifs" de l'image[3] [41]. L'étude montre en outre qu'il n'est pas nécessaire d'utiliser toutes les parties de l'image pour calculer l'opérateur W définissant la transformation fractale et que quelques régions représentatives peuvent suffire (les collages se font bien sûr toujours sur le partitionnement complet R du support de l'image).

6.5.1 Quantification des triangles source

Dans le but d'accélérer le codage sur un partitionnement carré régulier, Lepsøy [100] organise l'ensemble des blocs source dans une structure arborescente. Notre approche est différente puisqu'elle diminue le nombre de triangles source considérés.

3. Cette étude a été menée en collaboration avec M. Antonini et M. Barlaud du laboratoire I3S de l'Université de Nice-Sophia Antipolis

Il est à noter que le fait d'utiliser une triangulation D composée de blocs source qui ne se recouvrent pas est une première simplification par rapport à l'approche optimale qui consiste à rechercher les blocs source parmi tous les blocs possibles de l'image.

Quantification des histogrammes

La quantification des blocs triangulaires en vue du codage par fractales nous a conduit à définir une méthode permettant de comparer le contenu de triangles de formes et de tailles différentes. L'algorithme doit séparer les triangles incluant un contour des triangles texturés ainsi que des triangles homogènes. La figure 84 montre un exemple de deux triangles source qui doivent être regroupés au sein d'une même classe puisqu'ils peuvent être associés au même triangle destination par le biais d'une transformation affine.

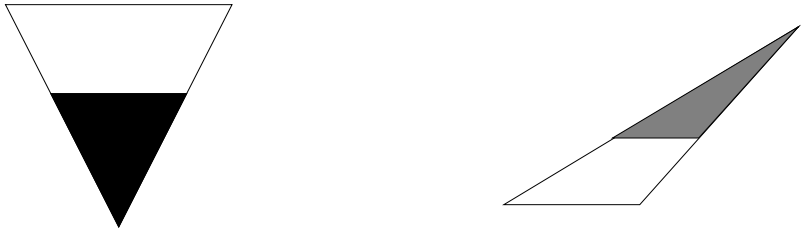


FIG. 84 - Exemple de deux triangles considérés identiques.

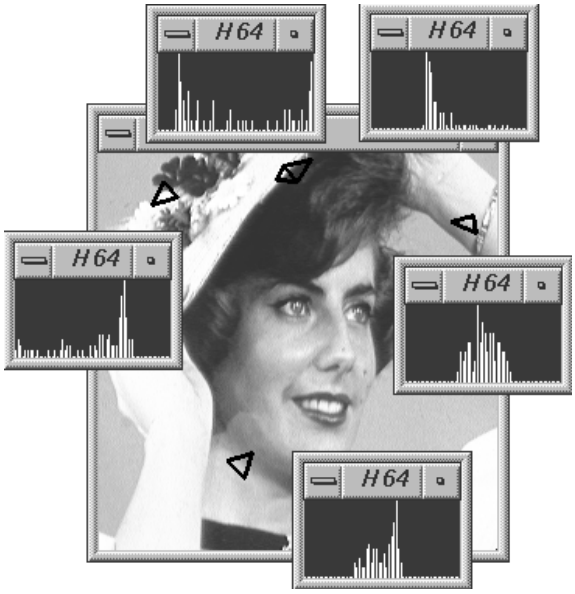


FIG. 85 - Exemples d'histogrammes à quantifier.

La solution retenue est de calculer l'histogramme centré et normalisé des niveaux de gris de chaque triangle source de façon à quantifier des vecteurs de longueur

constante et indépendants de la forme spatiale des blocs. La figure 85 montre cinq histogrammes caractérisant le contenu de triangles pris au hasard dans l'image.

La réduction du nombre de triangles source dans la partition D est faite par quantification vectorielle de leurs histogrammes, en utilisant une version modifiée de l'algorithme de Linde, Buzo et Gray [106], détaillé dans la section 2.7.4, page 25. Le critère de distorsion utilisé lors de la classification des vecteurs histogramme est celui de l'erreur quadratique, donné par l'équation suivante :

$$d(\mathbf{d}_i, \mathbf{d}_j) = \sum_{k=1}^{l_h} \left(\mathbf{hd}_i(k) - \mathbf{hd}_j(k)\right)^2 \qquad (72)$$

dans laquelle l_h est la longueur des vecteurs et $\mathbf{hd}_i(k)$, $\mathbf{hd}_j(k)$ sont les amplitudes d'indice k des histogrammes des triangles $\mathbf{d}_i$ et $\mathbf{d}_j$.

L'algorithme LBG est modifié de manière à ce qu'il génère un dictionnaire constitué de vecteurs appartenant à la séquence d'apprentissage (partition D). Pour cela nous modifions le point 5 de l'algorithme (détaillé dans la section 2.7.4) en ne retenant pas le vecteur "centroïde" de chacune des partitions du dictionnaire en cours d'élaboration, mais plutôt le vecteur histogramme **hd** le plus proche du centroïde, au sens de l'erreur (72).

La figure 86 illustre deux exemples de quantification obtenus à partir de la version modifiée de l'algorithme LBG[4]. Les triangles retenus sont en moyenne répartis sur des régions représentatives de l'image. Nous montrerons dans la section suivante que le faible nombre de triangles source ainsi quantifiés permet de calculer une transformation fractale pour laquelle l'attracteur reste proche de l'image originale.

FIG. 86 - Quantification d'une triangulation de l'image Femme. De gauche à droite : 576 triangles initiaux puis 32 et 64 triangles quantifiés.

4. La partition initiale est sur cette figure adaptée à l'image et ne correspond pas à la partition régulière D utilisée par notre algorithme de codage par fractales. La figure ne fait qu'illustrer l'algorithme de quantification.

6.5.2 Codage

L'algorithme de codage est le même que celui décrit dans le chapitre 5 mis à part le fait qu'il ne considère qu'un nombre restreint de triangles source au sein de la partition régulière D. L'avantage de la méthode est double puisqu'elle permet :

1. de diminuer très sensiblement la durée du codage (qui est dans ce cas sous-optimal);

2. d'augmenter (sous condition) le taux de compression.

Nous donnons ci-dessous les expressions permettant de calculer le taux de compression lorsque l'ensemble des triangles de la partition D sont considérés (eq. 73) et lorsque les triangles source sont quantifiés (eq. 74) :

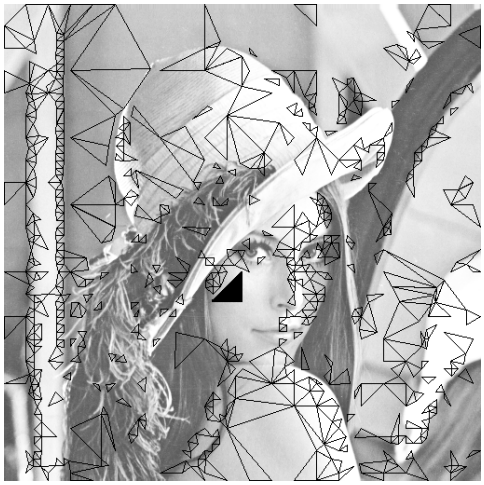
$$T_c = \frac{2^{2n} * 8}{N(n_i + n_c + n_s + n_o) + X} \tag{73}$$

$$T'_c = \frac{2^{2n} * 8}{N(n_c + n_s + n_o) + \overline{Q}(3 * 2 * n) + X} \tag{74}$$

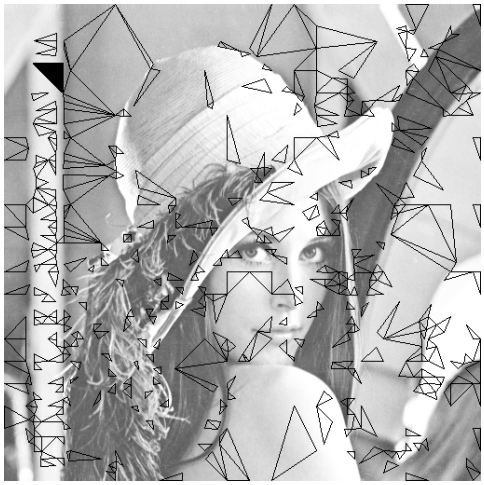
Dans les expressions précédentes, n est le nombre de bits nécessaires au codage de la largeur de l'image supposée carrée, N est le nombre de triangles destination, $\overline{Q}$ désigne le nombre de blocs source retenus après quantification de la partition D et X le nombre de bits codant la partition R. Les notations n_i, n_c, n_s et n_o sont les mêmes que celles utilisées dans la section 5.3.1. L'étape de quantification permet donc d'augmenter le taux de compression si elle vérifie l'expression $\overline{Q}(3 * 2 * n) < N(n_i)$.

La figure 87 est obtenue au terme de la phase de codage précédée d'une étape de quantification des triangles source réguliers. La partition R est adaptée à l'image. Les cinq sous-images (b), (c), (d), (e) et (f) montrent les triangles destinations associés à cinq triangles source pris au hasard parmi les triangles de la sous-image (a). Ces derniers sont issus de l'étape préliminaire de quantification vectorielle. La figure montre que les triangles source constants (e) ne sont pas utilisés par le codeur. Les blocs texturés sans contours nets (d) sont aussi peu utilisés puisque leur contenu très irrégulier rend difficile leur appariement avec un autre bloc. Les deux triangles source des images (b) et (c) sont quant à eux très sollicités. Ceci s'explique par le fait que leur surface n'est pas constante et peut être sensiblement modifiée par le facteur d'échelle β_2, de façon à optimiser les collages (l'information surfacique à l'intérieur des deux blocs peut plus facilement être appréciée en regardant une image de gradient telle que celle donnée par la figure 91). On remarque sur l'image (f) mais aussi dans beaucoup d'autres cas expérimentaux qu'un triangle qui inclut un contour contrasté est souvent associé aux triangles destination de même nature ainsi qu'aux triangles texturés.

(a) Extraction de triangles source $\mathbf{d}_k$



(b) Association $\mathbf{d}_1 \Leftrightarrow \mathbf{r}_i$



(c) Association $\mathbf{d}_2 \Leftrightarrow \mathbf{r}_i$

(d) Association $\mathbf{d}_3 \Leftrightarrow \mathbf{r}_i$

(e) Association $\mathbf{d}_4 \Leftrightarrow \mathbf{r}_i$

(f) Association $\mathbf{d}_5 \Leftrightarrow \mathbf{r}_i$

FIG. 87 - Associations après codage : blocs destination en traits fins, blocs source en noir.

6.5.3 Résultats expérimentaux

Nous proposons dans ce paragraphe de tester l'algorithme de codage sur les images Femme et Poivron de taille 256×256 pixels. Les figure 88 et 89 montrent différents résultats de décodage à partir d'un nombre variable de triangles source. L'évolution de la qualité visuelle des images reconstruites, mesurée à l'aide du rapport signal à bruit est illustrée sur la figure 90. L'intérêt évident de la quantification des triangles source est qu'elle permet de diminuer les temps de calculs. Lorsque les partitions R et D contiennent respectivement N et Q blocs, le nombre de comparaisons est égal à NQ. En ne conservant que $\overline{Q}$ triangles dans la partition D, le temps de calcul diminue d'un facteur $Q/\overline{Q}$. Les résultats montrent que le rapport signal à bruit chute de 0.5 dB lorsque le nombre de triangles source, et par conséquent le temps de codage, est réduit d'un facteur deux. La diminution rapide du rapport signal à bruit au delà de cette valeur s'explique par le fait que l'erreur entre l'image originale et l'image transformée est de plus en plus importante.

FIG. 88 - De bas en haut et de droite à gauche : Image Femme 256×256, reconstruction de 5 images à partir des 576 triangles source de D, puis 256, 128, 64, et 32 triangles. Les taux de compression sont respectivement égaux à : 18.3, 18.3, 22.8, 26.3 et 28.5:1.

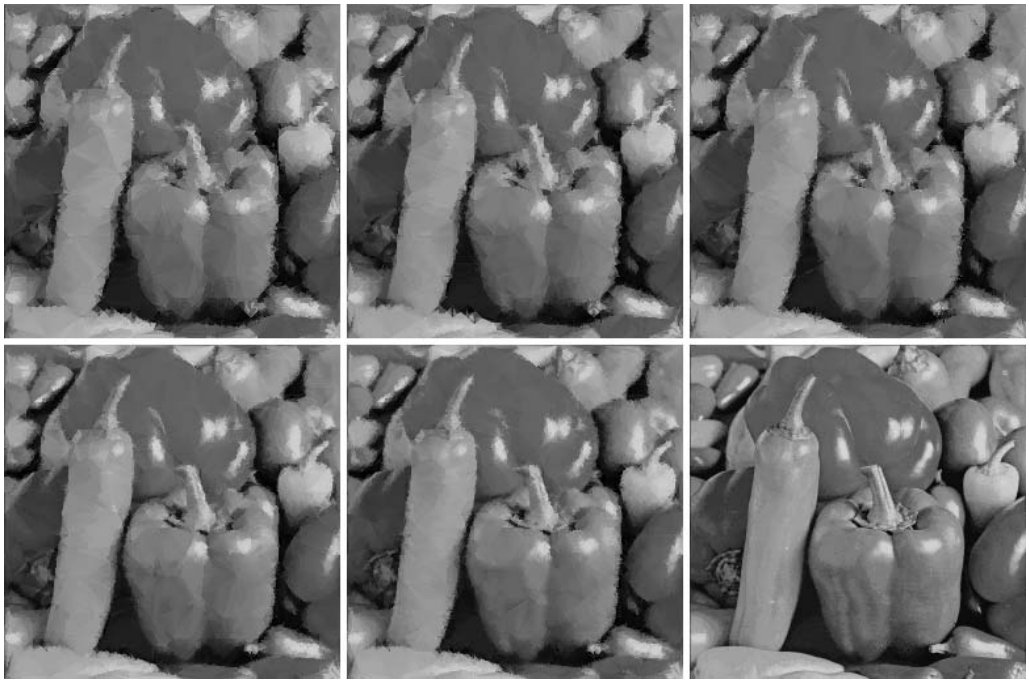


FIG. 89 - De bas en haut et de droite à gauche : Image Poivron 256×256, reconstruction de 5 images à partir des 576 triangles source de D, puis 256, 128, 64, et 32 triangles. Les taux de compression sont respectivement égaux à : 15.6, 16.3, 20.15, 22.8 et 24.5:1.

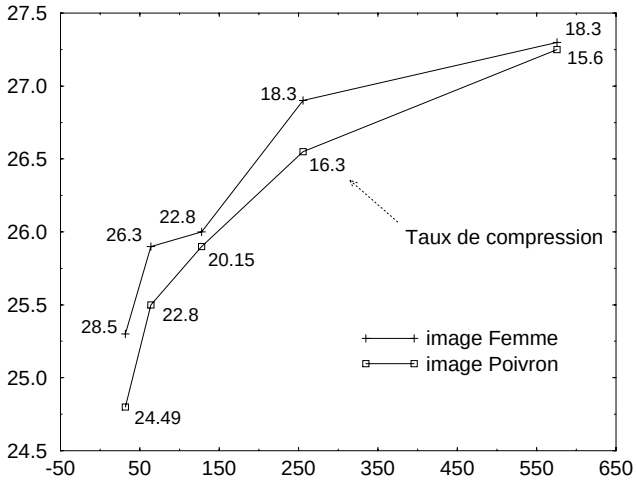


FIG. 90 - Rapport signal à bruit fonction du nombre de triangles source considérés dans la partition D.

Classification des triangles

Nous nous proposons dans cette section d'étudier l'influence des blocs source sur la qualité des images décodées par fractales. Une méthode simple nous permet de regrouper les triangles de la partition D dans trois classes distinctes. Une première étape consiste à calculer l'image de gradient de l'image originale de manière à, dans une seconde étape, séparer les triangles source incluant un contour contrasté des autres (voir figure 91). L'algorithme retourne ainsi une première classe de triangles. Les blocs n'incluant pas de contours sont regroupés dans deux autres classes en appliquant un seuil sur la variance du gradient.

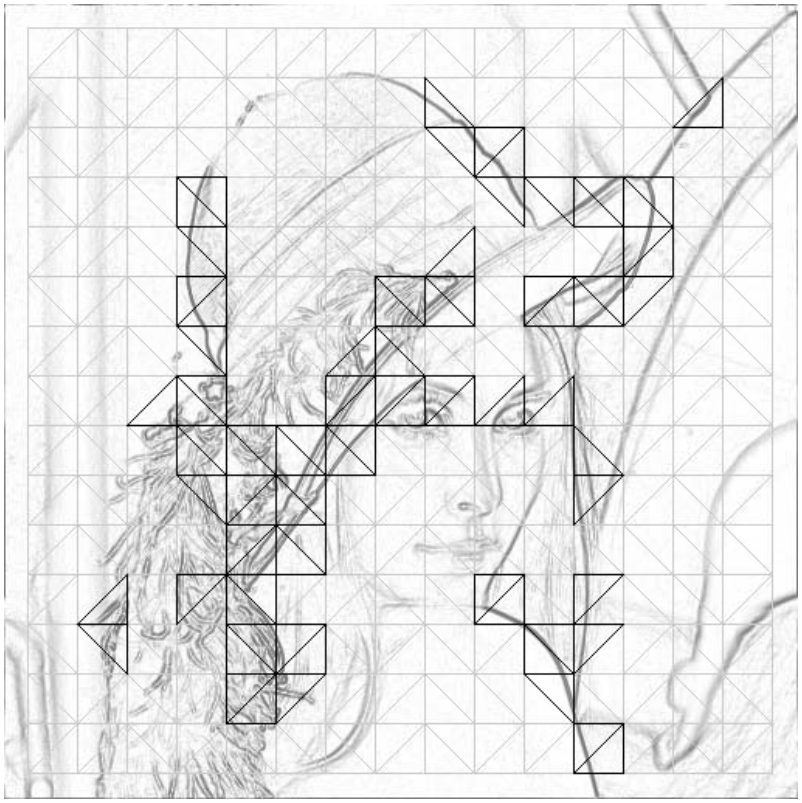
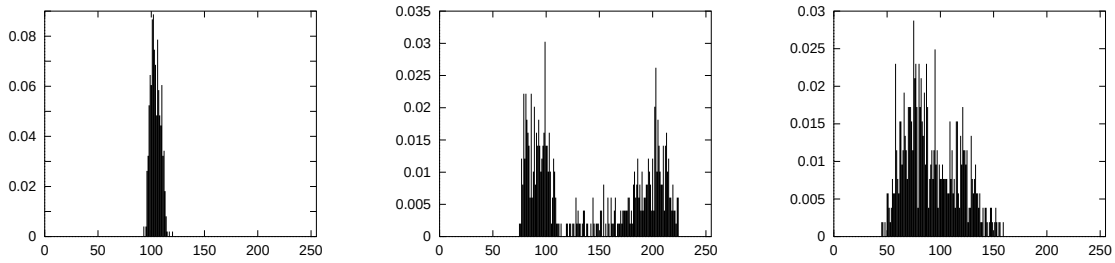


FIG. 91 - Extraction des triangles incluant un contour à partir de l'image de gradient.

La quantification des triangles au sein de chacune des trois classes est réalisée de la même façon que prédédemment, par quantification vectorielle "modifiée" [5]. La figure 93 montre trois exemples de vecteurs histogramme à quantifier.

5. Pour des raisons pratiques, l'algorithme de quantification vectorielle est celui de Kohonen [93]. Les résultats peuvent être différents de ceux obtenus avec l'algorithme LBG. Ce manque de précision est sans importance ici puisque notre but est de montrer "l'influence moyenne" du contenu des blocs source sur la qualité du codage.

FIG. 92 - 3 triangles appartenant à des classes différentes.



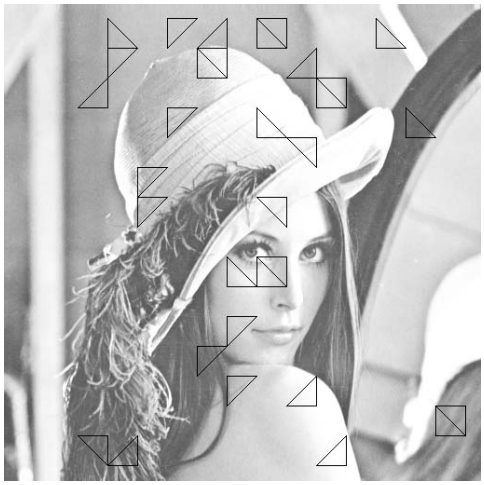
(a) triangle homogène (b) triangle avec contour (c) triangle texturé

FIG. 93 - Histogrammes des 3 triangles de la figure précédente.

Influence sur la qualité du codage

Différents résultats de décodage sont présentés sur la figure 94. Ils sont obtenus à partir de 32 triangles source étant chacun extraits de la même classe. L'image 94(c) montre les triangles quantifiés au sein de la classe "contours". Un certain nombre de triangles homogènes et texturés sont retenus à cause du fait que la quantification vectorielle conserve les vecteurs les plus différents les uns des autres. Les images 94(a) et (e) montrent 32 triangles issus des classes "homogènes" et "textures". Les images (b), (d) et (f) obtenues à partir de chacun des trois dictionnaires confirment l'importance des blocs source non homogènes (de forte variance) pour le codage par fractales. Ceci est très visible sur l'épaule de Lena reconstruite à partir de triangles homogènes. La valeur maximale, limitée à 1.4, du coefficient d'échelle des transformations massiques ne permet pas dans ce cas d'approximer correctement les contours contrastés. Nous avons vu dans la section 5.2.7 que son écrêtage est cependant utile pour assurer la contraction de la transformation fractale.
Le résultat expérimental de la figure 94 nous incite à conclure qu'il est inutile de

considérer des blocs source homogènes lors du codage, et qu'il est important que ces blocs incluent des contours et des textures de forte variance. Ceci est en outre confirmé par une étude géometrique de la transformation massique d'un bloc source dans l'espace vectoriel engendré par le vecteur constant $\mathbf{b}_1$ et le vecteur décimé $\mathbf{b}_2$ (eq. 29) [152].



(a) 32 vecteurs homogènes

(b) Reconstruction : PSNR = 27.42 dB

(c) 32 vecteurs avec contours

(d) Reconstruction : PSNR = 28.68 dB

(e) 32 vecteurs texturés

(f) Reconstruction : PSNR = 29.30 dB

FIG. 94 - Taux de compression = 21.8:1

6.6 Codage par TCD et fractales

Nous avons vu dans la section 2.8 que la norme JPEG de compression des images fixes est fondée sur la transformation en cosinus discrète (TCD), calculée sur chacun des blocs carrés de taille $N \times N$ d'un partitionnement régulier. La transformation fournit une représentation dans le domaine fréquentiel de chacun des blocs de l'image. Nous rappelons que l'avantage de cette transformation est d'être réelle et de retourner un ensemble de $N \times N$ coefficients concentrés autour de l'origine (fréquence (0,0)). Pour cette raison, la transformation est particulièrement intéressante pour la compression des images. La compression consiste à quantifier la suite des coefficients réordonnés en zig-zag. L'inconvénient provient du fait que lorsque le taux de compression demandé est trop important, tous les coefficients pondérant les hautes fréquences sont annulés, et seul reste un nombre très réduit de coefficients pondérant les basses fréquences : l'image reconstruite présente des effets de blocs très visibles. Par exemple, un bloc original visuellement homogène est approximé par un bloc constant, égal à sa valeur moyenne.

(a) $T_c = 36.5 : 1$ (b) $T_c = 50.2 : 1$

FIG. 95 - Deux exemples de compression selon la norme JPEG.

La méthode proposée par Barthel et al. dans [16] améliore sensiblement les résultats de la norme JPEG, à taux de compression élevé. Elle permet en outre de reconstruire des images de meilleure qualité que celles obtenues avec les schémas classiques de compression par fractales dans l'espace des luminances, à taux de compression égaux.
L'approche, basée sur un partitionnement carré fournissant des blocs destination, est hybride. Elle combine la transformation en cosinus discrète (codage de la redondance intra-bloc) et la transformation fractale (codage des similarités inter-blocs).

La méthode consiste, pour un bloc destination donné, à approximer la majeure partie de son spectre par une transformation fractale opérant sur les coefficients fréquentiels d'un autre bloc de l'image, appelé bloc source[6]. Les coefficients fréquentiels du bloc destination ne pouvant pas être correctement approximés sont quantifiés séparément puis codés. Nous présentons dans la suite de cette section l'algorithme de base, décrit dans [16].

Considérons la représentation fréquentielle d'un bloc destination $\mathbf{r}$ (resp. source $\mathbf{b}_2$), de taille $N \times N$, notée $\mathbf{F}$ (resp. $\mathbf{G}$). Les coefficients spectraux correspondants, réordonnés en zig-zag, sont notés F_i et G_i. L'approximation $\hat{\mathbf{F}}$ du spectre $\mathbf{F}$ est donnée par l'expression suivante :

$$\hat{F} = \begin{bmatrix} \hat{F}_0 \\ \hat{F}_1 \\ \hat{F}_2 \\ \hat{F}_3 \\ \dots \\ \hat{F}_4 \\ \dots \\ \hat{F}_{N^2-1} \end{bmatrix} = \begin{bmatrix} \theta_0 G_0 \\ 0 \\ \theta_2 G_2 \\ 0 \\ \dots \\ \theta_2 G_l \\ \dots \\ \theta_2 G_{N^2-1} \end{bmatrix} + \begin{bmatrix} \theta_1 \\ F_1 \\ 0 \\ F_3 \\ \dots \\ 0 \\ \dots \\ F_{N^2-1} \end{bmatrix} \qquad \theta_0 = 0.5.$$

Les coefficients fréquentiels F_i $(i \neq 0)$ sont quantifiés puis codés individuellement. L'ensemble de ces coefficients est noté S_{TC}. Le coefficient θ_1 optimal est donné par $\theta_1 = F_0 - \theta_0 G_0$. Le coefficient θ_2 optimal sert à l'approximation fractale de la composante dynamique du spectre du bloc destination. Il est donné par :

$$\theta_2 = \frac{\sum\limits_{i \notin S_{TC}} G_i F_i}{\sum\limits_{i \notin S_{TC}} G_i^2},$$

et doit être de module inférieur à 1 pour assurer la contraction de la transformation fractale. Les trois coefficients θ_i sont à rapprocher des trois coefficients γ_i de la formule(54), mais opèrent ici sur la représentation fréquentielle du bloc source décimé $\mathbf{b}_2$.
L'intérêt de la méthode est de ne coder qu'un nombre très réduit de coefficients spectraux F_i, sensiblement inférieur au nombre de coefficients codés dans le schéma classique de la norme JPEG. Ceci est possible puisque le reste du spectre est approximé par la transformation fractale, codée par les deux coefficients θ_2 et θ_1. La figure 96 illustre un résultat de décompression obtenu à partir de cette méthode, et montre que les effets de blocs sont peu visibles.

6. On note dès à présent que les blocs source et destination sont de même taille. La contraction spatiale n'est pas nécessaire.

FIG. 96 - Codage hybride par TCD et fractales. Taux de compression = 57:1, PSNR = 27.91 dB.

6.7 Multirésolution et fractales

Le codage d'une image par fractales tel que nous l'avons vu jusqu'à présent est fondé sur des transformations locales qui opèrent sur des blocs de l'image. Les effets de blocs sont dans ce cas difficiles à masquer lorsque le taux de compression est élevé. Récemment, différents checheurs ont montré que le schéma de codage par fractales peut être utilisé pour définir des relations entre les différents niveaux d'une décomposition multi-résolution de l'image (décomposition en sous-bandes ou par ondelettes). Dans [139], l'image originale est décomposée en n sous-bandes notées i, $(i = 0 \ldots n-1)$. L'image basse résolution (niveau $n-1$) ainsi que les trois sous-bandes du même niveau sont codées de manière exacte. Chacune des trois sous-bandes de niveau supérieur (niveau $n-2$) sont ensuite codées par une approche fractale[7] à partir des trois sous-bandes dans le niveau $n-1$. Le processus continue ainsi jusqu'à atteindre la résolution 0 de l'image originale.
D'autres études cherchent à lier plus directement la transformation fractale à la transformation en ondelette de Haar [96] [40]. Dans tous les cas, les blocs source et destination sont carrés. Simon [153] remarque qu'il est possible de calculer les coefficients de la transformation fractale directement à partir des coefficients de la

7. La recherche des similarités au sein des sous-bandes de fréquences spatiales se fait entre blocs source et destination de même taille.

représentation multirésolution de l'image. Il démontre que le coefficient de décalage est lié à l'image de basse résolution (niveau $n-1$) ainsi qu'aux trois images de détail du même niveau $n-1$. Le coefficient d'échelle définit un filtre permettant d'interpoler les images de détail de niveau i inférieur à $n-1$, à partir des images de détails qui leurs sont associées dans le niveau de résolution $i+1$.
Les résultats théoriques sont démontrés en considérant le cas particulier où un bloc source englobe les blocs destination auxquels il est associé (ce schéma est celui utilisé par Dudbridge et Monro dans le domaine spatial, détaillé dans la section 3.3.5).

L'avantage de ces méthodes est qu'elles réduisent ou annulent les effets de blocs dans les images reconstruites en utilisant respectivement l'ondelette de Haar ou des ondelettes plus lisses. La figure 97, extraite de [40], confirme cette remarque (la transformation fractale est dans ce cas calculée sur une décomposition multirésolution de l'image utilisant l'ondelette de Harr et l'ondelette spline). L'interpolation des images de détails au travers des différentes résolutions permet en outre de générer des sous-bandes intermédiaires de façon à améliorer la qualité des zooms.

(a) $T_c = 65{:}1$, PSNR = 28.2 dB (b) $T_c = 63.2{:}1$, PSNR = 29.9 dB

FIG. 97 - Images décompressées à partir du schéma de Davis liant le codage par fractales à une approche multirésolution par ondelette de Haar (a) et par ondelette spline (b).

6.8 Résultats comparatifs

6.8.1 Codage fractal par blocs

Dans le chapitre 4, nous avons présenté les différents modèles de partitionnement utilisés dans la littérature pour initialiser la phase de compression par fractales, à savoir :

- le partitionnement en arbre quaternaire (quadtree);
- le partitionnement Horizontal - Vertical (HV);
- le partitionnement de Delaunay.

Jacquin a également proposé le partitionnement pseudo-adaptatif à deux niveaux de résolution, présenté dans la section 3.3.2. Nous ne considérons pas ce modèle dans cette section puisqu'il donne de moins bons résultats visuels que le quadtree, à taux de compression égaux. Ceci est principalement dû au fait que les larges zones homogènes de l'image restent couvertes par des blocs de petite taille, en raison de la non adaptivité du partitionnement.

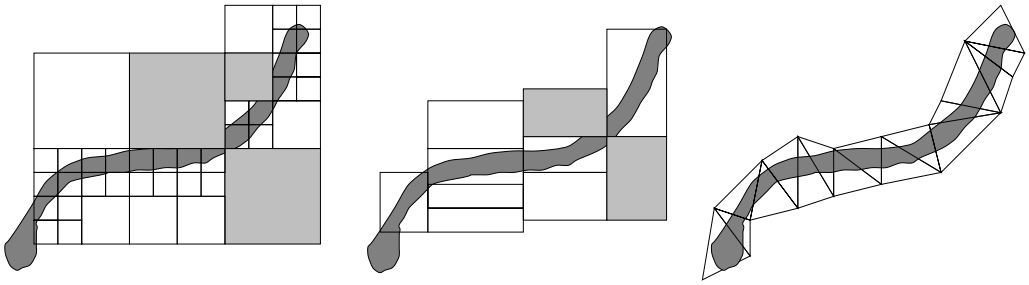


FIG. 98 - Positionnement des blocs en présence d'un contour. De gauche à droite : quadtree, partitionnement H-V et triangulation calculée par division-fusion.

La figure 98 illustre le recouvrement des contours de l'image par les différents partitionnements étudiés.

Dans le cas du quadtree, l'image reconstruite présente, à des taux de compression élevés (figures 99 et 100), des effets de blocs très visibles au niveau des contours obliques contrastés. Ceci peut s'expliquer par le fait que la forme de l'enveloppe des carrés incluant une partie d'un contour est "en escaliers". Les bords d'un contour épais peuvent être en partie recouverts par des carrés de grande taille et apparaissent découpés dans l'image reconstruite. Le processus récursif et rigide de division des carrés en quatre sous-carrés génère un grand nombre de blocs inutiles au voisinage des régions non-homogènes au détriment du taux de compression.

Les partitionnements rectangulaires (figure 101) et triangulaires (figures 102 et 103) améliorent quand à eux la qualité moyenne des images à taux de compression élevés. En effet, la figure 98 montre que l'enveloppe des triangles de Delaunay

obtenus par division-fusion est plus continue que les deux autres types d'enveloppe puisqu'elle n'est pas liée à la structure discrète de l'image. Elle permet ainsi de suivre des contours obliques et d'augmenter sensiblement la qualité visuelle des contours reconstruits par rapport à ceux obtenus sur le partitionnement quadtree. La qualité visuelle des images décodées est cependant moins bonne que celle des images décodées selon le partitionnement HV. Ceci s'explique par le fait que ce dernier permet une meilleure organisation des blocs sur les régions homogènes et texturées de l'image, fait que ne permet la triangulation de Delaunay.

Les courbes suivantes (figure 104) sont calculées à partir des méthodes de compression traitant des régions carrées (quadtree1 et quadtree2), rectangulaires (HV) et triangulaires (Delaunay). Les deux méthodes quadtree1 et quadtree2 ([61]) diffèrent de par le nombre de blocs source considérés au cours du codage. Le partitionnement associé à la courbe quadtree2 est composé de carrés destination de tailles 32×32, 16×16, 8×8 et 4×4. Selon leur taille, ils sont comparés au travers de 12769, 3721, 1024 et 256 blocs source. La courbe quadtree1 est obtenue à partir du même partitionnement mais, selon la taille des blocs destination, les blocs source sont recherchés parmi 50625, 58081, 62001 et 64009 carrés.

La courbe HV ([64]) est obtenue à partir du partitionnement rectangulaire. Un bloc destination de taille $n \times n$ est comparé à l'ensemble des blocs source de l'image de tailles $2n \times 2n$, $2n \times 3n$, $3n \times 2n$ et $3n \times 3n$. Le temps de codage d'une image 512×512 est dans ce cas compris entre 10 heures et 1000 secondes.

La courbe Delaunay est obtenue à partir d'une partition R triangulaire. En fonction du nombre de triangles dans cette partition, chaque triangle destination est comparé à 468 ou 908 triangles source (le nombre de triangles dans la partition D dépend de la partition R, de façon à ce que la différence de taille moyenne entre les blocs source et les blocs destination ne soit pas trop importante). Ceci nous permet de coder des images 512×512 avec des temps de calcul inférieurs à une heure. Pour cette raison, nous ne pouvons comparer directement (en termes de compression-distorsion) nos résultats obtenus à partir de la triangulation de Delaunay avec ceux de Fisher fondés sur le partitionnement HV.

La figure 104 confirme le fait qu'un partitionnent triangulaire améliore les résultats par rapport au partitionnement quadtree, lorsque le taux de compression est supérieur à 30:1. Ceci confirme l'intérêt d'utiliser un partitionnement souple pour compresser les images par fractales.

FIG. 99 - partitionnement quadtree. T_c = 61:3, PSNR = 25.9 dB.

FIG. 100 - partitionnement quadtree. T_c = 50:1, PSNR = 20.38 dB.

FIG. 101 - partitionnement HV. $T_c = 57{:}1$, PSNR $= 27.22$ dB.

FIG. 102 - partitionnement de Delaunay. $T_c = 61{:}1$, PSNR $= 26.45$ dB.

FIG. 103 - partitionnement de Delaunay. T_c = 50:2, PSNR = 21.12 dB.

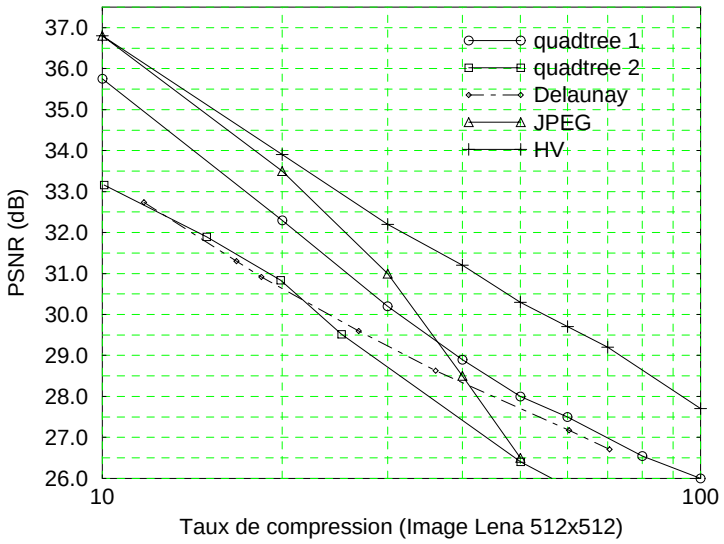


FIG. 104 - Méthodes par blocs : rapports signal à bruit fonctions du taux de compression.

6.8.2 Méthodes hybrides

Les courbes de la figure 105 sont calculées à partir des méthodes de compression fractale hybrides introduites dans ce chapitre. Les courbes DCT-fractales, ondelette Harr-fractales et ondelette spline-fractales illustrent respectivement les résultats en termes de débit-distorsion de Barthel et *al.* [16] et Davis [40]. Ces dernières confirment l'intérêt des méthodes hybrides pour la compression des images fixes.

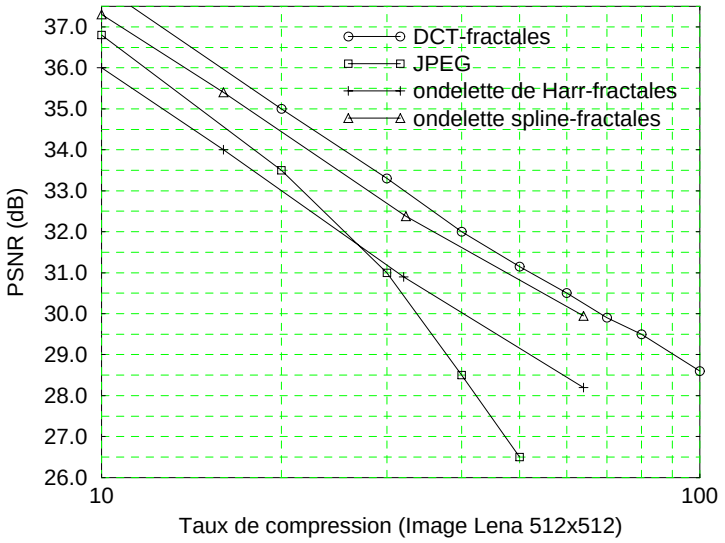


FIG. 105 - Méthodes hybrides : rapports signal à bruit fonctions du taux de compression.

Il faut cependant noter que le partitionnement triangulaire que nous avons proposé pour le codage par fractales des images fixes peut être étendu au cas du codage des séquences vidéo. Des nombreuses études portent actuellement sur le suivi et l'interpolation du mouvement par maillage triangulaire actif. Nous reviendrons ce point dans la conclusion de ce mémoire.

(a) Image Lena

(b) Image Vegas

(c) Image Cornouaille

(d) Image Poivron

(e) Image Goldhill

(f) Image Femme

FIG. 106 - Différentes images (512×512 et 256×256 - 8 bpp) utilisées pour les tests de compression.

Chapitre 7

Conclusion générale

Nous avons présenté dans ce mémoire une méthode de compression des images fixes par fractales, fondée sur un partitionnement triangulaire. La souplesse du partitionnement et l'approche adaptative de l'algorithme mis en œuvre pour son calcul nous ont permis d'améliorer la qualité visuelle des images reconstruites par rapport à celle obtenue à partir de partitionnements plus rigides tel que le quadtree.

Le début de ce mémoire traite des différentes méthodes de codage des données, et de compression des images fixes.

Nous avons ensuite introduit les notions liées à la théorie des IFS avant de montrer comment les utiliser pour compresser les images naturelles. Nous avons pour cela détaillé la méthode originale proposée en 1989 par Jacquin, qui consiste à coder les similarités entre blocs de l'image à l'aide de transformations affines contractantes. Nous avons aussi décrit une seconde approche proposée en 1992 par Dudbridge ne nécessitant pas la mise en correspondance des différentes régions de l'image.

Les méthodes classiques de compression par fractales consistent à approximer chacun des blocs d'une partition de l'image à partir d'autres blocs extraits de la même image, par des transformations contractantes. La compression est dans ce cas réalisée en codant les paramètres des transformations. Il est donc important que la partition contienne un minimum de blocs, et qu'elle soit le mieux possible adaptée au contenu de l'image. Nous avons pour cette raison présenté les deux modèles de partitionnement utilisés pour la compression fractale (quadtree, horizontal-vertical).

Notre contribution s'est portée essentiellement sur l'étude d'un modèle souple et adaptatif, à savoir la triangulation de Delaunay. Nous avons proposé deux méthodes permettant de calculer une triangulation de l'image. La première méthode de type division-fusion génère une partition formée de petits triangles au-dessus des zones texturées et des contours de l'image, et de plus grands triangles au-dessus des régions homogènes. Un algorithme efficace nous permet en outre de coder une telle partition. La seconde méthode positionne les triangles de petites tailles de manière précise le long des principaux contours détectés dans l'image. Un avantage de chacune de ces deux méthodes vient du fait que le positionnement des triangles n'est pas contraint par la structure discrète de l'image. Ceci supprime les effets de "marches d'escaliers" aux voisinages des contours, effets constatés sur les partitionnements quadtree et rectangulaires. Nous avons ensuite montré comment utiliser une telle partition triangulaire pour le codage par fractales, et avons détaillé un algorithme de compression. Nous avons en outre proposé deux optimisations visant à améliorer nos résultats en terme de compression-distorsion. La première a permis de diminuer le nombre de blocs en regroupant des triangles voisins au sein de quadrilatères, et par conséquent d'augmenter le taux de compression à qualité constante. La seconde a permis d'accélérer la phase de codage en réduisant le nombre de comparaisons inter-blocs par quantification des triangles. Nous avons par ailleurs présenté un ensemble de méthodes existantes, visant à optimiser les phases de codage et de décodage, ainsi que certaines méthodes hybrides liant les fractales aux représentations multirésolution (ondelettes) et à la transformée en cosinus discrète.

Les perspectives et les directions de recherches à suivre pour améliorer nos résultats

portent à la fois sur la quantification des coefficients de la transformation fractale et sur l'optimisation du partitionnement. Nous avons noté que les coefficients d'échelle et de décalage des transformations contractantes sont corrélés. Il serait dans ce cas intéressant d'étudier la quantification des vecteurs "échelle-décalage" (de dimension deux) au cours du calcul de la transformation fractale. La répartition géométrique particulière des vecteurs sur un plan devrait pouvoir être exploitée en utilisant un quantificateur vectoriel algébrique.

Le partitionnement triangulaire de Delaunay est le dual du partitionnement de Voronoï, et celui-ci est directement accessible à partir de notre structure de données. L'avantage d'un tel partitionnement est qu'il contient approximativement deux fois moins de blocs que la triangulation de Delaunay. L'inconvénient est que les polygones qui le composent n'ont pas un nombre constant de sommets et qu'il est dans ce cas impossible de les déformer à l'aide de transformations géométriques simples telles que celles étudiées dans ce mémoire. La solution serait, pour un polygone destination $\mathbf{r}$, de considérer un ensemble d'autres polygones $\mathbf{d}_i$, antécédents de X par une transformation affine contractante. De façon à limiter le nombre de comparaisons inter-blocs, les polygones $\mathbf{d}_i$ considérés pourraient par exemple être centrés sur les germes issus du partitionnement de Voronoï R adapté à l'image. Cette solution mériterait à mon avis d'être testée.

La méthode de codage présentée dans ce mémoire doit pouvoir en outre être directement étendue au codage des séquences d'images. De nombreux travaux portent aujourd'hui sur le suivi temporel du mouvement à partir de mailles actives triangulaires ou composées de quadrilatères. La compensation du mouvement se fait dans ce cas à l'aide de transformations affines [53] [104] [122] ou bilinéaires [163]. Fisher et *al.* [65] ont proposé un schéma de codage vidéo sur un partitionnement quadtree. Les carrés ne pouvant pas être compensés sont approximés à partir d'autres blocs de l'image de la même manière que dans le cas du codage par fractales des images fixes. La déformation continue de la triangulation permet quand à elle l'interpolation temporelle des images. L'adaptation de l'approche fractale sur ce type de partitionnement ne semble pas avoir encore été testée. Le partitionnement de Voronoï doit aussi pouvoir être utilisé pour le codage des séquences vidéo par fractales en utilisant la même approche que celle proposée en perspective de notre travail de thèse.

Bibliographie

[1] N. Ahuja, B. An, and B. Schachter. Image representation using Voronoi tessellation. *CVGIP*, 29:286–295, 1985.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, 1992.

[3] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proc. 5th annual ACM-SIAM symposium on discrete algorithms*, pages 573–582, 1994.

[4] S. E. Baker and R. M. Gray. Image compression using non-adaptive spatial vector quantization. *Proc. Conf. Rec. Sixteenth Asilomar Conf. Circuits, Syst., Comput.*, pages 55–61, October 1982.

[5] S. E. Baker and R. M. Gray. Differential vector quantization of automatic imagery. *Proc. Int. Picture Coding Symposium*, March 1983.

[6] Z. Barahav, D. Malah, and E. Karnin. *Hierarchical Interpretation of Fractal Image Coding and Its Applications*, pages 91–117. In Fisher [60], 1995.

[7] M. Barlaud, P. Solé, T. Gaidon, M. Antonini, and P. Mathieu. Pyramidal lattice vector quantization for multiscale image coding. *IEEE Transactions on Image Processing*, 3(4):367–381, 1994.

[8] M. F. Barnsley. *Fractal everywhere.* Academic Press, New York, 1988.

[9] M. F. Barnsley and S. G. Demko. Iterated function systems and the global construction of fractals. *Proc. Roy. Soc. Lond. A*, 399:243–275, 1985.

[10] M. F. Barnsley, S. G. Demko, J. H. Elton, and J. S. Geronimo. Invariant measures for Markov processes arising from iterated function systems with place-dependent probabilities. *Annales de l'Institute Henri Poincaré*, 24(3):367–394, 1988.

[11] M. F. Barnsley, V. Ervin, D. Hardin, and J. Lancaster. Solution of an inverse problem for fractals and other sets. *Proc. Natl. Acad. Sci. USA*, 83:1975–1977, April 1986.

[12] M. F. Barnsley and L. P. Hurd. *Fractal Image Compression.* AK Peters Ltd., Wellesley, 1993.

[13] M. F. Barnsley, A. Jacquin, F. Malassenet, L. Reuter, and A. D. Sloan. Harnessing chaos for image synthesis. *Computer Graphics*, 22(4):131–140, August 1988.

[14] M. F. Barnsley and A. D. Sloan. A better way to compress images. *Byte*, pages 215–223, January 1988.

[15] K. U. Barthel. Entropy constrained fractal image coding. *Fractals*, 1996. to appear.

[16] K. U. Barthel, J. Schüttemeyer, T. Voyé, and P. Noll. A new image coding technique unifying fractal and transform coding. In *IEEE International Conference on Image Processing*, pages 112–116, Austin, Texas, November 1994.

[17] K. U. Barthel and T. Voyé. Adaptive fractal image coding in the frequency domain. In *Proceedings of International Workshop on Image Processing: Theory, Methodology, Systems and Applications*, Budapest, Hungary, June 1994.

[18] K. U. Barthel and T. Voyé. Three-dimensional fractal video coding. In *ICIP*, volume 3, pages 260–263, Washington, D.C., 1995.

[19] G. Battle. A block spin construction of wavelets. part I lemarié functions. *Comm. Math. Phys.*, 1995.

[20] J. M. Beaumont. Image data compression using fractal techniques. *BT Tech-Nol. J.*, 9(4):93–109, 1991.

[21] T. J. Bedford, F. M. Dekking, M. Breeuwer, M. S. Kean, and D. Van Schooneveld. Fractal coding of monochrome images. *Signal Processing: Image Communication*, 6:405–419, 1994.

[22] L. Berger, J.-P. Mariot, and C. Launay. A new formulation for fast image coding using quadtree representation. *Pattern Recognition Letters*, 13:425–432, 1992.

[23] E. Bertin. Diagrammes de Voronoï 2D et 3D : Applications en analyse d'images. *Thèse de l'Université Joseph Fourier, Grenoble I, France*, 1994.

[24] A. Bogdan. Multiscale (inter/intra frame) fractal video coding. In *Proc. IEEE International Conference on Image Processing. ICIP '94*, Austin, TX, November 1994.

[25] P. Bolon. Filtrages d'ordre, vraisemblance et optimalité des prétraitements d'image. *Traitement du signal*, 9(3):225–250, 1992.

[26] A. Bowyer. Computing dirichlet tessellations. *The computer J.*, 24(2):162–166, 1981.

[27] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[28] A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-28:562–574, December 1980.

[29] C. A. Cabrelli, B. Forte, U. M. Molter, and E. R. Vrscay. Iterated fuzzy set systems: A new approach to the inverse problem for fractals and other sets. *Journal of Mathematical Analysis and Applications*, 171(1):79–100, November 1992.

[30] J. F. Canny. Finding edges and lines in images. Technical Report 720, Artificial Intelligence Lab., M.I.T., Cambridge, MA, June 1983.

[31] J.-M. Chassery, F. Davoine, and E. Bertin. Compression fractale par partitionnements de delaunay. In *Quatorziemme Colloque GRETSI*, volume 2, pages 819–822, Juan-Les-Pins, France, 1993.

[32] J.-M. Chassery and A. Montanvert. *Géométrie discrète en analyse d'images.* Edition Hermès, 1991.

[33] X. Chen and F. Schmitt. Split-and-merge image segmentation based on Delaunay triangulation. In *Proc. of The 7th Scandinavian Conf. on Image Analysis, Aalborg, Denmark*, pages 910–917, August 1991.

[34] J.-P. Cocquerez, S. Philipp, P. Bolon, J.-M. Chassery, D. Demigny, C. Graffigne, A. Montanvert, R. Zeboudj, and J. Zerubia. *Analyse d'images : filtrage et segmentation.* Masson, 1995. ouvrage collectif.

[35] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. Technical Report TM-11217-900529-07, AT&T Bell Labs.

[36] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups.* Springer-Verlag, New York, 1988.

[37] P. C. Cosman, R. M. Gray, and R. A. Olshen. Evaluating quality of compressed medic images: SNR, subjective rating, and diagnostic accuracy. *Proceedings of the IEEE*, 82(6):919–932, 1994.

[38] R. E. Crochiere, S. A. Webber, and J. L. Flanagan. Digital coding of speech in subbands. *Bell Syst. Tech. Journal*, 55:1069–1085, 1976.

[39] I. Daubechies. Orthonormal bases of compactly supported wavelets on a theme. *Comm. Pure Appl. Math.*, 41:909–996, 1988.

[40] G. Davis. Adaptive self-quantization of wavelet subtrees: a wavelet-based theory of fractal image compression. *SPIE conference on mathematical imaging: wavelet applications in signal and image processing*, July 1995.

[41] F. Davoine, M. Antonini, J.-M. Chassery, and M. Barlaud. Fractal image compression based on Delaunay triangulation and vector quantization. *IEEE Transactions on Image Processing - Special Issue on Vector Quantization*, February 1996.

[42] F. Davoine, E. Bertin, and J.-M. Chassery. From rigidity to adaptive tesselations for fractal image compression: Comparative studies. In *8th Workshop IEEE IMDSP*, pages 56–57, Cannes, France, September 1993.

[43] F. Davoine, E. Bertin, and J. M. Chassery. An adaptive partition for fractal image coding. *Fractals*, 1996. to appear.

[44] F. Davoine and J.-M. Chassery. Adaptive delaunay triangulation for attractor image coding. In *ICPR*, pages 801–803, Jerusalem, 1994.

[45] F. Davoine and J.-M. Chassery. Compression d'images par fractales. In *Journée d'études et d'échanges, "Nouvelles techniques pour la compression et la représentation des signaux audiovisuels"*, pages 169–176, CNET - CCETT, Rennes, France, 1995.

[46] F. Davoine and J. M. Chassery. Un partitionnement adaptatif en triangles et quadrilatères pour la compression des images par fractales. In *15th Workshop GRETSI*, volume 2, pages 721–724, Juan-Les-Pins, France, 1995.

[47] F. Davoine, J. Svenson, and J.-M. Chassery. A mixed triangular and quadrilateral partition for fractal image coding. In *ICIP*, volume 3, pages 284–287, Washington, D.C., 1995.

[48] R. Deriche. Fast algorithms for low-level vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87, January 1990.

[49] C. Diab, R. Prost, and R. Goutte. Exact subband image decomposition/reconstruction by DCT. *Signal Processing: Image Communication*, 1992.

[50] F. Dudbridge. *Image approximation by self-affine fractals*. PhD thesis, University of London, 1992.

[51] F. Dudbridge. Fast image coding by a hierarchical fractal construction. In *Preprint of the UCSD, San-Diego*, 1995.

[52] F. Dudbridge. *Least-Squares Block Coding by Fractal Functions*, pages 229–241. In Fisher [60], 1995.

[53] M. Dudon, C. Roux, and G. Eude. Treillis actif : estimation de mouvement et interpolation temporelle. In *Journée d'études et d'échanges, "Nouvelles techniques pour la compression et la représentation des signaux audiovisuels"*, CNET - CCETT, Rennes, France, 1995.

[54] J. H. Elton. An ergodic theorem for iterated maps. *Ergodic Theory and Dynamical Systems*, 7:481–488, 1987.

[55] D. Esteban and C. R. Galand. Application of quadrature mirror filters to split band voice coding schemes. In *Proceedings of ICASSP*, pages 191–195, Hartford CT, May 1977.

[56] K. Falconer. *Fractal Geometry.* Wiley, Chicester, 1990.

[57] P. Fiche, V. Ricordel, and C. Labit. Etude d'algorithmes de quantification vectorielle arborescente pour la compression d'images fixes. *Publication interne N 807, IRISA*, January 1994.

[58] M. A. Fischler and H. C. Wolf. Locating perceptually salient points on planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):113–129, February 1994.

[59] Y. Fisher. A discussion of fractal image compression. In Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe, editors, *Chaos and Fractals: New Frontiers of Science*, chapter Appendix A, pages 903–919. Springer-Verlag, New York, 1992.

[60] Y. Fisher, editor. *Fractal Image Compression: Theory and Application to Digital Images.* Springer Verlag, New York, 1995.

[61] Y. Fisher. *Fractal Image Compression with Quadtrees*, pages 55–77. In Fisher [60], 1995.

[62] Y. Fisher, E. W. Jacobs, and R. D. Boss. Iterated transform image compression. Technical Report 1408, Naval Ocean Systems Center, San Diego, CA 92152-5000, April 1991.

[63] Y. Fisher, E. W. Jacobs, and R. D. Boss. Fractal image compression using iterated transforms. In James A. Storer, editor, *Image and Text Compression*, chapter 2, pages 35–61. Kluwer Academic Publishers, Boston, MA, 1992.

[64] Y. Fisher and S. Menlove. *Fractal Encoding with HV Partitions*, pages 119–136. In Fisher [60], 1995.

[65] Y. Fisher, D. Rogovin, and T. P. Shen. Fractal (self-VQ) encoding of video sequences. In *Proceedings of the SPIE: Visual Communications and Image Processing*, Chicago, IL, September 1994.

[66] J. H. Friedman and J. L. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3(3):209–226, 1977.

[67] D. Le Gall and A. Tabatabai. Sub-band of digital images using symmetric kernel filters and arithmetic coding techniques. In *Proceedings of ICASSP*, pages 761–764, 1988.

[68] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1993.

[69] M. Gharavi-Alkhansari and T. S. Huang. A fractal-based image block-coding algorithm. In *Proceedings of ICASSP*, pages 345–348, 1993.

[70] P. J. Green and R. Sibson. Computing dirichlet tessellation in the plane. *The computer J.*, 21:168–173, 1978.

[71] F. Haurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM computing surveys*, 33(3):345–405, 1991.

[72] P. S. Heckbert. Fundamentals of texture mapping and image warping. Technical Report 516, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley. CA 94720, June 1989.

[73] S. L. Horowitz and T. Pavlidis. Picture segmentation by directed split and merge procedure. *Proc. 2nd Int. Joint Conf. Pattern Recognition*, pages 424–433, 1974.

[74] P. G. Howard and J. S. Vitter. Arithmetic coding for data compression. *Proceedings of the IEEE*, 82(6):857–865, 1994.

[75] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proc. of the IRE*, 40:1098–1101, September 1952.

[76] L. P. Hurd, M. A. Gustavus, and M. F. Barnsley. Fractal video compression. *Compcon Spring., conf. 37*, pages 41–42, 1992.

[77] B. Hürtgen. Contractivity of fractal transforms for image coding. *Electronics Letters*, 29(20):1749–1750, September 1993.

[78] B. Hürtgen and P. Büttgen. Fractal approach to low rate video coding. *Proceedings of SPIE*, 2094:120–131, 1993.

[79] B. Hürtgen and T. Hain. On the convergence of fractal transforms. In *Proceedings of ICASSP*, pages 561–564, 1994.

[80] B. Hürtgen, F. Müller, and C. Stiller. Adaptive fractal coding of still pictures. *Picture Coding Symposium*, 1993.

[81] J. Hutchinson. Fractals and self-similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

[82] E. W. Jacobs, R. D. Boss, and Y. Fisher. Fractal-based image compression, II. Technical Report 1362, Naval Ocean Systems Center, San Diego, CA 92152-5000, June 1990.

[83] E. W. Jacobs, Y. Fisher, and R. D. Boss. Image compression: A study of the iterated transform method. *Signal Processing*, 29:251–263, 1992.

[84] A. E. Jacquin. *A fractal theory of iterated Markov operators on spaces of measures with applications to digital image coding.* PhD thesis, Georgia Institute of Technology, 1989.

[85] 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.

[86] A. E. Jacquin. Fractal image coding: A review. *Proceedings of the IEEE*, 81(10):1451–1465, October 1993.

[87] A. K. Jain. Image data compression: A review. *Proceedings of the IEEE*, 69(3):349–389, 1981.

[88] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall Information and System Sciences series, 1989.

[89] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1421, 1993.

[90] G. Jourdain. Théorie de l'information. *ENSIEG, Grenoble*, pages 1–133, 1992.

[91] R. G. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 29(6), December 1981.

[92] Klinger. Data structures and pattern recognition. In *Proc. Int. Joint Conf. on Pattern Recognition*, pages 497–498, 1973.

[93] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[94] C. Kotropoulos, I. Pitas, and A. Maglara. Voronoi tessellation and Delaunay triangulation using euclidian disk growing in Z2. In *Proceedings of ICASSP*, pages 29–32, 1993.

[95] W. G. Kropatsch, M. A. Neuhausser, I. J. Leitgeb, and H. Bischof. Combining pyramidal and fractal image coding. In *Proc. 11th ICPR, The Hague, The Netherlands*, volume 3, pages 61–64, 1992.

[96] H. Krupnik, D. Malah, and E. Karnin. Fractal representation of images via the discrete wavelet transform. In *IEEE 18th Conv. of EE in Israel*, Tel-Aviv, March 1995.

[97] M. Kunt, A. Ikonomopoulos, and M. Kocher. Second-generation image-coding techniques. *Proceedings of the IEEE*, 73(4):549–574, 1985.

[98] F. Lallauret and D. Barba. Comparaison de méthodes de prédiction et de quantification adaptatives dans le codage intra-image d'images de télévision par décomposition en sous-bandes. *Traitement du Signal*, 12(1):73–92, 1995.

[99] M. S. Lazar and L. T. Bruton. Fractal block coding of digital video. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(3):297–308, June 1994.

[100] S. Lepsøy. *Attractor Image Compression - Fast Algorithms and Comparisons to Related Techniques*. PhD thesis, The Norwegian Institute of Technology, Trondheim, June 1993.

[101] J. Lévy-Véhel. About lacunarity, some links between fractal and integral geometry. *ICCV*, 1990.

[102] J. Lévy-Véhel and A. Gagalowicz. Fractal approximation of 2-D object. Technical Report 1187, INRIA - Rocquencourt, France, 1990.

[103] J. Lévy-Véhel and P. Mignot. Multifractal segmentation of images. *Fractals*, 2(3):371–377, 1994.

[104] H. Li and R. Forchheimer. A new motion-compensated technique for video compression. In *International Conference on Accoustics, Speech and Signal Processing*, pages 441–444, 1993.

[105] H. Lin and A. N. Venetsanopoulos. Incorporating nonlinear contractive functions into the fractal coding. In *Proc. Int. Workshop on Intelligent Signal Processing and Communication Systems*, pages 169–172, Seoul, Korea, October 1994.

[106] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantization design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[107] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, IT-28(2), March 1982.

[108] L. Lundheim. *Fractal Signal Modelling for source coding*. PhD thesis, The Norwegian Institute of Technology, Trondheim, September 1992.

[109] E. Lutton and J. Lévy-Véhel. Optimization of fractal functions using genetic algorithms. In *Fractal'93*, Londres, 1993.

[110] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[111] B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Co., San Francisco, 1982.

[112] G. Mantica and A. Sloan. Chaotic optimization and the construction of fractals: solution of an inverse problem. *Complex Systems*, 3:37–62, 1989.

[113] J. Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, IT-6:7–12, March 1960.

[114] Y. Meyer. Principe d'incertitude, bases hilbertiennes et algèbres d'opérateurs. *Séminaire Bourbaki*, 662:28–37, 1986.

[115] Y. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, 1990.

[116] D. M. Monro. Class of fractal transforms. *Electronics Letters*, 29(4):362–363, February 1993.

[117] D. M. Monro. Fractal transforms: Complexity versus fidelity. In *Image Processing: Theory and Applications, G. Vernazza, A. N. Venetsanopoulos, C. Braccini (Eds.) Elsevier Science Publishers B.V.*, pages 45–48, 1993.

[118] D. M. Monro and F. Dudbridge. Rendering algorithms for deterministic fractals. *IEEE computer graphics and applications*, pages 32–41, January 1995.

[119] D. M. Monro and J. A. Nicholls. Low bit rate colour fractal video. In *ICIP*, volume 3, pages 264–267, Washington, D.C., 1995.

[120] D. M. Monro and S. J. Woolley. Fractal image compression without searching. In *Proceedings of ICASSP*, volume V, pages 557–560, 1994.

[121] T. Murakami, K. Asai, and E. Yamazaki. Vector quantizer of video signals. *Elecron. Lett.*, 7:1005–1006, November 1982.

[122] Y. Nakaya and H. Harashima. An iterative estimation method using triangular patches for motion compensation. *SPIE Visual Communications and Image Processing*, 1605:546–557, 1991.

[123] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: A review. *IEEE Transactions on Communications*, 36(8):957–971, 1988.

[124] M. Nelson. *La compression de donnees : texte, images, sons*. Editions Dunod, 1993.

[125] J. P. Nougier. *Méthodes de calcul numérique*. Edition Masson, 3ieme édition, 1987.

[126] M. Novak. Attractor coding of images. *Licentiate Thesis, Department of Electrical Engineering, Linköping University*, 1993.

[127] H. J. Nussbaumer and m. Vetterli. Pseudo quadrature mirror filters. *Digital Signal Processing*, pages 8–12, 1984.

[128] G. E. Øien. *L2-optimal attractor image coding with fast decoder convergence*. PhD thesis, The Norwegian Institute of Technology, Trondheim, April 1993.

[129] G. E. Øien, Z. Baharav, S. Lepsøy, E. Karnin, and D. Malah. A new improved collage theorem with applications to multiresolution fractal image coding. In *International Conference on Accoustics, Speech and Signal Processing*, 1994.

[130] G. E. Øien and S. Lepsøy. Fractal-based image coding with fast decoder convergence. *Signal Processing*, 40:105–117, October 1994.

[131] A. Okabe, B. Boots, and Sugihara K. *Spatial tesselations: concept and applications of the Voronoi diagram*. John Wiley and Sons, New York, 1992.

[132] E. Polidori and J.-L. Dugelay. Zooming using Iterated Function Systems. *Fractals*, 1996. to appear.

[133] J. P. Preparata and M. I. S. Shamos. *Computational Geometry, an Introduction*. Springer Verlag, NewYork, 1988.

[134] M. Rabbani and P. W. Jones. *Digital Image Compression*. SPIE Optical Engineering Press, Vol. TT 7, 1991.

[135] B. Ramamurthi and A. Gersho. Image vector quantization with a perceptually-based cell classifier. *IEEE Proc. Int. Conf. Acoust, Speech, Signal Processing*, March 1984.

[136] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Transactions on Communications*, 34(11):1105–1115, 1986.

[137] E. Reusens. Partitioning complexity issue for iterated functions systems based image coding. In *Proc. of VII European Signal Processing Conference*, volume 1, pages 171–174, Edinburg, U.K., September 1994.

[138] V. Ricordel and C. Labit. Quantification vectorielle par emboîtage d'une hiérarchie de réseaux régulier de points. Technical Report 2667, INRIA, October 1995.

[139] R. Rinaldo and G. Calvagno. Image coding by block prediction of multiresolution subimages. *IEEE Transactions on Image Processing*, pages 909–920, July 1995.

[140] R. Rinaldo and A. Zakhor. Inverse and approximation problem for two-dimensional fractal sets. *IEEE Transactions on Image Processing*, 3(6):802–820, November 1994.

[141] A. Rosenfeld. Quadtrees and pyramids for pattern recognition and image processing. In *Proc. 5th. Int. Conf. on Pattern Recognition*, pages 802–811, University of Maryland, 1980.

[142] A. Saadane, H. Senane, and D. Barba. On the design of psychovisual quantizers for a visual subband image coding. In *VCIP*, Chicago, Illinois (USA), September 1994.

[143] M. J. Sabin and R. M. Gray. Product code vector quantizers for waveform and voice coding. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP–32:474–488, June 1984.

[144] P. Salembier. Morphological multiscale segmentation for image coding. *Signal Processing*, 38:359–386, 1994.

[145] H. Samet. Region representation: quadtrees from binary arrays. *CVGIP*, 13:88–93, 1980.

[146] H. Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16:187–260, 1984.

[147] D. Saupe. Accelerating fractal image compression by multi-dimensional nearest neighbor search. In J. A. Storer and M. Cohn, editors, *Proc. Data Compression Conference (DCC '95)*. IEEE Computer Society Press, March 1995.

[148] D. Saupe and R. Hamzaoui. A review of the fractal image compression literature. *Computer Graphics*, 28(4):268–276, 1994.

[149] F. Schmitt and H. Borouchaki. Algorithme rapide de maillage de delaunay dans Rd. In Springer, editor, *Proc. des journées de géométrie algorithmique*, pages 131–133, June 1990.

[150] J. Serra. *Image analysis and mathematical morphology, Vol 2: Theoretical advances*. London, Academic Press, 1988.

[151] C. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[152] J. Signes. Geometrical interpretation of IFS based image coding. *Fractals*, 1996. to appear.

[153] B. Simon. Explicit link between local fractal transform and multiresolution transform. In *ICIP*, volume 1, pages 278–281, Washington, D.C., 1995.

[154] M. J. T. Smith and T. P. Barnwell. Exact reconstruction techniques for tree-structured subband coders. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34:434–441, June 1986.

[155] E. Thiel and A. Montanvert. Etude et amélioration des distances du chafrein pour l'analyse d'images. *Techniques et sciences informatiques*, 11(4):9–41, 1992.

[156] L. Thomas and F. Deravi. Region-based fractal image compression using heuristic search. *IEEE Transactions on Image Processing*, 4(6):832–838, 1995.

[157] R. Y. Tsai and T. S. Huang. Estimating three didensional motion parameters of a rigid planar patch. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-29(6):1147–1152, December 1981.

[158] Axel van de Walle. Relating fractal image compression to transform methods. Master's thesis, University of Waterloo, Dept. of Applied Mathematics, 1995.

[159] Axel van de Walle. Merging fractal image compression and wavelet transform methods. *Fractals*, 1996. to appear.

[160] G. Vines. *Signal Modelling With Iterated Function Systems.* PhD thesis, Georgia Institute of Technology, May 1993.

[161] R. Vrscay. Moment and collage methods for the inverse problem of fractal construction with iterated function systems. In *Fractal 90 conference*, June 1990.

[162] K. Wall and P. E. Danielsson. A fast sequential method for polygonal approximation of digitized curves. *CVGIP*, 28:220–227, 1984.

[163] Y. Wang and O. Lee. Active mesh - a feature seeking and tracking image sequence representation scheme. *IEEE Transactions on Image Processing*, 3(5):610–624, 1994.

[164] D. L. Wilson, J. A. Nicholls, and D. M. Monro. Rate buffered fractal video. In *Proceedings of the ICASSP*, volume V, pages 505–508, 1994.

[165] S. J. Woolley and D. M. Monro. Rate-distortion performance of fractal transforms for image compression. *Fractals*, 2(6):395–398, 1994.

[166] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, pages 337–343, May 1977.

[167] J. Ziv and A. Lempel. Compression of individual sequances via variable rate coding. *IEEE Transactions on Information Theory*, IT-24:530–536, 1978.

[168] G. Zorpette. Fractals: Not just another pretty picture. *IEEE Spectrum*, October 1988.

[169] S. W. Zucker. Region growing: childhood and adolescence. *CGIP*, 5:382–399, 1976.