République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLAB – Blida 1
Faculté des sciences
Département de Mathématique

Mémoire de master
En Recherche Opérationnelle

Compression d'images Par Les Fractales

Prépare par :

Mr Belgheddouche Med Hamaed

Encadré par :

M^elle Benblidia Nadjia

Members de jury:

President: M^elle Reguieg F.Zohra – MAA– Université de Blida 1
Examinateur : M Talbi M.Amine–MCB–Université de Blida 1
Rapporteur : M^elle Benblidia Nadjia – Pr – Université de Blida 1

Année d'étude : 2013/2014

Remerciement

- Je tiens, avant tout, à exprimer ma profondeur gratitude à madame **Benblidia Nadjia** qui a accepté d'assurer la direction de ce mémoire. Le constat intérêt qu'il a manifesté pour ce travail, les orientations qu'il m'a prodigué, sa disponibilité et ses encouragements ont été autant de facteurs déterminants dans l'élaboration de ce travail, qu'il trouve ici la marque de ma plus sincère reconnaissance.
- Je tiens à remercier monsieur Azrou Isghi Ali et tous l'équipe de SNC-LAVALIN de Hadjrot -nos pour leur encouragements et conseils dont il fait preuve à mon égard.
- Mes profonds remerciements vont également à tous la famille de **technicome d'Ali Chentir** - Cherchell qui m'a beaucoup aidée dans l'élaboration de ce modeste travail.

BELGHEDDOUCHE Mohamed

A tous ceux qui, par leur encouragement ou leur amitié, ont contribue à l'aboutissement de ce travail.

ملخص

-كثيرون لديهم فكرة عن معنى كلمة" كسورية "ولكن قلة تعرف حقا ما هو عليه .اخترع في عام 1975 من قبل بينوا ماندلبروت، هذه الكلمة تعني كلا من" كسر "و" غير النظامية"، ويستخدم لتمثيل الأشكال الهندسية التي لها شكل غير منتظم للغاية .هذه التمثيلات تظهر عناصر للكشف على مجموعة واسعة من المقاييس. حيث تكون بنية الاجزاء مماثلة لبنية الكل.

-منذ أن لوحظ أن فركتلات هي عالميا موجودة في الطبيعة .لذلك فهم هذا المفهوم من أجل فهم أفضل لبيئتنا وتطبيقات فركتلات عديدة .في هذه المذكرة سوف نركز على تطبيقات الفركتلات في ضغط الصور المختلفة الحجم و النوع .

-إن ضغط الصور كسورية يمكن من الفك السريع و اتخاذ قرار مستقل ولكن يعاني من تأخير في الوقت اللازم للترميز .ويعرض هذا العمل نهجا للحد من الوقت اللازم للحساب باستخدام الشجرة الكوادرتية للترميز وفك شفرة الصورة في نفس الوقت .وأظهرت النتائج التجريبية أنه لدى تطبيق الشجرة الكوادرتية على أنواع مختلفة من الصور (ج ب ج , ب م ب....) قد وصلت إلى التوفير الملحوظ من الوقت مقارنة مع تحليل هوفمان وضغط المويجات .لم تتغير نوعية الصورة عن طريق هذا النهج وكانت هناك زيادة طفيفة في نسبة الضغط قليلا.

Résume

- Plusieurs ont une idée de la signification du mot « fractale » mais peu savent vraiment de quoi il s'agit. Inventé en 1975 par Benoit Mandelbrot, ce mot signifie à la fois « brisé » et « irrégulier » et sert à représenter géométriquement des objets dont la forme est extrêmement irrégulière. Ces objets présentent des éléments discernables sur une large gamme d'échelles et dont les parties ont approximativement la même structure que le tout.
- Depuis, on a observé que les fractales sont présentes de façon universelle dans la nature. Ainsi, comprendre ce concept permet de mieux comprendre notre environnement et les applications des fractales sont nombreuses. Dans ce mémoire nous intéressons de l'application des fractales sur la compression d'image.
- La compression fractale d'images permet un décodage rapide et une indépendance de la résolution mais souffre d'une lenteur dans le temps de codage. Le présent travail présente une approche visant à réduire le temps de calcul en utilisant la décomposition Quadtree pour code et décode l'image au mêmes temps .Les résultats expérimentaux ont montré que Quadtree appliquée à des différentes type des images (jpg, bmp, png..) a atteint un gain de temps remarquable comparent à la décomposition Huffman et la compression par ondelettes. La qualité de l'image n'a pas été altérée par cette approche et le taux de compression a légèrement augmenté.

Abstract

Many have an idea of what the word "fractal" but few really know what it is. Invented in 1975 by Benoit Mandelbrot, this word means both "broken" and "irregular" and is used to represent geometric objects whose shape is extremely irregular. These objects are detectable elements on a wide range of scales and whose parts have approximately the same structure as the whole.

- Since it was observed that fractals are universally present in nature. So understand this concept to better understand our environment and applications of fractals are numerous. In this paper we focus on the application of fractal image compression.
- The fractal image compression enables rapid decoding and resolution independence but suffers from a delay in encoding time. This work presents an approach to reduce the computation time using Quadtree decomposition to code and decode the image at the same time .The experimental results showed that Quadtree applied to different types of pictures (jpg, bmp, png. .) has reached a remarkable saving of time compared to the Huffman decomposition and wavelet compression. The quality of the image was not altered by this approach and the compression ratio increased slightly.

Table des matières

Introduction Générale

Dans les années 70, le champ d'action des mathématiques a pris une nouvelle dimension par l'ajout de la géométrie fractale. Depuis, il a été démontré que les fractales peuvent servir de modèle pour représenter la géométrie de la nature. Il n'est donc pas surprenant que celles-ci s'enrichissent de plus en plus d'applications dans divers domaines. Cependant, ce sujet d'actualité n'est bien connu que de l'élite et des passionnés des mathématiques mais aurait avantage à être intégré comme objet d'étude dans l'enseignement. Parallèlement, les jeunes semblent de moins en moins attirés par les études supérieures en mathématiques. Or, les fractales, étant à la fois visuellement attrayantes et intriguantes, possèdent des caractéristiques pour piquer la curiosité et stimuler le goût d'apprendre.

Au cours des dernières années, la demande et le développement de produits multimédias et de la capacité de stockage du dispositif de stockage ont augmenté à un rythme effréné. En raison de la compression qui est devenue une technique clé pour réduire la taille du nombre de bits dans la mesure du possible. Un processus inverse connu sous le nom de décompression (décodage) peut être appliqué à des données compressées pour obtenir l'image reconstruite.

Récemment, une nouvelle méthode de compression est largement utilisé pour la compression d'image notamment la compression de l'image fractale. Il est utile dans différents domaines d'application et les champs de recherche pour compresser l'image. Dans cet article, la méthode fractale de compression d'image est concentrée, qui est une méthode efficace pour la compression d'image avec perte qui fonctionne sur la propriété auto-similarité dans diverses fractions d'images. Le principal défi aujourd'hui est IFS a pris du temps d'encodage et affectant la qualité de l'image, par conséquent, la recherche récente a porté sur une meilleure valeur de PSNR, taux de compression plus élevé et tente de réduire les temps d'encodage sans nuire à la qualité de l'image

Par ce mémoire, nous voulons faire une liaison enter les fractales comme des concepts mathématiques et des applications concret dans noter cas nous avons choisir la compression des images. Ce mémoire suppose plusieurs objectifs :

- établir une revue historique concernent tout les travaux précédent sur les fractales.
- revoir la théorie mathématique qui sous-tend l'approche d'un certain type de fractales : les *Iterated function systems* (IFS).
- applique cette théorie sur la compression des images.
- concevoir et développer un système de la compression et la décompression des différents types d'images (jpg, png, bmp,..).

Pour arriver à remplir ces tâches, nous proposons en premier lieu une revue de la littérature. Nous cherchons ensuite à dégager l'essentiel de la théorie des fractales afin de pouvoir l'appliquer a la compression des images utilisant la compression fractales.

Chapitre 1
Introduction sur les fractales

1.1 Introduction

Lorsque vous traitez avec la classe d'objets géométriques appelées fractales, la géométrie euclidienne classique ne suffit pas de décrire leur nature complexe. Dans les dernières décennies, une nouvelle branche de la géométrie, appelé la géométrie fractale, ont grandi et reçu beaucoup d'attention de la part d'une variété de domaines.

Ce chapitre décrit la théorie générale des fractales et leur géométrie. Tout d'abord, nous allons voir une brève explication de ce qu'est un fractal est vraiment. Puis quelques différentes notions de dimension, quelque chose de très important dans la géométrie fractale, seront décrits, suivis d'une description de quelques méthodes pour estimer la dimension fractale.

1.2 Historique sur Les Fractales

Depuis très longtemps, l'homme a été fascine par des formes géométriques particulières qu'il observa dans la nature et qu'il créa pour décorer ses habitations, ses vêtements …Par la suite, ces formes ont été étudies par des mathématiciens. Au milieu de 19^{ieme} siècle, pour un mathématicien, une courbe est continue et possède une tangente en chacun de ces points sauf sur certains points dits singulières et elles sont de dimension 1. Tout tracé qui n'entre pas dans ce modèle est pathologique, Vers la fin du 19^{ieme} siècle, quelques mathématiciens s'intéressent à des courbes continues sans tangentes, vite qualifies de « monstres » par les contemporains. Et pourtant ceux-ci ne font qu'anticiper ce que l'on va nommer la géométrie fractale, développée et popularisée par Benoit Mandelbrot.

Déjà, en 1980 *Giuseppe Peano*(1858-1930)[1] exhibe une courbe sous le nom de courbe de *Peano* , qui remplit véritablement tout un carrée . *Helge Von Koch*[1] , En 1904 , dessina le flacon de neige le plus célèbre de l'histoire des mathématiques qui est un exemple de courbe sans tangente en aucun point, D'autres exemples d'objets tels que la poussière de *Cantor*, Les tringles et éponges de Sierpinsky qui sont des créations de l'esprit humane et très éloignées de la nature, faisaient dire à *Poincaré* , pourtant traducteur de Cantor , que l'on ne devrait pas exhiber ces exemples qui mettent en défaut le raisonnement de nos pères et n'utilisant

aucune théorie .Or des mathématiciens , comme Jordon qui essaie de trouver une définition du mot « courbe » mieux adaptée a la diversité des modèles mathématiques ou Cantor qui établit la bijection entre les points d'un segment et ceux d'un carré , montrent la nécessite d'élargir la notion de dimension d'un objet , Dés 1935 une dimension fractionnaire (d) d'environ 1.26 a été attribue au flacon de Von Koch , Ainsi la dimension fractale constitue une généralisation de la notion de dimension entière, propre à la géométrie euclidienne. Par ailleurs, Ces objets géométriques douteux n'acquièrent un statut à part entière que dans les années 70, Grace au mathématicien français *Benoit Mandelbrot* qui fait en fait L'objet d'une nouvelle discipline mathématique : la géométrie fractale. Ces objets que Cantor, Peano, Von Koch, Sierpinsky …etc. ont inventes et décrivent mieux la nature fractale prend donc le relais et permet d'étudier avec succès, ces objets appelés objets fractales ou encore fractales.

Apres Mandelbrot, bien d'autres mathématiciens et informaticien ont utilisé les possibilités immenses, en termes de calculs et d'imageries, offertes par l'informatique pour développer cette théorie. Les fractales sont toutes définies de façon constructives et itérative .Cela implique que l'on ne peut en présenter que des approximations obtenus grâce à des calculs numériques [4].

1.2 Qu'est-ce qu'une fractale?

Dans son article fondateur Benoit Mandelbrot A inventé le terme de Fractale, et l'a décrit comme suit :

Un fractale est une forme géométrique approximative ou fragmentée qui peut être subdivisée en partis, dont chacune est (au moins approximativement) une copie de taille réduite de l'ensemble [3]. Cette particularité d'auto-similarité est très étonnante, et les fractales ont bien d'autres propriétés, plus fascinantes les unes que les autre.

1.3 Les premières Monstres

Durant prés de 50 ans, les mathématiciens ont créé plusieurs figures aux propriétés non-intuitives, généralement dans le but de trouver des contre-exemples à des croyances mathématiques. Ces objets considérés par plusieurs comme des «monstres» seront plus tard, la source d'inspiration de Mandelbrot qui l'amènera à fonder une nouvelle branche des mathématiques soit la géométrie fractale [1].

1.3.1 Une courbe continue sans dérivée

En 1875 *Karl Weierstrass* donne le premier exemple explicite de fonction continue n'ayant de dérivée en aucun point

$$W_0(t) = (1 - w^2)^{-\frac{1}{2}} \sum_{n=0}^{\infty} w^n \exp(2\pi i b^n t) \quad (1.1)$$

Ou $b > 1 \in \mathbb{R}$ et w s'écrit soit $w = b^H$ avec $0 < H < 1$ soit $w = b^{D-2}$ avec $1 < D < 2$.

1.3.2 L'ensemble triadique de Cantor

L'ensemble triadique de Cantor $K \subset [0,1]$ est défini en 1877 comme une intersection $K = \bigcap_{n \in N} K_n$ ou les ensembles K_n sont définies par récurrence, On part de $K_0 = [0;1]$. Pour passer à K_1 on retire a K_0 un intervalle ouvert centrale (central signifie ici symétrique par rapport au milieu du segment initial) de longueur 1/3 , précisément l'intervalle $\left]\frac{1}{3};\frac{2}{3}\right[$. L'ensemble K_1 est donc réunion de deux segments de longueur 1/3 : $[0, 1/3]$ *et* $[2/3\,, 1]$, pour passer a K_2 on retire a chacun de ces segments un intervalle ouvert de longueur 1/9. on obtient une réunion de 4 segments disjoints, chacun de longueur 1/9 ; précisément$[0,\ 1/9\ ], [2/9, 1/3 = 3/9], [2/3 = 6/9\,, 7/9]$ *et* $[8/9\,, 1]$ On construit ainsi par récurrence K_n qui réunion de 2^n segments disjoints, chacun de longueur $1/3^n$, et on passe a K_{n+1} en retirant a chacun des segments constitutifs de K_n un intervalle ouvert central de longueur $1/3^{n+1}$.

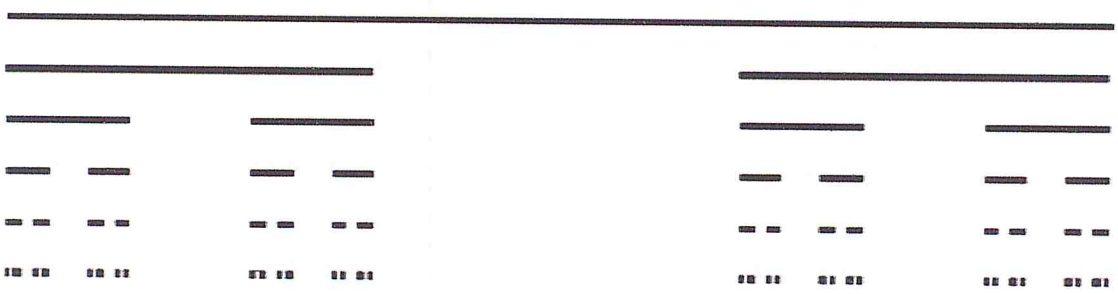


Fig. 1.1–Premières étapes de la construction de l'ensemble de Cantor

1.3.3 Une ligne qui remplit un carré

En 1890 *Peano* et *Hilbert* construisirent presque simultanément une courbe qui remplit un carré. Dans le premier cas, *Peano* a proposé une série de courbes semblables. Sa construction la plus célèbre consiste tout d'abord à tracer une diagonale du carré.

Pour réaliser la deuxième étape, on subdivise le carré initial en neuf carrés congrus et on parcourt tous les carrés en passant par une de leurs diagonales d'un seul trait de crayon tel qu'illustré a la figure 1.2. On reprend ensuite chacun des petits carrés qu'on subdivise à nouveau et on trace le même parcours. Le carré est entièrement recouvert lorsque le processus itératif tend à l' infini. Pour ce qui est de la construction proposée par *Hilbert*, on commence par diviser le carré initial en quatre carrés congrus et on relie le point central de chacun dans le sens horaire sans revenir au premier point. Ensuite, chaque carré est divisé à nouveau pour former quatre groupes de quatre carrés.

De la même façon, on relie les points centraux de façon a ce que le dernier point du groupe 1 soit relié avec le premier point du groupe 2 et ainsi de suite. En répétant cette itération jusqu'a l'infini on arrive à recouvrir le carré initial avec une courbe. Notons que dans les deux cas, la transformation de la courbe vers le carré est continue et surjective mais elle n'est pas injective. D'ailleurs, il ne peut en être autrement puisque, suite a la démonstration de *Cantor*, *Netto* prouva qu'une transformation bijective de l'intervalle [0, 1] vers le carré ne pouvait pas être continue [1].

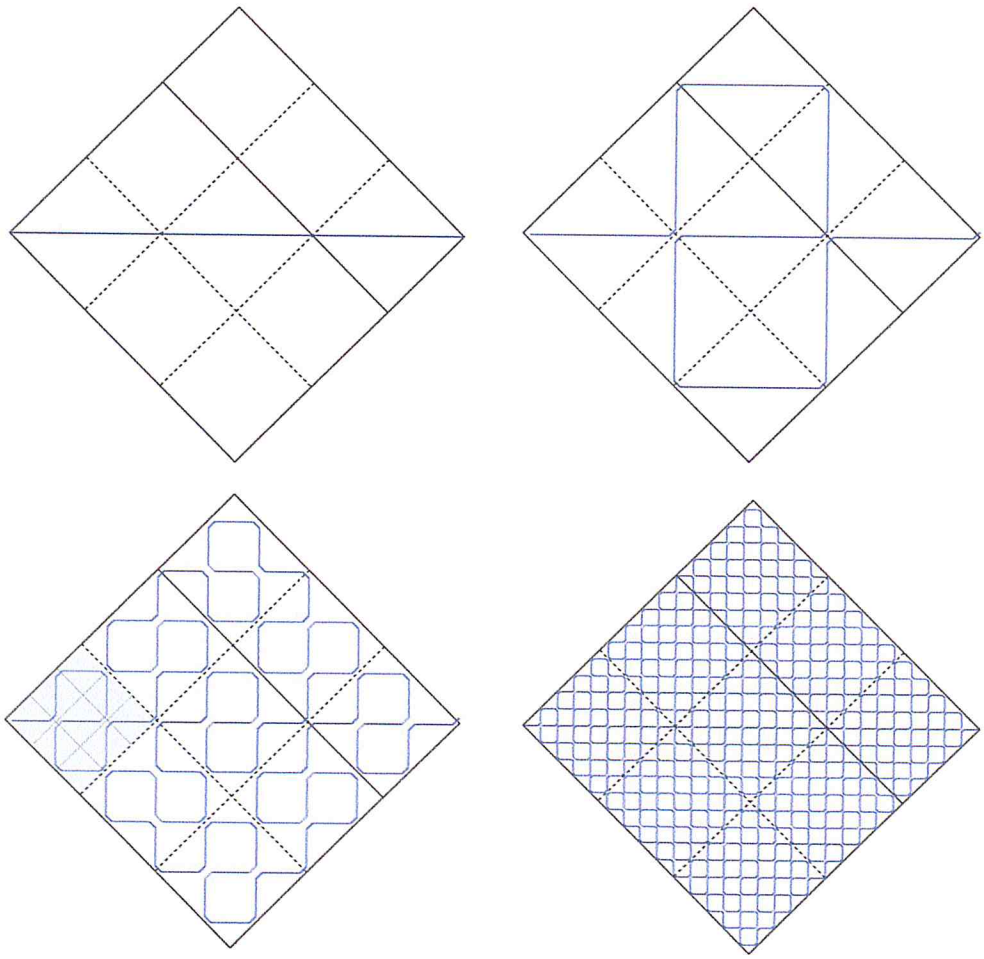


Fig. 1.2 – Courbe de *Peano* (Note : en réalité, la courbe parcourt les diagonales en entier et admet donc des points doubles. Sur l'illustration, les coins ont été arrondis pour faciliter la compréhension du parcours) [1].

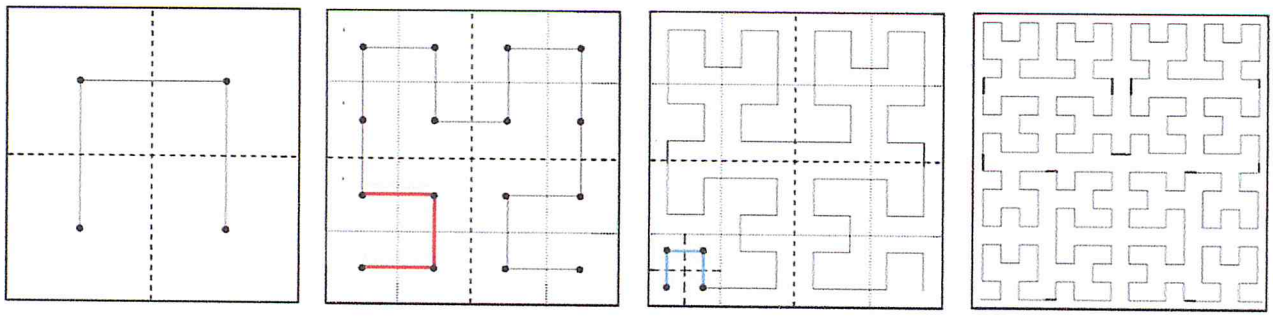


Fig. 1.3 – Courbe de Hilbert [1]

1.3.4 Courbe de Von Koch

Von Koch proposa en 1904 une construction extrêmement simple aboutissant à une courbe continue qui n'a pas de tangente. On prend un segment de longueur 1 et on remplace son tiers central par un «pic» formé de deux segments de longueur $\frac{l}{3}$. On refait le même processus pour chacun des quatres nouveaux segments et ainsi de suite. A l'infini, on obtient une courbe exclusivement formée de « pics» qui on le sait, n'admettent pas de tangente.

La courbe de Koch a une longueur infinie parce qu'a chaque fois qu'on applique les modifications ci-avant sur chaque segment de droite, La longueur totale augmente d'un tiers.

La surface délimitée par la courbe est cependant finie (car elle est contenue dans le demi-cercle dont le diamètre est le segment initial). Si on a choisi l'unité d'aire de telle sorte que le triangle construit à la première itération soit d'aire 1, alors l'aire de chacun des quatre triangles construits lors de la seconde itération est 1/9 : on a donc augmente l'aire totale de 4/9. pour l'itération n, on ajoute $4^{n-1} \times \left(\frac{1}{9}\right)^{n-1}$.

La surface totale s'obtient finalement en sommant une série géométrique :

$$\sum_{n=0}^{\infty} \left(\frac{4}{9}\right)^n = \frac{1}{1-4/9} = 9/5 \qquad (1.2)$$

La courbe de Koch constitue un exemple de courbe continue mais non dérivable en chacun de ses points

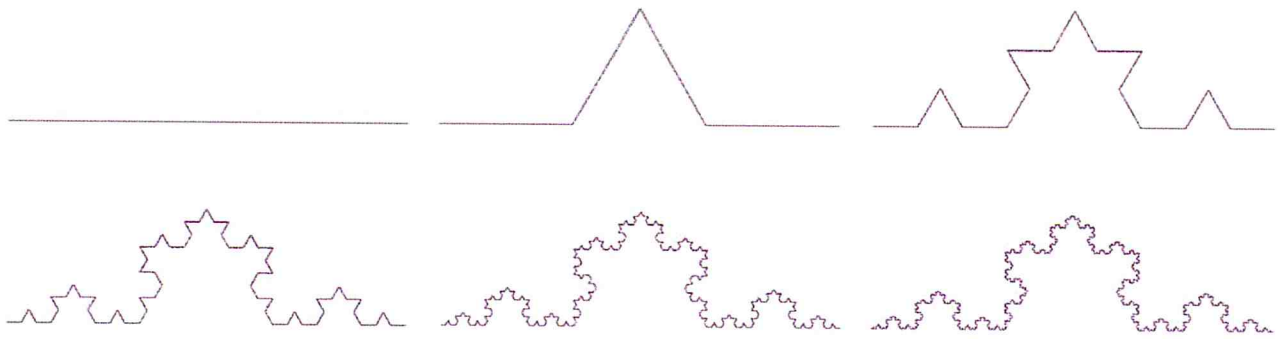


Fig. 1.4 –Etapes de la construction d'une courbe de Von Koch [4]

1.3.5 *Sierpinsky* : Triangle, Tapis et d'autres tétraèdres

Le triangle de Sierpinsky se construit à partir d'un triangle équilatéral, pour lequel on trace trois segments entre les trois milieux des cotes du triangle, ce qui délimite quatre nouveaux triangles. On enlève ensuite le petite triangle central, on obtient trois petites triangles qui touchent deux a deux par un sommet dont les longueurs des cotes sont la moitie de celles de départ et dont la surfaces est divisée par quatre. On répète cette procédure avec chacun des petites triangles obtenus et on poursuit le procédé jusqu'à l'infini. Ce même processus peut être généralisé à tous les polygones convexes réguliers. En prenant un carré et en lui retirant toujours le carré central, on obtient la Tapis de *Sierpinsky*[1]. Pour ce qui est de l'hexagone, il génère une figure qu'on appelle parfois le « napperon de Koch » puisque la frontière de son centre est constituée de trois courbes de Koch bout à bout formant ce qu'on appelle le « flocon de Koch ». De façon générale, pour un polygone à n côtes, la technique consiste à prendre récursivement n figures homothétiques de rapport $r_n = \frac{1}{2\sum_{k=0}^{\left[\frac{n}{4}\right]} cos\frac{2k\pi}{n}}$ Au polygone de départ et de les centrer aux sommets de celui-ci de sorte qu'elles soient toutes juxtaposées.

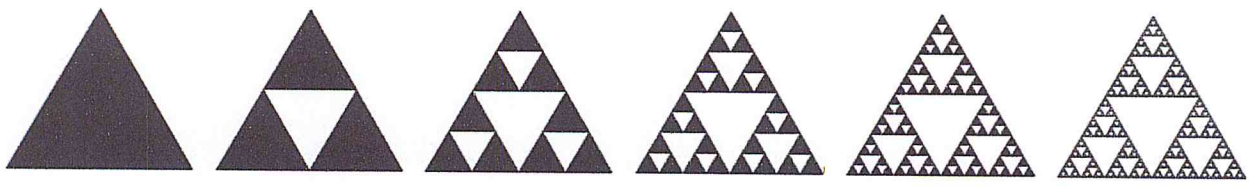


Fig. 1.5 – Illustration des premières étapes de la construction du triangle de Sierpinsky [3]

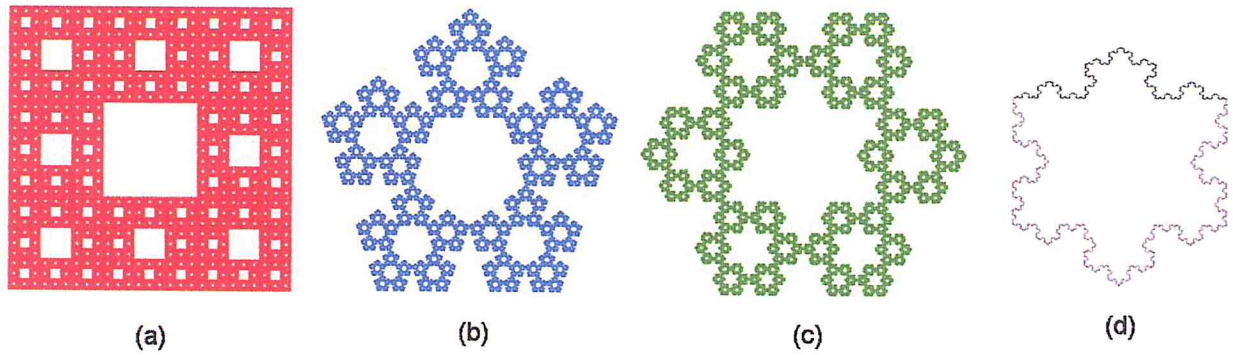


Fig. 1.6 – (a) Tapis de Sierpinsky, (b) Pentagone de Sierpinsky (c) Napperon de Koch : Hexagone de Sierpinsky dont le centre est délimité par un flocon de Koch (d) Flocon de Koch formé de trois courbes de Koch (en noir) [3]

1.4 La dimension

Une caractéristique importante de la géométrie fractale est qu'elle permet une caractérisation des irrégularités à différentes échelles que la géométrie euclidienne classique ne permet pas de. En conséquence, de nombreuses fonctions fractales ont été identifiés, dont la dimension fractale est une des plus importantes. Depuis les travaux d'Euclide, nous savons que la dimension d'un point est considéré comme 0, la dimension d'une ligne est 1, la dimension d'un carré est 2, et que la dimension d'un cube est 3. Environ, nous pouvons dire que la dimension d'un ensemble décrit combien d'espace l'ensemble remplit [9]. Nous allons construire la théorie de la dimension fractale de la définition de base euclidienne aux définitions plus mathématiquement exhaustive de *Hausdorff-Besicovitch*[1] et la dimension des boites (box-counting).

On peut se demander pourquoi il existe plusieurs définitions différentes de la dimension. Tout simplement parce que d'une certaine définition pourrait être utile à une fin, mais pas pour un autre. Dans de nombreux cas, les définitions sont équivalentes, mais quand ils ne sont pas, ce sont leurs propriétés particulières qui les rendent plus approprié pour la tâche à accomplir.

1.4.1 La dimension topologique

Il existe plusieurs façons de définir la dimension topologique. Dans tous les cas, les valeurs admises sont des entiers. Deux objets sont équivalents topologiquement s'il est possible de déformer l'un vers l'autre a l'aide d'un homéomorphisme : transformation bijective et continue (qui préserve la connexité). Ainsi, la dimension topologique d'un objet devrait être préservée sous une transformation homéo morphique. Poincaré a proposé une définition semblable à celle d'Euclide d'une telle dimension. Posons d'abord le vide de dimension-1. Ensuite, on procède par induction : si un objet

connexe peut être divisé en deux (ou plusieurs) objets disjoints en lui retirant une partie de dimension n (et qu'il n'est pas possible de le faire avec une partie de plus petite dimension) alors on dit qu'il est de dimension $n + 1$. Ainsi, un point ne peut pas être brisé en plusieurs morceaux donc il est de dimension $-1+1 = 0$. De plus, un nombre fini de points est totalement non-connexe et reste de dimension 0. Une ligne peut être brisée en deux lignes disjointes si on lui retire un point qui est de dimension 0, donc la ligne est de dimension $0 + 1 = 1$. Et ainsi de suite [1].

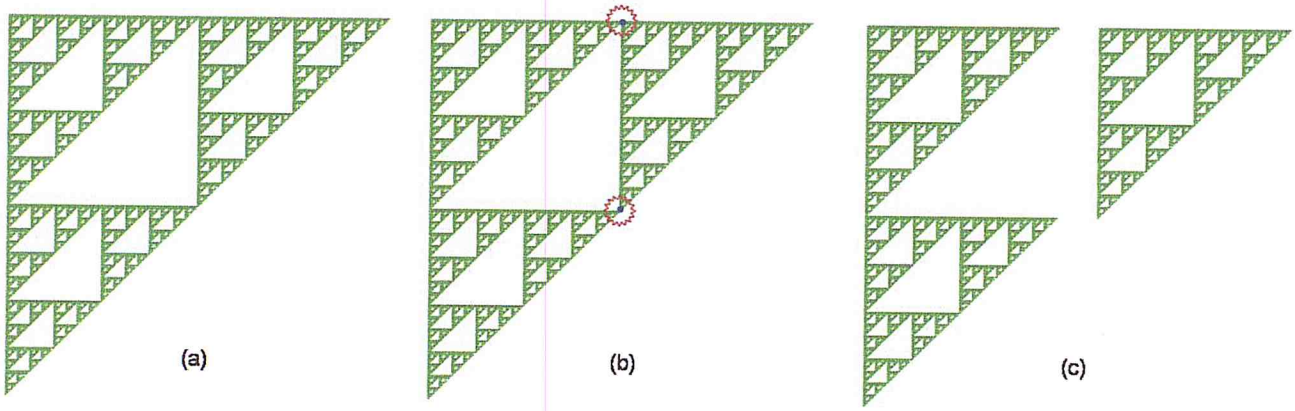


Fig. 1.7 – (a)La dimension topologique du triangle de Sierpinski est 1. (b) On retire un ensemble de deux points (dimension 0) (c) La figure est déconnectée en deux objets de dimension 1 (on peut leur retirer un ensemble fini de points pour les déconnecter et ainsi de suite) [1].

1.4.2 La dimension de Hausdorff-Besicovitch

Il est facile de calculer la mesure d'un objet auquel on a fait subir une contraction (ou une dilatation) de rapport k lorsque l'on travaille en dimension 1, 2 ou 3 : la mesure initiale est respectivement multipliée par k, k^2 ou k^3. De même pour les fractales, la mesure initiale est multipliée par $n = kd$ où d est la dimension fractale. Cette dernière diffère d'une dimension ordinaire : elle est un nombre réel quelconque. Une autre méthode pour déterminer la dimension fractale d'un objet est de compter le nombre moyen de motifs répétés contenus dans une sphère de rayon k centrée en un point donné de l'objet. Ce nombre de motifs est donné par $n = kd$ et la dimension fractale est ainsi égale à :

$$d = \frac{\ln n}{\ln k} \tag{1.3}$$

La dimension de Hausdorff peut être définie pour tout ensemble, mais elle n'est généralement pas aisée à calculer.

Pour un sous-ensemble non-vide U d'un espace euclidien de dimension n, R^n, on définit le diamètre de U, noté $diam(U)$ ou $|U|$ par : $diam(U) = |U| = sup\{|x - y| : x, y \in U\}$ où $|\cdot|$ est la distance euclidienne usuelle.

Si un ensemble F est recouvert par une collection dénombrable $\{U_i\}$ de diamètre au plus δ, c'est-à-dire $F \subset \bigcup_{i=1}^{\infty} Ui$ avec $0 < |Ui| \leq \delta$ pour tout i, on dit que $\{U_i\}$ est un δ - recouvrement de F.

Soient F un sous-ensemble de R^n et s un réel positif. Pour tout $\delta > 0$, on définit

$$H_\delta^s(F) = \inf\{\sum_{i=1}^{\infty} |U_i|^s : \{U_i\} \; est \; un \; \delta - recouvrement \; de \; F\} \quad (1.4)$$

Quand δ décroit, il y a moins de recouvrements possibles de F et $H_\delta^s(F)$ augmente, et donc a une limite quand $\delta \to 0$. On écrit

$$H^s(F) = \lim_{\delta \to 0} H_\delta^s(F) \quad (1.5)$$

Cette limite existe pour tout sous-ensemble F de R^n, et peut être 0 ou 1. On appelle $H^s(F)$ la mesure de Hausdorff s-dimensionnelle de F.

On admettra que H^s est une mesure borélienne sur R^n. La dimension de Hausdorff est la valeur de $\propto$ pour laquelle la $\propto$-mesure de recouvrement fait un saut de l'infini à 0 (voir figure 1.8). Il est admis que la dimension de Hausdorff est la plus ancienne et la plus importante. De fait, elle est définie pour tous les ensembles et elle est satisfaisante d'un point de vue mathématique puisqu'elle est basée sur une mesure. Cependant, elle est souvent difficile à évaluer.

C'est pourquoi, on préfère parfois utiliser la dimension des boites.

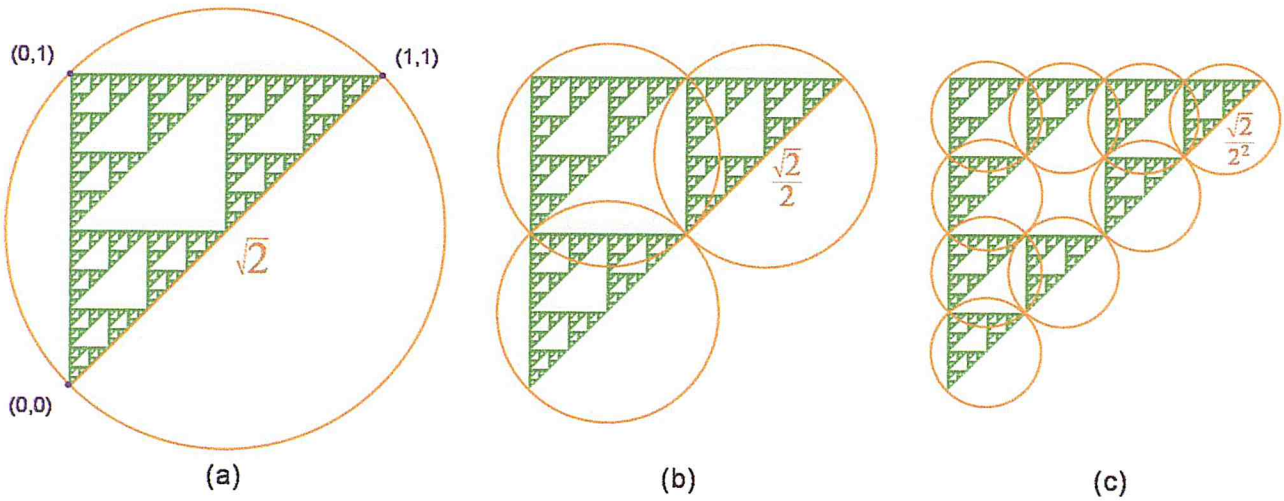


Fig. 1.8 – Recouvrements du triangle de Sierpinsky afin de calculer sa dimension de Hausdorff. Ici, par les images (a) n = 0, (b) n = 1 et (c) n = 2, on voit que le recouvrement optimal utilise des boules de diamètre $\epsilon = \sqrt{2}.\frac{1}{2^n}$ [1]

En effet, l'hypoténuse d'un triangle rectangle inscrit dans un cercle définit le diamètre de celui-ci. Notons que ϵ tend vers 0 lorsque n tend vers l'infini. Dans chaque cas, on utilise 3^n boules.

Ainsi, en supposant que ce recouvrement est celui pour lequel l'infime est atteint, on trouve la $\propto$-mesure de recouvrement :

$$m^{\alpha}(F) = \lim_{\varepsilon \to 0} inf\{\sum_{i=1}^{\infty}(\mathrm{diam}V_i)^{\propto} : F \subset \bigcup_{i=1}^{\infty} V_i \ , \ \mathrm{diam}V_i \leq \varepsilon\} = \lim_{n\to\infty} 3^n \left(\sqrt{2}.\frac{1}{2^n}\right)^{\propto} \quad (1.6)$$

ou, après réarrangement des termes, $m^{\alpha}(F) = \left(\sqrt{2}\right)^{\propto} \lim_{n\to\infty} \left(\frac{3}{2^{\propto}}\right)^n$. Or, cette limite tend vers 0 lorsque $\frac{3}{2^{\propto}} < 1$ et elle diverge à l'infini pour $\frac{3}{2^{\propto}} > 1$. Autrement dit, elle se situe entre 0 et 1 lorsque $\frac{3}{2^{\propto}} = 1$, c'est-a-dire pour $\propto = \frac{ln\,3}{ln2}$ Donc, la dimension de Hausdorff du triangle de Sierpinsky est $\frac{ln\,3}{ln2}$

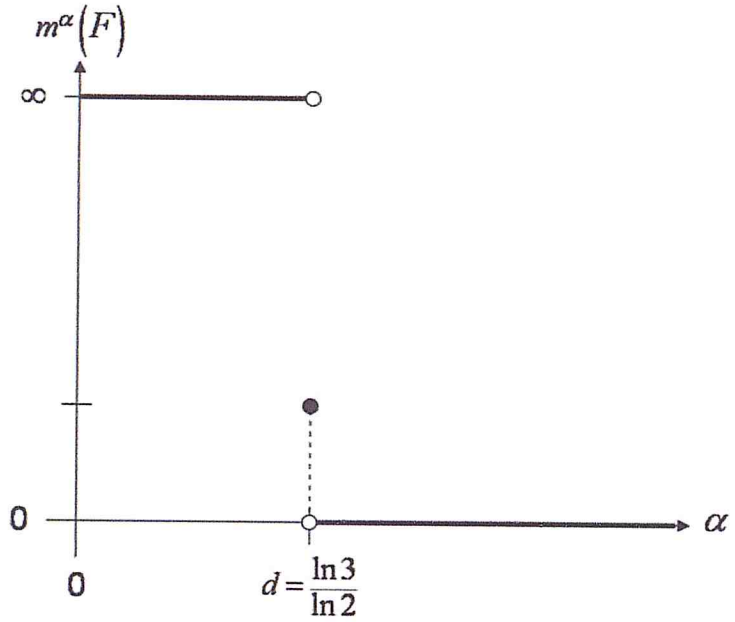


Fig. 1.9 – Graphique de la $\propto$-mesure de recouvrement d'un triangle de Sierpinsky en fonction de $\propto$. La dimension de Hausdorff est la valeur de $\propto$ correspondant au saut de ∞ à 0.

1.4.3 La dimension des boites (box-counting)

L'absence de méthodes générales pour le calcul d'un δ recouvrement minimum a conduit les mathématiciens à calculer la dimension fractale par méthode dite comptage de boites (Box-counting). Cette dernière considère un autre type de recouvrement de tel que la grille régulière. Elle est également appelée dimension de *Minkowski-Bouligand* [4].

Soit δ le pas de la grille servant a paver un objet F, nous parlons alors de δ pavage et nous notons $N_{\delta}(F)$ le nombre minimum de pavés de la grille interceptant l'objet F.

A l'instant de la mesure de *Hausdorff*, nous définissons la quantité :

$$b_{\delta}^{d}(F) = \inf\left\{\sum_{pavés\ intersectant\ F} \frac{\delta^d}{\delta - pavage\ de\ F}\right\} = N_{\delta}(F) \times \delta^d \quad (1.7)$$

En réduisant le pas δ de la grille, $b_\delta^d(F)$ prend alors une nouvelle valeur et lorsque δ tend vers 0, nous obtenons une approximation de la mesure de l'objet $b^d(F)$

$$b^d(F) = \lim_{\delta\to 0} b_\delta^d(F) = \lim_{\delta\to 0} N_\delta(F) \times \delta^d \quad (1.8)$$

Contrairement à la α −mesure de recouverement de *Hausdorff* (1.3.2), cette quantité n'est donc plus une mesure. Néanmoins comme elle suit une loi de puissance, elle permet de définir une dimension qui coïncide dans le nombreux cas avec la dimension de *Hausdorff* appelée dimension de boites de L'objet. notée $D_b(F)$. En effet quand δ devient suffisamment petit, $N_\delta(F)$ est devenu égale au nombre de points de l'ensemble, (tout en supposant que $\delta^d \approx 1$) d'où la limite suivante :

$$D_b(F) = \lim_{\delta\to 0} \frac{\log N_\delta(F)}{\log\left(\frac{1}{\delta}\right)} = -\lim_{\delta\to 0} \frac{\log N_\delta(F)}{\log(\delta)} \quad (1.9)$$

Cette dimension à la particularité d'être définie à partir du comptage du nombre pavés réguliers (δ) intercepte l'objet F et non a partir de la mesure de ces pavés.

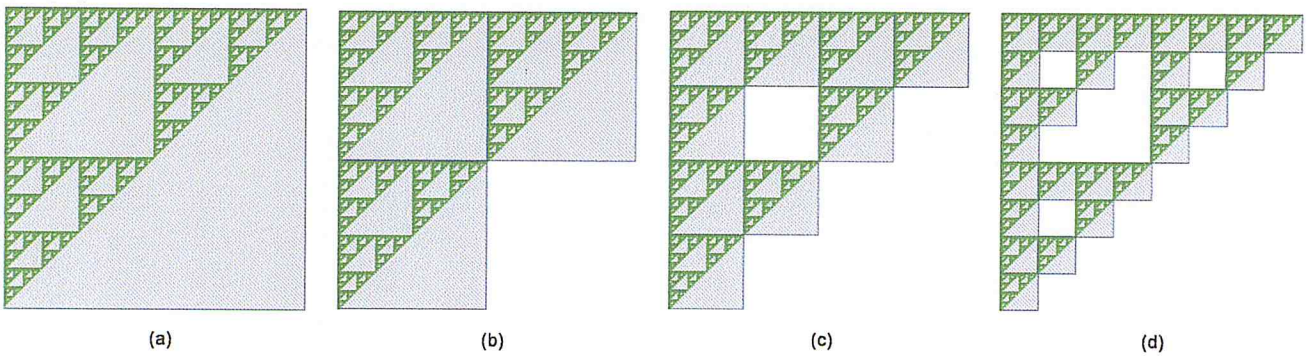


Fig. 1.10 – Considérons un triangle de Sierpinsky dont la mesure du côté est 1. (a) Il faut un carré de côté 1 pour le recouvrir (b) Il faut 3 carrés de mesure $\frac{1}{2}$ (c) Pour n = 2, $\left(\epsilon = \frac{1}{4}\right)$, on a $N_2 = 9 = 3^2$ (d) $N_3 = 27 = 3^3$, $\left(\epsilon = \frac{1}{16}\right)$. Ainsi, la dimension des boîtes du triangle est : $d = \lim_{n\to\infty}\left(\frac{\log N_n}{\log 2^n}\right) = \lim_{n\to\infty}\left(\frac{\log 3^n}{\log 2^n}\right) = \frac{\log 3}{\log 2} \approx 1.585$ [1]

1.5 La géométrie fractale

Inspiré par le slogan latin *Nomen* est *numen* (nommer c'est connaître)[1], Mandelbrot suggéra de donner un nom aux « monstres » mathématiques et d'étudier leurs propriétés communes. Ainsi, il créa en 1975 le mot « fractale » du latin « fractus » de verbe «frangere » qui signifie briser , présenter des irrégularités, fragmenter à toutes les échelles ou encore fractionner à l'infini. Cependant, ce n'est que sous la pression de la communauté scientifique qu'il propose une définition : « Un ensemble fractal (dans le plan ou dans l'espace) est un ensemble dont la dimension de *Hausdorff -Besicovitch* est strictement supérieure à sa dimension topologique» Il note qu'une fractale combine les caractéristiques suivantes [3]:

1. ses parties ont la même structure que le tout, a ceci prés qu'elles le sont a une échelle différente et peuvent être légèrement déformées ;
2. sa forme est extrêmement irrégulière ou fragmentée et le reste a toutes les échelles ;
3. elle contient des éléments discernables dans une large gamme d'échelles.

Mandelbrot à remarquer que les fractales sont présentes de façon universelle dans la nature. Ainsi, il a constaté que les nuages ne sont pas des sphères, les montagnes des cônes, ni les iles des cercles et leur description nécessite une géométrie adaptée.

1.5.1 L'ensemble de Julia

Cet ensemble est simplement obtenu par des itérations du polynôme $Z_{n+1} = Z_n^2 + c$ dans le plan des nombres complexes.

L'ensemble $J(c)$ rempli est l'ensemble des couples x, y du plan tels que la suite complexe (Z_n) a un module fini :

$$\begin{cases} Z_0 = x + iy \\ Z_{n+1} = Z_n^2 + c \end{cases} \quad (1.10)$$

Pour une valeur de c fixée, on évalue le comportement de la suite. Les points Z pour lesquels la suite est bornée forme l'ensemble de Julia rempli associé au point c (K_c).

L'ensemble de Julia $J(c)$ est la frontière de l'ensemble précédent alors que l'intérieur est appelé ensemble de *Fatou*[9]. Dans certains cas, les points sont rassemblés dans une seule surface connexe alors que pour d'autres valeurs de c, il est formé de points isolés et on dit que l'ensemble est non-connexe.

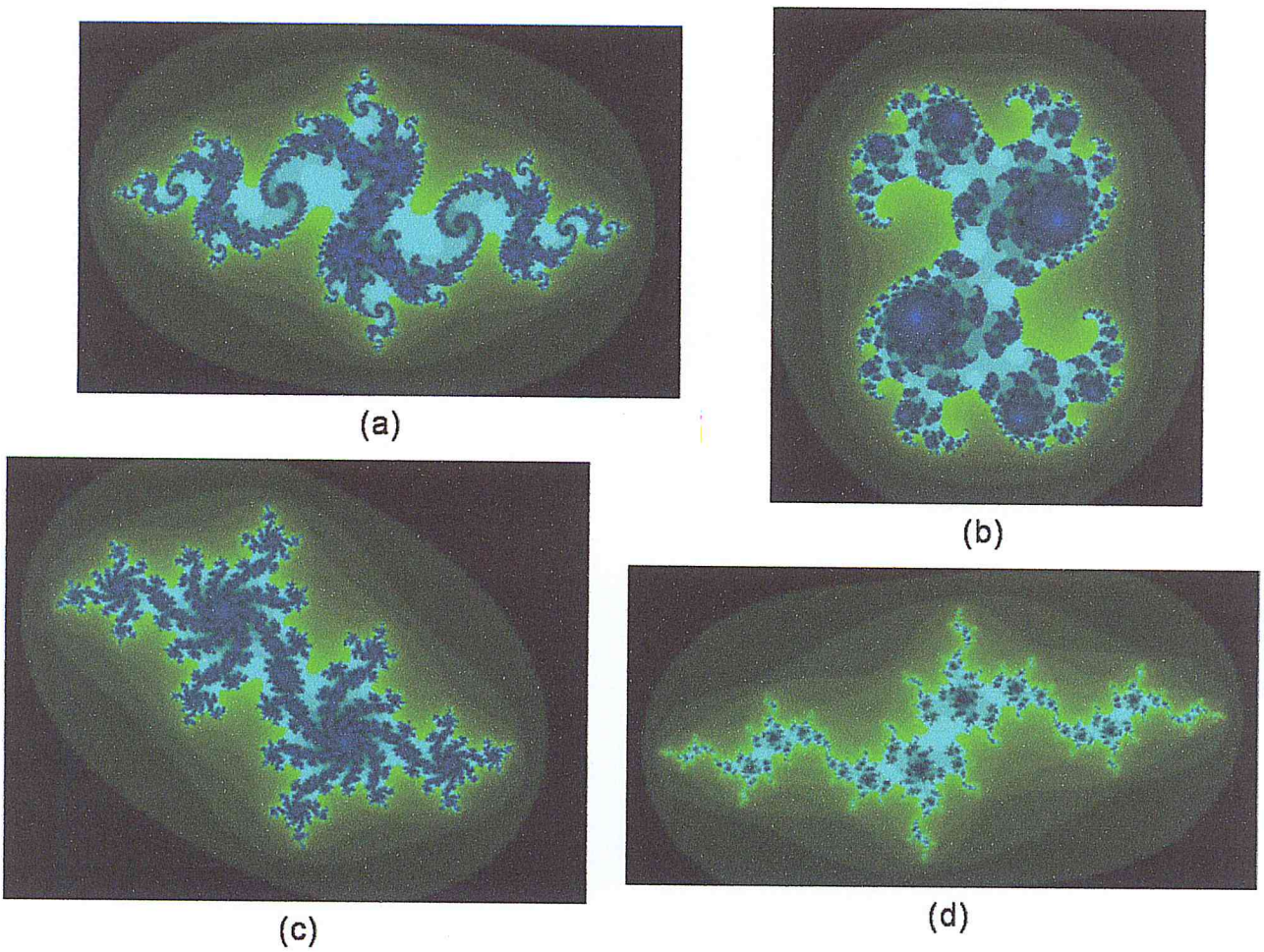


Fig. 1.11 – Ensembles de Julia associés au point c (a) c = −0, 8 + 0, 168i (b) c = 0, 328+0, 048i (c) c = −0, 344+0, 64i (d) c = −1, 096−0, 264i. Les couleurs illustrent la vitesse a laquelle la suite diverge.[4]

1.5.2 L'ensemble de *Mandelbrot*

Initialement, l'ensemble de *Mandelbrot* M a été défini comme l'ensemble des complexes $c \in \mathbb{C}$ pour lesquels l'ensemble de Julia $J(f_c)$ (f_c etant la fonction analytique d'expression $f_c(Z) = Z^2 + c$, $c \in C$) est connexe (Mandelbrot 1980)[6]

$$M\{c \in C \ \ tel\ que\ J(f_c)\ connexe\ \} \qquad (1.11)$$

Mandelbrot représenta graphiquement cette figure qui gagna rapidement en popularité. De fait, des résultats mathématiques montrent que l'on peut générer l'ensemble de *Mandelbrot* a l'aide du même polynôme $Z_{n+1} = Z_n^2 + c$. Cette fois, on fixe le point $Z_0 = 0$ et on évalue le comportement de la suite d'itérations pour chaque nombre c du plan. Si la suite est bornée, le point c fait partie de M. De plus, la position d'un paramètre nous donne quelques propriétés de son ensemble de Julia associé.

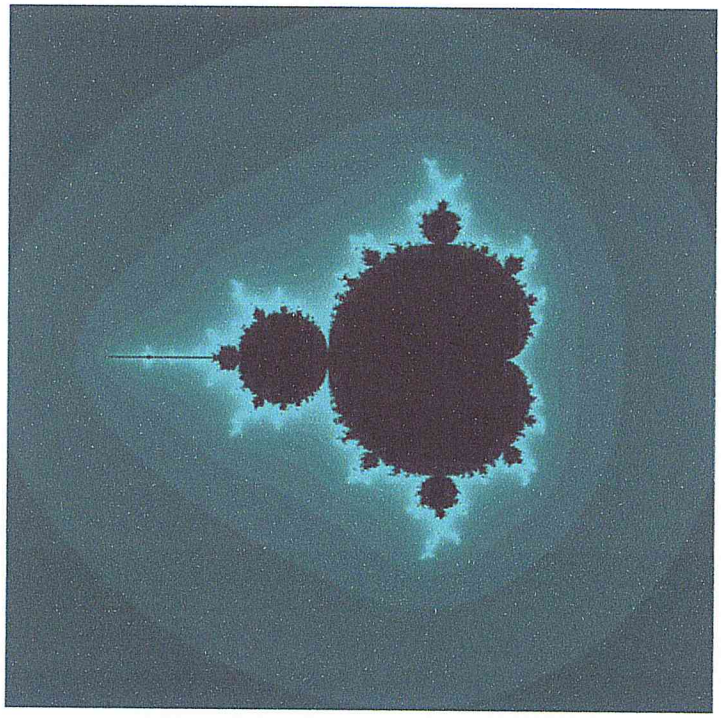


Fig. 1.12– La partie noire constitue l'ensemble de Mandelbrot. Autrement, plus la couleur est foncée plus la suite tend vers l'infini rapidement [4].

L'intérêt des mathématiciens pour cette figure les a menés a étudier l'ensemble de Mandelbrot dans les moindres détails. Il a été démontré que la dimension de Hausdorff de sa frontière est 2 et on a même réussi à en faire ressortir le nombre d'or ! Parmi les recherches intéressantes, mentionnons les généralisations de M pour les quaternions et les nombres bicomplexes.

1.6 Les L-Systems

Comme l'affirme Mandelbrot, les fractales constituent la géométrie de la nature. Il n'est donc pas surprenant que le biologiste *Aristid Lindenmayer* [1] ait eu recours à un processus fractal pour décrire la croissance des végétaux. En effet, en 1968, il publia un article dans lequel il formalise la description de la croissance d'une plante. C'est ce qu'on appelle les L-systems (« L » pour *Lindenmayer*). Ceux-ci se prêtent très bien à une implémentation informatique [1].

Un L-système est une grammaire formelle qui comprend :

1. Un alphabet V : l'ensemble des variables du L-système.V^* est l'ensemble des « mots » que l'on peut construire avec les symboles de , et V^+ l'ensemble des mots contenant au moins un symbole.
2. Un ensemble de valeur constantes S. Certains de ces symboles sont communs à tous les L-système (voir plus bas la tortue interprétation).
3. Un axiome de départ ω choisi parmi V^+ , c'est-à-dire l'état initial.

4. Un ensemble de règles, noté P , de reproduction des symboles de V.

Un L-système est noté $\{V, S, \omega, P\}$

Construction de la courbe de von Koch avec la tortue

Définition 1.6.1 : Soient V un alphabet et V^* l'ensemble de tous les mots formés a partir de cet alphabet. Un 0L-système est un triplet $\{V, \omega, P\}$ ou $\omega \in V^*$ est un mot non-vide appelé axiome et $P \subset V \times V^*$ un ensemble fini de règles de production.

Exemple 1.6.2 :

Pour construire la courbe de von Koch avec la tortue, on lui donne les instructions suivantes :

F : avance d'une distance donnée.

$+$: tourne a gauche (c'est- a -dire sens antihoraire) d'un angle de 60°.

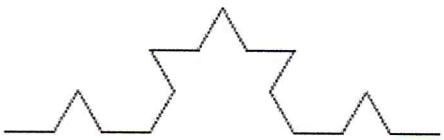
$-$: tourne a droite (c'est- a -dire sens horaire) d'un angle de 60°.

(i) On commence avec F (ce sera notre axiome).

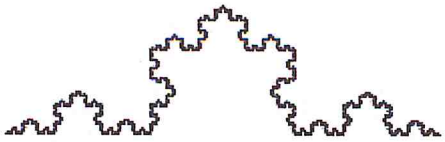
(ii) On remplace F par $F + F - -F + F$ (c'est notre règle de production) pour obtenir :

(iii) Dans la chaîne de caractères obtenue a l'étape précédente, on applique notre règle de production (c'est- a -dire on remplace F par $F + F - -F + F$), ce qui donnera

$F + F - -F + F + F + F - -F + F - -F + F - -F + F + F + F - -F + F$ et produira par la tortue



(iv) On continue ainsi en appliquant notre règle de production a la dernière chaîne de caractères obtenue, etc. Ainsi, à la sixième étape on aura :



qui donne une assez bonne approximation visuelle de la courbe de von Koch que l'on obtiendrait en itérant a l'infini.

Dans cette exemple

$V = \{F, +, -\}$

V^* = L'ensemble des chaînes de caractères formées des éléments de V

$\omega = F$

$P = (F, F + F - -F + F) \in V \times V^*$

Remarque : Dans ce qui suit, nous ne spécifierons pas explicitement l'alphabet. Celui-ci sera l'ensemble des caractères apparaissant dans l'axiome et les règles de production. Egalement, si une règle de production pour un caractère n'est pas énonce, nous supposerons

que c'est l'identité (par exemple $+ \rightarrow +$) qui s'applique a ce caractère.

1.7 Méthode des Iterated functions system (IFS)

Les fractales de les Système de fonctions itérées (IFS) sont créés sur la base de simples transformations polynomiales et affines sur le plan euclidien, comme mise à l'échelle, dislocation et la rotation des axes du plan [8]. Création d'une fractale IFS consiste en étapes suivantes:

1. Définir un ensemble de transformations du plan.
2. Dessinant un motif initial sur le plan (n'importe quel modèle).
3. Transformer le motif initial utilisant les transformations définies dans la première étape.
4. Transformer la nouvelle image (combinaison de motifs initiaux et transformées) en utilisant le même ensemble de transformations.
5. Répétant la quatrième étape autant de fois que possible (en théorie, cette procédure peut être répétée un nombre infini de fois).

Tout ensemble de cartes linéaires (transformations affines) et un ensemble de probabilités associée détermine un système de fonctions itérées (IFS). Chaque IFS a un "attracteur" unique qui est typiquement un ensemble fractal (objet). Spécification de seulement quelques cartes peut produire des objets très complexes.

Conception d'objets fractals est faite relativement simple et intuitive par la découverte d'une propriété mathématique importante concernant les ensembles fractals à l'IFS. La méthode prévoit également la possibilité de résoudre le problème inverse, compte tenu de la géométrie d'un objet, déterminer une IFS qui (environ) de générer que la géométrie.

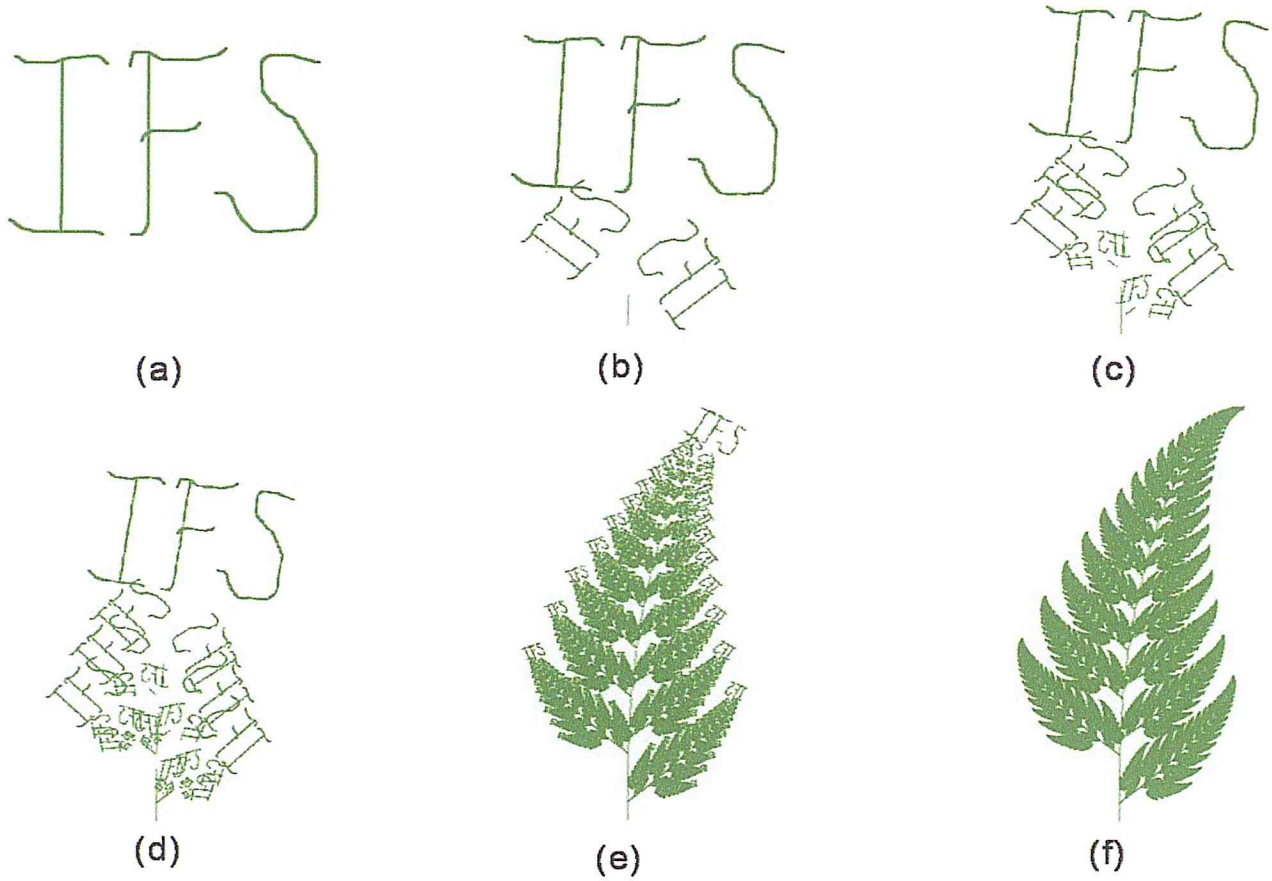


Fig. 1.13 – Utilisation de la méthode des IFS pour construire la fougère de Barnsley. (a) Image initiale (b) Première itération (c) 2 itérations (d) 3 itérations (e) 10 itérations (f) 25 itérations[1]

1.8 Conclusion

Bien cachées au cœur de notre environnement depuis toujours, les fractales ont mis beaucoup de temps à gagner notre attention. Cependant, depuis la contribution de Mandelbrot, elles sont devenues une véritable célébrité : les mathématiciens les explorent en détails et on leur trouve de nombreuses applications. En effet, puisqu'elles décrivent si bien la nature et que l'homme cherche constamment à comprendre et reproduire son environnement, les fractales jouent un rôle important en biologie, en infographie, en finance, en physique et plusieurs autres domaines. Relativement jeune, la géométrie fractale nous réserve-t-elle encore bien des surprises ?

Chapitre 2

Système de Fonctions Itérées

2-1 Introduction

La théorie des systèmes de fonctions itérées ou **IFS** (d'après le nom anglais *Iterated Function System*) est une théorie mathématique développée par *John Hutchinson* en 1981, utilisée dans le cadre de la géométrie fractale (depuis les travaux de *Michael Barnsley* en 1988 et son livre *Fractals Everywhere*). Cette théorie est entièrement fondée sur les invariances par changement d'échelle [3].

Un IFS peut être la représentation fonctionnelle d'une fractale. Cela donne une théorie parfaitement définie mathématiquement qui permet de nombreuses études sur les fractales (continuité, dérivabilité, approximation…). Pour arriver à démontrer sa validité, il est nécessaire de comprendre un certain nombre de notions de topologie telles que les espaces métriques complets, les ensembles compacts et les contractions. Les concepts sous-jacents à la théorie associée aux IFS sont détaillés dans ce chapitre.

2.2 Théorie pour la construction des IFS

La construction de fractales par la méthode des Iterated Functions System (IFS) repose essentiellement sur un seul théorème qui se veut un cas particulier du théorème des points fixes de *Banach*.

L'idée première est de construire un espace où les fractales existent et sur lequel on pourra définir les IFS. Nous devons donc présenter la notion des espaces métriques complets puisque c'est ce type d'espace qui est requis par les hypothèses du théorème des points fixes.

2-2-1 Notions d'analyse

Définition 2.1.1 Un espace X est un ensemble. Les points de l'espace sont les éléments de l'ensemble.

Définition 2.1.2 Une métrique est une fonction réelle $d : X \times X \to R$ mesurant la distance entre deux points $x, y \in X$ et ayant les propriétés suivantes :

1. $0 \leq d(x,y) < \infty$, $\forall\ x,y \in X$.
2. $d(x,y) = d(y,x), \forall\ x,y \in X$.
3. $d(x,y) = 0, x = y$, $\forall\ x,y \in X$.
4. $d(x,y) \leq d(x,z) + d(z,y)$, $\forall\ x,y,z \in X$.

Définition 2.1.3 Un espace métrique (X,d) est un espace X muni d'une métrique d.

Définition 2.1.4 la métrique euclidienne : Soit $d(x,y) = |x - y|$ une fonction réelle sur $X = R$. On a que

1. $0 \leq |x - y| < \infty$, $\forall x,y \in R$.
2. $d(x,y) = |x - y| = |-1(y - x)| = |-1||y - x| = 1 \cdot |y - x| = d(y,x)$. (2.1)
3. $d(x,y) = 0, |x - y| = 0, x = y$.
4. $d(x,y) = |x - y| \leq |x - z| + |z - y|$ *par l'inégalité du triangle*
   $= d(x,z) + d(z,y)$.

Donc, $d(x,y) = |x - y|$ est une métrique sur R et (R,d) est un espace métrique. On appelle d la métrique euclidienne.

Définition 2.1.5 Soit $S \subset X$, un sous-ensemble de l'espace métrique (X,d). Un point $x \in X$ est appelé point d'accumulation de S s'il existe une suite $\{x_n\}_{n\geq 1}$ de points $x_n \in S \setminus \{x\}$ telle que $\lim_{n\to\infty} x_n = x$.

Notation : L'ensemble des points d'accumulation de S est appelé ensemble dérivé de S et est noté S' ou $Acc(S)$.

Définition 2.1.6 Soit $S \subset X$, un sous-ensemble de l'espace métrique (X,d). On dit que S est fermé s'il contient tous ses points d'accumulation, c'est-a-dire $Acc(S) \subset S$.

Définition 2.1.7 Soit $S \subset X$, un sous-ensemble de l'espace métrique (X,d). S est dit borné s'il existe un point $a \in X$ et un nombre $R > 0$ tel que $d(a,x) < R, \forall x \in S$.

Définition 2.1.8 Une suite de points $\{x_n\}_{n\geq 1}$ d'un espace métrique (X,d) est appelée suite de Cauchy si

$\forall \epsilon > 0, \exists\ N(\epsilon)$ *tel que* $d(x_n, x_m) < \epsilon$ *dés que* $m,n > N(\epsilon)$. (2.2)

Théorème 2.1.1 Dans un espace métrique, toute suite convergente est une suite de Cauchy.

Preuve. Soit une suite $\{x_n\}_{n\geq 1}$ d'un espace métrique (X, d) convergeant vers $x \in X$ Alors

$$\forall \epsilon > 0, \exists N(\epsilon) tel\ que\ d(x_n, x) < \epsilon\ dés\ que\ n > N(\epsilon).$$

Prenons $m, n > N\left(\frac{\epsilon}{2}\right)$,

$$alors\ d(x_m, x_n) \leq d(x_m, x) + d(x, x_n) < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \quad (2.3)$$

et ainsi la suite est de Cauchy.

Définition 2.1.9 Un espace métrique (X, d) est dit complet si toutes les suites de Cauchy $\{x_n\}_{n\geq 1}$ dans X ont une limite $x \in X$.

Théorème 2.1.2 Montrons que (R, euclidienne) est complet

Preuve : Il suffit de montrer que si une suite est de Cauchy dans R, alors elle est convergente dans R. La preuve de ceci requiert deux lemmes.

Lemme 2.1.1 Dans (R, euclidienne), toute suite bornée possède une sous-suite convergente.

Preuve : Soit $\{x_n\}_{n\geq 1}$, une suite bornée d'éléments de R. Si le champ des éléments de la suite est fini, alors un de ces termes, disons x_i apparaît une infinité de fois. En prenant la sous-suite constante $\{x_i\}$, on a le résultat. Autrement, le champ de la suite possède une infinité d'éléments. A l'aide du théorème de Bolzano-Weierstrass, il est possible de construire une sous-suite convergente dans R de $\{x_n\}_{n\geq 1}$.

Lemme 2.1.2 Dans (R, euclidienne), toute suite de Cauchy est bornée.

Preuve. Soit $\{x_n\}_{n\leq 1}$ une suite de Cauchy d'éléments de R. Alors,

$$\exists N(1) tel\ que\ |x_n - x_m| < 1\ dès\ que\ m, n > N(1). \quad (2.4)$$

Pour $m > n\ et\ n = N + 1$, on a $|x_{N+1} - x_m| < 1$ et par conséquent, $|x_m| < |x_{N+1}| + 1$

pour tout $m > N + 1$. Prenons $M = max\{|x_1|, |x_2|, \ldots, |x_{N+1}|, |x_{N+1}| + 1\}$. On trouve que $\{x_n\}_{n\geq 1} < M$ pour tout n.

Considérons une suite de Cauchy $\{x_n\}_{n\geq 1}$ dans R. Les deux lemmes précédents assurent qu'il existe une sous-suite $\{x_{n_i}\}_{i\geq 1}$ de $\{x_n\}_{n\geq 1}$convergeant vers un élément x dans R.

Donc,

$$\forall \epsilon > 0, \exists\ N_1(\epsilon)\ tel\ que\ |x_{n_i} - x| > \epsilon\ dès\ que\ n_i > N_1(\epsilon). \quad (2.5)$$

De plus, la suite $\{x_n\}_{n\geq 1}$ est de Cauchy donc

$$\exists\ N_2(\epsilon) tel\ que\ |x_n - x_m| < \epsilon\ pour\ tout\ m, n > N_2(\epsilon). \quad (2.6)$$

Prenons $N(\epsilon) = max\{N_1(\frac{\epsilon}{2}), N_2(\frac{\epsilon}{2})\}$ alors si $l \in \{n_i\} > N(\epsilon)$, on a que $|x_l - x| < \frac{\epsilon}{2}$ et $|x_n - x_l| < \frac{\epsilon}{2}$ pour tout $n > N(\epsilon)$. Par l'inégalité du triangle, il vient que $|x_n - x| < \epsilon$ autrement dit la suite est convergente dans R.

Définition 2.1.10 Soit $S \in X$, un sous-ensemble de l'espace métrique (X, d). S est dit compact si toute suite infinie de S contient une sous-suite convergente dans S.

Théorème 2.1.3 Soit $S \in X$, un sous-ensemble de l'espace métrique (X, d). Si S est compact, alors il est fermé.

Preuve : Supposons que S est compact mais qu'il n'est pas fermé, c'est-a-dire qu'il existe un $x_1 \in X \backslash S$ et une suite $\{x_n\}_{n\geq 1}$ d'éléments de S qui converge vers x_1. Ainsi,

$$\forall \epsilon > 0, \exists N_1\left(\frac{\epsilon}{2}\right) \text{ tel que } d(x_n, x_1) < \epsilon \text{ dès que } n > N_1. \quad (2.7)$$

Puisque S est compact, on peut trouver une sous-suite $\{x_{n_k}\}_{n\geq 1}$ qui converge vers $x_2 \in S$. Ainsi,

$$\forall \epsilon > 0, \exists N_2\left(\frac{\epsilon}{2}\right) \text{ tel que } d(x_{n_k}, x_2) < \epsilon \text{ dès que } k > N_2. \quad (2.8)$$

Or, lorsque $k > max\{N_1, N_2\}$, on trouve ,

$$d(x_1, x_2) < d(x_1, x_{n_k}) + d(x_{n_k}, x_2) < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \quad (2.9)$$

Mais, x_1 et différent de x_2 et on a la contradiction recherchée.

Définition 2.1.11 Soit $S \subset X$, un sous-ensemble de l'espace métrique (X, d). S est dit totalement borné si, $\forall \epsilon > 0$, il existe un ensemble fini de points $\exists\{y_1, y_2, \ldots, y_n\} \subset S$ tel que

$$\forall x \in S, on\ a\ d(x, y_i) < \epsilon \quad (2.10)$$

pour un certain $y_i \in \{y_1, y_2, \ldots, y_n\}$.

Remarque : Cet ensemble de points est appelé un $\epsilon - net$.

Lemme 2.1.3 Soit $S \subset X$, un sous-ensemble de l'espace métrique (X, d). Si S est totalement borné, alors $A \subset S$ sera aussi totalement borné. Autrement dit, tout sous ensemble d'un ensemble totalement borné est lui aussi totalement borné.

Preuve : La définition S est totalement borné signifie que $\forall \epsilon > 0$, il existe un ensemble de points fini $\{y_1, y_2, \ldots, y_n\} \subset S$ tel que $\forall x \in S, on\ a\ d(x, y_i) < \epsilon$ pour un certain

$y_i \in \{y_1, y_2, \ldots, y_n\}$.. Autrement dit

$$S \subset \bigcup_{k=1}^{n} B(y_k, \epsilon) \ ou\ B(y_k, \epsilon) = \{x \in S : d(x, y_k) < \epsilon\}. \quad (2.11)$$

Montrons que $A \subset S$ est totalement borné, c'est-a-dire que :
$\forall \epsilon > 0, \exists \{z_1, z_2, \ldots, z_l\} \subset A,\ l \leq n \text{ tel que } A \subset \bigcup_{k=1}^{l} B(z_k, \epsilon)$
Ou $B(z_k, \epsilon) = \{x \in S : d(x, z_k) < \epsilon\}$.
Puisque $A \subset S$, on a que

$$A = A \cap S$$

$$\subset A \cap \bigcup_{k=1}^{n} B(y_k, \epsilon)$$

$$= \bigcup_{k=1}^{n} [A \cap B(y_k, \epsilon)]. \qquad (2.12)$$

Soit $\epsilon_0 > 0$, par (1), $\exists \{y_1, y_2, \ldots, y_n\} \subset S$ tel que $S \subset \bigcup_{k=1}^{n} B(y_k, \frac{\epsilon_0}{2})$. Donc, par (2.12), $A \subset \bigcup_{k=1}^{n} A \cap B(y_k, \frac{\epsilon_0}{2})$. `A partir de cette expression, construisons un $\epsilon - net$ pour A.
Pour chaque y_k, deux situations peuvent se produire : $y_k \in A$ ou : $y_k \notin A$.
– Si $y_k \in A$, on pose $z_k = y_k$ et on a $A \cap B(y_k, \frac{\epsilon_0}{2}) \subset B(z_k, \epsilon_0)$.
– Si $y_k \notin A$, soit $A \cap B(y_k, \frac{\epsilon_0}{2}) = \emptyset$ et ce y_k est éliminé (il n'a aucun rôle a jouer dans le ϵ_net). Sinon, $A \cap B(y_k, \frac{\epsilon_0}{2}) \neq \emptyset$ et alors il existe un élément z_k de A tel que $z_k \in B(y_k, \frac{\epsilon_0}{2})$. De plus, $A \cap B(y_k, \frac{\epsilon_0}{2}) \subset B(z_k, \epsilon_0)$. En effet, prenons un point $x_0 \in A \cap B(y_k, \frac{\epsilon_0}{2})$. Alors, $d(x_0, y_k) < \frac{\epsilon_0}{2}$ et $(y_k, z_k) < \frac{\epsilon_0}{2}$. Donc,

$$d(x_0, z_k) < d(x_0, y_k) + d(y_k, z_k) < \frac{\epsilon_0}{2} + \frac{\epsilon_0}{2} = \epsilon_0$$

Ainsi, $x_0 \in B(z_k, \epsilon_0)$ d'où $A \cap B(y_k, \frac{\epsilon_0}{2}) \subset B(z_k, \epsilon_0)$.
Donc, $\forall \epsilon_0 > 0$, $\exists \{z_1, z_2, \ldots, z_l\} \subset A$, $l \leq n$ tel que $A \subset \bigcup_{k=1}^{n} A \cap B(y_k, \frac{\epsilon_0}{2}) \subset \bigcup_{k=1}^{n} B(z_k, \epsilon_0)$. (2.13)

Théorème 2.1.4 Soit $S \subset X$ ou (X, d) est un espace métrique complet. S est compact si et seulement si S est fermé et totalement borné.

Preuve :

$\Longleftarrow$) Supposons que S soit fermé et totalement borné. Soit une suite $\{x_n\}_{n \geq 1}$ infinie de S. Montrons qu'elle contient une sous-suite convergente dans S.

Puisque S est totalement borné, il existe un nombre fini de boules ouvertes de rayon 1 dont l'union contient l'ensemble S. Puisque $\{x_n\}_{n=1}$ est infinie, au moins une de ces boules, disons $B1$, contient un nombre infini d'éléments de S. Prenons un point x_{N_1} tel que $x_{N_1} \in$ B1.

Considérons $S \cap B1$, en utilisant le lemme 2.1.3, on peut montrer que $S \cap B1$ est aussi totalement bornée. Donc, il existe un nombre fini de boules ouvertes de rayon $\frac{1}{2}$ dont l'union contient les éléments de $S \cap B1$. Une de ces boules, disons B2, contient un nombre infini d'éléments x_n. Prenons N_2 tel que

$$x_{N_2} \in B2 \; et \; N_2 > N_1.$$

En refaire le procédé, on trouve :

$$B1 \supset B2 \supset \ldots \supset Bn \supset \ldots$$

ou Bn a un rayon de $\frac{1}{2^{n-1}}$. On a ainsi une suite d'entiers $\{N_n\}_{n\geq 1}$ telle que $x_{N_n} \in Bn$. Il s'ensuit que la sous-suite $\{x_{N_n}\}_{n\geq 1}$ est de Cauchy et comme S est fermé, cette sous-suite est convergente dans S.

⇒) Supposons que S est compact alors il est nécessairement fermé par le théorème 2.1.3. Supposons maintenant qu'il n'est pas totalement borné, c'est-à-dire que pour $\epsilon > 0$, il n'existe pas de $\epsilon - net$ pour S. Cherchons une contradiction. Comme il n'y a pas de $\epsilon - net$ pour S, il existe une suite infinie de points distincts $\{x_{N_n}\}_{n\geq 1} \subset$ S avec $d(x_i, x_j) > \epsilon, \forall i \neq j$.

Mais comme S est compact, cette suite doit posséder une sous-suite convergente, disons $\{x_{N_i}\}_{n\geq 1}$. Ainsi, cette sous-suite est de Cauchy et on peut trouver une paire d'entiers N_1 et N_2 avec $N_1 \neq N_2$ tel que $d(x_{N_1}, x_{N_2}) < \epsilon$. Mais, $d(x_{N_1}, x_{N_2}) \geq \epsilon$ d'ou la contradiction recherchée.

Définition 2.1.12 Soit $S \subset X$, un sous-ensemble de l'espace métrique (X, d). On dit que $x \in X$ est un point intérieur de S si $\forall \epsilon > 0 \; tel \; que \; B(x, \epsilon) \subset S$.

Remarque : L'ensemble des points intérieurs de S est appelé l'intérieur de S et se note S^0 ou $int(S)$.

2.3 Théorème du point fixe dans R

Avant d'énoncer le théorème du point fixe dans R, on énonce quelques définitions et résultats fondamentaux en analyse réelle.

Définition 2.2.1. On dit que g : $R \rightarrow R$ est une contraction de facteur k s'il existe k tel que

$$0 \leq k < 1 \; et \; |g(x) - g(y)| \leq k|x - y| \; \forall x \; et \; \forall y \in R \qquad (2.14)$$

Puisque $k < 1$, la distance entre $g(P)$ et $g(Q)$ est plus petite que la distance entre P et Q.

Notation : Désignons par $g^n = \underbrace{gogog \dots\dots\dots..og}_{n \text{ fois}}$ n fois la fonction obtenue en composant g avec elle-même.

Théorème 2.2.1 (***Point fixe***). Soit $g : R \to R$ une contraction de facteur k $(0 \leq k < 1)$.

Alors :

(i) Il existe un unique point fixe $\bar{x}$ (c'est-a-dire un unique $\bar{x}$ tel que $g(\bar{x}) = \bar{x}$).

(ii) Quel que soit $x_0 \in R$, la suite $\{x_n\}_{n\in N}$ définie par $x_n = g_n(x_0)$ converge vers $\bar{x}$ et, en outre, on a

$$|x_n - \bar{x}| \leq \frac{k^n}{1-k} |x_1 - x_0|. \qquad (2.15)$$

Démonstration

(a) Montrons que si un point fixe existe, il est nécessairement unique. Supposons qu'il existe deux points fixes distincts $\bar{x}$ et $\bar{y}$, c'est-`a-dire $g(\bar{x}) = \bar{x}$, $g(\bar{y}) = \bar{y}$ et $\bar{x} \neq \bar{y}$. Puisque g est une contraction de facteur k, on a

$$|g(\bar{x}) - g(\bar{y})| \leq k|\bar{x} - \bar{y}|, mais\ g(\bar{x}) = \bar{x}\ et\ g(\bar{y}) = \bar{y}$$

D'ou on aurait $|\bar{x} - \bar{y}| < k|\bar{x} - \bar{y}|$, ce qui est impossible si $\bar{x} \neq \bar{y}$. car $0 \leq k < 1$.

(b) Soit $x_0 \in R$, puisque R est complet pour montrer la convergence de la suite $\{x_n\}_{n\in N}$, ou $x_n = g_n(x_0)$, il suffit de montrer que c'est une suite de Cauchy. Montrons d'abord les deux inégalités : soient n et $p \in N$, on a

$$|x_{n+1} - x_n| \leq k_n|x_1 - x_0| \qquad (2.16)$$

$$(2)\ |x_{n+p} - x_n| \_ \frac{k_n}{1-k}|x_1 - x_0| \qquad (2.17)$$

Preuve de (1) Puisque $|g(x) - g(y)| \leq k|x - y|, \forall x, \forall y \in R$ et ayant $x_m = g(x_{m-1}), \forall m \in N$, on obtient

$$\begin{aligned} |x_{n+1} - x_n| &= |g(x_n) - g(x_{n-1})| \\ &\leq k\,|x_n - x_{n-1}| = k\,|g(x_{n-1}) - g(x_{n-2})| \\ &\leq k_2\,|x_{n-1} - x_{n-2}|, \\ &\dots\dots\dots\dots\dots \\ &\leq k_n\,|x_1 - x_0| \end{aligned}$$

Preuve de (2) En utilisant l'inégalité triangulaire à répétition on obtient
$|x_{n+p} - x_n| \leq |x_{n+p} - x_{n+p-1}| + |x_{n+p-1} - x_{n+p-2}| + \ldots \cdot + |x_{n+2} - x_{n+1}| + |x_{n+1} - x_n|$
et d'après (1), on déduit

$$|x_{n+p} - x_n| \leq k^{n+p-1}|x_1 - x_0| + k^{n+p-2}|x_1 - x_0| + \cdots + k^{n+1}|x_1 - x_0| + k^n|x_1 - x_0|$$

$$\leq k^n (1 + k + \cdots + k^{p-1}) |x_1 - x_0|$$

Mais, ayant $0 \leq k < 1$, on a $\frac{1}{1-k} = 1 + k + k^2 + \cdots + k^{p-1} + k^p + \ldots$ d'où $1 + k^2 + \cdots + k^{p-1} < \frac{1}{1-k}$
. On obtient donc

$$|x_{n+p} - x_n| \leq k^n (1 + k + \cdots + k^{p-1}) |x_1 - x_0| \leq \frac{k^n}{1-k} |x_1 - x_0|$$

c'est-à-dire l'inégalité (2).

Si m et n sont des entiers positifs, on déduit

$$|x_m - x_n| \leq \frac{k^{\min(m,n)}}{1-k} |x_1 - x_0| d'ou \lim_{n,m\to\infty} |x_m - x_n| = 0 \qquad (2.18)$$

et donc la suite $\{x_n\}_{n\in N}$ est une suite de Cauchy. Puisque R est complet, elle est convergente. Désignons par $\bar{x}$ la limite de la suite.
Montrons que $\bar{x}$ est un point fixe de g. Puisque g est une contraction, g est continue. Ainsi, on a

$$\lim_{n\to\infty} g(x_n) = g\left(\lim_{n\to\infty} x_n\right) = g(\bar{x})$$

Mais

$$x_{n+1} = g(x_n), d'ou \lim_{n\to\infty} g(x_n) = \lim_{n\to\infty} x_{n+1} = \bar{x}$$

et donc $g(\bar{x}) = \bar{x}$. En outre, d'après (a), ce point fixe est unique. Il ne reste donc qu'a démontrer l'inégalité (*).
Pour tout n et tout $p \in N$, on a d'après l'inégalité

$$|x_n - x_{n+p}| = |x_{n+p} - x_n| \leq \frac{k^n}{1-k} |x_1 - x_0| \qquad (2.17)$$

d'où

$$\lim_{n\to\infty} |x_n - x_{n+p}| \leq \frac{k^n}{1-k} |x_1 - x_0|$$

Mais $\lim_{p\to\infty} |x_n - x_{n+p}| = |x_n - \bar{x}|$ car $\lim_{p\to\infty} x_{n+p} = \bar{x}$. On obtient donc

$$|x_n - \bar{x}| \leq \frac{k^n}{1-k} |x_1 - x_0|$$

Remarque. L'inégalité (*) donne une mesure de la vitesse convergence. Etant donné un $\varepsilon > 0$, elle permet de déterminer à partir de quel rang n on aura $|x_n - \bar{x}| \leq \varepsilon$. Il suffit de prendre n tel que $\frac{k^n}{1-k} |x_1 - x_0| \leq \varepsilon$ (ce qui est possible, car $0 \leq k < 1$).

Remarque. Dans R^n, tout comme dans R, toute suite de Cauchy converge. Ainsi, dans la preuve précédente, il suffirait que de remplacer $|x - y|$ par $d(x,y)$ pour avoir une version du théorème du point fixe dans R^n. Par ailleurs, il existe d'autres ensembles à l'intérieur des quels toute suite de Cauchy converge. Dans la prochaine section, on énonce un résultat analogue au théorème du point fixe pour ces ensembles particuliers.

2.3.1 Principe de contraction dans les espaces métriques complets

On a déjà définie c'est un distance a la suite on va donne des différentes exemples sur les distances

a) R avec la distance usuelle d définie par $d(x,y) = |x - y|$ est un espace métrique.

b) Dans R^2, plusieurs distances peuvent être définies.

i) R^2 avec la distance euclidienne d_2, qui est définie par

$$d_2(S,T) = \sqrt{(s_1 - t_1)^2 + (s_2 - t_2)^2} \quad ou \quad S = (s_1, s_2), T = (t_1, t_2) \qquad (2.19)$$

est un espace métrique.

ii) R^2 avec la distance donnée par $d_\infty(S,T) = max\{|s_1 - t_1|, |s_2 - t_2|\}$ est un espace métrique.

iii) R^2 avec la distance Manhattan d_1, qui est définie par $d_1(S,T) = |s_1 - t_1| + |s_2 - t_2|$, est un espace métrique.

iv) Plus généralement, pour $1 < p < 1$, on peut définir la distance d_p par

$$d_p(S,T) = (|s_1 - t_1|^p + |s_2 - t_2|^p)^{\frac{1}{p}} \qquad (2.20)$$

R^2 avec la distance d_p est un espace métrique.

c) Toutes ces distances peuvent être définies de façon analogue dans R^n.

i) R^n avec la distance euclidienne d_2, qui est définie par

$$d_2(S,T) = \sqrt{(s_1 - t_1)^2 + (s_2 - t_2)^2 + \cdots + (s_n - t_n)^2}$$

où $S = (s_1, \ldots, s_n), T = (t_1, \ldots, t_n)$, est un espace métrique.

ii) R^n avec la distance d_p, qui est définie par

$$d_p(S,T) = \left(\sum_{i=1}^{n} |s_i - t_i|^p \right)^{\frac{1}{p}}$$

où $S = (s_1, \ldots, s_n), T = (t_1, \ldots, t_n)$, et $1 < p < 1$, est un espace métrique.

d) Soit $\sum$ l'ensemble des suites formées de 0 et de 1, c'est-à-dire

$$\sum = \{s = (s_0, s_1, s_2, \ldots, s_n, \ldots) | s_i = 0 \; ou \; 1\}.$$

Si $s = (s_0, s_1, \ldots, s_n, \ldots)$ et $t = (t_0, t_1, \ldots, t_n, \ldots) \in \sum$, on définit

$$d(s,t) = \sum_{i=0}^{\infty} \frac{|s_i - t_i|}{2^i}$$

Définition 2.2.2 Soient X un espace métrique avec une distance d et $g : X \to X$. On dit que g est une contraction de facteur k s'il existe k tel que $0 \leq k < 1$ et $d\big(g(x), g(y)\big) \leq k\, d(x,y) \;\; \forall x \in X \; et \; \forall y \in X.$ (2.21)

Dans les espaces métriques complets, le principe de contraction est vrai, c'est-à-dire on a le résultat suivant qui est l'analogue pour les espaces métriques du théorème du point fixe dans R (théorème 1.2.1).

Théorème 2.2.2 (Principe de contraction). Soit X un espace métrique complet avec la distance d et soit $g : X \to X$ une contraction de facteur $(0 \leq k < 1)$. Alors :

(i) il existe un unique point fixe $\bar{x}$ de g ;

(ii) quel que soit $x_0 \in X$, la suite $\{x_n\}_{n \in N}$ définie par $x_n = g^n(x_0)$ converge vers $\bar{x}$ et $d(x_n, \bar{x}) \leq \frac{k_n}{1-k} d(x_1, x_0)$. (2.22)

2.4 Systèmes itératifs de fonctions

2.4.1 Espace métrique idoine

Nous intéressons ici aux fractals dans R^2. On procéderait de la même façon pour les fractals dans R^n.

Définition 2.3.1 Un ensemble $K \subset R^2$ est compact s'il est fermé et borné.

Nous allons considérer l'ensemble suivant :

$$\mathcal{K} = \{K | K \; est \; un \; sous - ensemble \; compact \; de \; R^2\}, \qquad (2.23)$$

c'est-à-dire les éléments de $\mathcal{K}$ sont les sous-ensembles compacts de R^2. Nous définirons une distance sur $\mathcal{K}$: la distance de Hausdorff.

2.4.2 Distance de Hausdorff

Au début du siècle, le mathématicien allemand *Felix Hausdorff* [3]a introduit une distance sur l'ensemble $\mathcal{K}$, c'est-à-dire une notion de distance entre deux sous-ensembles compacts A et B de R^2.

On procède comme suit. Si $K \subset R^2$ est compact et $\varepsilon > 0$, on définit le "ε-voisinage K" de K Par $K_\varepsilon = \{P \in R^2 \, | d(P,Q) < \varepsilon \; pour \; un \; certain \; Q \in K^2\}$,

où d est la distance sur R^2 (e.g . $d_1, d_2, d_\infty, dp, \ldots$)

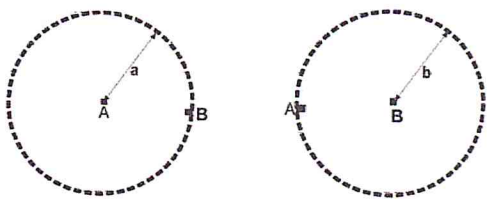
Soient A et $B \in \mathcal{K}$ (c'est-à-dire deux sous-ensembles compacts de R^2). La distance de Hausdorff $h(A,B)$ entre A et B est définie par :

$$h(A,B) = inf\{\varepsilon > 0 | A \subset B_\varepsilon \; et \; B \subset A_\varepsilon\}. \qquad (2.24)$$

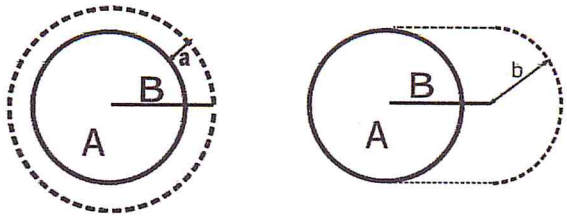
Pour obtenir $h(A,B)$, on calcule $a = inf\{\varepsilon > 0 | B \subset A_\varepsilon\}$ et $b = inf\{\varepsilon > 0 | A \subset B_\varepsilon\}$. Alors , on a $h(A,B) = max\{a,b\}$.

Les figures suivantes illustrent ces quantités a et b lorsque

(a) A et B sont des points (dans ce cas $h(A,B)$ est la distance usuelle entre 2 points)

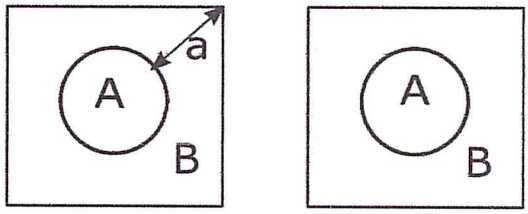


(b)A un disque et B un segment de

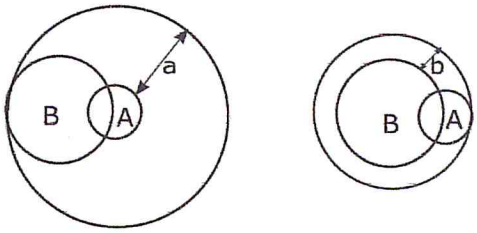


droite

(c) A un disque inclus dans un carré B
(ici on a $b = 0$ car $A \subset B$)



(d)A et B sont deux disques se rencontrant



Théorème 2.3.1 (Hausdorff) L'ensemble $\mathcal{K}$ est un espace métrique complet avec la distance h de *Hausdorff.*

Remarque. En particulier, le principe de contraction est vrai pour l'ensemble $\mathcal{K}$ muni de la distance h

2.4.3 Théorèmes de l'ensemble fixe et de collage

Définition 2.3.2. Un système itératif de fonctions (SIF) dans le plan est une famille finie $\{w_1, \ldots, w_N\}$ de fonctions de R^2 dans R^2.

Soit $\{w_1, \ldots, w_N\}$un SIF. Désignons par $W(E) = w_1(E) \cup w_2(E) \cup \cdots \cup w_N(E)$ ou E est un sous-ensemble de R^2. Comme on le fait pour les fonctions, on peut itérer W :

$$E_0 : \textit{un sous} - \textit{ensemble de } R^2$$
$$E_1 = W(E_0) = w_1(E_0) \cup \cdots \cup w_N(E_0)$$
$$E_2 = W(E_1) = W\big(W(E_0)\big) = W_2(E_0)$$
$$\ldots$$
$$E_n = W_n(E_0)$$

Théorème 2.3.2 (Hutchinson)[3] Soit $\{w_1, \ldots, w_N\}$ un SIF ou $w_1, \ldots, w_N$ sont des contractions sur R^2 de facteurs $k_1, \ldots, k_N$. Alors, W est une contraction sur $\mathcal{K}$ de facteur k où $k = max(k_1, \ldots, k_N)$ et

(i) il existe un unique ensemble fixe $\bar{A}$ de W, c'est-à-dire tel que $W(\bar{A}) = \bar{A}$

(ou encore $\bar{A} = w_1(\bar{A}) \cup w_2(\bar{A}) \cup \cdots \cup w_N(\bar{A})$) ;

(ii) pour tout sous-ensemble compact $A_0 \subset R^2$, la suite $\{A_n\}_{n \in N}$ de sous-ensembles compacts de R^2 définis par $A_n = W_n(A_0)$ converge vers $\bar{A}$, c'est-à-dire

$$\lim_{n \to \infty} h(A_n, \bar{A}) = 0$$

En outre, on a

$$h(A_n, \bar{A}) \leq \frac{k_n}{1-k}\ h(A_1, A_0)\ \forall n$$

Ce qui s'écrit aussi comme $(car\ A_1 = W(A_0))$

$$h(A_n, \bar{A}) \leq \frac{k_n}{1-k}\ h(W(A_0), A_0) \forall n. \tag{2.25}$$

En particulier, pour $n = 0$, on obtient

$$h(A_0, \bar{A}) \leq \frac{1}{1-k}\ h(W(A_0), A_0). \tag{2.26}$$

Démonstration : Il suffit de montrer que $W : \mathcal{K} \to \mathcal{K}$ est une contraction. Ensuite, on peut appliquer le théorème du point fixe puisque $\mathcal{K}$ est un espace métrique complet.

Théorème 2.3.3 (Collage)[1] Soient $L \in \mathcal{K}$ et $\varepsilon > 0$. Soit $\{w_1, \ldots, w_N\}$ un SIF ou $w_1, \ldots, w_N$ sont des contractions de facteurs $k_1, \ldots, k_N$ telles que

$$h(W(L), L) \leq \varepsilon$$

Alors, si $\bar{A}$ est l'ensemble fixe de W, on a

$$h(L, \bar{A}) \leq \frac{\varepsilon}{1-k}\ ou\ k = max\{k_1, \ldots, k_N\} \tag{2.27}$$

Démonstration : En prenant $A_0 = L$ dans l'équation 1.2, on obtient

$$\begin{aligned} h(L, \bar{A}) &\leq \frac{1}{1-k}\ h(W(L), L) \\ &\leq \frac{\varepsilon}{1-k} \qquad car \quad h(W(L), L) \leq \varepsilon \end{aligned}$$

En pratique, on utilise le théorème de collage pour approximer un sous-ensemble compact de R^2 (un fractal par exemple !) comme suit :
Soit L un sous-ensemble compact de R^2. Si on réussit a trouver un SIF $\{w_1, \ldots, w_N\}$ de contractions telles que $L \approx W(L) = w_1(L) \cup \cdots \cup w_N(L)$, c'est-à-dire $h(W(L), L)$ est petit, alors l'ensemble fixe $\bar{A}$ du SIF sera une bonne approximation de L , c'est-à-dire $h(L, \bar{A})$ sera petit.

Exemple 2.3.1 Comment obtenir un SIF pour le triangle de Sierpinski ?
Soit L l'ensemble illustré ci-dessous. Désignons par L_1, L_2 *et* L_3 les trois portions de L encerclées. Nous avons $L = L_1 \cup L_2 \cup L_3$.

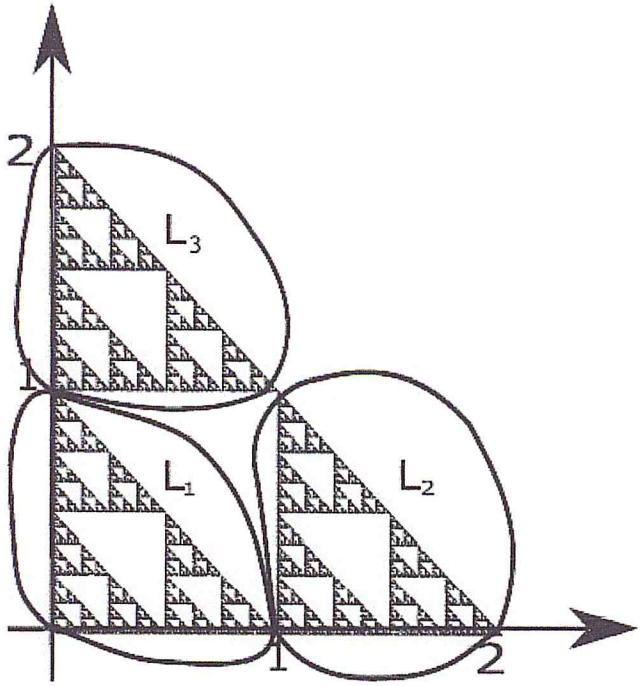


Fig 2.1 Obtenir un SIF pour le triangle de Sierpinski

Soient

$$w_1\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}0.5 & 0\\0 & 0.5\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \qquad d'où\ w_1(L) = L_1$$

$$w_2\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}0.5 & 0\\0 & 0.5\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + \begin{pmatrix}1\\0\end{pmatrix} \qquad d'où\ w_2(L) = L_2$$

$$w_3\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}0.5 & 0\\0 & 0.5\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix} \qquad d'où\ w_3(L) = L_3$$

Ainsi, $L = w_1(L) \cup w_2(L) \cup w_3(L)$ et donc l'ensemble fixe A de ce SIF est presque égal à L (en fait ici, L sera égal a A).

2.4.4 Méthodes de construction des IFS

Le théorème pour la construction des IFS que nous venons de présenter peut se programmer et nous permettre de générer des fractales. Nous allons donc présenter deux algorithmes génériques pour y arriver. La première façon consiste à appliquer systématiquement toutes les transformations de l'IFS à une figure de départ quelconque. Ensuite, nous faisons l'union des images obtenues. Cette nouvelle image remplace l'image de départ. Nous lui appliquons de nouveau les transformations et ainsi de suite. C'est ce que nous appelons la méthode déterministe. Une variante possible consiste a commencer avec un point et a ne choisir qu'une seule transformation a chaque itération. Ces choix sont faits aléatoirement. Chaque itération produit un nouveau point qu'on ajoute a la figure. C'est la méthode probabiliste[1].

2-4-4-1 Un premier algorithme (déterministe)

Soient $w_1, \ldots, w_N$ des contractions de R^2 dans R^2

– On choisit un point de départ P_0.

– On calcule

$$W(P_0) = w_1(P_0) \cup \cdots \cup w_N(P_0)$$

puis

$$W^2(P_0) = W(W(P_0))$$

$$\ldots$$

$$W^n(P_0) \ldots, etc.$$

Pour n assez grand, $W_n(P_0)$ est une approximation de l'ensemble fixe A (d'après le théorème de l'ensemble fixe).

Cependant, cet algorithme n'est pas très efficace car il exige beaucoup de calculs !

1 ère étape : on calcule N points.

2 eme étape : on calcule N_2 points.

3 eme étape : on calcule N_3 points.

...

n eme étape : on calcule N_n points.

Ça augmente vite !

Pour remédier a ce problème, on procède comme suit.

/*Début L'algorithme */
Effacer l'écran
Déclarer deux tableaux et mettre les éléments à 0.
s(taille,taille)=0
t(taille,taille)=0
Déclarer 6 tableaux (a,b,...,f) par fonction et les initialiser
avec les paramètres de la transformation.
a(1)=a1, b(1)=b1, c(1)=c1, d(1)=d1,e(1)=e1, f(1)=f1
a(2)=a2, b(2)=b2, c(2)=c2, d(2)=d2,e(1)=e2, f(1)=f2
...
Initialiser le tableau t avec la forme voulue.
Pour i allant de 1 à taille faire
t(taille/2,i)=1 Avec une ligne verticale PAR EXEMPLE.
fin pour
Faire une itération.
Pour i allant de 1 à taille faire
Pour j allant de 1 à taille faire
si t(i,j)=1 alors
s(a(1)*i+s(b(1)*j)+e(1), c(1)*i+d(1)*j+f(1))=1
s(a(2)*i+s(b(2)*j)+e(2), c(2)*i+d(2)*j+f(2))=1
...
fin si
fin pour
fin pour
Pour i allant de 1 à taille faire
Pour j allant de 1 à taille faire
t(i,j)=s(i,j) 'Copier le tableau s dans t.
s(i,j)=0 'Remettre tous les éléments de s à 0.
si t(i,j)=1 alors
Afficher (i,j)
fin si
fin pour
fin pour
Recommencer les itérations jusqu'a satisfaction!
/*fin d'algorithme*/

2-4-4-2 Un deuxième algorithme (stochastique)

Considérons un SIF $\{w_1,\dots,w_N\}$ et attachons a chaque w_i une probabilité p_i de telle sorte que $p_1 + p_2 + \cdots + p_N = 1$.
– Choisissons un point de départ Q_0.

– On choisit au hasard une des fonctions w_i (la probabilité que w_i soit choisie est p_i) et on calcule $Q_1 = w_i(Q_0)$.
– On choisit au hasard une des fonctions w_i et on calcule $Q_2 = w_i(Q_1)$
– Etc.

On peut montrer que la suite des itérés $\{Q_n\}$ finit par " recouvrir" l'ensemble fixe A.

/* Début de l'algorithme*/
Effacer l'écran
Déclarer les variables et les initialiser à 0.
x=0
y=0
newx=0
newy=0
k=0
Déclarer 6 tableaux (a,b,...,f) et les initialiser avec
les paramètres de la transformation.
a(1)=a1, b(1)=b1, c(1)=c1, d(1)=d1,e(1)=e1, f(1)=f1
a(2)=a2, b(2)=b2, c(2)=c2, d(2)=d2,e(2)=e2, f(2)=f2
...
Faire les 10 premières itérations sans les afficher.
Pour i allant de 1 à 10 faire
Générer un nombre aléatoire entre 1, 2 ou 3.
k=int(3*random())+1 'Ajuster si la probabilité n'est pas égale!
newx=a(k)*x+b(k)*y+e(k)
newy=c(k)*x+d(k)*y+f(k)
x=newx
y=newy
fin pour
Faire les autres itérations et les afficher.
Pour i allant de 1 à au choix faire
k=int(3*random())+1
newx=a(k)*x+b(k)*y+e(k)
newy=c(k)*x+d(k)*y+f(k)
x=newx
y=newy
afficher (x,y)
fin pour
Recommencer les itérations jusqu'a satisfaction!
/* fin de L'algorithme */

2.4.5 Un exemple simple

Soit la famille de fonctions affines dans R^2 suivante :

$$f_1 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$f_2 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 50 \end{pmatrix}$$

$$f_3 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 50 \\ 50 \end{pmatrix}$$

Puisque chaque f_i est une contraction (on fait subir une homothétie de rapport $1/2$), la fonction

$$f(A) = f_1(A) \cup f_2(A) \cup \ldots \cup f_n(A)$$

possède un unique attracteur. En itérant $A_n = f(A_{n-1})$ a partir de n'importe quel dessin dans R^2, on converge nécessairement vers cet attracteur qui est en réalité un triangle de *Sierpinsky* dont les sommets sont (0, 0), (0, 100) et (100, 100).

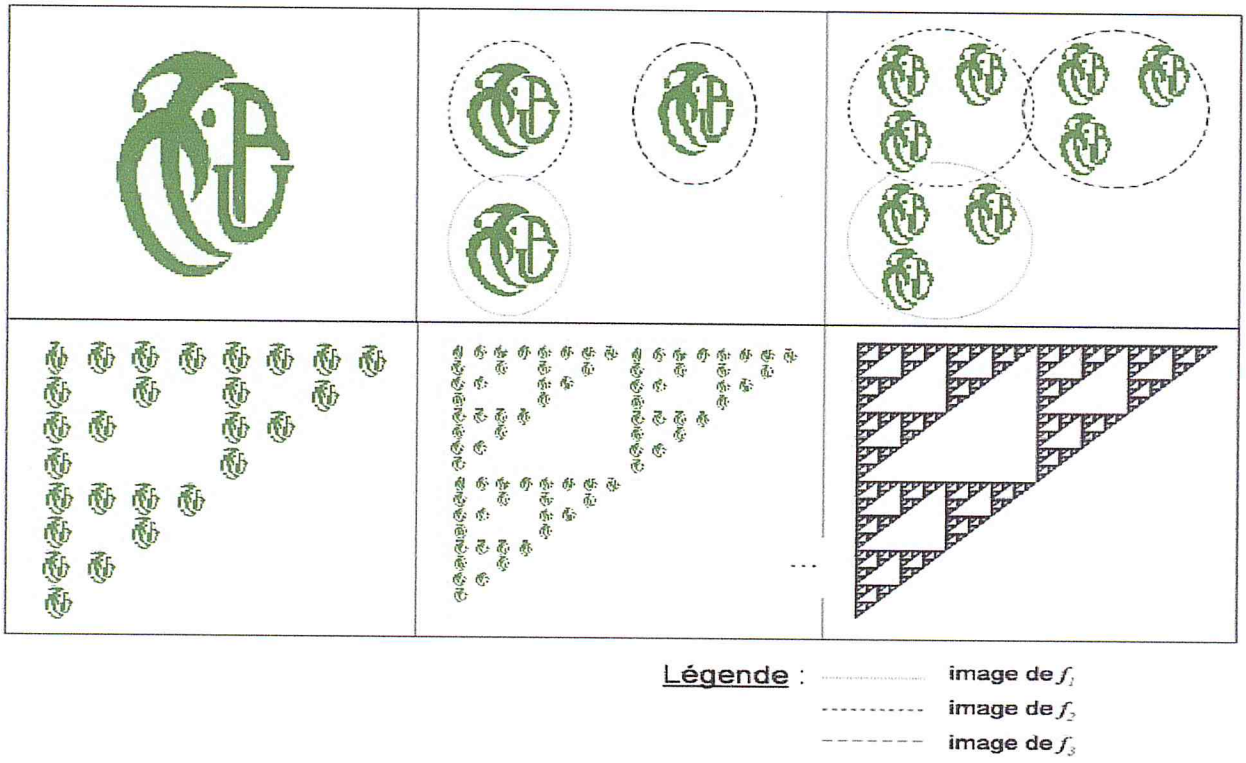


Fig. 2.2– Illustration de la convergence d'une image quelconque vers un triangle de *Sierpinsky*.

2.5 Conclusion

Une application répandue des fractales est la compression de données, et plus particulièrement d'images en ce qui nous concerne. L'idée de base de cette méthode de compression est de trouver des similarités dans l'image à différentes échelles. Cette application étant un axe majeur de nos travaux, elle sera détaillée plus loin dans le document.

Chapitre 3
La Compression D'Image

3.1 Introduction

Au cours des dernières années, la demande et le développement de produits multimédias et de la capacité de stockage de dispositif de stockage ont augmenté à un rythme effréné. En raison de la compression qui est devenue une technique clé pour réduire la taille du nombre de bits le plus possible. Un processus inverse connu sous le nom de décompression (décodage) peut être appliqués à des données compressées pour obtenir l'image reconstruite.

La compression de l'image en réduisant la taille en octets d'un fichier graphique sans nuire à la qualité des images qui réduisent à la fois la redondance spatiale et spectrale dans les données d'image pour stocker ou transmettre sous une forme efficace. Le processus de compression de l'image est de deux types qui sont la méthode de compression avec et sans perte. La compression sans perte est renvoyée pour des fins d'archivage et souvent pour l'imagerie médicale, des dessins techniques, et des Clip Art. Méthodes de compression avec perte, surtout lorsqu'il est utilisé à des débits faibles, génère des artefacts de compression. Les méthodes appropriées pour des images naturelles telles que le nuage, arbre, montagne et sont nommés en tant que méthode pertes en cas de perte imperceptible de fidélité est acceptable, pour obtenir une réduction substantielle des taux de bit. La plupart des méthodes sont utilisées peuvent être classées sous la compression avec perte. Cela signifie que l'image reconstruite est une approximation de la vraie image [10].

3.2 L'intérêt de compression d'image

La compression et le codage consiste a la réduction de la taille physique d'un bloc d'information (réduction du nombre de bits par pixel à stocker ou a transmettre). En exploitant la redondance informationnelle dans l'image. Trois sortes de redondances sont exploitées dans la compression d'images :

- La redondance spatiale entre pixels ou blocs voisins dans l'image ;
- La redondance temporelle entre images successives dans une séquence vidéo ;

- La redondance spectrale entre plans de couleur ou bandes spectrales ;

Les principaux critères d'évaluation de toute méthode de compression sont :

- La mesure de qualité : la qualité d'un système se mesure par la qualité de reconstruction de l'image ;
- La rapidité du codeur et décodeur ;
- La réduction des débits : le taux de compression dépend de l'application

3.3 Les différents types de compression

3.3.1 Compression physique et logique

3.3.1.1 La compression physique

C'est un traitement qui agit directement sur les donnés ; ce traitement est capable de comprimer l'information en agissant sur les données redondantes, afin qu'elles occupent un minimum de place, La lecteur du fichier compressé est incompréhensible pour l'utilisateur

3.3.1.2 La compression logique

C'est un traitement différent, c'est plutôt une réorganisation plus compacte des donnés l'information est mieux structurée et les données originales sont substituées par une autre information équivalente. Ce processus de substitution logique consiste à remplacer des symboles par d'autres plus petites, mais qui logiquement ont une même signification.

3.3.2 Compression symétrique et asymétrique

3. 3.2.1 La compression symétrique

Utiliser la même méthode pour compresser les donnes que la décompression. La quantité de travail effectue est la même dans les deux cas. Ce type de compression est généralement utilise dans la transmission de donnés.

3.3.2.2 La compression asymétrique

Nécessite une plus grande quantité de travail soit pour la phase de compression ou de la décompression. Tout dépend de l'application que l'on utilisé. Si on prend le cas de l'archivage, la compression sera plus rapide que la décompression, car on accède rarement à ces données.

3.3.3 Encodage adaptif, semi adaptif et non adaptif

3.3.3.1 Un encodeur non adaptif

C'est basé sur un dictionnaire spécifique à un jeu de données précis. Par exemple si on utilise un dictionnaire basé sur la langue français, alors l'application de l'encodeur ne pourra se faire que sur des fichiers textes dont le contenu est en français.

3.3.3.2 Un encodeur adaptif

Permet de s'adapter aux données suivant la technique utilise pour l'encodage des données un dictionnaire peut être construit au fur et à mesure de la lecture des données.

3.3.3.3 Un encodeur semi adaptif

C'est très proche d'un encodeur adaptif, la différence que l'on observe est que l'encodeur semi adaptif à besoin d'effectuer deux fois le lecteur des données. La première lecture va servir à construire le dictionnaire et la seconde lecture permettra d'encoder les données [7].

3.4 Modèle général pour l'analyse des méthodes de compression

La problématique de la compression peut être considère de différents points de vue. Le schéma général souvent utilise pour d'écrire le fonctionnement des algorithmes de compression est celui présenté dans la figure 3.1 :

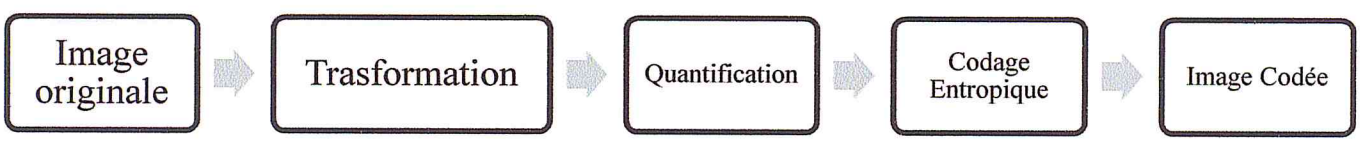


Fig 3.1 Schéma général de la compression

3.4.1 Transformation ou décorrélation

La dépendance existante entre chacun des pixels et ses voisins (la luminosité varie très peu d'un pixel à un pixel voisin) traduisent une corrélation très forte sur l'image. La décorrélation consiste à transformer les pixels initiaux en un ensemble de coefficients moins corrélés pour réduire le volume d'information, c'est une opération réversible.

3.4.2 Le codage entropique

Le codage entropique effectue un codage sans perte sur les valeurs quantifiées. Cette dernière étape est nécessaire dans les méthodes sans perte, mais elle est souvent présente aussi dans les algorithmes irréversibles, puisque les valeurs transformées et quantifiées contiennent davantage de redondances [11]. Cependant, l'absence du codeur entropique peut être justifie et nécessaire, à cause notamment des contraintes de transmission.

3.5 Classification des méthodes de compression

On peut regrouper les méthodes de compression en cinq classes à l'aide de différents entiers .A la suite nous détaillons les méthodes les plus courantes.

3.5.1 Méthodes avec ou sans pertes d'information

Cette première classification s'intéresse à la présence ou non d'une distorsion ou perte d'information introduite par la compression. Notons que le taux de compression est limité par l'entropie de l'image pour les méthodes réversibles ou sans perte d'information, par conter pour les méthodes irréversibles ou avec perte d'information, le taux de compression est sensiblement supérieur à l'entropie de l'image [12].

3.5.2 Méthodes par pixels, bloc de pixels, ou image entière (scène)

Cette deuxième classification concerne les méthodes de compression dont leurs algorithmes touchent : le niveau pixel, blocs de pixels, ou toute l'image, on peut citer quelques exemples :

- Codage individuel des pixels : codage de *Huffman*.

- Codage de Blocs de pixels : les normes standards JPEG et MPEG qui travaillent par bloc de 8 × 8 , la quantification vectorielle , et le codage fractal .
- Codage de l'image entière : Des approches de codage par région, transformée cosinus discrète DCT.

3.5.3 Méthodes Intra et Inter-images

Cette troisième classification s'applique aux séquences d'images (séries de coupes 3D). Les méthodes intra-images (intra-frame en anglais) effectuent la compression de chaque image indépendamment. Les méthodes inter-image (inter-frame) exploitent la redondance entre les images successives, Le standard MPEG code des séquences d'images en détectant le mouvement d'une image à l'autre [2].

3.5.4 Méthodes spatiales et méthodes par transformation

La quatrième classification s'intéresse au domaine dans lequel s'effectuent les opérations de base de la compression. Une image peut être représente de deux façons strictement équivalentes :

- Dans le domaine spatial, dans lequel l'image est représentée sous forme de pixels , c'est le domaine accessible visuellement à l'observateur .
- Dans le domaine fréquentiel, dans le quel l'image est représentée sous la forme de coefficients de fréquences spatiales, le passage d'un domaine à l'autre se fait par des transformations mathématiques les plus connues, telles que :
  - La transformation de *Fourier*.
  - La transformation en cosinus discrète (DCT Discrète cosine transform) valable pour les images fixes, elle traduit l'information spatiale en une information fréquentielle.

3.5.5 Méthodes adaptatives, non adaptatives

Cette cinquième classification indique si la méthode de compression est adaptative ou non.

- ✓ *Une méthode adaptative* : qui construit sa table de correspondance à l'analyse de l'information (modifier ses paramètres au fur et à mesure du codage, en s'adaptant aux données d'entrées) [2].
- ✓ *Une méthode non adaptative* : la table correspondance statistique et prédéfinie, Elle peut également y avoir une compression semi adaptative (mélange des deux méthodes).

3.6 Présentation des méthodes avec et sans pertes

On peut distinguer deux grandes familles d'algorithmes de compression ; les méthodes dites sans perte ou réversibles garantissent la restitution parfaite des images, alors que les méthodes dites avec perte ou irréversibles modifient plus ou moins la valeur des pixels.

3.6.1 Les méthodes réversibles ou sans pertes

Comme son nom l'indique, ce type de compression n'occasionne aucune perte de données .Cette compression conservatrice est utilisée dans des applications comme l'archivage des images médicales, l'imagerie satellitaire(le cout des images est élevé et les détails sont importants), les textes, les programmes et tout autre type de donnée nécessitant une conservation à l'identique des données, le seul critère d'évaluation des performances est dans ce cas « Les taux de compression ». Il existe de nombreux types de compression d'image sans perte de données .Voici les plus répandus :

3.6.1.1 Méthodes différentielles et prédictives

La méthode prédictive est l'une des plus anciennes, c'est une méthode décorrélative dont le principe est le suivant :

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 .Ce qui permet l'élimination de la redondance et en codant que la nouvelle information apportée par chaque pixel. Dans des systèmes plus complexes et performants, on établit une fonction de prédiction, qui permet d'estimer la valeur d'un pixel en fonction de la valeur des pixels voisins. On code alors l'erreur de prédiction, qui est l'écart entre la vraie valeur du pixel la valeur prédite [2].

3.6.1.2 Méthodes par plages

- *Le codage par répétition ou « Run Length Coding » (RLC)*

Plusieurs types d'algorithmes sont utilises pour compresser l'information vont de plus en plus simple au plus complexe, et leur efficacité varie suivant la na ture des données à comprimer. Le plus simple de ces algorithmes est le Run Length Encoding (RLE), méthode aussi appelée Run Length Codin (RLC).

C'est une technique avec mémoire (« avec mémoire » signifie qu'elle code les valeurs d'entrée en prenant en compte les valeurs précédentes). Son principe est de regrouper les valeurs voisines identiques et ne transmettre cette valeur qu'une seule fois, précédée par le nombre de répétition. Il est clair que cette approche fonctionne bien s'il y a beaucoup de répétitions dans le signal. Cet algorithme très simple, il peut aboutir à des taux de compression plus élevés. Il existe variantes dans lesquelles l'image est encodée par pavés de points, selon des lignes ou bien même en zigzag.

3.6.1.3 Codage de Shannon-Fano

C.Shannon du laboratoire *Bells* et *R.M .Fano* du MIT ont développés à peu prés en même temps une méthode de codage basée sur de simples connaissances de la probabilité d'occurrence de chaque symbole dans le message.

La procédure se décrit ainsi :

- Procédure de codage
  - Etape 1 : Classer les n fréquences non nulles $\{f_i\}$ par ordre décroissante.
  - Etape 2 : Deviser l'ensemble des messages en deux sous ensembles de fréquences aussi proches que possible.
  - Etape 3 : Attribuer à chaque sous ensemble un bit 0 ou 1.
  - Etape 4 : Rediviser chaque sous ensemble en deux nouveaux sous ensembles de fréquences équivalentes afin de réitérer l'algorithme jusqu'a ce qu'il n'y ait plus qu'un seul élément dans chaque sous ensemble.

Exemple

La chaine que nous allons traiter est : ACBBCDECECBCEEECCCCABCECBDBDBD

c-à-d $f(A) = 2, f(B) = 7, f(C) = 11, f(D) = 4, f(E) = 5$

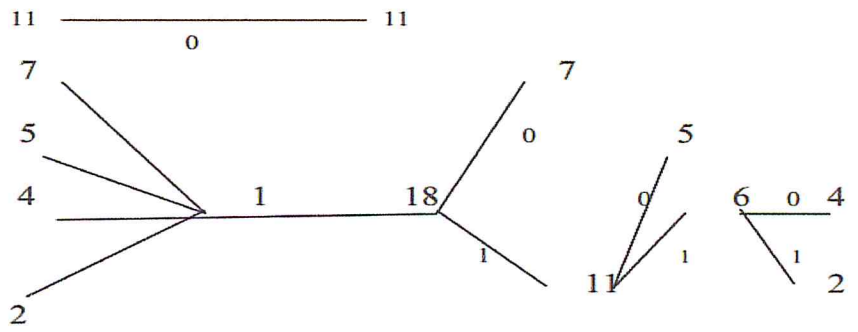
Octet	Fréquence processus	Code
C		0
B		10
E		110
D		1110
A		1111

Fig 3.2 Codage de Shanon –Fano [19]

On remarque que le fichier compresse comporte 64 bits contre 196 bits pour le fichier original.

3.6.1.4 Le codage de *Huffman*

En 1952, *David Huffman* inventa une nouvelle méthode de compression appelée *compression à Arbre de Huffman* [13]. Le codage de *Huffman* crée des codes à longueurs variables sur un nombre entier de bits, L'algorithme considère chaque message à coder comme étant une feuille probabilité, les mots codent les plus longs. Ces deux mots codes ne se 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.

- *Procédure de codage*
  - 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.
  - Les deux probabilités les plus faibles sont identifiées en fin de liste.
  - La somme des deux probabilités est placée à sa place dans liste triée. Elle constitue un nœud parent. Les deux enfants sont retirés de la liste.
  - Le chemin « enfant de plus faible probabilité, parent » est codé par un 1, l'autre par un 0.
  - La procédure reprend à l'étape 2 jusqu'à ce qu'il ne reste plus qu'une probabilité dans la liste.

Exemple

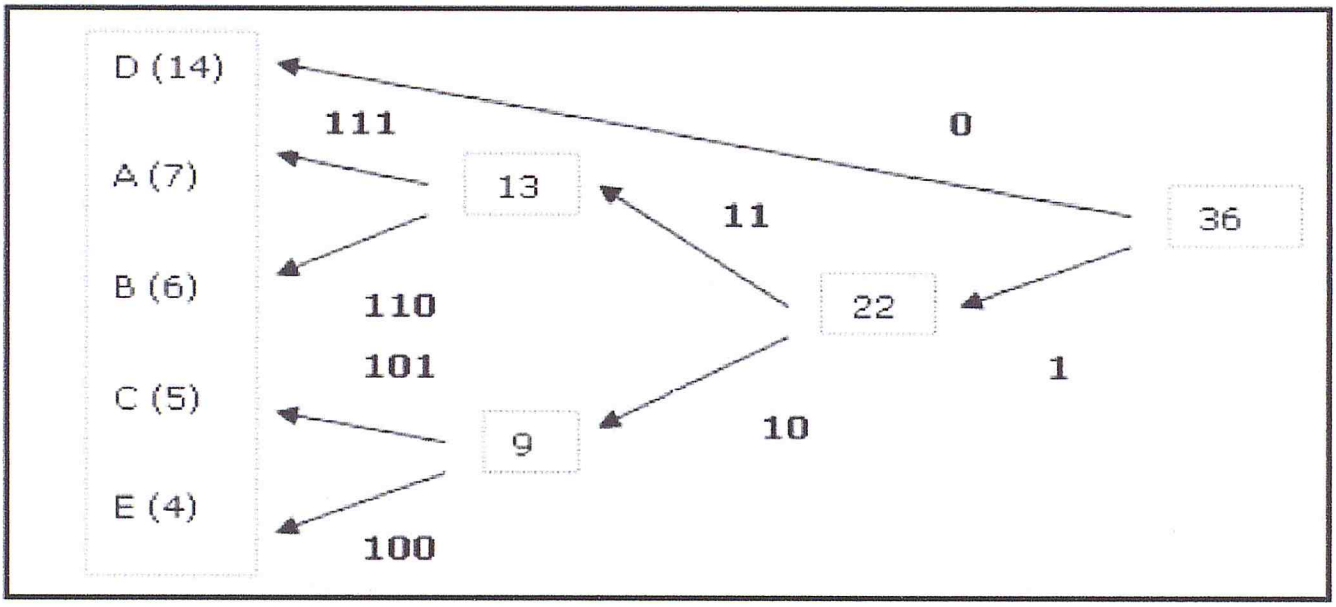


Fig 3.3 Algorithme de *Huffman*[5]

Le codage obtenu est donc :

D(occurrence=14) :0 A(occurrence=7) :111 B(occurrence=6) :110

C(occurrence=5) :101 E(occurrence=4) :100

3.6.1.5 Le codage arithmétique

Le codage arithmétique est un codage récent utilisant un modèle statistique, tout comme le codeur de *Huffman*. Contrairement à ce dernier, il produit un code pour une séquence de symboles tout entière, et non pas un code symbole .Chaque nouveau symbole lu modifie de façon incrémentale le code de sortie .Ce code de sortie est un nombre à virgule flottante compris entre 0 et 1, dont le nombre de chiffres après la virgule correspond au nombre de symboles, Contrairement à *Huffman*, il n'est pas obligatoire que chaque code ait un nombre entier de bits.

Par exemple un symbole de probabilité 0.9 a pour entropie 0.15, mais Huffman affectera probablement un code de un bit (ou plus), et la séquence codée aura un nombre de bits plus long qu'en théorie. Le codeur arithmétique est plus performent que le codeur de *Huffman*, mais il est plus complexe à implémenter.

■ *Algorithme de codage arithmétique*

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

❖ Etape 1 : Calculer la probabilité associée à chaque symbole dans la chaine à coder.

❖ Etape 2 : Associer à chaque symbole un sous intervalle proportionnel à sa probabilité dans l'intervalle [0,1[(l'ordre de rangement des intervalles sera mémorise car il est nécessaire au décodeur)

❖ Etape 3 : Initialiser la limite inférieure de l'intervalle de travail à la valeur 0 et la limite supérieure à la valeur 1.

❖ Etape 4 : Tant qu'il reste un symbole dans la chaine à coder :

- Largeur=limite supérieur-limite inférieur.
- Limite inférieure = limite inférieure + largeur × (limite basse du sous intervalle du symbole)
- Limite supérieure = limite inferieur+largeur ×(limite haute du sous intervalle du symbole)

❖ Etape 5 : La limite inferieure code la chaine de manière unique.

3.6.1.6 Codage par dictionnaire adaptatif (LZW) (Lempel-Ziv-Welch) ou LZ7

Elle utilise un dictionnaire qui n'est pas stocké dans le fichier compresse qu'elle construit dynamiquement, au cours de la compression et de la décompression, il s'agit cette fois de repérer des motifs composés de séries variables en longueur de bits ou d'octets. Chaque motif est copie dans le dictionnaire et se voit attribue un indice, puis chaque motif est remplacé dans le fichier par son indice, sachant qu'une valeur isolée n'est pas codée, elle a besoin d'un apprentissage pour été efficace, pour reconnaitre des longues chaines répètes, En effet, l'algorithme ne fonctionne pas sur un nombre fixe de motifs mais apprend les motifs du fichier durant le lecteur du fichier à compacter. Elle est donc peu efficace sur des petits fichiers. Cette méthode est très rapide.

L'algorithme suivant montre son mécanisme

```
w = Nul;
   tant que (lecture d'un caractère c) faire
       si (w + c existe dans le dictionnaire) alors
           w = w + c;
       sinon
           ajouter w + c au dictionnaire;
           écrire le code de w;
           w = c;
       fin si
   fin tant que
   écrire le code de w;
```

3.6.2 Les méthodes de compression avec pertes ou irréversible

Les méthodes irréversibles permettent des taux de compression assez élevés au prix d'une dégradation de la qualité de l'image. Outre, le taux de compression, une mesure de cette dégradation est nécessaire à l'évaluation des performances de ces méthodes. Plusieurs mesures existent pour l'évaluation des performances de ces méthodes comme le SNR ; MSE et PSNR (voir 3.9).

Avec ces méthodes, on peut aussi distinguer :

- ✓ Les méthodes spatiales (ou directes) qui agissent directement sur les échantillons d'une image dans le domaine spatial.
- ✓ Les méthodes par transformation qui reposant sur une transformée (en général linéaire) de l'image originale.

3.6.2.1 Quantification

En général, la quantification apparait en deuxième lieu dans un processus de compression pour réduire la quantité d'information, de manière souvent irréversible [14].

Ainsi, une quantification est utilisée pour simplifier la représentation, tout en préservant l'information la plus pertinente. On remplace ainsi les valeurs initiales par un ensemble fini d'éléments qui donneront des résultats acceptables lors de la phase de décompression. Ces éléments qui donneront des résultats acceptables lors de la phase de décompression. Ces éléments peuvent être scalaires lorsque, on remplace l'information en caque pixel par des nombres entiers, ou des vecteurs lorsque l'on considère des groupes de pixels représentant des configurations typiques.

Dans les deux cas, le principe est décrit par la figure 3.4. Un élément quelconque (scalaire ou vecteur) est remplacé par l'élément du dictionnaire le plus approprie

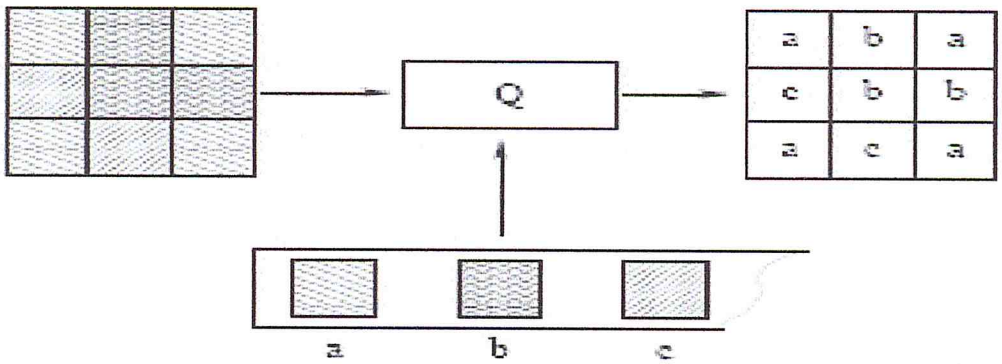


Fig 3.5 : Principe de la quantification [19]

3.6.2.2 Codage par transformée

Dans ces méthodes, l'image de dimension $N \times N$ est subdivisée en sous images ou blocs de taille réduite (la quantité de calcul demandée pour effectuer la transformation sur l'image entière est très élevée). Chaque bloc subit une transformation mathématique orthogonale inversible linéaire du domaine spatial vers le domaine fréquentiel, indépendamment des autres blocs (transformée en un ensemble de coefficients plus ou moins indépendants). Les coefficients obtenus sont alors quantifiés et codés en vue de leur transmission ou de leur stockage. Pour retrouvé l'intensité des pixels initiaux, on applique sur ces coefficients la transformation inverse.

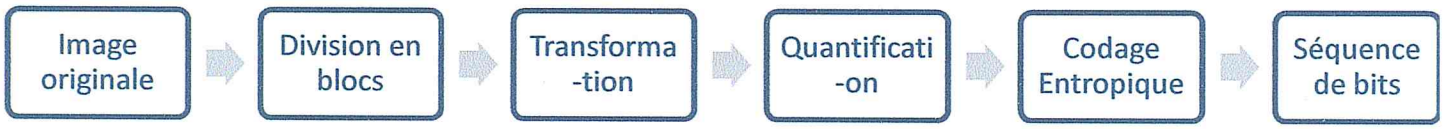
L'objectif de ces transformations est double : il s'agit de

* Décorréler les données, c'est-à-dire d'obtenir des coefficients transformés moins corrélés que les pixels de l'image
* Concentrer l'énergie sur un nombre réduit de coefficient, les coefficients ayant une valeur plus importante à la basse fréquence qu'aux fréquences.

Les méthodes de codage par transformation présentent des propriétés d'immunité au bruit de transmissions bien supérieures à méthodes précédentes. Une erreur affecte en effet seulement la valeur d'un coefficient, et sera lissé au décodage lors du calcul de la transformation inverse, L'effet sera ainsi peu visible.

Le principe d'un système de codage par transformation est le suivant :

Phase de compression



Phase de décompression

Fig 3.6 Schéma de principe de la compression/décompression par transformation

3.6.2.3 La Compression par ondelettes

Les ondelettes c'est d'abord un théorème mathématique récent d'analyse du signal, développée dans les années 80. On peut considérer qu'il s'agit d'une extension de l'analyse de *Fourier* [5].

Quand en enregistre une image en utilisant les ondelettes, on divise sa résiliation par deux et on code l'information perdue par des coefficients d'ondelettes. On commence donc à coder les détails les plus fins(Les hautes fréquences). On recommence l'opération autant de fois que nécessaire jusqu'à ce que l'image se réduise à 1 pixel. A chaque étape l'image est « lissée » et les détails perdus sont codés en coefficients d'ondelettes.

L'intérêt da la transformation par ondelettes par rapport aux méthodes de compression est celle-ci ne considère pas l'image dans son ensemble pour la coder mais, la travaille par couche, cherchant à enregistre les détails les plus importants (ceux qui se démarquant le plus du reste du signal) a chaque résolution.

* *Algorithme de compression par ondelettes ou Waveletes*

Passons maintenant à l'algorithme pyramidal utilisé. La décomposition en coefficients d'ondelettes n'utilise pas une fonction de moyenne, mais s'appuie sur deux filtres. Un filtre passe bas(L) et un filtre passe haut (H). La combinaison de ces filtres permet d'obtenir quatre sous images HH, HL, LH et LL. Ces filtres sont nommés miroirs et quadratures.

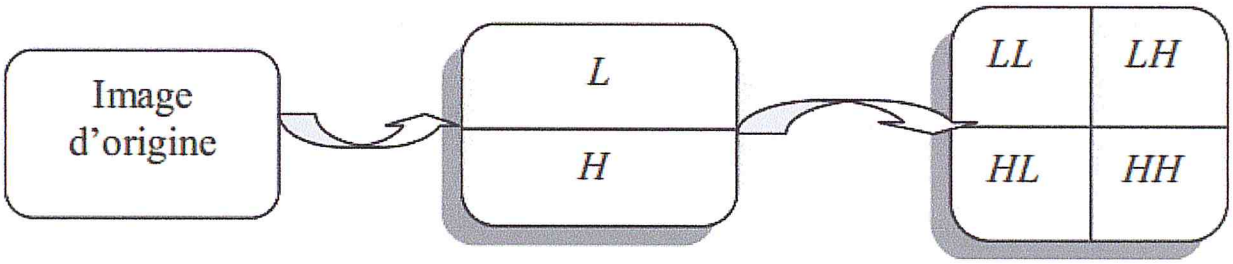


Fig 3.7 Transformation en colonnes et en lignes

Chacune des quatre images obtenues par la transformation représente des informations bien distinctes.

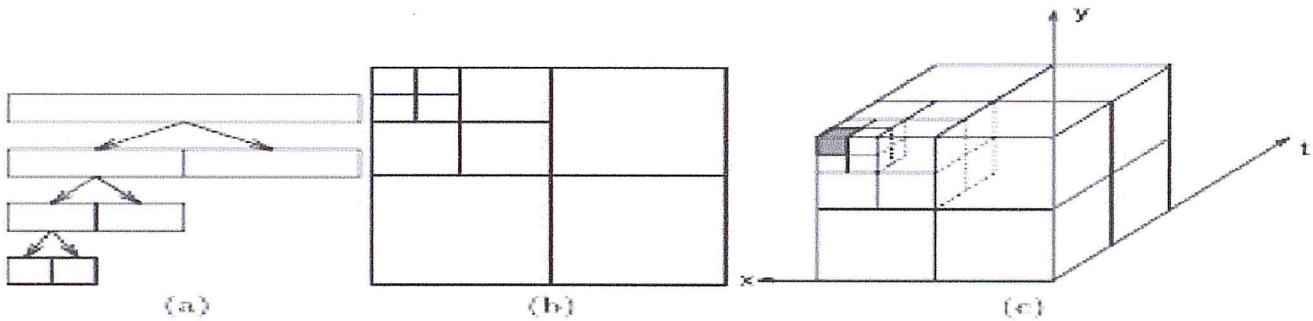


Fig 3.7 Transformation ondelette « pyramidal » ((a) 1-D,(b) 2-D,(c) 3-D)[13]

* *Les étapes de compression par ondelettes*

Tout d'abord, voici le principe de compression et décompression d'une image par ondelettes

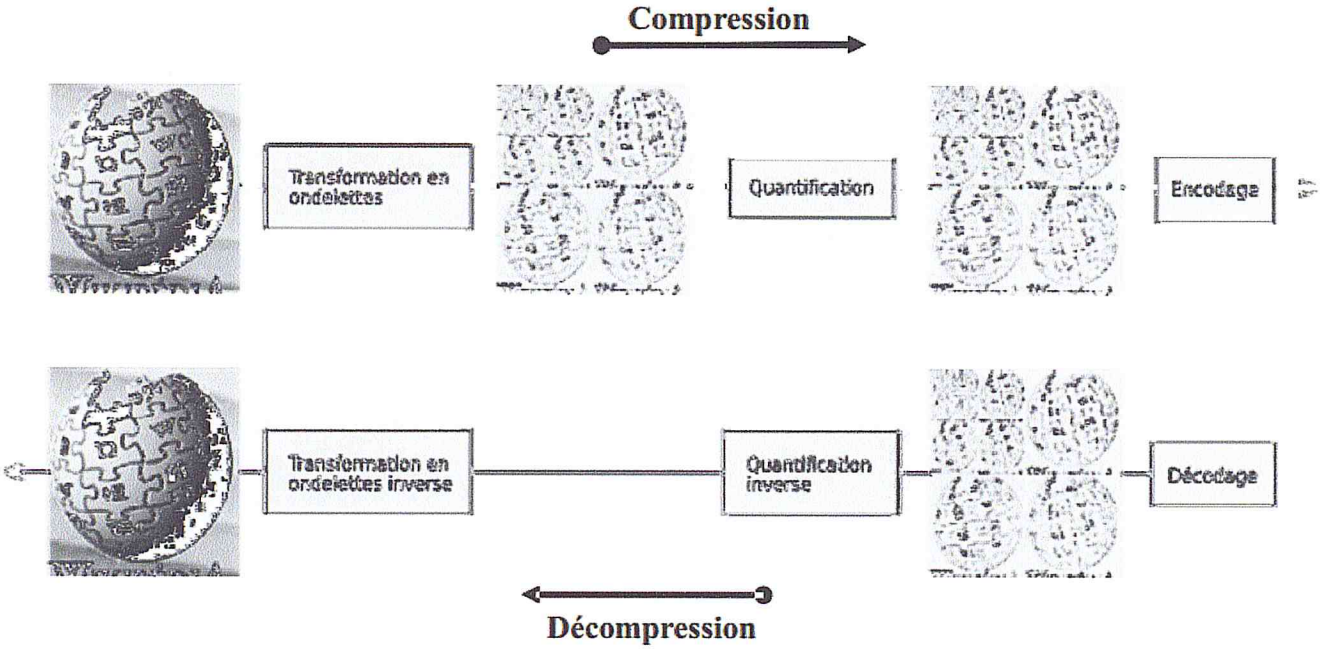


Fig 3.8 Le principe de compression /décompression par ondelette [13]

* *La compression ondelettes*

Les étapes pour compresser une image sont

1- Transformations par ondelettes.
2- Quantification : les valeurs des images de détails inférieurs à un certain niveau sont éliminées, en fonction de l'efficacité recherchée, c'est cette étape qui introduit des pertes.
3- Codage de valeurs restantes, les données restantes sont transmises à un encodeur entropie, c'est-à-dire à un algorithme de compression de données (LZW, *Huffman*,….)

* *Décompression ondelettes*

La transformation inverse par ondelettes reconstruit une image originale. La construction de l'image à partir des sous-bandes restitue l'image en mode progressif, l'affichage de l'image peut s'effectuer en deux modes :

- Soit la taille de l'image augment au fur et à mesure de la lecture du fichier compressé.
- Soit la résolution de l'image augmente au fur et à mesure de la lecture du fichier compressé.

3.6.2.4 **La méthode compression fractale(IFS)**

Suit a ce qui est déjà traite dans le chapitre 2, les méthodes fractales reposant sur l'idée que l'on peut identifier des objets mathématiques de type fractal dans l'image. Ceux-ci sont composés de façon récursive des copies d'eux-mêmes. Le codage par fractales consiste à repérer des zones de l'image qui peuvent être déductibles par une transformation géométrique d'une autre zone de taille différente [5].

3.6.2.4.1 Principe de la compression IFS

La théorie des IFS est une extension de la géométrie classique, elle utilisie un ensemble de transformations affines pour exprimer des relations entre les parties d'une même image. En utilisant seulement ces relations, on peut exprimer et définir des images complexes (images, feuille, arbres,…etc.)

Le principe de la méthode de compression dite par IFS ou fractales est l'exploitation des Auto similarités des images naturelles. Ou l'auto similarité désigne la propriété qui fait que certains objets peuvent être construits à partir de parties d'eux même moyennant des transformations spécifique. En effet, au lieu de rechercher la corrélation entre les pixels adjacents, on s'intéresse à des corrélations entre parties plus au moins espacées dans l'image.

La problématique de la compression fractale est donc, pour une image donnée A, de trouver un IFS dont l'attracteur soit le plus proche possible de A.

L'algorithme de bloc consiste à partitionner d'abord A en blocs destination. Pour chacun de bloc, on cherche ensuite, toujours dans A, un bloc « source », qui par une transformation contractante simple (translation et rotation spatiale, changement d'échelle, modification du contraste et du niveau de gris moyen) donne une bonne approximation du bloc destination.

Quand tous les blocs destination ont traités, on a obtenu un ensemble de fonctions contractantes qui constitue un IFS dit « partitionné » dont l'attracteur approxime A.

On peut ainsi atteindre des taux de compression importants en conservant une bonne qualité d'image.

3.6.2.4.2 Le codage fractal par bloc (Transformation fractale)

La compression d'une image par fractales repose sur une transformation qui consiste à transformer l'image à l'aide d'un operateur 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 la figure 3.9)

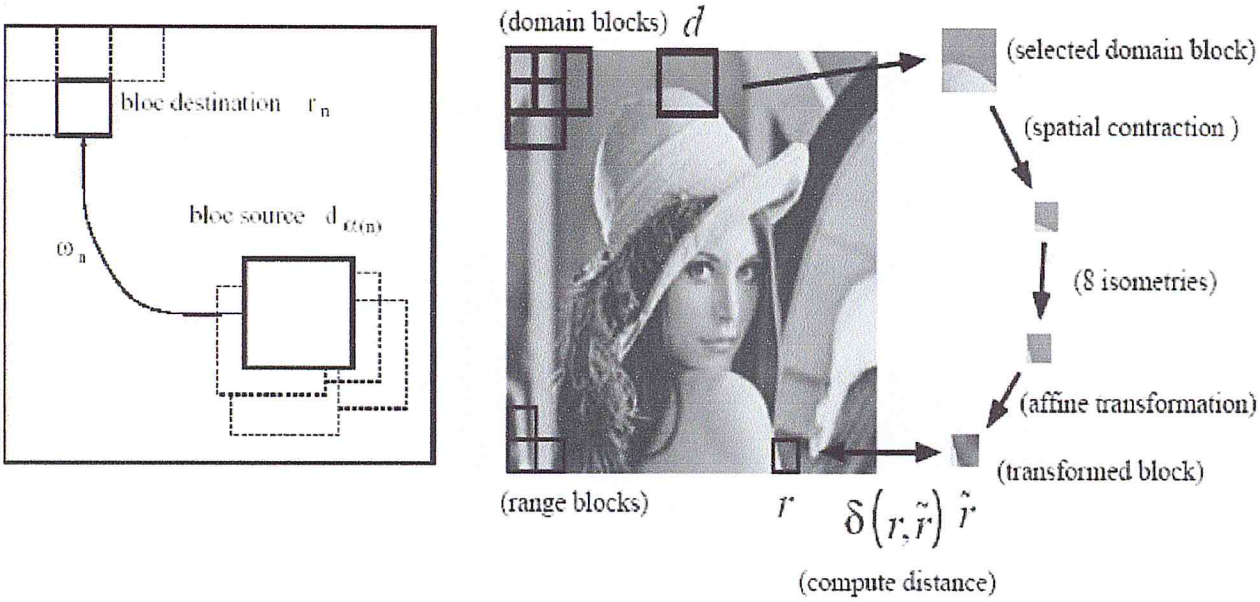


Fig 3.9 Principe de transformation fractale (Application sur l'image de Lena)[14]

L'image A est partitionnée en N blocs r_n appelés blocs destination (blocs range)
Elle s'écrit : $A = \bigcup_{i=1}^{n} r_i$ (3.1)
Cette partition du support de l'image en blocs destination est appelée partition. Chaque bloc destination est ensuite mis en correspondance avec un autre bloc transformé $w_i(d_i)$ lui « ressemblant » au sens d'une mesure d'erreur sur les niveaux de gris. Le bloc d_i appelé bloc source est recherché au travers d'une librairie composée de Q blocs appartenant à l'image. Les Q blocs ne forment pas nécessairement une partition de l'image, mais ils sont représentatifs de toute l'image.

La géométrie de la région d_i doit être proche de celle du bloc r_i , de façon à limiter les temps de calcul lors des comparaisons inter-blocs, puisque ceux-ci sont liés à la complexité de la transformation spatiale utilisée. Une seconde contrainte impose que les blocs source soient en moyenne de surface supérieure à celle des blocs destination.
La transformation W de l'image A est formulée à l'aide de l'équation suivant :

$$W(A) = \bigcup_{n=1}^{N} W_n(d_{\alpha n}) = \bigcup_{n=1}^{N} r_n \approx \bigcup_{n=1}^{N} \hat{r}_n = A \tag{3.2}$$

Où $\hat{r}$ est l'approximation du bloc destination r_i obtenue en transforment le bloc source d_i par w_i (l'opération permettant d'obtenir le bloc $\hat{r}$ à partir du bloc d_i est appelée opération de collage).
Chaque transformation w_i s'écrit sous la forme :

$$w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \tag{3.3}$$

Où

- s_i : contrôle le contraste.
- o_i : La luminance
- z : Niveau gris

Ce qui exige de calculer autant de fois les paramètres de la transformation affine W_i , s, o et d_{rms}

$$s = \frac{n \cdot \left(\sum_{i=1}^{n} d_i \cdot r_i \right) - \left(\sum_{i=1}^{n} d_i \right) \cdot \left(\sum_{i=1}^{n} r_i \right)}{n \cdot \sum_{i=1}^{n} d_i^2 - \left(\sum_{i=1}^{n} d_i \right)^2} \tag{3.4}$$

$$o = \frac{1}{n} \left(\sum_{i=1}^{n} r_i - s \sum_{i=1}^{n} d_i \right) \tag{3.5}$$

$$d_{rms} = E(D_i, R_i) = \frac{1}{n} \left[\sum_{i=1}^{n} r_i^2 + s \left(s \sum_{i=1}^{n} d_i^2 - 2 \sum_{i=1}^{n} d_i r_i + 2o \sum_{i=1}^{n} d_i \right) + o \left(o \cdot n - 2 \sum_{i=1}^{n} r_i \right) \right] \tag{3.6}$$

Pour que le codage 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 d_i. Lorsque ceux-ci sont recherchés dans toute l'image, la condition est généralement vérifiée. 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 les blocs source disponibles pour les recherches des similarités, et la ressemblance entre le nombre de blocs de la partition R et les blocs source d_i . Bien que dans la pratique, il est plus facile d'utiliser des blocs carrés, il est toujours possible d'utiliser d'autres formes telles que les rectangles, triangles, polygones,…etc.

3.6.2.4.2 Schéma général d'un codeur –décodeur fractal

On peut à présent donner le schéma général (Fig 3.10) et l'algorithme (pseudo-code) d'un codeur et décodeur fractal générique utilisant :

- Une partition R où les blocs destinations $r_n = \{r_i, i = 1, n\}$
- Un dictionnaire où les blocs sources $d_{\alpha(n)}$

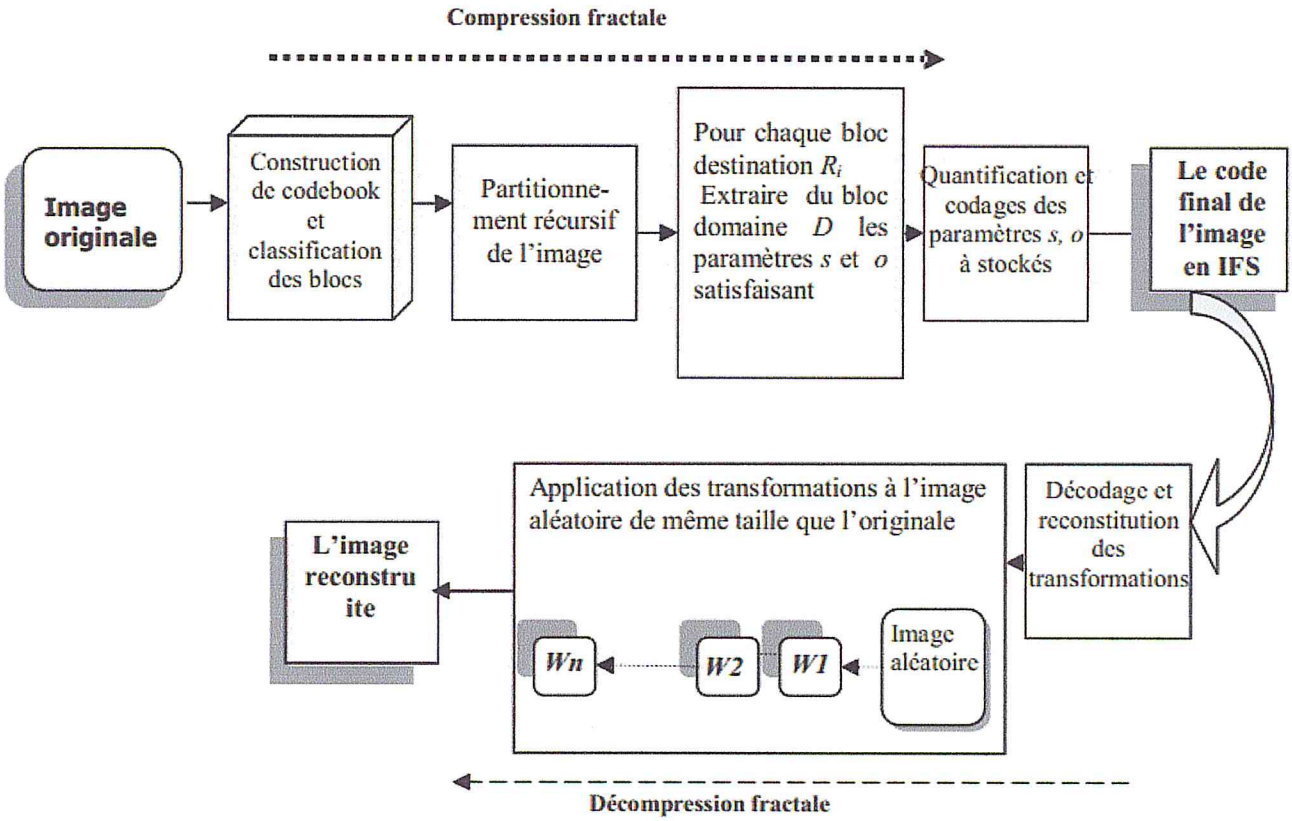


Fig 3.10 Schéma Général d'un codeur et décodeur fractale [22]

✓ *L'algorithme de compression*

Etape 1 : Lire l'image originale.

Etape 2 : Fixe le seuil (l'erreur RMS)

Etape 3 : Donner T_{min} , T_{max} (taille min et max du bloc à partitionner)

Etape 4 : Construction du dictionnaire des blocs source $d_{\alpha(n)}$ (domaines) et leurs classifications.

Etape 5 : Partitionnement (Quadtree, …) et récursif de l'image.

Etape 6 : Pour chaque bloc destination (range r_i $(i = 1,,,,,,,,,n)$)

- Classifier le bloc destination r_i
- Rechercher dans la classe du bloc destination le bloc source $d_{\alpha(n)}$ qui satisfait la distorsion calculée $<$ seul fixe RMS (l'erreur quadratique)
- Si oui stocker : les paramètres de la transformation W_i du bloc destination luminance o, contraste s, les cordonnées du bloc source ;
- Si non (aucun bloc ne satisfait la distorsion calculée $<$ seul RMS)
  - Si la taille du bloc $> T_{max}$ alors décomposer le bloc destination à nouveau
  - Si non la taille du bloc destination inferieur à la taille T_{min} en prend le bloc source (domaine) avec la plus faible erreur possible et ses paramètres sera stocké.

Etape 7 : Quantification des paramètres stockés et codage

Etape 8 : Fermer la bibliothèque domaine

Etape 9 : Terminer le processus de compression (résultats est un fichier IFS)

✓ *L'algorithme de décompression*

Etape 1 : Lire ou ouvrir IFS

Etape 2 : Décodage et reconstitution des transformations

Etape 3 : Charger une image A aléatoire de même taille que l'image originale à compresser

Etape 4 : Pour chaque transformation $W_i(i = 1..N)$

- Appliquer la transformation W_i à l'image A.
- Reconstruire l'image A
- Passer à la transformation suivante (aller à l'étape 4)

Etape 5 : Traitement et affichage de l'image résultante

3.7 Partitionnement de l'image

Différentes partitionnements de l'image ont été étudiés [22], [23] et [2] de façon à minimiser le nombre de transformations locales, et à coder le mieux des similarités inter-blocs dans la compression à bases des fractales. Nous distinguerons trois classes de partitionnement :

- Les partitionnements rigides
- Les partitionnements semi-rigides
- Les partitionnements souples

Nous limiterons ci-après à présenter quelques partitionnements selon une graduation allant du plus rigide au plus souple.

3.7.1 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 transformation contractante, 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 doivent être facile ;
- L'information nécessaire à son codage doit être minimale ;

Chacun des blocs destination r_i de la partition R est mis en correspondance avec une region source d_j de l'image, lui étant proche au sens des moindres carrés dans l'espace des niveaux de gris.

3.7.2 Partitionnement rigide(*Quadtree*)

Un partitionnement est qualifie de rigide s'il ne s'adapte pas à moindre cout aux formes des objets présents dans l'image.

✓ *Quadtree*

Le Quadtree ou arbre quaternaire est une structure initialement utilisé pour représente des images binaires, 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 utilise pour la représentation des images en niveau de gris

✓ *Phase de division*

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

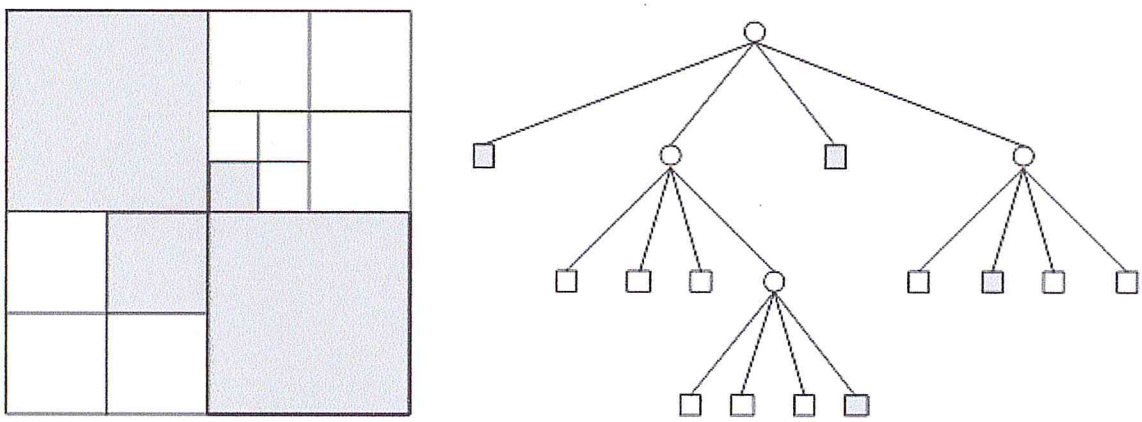


Fig 3.11 A gauche : principe de la division récursive d'une image .A droite : représentation arborescente du *Quatree* calcule sur une image de taille 8×8 pixeles, les cercles sont appelés *sommets* de l'arbre quaternaire et les carrés gris et blancs sont appelés *feuilles*.

- Ce type de partitionnement est certainement le plus répandu dans les programmes de compression par fractales, il est simple à utiliser et fournit des résultats correctes.

✓ *Phase de fusion*

La phase de fusion est utilisie en segmentation d'images. Elle consiste à regrouper les blocs adjacentes égaux selon le prédicat d'homogénéité considère. Il est à note qu'a l'issue de celle-ci, la structure de l'arbre quaternaire est perdue, l'intérêt de cette étapes est de réduire le nombre total des 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

✓ *Propriétés du Quadtree*

Le partitionnement obtenu est rigide puisqu'il est guide par le processus de découpage récursif en blocs carrés, il n'est pas adapté aux formes des objets présentes 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 autres partitionnements, aussi la construction récursive du Quadtree n'est pas une solution optimale en terme de temps de calculs puisque chaque pixel est visite un nombre de fois égal à sa profondeur finale dans l'arborescence.

✓ *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 r_n , de taille B^2, l'algorithme recherche un bloc source $d_{\alpha(n)}$ de taille D^2 avec $D = 2B$.

Le codage d'un bloc destination r se fait par recherche du bloc source d_n décimé qui permet de minimiser l'erreur r et l'approximation $\hat{r}$ donnée par :

$$\min d(r,\hat{r}) = \min_{\beta_1;\beta_2} d(r - \beta_1 b_1 - \beta_2 b_2) \qquad (3.6)$$

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 r dans le Quadtree démener inferieur a la profondeur maximale de ce dernier, alors le bloc destination est redivisé at la procédure de codage est relancée sur chacun des Quadtree sous blocs créés, si la distance minimale est inferieur au seuil fixé, le bloc r est codé par les coefficients β_1 et β_2 de la transformation massique retournant $\hat{r}$ et par la position dans le dictionnaire du bloc source d associé .Ces informations codent la transformation élémentaire w .

Dans noter application nous intéressons au Quadtree partitionnement (voir chapitre 4).

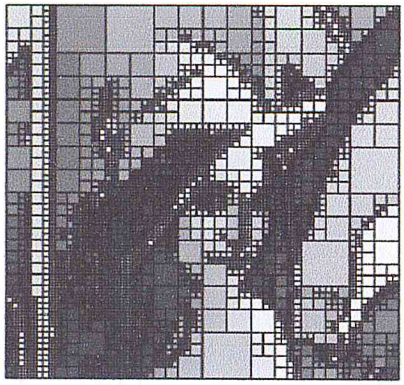


Fig 3.12 : Illustration d'un Partitionnement quadtree sur l'image Lena [5]

3.7.3 Partitionnement semi-rigide (H/V)

Le défaut de la partition Quadtree est qu'elle choisit l'ensemble des domaines *D* de façon indépendante du contenu. L'ensemble doit être rendu très grand pour qu'on puisse y trouvez une bonne correspondance pour un range donné. Une façon de remédier à cela, tout en augmentant la flexibilité de la partition en ranges, est d'utiliser une partition H/V

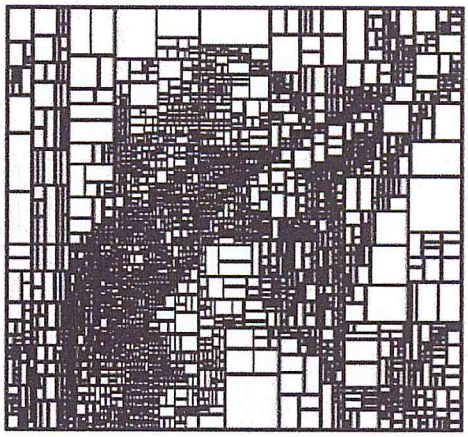


Fig 3.13 : Partitionnement H/V calculer sur l'image Lena [5]

✓ *Partitionnement horizontal/vertical*

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.

✓ *Intérêt du partitionnement H/V 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. Fisher a proposer deux méthodes de division qu'il utilisie en fonction de la nature du rectangle non homogène

1. Si le bloc est parcouru par une frontière oblique de l'image, la séparation est choisie de manière à ce que l'un des sous blocs soit parcouru diagonalement par la frontière et que s'il existe un autre sous bloc, celui-ci soit homogène.
2. Si le bloc inclut une frontière horizontale ou verticale, la séparation est choisie de manière à crée deux sous-bloc homogènes.

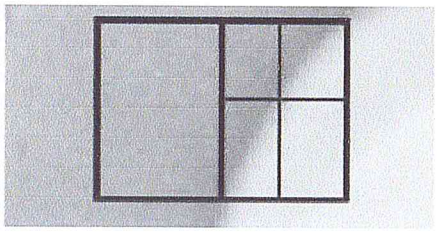


Fig 3.14 Partition H/V

méthode division (1)

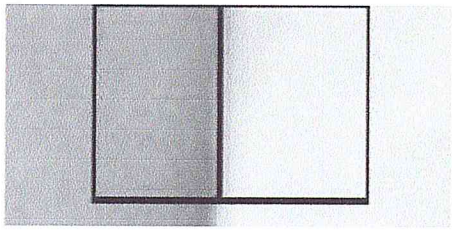


Fig 3.15 Partition H/V

méthode division (2)

3.7.4 Partitionnement souple (triangulaire)

Nous proposons ici un partitionnement de l'image en triangles de *Delaunay*. Celui-ci a l'avantage de ne pas être rigide (souples) puisqu'ils sont calculés sur un ensemble de points pouvant être positionnés à peu prés n'importe où sur le support de l'image, De plus, le diagramme est construit dans un environnement de type division et fusion, ce qui lui permet d'être adapte à l'image.

✓ Triangle de *Delaunay*

Un triangle $T = \{P_i, P_j, P_k\}$ où P_i, P_j, P_k sont dans P avec ($P = \{P_i \in \mathbb{R}^2, i = 1,,,,,,n\}$ les n points appelés germes ou sites) est un triangle de Delauanay si et seulement si : l'intérieur du cercle circonscrit à T ne contient aucun point de : $B(P_i, P_j, P_k) \cap (P - p_i, p_j, p_k) = \emptyset$ (3.7)

- Une propriété intéressante des triangles de *Delaunay* c'est qu'ils tous bornés

✓ *Codage d'une image*

L'objectif est d'avoir un maximum de ressemblance entre les différents blocs des partitions R *et* D. Pour cela, la triangulation de *Delaunay* est construite dans un environnement de Division et Fusion.

* *La phase de division* : consiste à ajouter des points sur barycentre des triangles dont le niveau de gris n'est pas homogène.
* *La phase de fusion* : consiste à supprimer les triangles inutiles, pour lesquels les voisins ont une valeur de triangles dans la partition R. Ceci aura pour effet de diminuer le temps de calcul pour coder l'image ainsi que d'augmenter le taux de compression.

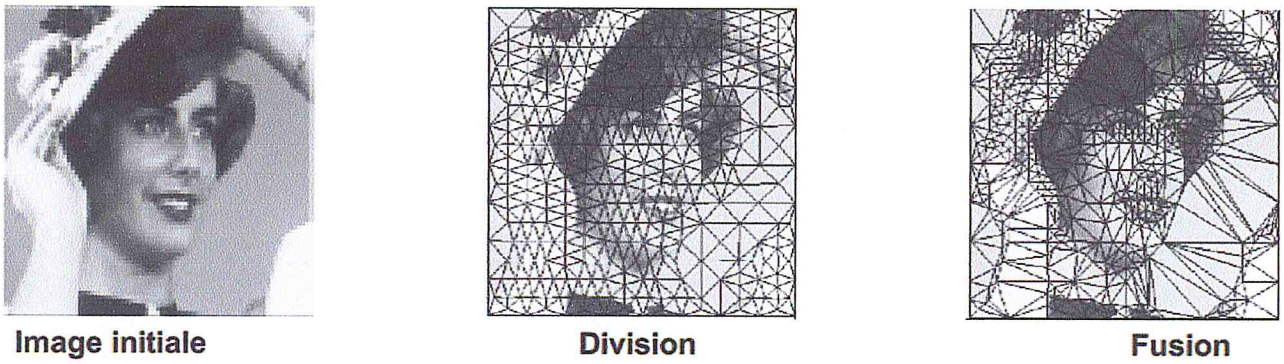


Fig 3.16 : Illustration de construction de l'initialisation de la triangulation de Delaunay à gauche est suivie de division et fusion à droite[2]

3.8 Mesures de performance de la compression d'image

3.8.1 Rapport et taux de compression

Le rapport de compression est l'une des caractéristique les plus importantes de toutes les méthodes de compression, il représente le rapport entre le nombre de bits de la forme canonique au nombre de bits après codage :

$$Rapport = CR = \frac{nombre\ de\ bits\ de\ l'image\ avant\ comperssion}{nombre\ de\ bits\ de\ l'image\ apres\ comperssion} = \frac{R_0}{R_1} \qquad (3.8)$$

3.8.2 Rapport signal à bruit

PSNR (*Peak Signal to Noise Ratio*) est une mesure de distorsion utilisée en image numérique, tout particulièrement en compression d'image. Il s'agit de quantifier la performance des codeurs en mesurant la qualité de reconstruction de l'image compressée par rapport à l'image originale.

Le PSNR est défini par :

$$PSNR = 10 \times \log_{10}\left(\frac{d^2}{EQM}\right)$$

où d est la dynamique du signal (la valeur maximum possible pour un pixel). Dans le cas standard d'une image où les composantes d'un pixel sont codées sur 8 bits, $d = 255$. EQM est l'erreur quadratique moyenne et est définie pour 2 images I_0 et I_r de taille $m \times n$ comme:

$$EQM = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\|I_0(i,j) - I_r(i,j)\|^2$$

Si le PSNR est utile pour mesurer la proximité de l'image compressée par rapport à l'original au niveau du signal, il ne prend pas en compte la qualité visuelle de reconstruction et ne peut être considéré comme une mesure objective de la qualité visuelle d'une image.

3.9 Conclusion

La compression des données est appelée a prendre un rôle encore plus important en raison du développement des réseaux et du multimédia.

Dans ce chapitre nous avons présente les différentes méthodes de codage de bases et de compression réversible où irréversible des images et nous avons introduit les différents modèles géométriques de partitionnement pour la compression par fractales à savoir le partitionnement Quadtree, le partitionnement H/V et le partitionnement triangulaire.

Chapitre 4
Mise en œuvre, tests et résultats

4.1 Introduction

- Ces derniers années les techniques de compression par fractales d'images(IFS) , ont gagnées plus d'intérêt en raison de leur capacité de réaliser des taux de compression très élevés tout en mettant à jour la qualité très bonne de l'image reconstruite , l'inconvénient principal de telles technique est le temps très élevé pour déterminer les codes d'IFS de l'image compressée .

- Pour résoudre ce problème, plusieurs techniques sont utilisées, dans ce chapitre nous allons présenter un code Matlab muni d'une interface qui permet d'appliquer la compression fractale sur des différents types des images pour augmenter les performances pour un meilleur rendement.

4.2 Environnement de travail

Pour réaliser ce projet nous avons utilisons :

4.2.1 Matériels

- le seule matérielle c'était un Pc portable a comme système :

  - CPU Intel core™ i3
  - Système d'exploitation : Windows 7

4.2.2 Langage programme

- Pour le langage programme nous avons choisi le Matlab R2012a avec l'outil GUI

4.2.2.1 Présentation de Matlab

MATLAB est un langage de haut niveau et un environnement interactif pour le calcul numérique, la visualisation et la programmation. Grâce à MATLAB, vous pouvez analyser des données, développer des algorithmes et créer des modèles et des applications. Le langage, les outils et les fonctions mathématiques intégrées vous permettent d'explorer diverses approches et d'arriver à une solution plus rapidement qu'en utilisant des feuilles de calcul ou des langages de programmation traditionnels, tels que C/C++ ou Java. Vous pouvez utiliser MATLAB pour une vaste gamme d'applications, en particulier le traitement du signal et des communications, le traitement des images et des vidéos, les systèmes de contrôle, le test et les mesures, la finance et la biologie. MATLAB est le langage de calcul scientifique pour plus d'un million d'ingénieurs et de scientifiques dans l'industrie et le monde académique.

4.2.2.2 Présentation du GUI

Le GUI (Graphical User Interface) permet de créer des interfaces où le créateur choisi plusieurs types d'objets (boutons, edit box, listbox.....) appelés handles. Ensuite, il doit réaliser la programmation pour obtenir l'interaction qu'il souhaite obtenir entre ces différents objets. La première étape est la création d'un GUI en tapant le mot clé GUIDE dans la fenêtre de commande Matlab ou bien entrez a partir de la barre des outils File →New → Gui→ Blank Gui →OK .Voici la fenêtre de départ :

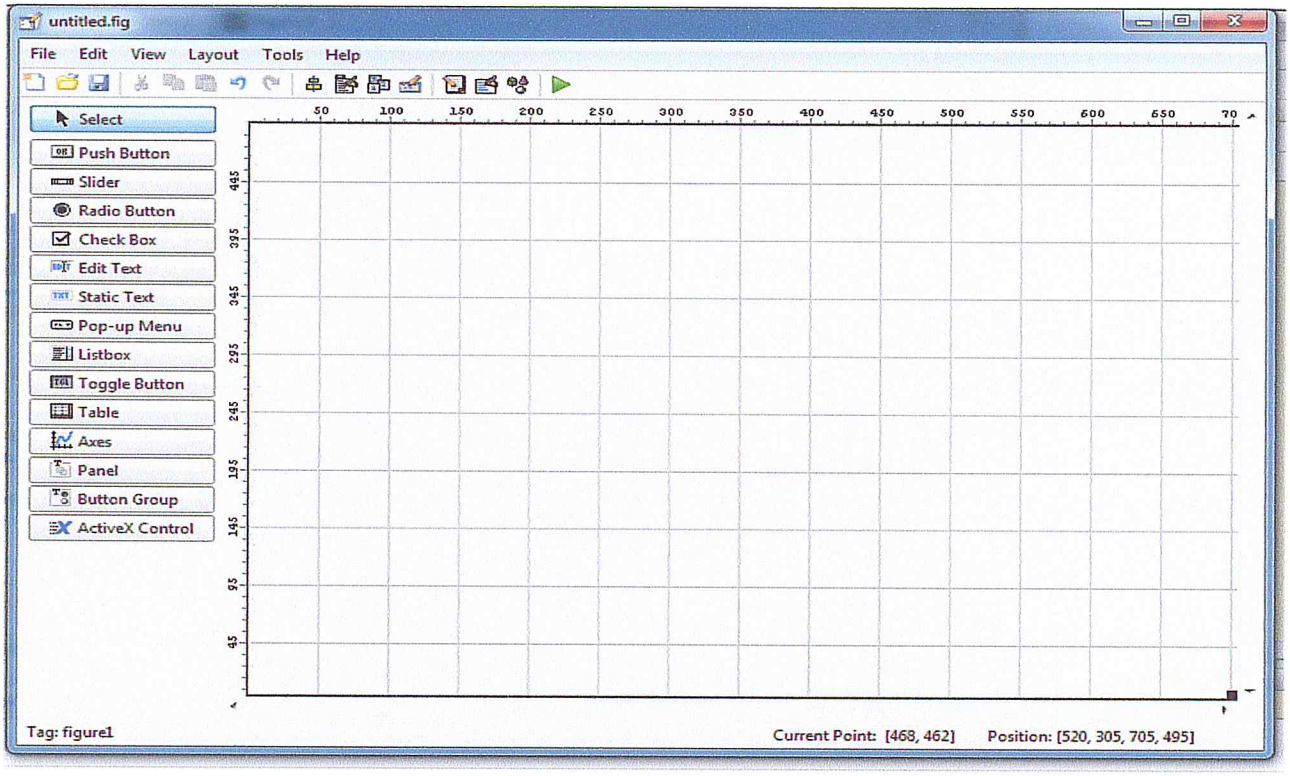
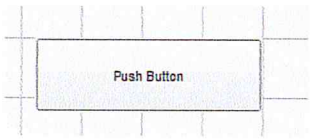


Fig 4.1 : interface principale de Matlab GUI

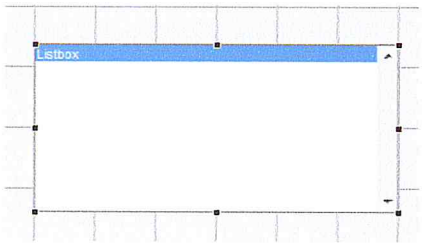
4.2.2.3 Présentation des différents objets

Pour crée un nouvel objet il suffit de glisse l'un des objets qui situe a cote gauche de l'écran de départ vers la grille

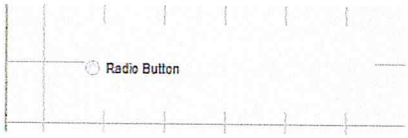
Dans nos interfaces, nous avons utilisé principalement 4 objets graphiques : –Radio button



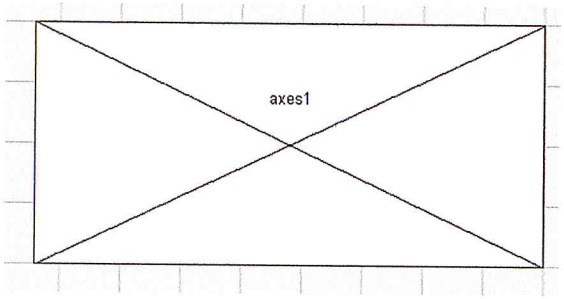
– la listbox



– le bouton (pushbutton)



–les axes



A la création du GUI (fichier d'extension .fig), un fichier d'extension .m est aussi généré et porte le même nom.

4.2.2.3 Les fonctions les plus utilisées

Pour chaque objet, dans le fichier .m est déclarée une fonction Callback précédée du nom de l'objet (modifiable grâce au champ Tag dans ses propriétés). Le code introduit dans cette fonction correspond à l'événement qui va se produire lors de l'action de la souris ou du curseur sur l'objet.

Par exemple :

```
function pushbutton2_Callback(hObject, eventdata, handles)
    returnValue    =    uigetdir(handles.ImageFolder,'Select
folder');
    if returnValue ~= 0
        handles.ImageFolder = returnValue;
        handles = LoadImageList(handles);
        set(handles.txtFolder,                    'string'
,handles.ImageFolder);
        guidata(hObject, handles);
        lastUsedImageFolder = handles.ImageFolder;
        save('mbprojet.mat', 'lastUsedImageFolder');
    end
    return
```

Le bouton s'appelle pushbutton2 et l'action sur ce bouton lance une nouvelle figure pour le choix du dossier input.

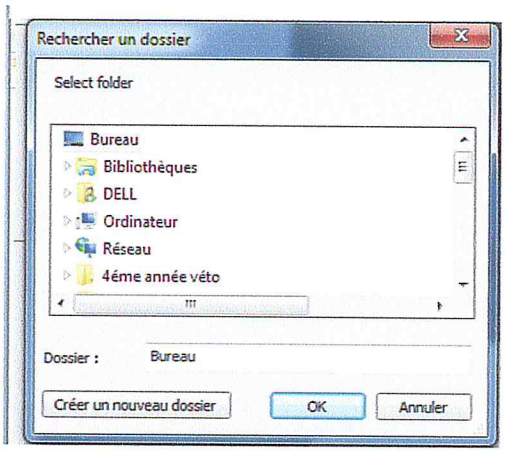


Fig 4.2 : interface utilise pour sélectionner le dossier source

Ensuite, pour manipuler ces objets, il existe 2 fonctions importantes :

- get qui permet de récupéré les valeurs.
- set qui permet de mettre des valeurs.

Par exemple , pour mettre une valeur qui se trouver dans la variable strResults dans une liste box qui a pour tag txtInfo , le code correspondant est :

```
set(handles.txtInfo, 'String', strResults);
```

Pour récupérer ce qu'il y a dans une liste box qui a pour tag 1stImageList :

```
selectedListboxItem = get(handles.lstImageList, 'value');
```

4.3 Interface générale

- Grace au logiciel Matlab et en particulier au GUI, nous avons réalise une interface composée de différents types d'objet :
  - Un botton pour le choix du dossier input.
  - Trois radios buttons pour le choix de méthode de compression.
  - Deux listes box pour les résultats numériques.
  - Trois axes pour l'affichage des images.
  - Un satitic text pour l'affichage de destination de l'image choisi.
  - Une barre des utiles.

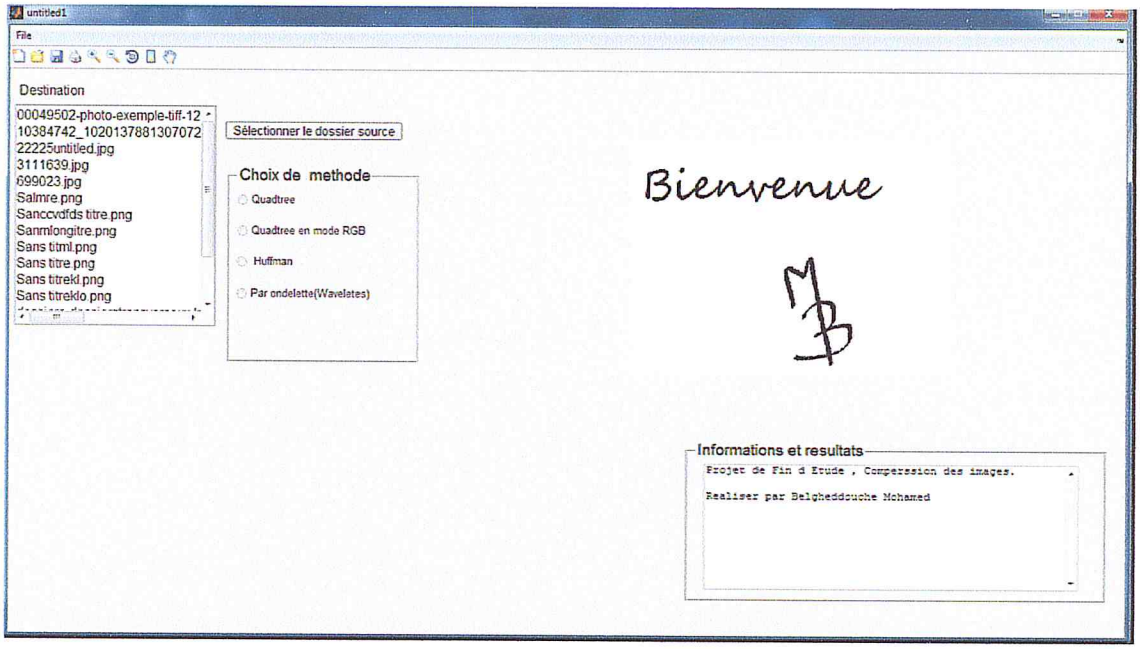


Fig 4.3 Interface principale de projet

Nous avons créé une interface qui permet de :

- Lire une image à partir d'un dossier source au choix.
- Choisir une méthode de compression
- Afficher l'image partitionnée et l'image compressée
- Visualiser les résultats calcules : CR, PSNR et le temps de compression et de décompression

4.3.1 Choix du dossier source

Quand l'utilisateur clique sur le bouton Sélectionner le dossier input, une petite figure centrée qui va donner la main a l'utilisateur pour le choix de son dossier cible (voir 4.2.2.3).

4.3.2 Lecteur et l'affichage de l'image sélectionné

Après le choix de dossier source tout les images inclue apparaitre dans la liste box1, n'importe quelle image, tous les formats d'image sont possibles (exemple : .jpg , .bmp) , et le -static text- afficher la chemin de dossier :

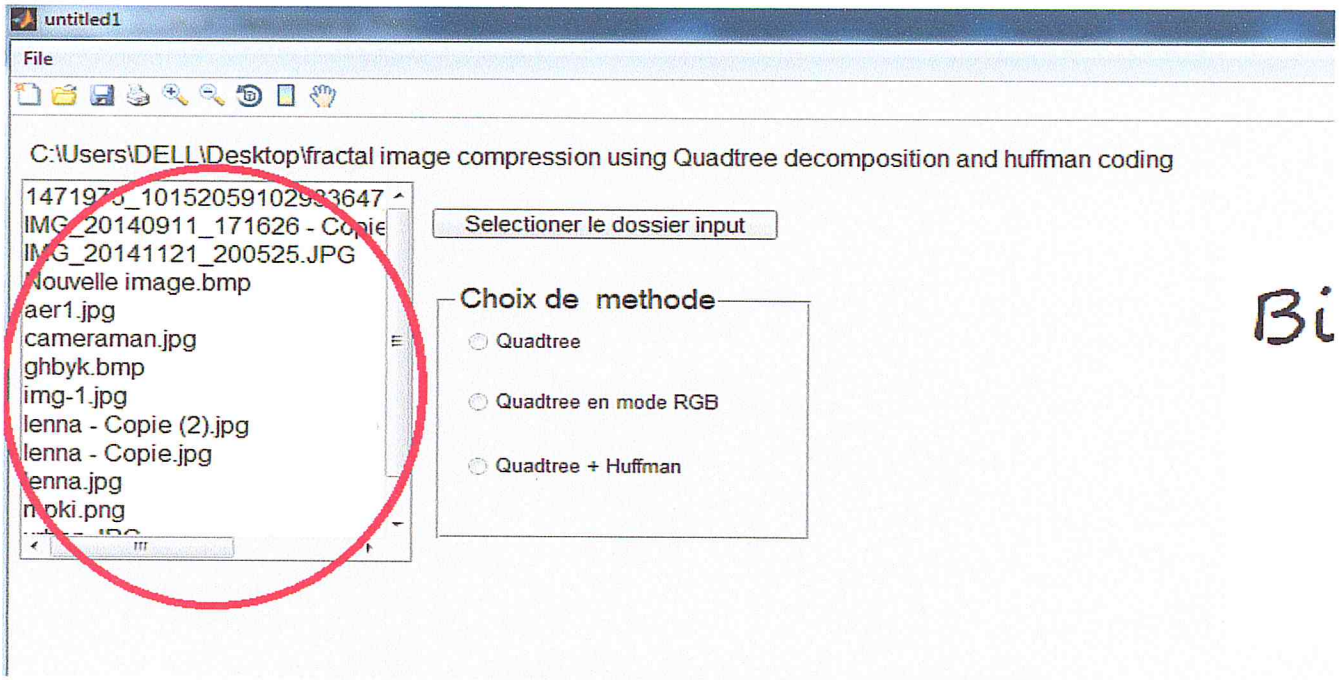


Fig 4.4 Choix de l'image à partir de l'interface générale

L'utilisateur a le choix de utiliser une seule image de la liste box pour applique l'un de trois méthodes, lorsque l'utilisateur sélectionner une image la fonction récupérés : le type, le non, le chemine de l'image et le rendre comme variable globale pour qu'il peut être utilisé pour d'autre fonctions, et sera afficher dans axesImage –(1)-

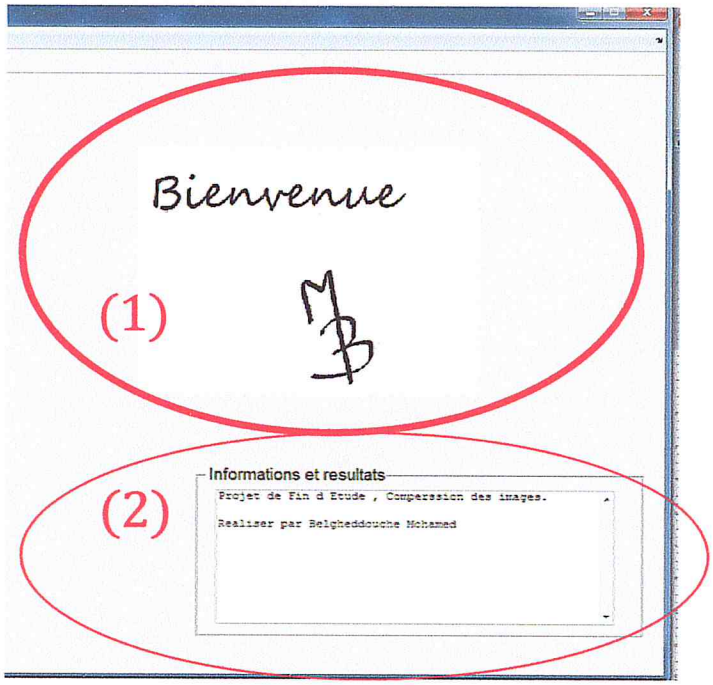


Fig 4.5 Affichage de image sélectionné

Et les propriétés de l'image (chemin, format, la taille…) apparaîtra sur l'infotxt- (2)-

4.3.3 Choix de la méthode de compression

La prochaine et l'important étape est le choix de méthode de compression, dans ce projet nous avons propose quatre méthodes de compression selon le format de l'image et ainsi le choix de l'utilisateur lui-même

Le choix fait à partir de panneaux de choix de méthode

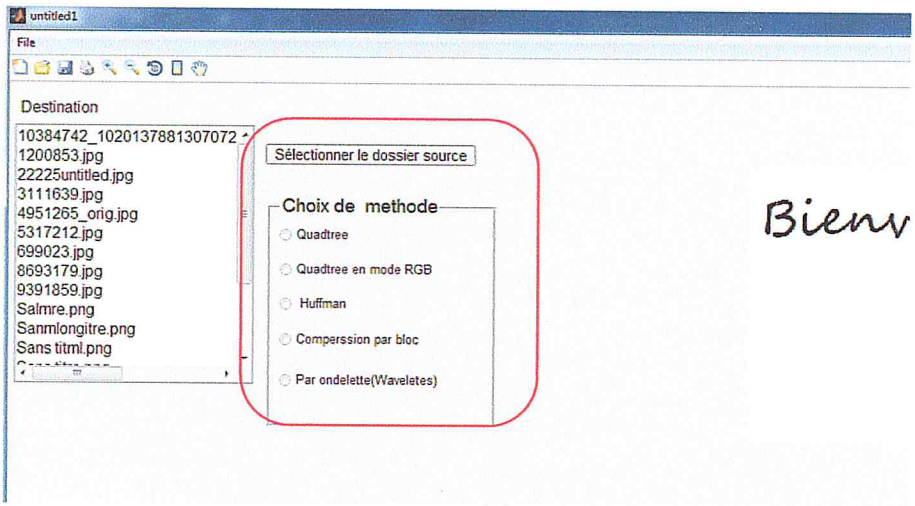


Fig 4.6 Choix de méthode de compression

4.3.4 Visualisation des résultats

La dernière étape après le choix de méthode et différente calcule selon chaque méthode, les résultats afficher contient : une image compressé sera afficher dans axes2, une image décompresse (finale) sera afficher dans l'axes3, et les résultats numérique comme le temps de compression, le taux de compression et le PSNR sera afficher dans l'infotxt.

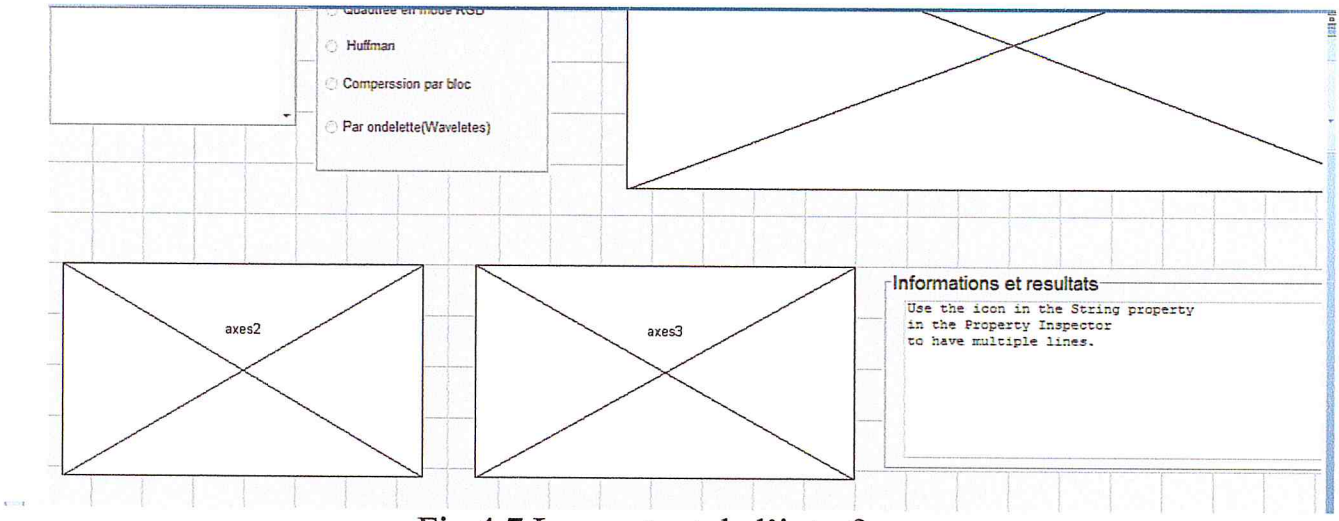


Fig 4.7 Les output de l'interface

4.4 Applications et résultats

Dans notre application nous avons choisir cinq différents images et les tester sur chaque méthode :

4.4.1 Quadtree

Cette méthode consiste à diviser l'image d'origine en blocs non chevauchent selon une valeur de seuil de 0,2 jusqu'a 0,5 , $mindim = 2$ et $maxdim = 64$, et après on utilise juste la méthode Quadtree pour code et décodée l'image en convertir l'image vers un matrice pour facilité les calculer. Les résultats suivants sont obtenus en utilisant la méthode Quadtree avec un seuil de 0.2 .

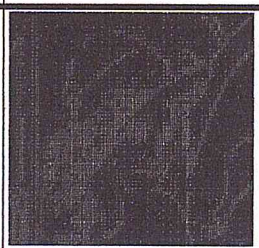
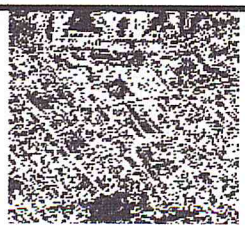
Non et type d'image	Image original	Image partitionnée	Image décodé
Lena.jpg			
Image satellite.jpg			

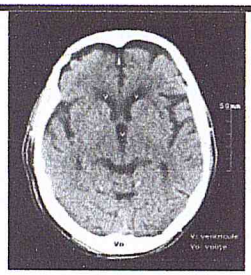
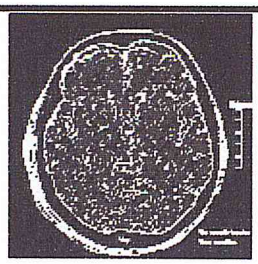
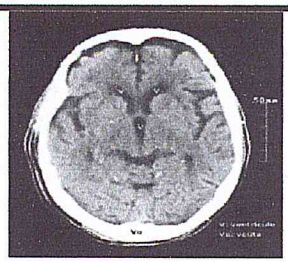
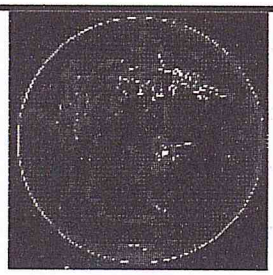
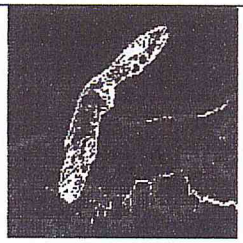
Image médical.jpg			
Mars.png			
hawk.bmp			

Tableau 1 : Images partitionnées et compressées par la méthode Quadtree

Non et type d'image	Temps de compression (sec)	Taux de compression	PSNR (DB)
Lena.jpg	5.81	2.37	36.11
Image satellite.jpg	14.73	2.84	36.87
Image médical.jpg	8.33	1.33	36.82
Mars.png	2.8	1.5	38.28
hawk.bmp	3.93	19.12	37.11

Tableau 2 : les résultats obtenu en utilisant la méthode Quadtree RGB

4.4.2 Quadtree en mode RGB

Cette méthode et presque comme la méthode précédent dans leur principe mais ce méthode utilise un matrice de taille M-par-N-par-3, par conter la méthode Quadtree classique utilise une matrice M-par-N

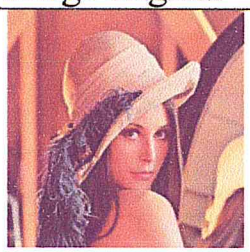
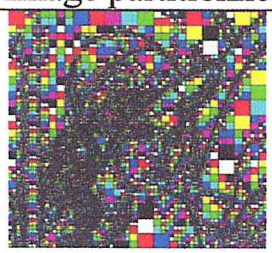
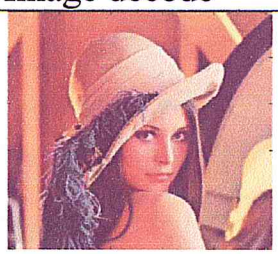
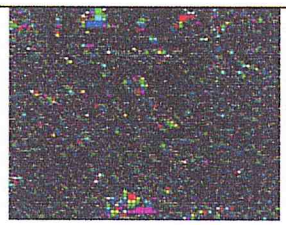
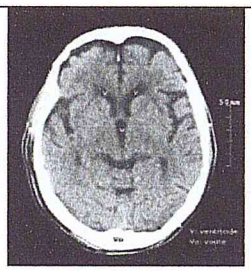
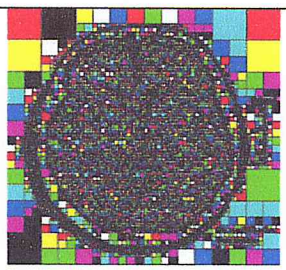
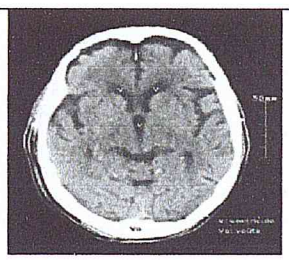
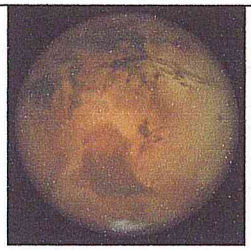
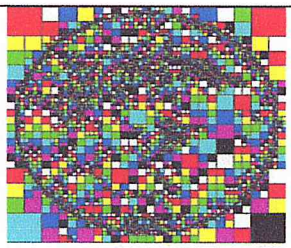
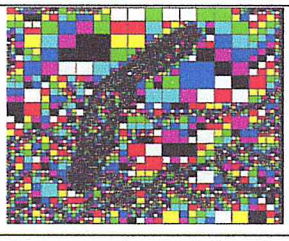
	Image original	Image partitionnée	Image décodé
Lena.jpg			
Image satellite.jpg			
Image médical.jpg			
Mars.png			
Hawk.bmp			

Tableau 3 : Images partitionnées et compressées par la méthode Quadtree RGB

Non et type d'image	Temps de compression (sec)	Taux de compression	PSNR (DB)
Lena.jpg	41.37	9.36	36.11
Image satellite.jpg	106.58	8.25	36.87
Image médical.jpg	45.64	11.29	36.82
Mars.png	20.07	3.82	38.28
hawk.bmp	24.43	0.99	37.11

Tableau 4 : les résultats obtenu en utilisant la méthode Quadtree RGB

4.4.3 La méthode de *Huffman*

En utilise dans le début une décomposition de Quadtree, puis en peut applique le codage et décodage de *Huffman* qui utilise un arbre de symbole et calculer la probabilité d'apparition de chaque symbole (la méthode est bien détaille dans le chapitre 3), l'image décodé est une image de niveau gris.

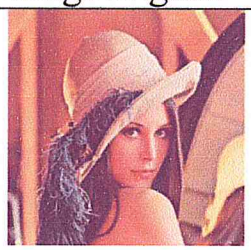
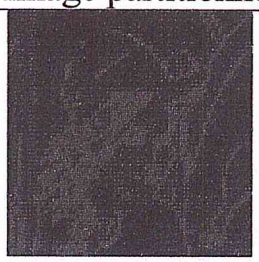
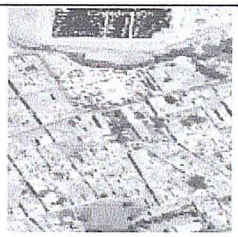
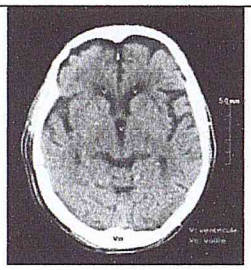
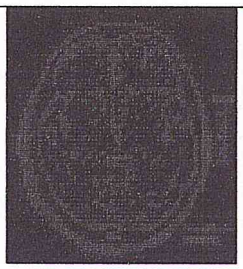
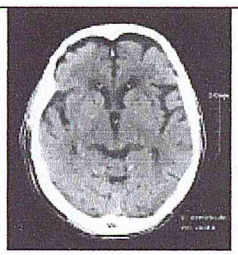
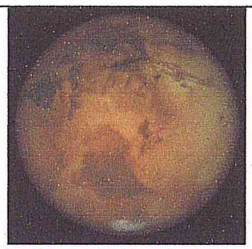
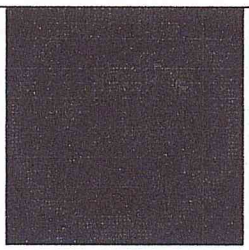
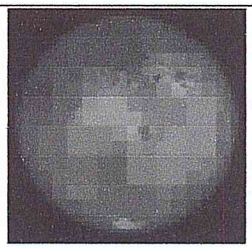
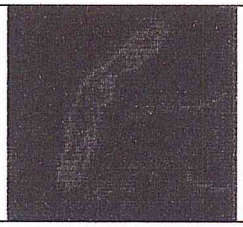
Non et type d'image	Image original	Image partitionnée	Image décodé
Lena.jpg			
Image satellite.jpg			
Image médical.jpg			
Mars.png			
Hawk.bmp			

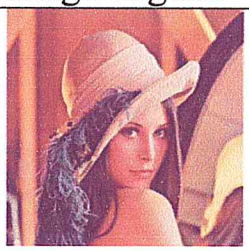
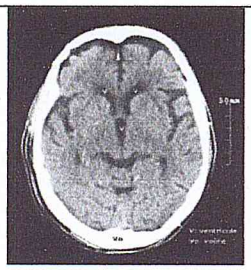
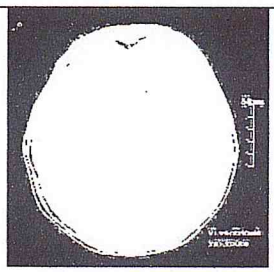
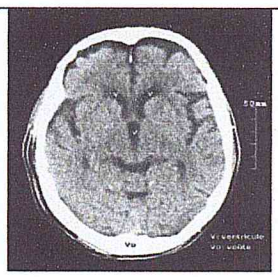
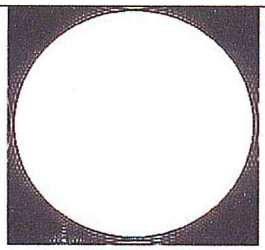
Tableau 5 : Images partitionnées et compressées par la méthode Huffman

Non et type d'image	Temps de compression (sec)	Taux de compression	PSNR (DB)
Lena.jpg	107.77	3.73	27.10
Image satellite.jpg	296.70	1.54	23.10
Image médical.jpg	123.76	3.43	24.28
Mars.png	12.94	25.68	30.57
hawk.bmp	54.74	7.72	27.71

Tableau 6 : les résultats obtenu en utilisant la méthode Huffman

4.4.4 Compression par ondelettes

Nous avons utilisé seulement pour comparer les résultats obtenus précédemment, et cela parce qu'il utilise une façon différente aux autres méthodes utilisées précédemment. Nous avons choisi 'haar' pour décomposer l'image.

Non et type d'image	Image original	Image compressé	Image décompressé
Lena.jpg			
Image satellite.jpg			
Image médical.jpg			
Mars.png			

Hawk.bmp			

Tableau 7 : Images partitionnées et compressées par la méthode compression par Ondelettes

Non et type d'image	Temps de compression (sec)	Taux de compression	PSNR (DB)
Lena.jpg	19.78	0.6	40.09
Image satellite.jpg	19.53	0.21	71.56
Image médical.jpg	20.97	0.31	66
Mars.png	45.96	0.73	45.71
hawk.bmp	19.12	0.67	43.08

Tableau 8 : les résultats obtenu en utilisant la méthode de compression par Ondelettes

- Pour faire une comparaison entre les quatre méthodes on grouper les données de chaque méthode de chaque image pour une seule valeur à la fois

	Mars	Hawk	Image M	Image S	Lena
Quadtree	2.8	3.93	8.33	14.73	5.81
Quadtree RGB	20.07	24.43	45.64	106.58	41.37
Huffman	54.74	12.94	123.76	296.7	107.77
Ondelette	45.96	19.12	20.97	19.53	19.78

Tableau 9 : Le temps de compression

	Mars	Hawk	Image M	Image S	Lena
Quadtree	1.5	19.12	1.33	2.84	2.37
Quadtree RGB	3.82	1.99	11.29	8.25	9.63
Huffman	25.68	7.72	3.43	1.54	3.73
Ondelette	0.73	0.67	0.31	0.21	0.6

Tableau 10 : Le taux de compression

	Mars	Hawk	Image M	Image S	Lena
Quadtree	38.28	37.11	36.82	36.87	36.11
Quadtree RGB	38.28	37.11	36.82	36.87	36.11
Huffman	30.57	27.71	24.28	23.10	27.10
Ondelette	45.71	43.08	66	71.56	40.09

Tableau 11 : Le PSNR

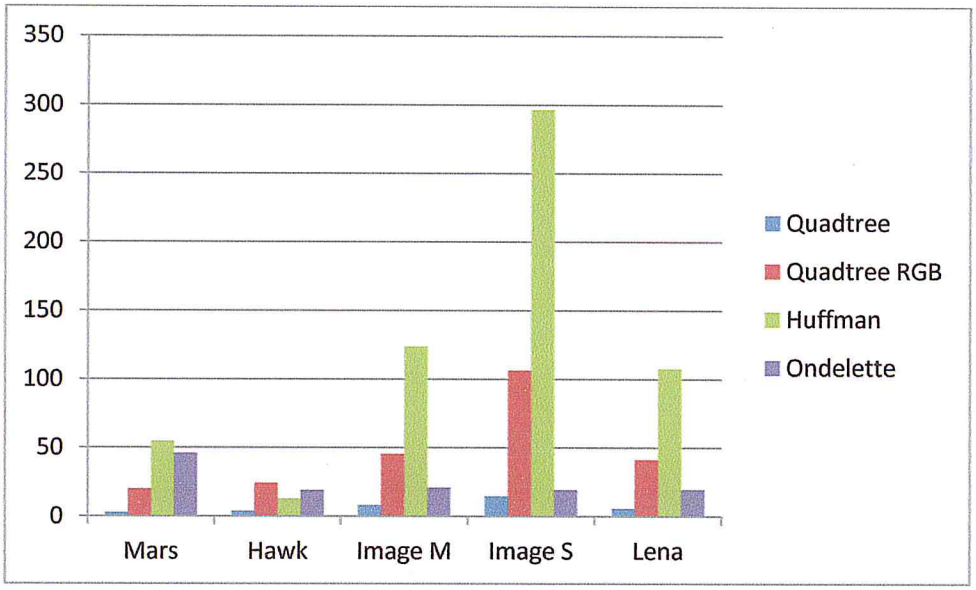


Fig 4.8 Comparaison entre le temps de compression

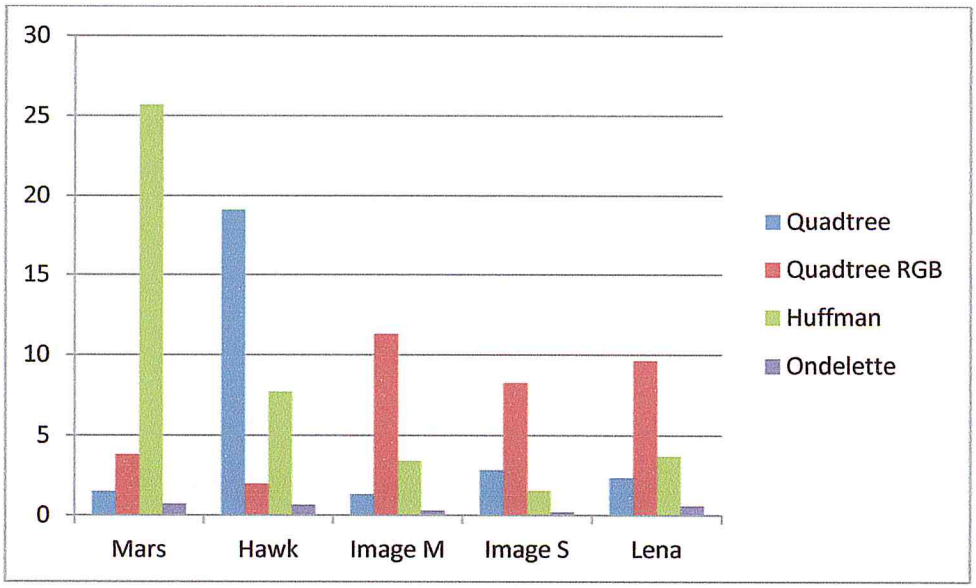


Fig 4.9 Comparaison entre le taux de compression

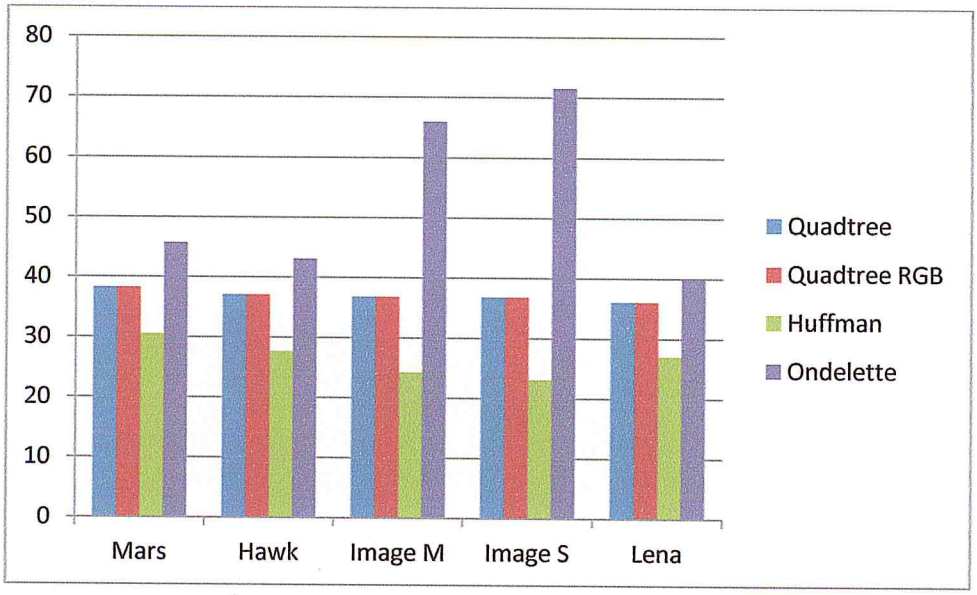


Fig 4.10 Comparaison entre le PSNR

4.5 Discussion

Nous avons présenté une application de quatre méthodes de compression d'images tels que, deux méthodes utilisant la compression fractales et deux méthode différente pour que nous peut comparer nos résultats, en se basant sur les résultats obtenus et les figures 4.8, 4.9, 4.10 on remarqué que :

- Pour le temps de compression, la méthode Quadtree est le plus rapide puisque il utilise un matrice de deux dimension $M \times N$, et utlise le codage de blocs de pixels, par contre, la méthode *Huffman* qui est le plus lent (jusqu'à 296,70 sec pour l'image satellite) elle utilise un codage individuel des pixels .Mais si on compare les méthodes fractales avec la méthode ondelette, on remarque que la compression fractales prend plus de temps pour compressé l'image.
- Pour le Taux de compression, contrairement a le temps de compression la méthode de Huffman présente la meilleure valeur de CR (jusqu'à 25.68 pour l'mage mars) suivi par la méthode Quadtree RGB, au la méthode de compression par ondelette représente le bas niveau de CR. On peut conclure que les méthodes

fractales sont les meilleures méthodes concernant le rendement de taux de compression.

- Pour le PSNR, malgré que la méthode de compression par ondelette représenté les hautes valeurs de PSNR, ca ne snifiez pas qu'elle est la meilleur méthode de compression, puisque le PSNR ne peut être considéré comme une mesure objective de la qualité visuelle d'une image car il ne prend pas en compte la qualité visuelle de reconstruction (on peut remarque la différence utilisant le tableau 1 et le tableau 7)

4.6 Avantages et inconvénients de la compression fractale

- A partir de ces résultats on peut conclure quelque avantages et incontinents de la compression fractale

4.6.1 Avantages

Les principaux avantages de cette technique fractale sont

- Des taux de compression élevés.
- Une représentation indépendante de l'échelle.
- Une procédure de décompression particulièrement simple, qui consiste simplement à interpréter la représentation à une échelle donnée.

4.6.2 Inconvénients

- La compression reste lente malgré toutes les améliorations (couteuse en temps)
- Elle peut nécessaire un hardware spécialisé.
- Ce type de compression n'a pas encore fait l'objet d'une norme, son utilisation n'est donc pas encore totalement démocratisée.

Conclusion Générale

Par ce mémoire, nous avons retracé l'histoire de la géométrie fractale, nous avons rappelé la théorie qui permet de générer des fractales selon la technique des Iterated Functions System (IFS), et nous avons applique cette théorie sur la compression des images en utilisant un code Matlab pour obtenir un meilleur rendement.

Notre expérimentation a été effectuée sur des images de différents types et tailles. Les tests ont été effectués sur chaque image pour chaque méthode. Nous avons fait un prélèvement des temps de compression, taux de compression et le PSNR sur chaque méthode. Nous avons remarqué des différences entre les résultats obtenus. Ces différences sont dues au système informatique utilisé.

Nos résultats montrent que la compression fractale est une méthode extrêmement performante pour stocker le maximum d'information dans le minimum de place. Ceci est explique par la nature du code IFS qui donne des taux de compression performants. Cependant la lenteur de cette méthode et sa complexité en termes de conception et de réalisation ne permet pas assez facilement de mener une telle amélioration.

Pour attendre des meilleures performances, mous proposons comme perspectives à notre travail d'utiliser des nouvelles méthodes de compression de l'image pour gagner plus de temps ou bien de utilise les L-systèmes pour décomposé l'image.

La géométrie fractale est une vieille idée qui a trouvé une nouvelle application avec l'avènement des techniques d'imagerie de l'ordinateur. Son acceptation, a donné naissance à un grand nombre de recherches et a fourni un nouvel outil pour observer la nature à travers une perspective différente. Nous devons faire attention à nous assurer que nos résultats sont en fait valables. Nous devons également commencer la fusion des

nombreuses techniques qui ont été développées dans le but de contrôler la croissance de ce concept et d'atteindre une véritable acceptation scientifique. Sans cette acceptation de la théorie sera critiqué (valablement) comme une idée imprécise et non prouvée. Ce serait un événement malheureux en raison du potentiel que possède la géométrie fractale. Il est l'espoir de l'auteur que ce travail a mis en lumière le sujet de la géométrie fractale et qu'il aidera les autres dans leur recherche. Le but et l'essence de la géométrie fractale est basée sur des concepts simples. Le lecteur ne doit pas être intimidé par la littérature actuelle et devrait conserver son point de vue avec une légère dose de scepticisme. Il ne doit pas être aveuglé par le scepticisme bien que le potentiel de la géométrie fractale n'a pas encore été réalisé. En dernière analyse, nous nous attendons à ce que même le lecteur sceptique va découvrir la beauté mathématique et la puissance applicative que la géométrie fractale possède.

Références Bibliographiques

[1] J. LAJOIE, La Géométrie Fractale, exigence partielle de la maitrise en mathématiques et informatique appliquées, Université Du Québec, Juin 2006

[2] A.Francoise , « Mise en ligne d'images 3D traitement » – Mémoire de Magister en informatique – Université Du Québec, Juin 2001.

[3] Benoit B. Mandelbrot, The Fractal Geometry of Nature, W. H.Freeman and Company, 1983.

[4] M.LEHAMEL–Segmentation d'images texturées à partir des attributs fractales – Mémoire de Magister en informatique – Université Mouloud Mameri de Tizi Ouzou – 2011.

[5] D.Zeroual–Implémentation d'un environnement parallèle pour la compression d'images a l'aide des fractales– Mémoire de Magister en informatique – Université de Batna–2006

[6] M.Aitouche –Dynamique Chaotique et Analyse Fractale en Géophysique – Thèse de Doctorat en informatique – Université M'hamed Bougara de Boumerdes – Juin 2007

[7] C.Stadler, « Les algorithmes de compression sans perte » – Mémoire de Magister en informatique –Ecole spécialisée de suisse , janvier 2005.

[8] J. Blue Kriska- Construction of Fractale Objects with Iterated Function Systemes- University of North Carolina at Chapel Hill – 2013.

[9] Vincent Garant-Pelletier- Ensembles De Mandelbrot Et De Julia Remplis Classiques Généralisées Aux Espaces Multicomplexes Et Théorème De Fatou-Julia Généralisé, exigence partielle de la maitrise en mathématiques et informatique appliquées ,- Université Du Québec –Mars 2011.

[10] Hybrid Fractal Image Compression Using Quadtree Décomposition with Huffman Coding-Shweta Pandey, Megha Seth-Département of Information Technology, Rungta College of Engineering & Technology, Kohka, Bhilai, CSVTU University, Chhattisgarh, India-juin 2014

[11] F.Charot, « architectures Parallèle Spécialisés pour le traitement d'image », INIRIA, N^0 1978, IRISA Campus Université de Beaulieu, France, Avril 1993.

[12] K.Chakib, « La Compression des images Fixes par les Approximation Fractales Basée sur la triangulation de Delaunay et la quantification Vectorielle »-Mémoire de magister en informatique- Université M'hamed Bougara de Boumerdes – 1999.

[13] Brault, Dougherty, « Les formats de compression d'image »- Mémoire de magister en informatique industrielle -Institut Universitaire de technologie de tours, 2004.

[14] G.Mercier, C.Roux et G.Martineau, « Technologies du Multimédia », ENST Bertange dpt ITI, BP 832,F-29280 Brest ,France .janvier 2003.

[15] B. Mandelbrot, Les objets fractals : forme, hasard et dimension, Flammarion, 1995.

[16] Didier Gonze –Fractales, theory and applications – Mémoire de magister en et informatique appliquées - Université Libre de Bruxelles, Belgique,2011.

[17] F. Davoine. Compression d'images par fractales basée sur la triangulation de Delaunay-Mémoire de magister-Institut National Polytechnique de Grenoble - INPG, 1995.

[18] Ali-Pacha et N Hadj-Said, « Compression des images fixes par Fractales : Partitionnement Quadtree »-Mémoire de magister en informatique- Universite de Mentouri Constantine, novembre 2000.

[19] P.Cédric et F.Oliver, « La compression d'image par ondelettes » Lycée Pothier MPSI 3, T.I.P.E d'informatique 1998.

[20] Y.Fisher, D.Rogovin et T.P .Dhen, « A Comparaison of fractal Methods with DCT and Wavelets » In Neural and stochastic methods in image and signal processing III, Volume Proc .SPIE 2304-16, San Diego,2001.

[21] F.Davoine, « Compression d'images par Fractales Basé sur la Triangulation de Delaunay », These L'INPG, Institut Nationel Polytechnique de Grembole , Decembre 1995.