Fractal approximation of curves and surfaces

Eric Guérin

Université Claude Bernard Lyon 1 – UFR informatique **École doctorale EDIIS**

Approximation fractale de courbes et de surfaces

THÈSE

présentée et soutenue publiquement le 18 décembre 2002

pour l'obtention du

Doctorat de l'université Claude Bernard — Lyon 1

(spécialité informatique)

par

Éric Guérin

Composition du jury

Rapporteurs :	Jean-Marc Chassery	Université Joseph Fourier, Grenoble
	Jacques Lévy-Véhel	INRIA Rocquencourt
	Marc Neveu	Université de Bourgogne, Dijon
Examinateurs :	Albert Cohen	Université Pierre et Marie Curie, Paris
	Atilla Baskurt	Université Claude Bernard, Lyon
	Éric Tosan	Université Claude Bernard, Lyon
	Denis Vandorpe	Université Claude Bernard, Lyon

Numéro d'ordre : 252 – 2002

Laboratoire d'Informatique Graphique, Image et Modélisation — EA 1899

Remerciements

Je tiens tout d'abord à remercier Atilla Baskurt, ainsi qu'Éric Tosan. C'est leur soutien scientifique permanent qui m'a permis d'arriver au bout de ce travail. Leurs compétences complémentaires ont largement contribué à l'avancement de mes travaux. Je tiens aussi à souligner leurs qualités relationnelles qui ont rendu ces quatre années agréables.

Merci à Bernard Péroche et à Denis Vandorpe de m'avoir accueilli au sein du LIGIM. Sans eux, je ne serais évidemment pas ici.

Merci aux rapporteurs, Jean-Marc Chassery, Jacques Lévy-Véhel et Marc Neveu, d'avoir pris le temps de lire en détail ce manuscrit. Je suis conscient d'avoir un style qui n'est pas parmi les plus faciles à lire... Merci pour les remarques qu'ils ont faites et qui ont grandement contribué à l'amélioration de la compréhension du document.

Merci à Albert Cohen d'avoir bien voulu compter parmi les membres du jury. C'est un honneur pour moi de lui présenter mes travaux.

Merci à Céline Durou et Michèle Guérin, pour leurs relectures multiples et la chasse qu'elles ont faite aux fautes d'orthographe les plus sournoises, ainsi que pour leur soutien moral.

Merci à Jean-Michel Jolion, ainsi que Christelle Debize, pour leur aide.

Merci à toutes les personnes qui ont pu m'aider de près ou de loin dans mon travail. Parmi elles, toutes les personnes du LIGIM, du LISI ou du PRISMA, thésards, maîtres de conférences et professeurs. Merci aux secrétaires, Isabelle Carteaux, Magali Griffon et Marie-Luce Jobard-Blanc.

À mon père,

Table des matières

Table des figures

Liste des tableaux

« Die ganze Zahl schuf der liebe Gott, alles Übrige ist Menschenwerk. »

« Dieu nous a donné les entiers naturels, tout le reste n'est qu'invention de l'homme. »

Kronecker

« Ein Mathematiker, der nicht irgendwie ein Dichter ist, wird nie ein vollkommener Mathematiker sein. »

« Un mathématicien qui n'est pas quelque part poête, ne sera jamais un mathématicien parfait. »

Weierstrass

Introduction

Beaucoup de phénomènes de la nature ne peuvent être décrits par des formes lisses. Un des premiers à observer ceci fut le botaniste écossais Brown (1773 - 1858). En 1810, il découvre que le mouvement des particules de pollen en suspension dans l'eau a des propriétés pour le moins étranges. Il s'agit de ce qu'on appellera plus tard un mouvement *brownien*, mais Brown ne le sait pas encore... Ce n'est qu'à la fin du XIXème siècle que les mathématiciens s'intéressent aux courbes irrégulières. Elles font l'objet de querelles entre Weierstrass et Kronecker, ce dernier n'acceptant pas la théorie du premier. À l'époque, Weierstrass invente des courbes continues mais nulle part dérivables pour démontrer l'indépendance des deux concepts. Tous les points sont donc reliés entre eux mais pourtant on ne peut définir aucune direction. Ces travaux ont de forts liens avec le concept fractal.

Sans tous ces efforts, il aurait été impossible d'inventer une théorie fractale. Il aura fallu en effet pratiquement deux siècles pour enfin arriver à définir et formaliser précisément les phénomènes fractals. Dans les années 1970, Mandelbrot observe les côtes bretonnes dont il veut mesurer la longueur. C'est avec étonnement qu'il constate que quelle que soit l'échelle à laquelle il regarde cette côte, celle-ci a le même aspect irrégulier[Man82]. Cependant, la longueur qu'il mesure augmente alors qu'il augmente la précision de ses relevés. Un coefficient semble régir cette loi : *la dimension fractale*. Ces observations ont ensuite été généralisées à de nombreux phénomènes naturels ainsi qu'à des formes naturelles. Depuis ces trente dernières années, on a ainsi vu se développer côte à côte des outils pour l'analyse fractale ainsi que des modèles fractals. À la suite de travaux de Hutchinson, Barnsley propose en 1988 un modèle capable de générer des formes naturelles, et en particulier des fougères, avec une vraisemblance étonnante : les IFS (Iterated Function Systems) [Bar88]. Ce modèle permet de produire des formes invariantes par l'application de plusieurs transformations. Depuis, l'utilisation des fractales dans la synthèse comme dans l'analyse d'images est devenue courante. En 1992, Jacquin imagine une méthode de compression d'images fondée sur le modèle IFS légèrement modifié.

Plus récemment, un modèle fondé sur les IFS et permettant d'utiliser le formalisme des formes à pôles a été développé au sein du laboratoire LIGIM : les IFS projetés[Zai98, TZTV97]. Tandis que les IFS introduisent l'aspect fractal accidenté de la forme, les points de contrôle des formes à pôles rendent possible la manipulation interactive comme dans un logiciel de CGAO (Conception Géométrique Assistée par Ordinateur). La thèse de Chems Eddine Zair a été l'occasion de formaliser ce modèle, et d'en exploiter les possibilités en termes de modélisation géométrique. Ceci a permis de générer des courbes et des surfaces fractales grâce à des paradigmes issus de la modélisation géométrique (produit tensoriel

par exemple), mais aussi des IFS (extension aux automates et grammaires).

En modélisation géométrique, une opération courante consiste à approximer un ensemble de points par une forme paramétrée comme une B-spline. Cela permet de décrire l'ensemble de points d'une manière compacte et modifiable. Les modèles utilisés pour ce genre d'approximation sont très souvent lisses. À l'opposé, dans le domaine des fractales, et plus précisément des IFS, l'approximation est une opération moins courante. Et pourtant, elle pourrait permettre de décrire non plus seulement des formes lisses mais aussi des formes accidentées, fractales, de manière compacte, ceci dans un but de modélisation ou de compression. Cette problématique est l'objet de cette thèse et a motivé nos travaux ces trois dernières années. Nous montrons qu'il est possible d'étendre l'approximation de formes (courbes, surfaces mais aussi images) au domaine des fractales. Pour cela nous utilisons le modèle d'IFS projetés que nous combinons avec des méthodes numériques.

Dans la première partie, nous faisons un tour d'horizon des méthodes de modélisation d'objets lisses, puis accidentés. Nous complétons cette étude en donnant une liste de contributions dans le domaine de l'approximation d'objets accidentés ou fractals.

Dans la deuxième partie, nous exposons le modèle fractal d'IFS projeté et les familles de courbes et surfaces qu'il décrit. Nous décrivons ensuite la méthode que nous employons pour approximer les formes grâce à ce modèle. Des résultats d'approximation de courbes, surfaces et images en niveaux de gris sont exposés.

Dans la dernière partie de cette thèse, nous proposons une extension du modèle par combinaison d'IFS projetés pour permettre l'approximation de surfaces ou images de tailles plus élevées et dont les caractéristiques varient spatialement. Deux méthodes associées à ce nouveau modèle sont exposées. L'une de ces méthodes permet notamment de compresser les images en niveaux de gris à de très faibles débits. Pour finir, nous donnons des résultats de compression associés à des comparaisons avec d'autres algorithmes classiques de compression d'images.

Notations

1 Théorie des IFS

$(\mathcal{X}, d)$	Espace d'itération $\mathcal{X}$ muni de la distance d. Il s'agit d'un espace métrique complet.
T	Transformation contractante de $(\mathcal{X}, d)$.
c	Le point fixe d'une transformation contractante.
$(\mathcal{H}(\mathcal{X}), d_H)$	Espace des compacts de $\mathcal{X}$ muni de la distance de HAUSDORFF d_H.
$\mathbb{T}$	Iterated Function System. Il est composé de N transformations contractantes $\{T_0, \ldots, T_{N-1}\}$.
K	Un compact quelconque appartenant à $\mathcal{H}(\mathcal{X})$.
$\mathcal{A}(\mathbb{T})$	Attracteur de l'IFS $\mathbb{T}$.

2 Adressage et paramétrage d'IFS

Σ	Ensemble d'indexation pour les IFS. On parlera aussi d'alphabet car celui-ci sert à générer des mots. On prendra $\Sigma = \{0, \ldots, N-1\}$.
Σ^ω	Ensemble des mots infinis de Σ.
Σ^*	Ensemble des mots finis de Σ. fait correspondre la transformation associée.
ρ, σ, τ	Différents symboles utilisés pour les mots infinis de Σ^ω.
ϕ	Fonction d'adressage de Σ^ω à valeurs dans $\mathcal{X}$. À un mot ρ de Σ fait correspondre le point de l'attracteur associé $\phi(\rho)$.
Φ	Fonction de paramétrage de $[0,1]$ ou $[0,1]^2$ à valeurs dans $\mathcal{X}$.

3 Attracteurs d'IFS projetés

J	Ensemble d'indices, pour les objets filaires on prendra $J = \{0, \ldots, m\}$ et pour les objets surfaciques $J = \{0, \ldots, m\} \times \{0, \ldots, m\}$.
$\mathcal{B}^J$	Espace barycentrique associé à J.

S_J	Semi-groupe barycentrique associé à l'espace barycentrique $\mathcal{B}^J$.
P	Vecteur de points de contrôle.
$(P, \mathbb{T})$	Couple représentant un modèle d'attracteur projeté : un vecteur de points de contrôle et un IFS.
$P\mathcal{A}(\mathbb{T})$	Attracteur projeté associé au couple $(P, \mathbb{T})$.
M	Matrice de passage inversible pour la définition d'IFS projetés équivalents.
F	Fonction de paramétrage de l'attracteur d'IFS projeté. À un scalaire t fait correspondre le point de l'attracteur projeté associé $F(t) = P\Phi(t)$.

4 Approximation

$\mathcal{D}$	Distance entre deux courbes ou surfaces échantillonnées.
$s_i(\mathbf{Q})$	Abscisse curviligne du i^{eme} point de la courbe échantillonnée $\mathbf{Q}$.
$\mathcal{G}_\mathbf{Q}$	Fonction de paramétrage linéaire de la courbe échantillonnée $\mathbf{Q}$.
$\mathbf{a}$	Vecteur de paramètres.
$\delta\mathbf{a}$	Variation du vecteur de paramètres.
$\mathbf{a}_{opt}$	Vecteur de paramètres optimal dans le sens de l'approximation.
$\mathbf{a}_{init}$	Vecteur de paramètres initial dans le processus itératif d'approximation.
$(P_\mathbf{a}, \mathbb{T}_\mathbf{a})$	Modèle d'attracteur projeté associé au vecteur de paramètres $\mathbf{a}$.
$F_\mathbf{a}$	Fonction de paramétrage associée au modèle $(P_\mathbf{a}, \mathbb{T}_\mathbf{a})$.

5 Quadtree

γ	Feuille d'un quadtree représenté sous la forme d'un mot fini de $\{0, 1, 2, 3\}$.
Γ	Ensemble de feuilles d'un quadtree représentant une coupure de celui-ci.

6 Divers

$\mathbb{R}$	L'ensemble des réels.
$\mathbb{N}$	L'ensemble des naturels.

Première partie

Représentation et approximation de formes géométriques

Chapitre 1

Les formes lisses

Dans ce chapitre, nous allons revenir dans l'histoire de quelques modèles lisses utilisés en CGAO (Conception Géométrique Assistée par Ordinateur). Il apparaît de manière très nette que la modélisation de formes lisses est fortement liée au concept d'approximation ou d'interpolation[Far92]. Nous nous pencherons ensuite sur l'émergence de nouveaux modèles lisses et l'utilisation que l'on peut en faire en approximation.

1.1 Naissance de la CGAO et formes à pôles

À la fin des années 50, une nouvelle génération de machines-outils fait émerger le concept de FAO (Fabrication Assistée par Ordinateur). Ces méthodes révolutionnaires se trouvent alors freinées par le manque de modèles et surtout de logiciels en amont de la chaîne de production. Les deux ingénieurs Bézier et De Casteljau sont sans conteste les précurseurs dans ce domaine[Béz70, Béz86, dC86].

Plaçons nous maintenant dans le contexte des années 60, dans le domaine de l'industrie automobile. Les concepteurs de pièces telles que la carrosserie utilisent comme modèle des points reliés à l'aide de pistolets à dessiner et de gabarits. Ce modèle sur papier ne peut pas coïncider avec le « modèle maître » nécessaire à la fabrication. Les ingénieurs mécaniciens voient alors dans l'informatisation une solution évidente à tous ces problèmes. Il s'agit donc *d'approcher* les lignes d'un tracé à la main par une courbe paramétrée. Malheureusement, si les ingénieurs mécaniciens de l'époque ont une bonne connaissance de la géométrie, ils possèdent une formation assez réduite dans les domaines que sont l'algèbre et l'analyse. Ce n'est qu'après de nombreux balbutiements que Bézier [Béz70, Béz86] et De Casteljau [dC86], par des études parallèles, arrivent à la même conclusion : les polynômes ont de bonnes propriétés pour la représentation des courbes utilisées dans l'industrie.

Ainsi est née la CGAO, dans le but d'approcher et d'interpoler les formes. Depuis, de nombreux autres modèles lisses ont été utilisés, tous plus ou moins dérivés des courbes de Bézier, qui sont d'une stabilité numérique exemplaire. Parmi eux, les B-splines rationnelles ou non rationnelles et les facettes de Coons.

1.2 Émergence de nouveaux modèles lisses

La section précédente fait apparaître un lien entre les modèles utilisés en CGAO et le concept d'approximation et d'interpolation. Bien d'autres modèles sont utilisés pour représenter des formes lisses. Si l'on tente de faire une classification, deux types de représentation ressortent : les modèles paramétriques et les modèles implicites. Cette classification est fondée sur un critère mathématique.

Dans le premier cas, les points d'une surface[1] $\mathcal{S}$ sont décrits avec une fonction $\mathcal{F}_\mathcal{S}$ dépendant de paramètres s et t :

$$\begin{array}{rccc} \mathcal{F}_\mathcal{S} : & \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & (s,t) & \mapsto & \mathcal{F}_\mathcal{S}(s,t) \end{array}$$

Dans la représentation implicite, il n'y a plus de paramètres, la surface $\mathcal{S}$ est définie comme étant un ensemble de points dont les coordonnées vérifient une équation fonctionnelle :

$$f_\mathcal{S}(x,y,z) = 0$$

Un même ensemble de points peut bien entendu être représenté de façon paramétrique ou implicite. Par exemple, une surface bilinéaire $\mathcal{S}_B$ peut s'écrire des deux façons suivantes :

$$\mathcal{F}_{\mathcal{S}_B}(s,t) = (s,t,st)$$

ou :

$$f_{\mathcal{S}_B}(x,y,z) = xy - z$$

Ces dernières décénies, on a vu apparaître de nombreux modèles, implicites ou paramétriques qui conduisent généralement à une bonne approximation des données d'origine. A titre d'exemple, nous allons aborder la modélisation et approximation des données volumiques par des superquadriques.

Les superquadriques sont une généralisation des quadriques et ont la particularité de pouvoir s'exprimer sous forme paramétrique et implicite. La formulation implicite est la suivante :

$$f(x,y,z) = \left(\left(\frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} = 1$$

Si l'on considère n'importe quel point de l'espace, en calculant la valeur que prend sa fonction implicite, on peut déterminer si ce point est à l'intérieur de la surface (< 1), sur la surface ($= 1$) ou à l'extérieur de celle-ci (> 1). Étant donné un ensemble de points, il est possible par une méthode numérique de trouver les paramètres du modèle (les a_i et ϵ_i) pour lesquels la distance globale entre la fonction et l'ensemble des points à approximer est minimisée [TM91, GB93, CJB01].

La modélisation des formes lisses est très employée dans le domaine de la CGAO. Ceci est dû au fait que l'on recherche des propriétés géométriques qui induisent ce type de modèles.

[1] pour simplifier le discours, il ne sera question que de surfaces pour cette comparaison

Nous avons vu dans ce chapitre que l'approximation liée aux modèles lisses était très largement étudiée et utilisée. Dans le chapitre suivant, nous allons comprendre pourquoi ces modèles lisses ne sont pas suffisants dès lors que la forme ou le phénomène que l'on cherche à modéliser est de type « naturel ».

Chapitre 2

Les formes rugueuses

Comme l'a observé Mandelbrot [Man82], la nature n'est pas faite des formes lisses relevant de la géométrie classique. Les lignes sont brisées, elles peuvent même être brisées en tout point.

Depuis qu'il a observé la nature fractale des côtes bretonnes [Man82], Mandelbrot a inspiré beaucoup de travaux. Dans tous les domaines qui impliquent des phénomènes naturels, on retrouve l'expression de cette géométrie fractale. Cela va de l'observation de la forme des plantes, jusqu'à la fréquence des rythmes cardiaques en passant par les cours de la bourse. Cependant, il existe deux catégories de phénomènes : ceux qui suivent un mécanisme aléatoire (ou stochastique) et par opposition ceux qui sont issus d'un mécanisme déterministe.

Dans ce chapitre, nous allons dans un premier temps décrire par un exemple un processus aléatoire autosimilaire. Nous verrons que l'approximation n'a pas de sens dans ce cas là. Dans un second temps, nous nous intéresserons aux processus déterministes pour lesquels l'approximation prend un sens.

2.1 Les phénomènes aléatoires

Prenons l'exemple d'une particule qui évolue dans l'espace et dont la trajectoire a une direction et une vitesse aléatoire à n'importe quel instant. Ce modèle qui porte le nom de *mouvement brownien*, explique un nombre considérable de phénomènes analogues à des trajectoires de particules. La figure 1 page suivante illustre un exemple de mouvement brownien. Nous avons généré dans cet exemple une courbe de 20000 points. Le programme qui a servi à créer cette courbe utilise un générateur aléatoire. Lorsque l'on relance celui-ci, on obtient un résultat différent. Pour étudier un phénomène, les physiciens font souvent appel à ce que l'on nomme la *dimension fractale*, qui sert à mesurer la manière dont une forme occupe l'espace. Le mouvement brownien par exemple a une dimension fractale de 2, il « occupe » autant d'espace qu'un plan. Si l'on observe la trace d'une particule, que l'on mesure la dimension fractale de celle-ci, et que l'on trouve 2, alors il s'agit *peut-être* d'un mouvement brownien. En effet, deux causes peuvent engendrer le même effet, une autre cause pourrait donc expliquer le phénomène. Ceci peut paraître un peu caricatural mais il s'agit en fait de la nature *empirique* de la physique – cela n'empêche en aucun cas

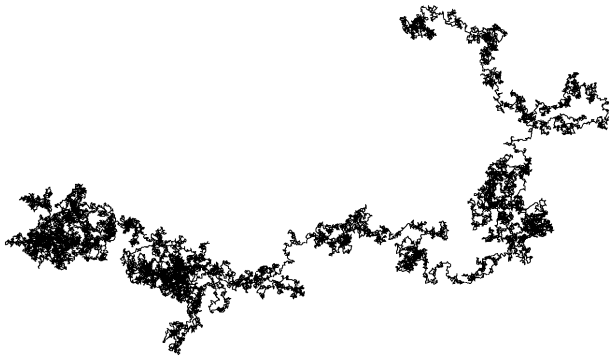


FIG. 1 – Mouvement brownien : à chaque instant, la direction de la particule est aléatoire

une étude ultérieure prouvant qu'il s'agit réellement de tel ou tel modèle. Cet exemple permet de montrer que l'approximation de phénomènes aléatoires par un modèle aléatoire n'est pas réalisable.

2.2 Les phénomènes déterministes

Il existe aussi une grande quantité de phénomènes naturels qui sont déterministes dont les plus connus se trouvent dans la botanique. La figure 2 page suivante montre l'exemple d'un chou de variété *romanesco*. Lorsque l'on considère uniquement une partie de ce chou, on a la même impression visuelle que quand on le regarde en entier : ceci illustre le concept *d'autosimilarité*. Dans le cas de phénomènes aléatoires, ce concept était aussi présent, mais sous une forme différente : il n'était pas possible de trouver une transformation qui régisse ce changement d'échelle. En réalité, les phénomènes qui engendrent des formes strictement autosimilaires déterministes n'existent pas au sens strict, il y a toujours une part de hasard, aussi faible soit elle.

S'inspirant des travaux de HUTCHINSON [Hut81], BARNSLEY [Bar88], en 1988, introduit la notion d'IFS (Iterated Function Systems) pour analyser la nature autosimilaire des objets et propose un modèle fondé sur des transformations élémentaires. La figure 3 page ci-contre donne un exemple d'IFS composé de quatre transformations affines qui sont chacune des combinaisons d'homotéties, translations et rotations :

$$\begin{aligned} T_0 &= H(0.5) \\ T_1 &= T(0, 0.5)H(0.5) \\ T_2 &= T(0, 1)R(\pi/4)H(0.5) \\ T_3 &= T(0, 1)R(-\pi/4)H(0.5) \end{aligned}$$

Etant donné qu'avec cette classe de modèles les phénomènes sont reproductibles, chaque instance de modèle génère une même figure. Il doit donc maintenant être possible de *retrouver* l'instance du modèle qui a généré telle ou telle figure. Suivant la nature de la méthode utilisée, on parlera de *problème inverse*, *d'approximation* ou même de *compression*. Le chapitre suivant propose une étude bibliographique du sujet.

FIG. 2 – Le chou romanesco : une partie quelconque extraite ressemble au chou entier

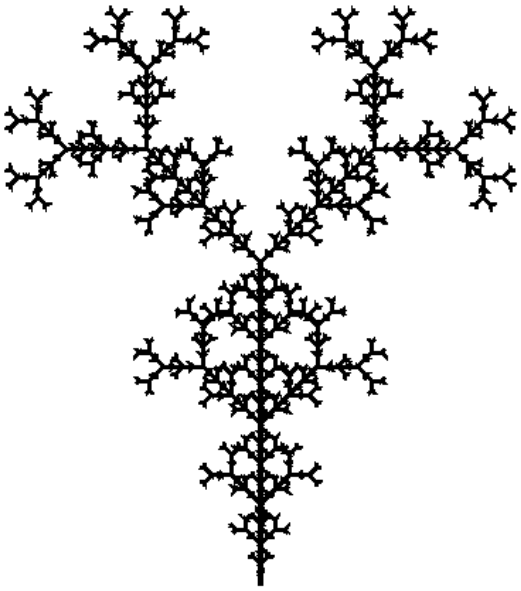


FIG. 3 – Exemple d'IFS composé de transformations affines dans le plan

Chapitre 3

Le problème inverse

Dans le chapitre précédent, nous avons vu que grâce aux modèles fractals déterministes, il est possible de poser la question générique : « Nous savons que tel phénomène est régi par des lois qui le rendent autosimilaire, quels sont les paramètres de cette autosimilarité ? » **Ce qu'on appelle le problème inverse consiste à répondre à cette question.** Malheureusement, la tâche n'est pas si facile, dans leur article [RH97], Ruhl et Hartenstein démontrent même qu'une catégorie de ce type de problème est NP-difficile. Ceci signifie qu'un algorithme qui fournit la réponse ne peut le faire en un temps polynomial par rapport à la taille des données d'entrée. Ce chapitre propose une liste de méthodes existantes qui apportent des solutions à notre problème. Dans certains cas, nous verrons que l'emploi d'un modèle donné a guidé l'utilisation d'une méthode de résolution précise. Dans d'autres cas, c'est la volonté d'appliquer une certaine méthode qui a conduit à utiliser un type précis de modèle.

3.1 Analyse par ondelettes

Différentes études ont mis en avant le lien qu'il y avait entre les ondelettes et les fractales, notamment les multifractales [MBA94]. En effet, l'autosimilarité présente dans les fractales engendre une structure bien précise dans le domaine fréquentiel. Les phénomènes donnant lieu à des observations multifractales sont maintenant analysés grâce à des outils adaptés[Mal99].

En se basant sur méthode ondelette, il est possible de trouver l'exposant de Hölder ponctuel en tout point d'une fonction [JM96]. Lévy-Véhel et Daoudi ont utilisé une méthode similaire pour analyser des signaux de paroles 1D [VD96, DV95]. Grâce à une modélisation par IFS généralisés, l'exposant de Hölder peut être reproduit dans un nouveau signal, imitant ainsi le signal original.

Si l'on s'intéresse maintenant au problème inverse, de récentes études sont parvenues à extraire des paramètres de transformations d'IFS à partir d'une analyse de type ondelettes [SDG95, Ber97]. Ces études ont été faites sur des données 1D synthétiques, avec un modèle de type IFS à transformations affines. La difficulté de cette méthode réside dans l'extraction d'informations des résultats donnés par l'analyse ondelettes. Cette complexité empêche pour l'instant de traiter des données réelles et de dimension supérieures.

L'analyse par ondelettes pour la résolution du problème inverse est donc une méthode qui a poussé les auteurs, de par la complexité des processus mis en œuvre, à utiliser un modèle simple.

3.2 Compression fractale d'images

Si la nature est faite de formes autosimilaires, cela signifie que des images de la nature peuvent présenter de l'autosimilarité. C'est à partir de cette constatation que Jacquin a imaginé une technique de compression d'images fondée sur un modèle IFS en 1992 [Jac92]. Cette technique permet d'obtenir une approximation des niveaux de gris de l'image originale (compression avec perte d'information). Pour ce faire, Jacquin a considéré des transformations géométriques simples comme base pour décrire les images. Celles-ci sont des combinaisons de translations dans des blocs de l'image, rotations simples et changements d'échelles simples. Étant donné qu'une image naturelle n'est pas stationnaire, une seule transformation ne peut modéliser l'intégralité des pixels. Un ensemble de transformations est considéré en appliquant chacune des transformations sur une petite partie de l'image. La figure 4 illustre ce principe. L'image est d'abord découpée en blocs « range ».

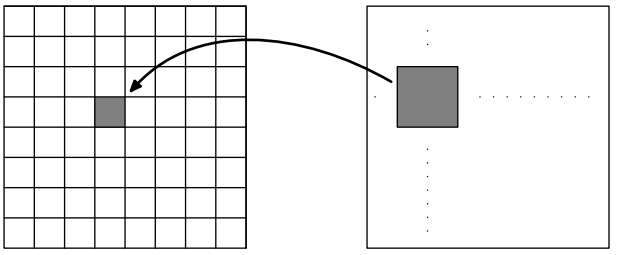


Fig. 4 – Principe de la compression fractale. On cherche pour tous les blocs range les blocs domaines susceptibles de correspondre

Pour chacun de ces blocs, on essaie de trouver dans l'image un bloc « domaine » de taille plus élevée (typiquement 2 fois plus) qui pourrait être mis en correspondance grâce à une transformation simple. Les taux de compression sont d'autant plus importants que l'image est réellement autosimilaire – pour une classe de transformations contractantes donnée.

Diverses études ont tenté d'améliorer la méthode à plusieurs niveaux [Fis92, Bel98, SHH96, WdJ99] :

– Temps de calcul.
  En effet, celui-ci est élevé car il faut analyser pour chaque combinaison de blocs si une des transformations est applicable ou non. Diverses stratégies ont été proposées pour réduire cette recherche exhaustive [Bel98]. De même, la complexité combinatoire dans la recherche de correspondances de blocs a été réduite, ou optimisée. Une technique courante consiste à diminuer la taille spatiale de recherche. De plus, une classification préalable des blocs domaines peut s'avérer utile, et permet d'emblée d'éliminer certaines correspondances.

– Partitionnement de l'image.
  La méthode de base propose une décomposition en blocs de tailles fixes, diverses études ont proposé des partitionnements plus évolués : quadtree [LY94], triangulaire (avec Delaunay) [DC94], horizontal-vertical, polygonal quelconque,... Il s'avère que le gain existe mais reste assez limité.
– Transformations utilisées.
  Il y a deux types de transformations à considérer : les tranformations spatiales qui déplacent spatialement les pixels dans l'image et les transformations sur le niveau de gris qui modifient l'intensité de ceux-ci. La première catégorie spatiale dépend de manière assez évidente du partitionnement utilisé. Peu d'améliorations ont été effectuées par rapport au modèle initial proposé par Jacquin [Jac92]. En ce qui concerne la deuxième catégorie, on considère aussi des transformations affines. Des techniques ont permis de décorréler la partie homothétie de la partie translation, grâce notamment à une projection orthogonale [iBL +94]. D'autre études ont travaillé sur l'application des transformations sur le domaine fréquentiel à l'aide d'une DCT (Discrete Cosine Transform) [BV94] ou d'une DWT (Discrete Wavelet Transform) [DACB96, BBBCP98]. Ces divers travaux ont permis de déboucher sur des méthodes hybrides plus efficaces.

Il semble que le partitionnement le plus adapté soit le quadtree, en combinaison avec des transformations qui agissent dans le domaine fréquentiel. Pour plus de précisions, le lecteur peut consulter l'article de synthèse de Saupe [SHH96] ou celui plus récent de Wohlberg [WdJ99], ainsi que le chapitre 7 de l'ouvrage *Compression codage des images et des vidéos* [BL02].

La compression fractale fait partie de ces méthodes pour lesquelles le modèle utilisé est indissociable de la méthode elle-même. Ainsi, le modèle des PIFS (Partitioned IFS) a-t-il été développé pour les besoins de la méthode de compression fractale.

3.3 Algorithmes génétiques

Plusieurs équipes se sont interressées aux algorithmes génétiques pour la résolution du problème inverse fractal [VR94, GMA94, LLVC +95].

Les algorithmes génétiques procurent une méthode générale d'optimisation. De ce fait, ils sont applicables à de nombreux modèles. Le premier modèle à avoir été utilisé est celui de la compression fractale : les PIFS. Plusieurs équipes sont parvenues à obtenir de meilleurs résultats que les méthodes combinatoires citées dans la section précédente [VR94, GMA94]. Malheureusement, ces méthodes mettent en œuvre de gros coûts calculatoires, pour un gain qui reste assez faible. Une autre approche consiste à dire qu'à l'aide d'algorithmes génétiques, il est possible d'utiliser des modèles beaucoup plus complexes. L'équipe fractale de l'INRIA a mis au point des méthodes qui permettent d'optimiser des fonctions fractales complexes : les IFS mixtes [LLVC +95]. Les transformations de ces IFS ne sont plus affines, elles comportent des fonctions sinus, racine carrée,...L'optimisation est alors combinatoire et non uniquement numérique : on cherche l'expression formelle de la fonction qui correspond le mieux, cette expression étant codée par un arbre. Cette technique a été utilisée pour l'approximation d'images [LLVC +95].

Le principal inconvénient des méthodes génétiques reste leur temps de calculs élevés. Malgré tout, celles-ci permettent d'ouvrir le domaine de modélisation à des horizons beaucoup plus larges.

3.4 Perturbation d'un modèle lisse

L'idée de départ est simple : puisque l'on maîtrise bien l'approximation par des modèles lisses classiques, pourquoi ne pas effectuer celle-ci en premier lieu puis perturber le modèle lisse et obtenir un modèle rugueux ? Blanc-Talon [BT97a, BT97b] propose cette méthode pour l'approximation de terrains. A partir des données de départ, une première approximation est faite grâce à l'utilisation de courbes de Bézier. En introduisant un paramètre de perturbation des matrices de subdivision, ce modèle est transformé pour obtenir un modèle rugueux. Le but est d'obtenir la même dimension fractale que dans les données d'entrée. Ce concept est très intéressant car il montre qu'à partir de la combinaison de méthodes très simples, il est possible d'obtenir des résultats satisfaisants rapidement. Son inconvénient est que le but recherché est uniquement l'apparence, impliquant que les détails générés par le modèle ne sont pas identiques à ceux des données d'entrée.

3.5 Notre contribution

Nous avons vu dans ce chapitre que le concept d'approximation de formes rugueuses par des modèles fractals a vu le jour il y a une dizaine d'années. La compression fractale est pionnière en la matière. Elle a permis de montrer que les modèles fractals pouvaient être efficaces pour la compression. Malheureusement, les limites du modèle commencent à se faire sentir, malgré de nombreux efforts pour essayer de les combler. Les méthodes à base de compression fractale les plus efficaces sont d'ailleurs des méthodes hybrides. Tour à tour, les équipes ont tenté d'approximer grâce à de nouveaux modèles, de nouvelles méthodes. Les méthodes à base d'ondelettes s'avèrent être efficaces dans le sens où elles permettent de résoudre dans certains cas le problème inverse de façon exacte. Dans d'autres cas elles servent à analyser des signaux pour en extraire de l'information qui sert à une reconstruction – pas forcément exacte. Les algorithmes génétiques ont permis de leur côté de travailler sur autre chose que des IFS à transformations affines : les IFS mixtes. La méthode de compression fractale permet de s'attaquer directement au problème de la compression d'images avec un algorithme et un modèle dédiés.

Qu'attendons-nous de la méthode que nous allons développer ? Les deux fonctionnalités dont nous souhaitons à tout prix disposer sont celles de pouvoir traiter les données réelles, ainsi que la possibilité de modélisation ultérieure, rendant possible l'utilisation de la méthode dans un programme de modélisation géométrique. Ainsi, le concept d'approximation lisse tel qu'il existe en modélisation géométrique standard sera transposé au domaine du rugueux. Le dernier exemple donné dans ce chapitre est ce qu'il y a de plus proche de notre but. Néanmoins, nous allons nous tourner vers d'autres solutions, en continuité avec les travaux déjà effectués au sein du laboratoire du LIGIM.

Nous retiendrons surtout de cette brève étude que la modélisation de formes rugueuses est souvent faite à l'aide d'IFS. Nous allons développer par la suite une méthode qui utilise des IFS, mais sous une forme *projetée*. Cette projection permet un meilleur contrôle de la forme, par l'intermédiaire de *points de contrôle*. Grâce à une formulation complètement numérique, le problème d'approximation va se transformer en un problème d'optimisation non-linéaire. L'objectif est double :

- Reconstruction de formes rugueuses : profils de terrain, contours, grilles d'altitudes,...
- Compression d'images par résolution du problème inverse.

La partie suivante développe cette méthode, en commençant par une description du modèle utilisé, puis de la méthode à proprement parler.

Deuxième partie

Approximation

Chapitre 1

Modèle général

Le modèle que nous allons utiliser a été développé dans notre laboratoire. Il a fait l'objet de la thèse de Chems Eddine Zair [Zai98]. Il repose sur la fusion de deux modèles : les IFS, modèle fractal bien connu, et les formes à pôles, formalisme très utilisé en CGAO. Le résultat est appelé *IFS projeté*.

Dans un premier temps, nous introduisons les IFS sans rentrer trop dans les détails. Le lecteur pourra se reporter aux études de Barnsley [Bar88, Bar93]. Dans la suite, nous explicitons le modèle utilisé. Pour cela, nous exposons les procédés de paramétrisation des attracteurs d'IFS, puis la notion d'attracteur d'IFS projetés. Afin d'approcher des courbes ou des surfaces, nous voyons comment il est possible de calculer les valeurs des fonctions fractales associées avec un nombre réduit d'itérations.

1.1 IFS : Iterated Function Systems

Fondés sur un formalisme mathématique rigoureux, les IFS sont un outil puissant pour l'analyse comme pour la synthèse d'objets fractals. Hutchinson[Hut81] fut le premier à les étudier en appliquant le théorème du point fixe aux ensembles de compacts d'un espace métrique complet. Barnsley[Bar88] a développé ce formalisme et l'a utilisé dans toute une série d'applications, entre autres, en infographie et en compression d'image. Dans cette section, nous allons rappeler la construction de ce modèle.

1.1.1 Théorème du point fixe

Le théorème du point fixe nécessite la définition d'un espace métrique complet ainsi que d'une application contractante.

Définition 1 (espace métrique) On appelle espace métrique tout ensemble $\mathcal{X}$ muni d'une distance d. On dira que $\mathcal{X}$ est métrique complet si toute suite de Cauchy définie dans $\mathcal{X}$ admet une limite dans $\mathcal{X}$.

Définition 2 (application contractante) Soient $(\mathcal{X}, d)$ un espace métrique complet et T une application de $\mathcal{X}$ dans $\mathcal{X}$. L'application T est dite contractante si :

$$\exists s < 1, \forall p_1, p_2 \in \mathcal{X} \ , \ d(T(p_1), T(p_2)) \leq sd(p_1, p_2)$$

Théorème 1 (du point fixe) *Soit $(\mathcal{X}, d)$ un espace métrique complet. Soit $T : \mathcal{X} \to \mathcal{X}$ une application contractante. Alors il existe un point unique $c \in \mathcal{X}$, appelé le point fixe de T, tel que $T(c) = c$.*

Pour une plus grande simplicité dans la suite, on notera T comme un opérateur $T\lambda = T(\lambda)$.

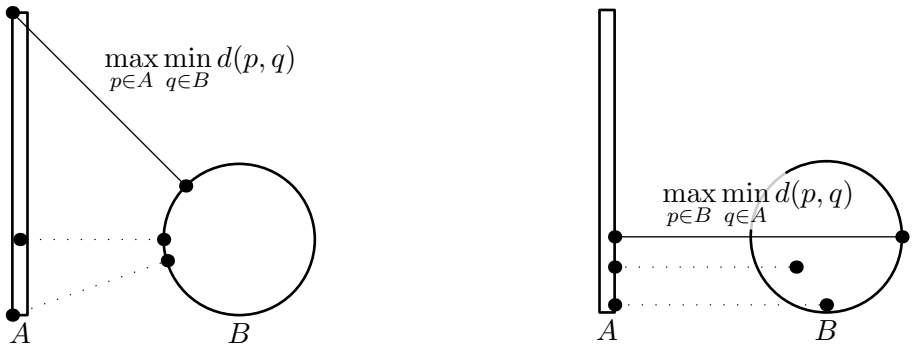
1.1.2 Espaces de compacts, opérateur de Hutchinson

La théorie des IFS repose sur la définition, à partir d'un espace métrique complet $(\mathcal{X}, d)$, d'un autre espace métrique complet, qui contient tous les compacts du premier. Nous noterons $\mathcal{H}(\mathcal{X})$ l'ensemble des compacts non vides de $(\mathcal{X}, d)$. La distance utilisée pour $\mathcal{H}(\mathcal{X})$ est construite grâce à d, elle est appelée distance de Hausdorff et est notée d_H.

Définition 3 (distance de Hausdorff) Soient A et B deux compacts non vides d'un espace métrique $\mathcal{X}$. La distance de Hausdorff de A à B est définie par :

$$d_H(A, B) = \max \left\{ \max_{p \in A} \min_{q \in B} d(p, q), \max_{p \in B} \min_{q \in A} d(p, q) \right\}$$

De façon plus parlante, $\max_{p \in A} \min_{q \in B} d(p, q)$ représente le maximum de la distance entre un point de A et le compact B. De la même façon, $\max_{p \in B} \min_{q \in A} d(p, q)$ représente le maximum de la distance entre un point de B et le compact A. La figure 5 donne un exemple de calcul de distance de Hausdorff. On comprend vite par cet exemple que la distance de Hausdorff fait intervenir des paramètres comme la forme et la disposition des compacts considérés, et non uniquement un éloignement moyen. Il s'agit donc d'une distance « visuelle ».



(a) Calcul du premier terme

(b) Calcul du second terme

Fig. 5 – Exemple de distance de Hausdorff entre deux compacts de $(\mathbb{R}^2, d)$

Le théorème suivant est la base de la théorie des IFS[Bar88].

Théorème 2 *Si $(\mathcal{X}, d)$ est un espace métrique complet, alors $(\mathcal{H}(\mathcal{X}), d_H)$ l'est aussi.*

Des applications contractantes dans $(\mathcal{H}(\mathcal{X}), d_H)$ peuvent être définies à partir d'applications contractantes dans $(\mathcal{X}, d)$. C'est le cas de l'opérateur de HUTCHINSON. Nous noterons l'action d'un opérateur sur un compact de la façon suivante :

$$TK = \bigcup_{p \in K} \{Tp\}$$

Proposition 1 (opérateur de HUTCHINSON) *Soient N opérateurs continus $T_i : \mathcal{X} \to \mathcal{X}, i = 0, \ldots, N-1$, on peut leur associer l'opérateur continu défini par :*

$$K \in \mathcal{H}(\mathcal{X}) \mapsto \mathbb{T}K \in \mathcal{H}(\mathcal{X}) = \bigcup_{i=0}^{N-1} T_i K$$

Celui-ci est appelé opérateur de HUTCHINSON *associé à $\mathbb{T}$.*

De plus, les propriétés de contraction de l'opérateur de HUTCHINSON sont liées à celles des T_i.

Proposition 2 *Si tous les opérateurs $T_0, \ldots, T_{N-1}$ sont contractants de constantes de contraction $s_0, \ldots, s_{N-1}$, alors l'opérateur de* HUTCHINSON *associé à $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$ est contractant pour la distance de* HAUSDORFF *et de constante de contraction $s = \max\{s_0, \ldots, s_{N-1}\}$.*

1.1.3 Définition d'un IFS

Définition 4 (IFS) Soit $(\mathcal{X}, d)$ un espace métrique complet. Nous appelons IFS tout ensemble fini $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$ d'opérateurs contractants sur $\mathcal{X}$.

Les propositions précédentes permettent d'associer à cet ensemble d'opérateurs un opérateur de HUTCHINSON qui est contractant dans l'espace métrique complet $(\mathcal{H}(\mathcal{X}), d_H)$. Il est donc possible d'appliquer le théorème du point fixe.

Théorème 3 (existence d'attracteur) *Pour tout IFS $\mathbb{T}$, il existe un compact unique non vide A de $\mathcal{H}(\mathcal{X})$ tel que :*

$$\begin{aligned} A &= \mathbb{T}A \\ &= T_0 A \cup \ldots \cup T_{N-1} A \end{aligned}$$

A est appelé attracteur de $\mathbb{T}$ et sera noté $\mathcal{A}(\mathbb{T})$.

Il est donc possible d'associer à tout IFS $\mathbb{T}$ un compact, appelé attracteur. Par définition, tout attracteur possède la propriété d'auto-similarité au sens large, c'est à dire :

$$A = T_0 A \cup \ldots \cup T_{N-1} A$$

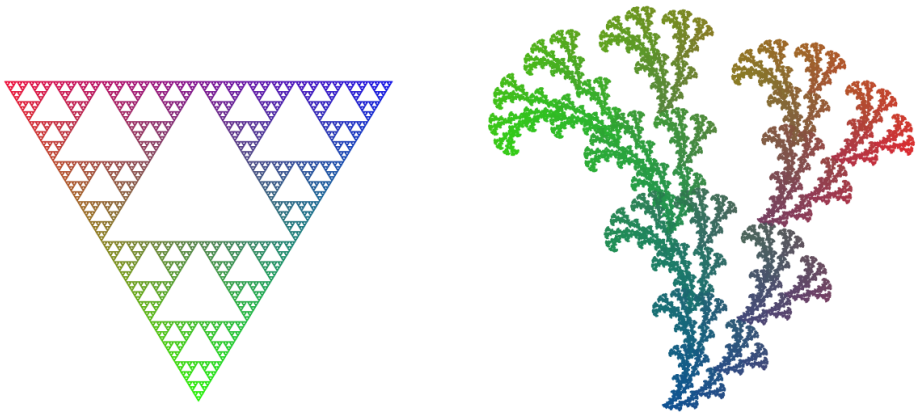
avec les T_i appartenant à une classe de fonctions contractantes.

1.1.4 Exemples

Prenons le cas où $\mathcal{X} = \mathbb{R}^2$, ce compact peut être assimilé à une figure du plan. Les figures 6 et 7 montrent quatre attracteurs d'IFS dont les transformations sont affines. La propriété d'autosimilarité est plus ou moins perceptible visuellement. Dans la figure 6a, elle est difficilement décelable tandis que dans les figures 7a ou 7b, elle saute aux yeux, dans les figures 6a et 6b elle est décelable mais moins facilement.

(a) (b)

FIG. 6 – Exemples d'attracteurs d'IFS dans le plan



(a) Le classique triangle de Sierpinsky

(b)

Fig. 7 – Exemples d'attracteurs d'IFS colorés

1.2 Paramétrisation d'attracteurs

Dans le paragraphe précédent, les attracteurs d'IFS sont définis comme des ensembles de points. Il est possible cependant d'en faire des *modèles paramétrés*. Pour cela, il faut numéroter les opérateurs de l'IFS, afin de construire une fonction d'adressage[Bar88, Edg90]. Dans le cas où l'attracteur est une courbe ou une surface, cette paramétrisation formelle peut être mise en correspondance avec une paramétrisation numérique.

1.2.1 Fonction d'adressage

Définition 5 (alphabet) Soit un IFS $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$. $\Sigma = \{0, \ldots, N-1\}$ sera appelé l'alphabet associé à $\mathbb{T}$.

En notant Σ^* l'ensemble des mots finis sur Σ, et Σ^ω l'ensemble des mots infinis sur Σ, on a le résultat suivant (voir Barnsley[Bar88]) :

Théorème 4 *Soit $\rho = \rho_1\rho_2\rho_3\ldots \in \Sigma^\omega$. Pour tout $\lambda \in \mathcal{X}$ la limite :*

$$\lim_{j\to\infty} T_{\rho_1} \ldots T_{\rho_j} \lambda$$

existe et est indépendante de λ.

On peut définir une fonction d'adressage, qui, à un mot infini sur Σ, associe un point de l'attracteur.

Définition 6 (fonction d'adressage) Soient $\mathbb{T}$ un IFS et Σ son alphabet associé. On appelle fonction d'adressage la fonction ϕ qui à un mot infini de Σ fait correspondre un élément de $\mathcal{X}$ de la façon suivante :

$$\begin{array}{rccl} \phi: & \Sigma^\omega & \to & \mathcal{X} \\ & \rho & \mapsto & \phi(\rho) = \lim\limits_{j\to\infty} T_{\rho_1} \dots T_{\rho_j} \lambda \end{array}$$

avec $\lambda \in \mathcal{X}$ quelconque.

1.2.2 Formes paramétrées

Cette fonction d'adressage est insuffisante pour décrire une forme paramétrée de type courbe ou surface. Dans ces cas, il est plus pratique de disposer d'une fonction dont les arguments sont des scalaires $s \in [0,1]$. Sous certaines conditions sur l'IFS $\mathbb{T}$, que nous verrons dans la suite, une telle fonction existe et définit un arc de courbe ou un carreau de surface. Pour cela, nous utilisons le développement de s en base N :

$$s = \sum_{i=1}^{\infty} \frac{1}{N^i} \rho_i$$

Définition 7 (fonction de paramétrage d'un IFS) Soient $\mathbb{T}$ un IFS, Σ son alphabet, et ϕ sa fonction d'adressage. La fonction de paramétrage Φ de l'IFS $\mathbb{T}$ est définie par :

$$\begin{array}{rccl} \Phi: & [0,1] & \to & \mathcal{X} \\ & s & \mapsto & \Phi(s) = \phi(\rho) \end{array} \tag{1.1}$$

avec $s = \sum_{i=1}^{\infty} \frac{1}{N^i} \rho_i$.

Cette fonction de paramétrage à un seul paramètre permet de décrire des objets tels que les courbes.

Lorsque l'on veut décrire un carreau de surface, il faut disposer d'une fonction de paramétrage à deux paramètres. Nous nous cantonnerons ici à rappeler comment, de façon formelle, il est possible d'obtenir un paramétrage à deux paramètres. Pour cela, il suffit de considérer que l'adresse donnée par le mot infini ρ n'est plus unidimensionnelle mais bidimensionnelle, grâce à un code de Péano. La figure 8 illustre comment, dans le

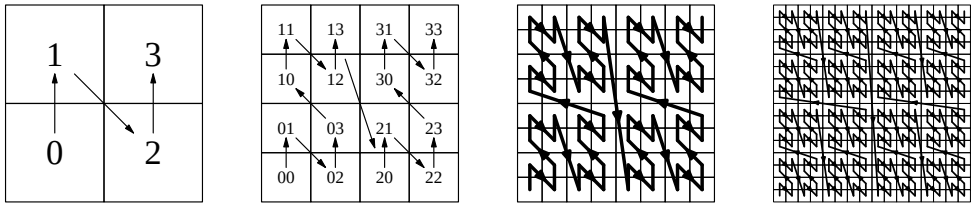


Fig. 8 – Adressage de Péano dans le cas où $N = 4$

cas où $N = 4$, à partir d'un seul mot infini, il est possible d'adresser le pavé $[0,1]^2$. D'une manière plus formelle, il faut que N soit un carré : $N = \tilde{N}^2$. À ce moment, le mot ρ peut se décomposer de la manière suivante :

$$\rho = \rho_1 \dots \rho_n \dots = \sigma_1 \natural \tau_1 \dots \sigma_n \natural \tau_n \dots$$

La fonction de paramétrage à deux paramètres devient alors :

$$\begin{array}{rccc} \Phi : & [0,1]^2 & \rightarrow & \mathcal{X} \\ & (s,t) & \mapsto & \Phi(s,t) = \phi(\rho) \end{array}$$

avec $s = \sum_{i=1}^{\infty} \frac{1}{\tilde{N}^i} \sigma_i$ et $t = \sum_{i=1}^{\infty} \frac{1}{\tilde{N}^i} \tau_i$.

Le symbole $\natural$ représente le code de Péano, c'est à dire :

$$i \natural j = \tilde{N}\, i + j$$

1.3 Formes fractales à pôles

Le modèle IFS est un outil puissant pour la création de formes fractales, en particulier dans le domaine artistique. Cependant, il est insuffisant dans un contexte de modélisation géométrique. Le problème apparaît lorsque l'on veut pouvoir déformer l'attracteur, action courante en modélisation géométrique. L'idée des attracteurs d'IFS projetés a vu le jour au LIGIM. Elle a été développée dans la thèse de Chems Eddine Zair [Zai98]. Le principe est de fusionner deux modèles : les IFS et les formes à pôles. Dans un premier temps, nous allons rappeler le formalisme des formes à pôles. Puis, nous verrons comment il est possible de l'étendre aux IFS. Enfin, nous montrerons des exemples de courbes et surfaces fractales qui sont des attracteurs d'IFS projetés.

1.3.1 Formes à pôles

L'idée principale des formes à pôles est de séparer la fonction qui représente une courbe ou une surface en deux parties : le polygone de contrôle $(p_j)_{j \in J}$ et les fonctions de mélange $(\Phi_j)_{j \in J}$.

La formule qui décrit une courbe est la suivante :

$$F(s) = \sum_{j \in J} \Phi_j(s) p_j$$

Le polygone de contrôle sert à contrôler la forme globale de la courbe à modéliser. Il permet de déformer celle-ci d'une manière intuitive. Il est noté $P = (p_j)_{j \in J}$ où J désigne l'ensemble des indices des points de contrôle. L'ensemble des indices pour les courbes est $\tilde{J} = \{0, \dots, m\}$ de manière à décrire un polygone de contrôle. Les fonctions de mélange quant à elles indiquent la façon dont les points de contrôle sont « mélangés » pour constituer la forme finale. Elles sont notées $(\Phi_j)_{j \in J}$. En modélisation géométrique classique, les fonctions de mélange sont fixées une fois pour toutes et l'utilisateur dispose des points

de contrôle pour modifier la forme. Ces fonctions de mélange sont des polynômes, des B-splines, des NURBS,...et possèdent la propriété suivante :

$$\forall s \in [0,1],\ \sum_{j \in J} \Phi_j(s) = 1$$

Elles permettent ainsi de combiner les points de contrôle. La figure 9 montre un exemple de modélisation géométrique grâce à une courbe B-spline.

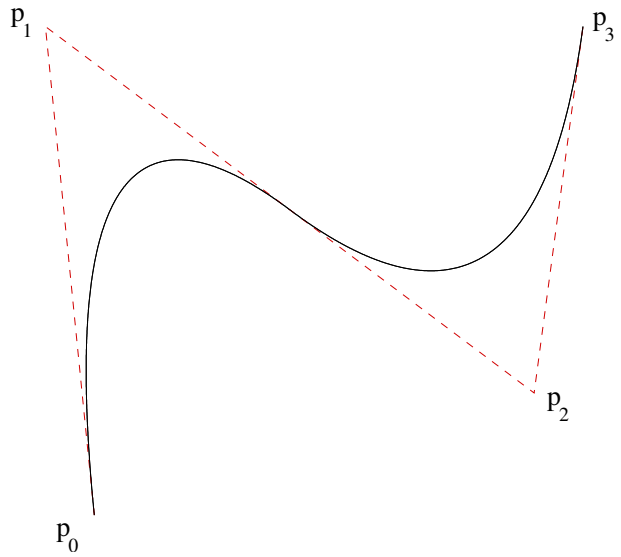


FIG. 9 – Courbe B-spline

Remarque : Les points de contrôle p_j sont choisis dans un espace dit de modélisation. En pratique ce sera $\mathbb{R}^2$ ou $\mathbb{R}^3$.

Cette notion s'étend aux surfaces grâce à un ensemble d'indices $J = \{0,\dots,m\} \times \{0,\dots,m\}$. La formule qui décrit la surface est alors :

$$F(s,t) = \sum_{j \in J} \Phi_j(s,t) p_j$$

et la propriété des fonctions de mélange se transpose ainsi :

$$\forall (s,t) \in [0,1]^2,\ \sum_{j \in J} \Phi_j(s,t) = 1$$

L'idée des formes fractales à pôles est de fabriquer des jeux de fonctions de mélange différents des fonctions standards et ayant des propriétés fractales.

1.3.2 IFS projeté

Pour fabriquer des fonctions de mélange fractales, il faut les définir en coordonnées barycentriques. Il faut donc choisir comme espace d'itération un espace barycentrique.

Définition 8 (espace barycentrique) Soit J un ensemble d'indices. L'espace barycentrique $\mathcal{B}^J$ associé à J est défini par :

$$\mathcal{B}^J = \left\{ (\lambda_j)_{j\in J} \mid \sum_{j\in J} \lambda_j = 1 \right\}$$

avec $\lambda_j \in \mathbb{R}$.

Il faut maintenant se fixer un semigroupe d'itération qui opère dans cet espace barycentrique. La solution la plus simple est d'utiliser des matrices dont les colonnes sont barycentriques. Dans toute la suite du document, nous appelons ces matrices *matrices de subdivision*. En effet, les matrices composant l'IFS correspondent aux matrices utilisées dans plusieurs schémas de subdivision de courbes ou de surfaces (algorithme de De Casteljau, Overveld, Daubechies,...).

Définition 9 (semigroupe de matrices barycentriques) Soit J un ensemble d'index. Le semigroupe de matrices barycentriques S_J associé à J est défini par :

$$S_J = \left\{ T \mid \sum_{j\in J} T_{ij} = 1, \forall i \in J \right\}$$

Remarque : On constatera facilement qu'un espace barycentrique muni de la distance euclidienne est complet[Zai98]. En effet, il s'agit d'un hyperplan, équivalant métriquement à $\mathbb{R}^{\mathrm{card}(J)-1}$.

Grâce à ces choix, il est possible de « projeter » un attracteur d'IFS.

Définition 10 (attracteur d'IFS projeté) Soient J un ensemble d'indices, $\mathbb{T}$ un IFS constitué d'opérateurs de S_J, et $P = (p_j)_{j\in J}$ une famille de points de contrôle.

L'attracteur d'IFS projeté associé à $\mathbb{T}$ est défini par :

$$\begin{aligned} P\mathcal{A}(\mathbb{T}) &= \{P\lambda \mid \lambda \in \mathcal{A}(\mathbb{T})\} \\ &= P\phi(\rho) \\ &= \textstyle\sum_{j\in J} \phi_j(\rho) p_j \end{aligned}$$

où $P\lambda = \sum_{j\in J} \lambda_j p_j$.

Ceci est une définition ensembliste de l'attracteur d'IFS projeté. Il est en fait possible de définir celui-ci de manière fonctionnelle (ou paramétrique) grâce à l'équation de paramétrisation de l'IFS :

$$F(s) = P\Phi(s) = \sum_{j\in J} \Phi_j(s) p_j \tag{1.2}$$

ou avec 2 paramètres dans le cas des surfaces :

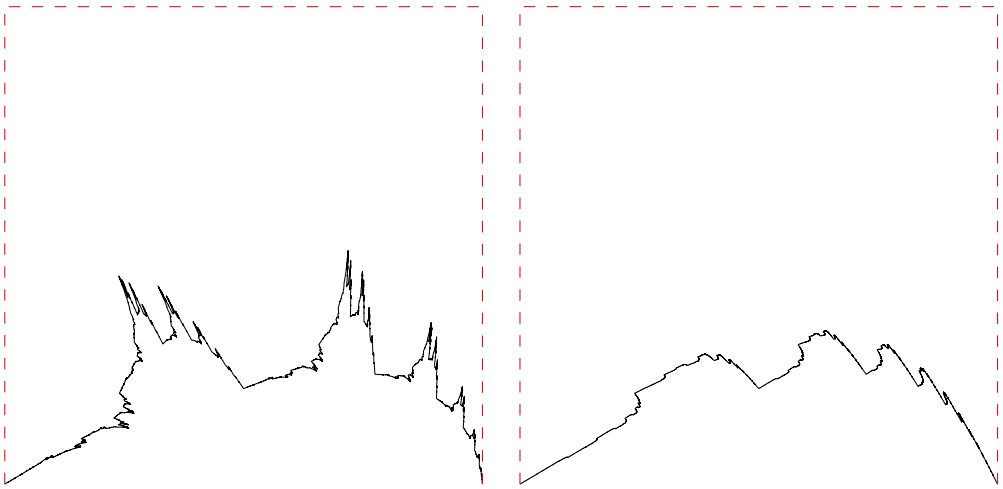
$$F(s,t) = P\Phi(s,t) = \sum_{j\in J} \Phi_j(s,t) p_j \tag{1.3}$$

1.3.3 Exemples

Nous allons ici donner des exemples d'attracteurs d'IFS projetés, pour illustrer la variété des formes qui peut être générée grâce à ce modèle.

Courbes

Les figures 10 à 13 sont des exemples de courbes fractales à pôles. Elles ont toutes été générées avec le même polygone de contrôle disposé en carré, et le même nombre de transformations : deux matrices de subdivision T_0 et T_1. Pour obtenir des courbes différentes, nous avons donc joué sur les coefficients contenus dans les matrices de subdivision. Lorsqu'il existe une symétrie entre les deux matrices, on obtient une courbe symétrique [Zai98] (figures 11a, 11b et 12b). Le caractère accidenté des courbes générées est très variable. Dans les figures 10a, 10b, et 13a, celui-ci est assez faible. Dans les figures 11b, 12a et 12b, il est plus prononcé. Les figures 14 et 15 illustrent les fonctions de mélange Φ, et leur équivalent fractal. La figure 14 montre un exemple de fonction de mélange (pour une B-spline) en explicitant chacune des quatres composantes associées aux quatre points de contrôle. La figure 15 illustre un cas fractal, avec la même représentation. Dans les deux figures, l'abscisse représente la valeur du paramètre de la courbe. Le formalisme est donc le même dans les deux cas, mais on a remplacé des fonctions à base de polynômes par des fonctions fractales définies par un modèle IFS.

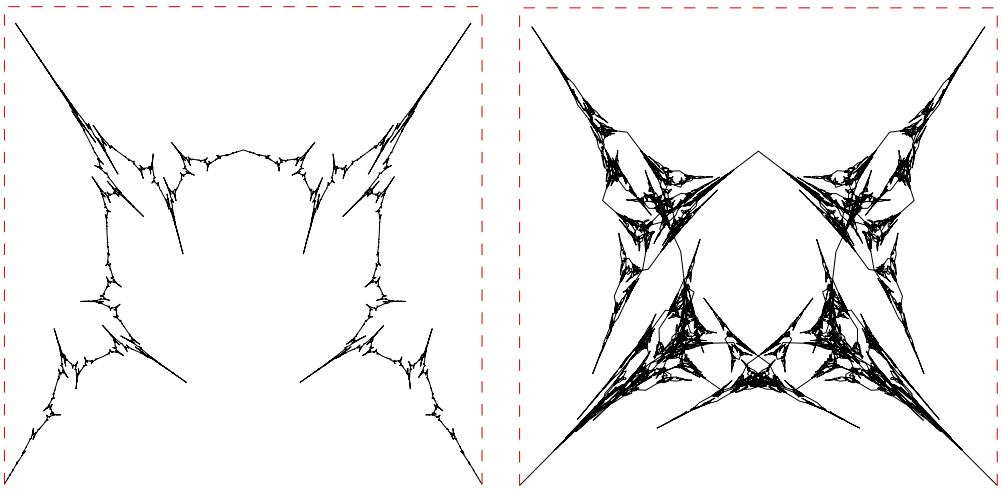


(a) (b)

FIG. 10 – Courbes faiblement accidentées

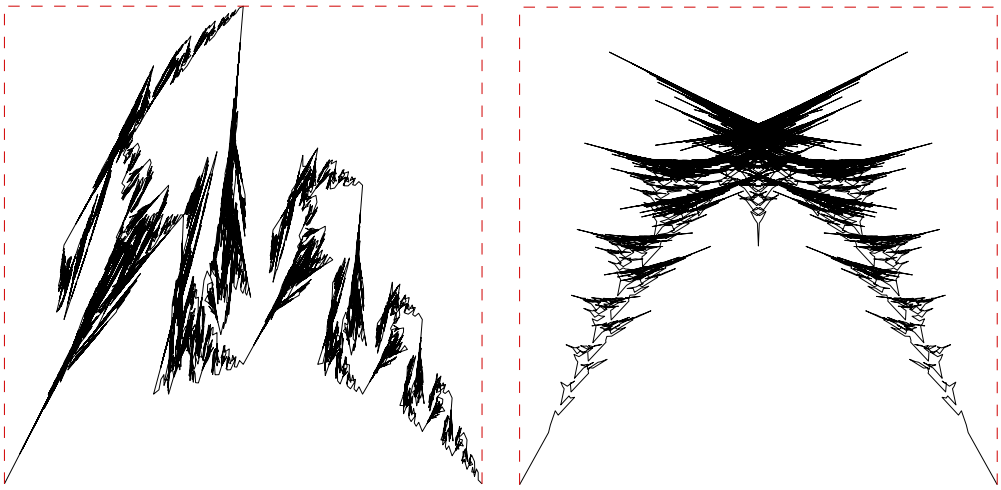
Surfaces

La figure 16 page 53 montre un exemple de surface fractale à pôles. Dans cet exemple, une grille de 5×5 points de contrôle a été utilisée.



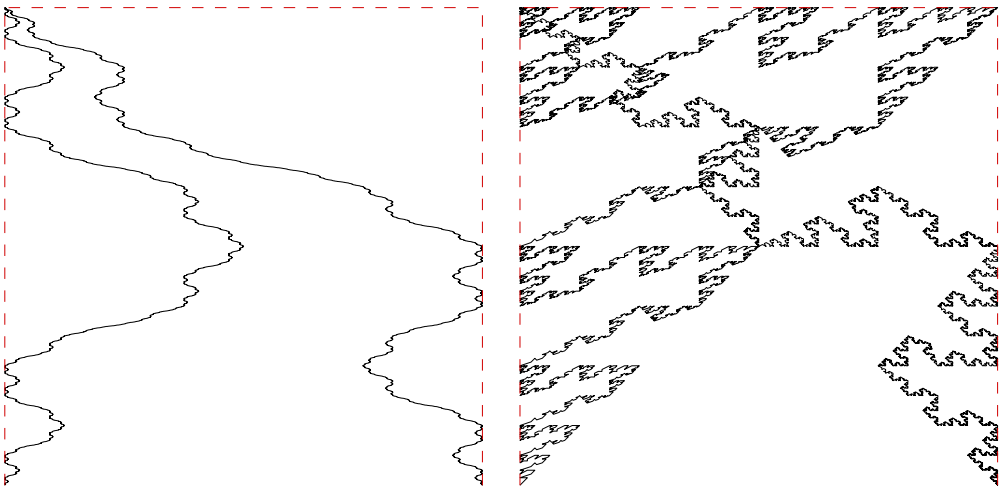
(a) (b)

FIG. 11 – Courbes symétriques



(a) (b)

FIG. 12 – Exemples de courbes



(a) (b)

FIG. 13 – Exemples de courbes

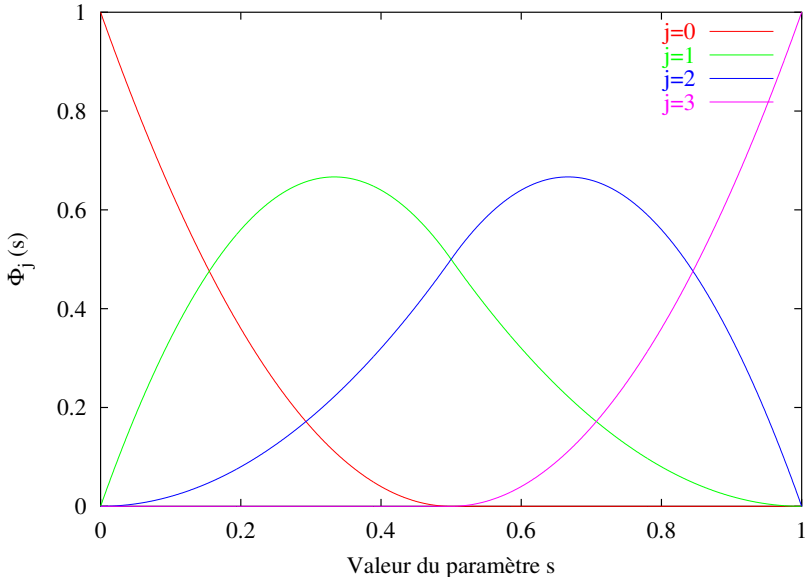


FIG. 14 – Exemple de fonction de mélange d'une B-spline

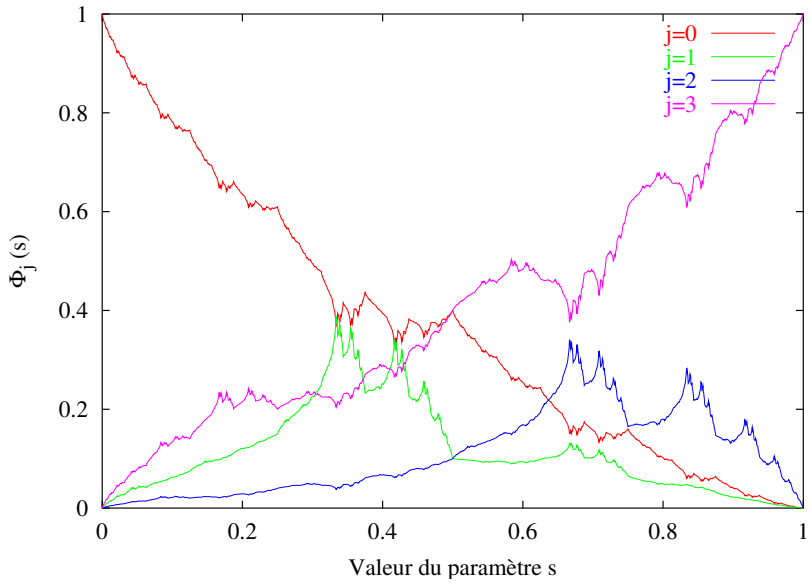


FIG. 15 – Exemple de fonction de mélange d'une courbe fractale

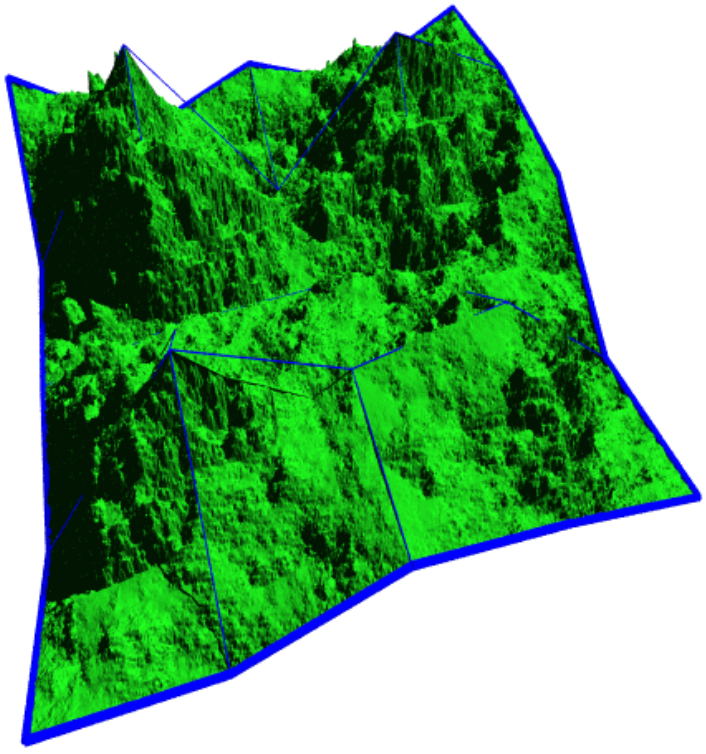


FIG. 16 – Exemple de surface fractale à pôles avec une grille de contrôle de 5×5 points de contrôle.

1.4 Calcul

1.4.1 Visualisation

Nous allons aborder ici le problème de la visualisation d'attracteurs d'IFS et d'attracteurs d'IFS projetés. Nous verrons que dans ce dernier cas, le procédé se rapproche de celui de la construction des courbes de Bézier par l'algorithme de De Casteljau.

À chaque IFS $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$ correspond un compact unique $\mathcal{A}(\mathbb{T})$ appelé attracteur. La méthode de construction de cet attracteur se déduit du théorème du point fixe. Le principe est de construire une suite qui converge vers l'attracteur

$$\lim_{n\to\infty} K_n = \mathcal{A}(\mathbb{T})$$

Le résultat général suivant permet de construire une telle suite :

Proposition 3 *Soient $(\mathcal{X}, d)$ un espace métrique complet et T une application contractante sur $\mathcal{X}$. On a :*

$$\forall \lambda \in \mathcal{X},\ \lim_{n\to\infty} T^n \lambda = c$$

où c désigne le point fixe de T. Autrement dit, toute suite définie par :

$$(a_n)_{n\in\mathbb{N}} = \left\{ \begin{array}{lcl} a_0 & = & \lambda \in \mathcal{X} \\ a_{n+1} & = & Ta_n, \forall n \geq 0 \end{array} \right.$$

converge vers c.

Pour appliquer ce procédé aux IFS, il suffit d'itérer l'opérateur de Hutchinson. On obtient le théorème fondamental de visualisation déterministe :

Théorème 5 (visualisation déterministe des IFS) *Soient $(\mathcal{X}, d)$ un espace métrique complet et $\mathbb{T}$ un IFS. Toute suite définie par :*

$$(K_n)_{n\in\mathbb{N}} = \left\{ \begin{array}{lcl} K_0 & = & K \in \mathcal{H}(\mathcal{X}) \\ K_{n+1} & = & \mathbb{T}K_n, \forall n \geq 0 \end{array} \right.$$

converge vers l'attracteur de l'IFS $\mathcal{A}(\mathbb{T})$.

Le procédé est donc simple, n'importe quel compact de départ produira le même résultat. En pratique, on prend un compact pour lequel les transformations sont faciles à calculer et qui présente un bon rendu visuel, une sphère par exemple pour $\mathcal{X} = \mathbb{R}^3$. Ce principe s'apparente à la construction récursive d'un arbre. La figure 17 page suivante montre l'exemple d'un arbre associé à la visualisation d'un IFS.

Intéressons nous maintenant au cas des attracteurs d'IFS projetés. L'exposé est fait pour les courbes, mais il suffit de remplacer le mot polygone par grille pour l'étendre aux surfaces. Sous sa forme paramétrique, un attracteur projeté est décrit par :

$$\begin{array}{rcl} F(s) & = & P\Phi(s) \\ & = & P\phi(\rho) \\ & = & P \lim_{j\to\infty} T_{\rho_1} \ldots T_{\rho_j} \lambda \end{array}$$

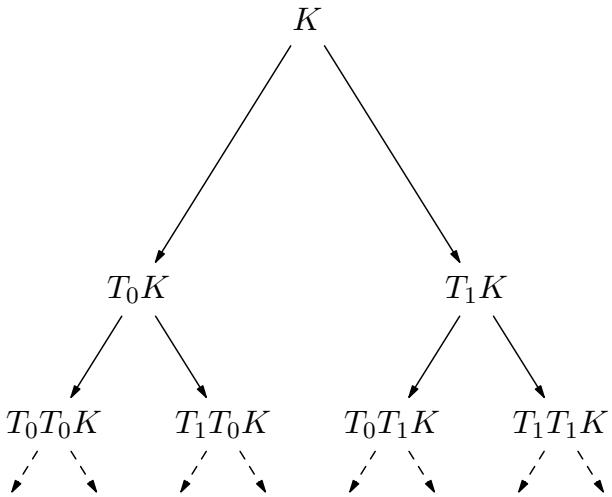


FIG. 17 – Principe de visualisation d'un IFS

Or l'action de P est affine, on peut donc le faire entrer dans la limite :

$$F(s) = \lim_{j\to\infty} PT_{\rho_1} \dots T_{\rho_j}\lambda$$

On peut déduire de cette formule un procédé de construction des attracteurs projetés qui consiste à construire une suite d'ensembles de polygones en appliquant l'IFS à droite, directement sur les points de contrôle :

Proposition 4 (visualisation d'attracteurs d'IFS projetés) *Soient $(\mathcal{X}, d)$ un espace métrique complet, $\mathbb{T}$ un IFS, et P un vecteur de points de contrôle. Soit la suite définie par :*

$$(S_n)_{n\in\mathbb{N}} = \begin{cases} S_0 & = & \{P\} \\ S_{n+1} & = & S_n\mathbb{T}, \forall n \geq 0 \end{cases}$$

Alors, on a :

$$\lim_{n\to\infty} S_nK = P\mathcal{A}(\mathbb{T})$$

avec $K \in \mathcal{H}(\mathcal{X})$.

Remarque : S_n est un ensemble fini de polygones que l'on peut développer de manière récursive grâce à un arbre (voir figure 18 page suivante) :

$$S_n = P\mathbb{T}^n = \{PT_{\rho_1} \dots T_{\rho_n} | \, |\rho| = n\}$$

Par la suite, c'est de cette façon que l'on construira les courbes fractales.

Cette façon de combiner les points de contrôle récursivement est identique au procédé récursif de construction des courbes de BÉZIER par la méthode de DE CASTELJAU[dC86]. Dans la pratique, on prendra pour K une primitive (un segment, une facette ou un polygone de $\mathcal{X}$). Il est donc possible de voir se dessiner progressivement, et avec de plus en plus de précision, les différentes étapes de construction d'une courbe fractale à pôles.

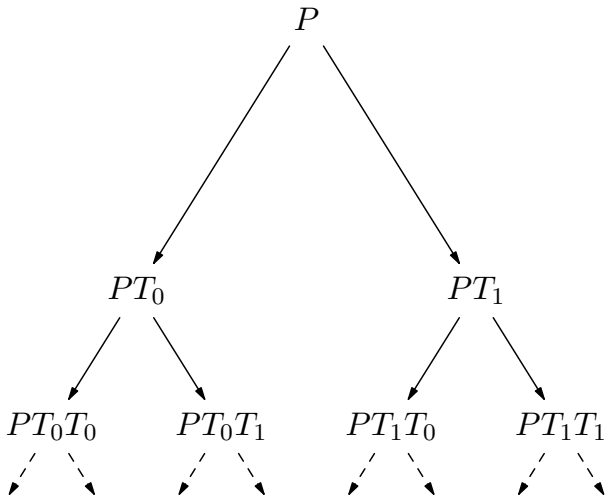


FIG. 18 – Arbre de construction d'un IFS projeté

La figure 19 page suivante illustre ce processus en montrant les différents polygones $PT_{\rho_1}T_{\rho_n}K$ composant S_nK des 4 premières itérations (figures 19a à 19e). La courbe finale, ici obtenue avec 12 itérations est aussi explicitée (figure 19f).

1.4.2 Tabulation (discrétisation)

Nous allons maintenant étudier comment il est possible de calculer des valeurs exactes de la fonction fractale avec un nombre fini et limité d'itérations. Les valeurs du paramètre pour lesquelles nous allons calculer la fonction sont celles qui sont multiples de $\frac{1}{N^p}$, où p est une profondeur déterminée à l'avance :

$$F(\frac{i}{N^p}) = P\Phi(\frac{i}{N^p})$$

avec $i \in \{0, \ldots, N^p\}$.

Dans ce cas particulier, le développement en base N du paramètre se termine par une suite infinie de 0, à partir du symbole $p+1$:

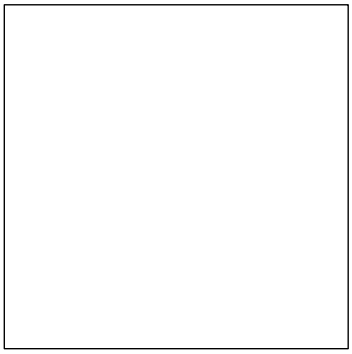
$$\rho = \rho_1 \ldots \rho_p 0^\omega$$

Alors, le calcul de la fonction paramétrée se simplifie en :

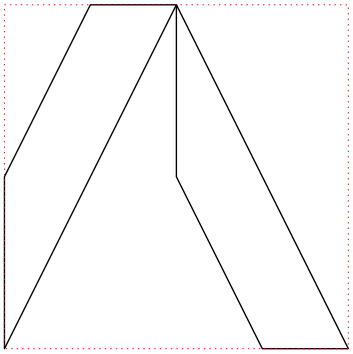
$$\begin{array}{rcl} F(\frac{i}{N^p}) & = & P\phi(\rho) \\ & = & PT_{\rho_1} \ldots T_{\rho_p}\phi(0^\omega) \\ & = & PT_{\rho_1} \ldots T_{\rho_p}\Phi(0) \end{array}$$

Nous allons faire l'hypothèse que $\Phi(0) = e_0$ où e_0 désigne le premier vecteur de base de $\mathcal{X}$. En d'autres termes, celle-ci signifie que :

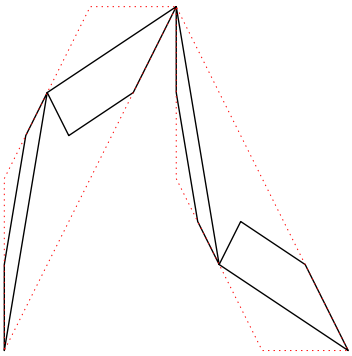
$$\Phi(0) = \lim_{n\to\infty} T_0^n \lambda = e_0$$



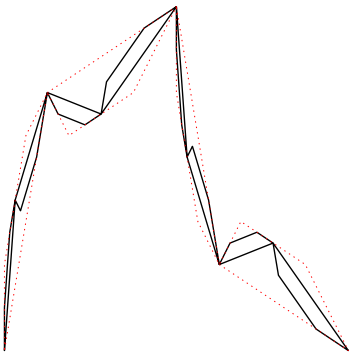
(a) Polygone de contrôle initial : $S_0K = PK$



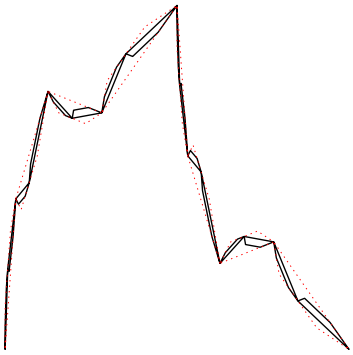
(b) Après une itération : $S_1K = P\mathbb{T}K$



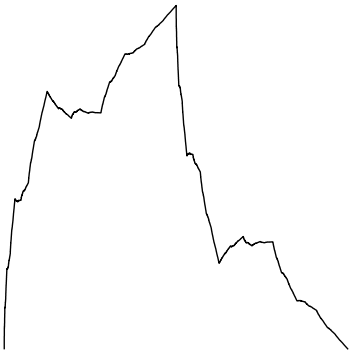
(c) Après 2 itérations : $S_2K = P\mathbb{T}^2K$



(d) Après 3 itérations : $S_3K = P\mathbb{T}^3K$



(e) Après 4 itérations : $S_4K = P\mathbb{T}^4K$



(f) Courbe finale obtenue avec 12 itérations

FIG. 19 – Les étapes récursives de construction d'une courbe fractale à pôles

donc que le point fixe de la transformation T_0 est e_0. Cela signifie aussi que :

$$\begin{aligned} F(0) &= PT_{\rho_1} \dots T_{\rho_p} \phi(0^\omega) \\ &= PT_0 \dots T_0 \dots \lambda \\ &= Pe_0 \end{aligned}$$

Ainsi, le premier point de la courbe, celui pour lequel le paramètre vaut 0, est aussi le premier point de contrôle. Cette hypothèse permet un meilleur contrôle de la courbe, puisque celle-ci passe par le premier point de contrôle. Tous les exemples donnés dans la section précédente respectent cette hypothèse. On prendra soin d'ajouter l'hypothèse symétrique qui précise que $F(1)$ coïncide avec le dernier point de contrôle. Ainsi, les valeurs *exactes* de la fonction peuvent être calculées avec un nombre réduit d'itérations.

Proposition 5 (tabulation) *Soit un IFS projeté défini par le couple* $(P, \mathbb{T})$ *et vérifiant les conditions suivantes :*

$$\begin{cases} \Phi(0) &= e_0 \\ \Phi(1) &= e_m \end{cases}$$

Les valeurs exactes de la tabulation de F *peuvent être calculées de la façon suivante :*

$$F(\frac{i}{N^p}) = \begin{cases} PT_{\rho_1} \dots T_{\rho_p} e_0 \text{ pour } i \in \{0, \dots, N^p - 1\} \\ Pe_m \text{ pour } i = N^p \end{cases}$$

Ce résultat a une grande importance, il va permettre de calculer rapidement des valeurs exactes de la fonction fractale, sans avoir à faire un grand nombre d'itérations. Il garantit donc qu'un minimum de calculs va être effectué, sachant que la méthode d'approximation demande un nombre important d'appels à ce calcul.

Remarque : En traitement du signal on parle d'échantillonnage de la fonction plutôt que de tabulation. Il s'agit exactement du même concept : représentation d'une fonction continue par un ensemble fini de valeurs.

Pour les surfaces, on dispose d'une formule équivalente :

$$F(\frac{i}{N^p}, \frac{j}{N^p}) = \begin{cases} PT_{\sigma_1 \natural \tau_1} \dots T_{\sigma_p \natural \tau_p} e_0 \text{ pour } i, j \neq N^p \\ Pe_m \text{ pour } i = j = N^p \end{cases}$$

1.5 Conclusion

La modélisation par IFS donne une base formelle assez simple pour décrire des formes fractales. En la fusionnant avec une technique issue de la modélisation géométrique, les formes à pôles, il est possible d'obtenir plus de contrôle sur la forme : c'est la principe des IFS projetés. Grâce à ces rappels, nous avons résumé les concepts principaux de la modélisation par IFS ainsi que par IFS projetés. Le chapitre suivant introduit de nouvelles notions et expose la manière dont nous procédons pour adapter ce modèle à nos besoins d'approximation.

Chapitre 2

Modèles de courbes et surfaces

Le chapitre précédent nous a permis de prendre connaissance du modèle général que nous allons utiliser pour approximer les formes rugueuses. Nous allons maintenant expliciter les modèles de courbes et surfaces que nous utiliserons. Nous allons ainsi définir des familles paramétrées de courbes et surfaces. En préliminaire, nous nous intéressons au problème du raccord de courbes et de surfaces : quelle est la condition pour qu'un IFS projeté soit une courbe ou une surface ? Nous abordons ensuite le problème de l'équivalence d'IFS projetés : deux IFS projetés différents qui génèrent la même figure. En combinant la condition de raccord et cette représentation multiple, il est possible de simplifier les matrices d'IFS. En allant au-delà de ces contraintes, on peut simplifier encore plus le modèle et rendre l'approximation plus performante. Enfin, nous expliquons comment il est possible de faire une paramétrisation du modèle, c'est à dire construire une famille de courbes ou surfaces fractales.

2.1 Représentation du modèle

2.1.1 Condition de raccord

Pour l'instant, nous avons défini la projection d'un attracteur d'IFS en nous plaçant dans un espace barycentrique. Mais rien ne garantit que cet attracteur projeté décrit une courbe ou une surface. En effet, la seule chose dont on soit sûr est qu'il s'agit d'un compact. Il faut donc ajouter des contraintes sur les transformations des IFS pour que l'attracteur soit bien une courbe ou une surface. L'ensemble de ces contraintes sera regroupé sous le nom générique de *condition de raccord*.

L'idée générale est que les raccords se font sur un objet topologique de dimension inférieure à celle de l'objet. Pour les courbes, le raccord se fait en un point et est très facile à formaliser. Pour les surfaces, il se fait le long d'une courbe, et nécessite l'introduction de plongements d'IFS.

Remarque : La condition de raccord concerne uniquement les transformations de l'IFS. On constatera assez facilement que si un attracteur décrit une courbe ou une surface, sa projection aussi. La réciproque est fausse.

Raccord de courbes

Considérons un IFS $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$ dont chacune des transformations T_i a pour point fixe c_i. Si l'IFS satisfait la condition suivante, alors son attracteur décrit une courbe (la démonstration figure dans la thèse de Chems Eddine Zair [Zai98]) :

$$T_i c_{N-1} = T_{i+1} c_0, \forall i = 0, \ldots, N-2$$

Intuitivement, cela implique que deux mots infinis $\sigma = i(N-1)^\omega$ et $\tau = (i+1)0^\omega$ qui sont le développement d'un même scalaire $s = \frac{i+1}{N}$ en base N, décrivent bien le même point de l'attracteur :

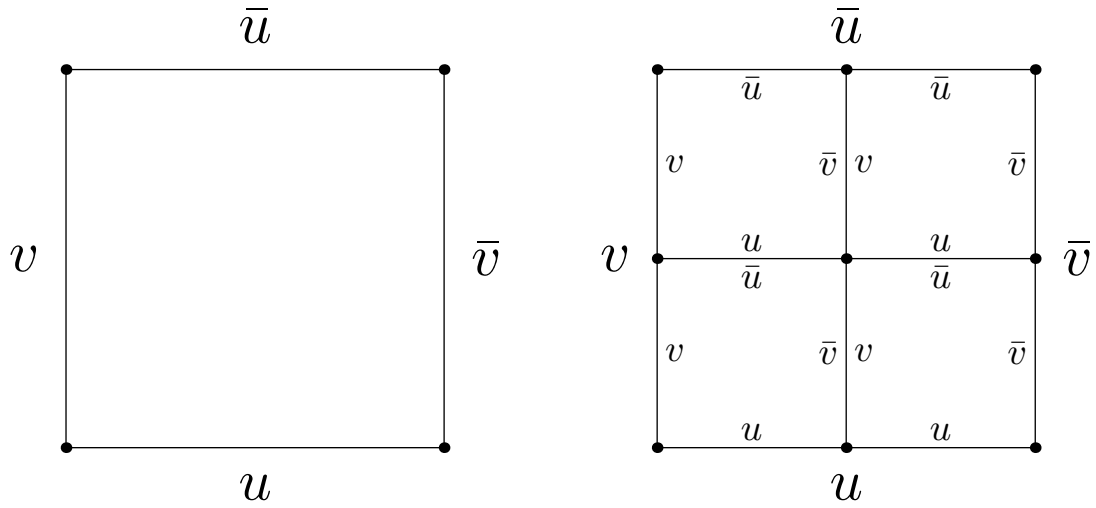
$$\Phi(s) = \phi(\sigma) = \phi(\tau)$$

Ainsi, les paramétrisations formelle et numérique se correspondent.

Raccord de surfaces

Classiquement, les carreaux de surface à pôles quadrangulaires sont obtenus par produits tensoriels d'arcs de courbes. Cette construction se généralise aux carreaux définis par IFS [MP87, Mas94, ZT96]. On définit ainsi un IFS de surface par produit tensoriel d'IFS d'arcs de courbes. Mais les produits tensoriels d'IFS d'arcs sont en fait des cas particuliers d'IFS de carreaux. Dans le cas général, il suffit que l'IFS vérifie une condition de raccord déduite de la structure de subdivision du pavé $[0,1]^2$ pour que le carreau paramétré soit défini.

Soit un IFS de subdivision quadrangulaire (voir figure 20).



(a) Bords simples

(b) Bords subdivisés avec leurs raccords

Fig. 20 – Bords associés aux subdivisions quadrangulaires

Les courbes qui bordent le carreau s'écrivent de la façon suivante :

$$\begin{array}{rcl} \Phi_u(s) & = & \Phi(s,0) \\ \Phi_{\bar{u}}(s) & = & \Phi(s,1) \\ \Phi_v(s) & = & \Phi(0,s) \\ \Phi_{\bar{v}}(s) & = & \Phi(1,s) \end{array}$$

Ces courbes sont définies par des IFS $\mathbb{T}^\gamma$ dont les indices sont extraits de ceux de l'IFS du carreau :

$$T_i^\gamma = T_{\xi_i^\gamma}$$

où les fonctions ξ^γ sont des correspondances d'indices :

$$\begin{array}{rcl} \xi_i^u & = & (i,0) \\ \xi_i^{\bar{u}} & = & (i,N-1) \\ \xi_i^v & = & (0,i) \\ \xi_i^{\bar{v}} & = & (N-1,i) \end{array}$$

pour tout $i = 0, \ldots, N-1$.

On peut étendre les fonctions ξ^γ aux mots :

$$\begin{array}{rccl} \xi^\gamma : & \tilde{\Sigma}^\omega & \rightarrow & \Sigma^\omega \\ & \rho & \mapsto & \xi^\gamma_{\rho_1} \ldots \xi^\gamma_{\rho_n} \ldots \end{array}$$

Les courbes qui bordent le carreau peuvent alors être directement calculées à partir de la fonction d'adressage du carreau :

$$\Phi_\gamma(s) = \phi^\gamma(\rho) = \phi(\xi^\gamma(\rho))$$

avec $\rho_i = \sigma_i \natural \tau_i$ et σ, τ les développements en base N de s et t.

Proposition 6 (raccord de surface quadrangulaire) *La condition pour que la fonction $\Phi : (s,t) \mapsto \phi(\rho)$ soit définie et continue est que les composantes $T_{i,j}\phi$ de la fonction d'adressage ϕ se raccordent le long de leurs bords $T_{i,j}\phi^\gamma$. Cette condition s'écrit sous la forme d'un système d'équations :*

$$\left\{ \begin{array}{lllll} T_{i\natural j}\phi^{\bar{u}}(\rho) = T_{i\natural j+1}\phi^u & \text{pour} & i = 0, \ldots, N-1 & \text{et} & j = 0, \ldots, N-2 \\ T_{i\natural j}\phi^{\bar{v}}(\rho) = T_{i+1\natural j}\phi^u & \text{pour} & i = 0, \ldots, N-2 & \text{et} & j = 0, \ldots, N-1 \end{array} \right.$$

La preuve est fournie dans [Tos99].

Ces contraintes peuvent être exprimées linéairement sur les matrices $T_{i,j}$. Pour cela on suppose que l'arc extrait Φ_γ est inclus dans un sous-espace associé à une partie J_γ de la grille J. Il peut alors s'écrire comme le plongement d'un arc défini dans un espace plus petit :

$$\phi_\gamma(\rho) = \amalg_\gamma \phi^\gamma(\rho)$$

En notant $\tilde{T}_i^\gamma$ les matrice de subdivision des arcs ϕ_γ, les T_i vérifient des équations supplémentaires :

$$T_{\gamma_i} \amalg_\gamma = \amalg_\gamma \tilde{T}_i^\gamma$$

Les équations de raccord se ramènent alors à des contraintes linéaires sur les T_i et les $\tilde{T}_i$ [Tos99] :

$$\begin{array}{rcl} T_{i\natural j}\phi^{\bar{u}} = T_{i+1\natural j}\phi^{u} & \Leftrightarrow & T_{i\natural j}\amalg_{\bar{u}}\tilde{\phi}^{\bar{u}} = T_{i+1\natural j}\amalg_{u}\phi^{u} \\ & \Leftrightarrow & T_{i\natural j}\amalg_{\bar{u}} = T_{i+1\natural j}\amalg_{u} \text{ et } \tilde{T}^{\bar{u}} = \tilde{T}^{\amalg}_{u} \end{array}$$

On déduit de la condition de raccord quadrangulaire le système d'équations suivant :

$$\left\{ \begin{array}{rcl} T_{i\natural j}\amalg_{\bar{u}} & = & T_{i+1\natural j}\amalg_{u}, \forall i = 0, \ldots, N-2 \\ T_{i\natural j}\amalg_{\bar{v}} & = & T_{i\natural j+1}\amalg_{v}, \forall j = 0, \ldots, N-2 \\ \tilde{T}^{u} & = & \tilde{T}^{\bar{u}} \\ \tilde{T}^{v} & = & \tilde{T}^{\bar{v}} \end{array} \right.$$

2.1.2 Équivalence entre IFS projetés

Nous allons voir dans cette section qu'un attracteur d'IFS projeté est défini à un changement de base près. Cela signifie qu'il existe une relation d'équivalence entre descripteurs d'IFS projetés. Cette notion est déduite de celle de matrices semblables. Dans la suite, nous appelerons *IFS projeté* un couple $(P, \mathbb{T})$ avec P polygone ou grille de contrôle et $\mathbb{T}$ IFS défini dans l'espace barycentrique associé. $(P, \mathbb{T})$ sert de descripteur à l'IFS projeté $P\mathcal{A}(\mathbb{T})$ et à la forme à pôles fractale $P\Phi(s)$.

Définition 11 (équivalence de deux IFS projetés) Soient deux IFS projetés définis par les couples $(P, \mathbb{T})$ et $(P', \mathbb{T}')$. L'ensemble des indices J est le même pour les deux modèles. On dit que $(P, \mathbb{T})$ est équivalent à $(P', \mathbb{T}')$ s'il existe une matrice dite de passage M, inversible, telle que :

$$\left\{ \begin{array}{rcl} T'_j & = & M^{-1}T_jM, \ \forall j \in J \\ P' & = & PM \end{array} \right.$$

De cette définition découle une propriété importante des IFS projetés.

Proposition 7 *Deux IFS projetés équivalents décrivent le même attracteur projeté.*

Preuve L'attracteur d'IFS projeté associé à $(P', \mathbb{T}')$ s'écrit de la façon suivante :

$$\begin{array}{rcl} P'\mathcal{A}(\mathbb{T}') & = & P' \lim\limits_{n\to\infty} \mathbb{T}'^{n}K' \\ & = & PM \lim\limits_{n\to\infty} \left(M^{-1}\{T_0, \ldots, T_{N-1}\}M\right)^{n} K' \\ & = & PMM^{-1} \lim\limits_{n\to\infty} \mathbb{T}^{n}MK' \end{array}$$

On peut choisir n'importe quel compact de $(\mathcal{X}, d)$ pour K'. Prenons $K' = M^{-1}K$, ce qui est toujours possible car M est inversible. On a alors :

$$\begin{array}{rcl} P'\mathcal{A}(\mathbb{T}') & = & PMM^{-1} \lim\limits_{n\to\infty} \mathbb{T}^{n}MM^{-1}K \\ & = & P \lim\limits_{n\to\infty} \mathbb{T}^{n}K \\ & = & P\mathcal{A}(\mathbb{T}) \end{array}$$

■

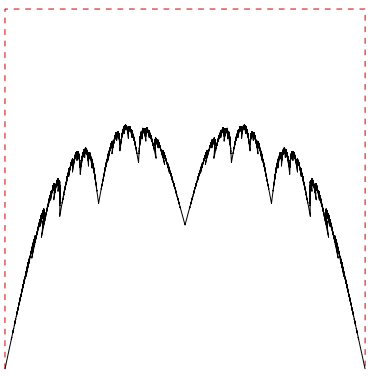
Remarque : Les relations suivantes sont aussi vérifiées :

$$\begin{aligned} \phi'(\rho) &= M^{-1}\phi(\rho) \\ F'(s) &= F(s) \end{aligned}$$

et se démontrent de la même manière.

Il existe donc une relation d'équivalence sur les descripteurs. La figure 21 illustre cette propriété en montrant deux attracteurs d'IFS projetés identiques qui ont été générés par deux descripteurs différents, c'est à dire des points de contrôle et des matrices de subdivision différents. La matrice de passage qui a été utilisée est la suivante :

$$M = \begin{pmatrix} 1 & 0 & 0 & -0.25 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1.25 \end{pmatrix}$$

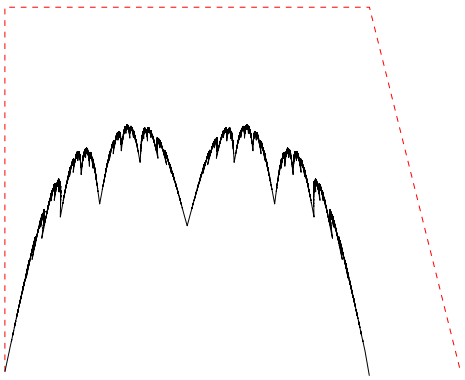


$$P = \begin{pmatrix} -1 & -1 & 1 & 1 \\ 0 & 2 & 2 & 0 \end{pmatrix}$$

$$T_0 = \begin{pmatrix} 1 & 0.2 & 0.05 & 0.3 \\ 0 & 0.7 & 0.5 & 0.2 \\ 0 & 0.1 & 0.4 & 0.2 \\ 0 & 0 & 0.05 & 0.3 \end{pmatrix}$$

$$T_1 = \begin{pmatrix} 0.3 & 0.05 & 0 & 0 \\ 0.2 & 0.4 & 0.1 & 0 \\ 0.2 & 0.5 & 0.7 & 0 \\ 0.3 & 0.05 & 0.2 & 1 \end{pmatrix}$$

(a) Le modèle initial



$$P' = \begin{pmatrix} -1 & -1 & 1 & 1.5 \\ 0 & 2 & 2 & 0 \end{pmatrix}$$

$$T'_0 = \begin{pmatrix} 1 & 0.2 & 0.06 & 0.2 \\ 0 & 0.7 & 0.5 & 0.25 \\ 0 & 0.1 & 0.4 & 0.25 \\ 0 & 0 & 0.04 & 0.3 \end{pmatrix}$$

$$T'_1 = \begin{pmatrix} 0.36 & 0.06 & 0.04 & 0.16 \\ 0.2 & 0.4 & 0.1 & -0.05 \\ 0.2 & 0.5 & 0.7 & -0.05 \\ 0.24 & 0.04 & 0.16 & 0.94 \end{pmatrix}$$

(b) Après le changement de base

FIG. 21 – Deux attracteurs d'IFS projetés équivalents

Nous allons voir dans ce chapitre comment on peut rendre le modèle plus simple, afin d'optimiser le processus d'approximation mais aussi de diminuer la taille du descripteur associé au modèle.

2.1.3 Principe général de simplification

Dans tout IFS projeté $(P, \mathbb{T})$, s'il y a $m+1$ points de contrôle, alors chaque transformation est une matrice $(m+1) \times (m+1)$. Lorsque m augmente, on améliore la maniabilité et les potentialités de la forme, mais le nombre de paramètres nécessaires pour décrire les matrices de subdivision devient vite prohibitif. Pour répondre à ce problème, il y a plusieurs manières de réduire le nombre de paramètres des matrices. La première consiste à bien formuler les conditions de raccord des IFS. La seconde consiste à choisir de rendre locale l'action des points de contrôle. Cela signifie que bouger un point de contrôle ne va pas forcément modifier toute la forme, mais seulement une partie proche de ce point. Ce procédé est utilisé, entre autres, pour les B-splines. Comment mettre en place ce genre de mécanisme ? Il faut introduire des contraintes d'interpolation et de dépendance des nouveaux points dans les matrices de subdivision. Nous distinguons, pour plus de clarté, le cas des courbes de celui des surfaces.

Courbes

Nous avons vu dans le chapitre précédent qu'il était plus simple, pour des raisons d'affichage (possibilité de tabulation ou échantillonnage), de considérer que le premier point de la courbe coïncide avec le premier point de contrôle. De la même façon, le dernier point de la courbe coïncide avec le dernier point de contrôle. Cela entraîne des simplifications dans les matrices. Autrement dit, la première colonne de T_0 sera $(1, 0, \ldots, 0)^T$ et la dernière colonne de T_{N-1} sera $(0, \ldots, 0, 1)^T$. La condition de raccord des courbes se traduit par une égalité de colonnes :

$$T_0 e_0 \ ; \ T_{N-1} e_m = e_m \ ; \ T_0 e_m = T_{N-1} e_0$$

Ces simplifications n'impliquent pas pour le moment de réduction de l'ensemble de modélisation.

Nous allons introduire des simplifications supplémentaires, qui réduisent le nombre de paramètres des matrices de subdivision, mais en réduisant aussi le domaine de modélisation.

Prenons l'exemple d'une courbe fractale à pôles ayant 4 points de contrôle et 2 transformations. La forme générale des transformations est la suivante :

$$T_0 = \begin{pmatrix} 1 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} \text{ et } T_1 = \begin{pmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 1 \end{pmatrix}$$

où les $*$ désignent des valeurs non nulles. Si l'on réduit la forme générale à celle-ci :

$$T_0 = \begin{pmatrix} 1 & * & * & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ et } T_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ 0 & * & * & 1 \end{pmatrix}$$

le nombre de paramètres est réduit, et l'action des points de contrôle est plus locale. On notera que la dernière colonne de T_0 et la première de T_1 doivent être identiques pour des raisons de raccord. La figure 22 montre la différence entre un modèle avec des matrices « pleines » et un modèle avec des matrices simplifiées. La figure 22a illustre le cas général où la matrice ne comporte pas de simplifications. Dans ce cas, le fait de bouger un point de contrôle a une influence sur toute la courbe (la flèche indique le mouvement du point de contrôle). À l'inverse, la figure 22b montre le cas où les matrices de subdivision ont été simplifiées. L'action du point de contrôle extrême est localisée sur la moitié de la courbe, c'est à dire la partie de la courbe pour laquelle le paramètre est supérieur à $0,5$. On peut démontrer assez aisément que si $s \in [\frac{i}{N}, \frac{i+1}{N}]$, son développement en base N s'écrit $i\,\rho_1 \dots \rho_n \dots$, on a alors :

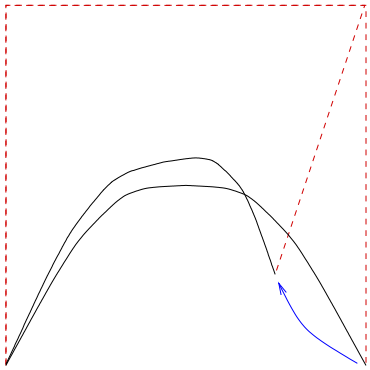
$$F(s) = P\phi(i\,\rho) = PT_i \lim_{n\to\infty} T_{\rho_1} \dots T_{\rho_n} \lambda \in PT_i \mathcal{B}^J$$

Dans le cas où T_i possède des lignes nulles, PT_i dépend d'un sous-ensemble de $\{p_0, \dots, p_n\}$.Pour l'exemple donné dans la figure 22b, nous avons $N = 2$, et lorsque $s \in [0, \frac{1}{2}]$, $F(s)$ ne dépend que de p_0, p_1 et p_2. Ainsi, le fait de bouger p_3 n'influe pas sur la moitié de la courbe.

Surfaces

Pour les surfaces, il existe aussi des conditions de raccord, mais celles-ci sont plus complexes. Dans la section 2.1.1 page 59, nous avons mis en évidence celles-ci de façon formelle. Nous allons maintenant le faire de façon plus intuitive, c'est à dire visuellement. Pour cela, plaçons nous dans le cas le plus simple où le nombre de transformations est 2×2 et le nombre de points de contrôle est 3×3. Ce cas nous amène à considérer des matrices de subdivision de taille 9×9. Comme pour les courbes, le raccord va s'exprimer par une contrainte d'égalité de colonnes entre ces matrices.

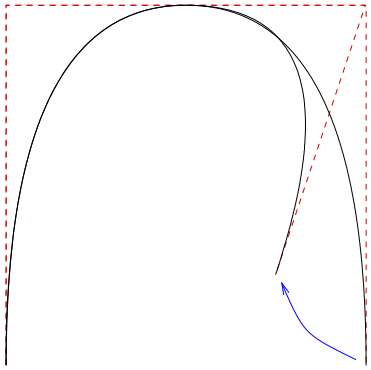
La figure 23 montre de manière intuitive comment les raccords vont être effectués. Ainsi, un exemple de contrainte de raccord sera l'égalité de la colonne 8 de T_0 et de la colonne 2 de T_2. Cet exemple est matérialisé dans la figure 23b par la couleur rouge. On peut qualifier ce type de contrainte entre les matrices de subdivision de « premier ordre ». En effet, on voit facilement que s'il n'est pas satisfait dès la première subdivision la surface ne sera pas raccordée. Il existe un autre type de contrainte qui implique les subdivisions suivantes. En effet, pour que le raccord soit effectué, il faut que les arcs de courbes qui bordent les carreaux se raccordent. La figure 23b montre en bleu un exemple de ce type de raccord. Ici encore, des égalités entre colonnes de matrices vont en découler, mais avec avec une certaine mise en forme des indices. Il ne s'agit donc pas d'une égalité de colonnes au sens strict. Pour satisfaire cette contrainte, il est plus aisé de considérer



$$T_0 = \begin{pmatrix} 1 & 0.4 & 0.25 & 0.25 \\ 0 & 0.4 & 0.4 & 0.25 \\ 0 & 0.1 & 0.25 & 0.25 \\ 0 & 0.1 & 0.1 & 0.25 \end{pmatrix}$$

$$T_1 = \begin{pmatrix} 0.25 & 0.1 & 0.1 & 0 \\ 0.25 & 0.25 & 0.1 & 0 \\ 0.25 & 0.4 & 0.4 & 0 \\ 0.25 & 0.25 & 0.4 & 1 \end{pmatrix}$$

(a) Influence globale des points de contrôle

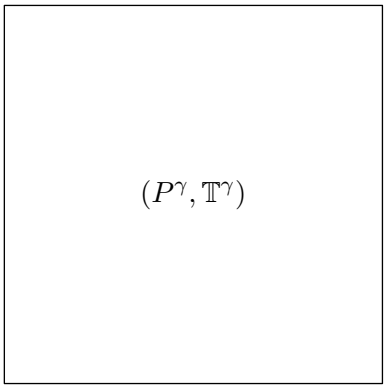


$$T'_0 = \begin{pmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.75 & 0.5 \\ 0 & 0 & 0.25 & 0.5 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

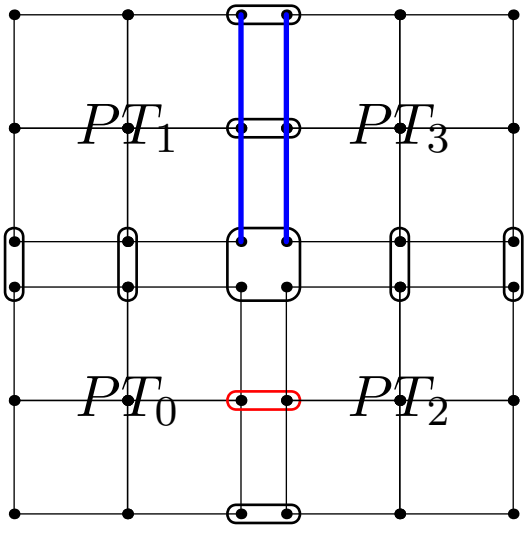
$$T'_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0.25 & 0 & 0 \\ 0.5 & 0.75 & 0.5 & 0 \\ 0 & 0 & 0.5 & 1 \end{pmatrix}$$

(b) Influence locale des points de contrôle

FIG. 22 – Comparaison de l'action globale et locale des points de contrôle



(a) La grille de contrôle



(b) La grille de contrôle après application des matrices de subdivision

FIG. 23 – Raccord intuitif de surfaces

que les nouveaux points qui bordent les carreaux ne dépendent que des anciens points qui le bordent déjà. Cela revient encore une fois à simplifier les matrices de subdivision en y insérant des zéros.

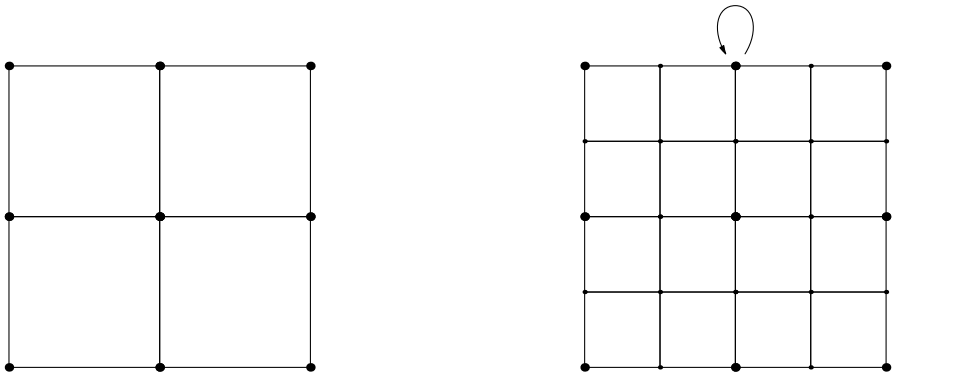
À tout cela nous pouvons ajouter le principe de localisation de l'action des points de contrôle, comme pour les courbes. Ce procédé se traduit par l'égalité de certains coefficients dans la matrice de subdivision à la valeur zéro.

En poussant le raisonnement jusqu'au bout, il apparaît plus simple alors de donner l'IFS projeté sous la forme d'un *schéma de subdivision*. Pour rendre cela possible il faut supposer que le nombre de transformations est le même que le nombre de points de contrôle, c'est à dire :

$$N = m$$

Cela signifie que l'on a un modèle avec une grille de $(m+1)^2$ points de contrôle et m^2 transformations. Dans ces conditions, on peut forcer la grille subdivisée à inclure la grille non-subdivisée. Cela va se traduire par des colonnes dans les matrices de subdivision qui ne possèdent qu'une valeur non nulle égale à 1. Les figures 24 et 25 page suivante illustrent ce principe. On retrouve les points de la grille initiale (voir figure 24b).

Les points qui sont alignés avec la grille initiale sont combinaison barycentrique de trois points (voir figure 25a). Les autres points sont une combinaison barycentrique de quatre points (voir figure 25b). L'utilisation de schémas de subdivision permet de réduire



(a) Grille de points initiale

(b) Exemple de point restant inchangé

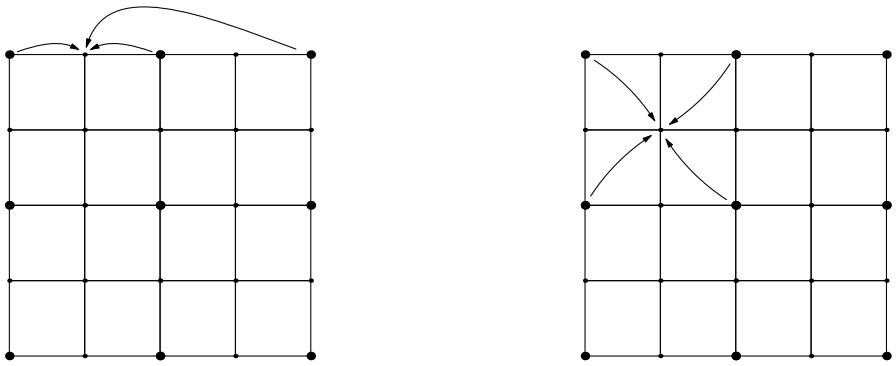
FIG. 24 – Schéma de subdivision pour la construction de surfaces

considérablement la complexité du modèle. Dans le cas ou $m = N = 2$ par exemple, il ne faudra que 28 paramètres pour coder les transformations :

- 16 pour les points combinaison de 3 autres points (8×2)
- 12 pour les points combinaison de 4 autres points (4×3)

Il faudra aussi 9 paramètres pour coder les scalaires de contrôle, soit un total de 37 paramètres. Dans la version la plus générale du modèle à 3×3 points de contrôle et 2×2 transformations, il aurait fallu environ 350 paramètres (8×9 par transformation).

Ainsi, trois éléments majeurs rentrent en compte dans la simplification du modèle :



(a) Exemple de point combinaison de 3 points de la grille initiale

(b) Exemple de point combinaison de 4 points de la grille initiale

FIG. 25 – Schéma de subdivision pour la construction de surfaces

- Condition de raccord. Elle est obligatoire et entraîne des identifications entre les colonnes de matrices de subdivision.
- Représentation « canonique ». Du fait que les IFS projetés sont définis à un changement de base près, il est possible de faire intervenir des contraintes qui ne nuisent pas à la généralité du modèle. Ces contraintes impliquent des simplifications dans les matrices de subdivision (des zéros).
- Localisation de l'action. De la même façon que les B-splines sont des arcs de BÉZIER raccordés entre eux, il est possible d'obtenir des courbes ou surfaces dont le degré est nettement inférieur aux nombre de points de contrôle. Ceci se traduit par des matrices de subdivision singulières, présentant des lignes nulles.

Dans le cas des surfaces, une bonne manière de mettre en œuvre ces simplifications est de coder le modèle grâce à un schéma de subdivision.

2.2 Paramétrisation du modèle

Les coefficients des matrices de subdivision ainsi que les coordonnées des points de contrôle, sont des scalaires. Il est donc possible de décrire une instance d'un modèle d'attracteur d'IFS projeté par la donnée d'un vecteur de paramètres. Appelons **a** ce vecteur de paramètres. La longueur de celui-ci dépendra de la complexité du modèle générique employé. Plus le modèle possèdera de points de contrôle ou de transformations, plus le vecteur de paramètres **a** contiendra d'éléments. Prenons l'exemple d'une courbe fractale à pôles composée de deux transformations et 4 points de contrôle. On pourra faire le choix d'étiquetage suivant :

$$T_0 = \begin{pmatrix} 1 & a_0 & a_3 & a_6 \\ 0 & a_1 & a_4 & a_7 \\ 0 & a_2 & a_5 & a_8 \\ 0 & 1-a_0-a_1-a_2 & 1-\ldots & 1-\ldots \end{pmatrix} \quad T_1 = \begin{pmatrix} a_9 & a_{12} & a_{15} & 0 \\ a_{10} & a_{13} & a_{16} & 0 \\ a_{11} & a_{14} & a_{17} & 0 \\ 1-\ldots & 1-\ldots & 1-\ldots & 1 \end{pmatrix}$$

et

$$P = \begin{pmatrix} a_{18} & a_{20} & a_{21} & a_{23} \\ a_{19} & a_{21} & a_{22} & a_{24} \end{pmatrix}$$

Dans ce cas, un vecteur de 25 paramètres serait donc suffisant pour décrire une instance du modèle. Dans la suite de l'exposé, on notera avec $\mathbf{a}$ en indice tous les objets paramétrés par ce vecteur : $\mathbb{T}_{\mathbf{a}}$, $P_{\mathbf{a}}$, mais aussi $F_{\mathbf{a}}$, $\Phi_{\mathbf{a}}$, $\phi_{\mathbf{a}}$, *etc.* On aura donc ce type d'égalité :

$$F_{\mathbf{a}}(s) = P_{\mathbf{a}}\Phi_{\mathbf{a}}(s)$$

Cette notation permet d'introduire la notion de famille fractale de courbes (ou surfaces), caractérisée par le vecteur $\mathbf{a}$.

2.3 Conclusion

Nous avons vu qu'il était possible des simplifier le modèle d'IFS projeté sans perdre en généralité, uniquement en utilisant le principe d'équivalence entre modèles. En faisant des hypothèses restrictives, nous avons simplifié encore le modèle, en réduisant le nombre de paramètres. Ces simplifications correspondent à un contrôle plus local de la forme. Pour finir, la paramétrisation d'un modèle IFS projeté peut se faire aisément grâce à un vecteur de paramètres contenant les coordonnées de points de contrôle, ainsi que les coefficients des matrices de subdivision. On définit ainsi des familles de courbes ou de surfaces fractales. Nous allons maintenant voir comment l'approximation de forme est effectuée.

Chapitre 3

Méthode d'approximation

3.1 Formulation du problème

Imaginons que nous disposons d'une *fonction d'écart* entre la fonction fractale $F_{\mathbf{a}}$ et les données d'entrées $\mathbf{Q}$: $\mathcal{E}(F_{\mathbf{a}}, \mathbf{Q})$. Nous pouvons alors considérer que le problème d'approximation se ramène au problème d'optimisation suivant :

$$\mathbf{a}_{opt}(\mathbf{Q}) = \underset{\mathbf{a} \in A}{\operatorname{argmin}}\, \mathcal{E}(F_{\mathbf{a}}, \mathbf{Q})$$

et consiste à trouver le vecteur de paramètres $\mathbf{a}_{opt}$ qui minimise l'écart entre la famille fractale $F_{\mathbf{a}}$ et les données d'entrée $\mathbf{Q}$. L'idéal que nous nous fixons est que cette fonction d'écart ait de bonnes propriétés, c'est à dire qu'elle traduise un écart visuel et qu'elle soit dérivable, rendant ainsi possible l'utilisation de méthodes numériques et non stochastiques. C'est ce formalisme que nous allons utiliser dans la suite.

3.2 Critère d'approximation

La fonction d'écart va être différente pour les courbes et les surfaces. Dans le cas des courbes, il s'agit d'évaluer une distance entre lignes brisées. Dans le cas des surfaces, il faudra se ramener à un modèle projetable, pour pouvoir calculer une distance entre grilles uniformes.

3.2.1 Courbes

La première chose qui vient à l'esprit pour définir une fonction d'écart est l'utilisation d'une distance. Mais l'utilisation d'une distance sous-entend l'existence d'une représentation unifiée des objets à comparer. Dans le cas des courbes, nous distinguons les trois représentations suivantes :

– Courbe paramétrée : fonction continue à valeurs dans $\mathbb{R}^2$

$$t \in [0,1] \;\mapsto\; F(t) \in \mathbb{R}^2$$

– Courbe ensembliste : ensemble compact de points de $\mathbb{R}^2$

$$F([0,1]) = \{F(t) \mid t \in [0,1]\}$$

– Courbe échantillonnée : suite de points de $\mathbb{R}^2$

$$(Q_i)_{i=0,\ldots,m}$$

Suivant la représentation choisie, des distances fonctionnelles, ensemblistes ou des distances entre N-uplets peuvent être mises en œuvre. En géométrie fractale, la distance la plus employée est la distance de Hausdorff, il s'agit d'une distance ensembliste. La fonction qui à un point associe la distance entre ce point et une courbe $F(s)$ est irrégulière et non dérivable en général :

$$\max_i \min_{s\in[0,1]} d(Q_i, F(s))$$

Cette fonction sert de base au calcul de la distance de Hausdorff. Sa minimisation peut se faire avec des algorithmes d'optimisation non dérivables de type génétique ou recuit simulé. Il paraît donc plus simple de se tourner vers une représentation fonctionnelle ou échantillonnée, mais en gardant à l'esprit que l'on veut une fonction d'écart visuelle.

Nous avons vu, dans la section 1.4.2 page 56, qu'il était possible de calculer les valeurs exactes du modèle pour des valeurs de paramètre uniformes (tabulation de la fonction). Cette tabulation constitue un échantillonnage de la courbe. Cette représentation échantillonnée est similaire à la représentation que l'on va utiliser comme donnée d'entrée numérique pour le problème d'approximation. Une première idée consisterait à prendre comme fonction d'écart la somme des distances point à point des représentations échantillonnées de la courbe fractale et de la courbe à approximer. Ceci correspond à une contrainte d'interpolation fonctionnelle qui est trop restrictive puisqu'elle oblige à considérer le même nombre d'échantillons. Nous allons donc plutôt nous tourner vers une distance entre les lignes brisées définies par les échantillons des deux courbes. Ceci permet de se rapprocher d'un critère visuel tout en ayant les bonnes propriétés de dérivabilité. Nous allons donc construire une distance entre deux fonctions elles-mêmes définies par des courbes échantillonnées.

Définition 12 (abscisse curviligne d'une courbe échantillonnée) Soit $(Q_i)_{i=0,\ldots,M}$ une courbe échantillonnée, $Q_i \in \mathbb{R}^2$. L'abscisse curviligne $s_i(\mathbf{Q})$ du point i de cette courbe est définie par :

$$\begin{array}{lll} \text{pour } i = 1,\ldots,M & : & s_i(\mathbf{Q}) = \frac{1}{l(\mathbf{Q})} \sum_{j=0}^{i-1} d(Q_j, Q_{j+1}) \\ \text{pour } i = 0 & : & s_i(\mathbf{Q}) = 0 \end{array}$$

$l(\mathbf{Q})$ étant la longueur totale de la courbe échantillonnée $\mathbf{Q}$:

$$l(\mathbf{Q}) = \sum_{j=0}^{M-1} d(Q_j, Q_{j+1})$$

On peut maintenant définir la fonction de paramétrage affine par morceaux associée à cet échantillon. On définit ainsi une ligne brisée qui est paramétrée par une abscisse curviligne normalisée.

Définition 13 (fonction de paramétrage) Soit $(Q_i)_{i=0,\ldots,M}$ une courbe échantillonnée, $Q_i \in \mathbb{R}^2$. La fonction de paramétrage $\mathcal{G}_Q$ associée à cette courbe échantillonnée est définie par :
$\forall i = 0, \ldots, M-1$ et $\forall t \in [s_i(\mathbf{Q}), s_{i+1}(\mathbf{Q})]$,

$$\begin{array}{rccl} \mathcal{G}_Q : & [0,1] & \rightarrow & \mathbb{R}^2 \\ & t & \mapsto & \mathcal{G}_Q(t) = \left[\frac{t-s_i(\mathbf{Q})}{s_{i+1}(\mathbf{Q})-s_i(\mathbf{Q})}\right] Q_{i+1} + \left[1 - \frac{t-s_i(\mathbf{Q})}{s_{i+1}(\mathbf{Q})-s_i(\mathbf{Q})}\right] Q_i \end{array}$$

Ainsi, on peut comparer deux courbes échantillonnées à l'aide d'une distance fonctionnelle entre lignes brisées.

Définition 14 (distance entre courbes échantillonnées) Soient $(Q_i)_{i=0,\ldots,M}$ et $(Q'_i)_{i=0,\ldots,M'}$ deux courbes échantillonnées de $\mathbb{R}^2$. La distance entre ces deux courbes est définie par :

$$\mathcal{D}(\mathcal{G}_Q, \mathcal{G}_{\mathbf{Q}'}) = \sum_{i=0,\ldots,M''} \left[d(\mathcal{G}_Q(\frac{i}{M''}), \mathcal{G}_{Q'}(\frac{i}{M''})) \right]^2$$

où M'' est un entier non nul donnant la précision de rééchantillonnage et où d représente la distance euclidienne.

Ainsi, la fonction d'écart retenue pour les courbes est la suivante :

$$\begin{array}{rcl} \mathcal{E}(F_{\mathbf{a}}, \mathbf{Q}) & = & \mathcal{D}(\mathcal{G}_{\mathcal{Q}_{\mathbf{a}}}, \mathcal{G}_{\mathbf{Q}}) \\ & = & \sum_{i=0,\ldots,M''} \left[d(\mathcal{G}_{\mathcal{Q}_{\mathbf{a}}}(\frac{i}{M''}), \mathcal{G}_Q(\frac{i}{M''})) \right]^2 \end{array}$$

où $\mathcal{Q}_{\mathbf{a}i} = F_{\mathbf{a}}(\frac{i}{N^p})$. Cette définition peut être facilement généralisée aux courbes définies dans $\mathbb{R}^3$.

Remarque : Par l'intermédiaire de cette définition de la fonction d'écart, nous traduisons une interpolation plus qu'une approximation. Dans la pratique, nous avons observé une bonne stabilité des points en dehors des points d'échantillonnage. Il suffit pour cela de considérer un nombre de points d'échantillonnage qui ne soit pas trop faible.

3.2.2 Surfaces

Pour les surfaces, le problème de l'approximation est beaucoup plus difficile à traiter. Il n'y a en effet plus de notion d'abscisse curviligne. C'est pourquoi nous nous sommes intéressés à des données projetables, c'est à dire définies par des graphes de fonction :

$$(s,t) \mapsto (s,t,F(s,t))$$

Dans la pratique, ce seront des données géographiques ou des images en niveau de gris définies par des grilles d'altitude :

$$\mathbf{Q} = \{Q_{ij} \in \mathbb{R},\ i = 0, \ldots, M,\ j = 0, \ldots, M\}$$

La fonction F est définie par des scalaires de contrôle z_j, $j \in J$ (qui indiquent des hauteurs de surface) :

$$\begin{array}{rccl} F: & [0,1]^2 & \rightarrow & \mathbb{R} \\ & (s,t) & \mapsto & F(s,t) = \sum_{j \in J} \Phi(s,t) z_j \end{array}$$

Grâce à la tabulation de $\Phi(s,t)$, on peut calculer de façon exacte une grille de valeurs de cette fonction scalaire. Ainsi, la distance se calcule comme une distance entre images.

Définition 15 (distance entre grilles d'altitude) Soient $(Q_{ij})_{i=0,...,M,\ j=0,...,M}$ et $(Q'_{ij})_{i=0,...,M,\ j=0,...,M}$ deux grilles d'altitude de $\mathbb{R}^3$. La distance entre ces deux grilles est définie par :

$$d^2(\mathbf{Q}, \mathbf{Q}') = \sum_{i,j=0,...,M} \left(Q_{ij} - Q'_{ij}\right)^2$$

La fonction d'écart est cette distance :

$$\begin{array}{rcl} \mathcal{E}(F_{\mathbf{a}}, \mathbf{Q}) & = & d^2(\mathcal{Q}_{\mathbf{a}}, \mathbf{Q}) \\ & = & \sum_{i,j=0,...,M} \left(\mathcal{Q}_{\mathbf{a}ij} - Q_{ij}\right)^2 \end{array}$$

où $\mathcal{Q}_{\mathbf{a}ij} = F_{\mathbf{a}}(\frac{i}{N^p}, \frac{j}{N^p})$.

Dans le domaine de l'image, on utilise plus souvent la distorsion entre images, représentée par le PSNR (Peak Signal-to-Noise Ratio). En voici la définition :

Définition 16 (PSNR) Soit deux images Q et Q' de taille $x \times y$. Le PSNR (en dB) entre ces deux images est donné par la formule suivante :

$$\mathrm{PSNR}(Q, Q') = 10 \log_{10} \left(\frac{\mathrm{Ech}^2}{\mathrm{MSE}(Q, Q')} \right)$$

Ech étant l'étendue des données (typiquement 255 pour les images en niveau de gris) et MSE (Mean Squared Error) défini par :

$$\begin{array}{rcl} \mathrm{MSE}(Q, Q') & = & \frac{\sum_{i=1...x, j=1...y} \left(Q_{i,j} - Q'_{i,j}\right)^2}{x\,y} \\ & = & \frac{1}{x\,y} d^2(Q, Q') \end{array}$$

3.3 Résolution

C'est ici que se situe toute la difficulté. Il faut choisir une méthode de résolution. Pour cela, plusieurs observations préalables sont à prendre en compte :

– La fonction d'évaluation est coûteuse, surtout pour les surfaces, même avec des optimisations. En effet, elle nécessite le calcul de la fonction fractale, récursive. Il faut donc une méthode de résolution peu gourmande en évaluations.

– Une étude de la fonction distance montre que celle-ci a de bonnes propriétés, elle est continue et dérivable (voir section suivante pour la démonstration). Les figures 26 et 27 illustrent cette propriété sur deux paramètres. Dans ces deux exemples, les graphes obtenus possèdent un seul minimum sur le domaine étudié. Ces tests ont été faits sur une courbe. Nous n'avons pas de résultat théorique sur l'unicité du minimum.
– Bien plus qu'une simple fonction d'écart globale, nous disposons d'une distance fonctionnelle : en plusieurs points, il est possible de connaître l'écart entre le point des données d'entrée et le point issu du calcul de la fonction fractale :

$$d(\mathcal{G}_Q(\frac{i}{M}), \mathcal{G}_{Q'}(\frac{i}{M}))$$

Cela signifie qu'une méthode de résolution qui peut optimiser plusieurs valeurs en même temps pourra tirer partie de cette fonctionnalité.

Nous pouvons donc écarter toutes les méthodes d'optimisation non différentielles, *i.e.* celles qui n'utilisent pas la propriété de différentiabilité de la fonction à optimiser. Parmi ces méthodes, on trouve toutes les méthodes probabilistes, recuit simulé, algorithmes génétiques, *etc.* De plus ces méthodes nécessitent un grand nombre d'évaluations. Les méthodes de type simplexe généralisé seront aussi écartées, celles-ci n'utilisant que de façon indirecte une certaine forme de différentiabilité. Des tests ont été faits avec la méthode du simplexe (Downhill Simplex Method[PFTV93a]), mais la convergence est très lente. Une méthode s'est révélée beaucoup plus appropriée : la méthode de Levenberg-Marquardt[PFTV93b]. Cependant, celle-ci requiert le calcul des dérivées.

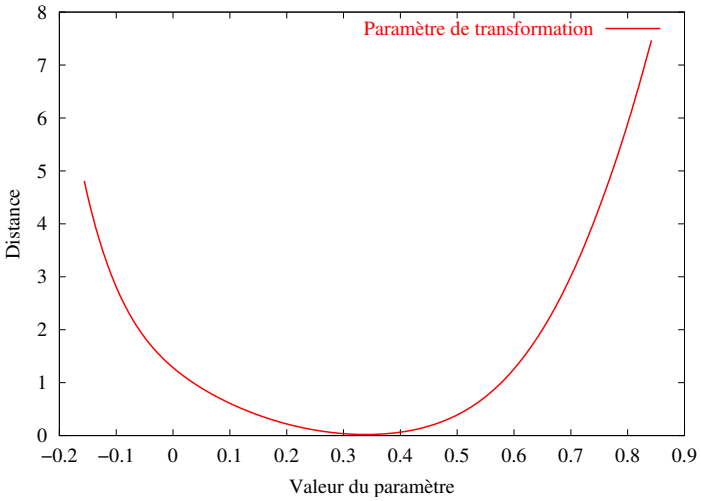


Fig. 26 – Impact de la variation d'un paramètre de transformation sur la distance

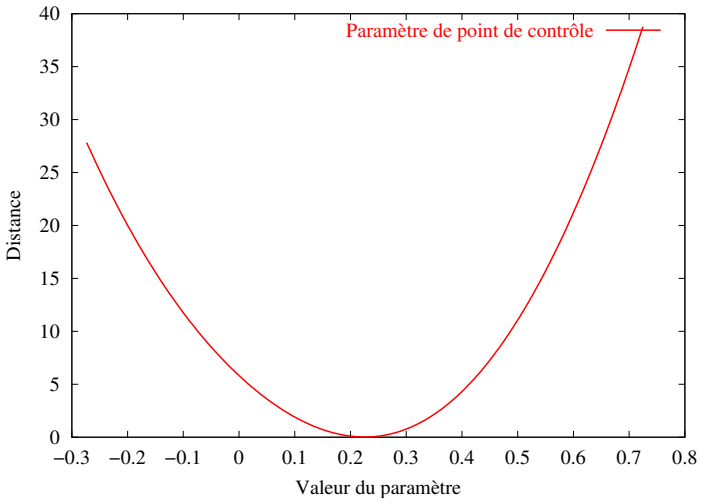


FIG. 27 – Impact de la variation d'un paramètre de point de contrôle sur la distance

3.4 Dérivabilité de la fonction d'écart

Cette section donne des éléments théoriques qui permettent de justifier l'utilisation d'une méthode différentielle pour la résolution de l'approximation. Les fonctions mises en œuvre sont polynomiale en $\mathbf{a}$, donc C^∞.

Proposition 8 (dérivabilité de la fonction tabulation) *La fonction de tabulation de l'IFS projeté* $(P_\mathbf{a}, \mathbb{T}_\mathbf{a})$ *est polynomiale par rapport au vecteur de paramètre* $\mathbf{a}$ *:*

$$\mathbf{a} \quad \mapsto \quad F_\mathbf{a}(\tfrac{i}{N^p})$$

où $i = 0, \ldots, N^p$.

ÉLÉMENTS DE PREUVE. La fonction $\mathbf{a} \mapsto (P_\mathbf{a}, \mathbb{T}_\mathbf{a})$ est linéaire. De plus la fonction $(P_\mathbf{a}, \mathbb{T}_\mathbf{a}) \mapsto F_\mathbf{a}(\frac{i}{N^p}) = P_\mathbf{a} T_{\rho_1} \ldots T_{\rho_p} e_0$ est polynomiale de degré p au plus (linéaire pour $P_\mathbf{a}$). La combinaison de ces deux fonctions est polynomiale de degré p. ■

La fonction de tabulation est polynomiale donc continue et dérivable, mais nous avons besoin de la dérivabilité de la fonction d'écart entre les données d'entrée et le modèle fractal. Pour cela, nous distinguons les deux types de fonctions d'écart : pour les courbes et pour les surfaces.

Proposition 9 (dérivabilité de la fonction d'écart pour les courbes) *Soient* Q *une courbe à approximer et* $F_\mathbf{a}$ *la famille de courbes fractales paramétrée par le vecteur* $\mathbf{a}$*. La fonction d'écart entre ces deux courbes est continue et dérivable par rapport à* $\mathbf{a}$ *:*

$$\mathbf{a} \quad \mapsto \quad \mathcal{E}(F_\mathbf{a}, \mathbf{Q}) = \sum_{i=0,\ldots,M''} \left[d(\mathcal{G}_{\mathcal{Q}_\mathbf{a}}(\tfrac{i}{M''}), \mathcal{G}_Q(\tfrac{i}{M''})) \right]^2$$

Éléments de preuve. La fonction distance entre courbes est la somme de distances euclidiennes entre les points des deux courbes rééchantillonnées uniformément. Le rééchantillonnage implique le combinaison avec une fonction rationnelle et une fonction racine carrée. La seule condition pour que le dénominateur de la fonction rationnelle s'annule est l'égalité de deux points consécutifs $Q_i = Q_{i+1}$. Ceci ne peut arriver que si l'IFS est dégénéré, on veillera à ne pas se trouver dans une telle situation. Le dénominateur de la fonction rationnelle ne pouvant s'annuler, ces fonctions sont continues et dérivables. La combinaison finale de toutes les fonctions est continue et dérivable. ■

Proposition 10 (dérivabilité de la fonction d'écart pour les surfaces) *Soient Q une surface (image) à approximer et $F_{\mathbf{a}}$ la famille de surfaces fractales paramétrée par le vecteur* $\mathbf{a}$*. La fonction d'écart entre ces deux surfaces est dérivable par rapport à* $\mathbf{a}$ *:*

$$\mathbf{a} \;\mapsto\; \mathcal{E}(F_{\mathbf{a}}, \mathbf{Q}) = \sum_{i,j=0,\ldots,M} \left(\mathcal{Q}_{\mathbf{a}ij} - Q_{ij}\right)^2$$

Éléments de preuve. Ici, il n'y a même pas de rééchantillonnage, simplement la somme de différences au carré entre les valeurs de la surface à approximer et celles de la fonction tabulée. On construit ainsi un polynome de degré $2p$. Cette fonction est donc continue et dérivable. ■

> **Remarque :** Il s'agit de résultats forts qui vont nous permettre d'utiliser des méthodes différentielles. Cependant, aucun résultat ne sera donné quant à la généralisation de celui-ci lorsque p tend vers l'infini.

Nous n'avons pas de formule exacte pour le calcul de la dérivée. Cependant, il paraît logique que cette formule soit plus complexe que celle de l'évaluation de la fonction fractale. C'est pourquoi le calcul de la dérivée sera effectué de façon numérique, approchée.

3.5 Méthode de régression

Nous allons ici décrire le fonctionnement de l'algorithme de Levenberg-Marquardt, puis l'adaptation de celui-ci à nos besoins. Il faudra entre autres utiliser un calcul d'approximation des dérivées.

L'algorithme de Levenberg-Marquardt est itératif. Cela signifie qu'il a besoin d'une estimation du résultat, et qu'il va, par itérations successives, tenter d'améliorer le résultat. Cette procédure est répétée tant que l'erreur totale diminue. D'une manière pratique, l'algorithme de Levenberg-Marquardt est une solution numérique au problème de régression suivant :

$$\begin{aligned} \mathbf{a}_{opt} &= \operatorname{argmin}_{\mathbf{a}} \sum_{i=0}^{M} (v_i - f_i(\mathbf{a}))^2 \\ &= \operatorname{argmin}_{\mathbf{a}} \chi^2(\mathbf{a}) \end{aligned}$$

où $\mathbf{v}$ est le vecteur de données d'entrée et f le modèle de régression paramétré par $\mathbf{a}$.

Une simple méthode utilisant le gradient de la fonction χ^2 procèderait de la manière suivante :

$$\mathbf{a}_{suivant} = \mathbf{a}_{courant} - c\,\nabla\chi^2(\mathbf{a}_{courant})$$

où c est une constante et $\nabla\chi^2$ le gradient de la fonction χ^2 :

$$\nabla\chi^2(\mathbf{a}) = \begin{pmatrix} \frac{\partial\chi^2}{\partial\mathbf{a}_1} \\ \vdots \\ \frac{\partial\chi^2}{\partial\mathbf{a}_n} \end{pmatrix}$$

Cette méthode qui est utilisable quand on ne se trouve pas près de la solution (mais tout de même dans le bassin d'attraction), peut être largement améliorée lorsqu'on se trouve proche de celle-ci. En effet, une approximation quadratique de la fonction χ^2 est alors plus plus performante en terme de vitesse de convergence. Cette approximation quadratique requiert le calcul du Hessien de la fonction χ^2.

> **Remarque :** Nous considérons qu'à l'initialisation, nous nous trouvons dans le bassin d'attraction. Sans cette hypothèse, il faudrait implémenter en aval une méthode stochastique. Dans la pratique, nous avons pu vérifier que cette hypothèse était vérifiée.

Une fois exprimé en fonction de f, on néglige les termes de second ordre, de telle sorte que celui-ci ne s'exprime qu'en fonction du gradient ∇f. Dans la méthode Levenberg-Marquardt[PFTV93b], les deux types d'approximation de χ^2 (linéaire et quadratique) sont unifiés en introduisant la somme pondérée de la matrice diagonale gradient et de celle du Hessien. La résolution est faite et si le résultat est moins bon, c'est qu'il faut utiliser une méthode plus orientée gradient, on augmente alors la part de la matrice diagonale gradient dans la résolution. Si le résultat est meilleur, on modifie la valeur des paramètres et on continue. La figure 28 illustre dans un exemple simple (une fonction de $\mathbb{R} \to \mathbb{R}$) les deux différentes méthodes d'approximation de la fonction χ^2.

Pour adapter Levenberg-Marquardt à notre problème d'approximation, nous devons obtenir :

$$\chi^2(\mathbf{a}) = \mathcal{E}(F_\mathbf{a}, \mathbf{Q})$$

car nous voulons minimiser la fonction d'écart. Nous devons alors distinguer les deux cas des courbes et des surfaces.

Dans le cas des courbes, le vecteur $\mathbf{v}$ cible que l'on veut obtenir est un vecteur nul, sachant que le modèle de régression va directement traduire la distance entre les points d'entrée et la fonction fractale :

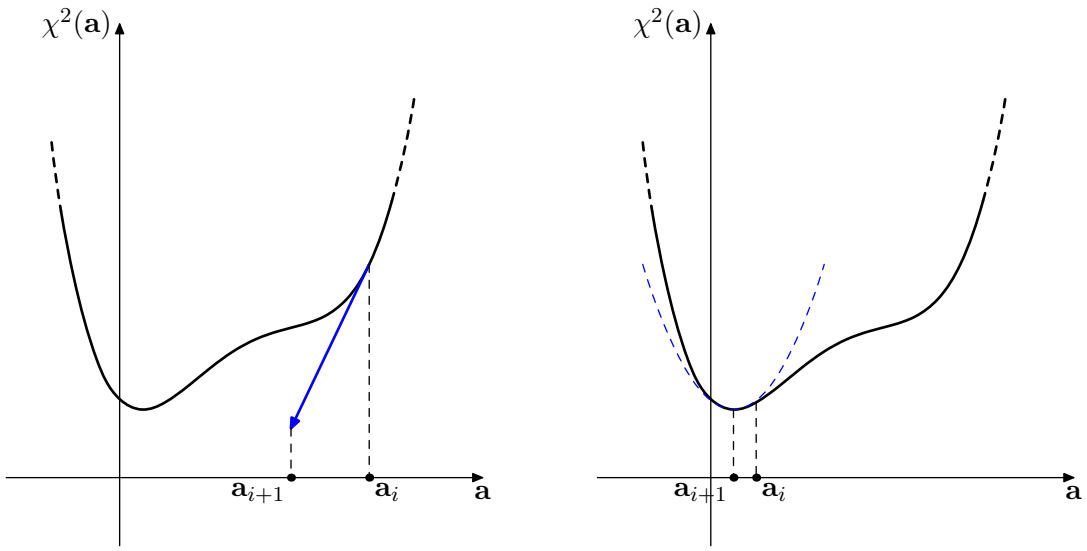
$$f_i(\mathbf{a}) = d((\mathcal{Q}_\mathbf{a})_i, Q_i)$$

On a ainsi :

$$\begin{array}{rcl} \chi^2(\mathbf{a}) & = & \sum_{i=0}^{M}(0 - d((\mathcal{Q}_\mathbf{a})_i, Q_i))^2 \\ & = & \mathcal{E}(F_\mathbf{a}, \mathbf{Q}) \end{array}$$

Dans le cas des surfaces, on utilisera des données d'entrées à deux dimensions v_{ij} et $f_{ij}(\mathbf{a})$. Pour faire correspondre $\chi^2(\mathbf{a})$ et $\mathcal{E}(F_\mathbf{a}, \mathbf{Q})$, il faut choisir :

$$\left\{ \begin{array}{rcl} v_{ij} & = & Q_{ij} \\ f_{ij}(\mathbf{a}) & = & \mathcal{Q}_{\mathbf{a}ij} \end{array} \right.$$



(a) Approximation localement linéaire grâce au gradient

(b) Approximation quadratique grâce au Hessien

FIG. 28 – Les deux types d'approximation de la fonction utilisés dans LEVENBERG-MARQUARDT

La méthode LEVENBERG-MARQUARDT est sensible au point initial. Il faut donc veiller à ce que la première valeur de paramètre communiquée $\mathbf{a}_{init}$ soit la plus proche de la solution possible. En pratique, on veillera à prendre des points de contrôle sur la courbe ou la surface et un modèle lisse interpolant. Des expérimentations ont permis de mettre en évidence la sensibilité de l'algorithme aux paramètres initiaux. Heureusement, cette sensibilité est assez réduite, on observe des différences assez faibles dans l'amélioration des résultats.

Le calcul des dérivées se fait grâce à une approximation. On reprendra tout simplement la formule de la dérivée en considérant une valeur de ϵ faible :

$$\frac{\partial f}{\partial \mathbf{a}_i}(\mathbf{a}) \approx \frac{f_{\mathbf{a}+\delta\mathbf{a}}(s) - f_{\mathbf{a}}(s)}{\epsilon}$$

On choisit empiriquement la valeur de ϵ pour qu'elle ne soit pas trop petite et provoque des imprécisions numériques, ni trop grande auquel cas l'approximation de la dérivée serait erronée.

3.6 Conclusion

Avec des critères d'approximation adaptés, on peut utiliser une méthode de résolution qui utilise les propriétés de dérivabilité de la famille fractale que nous avons construite. La méthode de résolution que nous avons choisie est celle de LEVENBERG-MARQUARDT. Elle a été adaptée à nos besoins, grâce notamment à un calcul approché de la dérivée. Nous

allons maintenant exposer les résultats obtenus pour l'approximation de courbes puis de surfaces.

Chapitre 4

Résultats

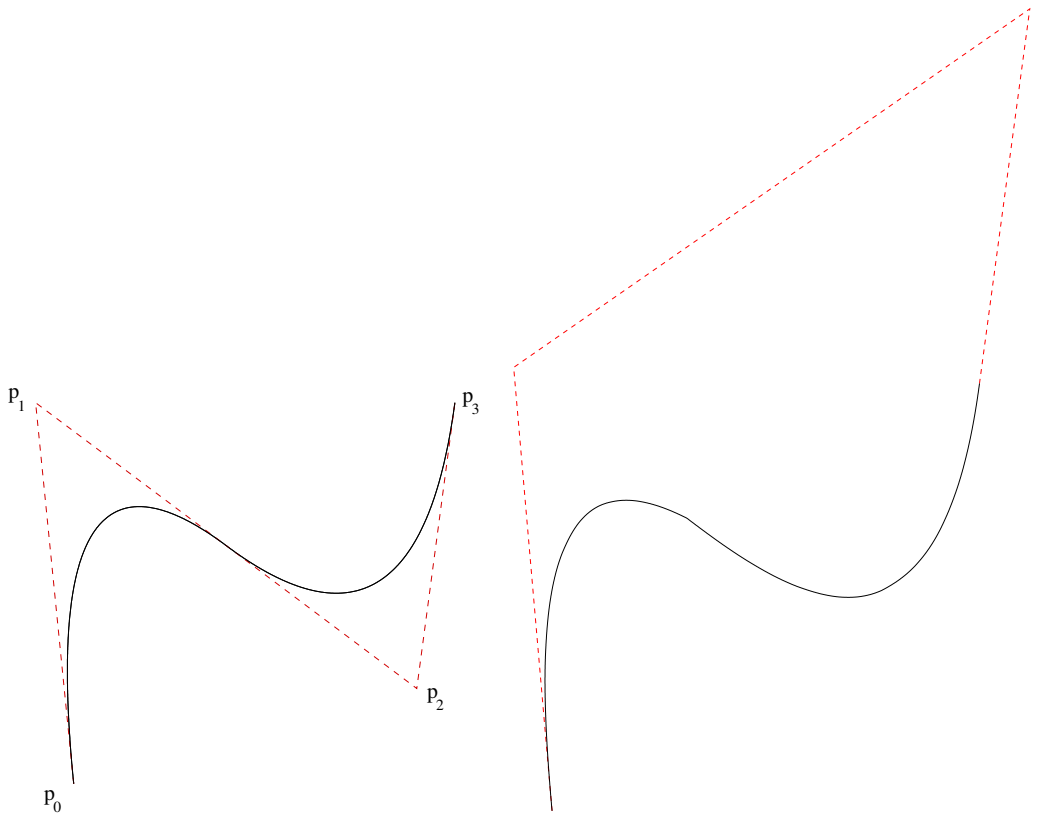
Ce chapitre regroupe les résultats obtenus pour l'approximation de courbes, puis de surfaces et d'images, grâce à la méthode que nous avons proposée. Nous nous attacherons à donner des résultats sous forme numérique (temps de calcul, distance finale entre l'objet et son approximation) ainsi que sous forme visuelle.

4.1 Courbes

Nous allons ici exposer les résultats concernant l'approximation de courbes. Dans un premier temps, des tests de validation ont été effectués. Ces tests mettent en œuvre des données d'entrées synthétiques. Dans un second temps, nous voyons comment se comporte la méthode face à des données réelles, telles que des contours d'objets dans des images ou des lignes de niveaux de données géographiques. Enfin, nous analysons l'impact que peut avoir la quantification des paramètres du modèle sur la distorsion.

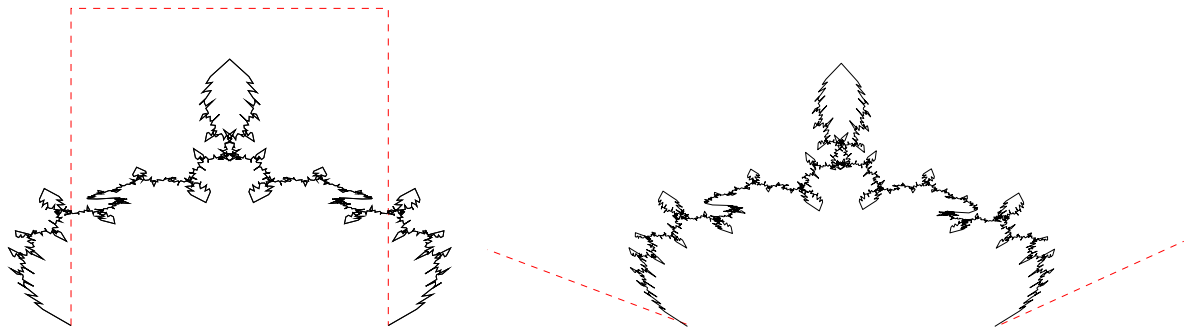
4.1.1 Données synthétiques

Dans les résultats qui suivent, nous avons utilisé des modèles avec 4 points de contrôle et 2 transformations. Les figures 29, 30 et 31 illustrent les résultats obtenus. La reconstruction d'une courbe lisse (spline) est parfaitement obtenue (figure 29). Il en est de même pour des courbes plus complexes comme illustré dans la figure 30 page suivante avec une courbe spiralée et dans la figure 31. Dans une optique de problème inverse, on pourrait croire que ces résultats ne sont pas satisfaisants. En effet, on ne trouve pas les mêmes polygones de contrôle que ceux initiaux. On ne retrouve donc pas non plus les mêmes transformations. Ceci est dû au fait que le résultat est à un changement de base près. Notre méthode ne donne donc pas un résultat *canonique*. Un modèle canonique est en effet difficile à construire, car il nécessite la définition de critères pour le positionnement des points de contrôle. Un simple changement de base permettra d'obtenir *exactement* les transformations initiales. Les résultats numériques sont donnés dans le tableau 1 page 83.



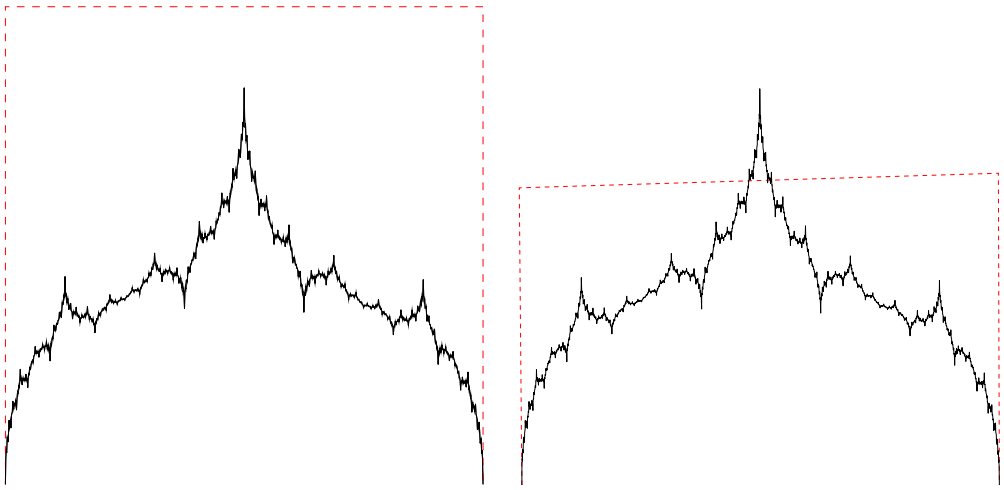
(a) Courbe originale (b) Courbe approximée

FIG. 29 – Approximation d'une spline



(a) Courbe originale (b) Courbe approximée

FIG. 30 – Approximation d'une courbe spiralée



(a) Courbe originale (b) Courbe approximée

FIG. 31 – Approximation d'une autre courbe synthétique

Figure	29	30	31
Nombre de points donnés	1025	1025	4097
Nombre de points utilisés pour le rééchantillonnage	500		
Profondeur	9		
Nombre d'itérations	136	156	552
Temps de calcul (Athlon $1,8Ghz$)	$29''$	$36''$	$123''$
Écart final $\mathcal{E}(F_{\mathbf{a}_{opt}}, \mathbf{Q})$	$0,00119$	$0,754$	$0,0203$

TAB. 1 – Résultats numériques – problème inverse

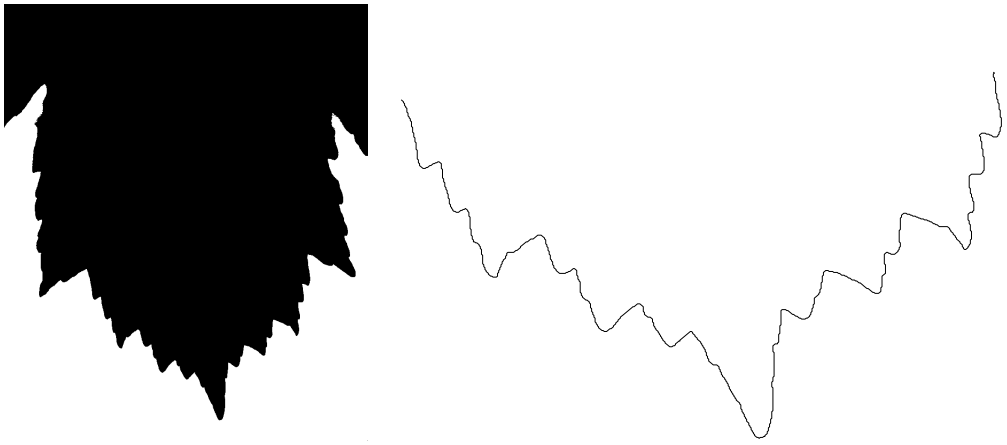
4.1.2 Exemple complet

Dans cette section, intéressons-nous à un exemple complet d'approximation, en analysant les divers points :

- Description des données d'entrée
- Description du modèle utilisé
- Initialisation et paramètres de l'algorithme
- Résultats numériques, convergence

Données d'entrée

Il s'agit d'une feuille de platane scannée à l'aide d'un scanner standard. L'image a été ensuite convertie en binaire (figure 32a). Un programme d'extraction de contour a été utilisé pour en déduire une partie de la courbe du contour de cette feuille (figure 32b). Cette courbe comporte 1250 points.



(a) Image binaire extraite de la feuille scannée

(b) Courbe extraite

FIG. 32 – Données d'entrée

Modèle utilisé

Le modèle utilisé pour l'approximation possède 6 points de contrôle et 3 matrices de subdivision. Ces matrices sont générales, cela signifie qu'aucun zéro n'a été introduit pour simplifier. Le nombre total de paramètres est donc 82 :

- $12 = 6 \times 2$ pour les 6 points de contrôle
- $70 = 25 + 25 + 20$ pour les matrices de subdivision

La figure 33 montre chacune des matrices de subdivision avec les paramètres qu'elle contient.

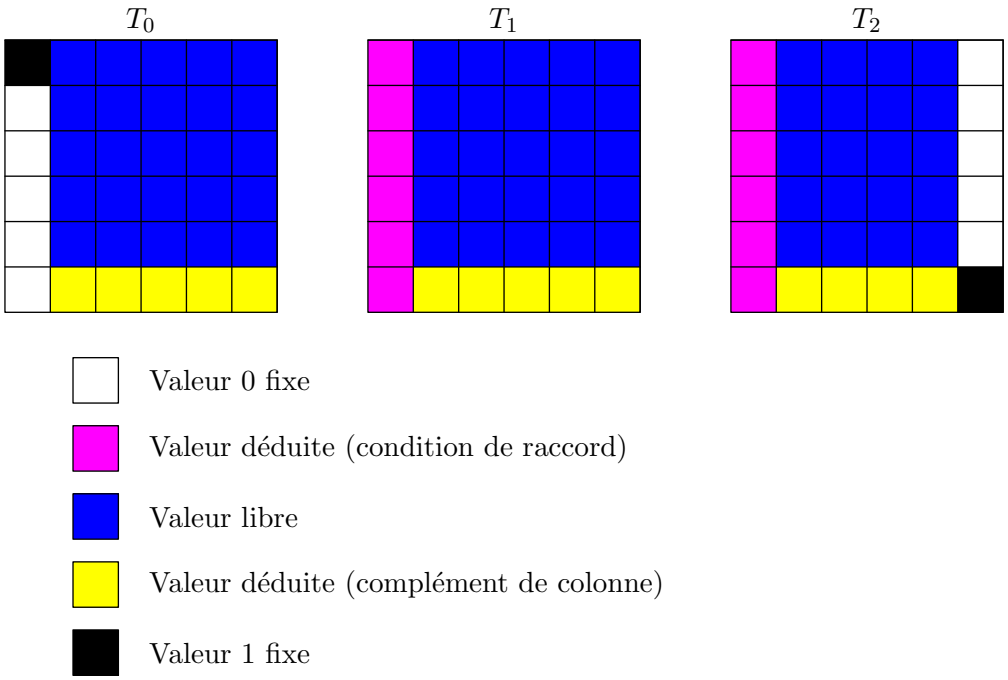


FIG. 33 – Contenu des matrices de subdivision

Initialisation et paramètres de l'algorithme

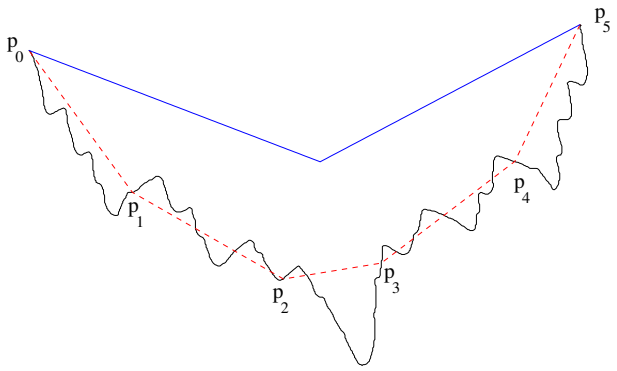


FIG. 34 – Initialisation du modèle

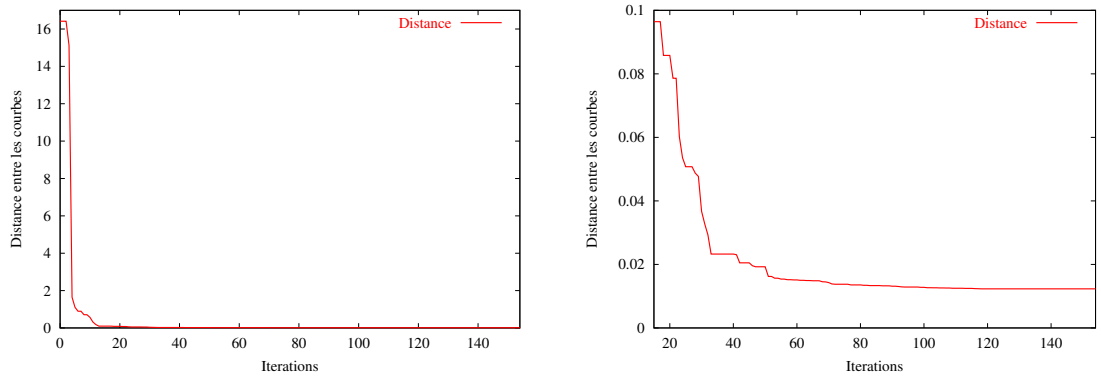
Le modèle initial est donné dans la figure 34, il s'agit d'une ligne brisée représentée en bleu sur le schéma. Les 6 points de contrôle sont choisis uniformément sur la courbe à approximer (en rouge pointillé sur le schéma). Pour l'algorithme, les paramètres suivants ont été utilisés :

- Une profondeur de 5 pour la synthèse de la courbe fractale, ce qui génère une courbe de 1216 points.
- Un rééchantillonnage uniforme sur 1000 points. C'est celui qui donne le meilleur résultat.

– Un maximum de 300 itérations dans la méthode de Levenberg-Marquardt, celui-ci n'a pas été atteint dans notre exemple, car la convergence a eu lieu en 154 itérations.

Résultats numériques, convergence

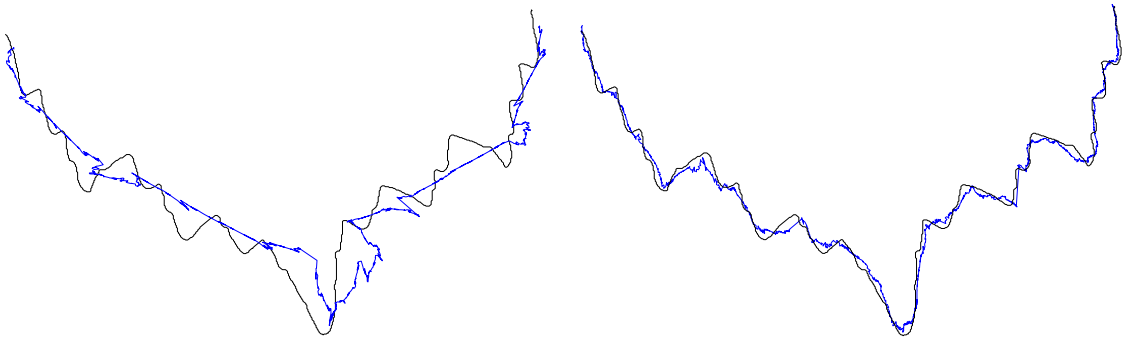
La figure 35 montre la valeur de $\mathcal{E}(F_{\mathbf{a}}, \mathbf{Q})$ en fonction du nombre d'itérations. Les figures 36 et 37 page ci-contre montrent des exemples de résultats intermédiaires obtenus à diverses itérations. À l'itération 70, on voit que le résultat final est presque obtenu. En effet, on a $\mathcal{E}_{70}(F_{\mathbf{a}}, \mathbf{Q}) = 0,0143$ et $\mathcal{E}_{154}(F_{\mathbf{a}}, \mathbf{Q}) = 0,0123$, ces valeurs se traduisent aussi visuellement (voir figure 37 page suivante). La figure 38 page ci-contre montre le résultat final ainsi que le polygone de contrôle.



(a) Depuis l'itération 0

(b) Depuis l'itération 15

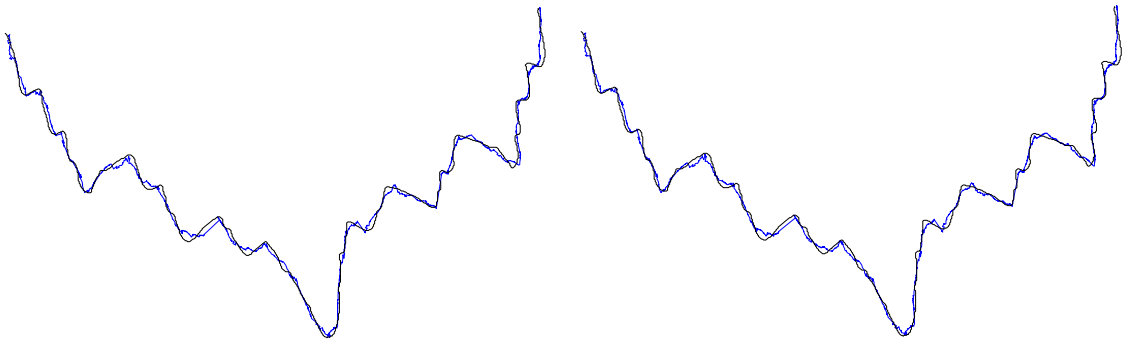
Fig. 35 – Convergence numérique



(a) Itération n°5, $\mathcal{E}_5(F_{\mathbf{a}}, \mathbf{Q}) = 1,1066$

(b) Itération n°25, $\mathcal{E}_{25}(F_{\mathbf{a}}, \mathbf{Q}) = 0,0508$

Fig. 36 – Résultats intermédiaires



(a) Itération n°70, $\mathcal{E}_{70}(F_{\mathbf{a}}, \mathbf{Q}) = 0,0143$ (b) Itération n°154, $\mathcal{E}_{154}(F_{\mathbf{a}}, \mathbf{Q}) = 0,0123$

FIG. 37 – Résultats intermédiaires

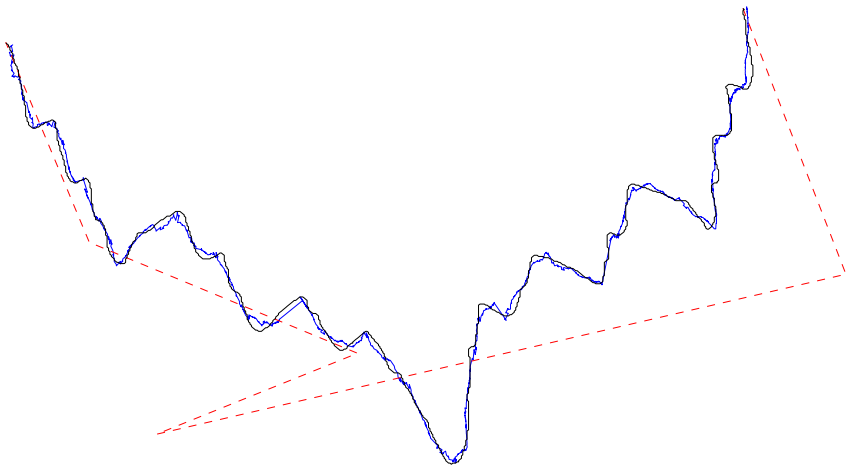
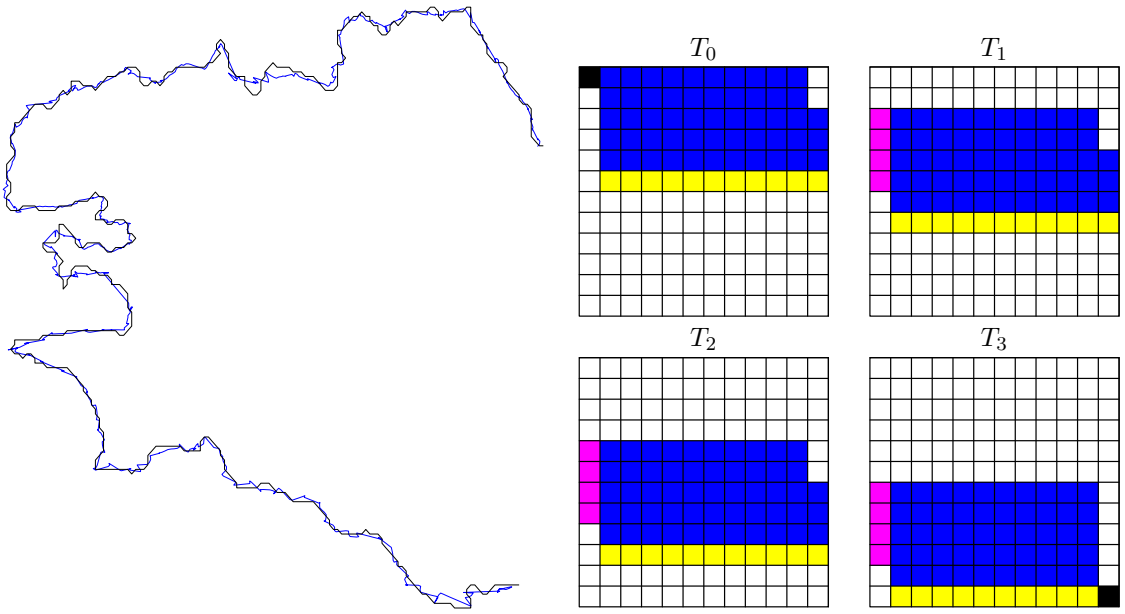


FIG. 38 – Résultat final de l'approximation

4.1.3 Autres exemples de contours

D'autres approximations de courbes réelles ont été effectuées. La figure 39 montre l'approximation d'une partie de la péninsule du Cotentin. La courbe a été construite grâce à des données géographiques DTED (Digital Terrain Elevation Data) précises à 30 secondes d'arc recueillies sur site de GeoCommunity[Fra]. Le résultat met en œuvre un modèle très complexe puisqu'il possède 233 paramètres répartis dans 12 points de contrôle et 4 transformations. La figure 39b indique la façon dont sont simplifiées les matrices pour réduire le nombre de paramètres[2].

La figure 41 page ci-contre montre l'approximation du contour d'une montagne. Ce contour a été extrait manuellement depuis une l'image de la figure 40 page suivante. La figure 42 page 90 illustre l'approximation d'une ligne de niveau issue de données géographiques de l'Alaska[Ala].



(a) Courbe originale et approximation

(b) Structure des 4 transformations utilisées

FIG. 39 – Approximation de la péninsule du Cotentin

4.1.4 Quantification

Nous étudions ici l'effet de la quantification des paramètres du modèle sur l'écart final entre les courbes. Les résultats portent sur l'approximation de la feuille de platane développée dans la section 4.1.2 page 84. Tous les paramètres ont subi une même quantification uniforme sur un nombre de bits prédéfini. Les résultats sont donnés dans le graphe de la

[2]Pour la légende, se reporter à la figure 33 page 85

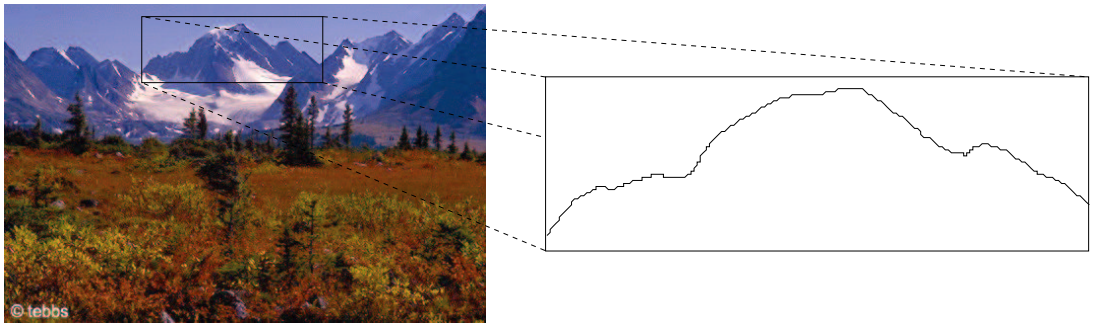
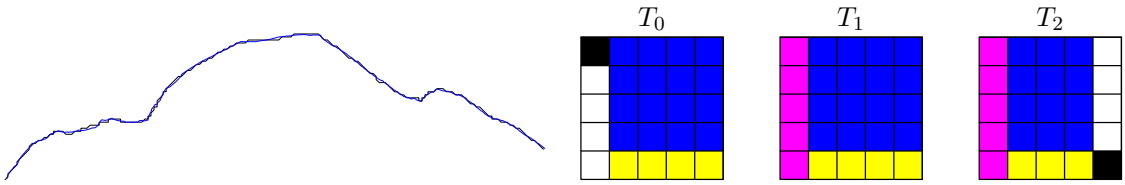


FIG. 40 – Extraction du contour d'une montagne



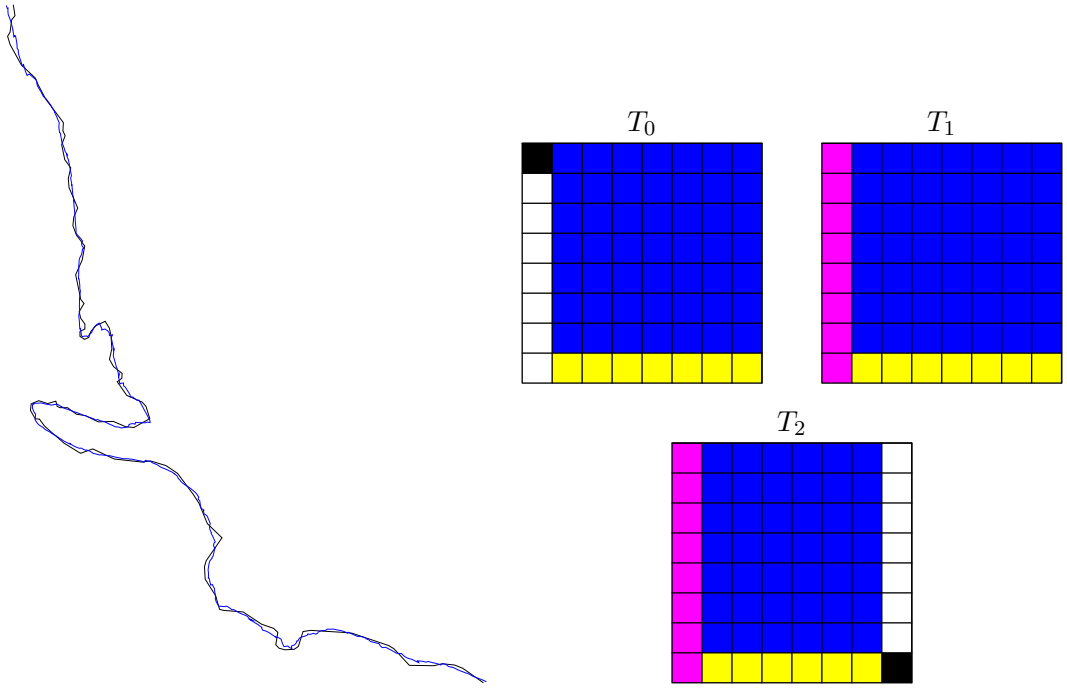
(a) Courbe originale et approximation

(b) Structure des 3 transformations utilisées

FIG. 41 – Approximation du contour d'une montagne

Figure	39	41	42
Nombre de points donnés	500	210	109
Nombre de points utilisés pour le rééchantillonnage	1000	500	
Profondeur	3	5	4
Nombre d'itérations	300	92	176
Temps de calcul	9′58″	18″	2′16″
Distorsion χ^2	$0,0322$	$9,70.10^{-4}$	$7,10.10^{-3}$

TAB. 2 – Résultats numériques – approximation de courbes



(a) Courbe originale et approximation

(b) Structure des 3 transformations utilisées

FIG. 42 – Approximation d'une ligne de niveau

figure 43. Les performances commencent à se dégrader lorsque l'on passe la barre des 8

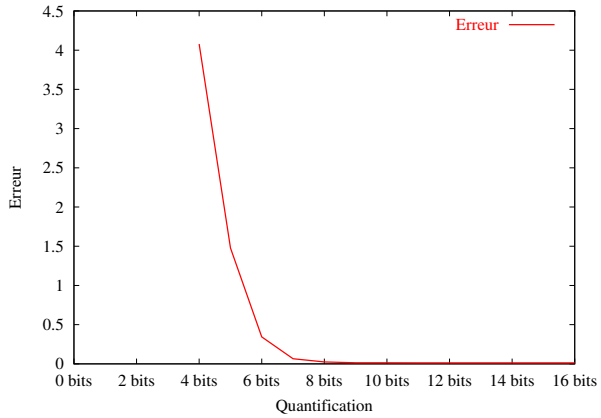
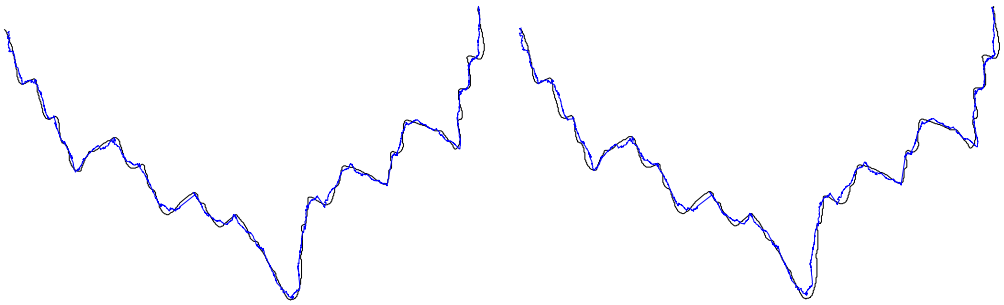


FIG. 43 – Effet de la quantification des paramètres

bits. Les figures 44 et 45 page suivante montrent visuellement l'effet de la quantification des paramètres. À 9 ou 8 bits par paramètre, aucune différence n'est perceptible par rapport à l'approximation de la figure 38 page 87. À 7 bits par paramètre, la déformation est faible mais visible, à 6 bits par paramètre, elle est très visible.

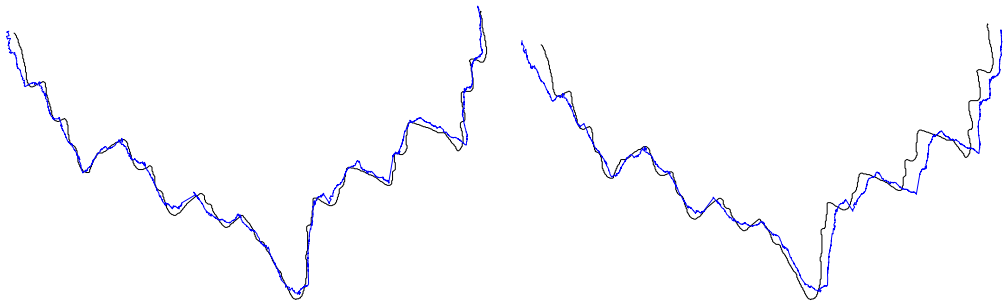


(a) Quantification à 9 bits par paramètre

(b) Quantification à 8 bits par paramètre

FIG. 44 – Quantification non visible (ou très peu)

Remarque : Un codage brut de ces paramètres sur 8 bits permettrait de décrire la courbe à l'aide de 8×82 soit 656 bits. La courbe initiale était codée sur $1250 \times 2 \times 8$ soit 20000 bits. Le facteur de compression est donc de $30,5$, et avec un codage brut des paramètres (quantification uniforme et codage avec code de longueur fixe).



(a) Quantification à 7 bits par paramètre

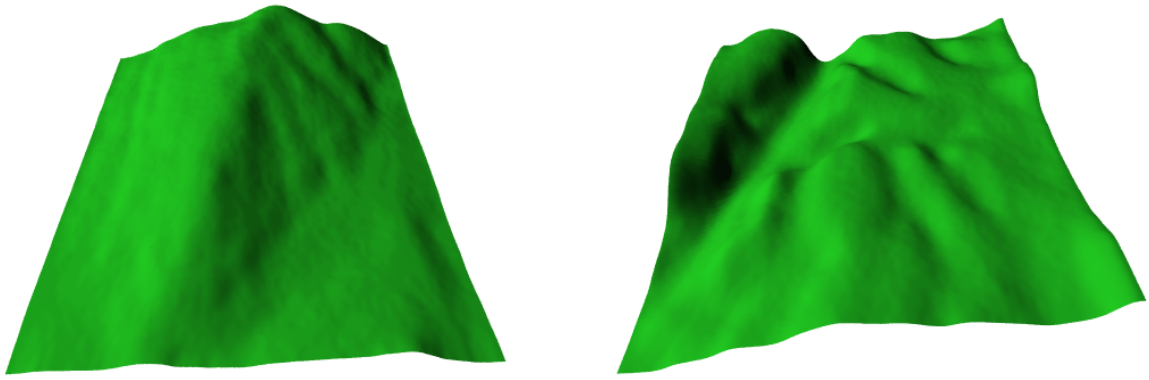
(b) Quantification à 6 bits par paramètre

FIG. 45 – Effet de la quantification des paramètres sur la précision de l'approximation

4.2 Surfaces

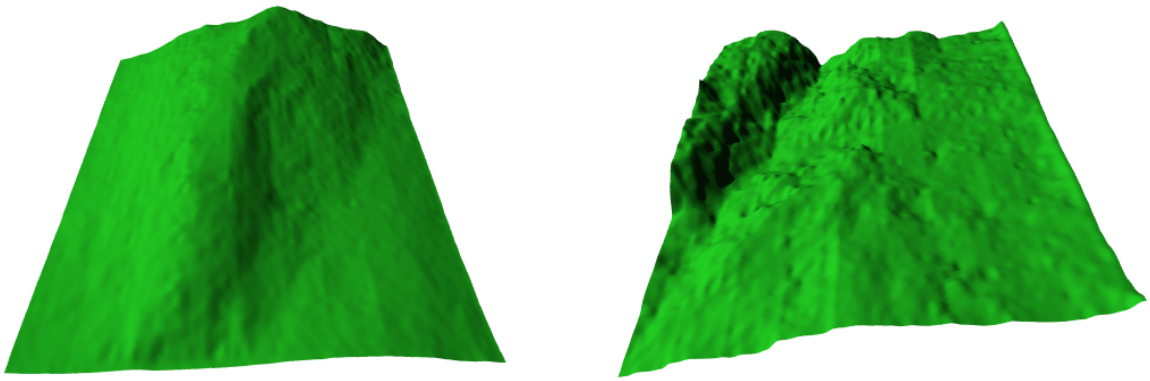
Nous exposons ici des résultats concernant l'approximation de surfaces réelles par un seul carreau fractal.

Les figures 46, 47 et 48 montrent l'approximation de deux surfaces dont les données sont issues des DEM (Digital Elevation Model) de l'USGS (United States Geological Survey) [DEM]. Ces données correspondent à des relevés de terrain. La figure 46 représente les deux surfaces originales. Les figures 47 et 48 représentent les surfaces approximées. Ces surfaces sont visualisées à l'aide du logiciel de lancer de rayons PovRay[Pov] et une interpolation bicubique des points lorsque la surface en possède peu. Dans ce cas précis, les surfaces originales contiennent 82×82 points. Ces données ont été interpolées pour ne pas faire apparaître les facettes. Le nombre d'itérations nécessaires pour la convergence est plus petit que pour les courbes. Ceci est probablement dû à la nature non adaptative de la fonction distance utilisée. En pratique, 15 itérations sont suffisantes pour garantir un résultat qui ne sera que très peu amélioré dans la suite du processus d'approximation. Les modèles qui ont été utilisés ici comportent 3×3 transformations et 4×4 points de contrôle dans le cas de la figure 47. Dans la figure 48, ce sont aussi 3×3 transformations qui ont été utilisées, mais avec une grille de contrôle de 10×10 points. Le tableau 3 page 94 résume les données numériques relatives à l'approximation de ces deux surfaces.



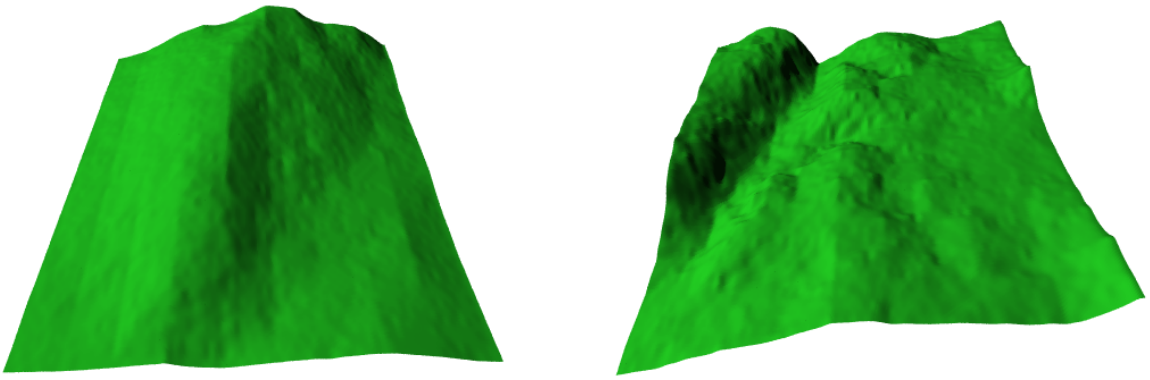
(a) Surface originale 1 (b) Surface originale 2

FIG. 46 – Surfaces originales



(a) Surface approximée 1 (b) Surface approximée 2

FIG. 47 – Approximation grâce à un modèle de 232 paramètres



(a) Surface approximée 1 (b) Surface approximée 2

FIG. 48 – Approximation grâce à un modèle de 316 paramètres

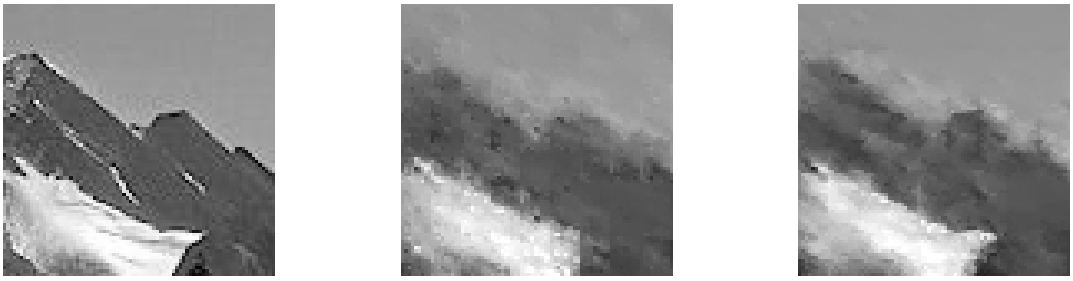
Figure	47a	47b	48a	48b
Taille des données d'entrée	82×82			
Taille de la grille de points de contrôle	4×4		10×10	
Nombre de transformations	3×3			
Nombre de paramètres	232		316	
Écart final $\mathcal{E}(F_{\mathbf{a}}, \mathbf{Q})$	$0,397$	$2,79$	$0,218$	$1,01$
PSNR	$42,3$ dB	$33,8$ dB	$44,9$ dB	$38,2$ dB
Nombre d'itérations	limité à 15			
Temps de calcul	$2'30''$	$2'30''$	$4'43''$	$4'43''$

TAB. 3 – Résultats numériques – approximation de surfaces

4.3 Images

Les résultats concernant les images utilisent exactement le même algorithme que dans la section précédente concernant les surfaces. La seule différence réside dans la nature des données, chaque point représente maintenant un niveau de gris et non une altitude. L'approximation par un seul carreau fractal ne permet pas de traiter de grosses images. Nous nous sommes donc limités à des imagettes de taille 129×129 maximum. Les figures 49, 50 et 51 montrent les résultats visuels de l'approximation de trois images. La première image est une partie d'une image plus grande déjà montrée dans la figure 40 page 89. La deuxième représente l'image astronomique d'une naine blanche (figure 50). La dernière est une texture de matériau de construction (figure 51). À chaque fois, l'image a été approximée grâce à deux modèles différents. Les détails de ces modèles ainsi que les résultats numériques sont donnés dans le tableau 4 page suivante.

L'approximation, tant d'un point de vue visuel que numérique n'est pas acceptable, à part peut-être pour l'image de la naine blanche pour laquelle l'erreur devient raisonnable (au dessus de 25 dB).

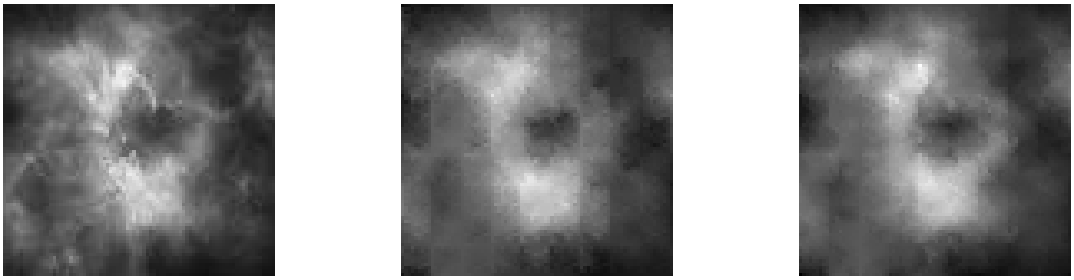


(a) Image originale

(b) Image approximée (232 paramètres)

(c) Image approximée (316 paramètres)

FIG. 49 – Approximation de la partie d'une image de montagne

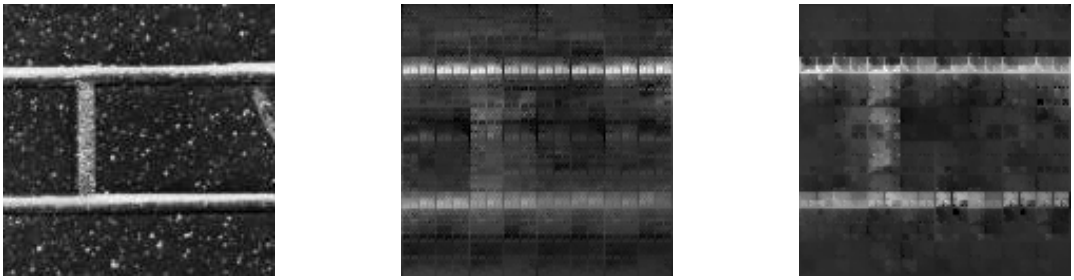


(a) Image originale

(b) Image approximée (232 paramètres)

(c) Image approximée (316 paramètres)

FIG. 50 – Approximation d'une image de naine blanche



(a) Image originale

(b) Image approximée (137 paramètres)

(c) Image approximée (193 paramètres)

FIG. 51 – Approximation d'une texture

Image originale	49a		50a		51a	
Image approximée	49a	49b	50a	50b	51a	51b
Taille des données d'entrée	82×82				129×129	
Taille de la grille de points de contrôle	4×4	10×10	4×4	10×10	5×5	9×9
Nombre de transformations	3×3				2×2	
Nombre de paramètres	232	316	232	316	137	193
Écart final $\mathcal{E}(F_{\mathbf{a}}, \mathbf{Q})$	$58,7$	$44,9$	$14,6$	$10,3$	334	280
PSNR	$20,6$ dB	$21,7$ dB	$26,6$ dB	$28,1$ dB	$17,0$ dB	$17,7$ dB
Nombre d'itérations	limité à 15					
Temps de calcul	$2'30''$	$4'46''$	$2'30''$	$4'44''$	$2'17''$	$4'17''$

TAB. 4 – Résultats numériques – approximation d'images

4.4 Conclusion

L'approximation de courbes grâce à un seul modèle fractal semble être bonne avec la méthode que nous utilisons.

Il en est autrement pour ce qui est de l'approximation de surfaces et d'images. En effet, la qualité d'approximation n'est pas assez satisfaisante pour que cette méthode soit utilisée. En théorie, il devrait être possible d'obtenir de meilleurs résultats en utilisant des modèles avec plus de transformations et plus de points de contrôle. Malheureusement, les coûts en calculs induits par de tels modèles sont prohibitifs et nous nous sommes limités à des modèles de 316 paramètres.

Il est par ailleurs réaliste de penser qu'un seul modèle ne peut pas permettre l'approximation d'un terrain ou d'une image de grande taille, tant les caractéristiques (structurelles ou statistiques) de ces dernières varient spatialement. Seul un ensemble de modèles localisés peut conduire à une approximation acceptable.

Partant de ce constat, nous avons imaginé un modèle composé de plusieurs IFS projetés combinés entre eux dans une structure adaptée : le quadtree. La partie qui suit expose ce modèle et propose deux algorithmes d'approximation associés.

Troisième partie

Surfaces composées

Chapitre 1

Modèle

Nous avons vu dans la partie précédente une méthode d'approximation de surface par un modèle d'IFS projeté. Lorsque les données d'entrée représentant une surface forment un gros volume, ou que la complexité de celles-ci rendent cette approximation impossible, il semble intuitif de faire la combinaison de plusieurs modèles pour parvenir à une précision acceptable. Dans cette partie, nous allons aborder ce problème sous plusieurs angles. Tout d'abord, d'un point de vue théorique, nous verrons comment il est possible de formaliser la combinaison de modèles surfaciques sous la forme d'un quadtree, c'est l'objet de ce chapitre. Cette modélisation peut être rapprochée de la modélisation par B-splines hiérarchiques proposée par Forsey et Bartels[FB88]. Ensuite, nous proposerons deux méthodes pour approximer des surfaces et des images en niveau de gris. La première méthode est fondée sur un critère de minimisation de la distorsion globale, sans soucis d'optimisation de la complexité du modèle. La seconde, à partir d'un critère de débit, va tenter la minimisation de la distorsion globale grâce à une méthode de type multiplicateur de Lagrange.

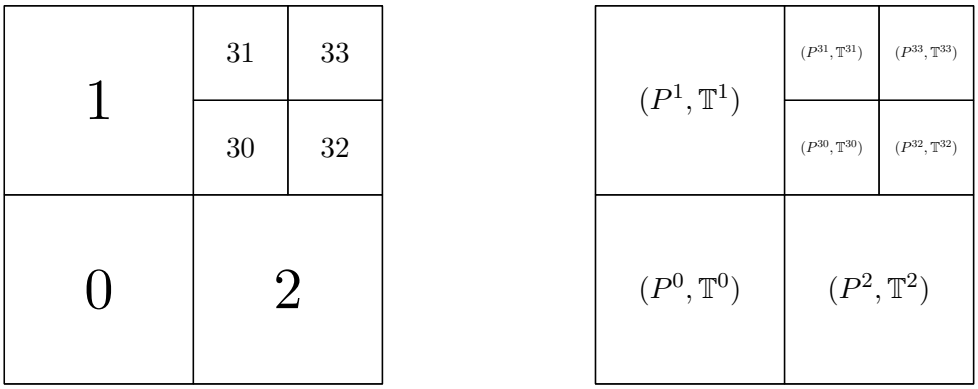
1.1 Formalisme

Dans un premier temps, nous allons définir les notations pour les quadtree. Un quadtree est un arbre quaternaire dont les nœuds sont identifiés par les mots de $\{0,1,2,3\}^*$. La coupure d'un quadtree est un sous-ensemble de $\{0,1,2,3\}^*$ correspondant à une subdivision d'un carré (voir figure 52).

Notation 1 (coupure et feuille d'un quadtree) Nous notons Γ la coupure d'un quadtree et $\gamma \in \Gamma$ une feuille de cette coupure. L'adressage de subdivision est de type Péano comme dans la figure 8 page 46.

Exemple : La figure 52 page suivante représente un exemple simple de coupure de quadtree. Dans ce cas précis, on aura :

$$\Gamma = \{0,1,2,30,31,32,33\}$$



(a) Coupure de quadtree (b) Attracteurs projetés associés

FIG. 52 – Exemple de coupure d'un quadtree avec les attracteurs projetés associés

À chaque feuille, nous pouvons maintenant associer un modèle complet d'attracteur d'IFS projeté.

Définition 17 (quadtree d'attracteurs projetés) Soit Γ une coupure de quadtree. Un quadtree d'attracteurs projetés est défini par l'ensemble des couples $(P^\gamma, \mathbb{T}^\gamma)$, avec $\gamma \in \Gamma$.

Grâce à ce formalisme, il est possible de décrire une surface paramétrée. Pour cela, nous introduisons les subdivisions élémentaires de l'intervalle $I = [0, 1]$:

$$\mathrm{T}_0^I(s) = \frac{1}{2}s \text{ et } \mathrm{T}_1^I(s) = \frac{1}{2}(s+1), s \in I$$

Il en découle les subdivisions du pavé $[0, 1]^2$:

$$\begin{array}{lll} \mathrm{T}_0(s,t) = (\mathrm{T}_0^I(s), \mathrm{T}_0^I(t)) & ; & \mathrm{T}_1(s,t) = (\mathrm{T}_0^I(s), \mathrm{T}_1^I(t)) \\ \mathrm{T}_2(s,t) = (\mathrm{T}_1^I(s), \mathrm{T}_0^I(t)) & ; & \mathrm{T}_3(s,t) = (\mathrm{T}_1^I(s), \mathrm{T}_1^I(t)) \end{array}$$

La construction de la surface est déduite de ces subdivisions :

$$F(s,t) = \sum_{\gamma \in \Gamma} \chi_\gamma(s,t)\ P^\gamma\ \Phi^\gamma(\mathrm{T}_\gamma^{-1}(s,t))$$

où :

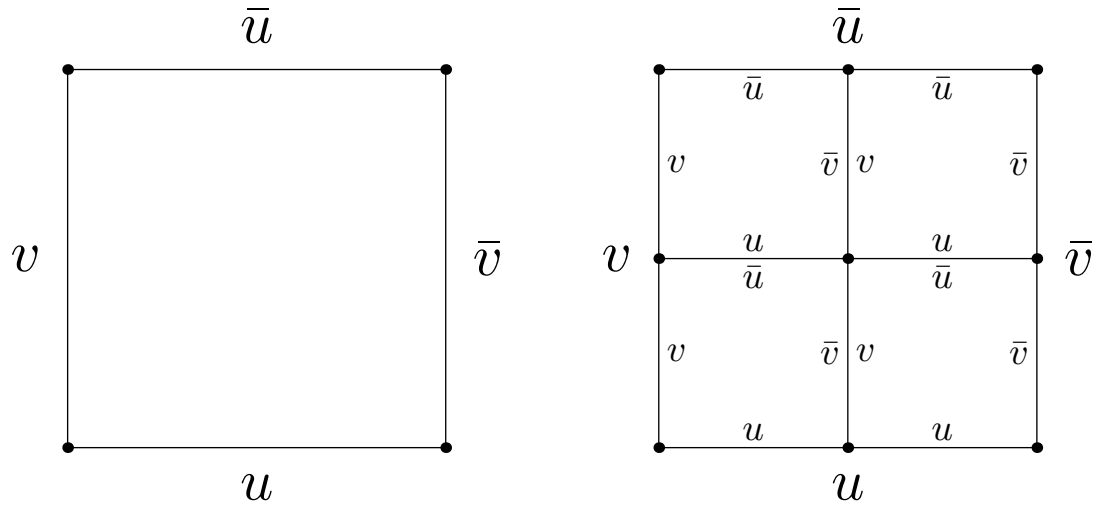
$$\chi_\gamma(s,t) = \begin{cases} 1 & \text{si } (s,t) \in \mathrm{T}_\gamma[0,1]^2 \\ 0 & \text{sinon} \end{cases}$$

et $\mathrm{T}_\gamma^I = \mathrm{T}_{\gamma_1}^I \circ \ldots \circ \mathrm{T}_{\gamma_k}^I$.

Cette définition n'est pas suffisante pour obtenir une fonction continue. L'objet de la section suivante est de décrire les contraintes de raccord nécessaires à cela.

1.2 Condition de raccord

Pour obtenir une fonction C^0, il est nécessaire d'introduire des contraintes de raccord. L'ensemble de ces contraintes sera appelé *condition de raccord*. Nous avons déjà vu dans le chapitre 2.1.1 page 59 qu'il existe une condition pour qu'un attracteur projeté soit décrit par une fonction C^0. Les contraintes globales de raccord de la surface définie par un quadtree d'attracteurs projetés devront donc inclure ces contraintes que nous appellerons *constraintes internes*. Mais elles devront inclure aussi toutes les contraintes induites par les raccords entre les carreaux associés aux feuilles du quadtree. Ces nouvelles contraintes seront dites *externes*. Nous allons ici voir comment il est possible de regrouper l'ensemble des contraintes internes et externes sous le même formalisme.



(a) Bords simples

(b) Bords subdivisés avec leurs raccords

FIG. 53 – Bords associés aux subdivisions quadrangulaires

Notons Π_u, $\Pi_{\bar{u}}$, Π_v et $\Pi_{\bar{v}}$ les plongements associés aux bords u, $\bar{u}$, v et $\bar{v}$ (figure 53). Les deux types de contraintes vont impliquer deux types d'équations différentes :

- Contrainte interne (voir figure 54b). Étant donnée une feuille γ du quadtree et son modèle associé $(P^\gamma, \mathbb{T}^\gamma)$, les contraintes impliquent uniquement $\mathbb{T}^\gamma$.
- Contrainte externe (voir figure 54a). Étant données deux feuilles adjacentes γ et α, les contraintes impliquent les deux modèles complets associés $(P^\gamma, \mathbb{T}^\gamma)$ et $(P^\alpha, \mathbb{T}^\alpha)$.

Cela nous mène à une représentation unifiée des contraintes.

Proposition 11 (condition de raccord) *Soit $(P^\gamma, \mathbb{T}^\gamma)_{\gamma\in\Gamma}$ un quadtree d'attracteurs projetés. La condition pour que la fonction qui définit sa surface soit C^0 est :*

$$P^\gamma\, T^\gamma_{\beta_1} \dots T^\gamma_{\beta_k}\, \Pi_{\bar{u}} = P^\alpha\, T^\alpha_{\delta_1} \dots T^\gamma_{\delta_l}\, \Pi_u$$

avec $|\gamma| + |\beta| = |\alpha| + |\delta| > 0$ et $\gamma\,\beta \overset{u\bar{u}}{\Upsilon} \alpha\,\delta$. $\overset{u\bar{u}}{\Upsilon}$ désigne la relation d'adjacence horizontale de deux carreaux. À cela, on doit ajouter les contraintes verticales qui s'écrivent de la

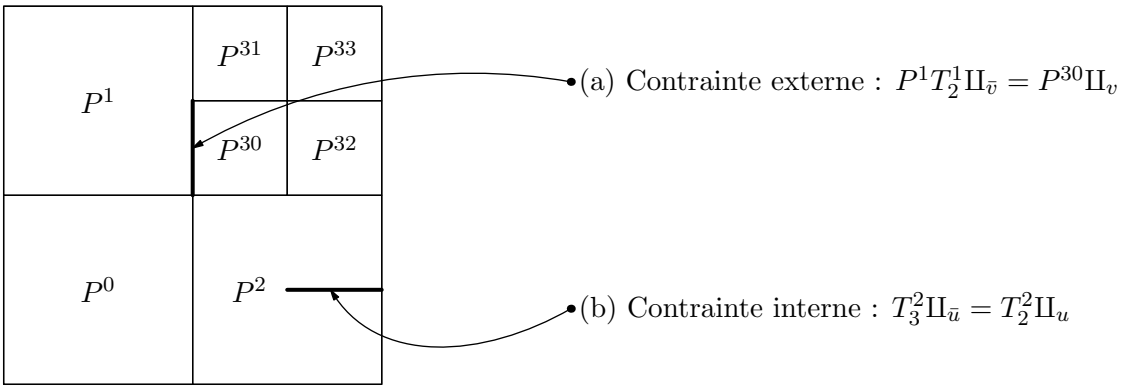


FIG. 54 – Contraintes internes et externes

même façon avec les bords v *et* $\bar{v}$*, les plongements associés* $\amalg_v$ *et* $\amalg_{\bar{v}}$*, ainsi que la relation d'adjacence verticale* $\overset{v\bar{v}}{\Upsilon}$*.*

Les raccords de carreaux sont expliqués plus en détail dans l'annexe A page 153.

1.3 Exemple

Nous allons ici montrer comment, grâce à un quadtree d'IFS projetés, il est possible de modéliser des surfaces beaucoup plus générales et surtout plus hétérogènes.

La figure 55 page suivante montre l'exemple d'une surface composée de plusieurs modèles dans une structure de quadtree à deux niveaux. La figure 56 page ci-contre montre la structure en quadtree associée à cette surface (a) ainsi que les modèles qui ont été utilisés (b). Parmi eux, on trouve des modèles allant de 3×3 à 9×9 points de contrôle. Certains modèles sont lisses, cela signifie que les coefficients des matrices de subdivision ont été choisis pour donner des propriétés de lissage. D'autres sont rugueux ; en pratique, une simple perturbation des coefficients permet d'obtenir un modèle rugueux.

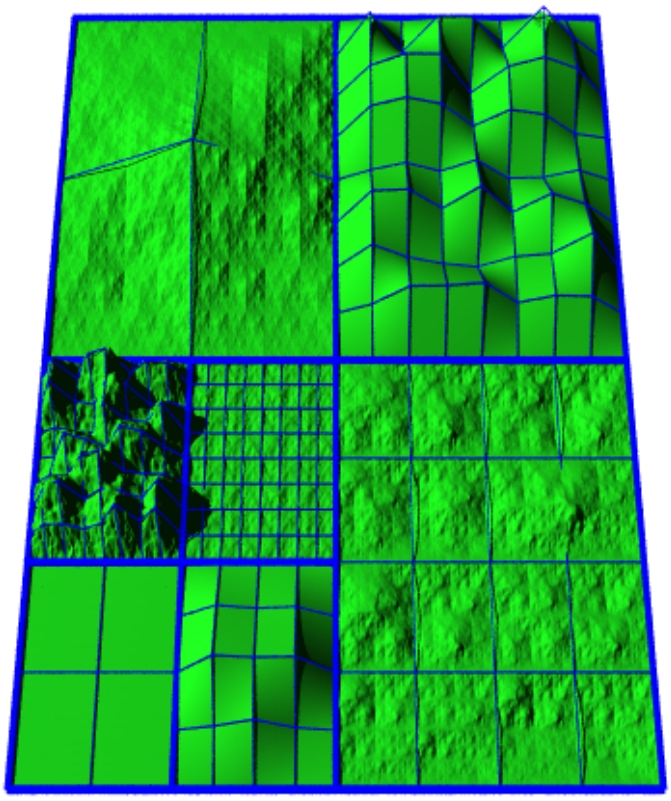


FIG. 55 – Exemple de surface composée

<table>
<tr><td colspan="2">1</td><td rowspan="2">3</td></tr>
<tr><td>01</td><td>03</td></tr>
<tr><td>00</td><td>02</td><td>2</td></tr>
</table>

(a) Structure et adresses

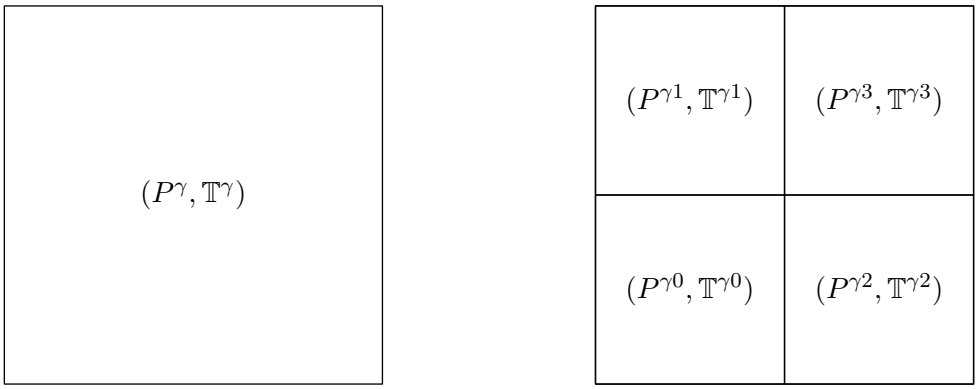
<table>
<tr><td colspan="2">3 × 3
rugueux</td><td>9 × 9
lisse</td></tr>
<tr><td>9 × 9
rugueux</td><td>9 × 9
rugueux</td><td rowspan="2">5 × 5
rugueux</td></tr>
<tr><td>3 × 3
lisse</td><td>5 × 5
lisse</td></tr>
</table>

(b) Types de modèles utilisés

FIG. 56 – Structure en quadtree et modèles associés

1.4 Raffinement

Grâce à cette structure en quadtree, nous introduisons le concept de raffinement. Ceci consiste à subdiviser une feuille du quadtree γ, pour la remplacer par quatre nouvelles feuilles $\gamma 0$, $\gamma 1$, $\gamma 2$, $\gamma 3$, contenant des IFS projetés indépendants. La figure 57 illustre ce procédé de raffinement. À gauche se trouve le modèle initial, ne possédant qu'une feuille, et à droite le modèle raffiné. Le quadtree raffiné contient quatre IFS projetés indépendants, cela signifie qu'il y a plus de points de contrôle, mais aussi que les matrices de subdivision peuvent être différentes (les conditions de raccords étant mises de côté). Des zones dont les caractéristiques varient spatialement peuvent ainsi être modélisées.



(a) Modèle initial à une feuille

(b) Modèle rafiné composé de quatre feuilles indépendantes

FIG. 57 – Principe de raffinement du quadtree

Grâce à ce principe, il est donc possible de construire un algorithme d'approximation récursif, qui raffine le quadtree jsuqu'à obtention d'une précision voulue. C'est l'objet du chapitre suivant.

Chapitre 2

Méthode fondée sur un critère de distorsion globale

La méthode présentée ici permet, étant donné une distorsion maximale globale tolérée, de trouver le quadtree d'IFS projetés qui répartit de manière conservative cette distorsion. Cela signifie que chaque feuille du quadtree contient un modèle qui génère un carreau de surface (ou image) ayant une distorsion relative inférieure au seuil fixé. Cette méthode est assez facile à implémenter mais – de manière assez évidente – ne fournit pas un résultat optimal en termes de débit/distorsion. Le chapitre suivant présente une méthode qui fournit un résultat optimal, mais qui est beaucoup plus coûteuse en temps de calcul.

2.1 Principe

Étant donnée la nature hiérarchique du modèle de surfaces composées, il paraît assez logique d'implémenter un algorithme d'approximation récursif. Le principe en est simple, on dispose d'une liste de modèles simples pouvant servir à approximer un bout de surface. Si, à l'aide de tous ces modèles, même le plus complexe, le critère de distorsion maximum n'est pas atteint, le bout de surface est subdivisé en quatre parties, et l'algorithme est appliqué sur chacune de ces parties. Au fur et à mesure, l'arbre est construit, et à chaque feuille est associé un modèle.

2.2 En pratique

Dans la pratique, l'algorithme n'est pas exactement exécuté dans les conditions idéales. En effet, lorsque l'image est grande, tenter une approximation par un seul modèle mène en général à un échec[3]. De plus cette étape est très coûteuse du fait du grand nombre de points. Ainsi, des subdivisions sont faites automatiquement jusqu'à l'obtention de carreaux de surface d'une taille raisonnable, c'est à dire plus grande que la taille de la grille de contrôle.

[3] à moins de l'appliquer à une surface complètement plane !

Nous avons utilisé quatre types de modèles différents, avec un degré de complexité croissant :

- 3×3 points de contrôle, 2×2 transformations (37 paramètres)
- 5×5 points de contrôle, 2×2 transformations (53 paramètres)
- 5×5 points de contrôle, 4×4 transformations (137 paramètres)
- 9×9 points de contrôle, 4×4 transformations (193 paramètres)

Le point commun de ces modèles est qu'ils engendrent, à des degrés de subdivision différents, des surfaces de même taille, c'est à dire 17×17, 33×33, 65×65, *etc.* Ils permettent donc, d'une façon assez aisée, d'approximer la même surface sans avoir à mettre en œuvre des algorithmes de redimensionnement.

2.3 Résultats

Sont présentés ici quelques résultats concernant l'approximation de surfaces et d'images avec la méthode décrite précédemment.

2.3.1 Surfaces

La figure 58 page suivante présente la surface originale. Il s'agit du massif central dont les données sont issues des DTED[Fra]. Cette surface est formée de 257×257 points. L'algorithme ne prend qu'un paramètre : la distorsion maximale tolérée. En pratique, on exprime celle-ci par le PSNR (Peak Signal-to-Noise Ratio) minimum, cette valeur étant relative (car rapportée au pixel), chaque carreau d'approximation doit avoir un PSNR supérieur à la valeur fixée.

Pour la figure 59, le PSNR a été fixé à $30\,dB$. On peut voir que dans ce cas là, le quadtree (figure 59b) généré par la méthode est simple : il ne comporte que deux subdivisions portant le nombre de feuilles à 16. Dans cette figure, les éléments du quadtree ont été coloriés par niveaux de gris, rendant ainsi compte de la complexité du modèle (en terme de nombre de paramètres) associée à la feuille. Plus le niveau de gris d'une feuille est sombre, plus le modèle local est complexe. Le modèle est simple, mais le rendu de la surface n'est pas suffisamment précis. En approximant avec un critère de $35\,dB$, la surface devient plus proche de l'originale (figure 60a), et le modèle plus complexe (figure 60b). La meilleure approximation est obtenue en donnant un critère de $40\,dB$ (figure 61). Pour chacune des approximations, nous avons indiqué la valeur réelle finale du PSNR entre la surface originale et celle approximée. Par construction, le PSNR final doit être supérieur à celui fixé. Pour le dernier cas, on note cependant que ce n'est pas vrai. Ceci est simplement dû au fait que l'on s'interdit de descendre trop en profondeur dans le quadtree. Une dernière figure montre l'approximation de la surface du Massif Central, mais avec une profondeur de récursion autorisée plus importante. On peut alors atteindre le critère de distorsion que l'on s'était fixé (figure 62). Dans ce cas précis, nous avons limité le nombre de modèles utilisés à deux.

En effet, des modèles trop complexes appliqués à des portions de surface trop petites engendrent des problèmes numériques. Lorsque le nombre de paramètres pour coder cette portion de surface devient plus grand que le nombre de points de la surface, les matrices

de résolution deviennent singulières. Ceci explique la différence entre les deux quadtree sur certaines zones où la profondeur maximale n'était pas atteinte.

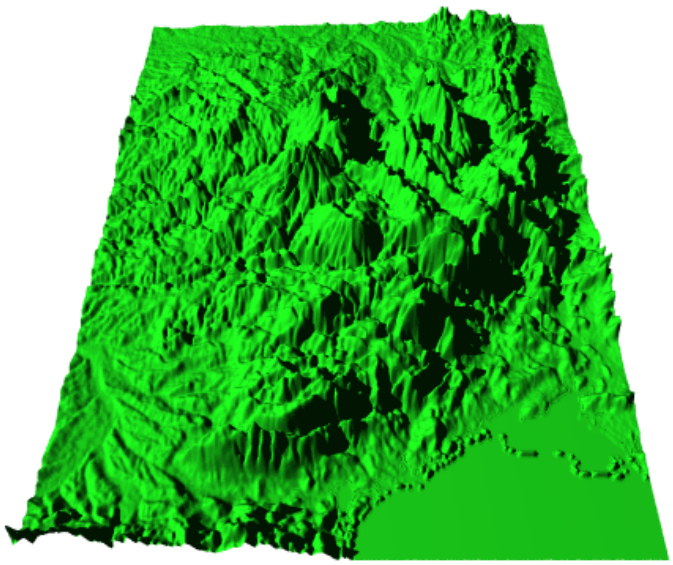
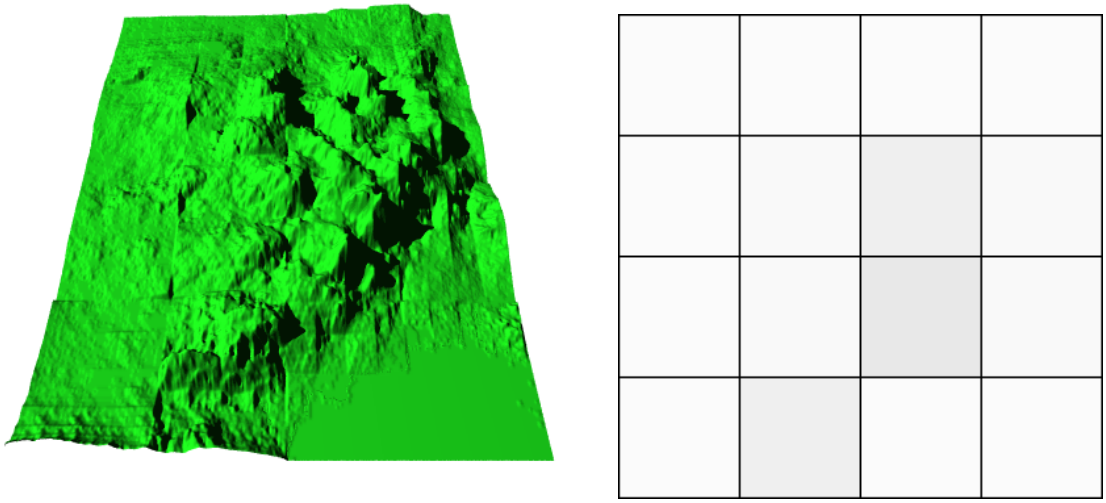


FIG. 58 – Surface originale : le massif central

La figure 63a montre la surface originale d'un terrain dans la région de Dijon. Cette surface est formée de 129×129 points. La figure 63b donne le résultat de l'approximation avec un critère de $25\,dB$. La figure 64 donne le résultat de l'approximation avec des critères de $30\,dB$ et $35\,dB$.

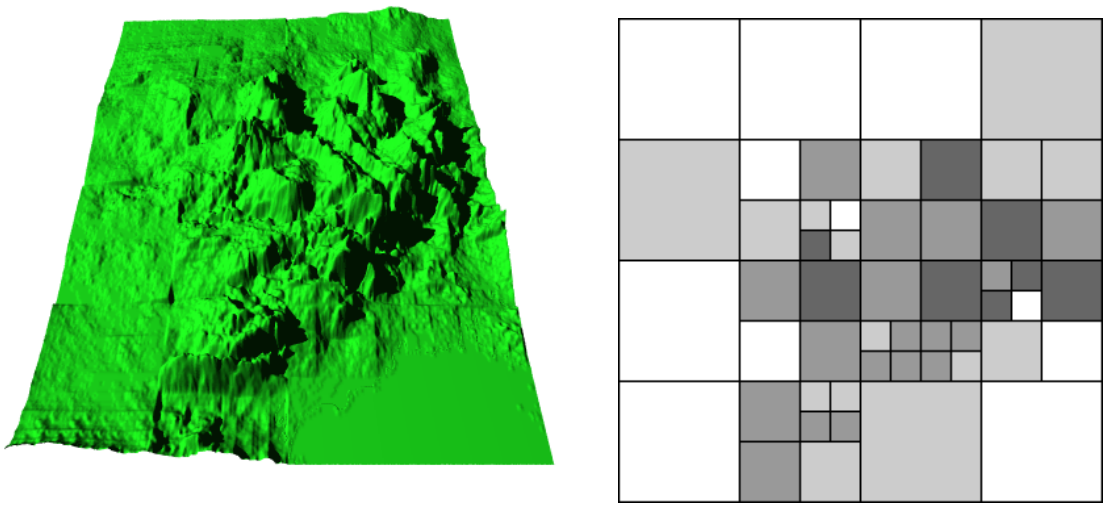
2.3.2 Images

Nous montrons ici l'approximation de deux images en niveaux de gris. La première image est celle d'un avion de chasse (portion 257×257 de l'image *airplane*). La figure 65 montre l'original de cette image. La figure 66 montre le résultat de l'approximation pour deux valeurs de PSNR : $25\,dB$ et $30\,dB$. La deuxième image est une portion 129×129 de l'image *Lena* en niveaux de gris. La figure 67 montre l'original de l'image. La figure 68 montre l'approximation de celle-ci pour deux valeurs de PSNR : $26\,dB$ et $30\,dB$.



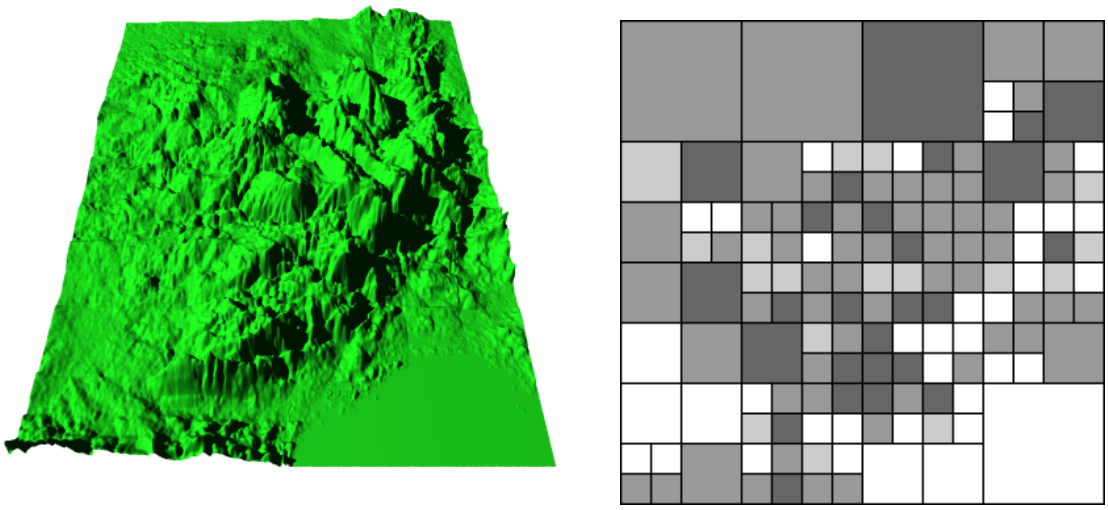
(a) Surface approximée (b) Quadtree généré

FIG. 59 – Approximation à PSNR $> 30\,dB$, PSNR réel : $33,1\,dB$



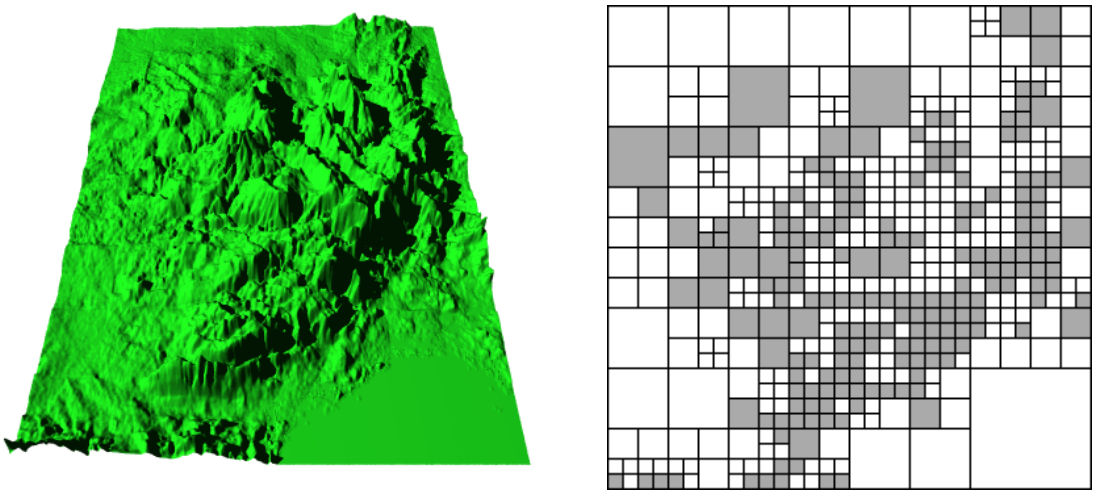
(a) Surface approximée (b) Quadtree généré

FIG. 60 – Approximation à PSNR $> 35\,dB$, PSNR réel : $36,2\,dB$



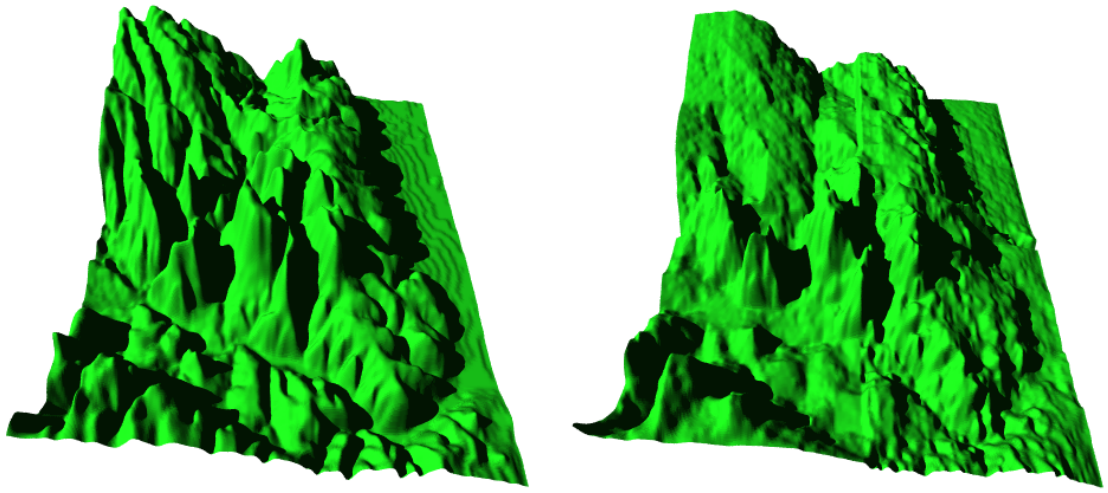
(a) Surface approximée (b) Quadtree généré

FIG. 61 – Approximation à PSNR $> 40\,dB$, PSNR réel : $39,0\,dB$



(a) Surface approximée (b) Quadtree généré

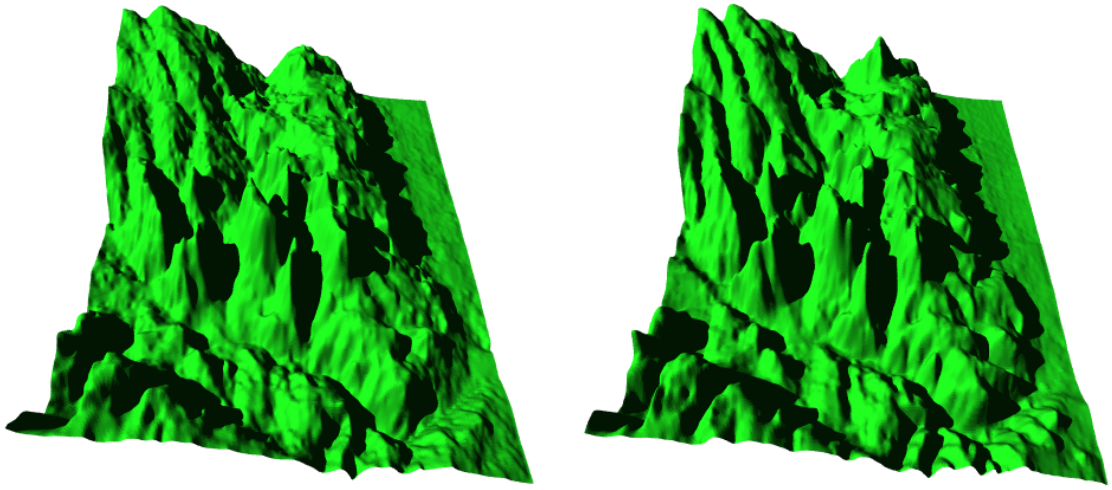
FIG. 62 – Approximation à PSNR $> 40\,dB$, PSNR réel : $41,6\,dB$ – profondeur de quadtree plus importante



(a) Surface originale

(b) Surface approximée à PSNR $> 25\,dB$, PSNR réel : $26,2\,dB$

FIG. 63 – Surface originale et approximation à $25\,dB$



(a) Surface approximée à PSNR $> 30\,dB$, PSNR réel : $31,2\,dB$

(b) Surface approximée à PSNR $> 35\,dB$, PSNR réel : $35,0\,dB$

FIG. 64 – Approximations à 30 et $35\,dB$

FIG. 65 – Image originale, portion de l'image *airplane*

(a) Avec PSNR $> 25\,dB$

(b) Avec PSNR $> 30\,dB$

FIG. 66 – Images approximées (*airplane*)

FIG. 67 – Image originale, portion de l'image *Lena*

(a) Avec PSNR $> 26\,dB$ (b) Avec PSNR $> 30\,dB$

FIG. 68 – Images approximées (*Lena*)

2.4 Conclusion

Pour cette méthode fondée sur un critère de distorsion globale, les résultats d'approximation des surfaces comme des images sont satisfaisants. On peut en effet atteindre n'importe quel niveau de précision. Mais nous ne nous sommes pas souciés de la complexité du quadtree d'IFS projetés généré. Par ailleurs, nous n'avons pas soulevé le problème du codage de ce quadtree et des modèles fractals utilisés dans le but de comprimer les données. Le chapitre suivant répond à ces attentes en proposant une méthode complète d'optimisation du codage avec un critère de débit.

Chapitre 3

Codage optimal fondé sur un critère de coût

Dans ce chapitre, nous allons nous pencher plus vers le problème de la compression de données (représentant des surfaces et des images) en mettant en œuvre une chaîne complète de traitement. Cette chaîne comprend les éléments suivants :

- Recherche des différentes représentations
- Quantification des paramètres de l'espace de représentation
- Codage des paramètres quantifiés
- Optimisation du choix de la représentation avec un critère de coût donné
- Codage final

Pour qu'une optimisation finale soit possible, il faut disposer de plusieurs représentations de la surface. Chaque représentation de la surface correspond à une surface composée, c'est à dire un quadtree d'IFS projetés. Il est alors pratique de disposer d'une analyse exhaustive de la surface.

3.1 Compression au sens débit/distorsion

En compression d'image, on utilise souvent le concept d'optimisation « au sens débit/distorsion ». Le débit associé à une image est la quantité d'information nécessaire à la reconstruction de cette image, avec une certaine erreur nommée distorsion. Ce débit peut être exprimé de façon absolue en bits, ou de façon relative par rapport à la taille de l'image : en bits par pixel. Optimiser au sens débit/distorsion, c'est à partir d'une contrainte de débit par exemple trouver la représentation qui va minimiser la distorsion associée. Dans la suite de ce chapitre, nous raisonnons en ces termes.

3.2 Recherche exhaustive de modèles

Alors que dans le chapitre précédent nous nous attachions à obtenir une distorsion uniforme, nous allons ici nous efforcer d'analyser la surface de façon exhaustive, le seul critère d'arrêt étant la taille minimale d'un bloc. L'algorithme 1 page suivante explique

cette recherche exhaustive. Une fois cette opération terminée, nous avons donc à disposition un très grand choix de modèles IFS projetés locaux pour décrire chaque partition de la surface. Pour un carreau donné associé à l'adresse γ d'un nœud dans le quadtree, nous disposons alors d'un ensemble de couples modèle/distorsion $(P_i^\gamma, \mathbb{T}_i^\gamma), D_i^\gamma$, avec $i = 1, \ldots, n_{\text{mod}}$, n_{mod} étant le nombre de modèles différents utilisés pour l'approximation d'un même carreau. Le nombre de descriptions possibles pour l'image entière devient assez vite important. Avec une profondeur de 2 nous avons déjà $n_{\text{mod}}^4 + n_{\text{mod}}$ possibilités.

Algorithme 1 : Approximation récursive d'une surface.
Entrée : Q : la surface à approximer
γ : mot représentant l'endroit où l'on se trouve dans le quadtree
TailleMin : la taille minimum d'un bloc à approximer
Sortie : Pas de sortie, les résultats sont stockés grâce à la fonction MÀJ-QUADTREE

```
APPROXIMATION-SURFACE(Q, γ, TailleMin)
    pour chaque modèle i
        (P, T) ← APPROXIMATION-SIMPLE(Q^γ, i)
        MÀJ-QUADTREE(γ, P, T, i)
    si TAILLE(Q^γ) > TailleMin
    alors
        pour k = 0 à 3
            APPROXIMATION-SURFACE(Q, γ k, TailleMin)
```

3.3 Distribution des coefficients

Que ce soit pour la quantification ou le codage, il est indispensable de connaître de façon précise la distribution des coefficients qu'il va falloir traiter. Pour cela, nous différencions deux types de coefficients : ceux qui décrivent les points de contrôle et ceux qui décrivent les transformations.

3.3.1 Points de contrôle

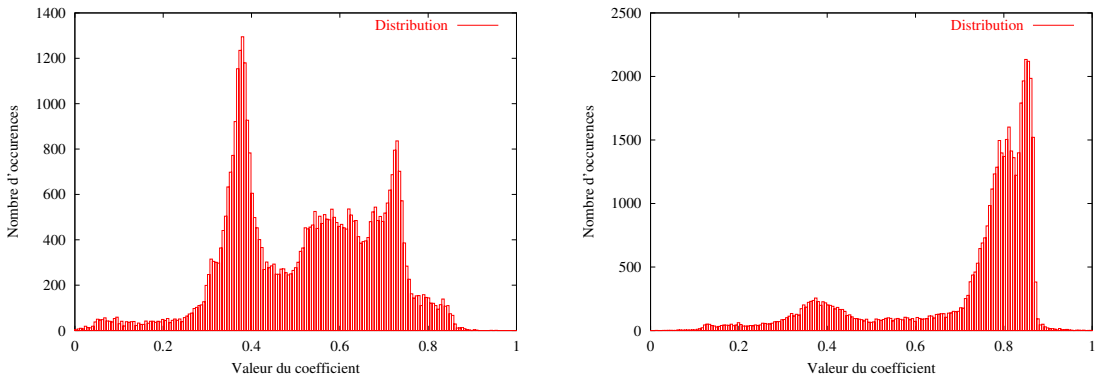
Comme cela est prévisible, la distribution de ces coefficients est directement liée à la distribution des altitudes dans la surface, ou des niveaux de gris dans l'image. Ceci est du au caractère interpolant des surfaces de subdivision employées pour l'approximation. La figure 70 montre la distribution de ces coefficients pour les deux images de la figure 69. La figure 71 montre pour les deux mêmes images la distribution des altitudes. Le lien entre les deux est alors clair.

3.3.2 Matrices de subdivision

À l'opposé, les coefficients des matrices de subdivision s'avèrent avoir une distribution qui dépend beaucoup moins des données à traiter. Celle-ci semble en effet obéir en

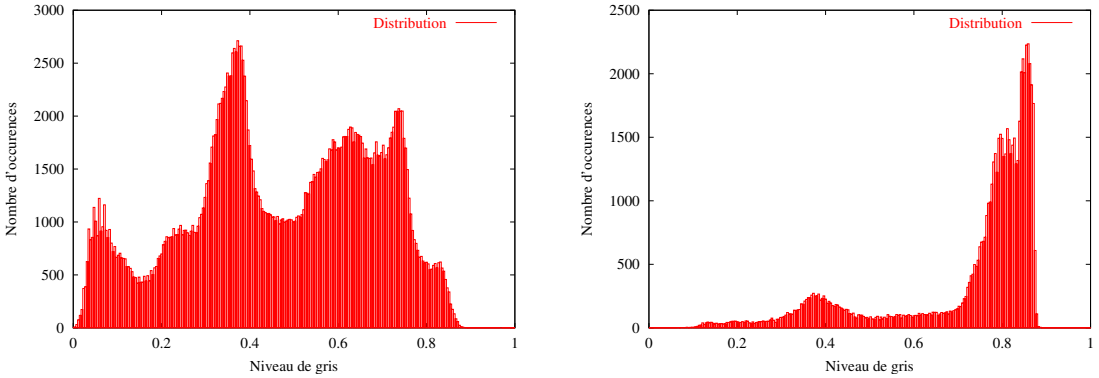
(a) peppers (b) airplane

FIG. 69 – Les deux images analysées



(a) peppers (b) airplane

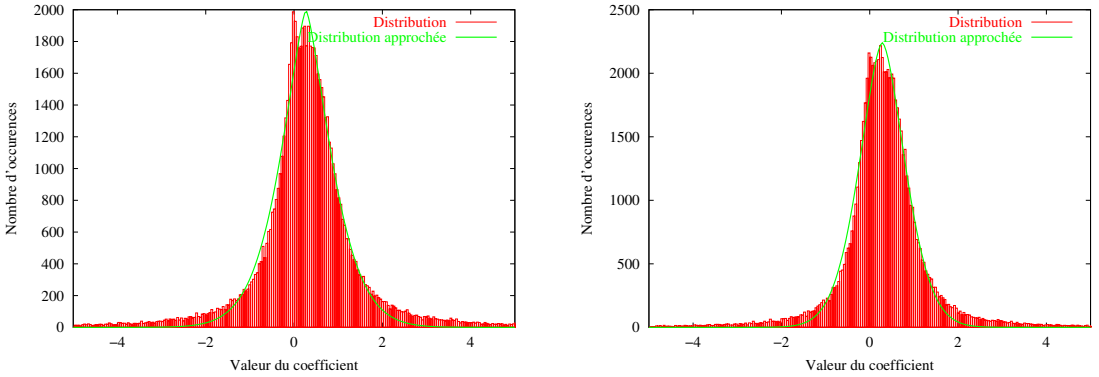
FIG. 70 – Distributions des valeurs des scalaires de contrôle



(a) peppers (b) airplane

FIG. 71 – Distributions des niveaux de gris

bonne approximation à une loi gaussienne généralisée. La figure 72 montre la distribution des valeurs de coefficients de matrices de subdivision issus de l'analyse des deux images précédentes.



(a) peppers (b) airplane

FIG. 72 – Distributions des valeurs de coefficients des matrices de subdivision

3.4 Quantification

Nous avons expérimenté deux méthodes de quantification. La première, classique, est uniforme. Elle consiste à découper un intervalle de façon uniforme. La deuxième consiste à rapprocher les valeurs lorsque celles-ci sont plus nombreuses au sens de la distribution. Commençons par définir ce que nous entendons par quantification.

Définition 18 (quantification) Une quantification en ν valeurs est la donnée d'un vecteur de $2\nu - 1$ scalaires tels que :

$$q_1 < x_1 < q_2 < x_2 < \ldots < q_{\nu-1} < x_{\nu-1} < q_\nu$$

À ces valeurs sont associés des intervalles :

$$\begin{array}{rcl} Q_1 & = &]-\infty, x_1] \\ Q_2 & = &]x_1, x_2] \\ & \vdots & \\ Q_{\nu-1} & = &]x_{\nu-2}, x_{\nu-1}] \\ Q_\nu & = &]x_{\nu-1}, \infty[\end{array}$$

La quantification consiste alors à associer à toute valeur de x appartenant à l'intervalle Q_j la valeur quantifiée q_j.

3.4.1 Quantification uniforme

Cette méthode de quantification ne tient pas compte de la distribution des coefficients. Elle se contente de prendre une valeur minimum, une valeur maximum, et de répartir de façon uniforme les valeurs sur cet intervalle.

3.4.2 Quantification adaptative

Avec cette méthode, il va être possible d'augmenter la précision des valeurs lorsque celles-ci sont plus fréquentes. Le but est de minimiser la distorsion finale des coefficients.

Les premiers travaux concernant ce type de quantification optimale ont été publiés par Max, ils concernent uniquement des variables dont la distribution est gaussienne[Max60]. Plus tard, Lloyd étudie le cas plus général et retrouve les mêmes résultats pour les distributions gaussiennes[Llo82]. C'est cette méthode que nous avons appliquée. Elle permet, à partir d'une distribution donnée et d'un nombre de valeurs recherché, de calculer l'ensemble des valeurs quantifiées ainsi que les intervalles associés à celles-ci. Pour cela, Lloyd démontre que la meilleure quantification (au sens de la distorsion) possède les deux propriétés suivantes :

– q_α doit être le centre d'inertie de l'intervalle Q_α :

$$q_\alpha = \frac{\int_{x \in Q_\alpha} x \, dF(x)}{\int_{x \in Q_\alpha} dF(x)}$$

– x_α doit se trouver au centre des deux valeurs quantifiées qui l'entourent :

$$x_\alpha = \frac{1}{2}(q_\alpha + q_{\alpha+1})$$

La méthode développée par Lloyd prend des valeurs aléatoires pour initialiser les x_α, puis tour à tour effectue deux opérations qui ne peuvent que diminuer la distorsion :

– Les nouvelles valeurs de q_α sont les centres d'inertie des intervalles Q_α

– Les nouvelles valeurs de x_α sont les centres de q_α et $q_{\alpha+1}$

Comme la distorsion est positive, l'algorithme ne peut que converger.

> **Remarque :** Dans le même article[Llo82], un autre algorithme est donné, utilisant une valeur initiale pour le premier intervalle, construisant la quantification et modifiant cette valeur initiale jusqu'à convergence. Nous l'avons expérimenté et il semble plus lent que l'autre.

La figure 73 donne un exemple de quantification uniforme sur 5 bits. La figure 74 montre, pour les mêmes données, une quantification adaptative sur 5 bits.

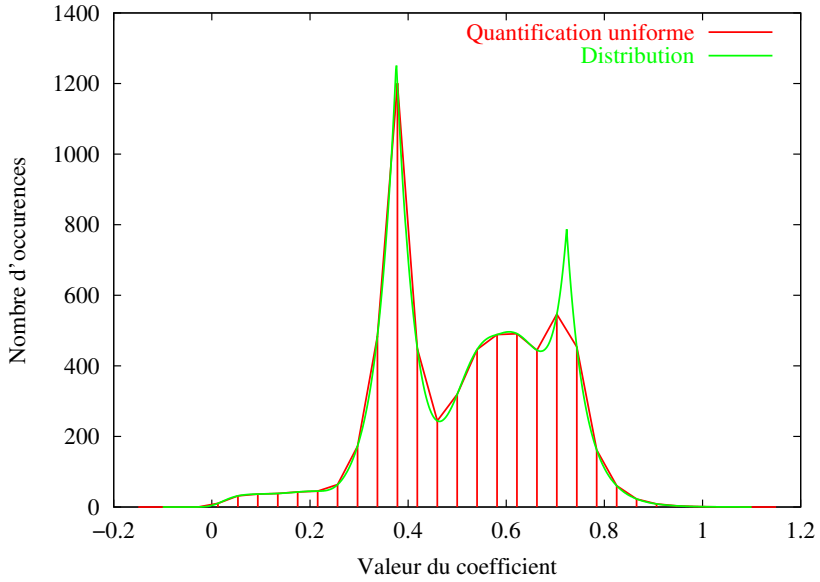


FIG. 73 – Quantification uniforme

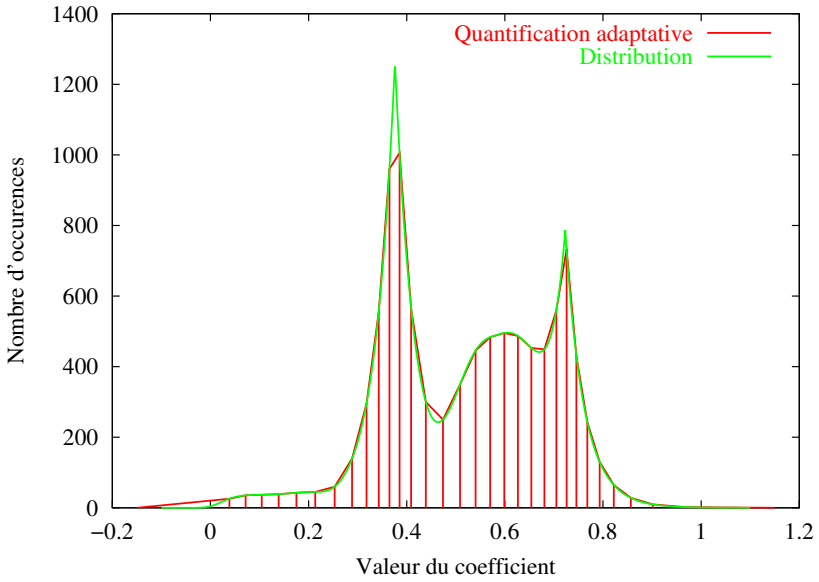


FIG. 74 – Quantification adaptative

3.5 Codage

Une seule méthode de codage a été expérimentée. Il s'agit d'un codage arithmétique fondé sur les distributions réelles des coefficients issus de l'analyse de la surface [PFTV93c]. L'avantage de se baser sur une distribution réelle est de mieux coller à la réalité, ce qui permet un codage plus performant. L'inconvénient est qu'il faut modéliser les distributions et coder les paramètres de cette modélisation. Il faut donc choisir un modèle performant pour la description des distributions. Ce modèle est bien entendu différent pour les scalaires de contrôle et pour les transformations.

3.5.1 Points de contrôle

La distribution des scalaires de contrôle dépend énormément de l'image ou de la surface considérée. Nous avons choisi de modéliser celle-ci grâce à la somme de plusieurs gaussiennes généralisées – quatre au maximum suivant la complexité de la distribution.

La méthode proposée pour trouver les paramètres de ces gaussiennes se décompose en deux phases :

- Initialisation. La courbe de la distribution est lissée plusieurs fois jusqu'à l'obtention d'une courbe qui ne possède que quatre maxima. Ceux-ci servent à localiser les pics les plus significatifs ainsi qu'à leur attribuer une hauteur approchée. Chaque degré est initialisé à une valeur de 2 (valeur pour une gaussienne non généralisée).
- Optimisation. Grâce à une méthode fondée sur les dérivées littérales, une optimisation des paramètres est effectuée.

La figure 75 page suivante montre le résultat de l'optimisation avec la décomposition en quatre gaussiennes généralisées.

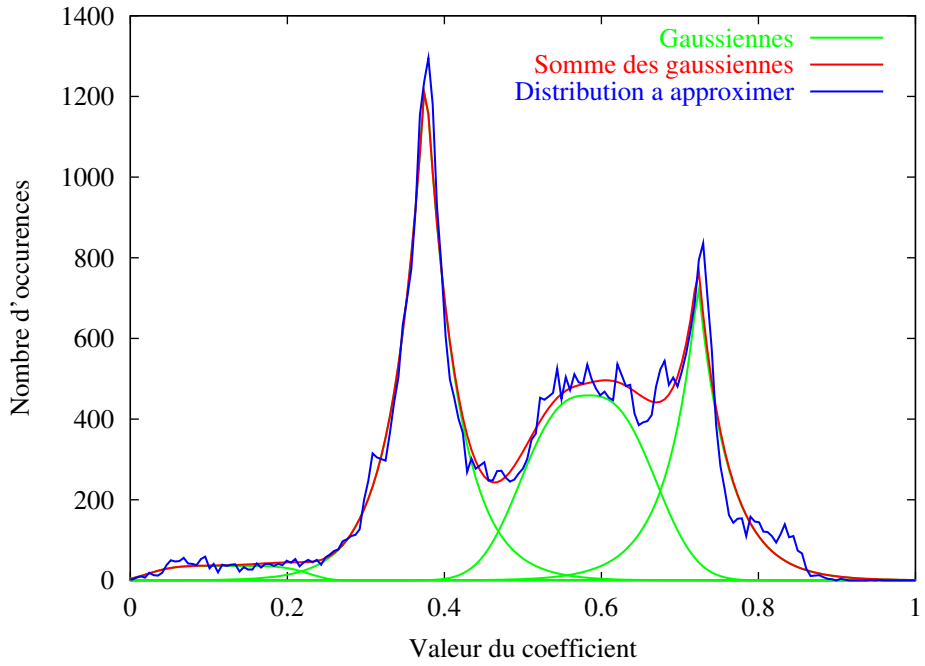


FIG. 75 – Approximation de la distribution en la somme de quatre gaussiennes

3.5.2 Transformations

Pour les coefficients des transformations, la modélisation est effectuée grâce à une gaussienne généralisée. La figure 72 page 120 montre le résultat de cette approximation. Une méthode identique à celle utilisée pour les coefficients de points de contrôle est utilisée : localisation du pic par lissages successifs, optimisation des paramètres de la gaussienne généralisée grâce à une méthode utilisant les dérivées.

3.6 Optimisation

Nous disposons donc d'une multitude de façons de décrire la surface ou l'image, avec des degrés de distorsion variés. La phase d'optimisation consiste à trouver, le débit étant fixé, la description qui donne la distorsion la plus faible. Nous avons utilisé une méthode de bissection inspirée de RAMCHANDRAN et VETTERLI[RV93]. Cette méthode utilise la notion de multiplicateur de LAGRANGE que nous allons expliquer brièvement.

3.6.1 Multiplicateur de Lagrange

Les multiplicateurs de LAGRANGE, d'une façon générale, sont utilisés pour trouver l'extremum d'une fonction de plusieurs variables soumise à une contrainte. Dans notre cas, les variables sont les paramètres de modélisation, c'est à dire le choix de la représentation (paramètres discrets donc). La fonction à minimiser est D et la contrainte est $R - R_{\max} < 0$. Introduisons maintenant le multiplicateur de LAGRANGE λ et la fonction

de *coût lagrangien* associée :

$$J = D + \lambda(R - R_{\max})$$

λ étant fixé, il est alors possible de minimiser le coût lagrangien. Comme $R_{\max}$ est une constante on peut se contenter de la minisation de $J = D + \lambda R$. La figure 76 illustre ce principe. La traduction géométrique est assez intuitive, fixer λ revient à fixer une pente sur le graphe de D en fonction de R. Ainsi, la minimisation de J revient à faire glisser une droite jusqu'au dernier point (R, D). Les points trouvés se trouvent donc sur l'enveloppe convexe de la totalité des points (R_i, D_i). Il faut donc maintenant élaborer une méthode

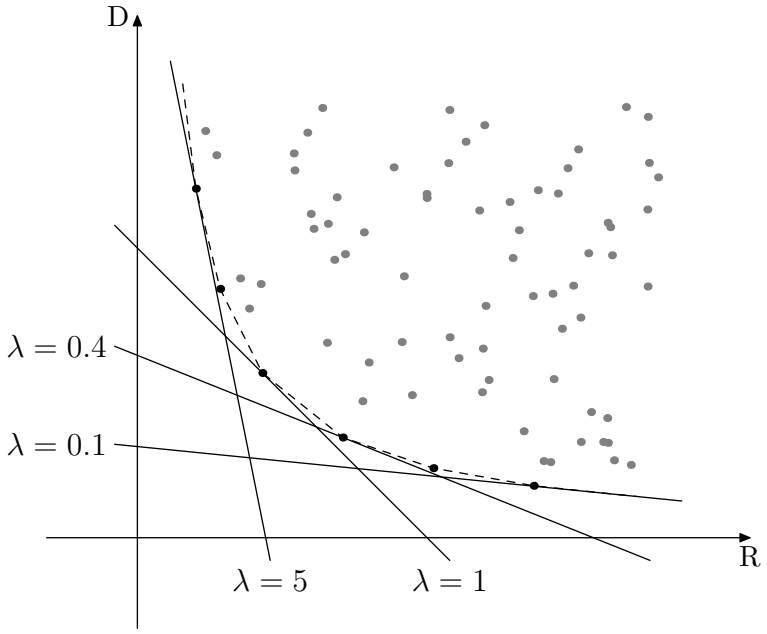


Fig. 76 – Multiplicateur de Lagrange

qui permet de trouver quelle description va minimiser le coût lagrangien.

3.6.2 Minimisation du coût lagrangien

Le multiplicateur de Lagrange étant fixé, il faut un algorithme capable de trouver rapidement quelle description minimise le coût lagrangien. Étant donnée la nature hiérarchique du modèle, un algorithme très simple, récursif, permet d'y parvenir. Considérons que l'on se trouve à un nœud γ. Il est facile de calculer le minimum du coût lagrangien pour tous les modèles de ce nœud, ainsi que pour toutes les quantifications. Il convient de garder ce nœud si la somme des quatre minima des sous-noeuds est supérieure au minimum trouvé pour le nœud ou bien s'il n'y a pas de sous-nœuds. L'algorithme 2 page suivante donne l'algorithme récursif correspondant.

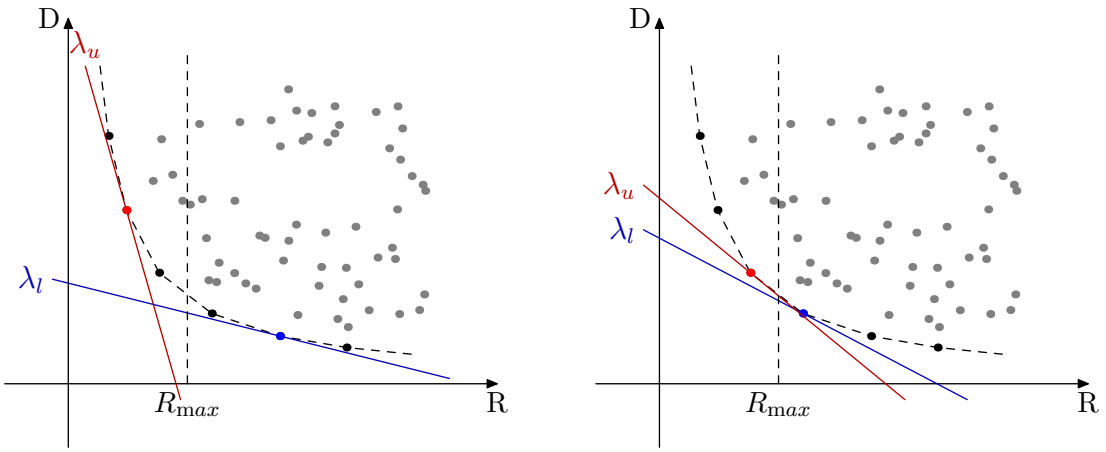
Algorithme 2 : Minimisation du coût lagrangien.
Entrée : γ : nœud du quadtree
λ : multiplicateur de Lagrange
Sortie : La valeur minimum du coût lagrangien
MINIMISE-LAGRANGE(γ, λ)
$J_{\min} \leftarrow$ CALCULE-COÛT-LAGRANGIEN$(\gamma, \lambda, 1, 1)$
$i_{\min} \leftarrow 1$
$j_{\min} \leftarrow 1$
pour chaque modèle i
pour chaque quantificateur j
si CALCULE-COÛT-LAGRANGIEN$(\gamma, \lambda, i, j) < J_{\min}$
alors
$J_{\min} \leftarrow$ CALCULE-COÛT-LAGRANGIEN(γ, λ, i, j)
$i_{\min} \leftarrow i$
$j_{\min} \leftarrow j$
MÀJ-CHOIX-MODÈLE$(\gamma, i_{\min}, j_{\min})$
$J_{\text{sub}} \leftarrow 0$
pour $k = 0$ **à** 3
$J_{\text{sub}} \leftarrow J_{\text{sub}} +$ MINIMISE-LAGRANGE$(\gamma\, k, \lambda)$
si $J_{\text{sub}} < J_{\min}$
alors
MÀJ-CHOIX-MODÈLE$(\gamma, -1, -1)$

3.6.3 Algorithme de bissection

Cet algorithme est inspiré de l'article de RAMCHANDRAN et VETTERLI[RV93]. Nous introduisons la notation suivante : $R(\lambda)$ et $D(\lambda)$ sont les valeurs de R et D associées à la minimisation de $J(\lambda)$. Les étapes de l'algorithme sont les suivantes :

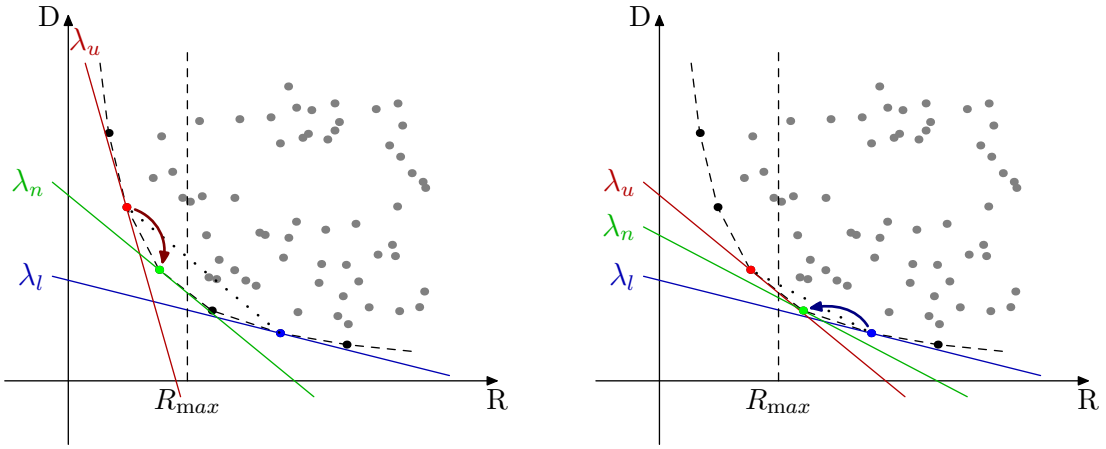
1. Initialisation : trouver des valeurs λ_{u} et λ_{l} telles que $R(\lambda_{\text{u}}) < R_{\max}$ et $R(\lambda_{\text{l}}) > R_{\max}$ (voir figure 77a).
2. Calculer λ_{n} :
$$\lambda_{\text{n}} \leftarrow \frac{D(\lambda_{\text{l}}) - D(\lambda_{\text{u}})}{R(\lambda_{\text{l}}) - R(\lambda_{\text{u}})} + \epsilon$$
3. Si $R(\lambda_{\text{n}}) = R(\lambda_{\text{u}})$, alors la solution est trouvée (voir figure 77b).
4. Sinon
   – Si $R(\lambda_{\text{n}}) > R_{\max}$ alors $\lambda_{\text{l}} \leftarrow \lambda_{\text{n}}$ (voir figure 78b)
   – Sinon $\lambda_{\text{u}} \leftarrow \lambda_{\text{n}}$ (voir figure 78a)
5. Aller à 2.

Remarque : L'algorithme de bissection donne un résultat optimal dans le sens où celui-ci se trouve sur l'enveloppe convexe. Cependant, il peut exister une solution plus proche du seuil fixé et donnant une meilleure approximation. Nous n'avons pas cherché à améliorer le codage en cherchant ce type de solution.



(a) Initialisation (b) Fin de l'algorithme

FIG. 77 – Algorithme de bissection – initialisation et fin



(a) Cas où $\lambda_u \leftarrow \lambda_n$ (b) Cas où $\lambda_l \leftarrow \lambda_n$

FIG. 78 – Algorithme de bissection – phases intermédiaires

3.7 Codage final

Ici nous décrivons comment est effectué le codage final de la surface. Celui-ci devra permettre la reconstruction totale de la surface. Le tableau 5 donne la structure de l'en-tête dans laquelle figurent des informations générales sur l'image. On peut noter que la taille de cet en-tête est faible. Pour une image de taille moyenne (257×257 par exemple) codée à très faible débit (0.05 bit/pixel par exemple), cet en-tête ne représente qu'une faible partie du code final ($\frac{1}{150}$). Vient ensuite le codage du quadtree et des coefficients.

Description	Taille (bits)
Taille de l'image	24
Version	8
Paramètres de la gaussienne généralisée (transformations)	24
largeur	8
localisation	8
exposant	8
Paramètres des quatres gaussiennes généralisées (points de contrôle)	120
première	24
deuxième	32
troisième	32
quatrième	32
Total	176

TAB. 5 – Structure de l'en-tête et bits alloués

Pour le quadtree, un codage simple (mais non optimal) utilise 1 bit par nœud. Pour ce qui est des coefficients de chaque modèle, il faut un en-tête pour indiquer quel type de modèle et quel type de quantification sont utilisés. Le tableau 6 donne le détail de cet en-tête avec le nombre de bits alloués. De la même façon que pour l'en-tête général, il reste de petite taille par rapport aux données. Si l'on veut coder 53 paramètres sur 5 bits chacun, que le codage arithmétique donne un ratio de $0,7$, la proportion de la taille de l'en-tête est de $\frac{1}{37}$, or il s'agit du cas le plus pénalisant.

Description	Taille (bits)
Type de modèle	2
Type de quantification	3
Codage des coefficients	...

TAB. 6 – Structure de l'en-tête de codage de chaque modèle IFS projeté

Remarque : L'en-tête général est déduit du budget fixé (car il est fixe), alors que les en-têtes de codage sont déduits au moment du calcul de chaque débit impliqué par tel modèle, telle quantification, à tel endroit du quadtree.

3.8 Résultats

Nous exposons ici les résultats de compression de surfaces et d'images par la méthode d'optimisation détaillée précédemment. Pour chaque exemple, l'algorithme effectue une compression à divers débits. En effet, la majorité du temps de calcul est prise par l'approximation récursive de la surface ou image avec divers modèles. Une fois cette analyse exhaustive faite, il est assez aisé d'appliquer la méthode d'optimisation du codage pour une série de débits donnés, l'arbre des modèles restant le même. Ainsi, pour chaque exemple, nous donnons un temps de calcul qui tient compte de la compression à tous les débits donnés. Il est important de noter aussi que le critère de débit donné n'est souvent pas atteint car on cherche un point optimal le plus proche. Ainsi, deux courbes débit/distorsion seront tracées pour chaque exemple : une courbe avec le débit nominal (fixé par le critère), et une autre avec le débit réel.

3.8.1 Surfaces

Nous exposons ici les résultats de compression de deux surfaces de profondeur. Il s'agit des mêmes données que dans le chapitre précédent, section 2.3 page 108. La surface originale, le massif central, est représentée sur la figure 79. Deux exemples de résultat de compression sont montrés dans la figure 80. Les débits nominaux sont $0,5$ bit/pixel[4] et $1,2$ bits/pixel respectivement pour les sous-figures a et b. Les débits réels, c'est à dire après optimisation, sont de $0,47$ et $1,00$ bit/pixel respectivement. La courbe de la figure 81 montre les valeurs de débits nominaux et réels, ainsi que les distorsions associées pour d'autres valeurs entre $0,05$ bit/pixel et 4 bits/pixel. Les résultats pour la deuxième surface sont présentés de la même manière : la figure 82 est la surface originale de la région de Dijon, la figure 83 représente deux résultats de compression et la figure 84 montre l'évolution de la distorsion en fonction du débit nominal et réel.

Même pour des débits faibles de l'ordre de $0,5$ bits/pixel (figures 80a et 83a), les résultats visuels sont convaincants. Seuls les détails les plus fins sont manquants. Ces détails apparaissent lorsque l'on monte en débit (figures 80b et 83b). La figure 85 montre un gros plan de la surface du massif central originale (a) et compressée à $0,5$ bit/pixel (b). On voit très nettement les différences. Lorsque le débit est plus élevé, les détails sont plus présents. La figure 86 fait apparaître ces détails en montrant le même gros plan pour des débits nominaux de $1,2$ et 2 bits/pixel. Le tableau 7 donne des informations sur les données des surfaces étudiées.

3.8.2 Images

Nous exposons ici les résultats de compression d'images en niveaux de gris sur trois exemples. Chacun de ces exemples sera agrémenté d'une courbe débit/distorsion, faisant apparaître la comparaison avec JPEG, la compression fractale [Jac92], et la compression EZW [Sha93]. Pour ce qui est de la méthode de compression fractale, il s'agit d'une compression par quadtree avec accélération par centres de masse, d'après le programme

[4] on devrait dire bit/point puisqu'il s'agit d'une surface

Surface	Massif Central	Région de Dijon
Figure (original)	79	82
Figure débit/distorsion	81	84
Figure (compression)	80	83
Taille	257×257	129×129
Temps de calcul[a]	$18'26''$	$5'10''$

TAB. 7 – Données numériques – compression de surfaces

[a]Matériel utilisé : Athlon 1, 2 GHz, 512 Mo

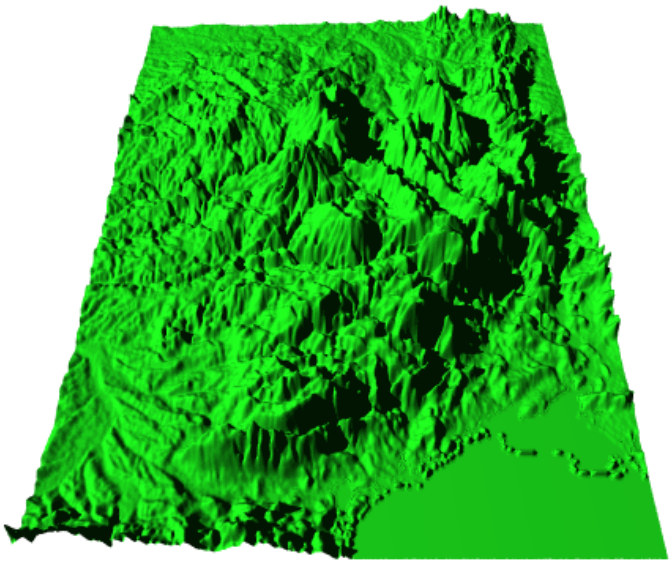
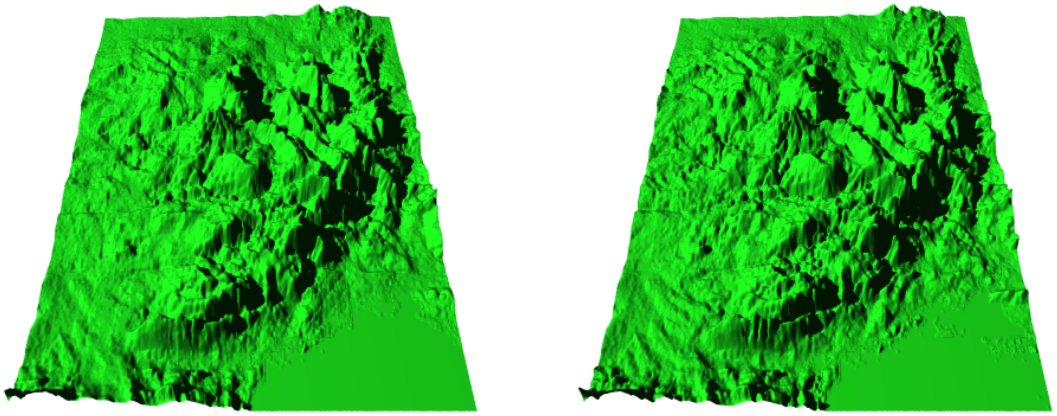


FIG. 79 – Massif Central – surface originale



(a) Débit nominal 0, 5 bit/pixel, réel 0, 47 bit/pixel

(b) Débit nominal 1, 2 bits/pixel, réel 1, 00 bit/pixel

FIG. 80 – Massif Central – deux exemples de compression

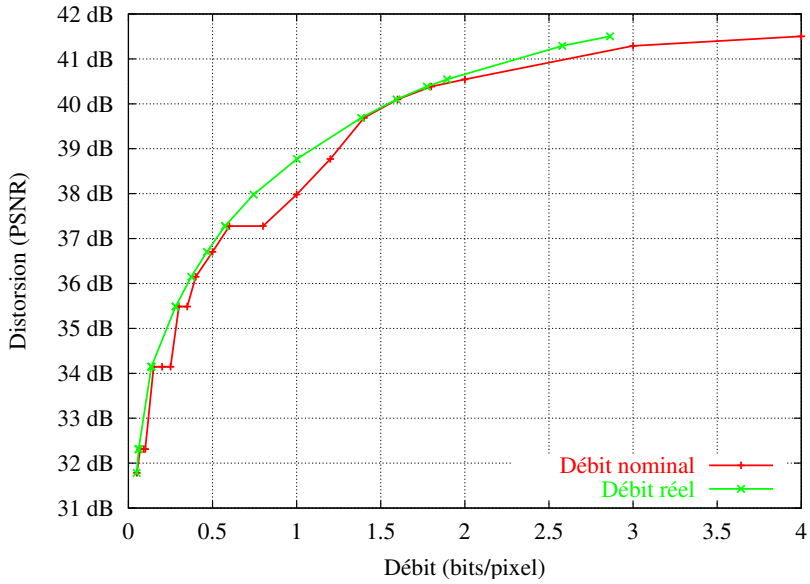


FIG. 81 – Massif Central – courbe débit/distorsion

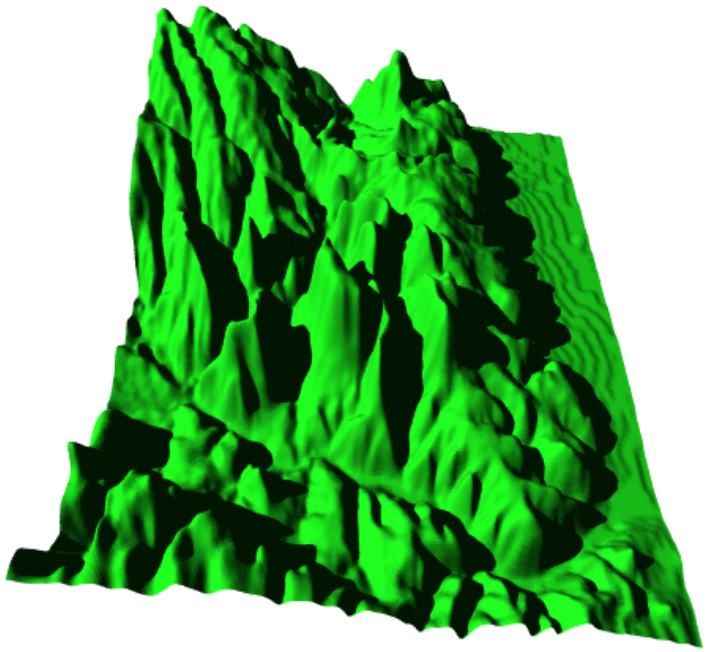
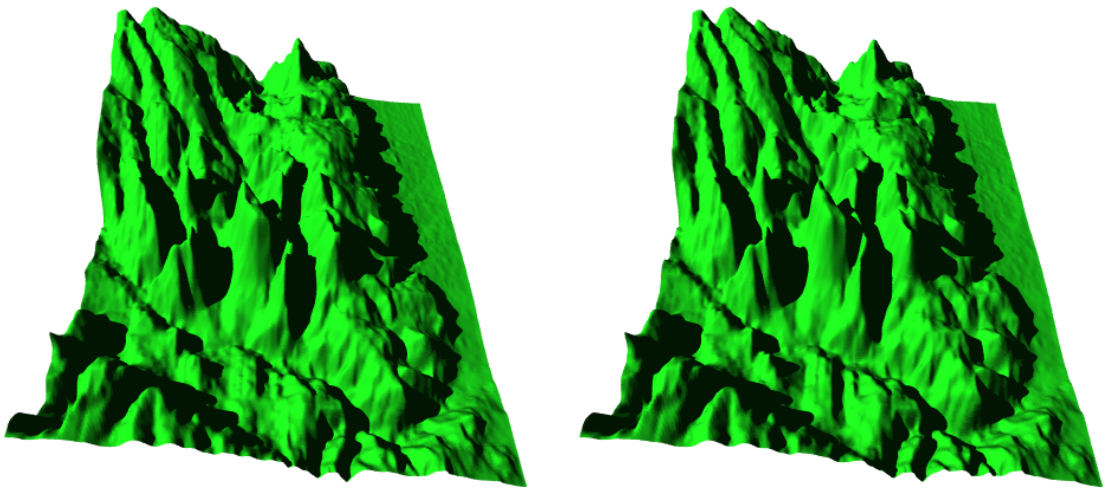


FIG. 82 – Région de Dijon – surface originale



(a) Débit nominal $0,6$ bit/pixel, réel $0,59$ bit/pixel

(b) Débit nominal $1,2$ bits/pixel, réel $1,11$ bit/pixel

FIG. 83 – Région de Dijon – deux exemples de compression

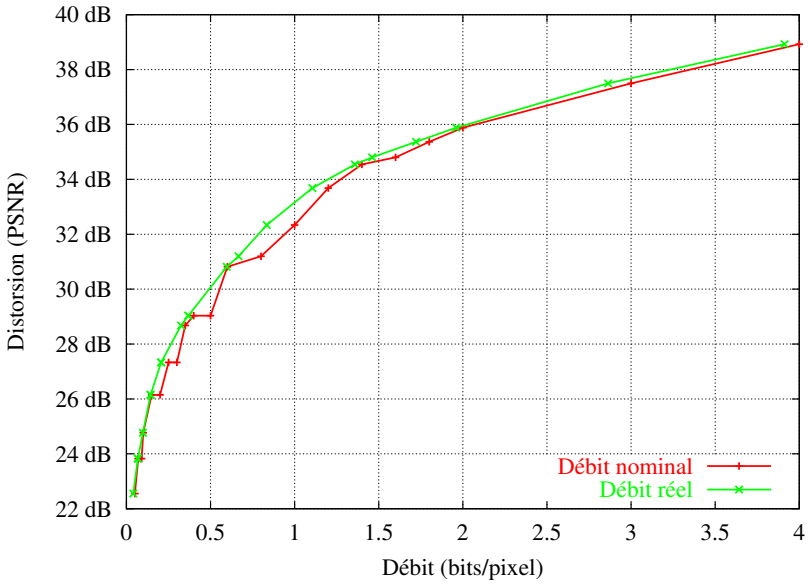
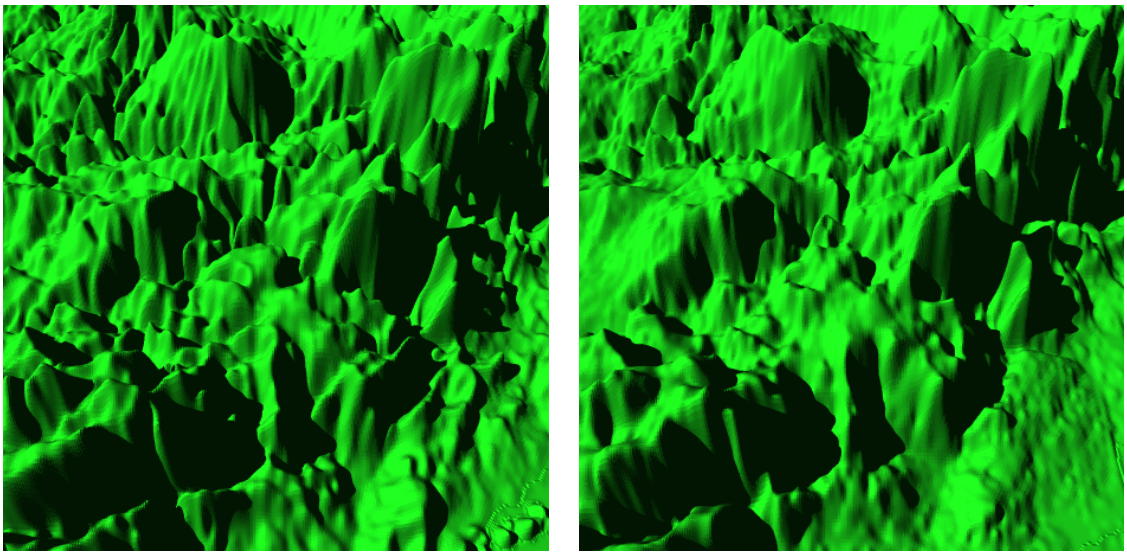


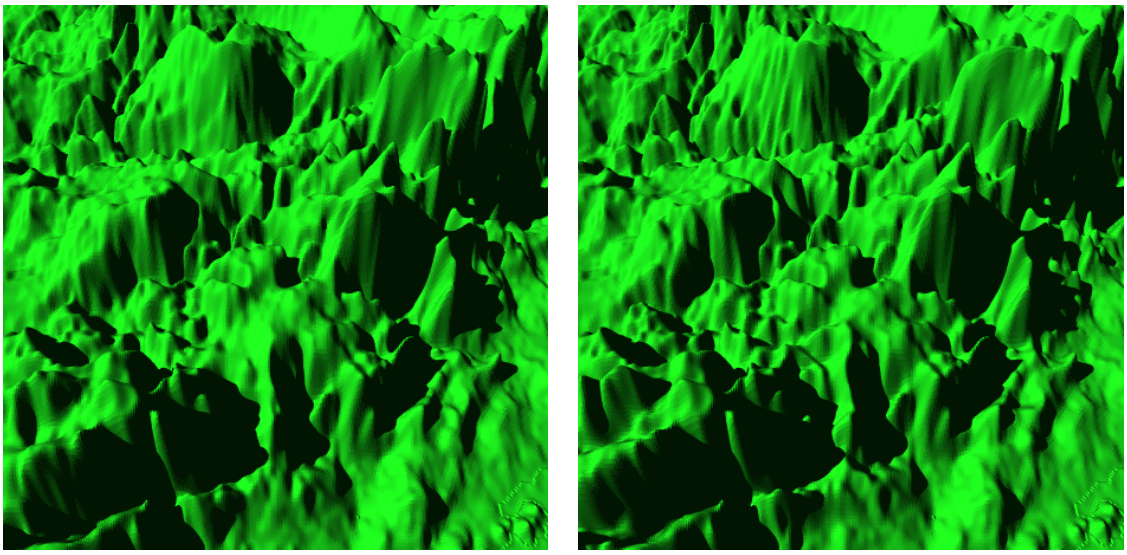
FIG. 84 – Région de Dijon – courbe débit/distorsion



(a) Surface originale

(b) Débit nominal $0,5$ bits/pixel

FIG. 85 – Gros plan sur une partie de la surface originale et compressée à $0,5$ bit/pixel



(a) Débit nominal 1, 2 bits/pixel (b) Débit nominal 2 bits/pixel

FIG. 86 – Gros plan sur une partie de la surface compressée à 1, 2 et 2 bits/pixel

Mars de Mario Polvere [Pol]. Pour la compression EZW, le logiciel *Kakadu 3.4* [kdu] a été utilisé, il est compatible avec la norme JPEG2000.

La figure 87 montre le premier exemple, il s'agit de l'image *baboon*. Les figures 88 à 92 donnent le résultat de la compression avec perte de cette image par quatre méthodes différentes : JPEG, compression fractale, EZW et notre méthode. Pour les figures 88 et 89, le débit est de 0, 46 bit/pixel, pour les figures 91 et 92, il est de 0, 19 bit/pixel. Enfin, la figure 94 donne un récapitulatif des courbes débit/distorsion pour chacune des méthodes de compression utilisées. Nous pouvons tirer de cette courbe trois résultats importants :

- Notre méthode est globalement moins bonne dans le sens où la distorsion, pour un même débit, est plus importante que pour les autres méthodes exposées. Cela se traduit par des ordonnées en moyenne plus basses.
- De très faibles débits peuvent cependant être obtenus grâce à notre méthode, cela se traduit par des abscisses faibles.
- Notre méthode est plus performante dans les faibles débits (inférieurs à 0, 4 bit/pixel) que la méthode classique de compression fractale.

De plus, nous pouvons tirer de l'observation visuelle des figures 88 à 92 que les effets de blocs ou artéfacs présents dans les autres méthodes sont beaucoup plus effacés dans la nôtre. Tandis que JPEG et la compression fractale font apparaître des blocs dans les bas débits (voir figure 91), notre méthode n'en génère pas ou peu. De plus la méthode EZW fait apparaître des artéfacs dès 0, 46 bit/pixels. Ce défaut est illustré sur la figure 90 sur laquelle on a zoomé les parties où ces erreurs sont visibles.

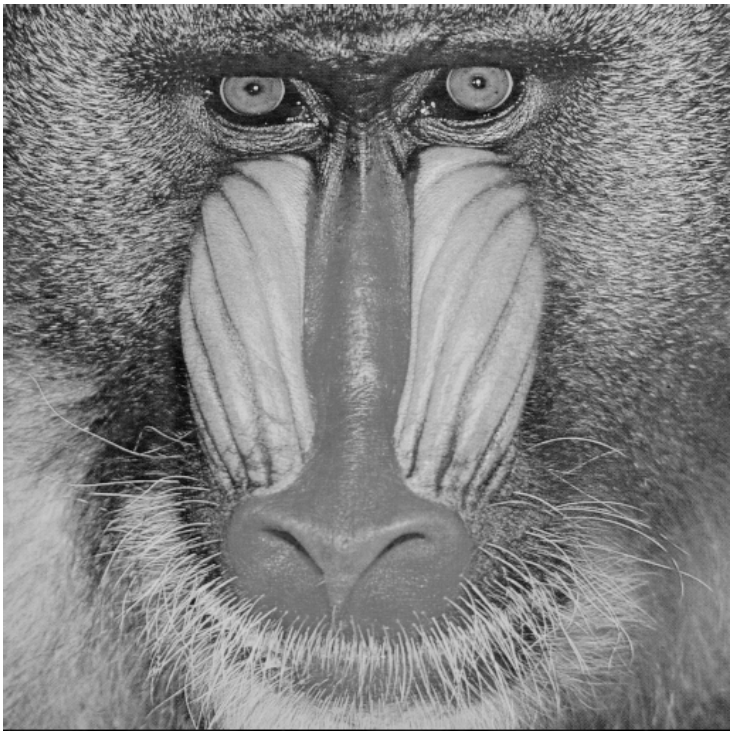
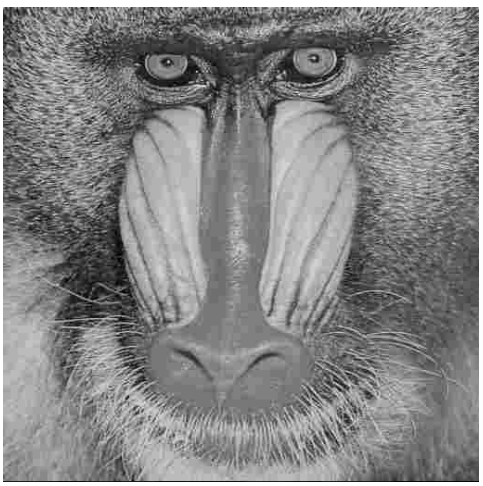
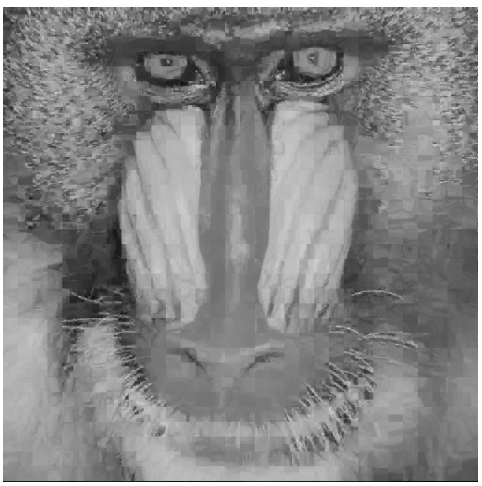


FIG. 87 – Image *baboon* originale 513×513

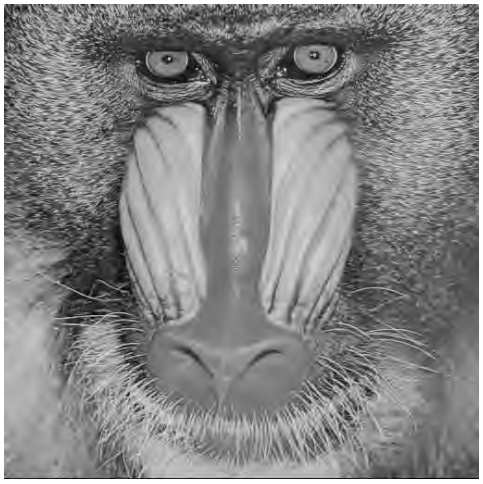


(a) JPEG – PSNR $= 23,9\,dB$

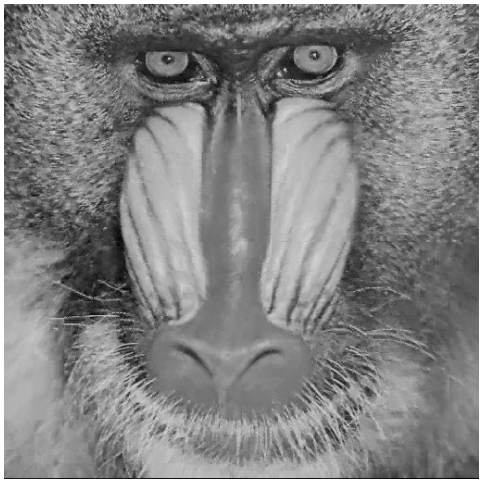


(b) Compression fractale – PSNR $= 21,6\,dB$

FIG. 88 – Compression de *baboon* à $0,46$ bit/pixel



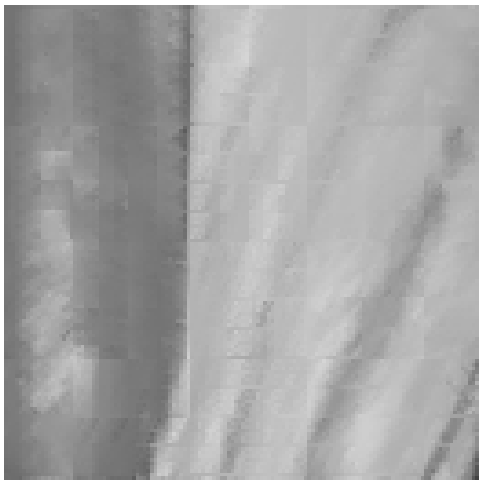
(a) EZW – PSNR $= 25,2\,dB$



(b) Notre méthode – PSNR $= 23,6\,dB$

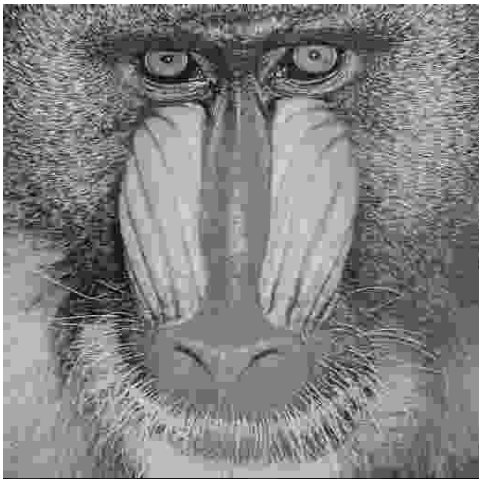
FIG. 89 – Compression de *baboon* à $0,46$ bit/pixel

(a) EZW

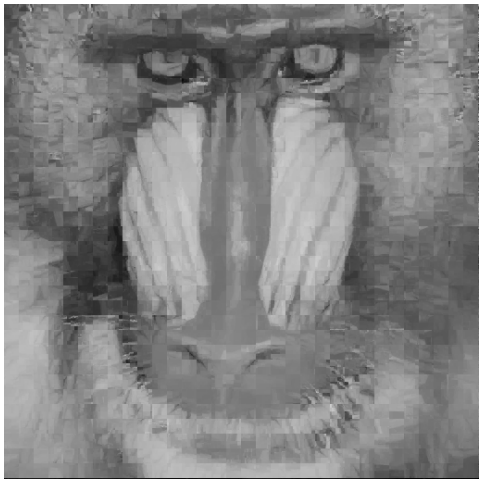


(b) Notre méthode

FIG. 90 – Compression de *baboon* à $0,46$ bit/pixel – zoom

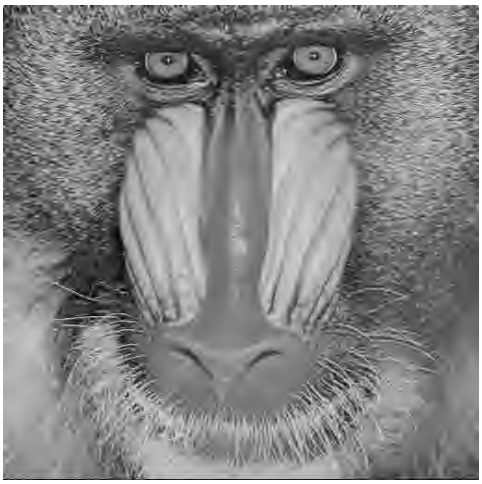


(a) JPEG – PSNR $= 21,6\,dB$

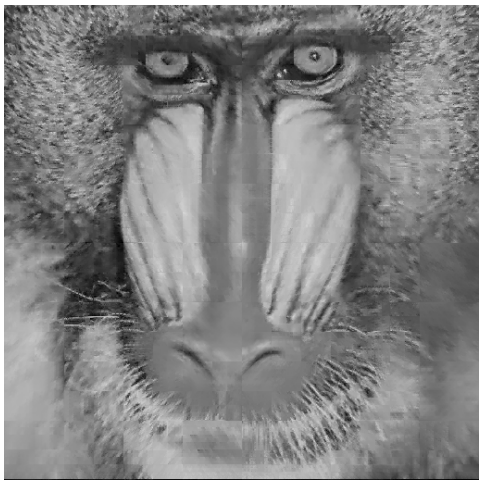


(b) Compression fractale – PSNR $= 20,5\,dB$

FIG. 91 – Compression de *baboon* à $0,19$ bit/pixel



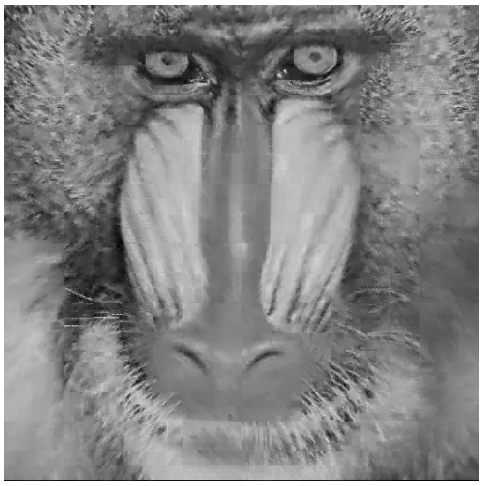
(a) EZW – PSNR $= 22,5\,dB$



(b) Notre méthode – PSNR $= 22,1\,dB$

FIG. 92 – Compression de *baboon* à $0,19$ bit/pixel

(a) EZW – PSNR $= 21,5\,dB$



(b) Notre méthode – PSNR $= 21,4\,dB$

FIG. 93 – Compression de *baboon* à $0,11$ bit/pixel

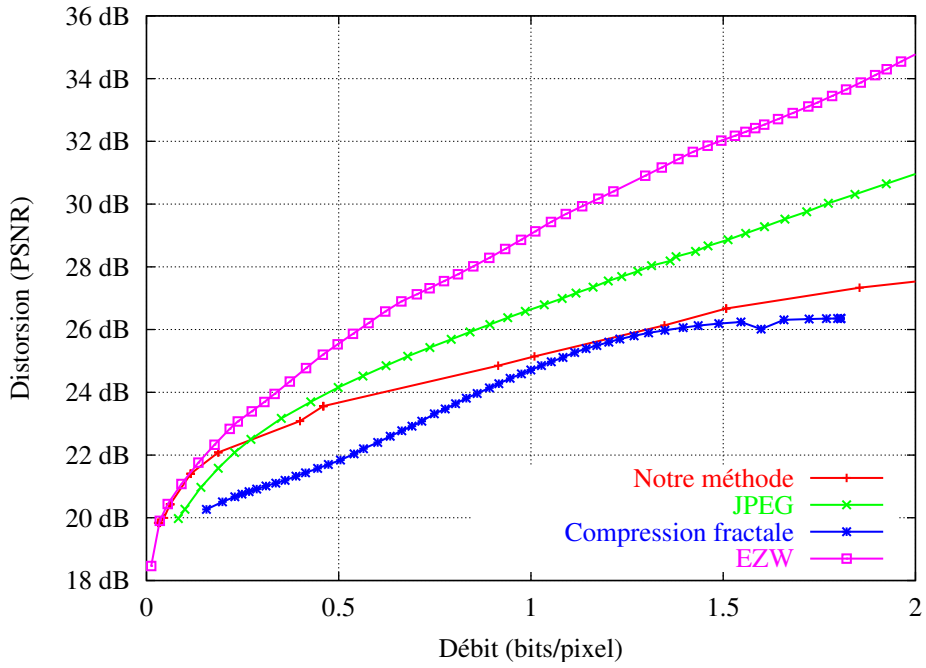


FIG. 94 – Courbes débit/distorsion pour l'image *baboon* et quatre méthodes de compression

La figure 95 montre le deuxième exemple que nous allons développer. Il s'agit d'une portion de taille 257×257 de l'image *airplane*. Comme pour l'exemple précédent, nous avons comparé notre méthode avec la compression JPEG, la compression fractale et EZW. Les figures 96, 97, 98 et 99 montrent le résultat de cette comparaison pour deux valeurs de débit : $0,5$ et $0,2$ bit/pixel. Enfin, la figure 100 propose les quatre courbes débit/distorsion associées aux différentes méthodes de compression. Ici nous pouvons encore constater que notre méthode permet d'aller très bas en débit (inférieur à $0,05$ bit/pixel) sans trop dégrader la qualité. La distorsion devient même plus faible qu'avec EZW. Pour illustrer ce constat, nous avons représenté les quatre courbes débit/distorsion pour la plage de débit allant jusqu'à $0,5$ bit/pixel dans la figure 101. Les méthodes JPEG et compression fractale sont limitées par un seuil minimal de débit : la compression fractale à $0,2$ bit/pixel, JPEG à $0,1$ bit/pixel. Notre méthode permet d'atteindre un seuil encore plus bas : $0,03$ bit/pixel. EZW quant à lui permet d'obtenir des seuils très bas, mais en dessous de $0,055$ bit/pixel (illustré par le point A sur la figure 101), il chute en qualité par rapport à notre approche qui reste stable en PSNR.

FIG. 95 – Image *airplane* originale 257×257

(a) JPEG – PSNR $= 34,3\,dB$

(b) Compression fractale – PSNR $= 31,2\,dB$

FIG. 96 – Compression de *airplane* à $0,5$ bit/pixel

(a) EZW – PSNR $= 37,1\,dB$

(b) Notre méthode – PSNR $= 32,0\,dB$

FIG. 97 – Compression de *airplane* à $0,5$ bit/pixel

(a) JPEG – PSNR $= 28,3\,dB$

(b) Compression fractale – PSNR $= 23,3,\ dB$

FIG. 98 – Compression de *airplane* à $0,2$ bit/pixel

(a) EZW – PSNR $= 30,9\,dB$

(b) Notre méthode – PSNR $= 27,2\,dB$

FIG. 99 – Compression de *airplane* à $0,2$ bit/pixel

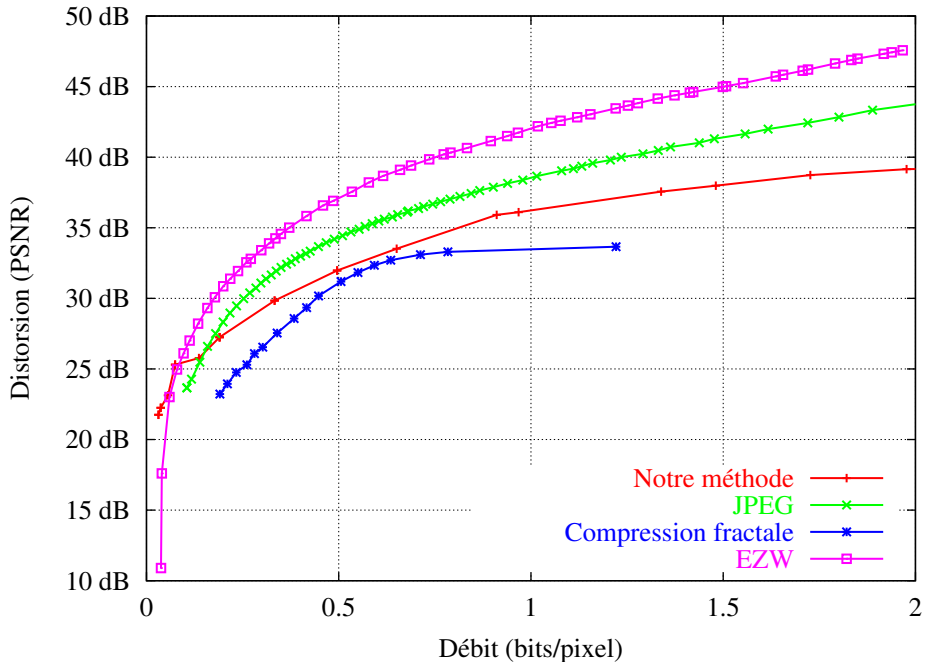


FIG. 100 – Courbes débit/distorsion pour l'image *airplane* et quatre méthodes de compression

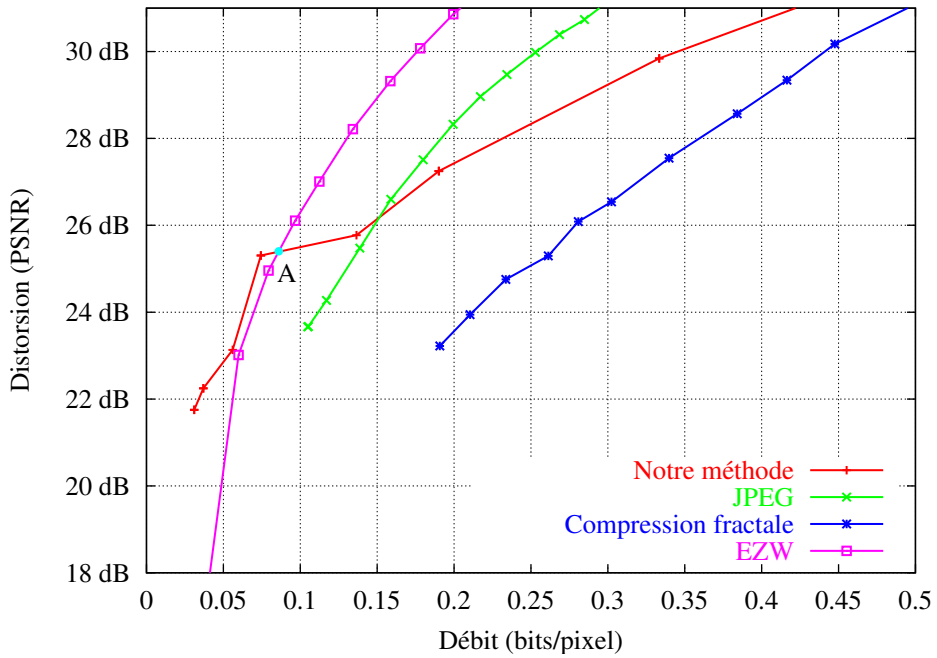


FIG. 101 – Courbes débit/distorsion pour l'image *airplane* et quatre méthodes de compression – zoom sur le bas débit

Le dernier exemple que nous allons développer concerne l'image de la figure 102, une portion de taille 257×257 de l'image *peppers*. Les figures 103, 104 et 105 représentent la comparaison des différentes méthodes de compression pour des débits de $0,44$ et $0,12$ bit/pixel respectivement. Il est à noter que le débit de $0,12$ n'a pas pu être atteint avec la méthode de compression fractale. Pour finir, la figure 107 donne les courbes de débit/distorsion pour chacune des méthodes. Cette fois-ci, nous pouvons observer que la courbe de débit/distorsion de JPEG passe en dessous de celle de notre méthode pour les faibles débits. De plus, l'effet de bloc pour JPEG est très nettement visible dans la figure 105, alors qu'il est peu présent pour notre méthode.

FIG. 102 – Image *peppers* originale 257×257

(a) JPEG – PSNR $= 33,0\,dB$

(b) Compression fractale – PSNR $= 31,4\,dB$

FIG. 103 – Compression de *peppers* à $0,44$ bit/pixel

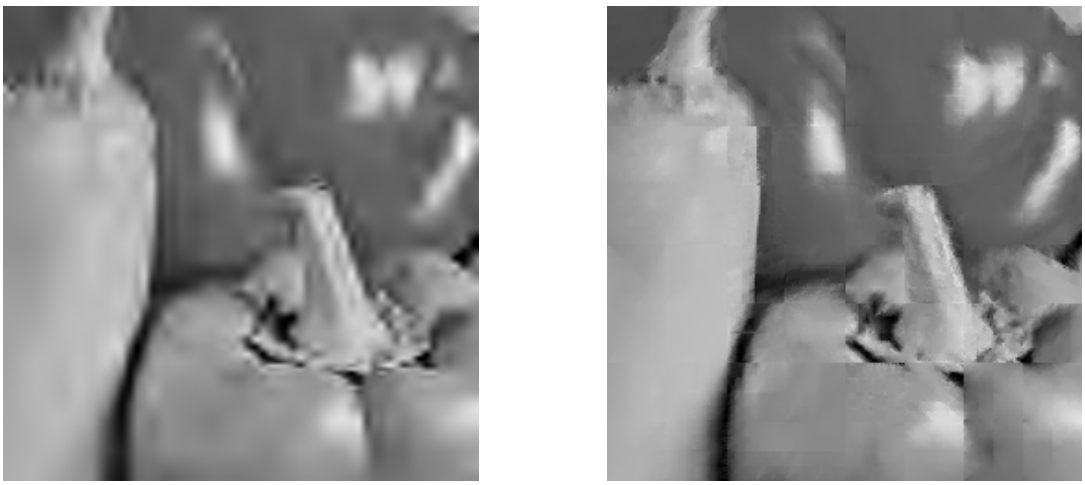
(a) EZW – PSNR $= 35,1\,dB$

(b) Notre méthode – PSNR $= 32,8\,dB$

FIG. 104 – Compression de *peppers* à $0,44$ bit/pixel

(a) JPEG – PSNR = $25,0\,dB$ (b) EZW – PSNR = $29,5\,dB$ (c) Notre méthode – PSNR = $28,5\,dB$

FIG. 105 – Compression de *peppers* à $0,12$ bit/pixel



(a) EZW – PSNR = $27,6\,dB$ (b) Notre méthode – PSNR = $28,3\,dB$

FIG. 106 – Compression de *peppers* à $0,085$ bit/pixel

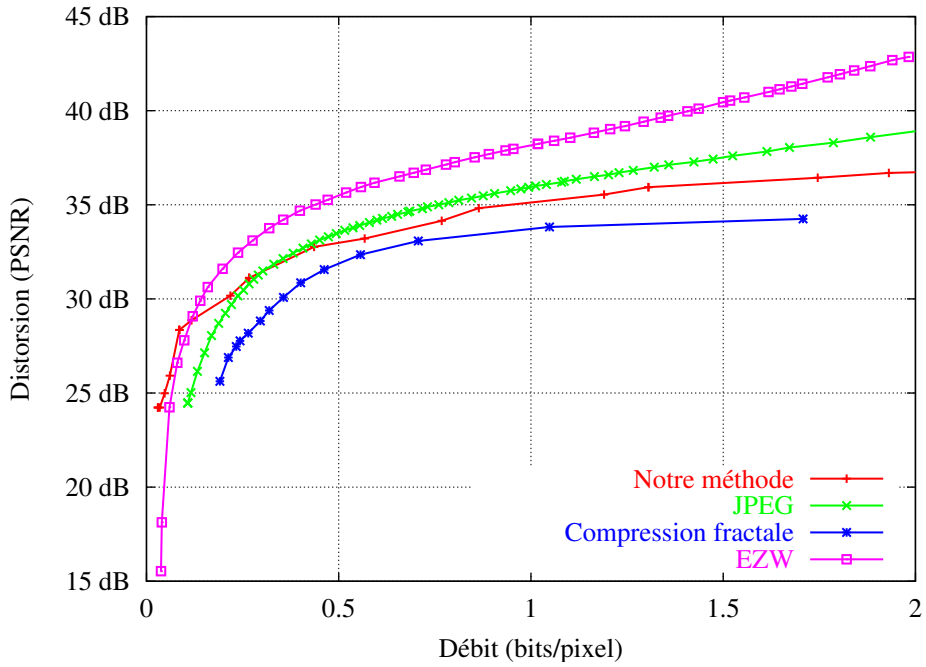


FIG. 107 – Courbes débit/distorsion pour l'image *peppers* et quatre méthodes de compression

Chapitre 4

Conclusion

L'approximation de surfaces ou d'images complexes, grâce au modèle d'attracteurs d'IFS projetés telle qu'elle a été présentée dans la partie précédente est difficile, voire impossible. Nous avons essayé de combler ce vide grâce à l'introduction d'une structure de quadtree qui permet d'adapter les modèles aux caractéristiques locales des données d'origine. Chaque feuille du quadtree est un modèle d'attracteur d'IFS projeté indépendant – outre les conditions de raccord. De cette structure, nous avons tiré deux algorithmes permettant, à partir d'une surface ou image donnée, de construire un quadtree adapté à une contrainte. Cette contrainte peut être de deux types :

- Une distorsion globale à ne pas dépasser.
  Dans ce cas là, nous avons élaboré un algorithme permettant de répartir de manière uniforme la distorsion. Les résultats montrent que l'on peut alors arriver à modéliser des terrains ou des images en niveaux de gris de façon très précise.
- Un débit nominal à ne pas dépasser.
  Pour cette contrainte, nous avons construit un algorithme qui permet un codage optimisé de la surface ou de l'image. Une recherche exhaustive de tous les modèles, à toutes les échelles spatiales, avec diverses quantifications des paramètres, est effectuée. Une dernière phase permet de choisir quelle représentation est optimale avec le critère de débit que l'on s'est fixé. Les résultats montrent que la technique de compression ainsi construite donne des résultats corrects par rapport à d'autres méthodes de compression, et de meilleurs résultats pour ce qui est des faibles débits grâce à un en-tête fixe de très faible taille.

Il reste néanmoins que ces deux algorithmes sont prohibitifs en terme de temps de calcul. Malgré le fait qu'une seule éxécution permet le codage à tous les débits désirés, cette éxécution demeure très longue (50 minutes pour une image 513×513 et 20 minutes pour une image 257×257).

Conclusion

Nous avons proposé une méthode générique pour approximer des formes accidentées. Ces formes ont la particularité d'avoir une seule propriété géométrique garantie : la continuité. Il s'agit de courbes, de surfaces de profondeur et d'images en niveaux de gris. Le modèle que nous employons pour décrire ces formes porte le nom d'IFS projeté, il a été développé au sein du laboratoire du LIGIM lors de la thèse de Chems Eddine Zair[Zai98]. Il combine l'avantage de la simplicité d'utilisation des IFS pour décrire les objets fractals, et la maniabilité des formes à pôles.

Mais ceci concerne uniquement le domaine de la synthèse et de la modélisation géométrique. Il existe un domaine complémentaire à cette synthèse : l'analyse. Étant donnée une forme réelle, quel modèle décrit le mieux celle-ci parmi une classe de modèles : c'est le problème inverse ou l'approximation. L'approche originale que nous avons mise en œuvre consiste à transcrire le problème d'approximation en problème de régression non-linéaire. En choisissant de bons critères d'approximation, adaptés aux courbes et aux surfaces, nous avons obtenu des propriétés de différentiabilités pour la fonction à optimiser. Ainsi, nous avons pu mettre en œuvre une méthode de résolution fondée sur l'algorithme de Levenberg-Marquardt. En effet, celui-ci utilise la propriété de différentiabilité du modèle par rapport aux paramètres qui le décrivent. Ce modèle a permis d'approximer de façon précise des courbes de $\mathbb{R}^2$, lisses comme accidentées. Par contre, il s'avère que l'approximation de surfaces ou d'images reste limitée à des données de petite taille et la précision obtenue n'est pas satisfaisante.

Nous avons donc imaginé une extension du modèle surfacique en l'englobant dans une structure de type quadtree. Cette extension permet de modéliser des surfaces et des images dont les propriétés varient spatialement. Nous avons dégagé de ce nouveau modèle deux algorithmes permettant de générer le quadtree d'IFS projetés approximant une surface ou une image en niveaux de gris. Le premier algorithme répartit uniformément une distorsion donnée comme paramètre. Cet algorithme a l'avantage d'être simple mais n'effectue aucune optimisation de la complexité du modèle généré. Le second algorithme permet de faire une optimisation du ratio débit/distorsion, mettant en œuvre une chaîne complète de codage. Celui-ci permet donc d'effectuer de la compression de surfaces et d'images en niveau de gris.

Nous avons comparé les résultats de compression d'images avec d'autres méthodes de compression classiques : JPEG, EZW et la compression fractale de type Jacquin. Les résultats montrent que notre modèle est performant dans le domaine de la compression bas débit.

Ces premiers résultats de compression d'images semblent prometteurs et ouvrent un

nouvel horizon aux méthodes de compression fondées sur les fractales. De futurs travaux pourraient à court terme améliorer les points suivants :

- **Extension aux contours.** En principe, appliquer l'approximation à des contours ne devrait pas poser de problème. On pourra, comme pour les surfaces et images, utiliser plusieurs IFS projetés pour coder un seul contour, dans une structure de type arbre binaire ou tout simplement liste.
- **Diminuer le temps de calcul.** Celui-ci reste très élevé : il est supérieur à celui des méthodes de compression fractale de type Jacquin qui pourtant sont réputées pour être gourmandes.
- **Tirer partie des conditions de raccord.** En effet, dans les deux méthodes qui ont été développées, les carreaux sont indépendants et sont codés de manière indépendante. Pourtant, ceux-ci ont des bords communs et cela engendre des contraintes de raccord. Il pourrait donc être intéressant d'essayer de supprimer l'information redondante.
- **Adapter à la transmission progressive d'information.** La méthode de compression étant performante dans le bas débit, il paraît logique de l'utiliser pour de la transmission progressive d'information. On peut imaginer plusieurs stratégies combinables : transmission des bits de poids fort d'abord puis ceux de poids faible, rafinement progressif de la structure du quadtree,...

Au delà de ces améliorations, on peut imaginer à plus long terme une analyse plus profonde du problème :

- **Construction d'un dictionnaire de fonctions fractales.** Ceci nécessiterait une analyse poussée sur de nombreuses images, pour essayer d'en tirer une information pertinente sur les matrices de subdivision. On pourrait en déduire un jeu de matrices standard qui seraient combinées entre elles pour former la matrice qui colle le mieux aux données. L'avantage serait bien entendu de gagner de la place pour le codage, mais aussi d'obtenir une méthode de compression plus rapide.
- **Construction d'une base de fonctions.** Il s'agit de pousser l'idée précédente un peu plus loin pour essayer de parvenir à construire une base de fonctions fractales, ayant de bonnes propriétés (orthogonalité entre autres). Évidemment, ce résultat serait idéal, mais reste à nos yeux très difficile à obtenir.

Nous noterons simplement que notre méthode est assez facilement adaptable à la recherche d'une combinaison de vecteurs de paramètres plutôt qu'à la recherche d'un vecteur de paramètre précis.

On peut aussi imaginer que l'approximation de formes servent à des domaines connexes comme l'indexation, la comparaison, la recherche de formes dans une base de données. En effet, la compacité des descripteurs rend possible ce genre d'applications. Toutefois, on veillera à comparer des choses comparables. Les transformations d'un IFS projeté ne sont pas comparables directement. Deux solutions s'offrent à nous pour palier ce problème :

- **Trouver une représentation canonique.** Cette opération semble délicate mais une fois effectuée, permet de comparer directement les matrices de subdivision avec une distance par exemple.
- **Extraire de l'information des matrices de subdivision.** On peut imaginer que les principales propriétés de la forme sont descriptibles avec peu de paramètres. Par exemple, la valeur propre la plus élevée et son vecteur associé nous donne une

information sur la direction de la tangente aux extrémités (ou son absence dans certains cas). Les autres valeurs propres vont déterminer le type de courbe ou de surface.

Une première comparaison pourrait être faite avec les points de contrôle pour éliminer des formes qui ne peuvent pas se ressembler et ensuite une analyse plus poussée des propriétés des matrices de subdivision.

Par l'intermédiaire de cette thèse, un pas a été franchi dans le domaine de l'approximation de formes accidentées, il semble même que c'est la première fois que des méthodes numériques soient utilisées pour faire de l'approximation fractale. Il en ressort que les IFS projetés sont un bon modèle pour la description de formes accidentées. Associés à notre méthode d'approximation et à une optimisation du codage, ils permettent d'obtenir un descripteur compact. Ceci permet à notre travail d'être utilisé pour la compression, et notamment la compression d'images à très bas débits, avec des possibilités futures d'utilisation dans des outils d'indexation, de comparaison et de recherche de formes ou images.

Annexe A

Raccord de carreaux de quadtree

A.1 Préliminaires

Le quadtree classique correspond aux subdivisions du pavé $[0,1]^2$ par l'IFS $\{\mathrm{T}_0, \mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3\}$. Au niveau n, $[0,1]^2$ est subdivisé en 2^{2n} pavés $\mathrm{T}_\gamma[0,1]^2$ en posant $\mathrm{T}_\gamma = \mathrm{T}_{\gamma_1} \dots \mathrm{T}_{\gamma_n}$.

La fonction d'adressage associée $\phi^\square$ correspond au code de Peano :

$$\forall (s,t) \in [0,1]^2 \ (s,t) = \phi^\square(\rho)$$

avec $\rho \in \Sigma^\omega$, $\rho_i = \sigma_i \natural \tau_i$, σ et τ les développements en base 2 de s et t.

Aux quatre côtés sont associés des fonctions d'adressage extraites :

$$\begin{array}{rcl} (s,0) & = & \phi^\square_{\bar{u}}(\sigma) = \phi^\square \circ \xi_{\bar{u}}(\sigma) \\ (s,1) & = & \phi^\square_{u}(\sigma) = \phi^\square \circ \xi_{u}(\sigma) \\ (0,t) & = & \phi^\square_{\bar{v}}(\sigma) = \phi^\square \circ \xi_{\bar{v}}(\sigma) \\ (1,t) & = & \phi^\square_{v}(\sigma) = \phi^\square \circ \xi_{v}(\sigma) \end{array}$$

Les raccords entre pavés de la subdivision se font sur des arcs paramétrés de la forme :

- $\mathrm{T}_\gamma \phi^\square_{\bar{u}} = \mathrm{T}_\alpha \phi^\square_u$ pour les raccords horizontaux
- $\mathrm{T}_\gamma \phi^\square_{\bar{v}} = \mathrm{T}_\alpha \phi^\square_v$ pour les raccords verticaux

La structure de raccord peut être caractérisée de manière formelle à l'aide de deux relations sur Σ^* :

- $(\gamma, \alpha) \in \overset{u\bar{u}}{\Upsilon} \Leftrightarrow \mathrm{T}_\gamma \phi^\square_{\bar{u}} = \mathrm{T}_\alpha \phi^\square_u$
- $(\gamma, \alpha) \in \overset{v\bar{v}}{\Upsilon} \Leftrightarrow \mathrm{T}_\gamma \phi^\square_{\bar{v}} = \mathrm{T}_\alpha \phi^\square_v$

Chaque couple (γ, α) est en bijection avec une arête de raccord (c'est à dire hors de la bordure de $[0,1]^2$). Tout point (s,t) commun à deux pavés $\mathrm{T}_\gamma[0,1]^2$ et $\mathrm{T}_\alpha[0,1]^2$ est sur une arête de raccord, celle-ci correspondant au côté le plus petit.

Autrement dit, on a : si $(s,t) \in \mathrm{T}_\gamma[0,1]^2 \cap \mathrm{T}_\alpha[0,1]^2$ et si $|\gamma| \geq |\alpha|$ alors :

$\exists \beta$, $|\beta| = |\gamma| - |\alpha|$ tel que :

- soit $(\gamma, \alpha\beta) \in \overset{u\bar{u}}{\Upsilon}$, $\exists \sigma \in \{0, \dots, N-1\}^\omega$, $(s,t) = T_\sigma \phi^\square_{\bar{u}}(\sigma) = T_\alpha T_\beta \phi^\square_u(\sigma)$.
- soit même chose avec $(\gamma, \alpha\beta) \in \overset{v\bar{v}}{\Upsilon}$

Les relations de raccords formelles $\overset{u\bar{u}}{\Upsilon}$ et $\overset{v\bar{v}}{\Upsilon}$ définissent une relation d'adjacence formelle : γ et α seront dits adjacents si :

$$\begin{cases} |\gamma| \geq |\alpha|, & \exists\beta,\ |\beta| = |\gamma| - |\alpha| \quad (\gamma, \alpha\beta) \in \overset{u\bar{u}}{\Upsilon} \text{ ou } (\gamma, \alpha\beta) \in \overset{v\bar{v}}{\Upsilon} \\ |\gamma| < |\alpha|, & \exists\delta,\ |\delta| = |\alpha| - |\gamma| \quad (\gamma\delta, \alpha) \in \overset{u\bar{u}}{\Upsilon} \text{ ou } (\gamma\delta, \alpha) \in \overset{v\bar{v}}{\Upsilon} \end{cases}$$

Cette relation s'étend aux IFS projetés organisés en quadtree $(P^\gamma, \mathbb{T}^\gamma)$, $\gamma \in \Sigma^*$ et permet d'exprimer les conditions de raccord d'un carreau construit sur une coupure Γ.

A.2 Raccord

Muni de cette propriété, il est possible de définir des carreaux, c'est à dire des fonctions continues F qui sont des combinaisons de carreaux définis par IFS projeté.

Des hypothèses sur les $(P^\gamma, \mathbb{T}^\gamma)$ sont nécessaires pour garantir le raccord des carreaux $P^\gamma\,\Phi^\gamma$ entre eux. Étant donné $((P_\gamma, \mathbb{T}^\gamma))_{\gamma\in\Gamma}$ une famille d'IFS projetés avec Γ coupure du quadtree, on dira que cette famille vérifie la propriété d'adjacence si on a :

- En prenant $|\gamma| \geq |\alpha|$:
  - Soit un raccord horizontal :

$$(\gamma, \alpha\beta) \in \overset{u\bar{u}}{\Upsilon} \Rightarrow \begin{cases} P^\gamma \amalg_{\bar{u}} = P^\alpha \mathbb{T}^\alpha_\beta \amalg_u \\ \tilde{T}^\gamma_{\bar{u}} = \tilde{T}^\gamma_u \end{cases}$$

  - Soit un raccord vertical :

$$(\gamma, \alpha\beta) \in \overset{v\bar{v}}{\Upsilon} \Rightarrow \begin{cases} P^\gamma \amalg_{\bar{v}} = P^\alpha \mathbb{T}^\alpha_\beta \amalg_v \\ \tilde{T}^\gamma_{\bar{v}} = \tilde{T}^\gamma_v \end{cases}$$

- Le même type de relations avec $|\gamma| < |\alpha|$.

Proposition 12 (adjacence d'IFS ⇒ adjacence des fonctions d'adressage)

$$(\gamma, \alpha\beta) \in \overset{u\bar{u}}{\Upsilon} \Rightarrow P^\gamma \phi^\gamma_{\bar{u}} = P^\alpha T^\alpha_\beta \phi^\alpha_u$$

On a des propriétés analogues dans les trois autres cas d'adjacence.

Index

Index des mots utilisés.

Bibliographie

[Ala] Elevation contours (100-foot interval) - kbec database. – Alaska Department of Fish and Game. – *http ://www.state.ak.us/adfg/habitat/geninfo/nerr/kbec/physdata/contour.html*.

[Bar88] Barnsley (Michael). – *Fractals everywhere*. – Academic Press, 1988.

[Bar93] Barnsley (Michael). – *Fractals everywhere (second edition)*. – Academic Press Professional, 1993.

[BBBCP98] Belloulata (K.), Baskurt (A.), Benoit-Cattin (H.) et Prost (R.). – Fractal coding of subbands with an oriented partition. *Signal Processing : Image Communication*, vol. 12, Juin 1998, pp. 243–252.

[Bel98] Belloulata (Kamel). – *Compression d'images par fractales : études sur la mesure et le domaine de recherche (spatial ou transformé) de l'auto-similarité et sur l'accélération de la génération du modèle fractal*. – Thèse de doctorat, Institut National des Sciences Appliquées, Février 1998.

[Ber97] Berkner (K). – A wavelet-based solution to the inverse problem for fractal interpolation functions. *In : Fractals in engineering'97*, éd. par Levy Vehel, Lutton (Tricot), pp. 81–92. – Springer Verlag, 1997.

[BL02] Barlaud (Michel) et Labit (Claude). – *Compression codage des images et des vidéos*. – Paris, Hermès, Janvier 2002.

[BT97a] Blanc-Talon (Jacques). – Controlled Approximation with Recursively Defined Curves. *In : Fractals in Engineeringi, Arcachon, France*, éd. par INRIA.

[BT97b] Blanc-Talon (Jacques). – Self-controlled fractal splines for terrain reconstruction. *In : IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics*. – Berlin, 1997.

[BV94] Barthel (K. U.) et Voyé (T.). – Adaptive fractal image coding in the frequency domain. *In : Proceedings of the International Workshop on Image Processing*, pp. 33–38. – Budapest, Hungary, Juin 1994.

[Béz70] Bézier (Pierre). – *Emploi des machines à commandes numériques*. – Paris, Masson et Cie, 1970.

[Béz86] Bézier (Pierre). – *Courbes et Surfaces*. – Paris, Hermès, 1986.

[CJB01] Chevalier (Laurent), Jaillet (Fabrice) et Baskurt (Atilla). – 3d shape coding with superquadrics. *In : ICIP'2001 Proceedings*.

[DACB96] Davoine (F.), Antonini (M.), Chassery (J.-M.) et Barlaud (M.). – Fractal image compression based on delaunay triangulation and vector quantization. *IEEE Transactions on Image Processing*, vol. 5, N° 2, Février 1996, pp. 338–346.

[dC86] de Casteljau (Paul). – *Formes à pôles – Mathématiques et CAO 2.* – Paris, Hermès, 1986.

[DC94] Davoine (Franck) et Chassery (Jean-Marc). – Adaptive Delaunay triangulation for attractor image coding. *In : Proceedings of the 12th International Conference on Pattern Recognition*, pp. 801–803. – Jerusalem, Israel, 1994.

[DEM] Free gis data - gis data depot - usgs dem resources. – Geo Community. – *http ://data.geocomm.com/dem/demdownload.html.*

[DV95] Daoudi (Khalid) et Véhel (Jacques Lévy). – Speech signal modeling based on local regularity analysis. *In : IASTED/IEEE International Conference on Signal and Image Processing (SIP'95), Las Vegas, November 20-23.*

[Edg90] Edgar (Gerald A.). – *Measure, Topology, and Fractal Geometry.* – Springer Verlag, 1990.

[Far92] Farin (Gerald). – *Courbes et Surfaces pour la CGAO.* – Masson, 1992.

[FB88] Forsey (David R.) et Bartels (Richard H.). – Hierarchical B-Spline Refinemant. *Computers Graphics*, vol. 22, N° 4, Août 1988.

[Fis92] Fisher (Yuval). – A discussion of fractal image compression. *In : Chaos and fractals : New Frontiers of Science*, éd. par H O Peitgen (H Jurgens) et Saupe (D). – Springer Verlag, 1992.

[Fra] France, digital terrain elevation data (dted) level 0 - free gis data - gis data depot. – Geo Community. – *http ://www.gisdatadepot.com/catalog/FR/group121.html.*

[GB93] Gupta (A) et Bajcsy (R). – Volumetric segmentation of range images of 3D objects using superquadrics models. *CVGIP : Image understanding*, vol. 58, N° 3, Novembre 1993, pp. 302–326.

[GMA94] Goertzel (Ben), Miyamoto (Hiroo) et Awata (Yoshimasa). – Fractal image compression with the genetic algorithm. *Complexity International*, vol. 1, 1994.

[GTB99] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – Description et reconstruction de courbes par une approche fractale. *In : AFIG'99*, pp. 146–155. – Reims, Novembre 1999.

[GTB00a] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – Approximation de courbes par une méthode fractale. *In : CORESA 2000.* – Poitiers, Octobre 2000.

[GTB00b] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – Fractal coding of shapes based on a projected IFS model. *In : ICIP 2000*, pp. 203–206.

[GTB01a] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – Approximation de courbes et surfaces par un modèle fractal. *In : Journée Coopération Analyse d'Image et Modélisation.* – Lyon, Juin 2001.

[GTB01b] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – Approximation de courbes et surfaces par une méthode fractale. *In : CORESA 2001.* – Dijon, Novembre 2001.

[GTB01c] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – A fractal approximation of curves. *Fractals*, vol. 9, N° 1, Mars 2001, pp. 95–103.

[GTB01d] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – Fractal Approximation of Surfaces based on projected IFS attractors. *In : Proceedings of EUROGRAPHICS'2001, short presentations.*

[GTB02] Guérin (Eric), Tosan (Eric) et Baskurt (Atilla). – Modeling and approximation of fractal surfaces with projected IFS attractors, 2002.

[Gué99] Guérin (Eric). – Description et reconstruction de courbes par une approche fractale, Juillet 1999. Université Claude Bernard Lyon 1, Rapport de DEA d'Informatique de Lyon.

[Hut81] Hutchinson. – Fractals and self-similarity. *Indiana University Journal of Mathematics*, vol. 30, 1981, pp. 713–747.

[iBL +94] Øien (G. E.), Baharav (Z.), Lepsøy (S.), Karnin (E.) et Malah (D.). – A new improved collage theorem with applications to multiresolution fractal image coding. *In : Proceedings ICASSP-94 (IEEE International Conference on Acoustics, Speech and Signal Processing)*, pp. 565–568. – Adelaide, Australia, Avril 1994.

[Jac92] Jacquin (A E). – Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Trans. on Image Processing*, vol. 1, Janvier 1992, pp. 18–30.

[JM96] Jaffard (S.) et Meyer (Y.). – Wavelet methods for pointwise regularity and local oscillations of functions. *Mem. Amer. Math. Soc.*, vol. 123, N° 587, 1996.

[kdu] Kakadu software front page. – Kakadu Software. – *http ://www.kakadusoftware.com/.*

[Llo82] Lloyd (Stuart L.). – Least square quantization in pcm. *IEEE Transactions on Information Theory*, vol. IT-28, N° 2, Mars 1982.

[LLVC +95] Lutton (Evelyne), Levy-Vehel (Jacques), Cretin (Guillaume), Glevarec (Philippe) et Roll (Cédric). – *Mixed IFS : resolution of the inverse problem using Genetic Programming.* – Rapport de recherche N° 2631, Institut National de Recherche en Informatique et en Automatique, Août 1995. Programme 5.

[LY94] Lu (G.) et Yew (T.-L.). – Image compression using quadtree partitioned iterated function systems. *Electronic Letters*, vol. 30, N° 1, 1994, pp. 23–24.

[Mal99] Mallat (Stéphane). – *A Wavelet Tour of Signal Processing.* – Academic Press, 1999. 2nd edition.

[Man82] Mandelbrot (Benoît). – *The fractal geometry of nature.* – San Fransisco, Freeman, 1982.

[Mas94] Massopust (P.). – *Fractal Functions, Fractal Surfaces and Wavelets.* – Academic Press, 1994.

[Max60] Max (Joel). – Quantizing for minimum distortion. *IRE Trans. Inform. Theory*, vol. IT-6, Mars 1960, pp. 7–12.

[MBA94] Muzy (J. F.), Bacry (E.) et Arneodo (A.). – The multifractal formalism revisited with wavelets. *International Journal of Bifurcation and Chaos*, vol. 4, N° 2, 1994, pp. 245–302.

[MP87] Micchelli (C A) et Prautzsch (H). – Computing surfaces invariant under subdivision. *Computer Aided Geometric Design*, N° 4, 1987, pp. 321–328.

[PFTV93a] Press (W. H.), Flannery (B. P.), Teukolsky (S. A.) et Vetterling (W. T.). – *Numerical Recipes in C : The Art of Scientific Computing*, chap. Downhill Simplex Method in Multidimensions. – Cambridge University Press, 1993.

[PFTV93b] Press (W. H.), Flannery (B. P.), Teukolsky (S. A.) et Vetterling (W. T.). – *Numerical Recipes in C : The Art of Scientific Computing*, chap. Nonlinear Models. – Cambridge University Press, 1993.

[PFTV93c] Press (W. H.), Flannery (B. P.), Teukolsky (S. A.) et Vetterling (W. T.). – *Numerical Recipes in C : The Art of Scientific Computing*, chap. Arithmetic Coding. – Cambridge University Press, 1993.

[Pol] Polvere (Mario), Mars – version 1.0. – The Institute for Nonlinear Science. – *http ://inls.ucsd.edu/y/Fractals/Mars-1.0.tar.gz*.

[Pov] Povray - the persistance of vision raytracer. – povray.org. – *http ://www.povray.org*.

[RH97] Ruhl (Matthias) et Hartenstein (Hannes). – Optimal fractal coding is NP-hard. *In : Proceedings DCC'97 (IEEE Data Compression Conference)*, pp. 261–270. – Snowbird, UT, USA, 1997.

[RV93] Ramchandran (Kannan) et Vetterli (Martin). – Best wavelet packet bases in rate-distortion sense. *IEEE Transactions on Image Processing*, vol. 2, N° 2, Avril 1993.

[SDG95] Struzik (Z R), Dooijes (E H) et Groen (F C A). – The solution of the inverse fractal problem with the help of wavelet decomposition. *In : Fractals reviews in the natural and applied sciences*, éd. par Novak (M M), pp. 332–343. – Chapman and Hall, Février 1995.

[Sha93] Shapiro (J. M.). – Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, vol. 41, N° 12, 1993, pp. 3445–3462.

[SHH96] Saupe (D.), Hamzaoui (R.) et Hartenstein (H.). – Fractal image compression : an introductory overview, 1996.

[TGB02] Tosan (Eric), Guérin (Eric) et Baskurt (Attila). – Modélisation et reconstruction de surfaces fractales basées sur les ifs. *In : Colloque Autosimilarité et Applications, Clermont-Ferrand*.

[TM91] Terzopoulos (D) et Metaxas (D). – Dynamic 3D models with local and global deformations : deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, N° 7, Juillet 1991, pp. 703–714.

[Tos99] Tosan (Eric). – Surfaces fractales définies par leurs bords. *In : Journées "Courbes, surfaces et algorithmes", Grenoble*, éd. par Briard (L.), Szafran (N.) et B.Lacolle.

[TrGB02] Tosan (Eric), Éric Guérin et Baskurt (Attila). – Design and reconstruction of fractal surfaces. *In : 6th International Conference on Information Visualisation IV 2002, London, UK*, éd. par Society (IEEE Computer), pp. 311–316.

[TZTV97] THOLLOT (Joëlle), ZAIR (Chems Eddine), TOSAN (Eric) et VANDORPE (Denis). – Modeling fractal shapes using generalizations of IFS techniques. *In : 3^{rd} Conference Fractals in Engineering, Arcachon, France*, éd. par INRIA.

[VD96] Véhel (Jacques Lévy) et Daoudi (Khalid). – Generalized ifs for signal processing. *In : IEEE DSP Workshop, Loen, Norway, September 1-4*.

[VR94] Vences (Lucia) et Rudomin (Isaac). – *Fractal Compression of Single Images and Image Sequences using Genetic Algorithms*. – Rapport technique, 1994.

[WdJ99] Wohlberg (Brendt) et de Jager (Gerhard). – A review of the fractal image coding literature. *IEEE Transactions on Image Processing*, vol. 8, N° 12, Décembre 1999, pp. 1716–1729.

[Zai98] Zair (Chems Eddine). – *Formes fractales à pôles basées sur une généralisation des IFS*. – Thèse de doctorat, Université Claude Bernard Lyon 1, Juin 1998.

[ZT96] Zair (Chems Eddine) et Tosan (Eric). – Fractal modeling using free form techniques. *Computer Graphics Forum*, vol. 15, N° 3, Août 1996, pp. 269–278. – EUROGRAPHICS'96 Conference issue.

Résumé

L'approximation des objets naturels (courbes, surfaces ou images) par des formes fractales constitue aujourd'hui un important centre d'intérêt pour la recherche. Sous le nom générique de problème inverse, cette problématique intéresse en effet des domaines d'applications aussi divers que la représentation synthétique de l'information pour la transmission et la compression d'images, la reconstruction 3D pour la visualisation ou la CAO. Diverses études, guidées par l'utilisation de modèles ou de méthodes spécifiques, ont été menées pour répondre à ce problème inverse fractal. La plus connue est certainement la méthode dite de compression fractale d'images introduite par Jacquin. D'un manière générale, ces techniques existentes souffrent aujourd'hui d'un manque de souplesse en terme de contrôle sur la forme utilisée pour l'approximation. De plus, l'espace d'itération utilisé se limite à l'espace de visualisation, c'est à dire $\mathbb{R}^2$. Des travaux antérieurs ont permis de dégager un modèle générique fractal : les formes fractales à pôles. Ce modèle, par l'intermédiaire d'une projection, permet de définir les auto-similarités d'un objet dans un espace de dimension supérieure. Nous proposons une résolution du problème inverse fondée sur ce modèle et sur un algorithme de régression non-linéaire. Cette méthode générale permet d'approximer des courbes et des surfaces, aussi bien lisses que rugueuses, ainsi que des images en niveaux de gris. Une extension hiérarchique du modèle est introduite pour la modélisation d'objets hétérogènes dont les caractéristiques varient dans l'espace. Deux algorithmes sont proposés pour résoudre le problème d'approximation associé. Le premier calcule, à partir d'une image et d'un critère de distorsion, le modèle qui uniformise cette distorsion. Le deuxième décrit une méthode complète d'optimisation du codage du modèle en vue de comprimer les images à partir d'un critère de débit. Les résultats montrent l'avantage de ce type de modèle dans le domaine de la compression à faible débit.

Mots-clés: fractal, approximation, IFS, courbes, surfaces, images, compression.

Abstract

Approximation of natural objects (curves, surfaces, or images) with fractal models is an important center of interest for research. The general inverse problem paradigm concerns many application fields, including information representation for image transmission or compression, 3D reconstruction for visualization or CAGD. A large variety of studies, using specific models or methods, have been proposed to address this inverse fractal problem. The most known of them is the fractal image compression method introduced by Jacquin. Generally speaking, these techniques lack of flexibility in term of control over the approximated shape. Furthermore, iteration space used is the visualisation space, $\mathbb{R}^2$. Previous work achieved a general framework for fractal modeling : fractal free forms. This model allows user to define self-similar objects in a space of a higher dimension. We propose a resolution of the inverse problem base on this model and a non-linear regression algorithm. This versatile method allows the approximation of curves and surfaces, even rough or smooth, and also grey-level images. A hierachical extension of this model is introduced for modeling heterogeneous objects, for which characteristics are varying in space. Two algorithms are proposed for the associated approximation problem. The first one computes, given a grey-level image and a distortion criteria, the model that gives a uniform repartition of this distortion. The second one is a complete optimisation of the rate/distortion ratio, given a rate budget. Results show that this type of model is interesting for low rate compression.

Keywords: fractal, approximation, IFS, curves, surfaces, images, compression.