Nouvelles méthodes de traitement de signaux multidimensionnels par décomposition suivant le théorème de Superposition de Kolmogorov

Pierre-Emmanuel Leni

Université de Bourgogne
Ecole Doctorale Environnement - Santé / STIC (E2S)

Thèse

présentée par

Pierre-Emmanuel Leni

pour obtenir le grade de

Docteur de l'Université

discipline Instrumentation et Informatique de l'Image

Nouvelles méthodes de traitement de signaux multidimensionnels par décomposition suivant le théorème de Superposition de Kolmogorov

---

Soutenue le 23 Novembre 2010

Jury

Atilla BASKURT	Professeur à l'INSA Lyon	Rapporteur
Jocelyn CHANUSSOT	Professeur au Grenoble Institute of Technology	Rapporteur
Boris IGELNIK	Associate Professor à l'Université de Case Western Reserve	Examinateur
David A. SPRECHER	Professeur à l'Université de Californie (Santa Barbara)	Examinateur
Cédric DEMONCEAUX	Maitre de Conférence à l'Université de Bourgogne	Examinateur
Yohan FOUGEROLLE	Maitre de Conférence à l'Université de Bourgogne	Examinateur
Frédéric TRUCHETET	Professeur à l'Université de Bourgogne	Examinateur

Remerciements

Cette thèse s'est déroulée au Laboratoire LE2I [1]. Elle a bénéficié d'un financement du CNRS à travers une bourse BDI.

Je remercie les Professeurs Jean-Marie BILBAULT et David FOFI, respectivement directeur du laboratoire et responsable de l'équipe creusotine, de m'avoir accueilli au Laboratoire LE2I.

Je souhaite vivement remercier Monsieur Atilla Baskurt, Professeur à l'université de Lyon 2, et Monsieur Jocelyn Chanussot, Professeur à l'université de Grenoble d'avoir accepté d'être rapporteurs de ce mémoire. Je tiens à remercier également messieurs David Sprecher, Professeur à l'université de Californie, Boris Igelnik, Associate Professor à l'université de Case Western Reserve, et Cédric Demonceaux, Maitre de Conférence à l'Université de Bourgogne pour l'honneur qu'ils m'ont fait en acceptant d'être membre de mon jury.

Je ne remercierai jamais assez Messieurs Frédéric Truchetet et Yohan Fougerolle, respectivement directeur de thèse et encadrant, pour leurs précieux conseils, leur disponibilité et leur soutien tout au long de ma thèse.

Mes remerciements s'adressent également à tous les enseignants-chercheurs, doctorants et stagiaires du Creusot, Nathalie, sans oublier l'équipe *Revenge* avec qui j'ai eu l'immense plaisir de travailler durant cette thèse.

Et enfin, je souhaite remercier ma famille, mes amies et amis, pour leur soutien et leur sympathie durant ces trois années de thèse.

1. Laboratoire LE2I UMR CNRS 5158, 12 rue de la fonderie, 71200 LE CREUSOT

Résumé

Le traitement de signaux multidimensionnels reste un problème délicat lorsqu'il s'agit d'utiliser des méthodes conçues pour traiter des signaux monodimensionnels. Il faut alors étendre les méthodes monodimensionnelles à plusieurs dimensions, ce qui n'est pas toujours possible, ou bien convertir les signaux multidimensionnels en signaux $1D$. Dans ce cas, l'objectif est de conserver le maximum des propriétés du signal original.

Dans ce contexte, le théorème de superposition de Kolmogorov fournit un cadre théorique prometteur pour la conversion de signaux multidimensionnels. En effet, en 1957, Kolmogorov a démontré que toute fonction multivariée pouvait s'écrire comme sommes et compositions de fonctions monovariées. Notre travail s'est focalisé sur la décomposition d'images suivant le schéma proposé par le théorème de superposition, afin d'étudier les applications possibles de cette décomposition au traitement d'image. Pour cela, nous avons tout d'abord étudié la construction des fonctions monovariées. Ce problème a fait l'objet de nombreuses études, et récemment, deux algorithmes ont été proposés. Sprecher a proposé dans [Sprecher, 1996; Sprecher, 1997] un algorithme dans lequel il décrit explicitement la méthode pour construire exactement les fonctions monovariées, tout en introduisant des notions fondamentales à la compréhension du théorème. Par ailleurs, Igelnik et Parikh ont proposé dans [Igelnik and Parikh, 2003; Igelnik, 2009] un algorithme pour approcher les fonctions monovariées par un réseau de splines.

Nous avons appliqué ces deux algorithmes à la décomposition d'images. Nous nous sommes ensuite focalisé sur l'étude de l'algorithme d'Igelnik, qui est plus facilement modifiable et offre une représentation analytique des fonctions, pour proposer deux applications originales répondant à des problématiques classiques de traitement de l'image :

– pour la compression : nous avons étudié la qualité de l'image reconstruite par un réseau de splines généré avec seulement une partie des pixels de l'image originale. Pour améliorer cette reconstruction, nous avons proposé d'effectuer cette décomposition sur des images de détails issues d'une transformée en ondelettes. Nous avons ensuite combiné cette méthode à JPEG 2000, et nous montrons que nous améliorons ainsi le schéma de compression JPEG 2000, même à bas bitrates.
– pour la transmission progressive : en modifiant la génération du réseau de splines, l'image peut être décomposée en une seule fonction monovariée. Cette fonction peut être transmise progressivement, ce qui permet de reconstruire l'image en augmentant progressivement sa résolution. De plus, nous montrons qu'une telle transmission est résistante à la perte d'information.

Mots-clefs : Théorème de Superposition de Kolmogorov, décomposition de fonctions multivariées, traitement de signal, compression d'image, transmission progressive d'image

Abstract

Novel processing methods for multidimensional signals using decompositions by the Kolmogorov Superposition theorem

The processing of multidimensional signal remains difficult when using monodimensional-based methods. Therefore, it is either required to extend monodimensional methods to several dimensions, which is not always possible, or to convert the multidimensional signals into $1D$ signals. In this case, the priority is to preserve most of the properties of the original signal.

In this context, the Kolmogorov Superposition Theorem offers a promising theoretical framework for multidimensional signal conversion. In 1957, Kolmogorov demonstrated that any multivariate function can be written as sums and compositions of monovariate functions. We have focused our study on the research of applications in image processing of the image decomposition according to the superposition theorem scheme. We have first studied the monovariate function constructions. Various studies have dealt with this problem, and recently, two algorithms have been proposed. Sprecher has proposed in [Sprecher, 1996; Sprecher, 1997] an algorithm in which the method to exactly build the monovariate functions is described, as well as fundamental notions for the understanding of the theorem. Igelnik and Parikh have proposed in [Igelnik and Parikh, 2003; Igelnik, 2009] an algorithm to approximate the monovariate functions by a Spline network.

We have applied both algorithms to image decomposition. We have chosen to use Igelnik's algorithm which is easier to modify and provides an analytic representation of the functions, to propose two novel applications for classical problems in image processing :

– for compression : we have studied the quality of a reconstructed image using a spline network built with only a fraction of the pixels of the original image. To improve this reconstruction, we have proposed to apply this decomposition on images of details obtained by wavelet transform. We have then combined this method with JPEG 2000, and we show that the JPEG 2000 compression scheme is improved, even at low bitrates.
– for progressive transmission : by modifying the spline network construction, the image can be decomposed into one monovariate function. This function can be progressively

transmitted, which allows to reconstruct the image by progressively increasing its resolution. Moreover, we show that such a transmission is resilient to information lost.

Keywords : Kolmogorov superposition theorem, multidimensional function decomposition, signal processing, image compression, Progressive Image Transmission

Table des matières

Introduction

En traitement du signal, la plupart des méthodes existantes sont soit basées sur des techniques utilisant une représentation monodimensionnelle, soit, dans le cas de signaux bidimensionnels comme des images par exemple, sur des bases de fonctions séparables, comme les ondelettes. Une problématique récurrente est l'extension de ces méthodes au traitement des signaux multidimensionnels pour lesquels aucun ordonnancement "naturel" n'est disponible. La question se pose notamment pour les méthodes et techniques de filtrage, d'analyse statistique locale, de corrélation ou d'analyses multirésolutions.
Une première solution, la plus intuitive, consiste à étendre une méthode monodimensionnelle à plusieurs dimensions, ce qui pose plusieurs problèmes. Par exemple, considérons la notion de voisinage. En $1D$, il est très facile de connaitre le voisinage d'un point. Cette tâche devient déjà plus complexe en $2D$, et très ardue en $3D$. De plus, une telle extension peut également poser des problèmes de complexité algorithmique, qui n'augmente pas nécessairement linéairement avec le nombre de dimensions.
Plutôt que d'essayer d'étendre des techniques à des dimensions supérieures, une alternative consiste à décomposer le signal multidimensionnel en un ou plusieurs signaux monodimensionnels. Dans le cas d'un signal échantillonné, le problème se ramène souvent à essayer de conserver le maximum de propriétés du signal original, mais pour un signal continu, le problème est nettement plus délicat. La solution passe souvent par une approximation échantillonnée et s'accompagne d'une perte potentielle d'information. Par exemple, la notion de voisinage d'un pixel en $2D$ est généralement définie en utilisant les 4 ou 8 pixels situés autour du pixel considéré. Cette information est directement obtenue puisque les images ne sont qu'un tableau bidimensionnel, et bénéficient d'une indexation. Pourtant, lorsqu'une image est convertie en $1D$, pour être filtrée par exemple, les lignes

ou les colonnes sont simplement mises les unes à la suite des autres (codage TV). Avec l'augmentation des dimensions, la notion de voisinage devient rapidement délicate à gérer efficacement. En outre, rien ne garantit la préservation d'une indexation directe comme pour les images. Par exemple, il suffit de considérer le cas de surfaces représentées par des maillages pour lesquels les voisinages de points ne sont plus nécessairement constants en nombre, bien que toujours représentant un espace $2D$ mais plongé dans un espace $3D$.

Dans ce contexte, le théorème de Superposition de Kolmogorov (TSK) se présente comme un outil particulièrement puissant puisqu'en 1957, Kolmogorov a démontré que toute fonction continue multivariée pouvait se décomposer en sommes et compositions de fonctions monovariées continues. Ce théorème fournit donc un cadre commun pour la conversion de signaux multidimensionels en signaux monodimensionnels. Malheureusement, Kolmogorov ayant seulement démontré l'existence des fonctions monodimensionnelles, ce théorème fondamental n'a pas pu être appliqué immédiatement. Ainsi, malgré plusieurs contributions mathématiques et théoriques, il est resté inexploité, et certains auteurs ont même fini par penser qu'une implémentation n'était pas possible ([Girosi and Poggio, 1989]). Heureusement, à la fin des années 90, plusieurs chercheurs ont commencé à proposer des algorithmes pour construire ces fonctions monodimensionnelles. L'article le plus complet et le plus accessible est dû à Sprecher ([Sprecher, 1996; Sprecher, 1997]), qui a proposé un algorithme directement implémentable pour construire des fonctions monodimensionnelles exactes à une précision donnée.
La deuxième contribution majeure pour la construction des fonctions monovariées a été proposée par Igelnik et Parikh dans [Igelnik and Parikh, 2003] et [Igelnik, 2009]. L'approche retenue est cette fois différente : l'architecture du théorème est assimilée à la construction d'un réseau de neurones, et les fonctions monovariées sont approchées par des splines.

Nous avons implémenté ces deux algorithmes dans le but d'explorer les possibilités offertes par une décomposition selon le schéma de Kolmogorov dans le domaine du traitement d'image. Plus précisément, nous avons tout d'abord appliqué l'algorithme de Sprecher à la décomposition d'images. Il s'est alors posé le problème de pouvoir étudier plus précisément les fonctions monodimensionnelles obtenues et surtout de les modifier, ce qui reste

difficile avec cet algorithme qui propose un schéma rigide sans espace de paramétrage. Nous avons alors implémenté l'algorithme d'Igelnik, qui propose une représentation moins précise mais beaucoup plus flexible. L'étude et l'adaptation de cet algorithme nous a conduit à proposer deux nouvelles approches répondant de façon originale à deux problématiques classiques en traitement d'images. Nous avons tout d'abord proposé une nouvelle méthode de compression, s'appuyant sur la décomposition en fonctions monodimensionnelles d'images issues d'une décomposition en ondelettes, puis nous avons modifié l'algorithme original afin de proposer une méthode de transmission progressive des images résistante au bruit.

Contributions

Dans un premier temps, nous proposons une présentation avec une notation unifiée des algorithmes de Sprecher et d'Igelnik, permettant de décomposer un signal multidimensionnel selon le schéma proposé par le théorème de superposition. Cette présentation est suivie d'une comparaison et d'une implémentation de ces deux algorithmes dans le cas de la décomposition d'images, pouvant être vues comme une discrétisation de fonctions bidimensionnelles continues.

Nous présentons ensuite un nouveau schéma de compression utilisant une version modifiée de l'algorithme d'Igelnik. Cette approche, s'appuyant sur une décomposition en ondelettes des images, a été intégrée à JPEG 2000 afin d'améliorer les performances de ce standard, et surtout de positionner la méthode dans l'état de l'art. Plus précisément, nous proposons de décomposer les images de détails issues de la transformée en ondelettes, puis d'effectuer des simplifications sur les fonctions monodimensionnelles, afin de réduire la taille des images de détails en réduisant le nombre de coefficients qu'elles contiennent. Nous montrons que la méthode apporte une amélioration sensible des performances de JPEG 2000, même si toutes les potentialités ne peuvent être valorisées dans ce contexte. Nous avons également proposé une étude portant sur l'influence de la modification des données dans l'espace $1D$ sur l'entropie de l'image décomposée, afin de mieux comprendre le fonctionnement de la compression obtenue par notre approche.

Nous présentons également une modification de l'algorithme d'Igelnik pour la transmission progressive d'images : en tirant parti de la flexibilité de l'algorithme, il est possible d'effectuer la transmission de l'image dans l'espace des fonctions monodimensionnelles. Ces fonctions peuvent être transmises progressivement, et permettent de reconstruire une image globalement fidèle à l'originale en cas d'erreurs survenues pendant la transmission. L'algorithme montre une résistance particulièrement bonne aux erreurs en paquets.

Organisation du document

Le chapitre 1 expose le contexte historique lié au théorème de superposition de Kolmogorov : les différentes contributions mathématiques et applications proposées par les chercheurs, son énoncé, ainsi que des explications générales sur l'architecture du théorème. Dans une deuxième partie, nous passons en revue quelques notions de traitement d'images utiles à la compréhension des travaux présentés par la suite.

Dans le chapitre 2, nous présentons les deux algorithmes implémentés au cours de ce travail de thèse, ainsi qu'une discussion sur ces deux approches, où nous exposons la raison du choix de l'algorithme d'Igelnik pour les applications à la compression et à la transmission progressive que nous proposons.

Le chapitre 3 présente notre application du théorème de superposition par l'algorithme d'Igelnik à la compression d'images. Nous détaillons les différentes étapes de notre approche et les résultats obtenus. Nous détaillons également l'intégration de notre approche dans JPEG 2000, ainsi que les résultats obtenus avec ce schéma JPEG 2000 modifié, comparés à ceux de la version originale.

Dans le chapitre 4, nous présentons une seconde modification de l'algorithme d'Igelnik pour la transmission d'images progressive en résolution. Après avoir détaillé les modifications de l'algorithme, nous illustrons la flexibilité de notre approche sur plusieurs exemples. Dans une deuxième partie, nous étudions la résistance au bruit de notre approche, que nous illustrons avec divers exemples de perturbations et les reconstructions obtenues.

Dans la dernière section, nous présentons nos conclusions et les perspectives de recherche liées à l'application du théorème de superposition au traitement des signaux multidimensionnels.

Chapitre 1

Contexte théorique

Dans la première partie de ce chapitre, nous présentons le contexte historique du théorème de superposition et les contributions les plus importantes depuis sa démonstration par Kolmogorov. Dans ce mémoire, nous proposons deux applications au théorème de superposition liées à deux problématiques distinctes de traitement d'images : la compression et la transmission progressive. Dans un souci de clarté, nous avons choisi d'inclure les états de l'art correspondant à ces deux domaines aux chapitres 3 et 4 présentant ces applications, et de présenter dans ce chapitre uniquement le contexte général de nos travaux et le théorème de superposition. Dans la deuxième partie de ce chapitre, nous rappelons les notions de traitement d'image que nous utiliserons dans la suite de ce mémoire.

1.1 Le théorème de Superposition

1.1.1 Historique et contributions

En 1900, Hilbert a énoncé une série de 23 problèmes qui représentaient selon lui les principaux défis à relever pour les mathématiciens. Pendant le siècle suivant, ces problèmes ont en effet conduit à de nombreux développements en mathématiques ([Grattan-Guinness, 2000]),

et une de ces 23 conjectures a été résolue par la démonstration du théorème de superposition de Kolmogorov (TSK). Dans le 13^{e} problème, Hilbert énonce l'hypothèse selon laquelle il serait impossible de résoudre l'équation algébrique générale du septième degré (de la forme $a_7x^7 + a_6x^6 + a_5x^5 + ... + a_1x + a_0 = 0$) par des fonctions de seulement deux variables, comme c'était à l'époque le cas pour les équations jusqu'au degré 6. Au moyen de fonctions d'une variable et d'additions, la solution de l'équation générale du 7^{e} degré se ramène à celle de l'équation $x^7 + ux^3 + vx^2 + wx + 1 = 0$, c'est-à-dire à une fonction des trois variables u, v et w. Hilbert conjecture qu'on ne peut pas aller plus loin, *i.e.* qu'on ne peut obtenir cette solution en superposant des fonctions de deux variables. Ce sont Arnold et Kolmogorov qui ont prouvé en 1957 que cette conjecture était fausse en donnant une preuve d'existence des fonctions monodimensionnelles composant le TSK. Pour plus de détails sur le 13^{e} problème de Hilbert et la théorie se rapportant au théorème de superposition, le lecteur intéressé peut se reporter à [Lorentz, 1976] et [Vitushkin, 2004].

La plupart des contributions liées au TSK concernent les différentes techniques de sa représentation sous forme de réseau de neurones, le choix des fonctions monodimensionnelles et leur calculabilité. La représentation sous forme de réseau de neurones est proposée par Hecht-Nielsen, qui montre en 1987 (dans [Hecht-Nielsen, 1987]) que le TSK peut s'écrire sous la forme d'un réseau de neurones non récurrent, avec une couche d'entrée, une couche cachée et une couche de sortie. Malheureusement, les fonctions internes qu'il propose ne sont pas calculables, car définies par des sommes infinies de fonctions. De plus, les fonctions externes n'ont pas de forme paramétrique. Ces problèmes amènent Girosi et Poggio à considérer que le TSK est inapplicable sous la forme d'un réseau de neurones ([Girosi and Poggio, 1989]). Cette hypothèse a été réfutée par Kurkova, qui a démontré dans [Kurková, 1991] et [Kurková, 1992] que le réseau de neurones pouvait être implémenté en approchant les fonctions monodimensionnelles. Une estimation de la qualité de l'approximation et du nombre de neurones constituant le réseau est proposée. Cette approche s'appuie sur le travail de Sprecher, qui démontre dans [Sprecher, 1965] et [Sprecher, 1972] que le nombre de fonctions monodimensionnelles à construire est inférieur à ce que proposait Kolmogorov dans l'énoncé original (voir la section 1.1.3).
Arnold et Kolmogorov ont prouvé le théorème de superposition en démontrant l'existence

des fonctions monodimensionnelles, mais ils n'ont pas proposé de méthode de construction explicite. La construction de ces fonctions monodimensionnelles, nécessaire pour envisager de possibles applications au théorème, a fait l'objet de nombreuses recherches. Nees propose dans [Nees, 1994] une preuve constructive en introduisant une approximation bornée. Nakamura, dans [Nakamura et al., 1993], propose également un algorithme de constructions des fonctions monodimensionnelles, tel que le nombre de ces fonctions soit indépendant de la précision de l'approximation recherchée. Toutefois, dans ce domaine des algorithmes de construction des fonctions monodimensionnelles, une des contributions les plus remarquables est due à Sprecher, qui propose dans [Sprecher, 1996] et [Sprecher, 1997] un algorithme clair et directement implémentable pour la construction des fonctions monodimensionnelles. Toutefois, cette approche repose sur le calcul des fonctions monodimensionnelles à une précision donnée, et elles ne sont pas exprimées sous une forme explicite. Brattka, dans [Brattka, 2004] a démontré que les fonctions continues mentionnées dans le TSK pouvaient être remplacées par des fonctions calculables (au sens des fonctions pouvant être évaluées par une machine de Turing), et a ainsi validé l'implémentation proposée par Sprecher. Braun, dans [Braun and Griebel, 2009], a amélioré l'algorithme de Sprecher : il a noté qu'une des fonctions construite par Sprecher n'était pas continue et monotone sur tout son domaine de définition, et a proposé une nouvelle construction pour cette fonction. Un deuxième algorithme de construction a été proposé par Igelnik et Parikh, dans [Igelnik and Parikh, 2003] et [Igelnik, 2009]. Cet algorithme approche la fonction multidimensionnelle considérée, mais permet une représentation analytique claire des fonctions monodimensionnelles par des combinaisons de fonctions splines, appelées Kolmogorov Spline Network (KSN). Cette approche fournit ainsi une représentation paramétrique ouvrant la porte aux travaux présentés dans ce manuscrit. L'interpolation par des splines assure la continuité des fonctions monodimentionnelles et la convergence de l'algorithme est démontrée lorsque le nombre de ces fonctions croît.

1.1.2 Applications existantes

Bien que le TSK soit un des théorème les plus utilisés dans des domaines comme les réseaux de neurones, ses applications directes tout comme ses adaptations pour des problèmes

spécifiques restent peu nombreuses dans la littérature. Parmi ces applications, Pednault, dans [Pednault, 2006], propose un algorithme de modélisation prédictive. Une fonction multidimensionnelle est décomposée selon le schéma du TSK en un réseau de fonctions monodimensionnelles. Ce réseau est ensuite utilisé comme un réseau de neurones, pour extrapoler les données originales utilisées pour sa construction, en mettant à profit la réorganisation de l'information à l'intérieur des fonctions monodimensionnelles.

Dans le domaine du traitement de l'image, Köppen, dans [Köppen, 2002], a appliqué l'algorithme de Sprecher à la décomposition d'images en niveau de gris et propose dans [Köppen and Yoshida, 2005] une modification de l'algorithme adaptée à la décomposition de fonctions multidimensionnelles discrètes. Plus précisément, il propose d'abandonner la continuité de certaines fonctions monodimensionnelles lorsque l'algorithme est utilisé pour la décomposition d'images. De plus, cette approche peut être utilisée pour compresser l'image en effectuant des simplifications dans les fonctions monodimensionnelles. Seule une estimation empirique des résultats est proposée, et cette approche n'a pas pour but de concurrencer les méthodes de construction existantes. Cependant, les approches de Pednault et Köppen ont un point commun, celui d'interpoler des données multidimensionnelles à partir de leur décomposition en fonctions monodimensionnelles. En effet, les données contenues dans les fonctions monodimensionnelles sont ré-ordonnées avec des relations de voisinage différentes des données originales. Ces approches cherchent donc à exploiter les nouvelles interpolations possibles une fois les données décorrélées de leurs positions spatiales originales. Une thèse peut également être signalée : dans [Bryant, 2008], l'auteur expose une construction des fonctions proches de Sprecher, et propose d'utiliser une partie des fonctions monodimensionnelles issues de la décomposition d'images comme fonctions de hashage, afin d'indexer ces images dans une base de données.

1.1.3 Enoncés

Kolmogorov, dans [Kolmogorov, 1957] et [Kolmogorov, 1963], a montré que toute fonction continue f à d dimensions, $d \geqslant 2$, définie sur l'hypercube unité $[0,1]^d$, était décomposable

en sommes de fonctions continues monodimensionnelles ψ_{ni} et g_n sous la forme :

$$\begin{cases} f(x_1, ..., x_d) = \sum_{n=0}^{2d} g_n\Big(\xi_n(x_1, ..., x_d)\Big) \\ \\ \xi_n(x_1, ..., x_d) = \sum_{i=1}^{d} \psi_{ni}(x_i), \end{cases} \tag{1.1}$$

où les fonctions ψ_{ni} sont croissantes monotones et indépendantes de f.

Le nombre de ces fonctions monodimensionnelles a par la suite été réduit. Lorentz, dans [Lorentz, 1962], a prouvé qu'une seule fonction g était nécessaire, au lieu de $2d+1$ fonctions g_n. Puis Sprecher, dans [Sprecher, 1965] et [Sprecher, 1972], a remplacé les fonctions ψ_{ni} par la combinaison linéaire $\lambda_i\psi_n$, avec λ_i des constantes linéairement indépendantes et ψ_n une fonction identique pour toutes les dimensions $i, 1 \leqslant i \leqslant d$. Le TSK, reformulé en tenant compte des simplifications apportées par Sprecher et par Lorentz, s'écrit finalement :

Théorème 1 (Le théorème de Superposition de Kolmogorov) *Toute fonction continue définie sur l'hypercube identité, $f : [0,1]^d \longrightarrow \mathbb{R}$, avec $d \geqslant 2$, peut s'écrire comme sommes et compositions de fonctions monodimensionnelles continues :*

$$\begin{cases} f(x_1, ..., x_d) = \sum_{n=0}^{2d} g\Big(\xi(x_1 + bn, ..., x_d + bn)\Big) \\ \\ \xi(x_1 + bn, ..., x_d + bn) = \sum_{i=1}^{d} \lambda_i\psi(x_i + bn), \end{cases} \tag{1.2}$$

avec $\lambda_i, i \in [\![1, d]\!]$ et b des constantes. ψ est appelée fonction interne, et g fonction externe.

Les coordonnées $x_i, i \in [\![1, d]\!]$ de chaque dimension i sont combinées en un nombre réel par une fonction de hashage ξ (obtenue par combinaison linéaire des fonctions internes ψ). Ce nombre réel est associé à la valeur de f correspondant à ces coordonnées par la fonction g. On peut noter que les fonctions ξ et ψ sont indépendantes de f. Le nombre de couches n est constant et dépend de la dimensionnalité d de la fonction f.

L'implémentation de Sprecher s'appuie sur cette dernière formulation, tandis que l'algorithme d'Igelnik et Parikh emprunte aux deux énoncés 1.1 et 1, afin de pouvoir utiliser une construction similaire à un réseau de neurones. Ces deux implémentations du théorème

sont détaillées dans le chapitre 2, qui contient également une discussion sur les différences entre ces deux approches. Toutefois, nous détaillons dans la section suivante l'organisation et le rôle des fonctions monodimensionnelles définis par le théorème, et qui sont donc communs aux deux algorithmes.

1.1.4 Principes de la décomposition

Les fonctions monodimensionnelles construites lors de la décomposition d'un signal multidimensionnel se séparent en deux familles, chaque famille de fonctions assurant un rôle spécifique. On distingue les fonctions dites internes, qui contiennent les informations de localisation dans l'espace multidimensionnel ; et les fonctions dites externes, qui contiennent les données de la fonction multidimensionnelle.

Les fonctions internes sont construites indépendamment de la fonction f et dépendent uniquement de la dimension d. Leur rôle est d'identifier toute position de l'hypercube $[0,1]^d$ par un réel unique qui sera l'argument des fonctions externes. Pour cela, chaque fonction interne subdivise une dimension en intervalles disjoints identifiés de façon unique. Par extension, le domaine de définition $\mathscr{D}_f$ de la fonction f, l'hypercube $[0,1]^d$, est découpé en pavage d'hypercubes disjoints, qui sont tous identifiés de manière unique par les fonctions internes. Afin de couvrir tout l'espace $\mathscr{D}_f$, plusieurs couches de pavages sont générées par translation. Autrement dit, un sous-échantillonnage de la fonction multidimensionnelle est réalisé par les fonctions internes. Ce sous-échantillonnage est original dans le sens où les informations sont réorganisées spatialement : le voisinage dans la fonction externe dépend de l'identification des coordonnées par les fonctions internes. Ainsi, les relations de voisinage contenues dans les données originales sont totalement modifiées. Par la suite, une fois une position identifiée par un réel, il faut alors calculer les valeurs de la fonction multidimensionnelle correspondant à cette position : c'est le rôle des fonctions externes.

Les fonctions externes peuvent être vues comme un sous-échantillonnage particulier de la fonction multidimensionnelle. En effet, ce sous-échantillonnage est double : d'une part, la fréquence de l'échantillonnage est déterminée par la densité du pavage (contrôlée par

les fonctions internes) ; et d'autre part, chaque échantillon contient seulement une fraction (déterminée dans la fonction externe) de la valeur de f : c'est la superposition (la somme) de ces approximations successives pour une position spatiale qui converge vers la valeur de f à cette position. Plus précisément, à chacun des réels obtenus par les fonctions internes correspond une zone ou une position dans l'espace multidimensionnel de la fonction f. La fonction externe associe à ces réels une valeur, qui une fois additionnée aux approximations partielles obtenues avec les autres pavages, permet de reconstruire la fonction f. Ainsi, la position des données dans la fonction externe (monodimensionnelle), dépend des valeurs obtenues dans les fonctions internes. Autrement dit, la combinaison linéaire des fonctions internes fournit un ordre de lecture des données multidimensionnelles, qui détermine l'ordre des données dans les fonctions externes. Cela signifie que le voisinage des données originales n'est pas préservé dans les fonctions externes, puisque ce parcours des données multidimensionnelles dépend des fonctions internes, construites indépendamment de la fonction multidimensionnelle décomposée. Le signal obtenu lors de la construction des fonctions externes est de nature différente du signal obtenu par un balayage ligne par ligne de l'image : les fonctions externes ont la forme d'un bruit blanc. Ce changement signifie qu'un traitement de filtrage par exemple, qui calcule les nouvelles valeurs des pixels en fonction de ses voisins, n'a pas de sens s'il est appliqué directement sur les fonctions monodimensionnelles. Toutefois, les statistiques du signal restent comparables à celles de l'image originale, car le contenu de ces fonctions est dépendant des pixels de l'image. Cette décomposition originale de l'image en signaux "bruités" permet d'envisager de nombreuses perspectives, comme par exemple un codage s'appuyant sur les haute-fréquences du signal. On peut également envisager des applications de cryptage et de Watermarking, puisque les fonctions externes ont un comportement erratique, ce qui rend possible la dissimulation d'informations.

1.2 Traitement d'image

Les deux applications du théorème de superposition que nous proposons sont liées à deux problématiques classiques souvent abordées en traitement d'image, la compression et la

transmission progressive. Ces deux domaines utilisent des notions communes que nous rappelons dans cette section : le calcul de la mesure de qualité des images et la problématique du codage des images.

1.2.1 Mesures de qualité

Les mesures de qualité permettent d'évaluer la distortion entre deux images, généralement entre une image originale et une image ayant subi un traitement (une compression ou une transmission par exemple). Pour cela, il s'agit de comparer deux images afin de déterminer quels pixels ont été altérés, et de combien. Parmi les mesures d'erreur, trois sont particulièrement utilisées dans la littérature : l'erreur quadratique moyenne ("Mean Square Error (MSE)"), dont on peut déduire le rapport signal sur bruit ("Signal to Noise Ratio (SNR)"), qui permet de calculer la valeur de pic du SNR ("Peak Signal to Noise Ratio (PSNR)"). Ces mesures comparent pixel à pixel la distortion entre deux images, l'originale qui sert de référence, et l'image dont on cherche à évaluer la qualité, qui présente éventuellement des erreurs par rapport à l'image originale. Cette distortion peut être vue comme la distance (ou la différence) entre ces deux images.

$$MSE = \frac{1}{MN} \sum_{y=1}^{M} \sum_{x=1}^{N} \left[I(x,y) - I'(x,y) \right]^2$$

$$SNR = \frac{\sum_{y=1}^{M} \sum_{x=1}^{N} I(x,y)}{\sqrt{MSE}}$$

$$PSNR = 20 \times \ln_{10} \left(\frac{\max\limits_{1 \leqslant x \leqslant N, 1 \leqslant y \leqslant M} I(x,y)}{\sqrt{MSE}} \right)$$

$I(x,y)$ et $I'(x,y)$ sont les niveaux de gris des images aux coordonnées x et y. M et N sont les résolutions en largeur et en hauteur de l'image.

Une faible erreur quadratique traduit donc deux images presque identiques. Inversement, cela se traduit par un PSNR élevé (voir même infini dans le cas d'une image non dégradée), ce qui signifie que le rapport signal sur bruit est plus élevé, sachant que dans ce cas, le "signal" est l'image originale, et le "bruit" est l'ensemble des erreurs de l'image reconstruite par rapport à l'image originale.

Néanmoins, ces mesures évaluent la dégradation, mais ne renseignent pas sur la nature de la dégradation. De plus, elles ne sont pas parfaitement représentatives de la qualité perçue par l'œil humain, qui ne considère pas l'image pixel par pixel mais se focalise sur certaines propriétés, notamment spatiales (les contours) et chromatiques. Ainsi, entre deux images dégradées obtenues d'une même image originale, celle ayant la valeur de PSNR la plus élevée peut correspondre à une qualité visuelle perçue plus faible. Pour combler ces lacunes, d'autres approches ont été proposées, adaptées à certains types de documents (par exemple la mesure de distortion DRDM "Distance-Reciprocal Distortion Measure" ([Lu et al., 2002]) pour les images binaires), ou essayant d'imiter la subjectivité de la vision humaine. On peut citer par exemple la mesure introduite dans [Beghdadi and Pesquet-Popescu, 2003], qui calcule la distortion dans l'espace ondelette, ce qui permet d'obtenir un critère d'évaluation plus cohérent avec la qualité perçue par un observateur humain. Il existe également des évaluations subjectives, s'appuyant sur un panel d'utilisateurs qui évaluent la qualité des images.
Nous avons choisi d'utiliser le PSNR, qui a l'avantage d'être simple et populaire, et d'accompagner les principaux résultats par des illustrations visuelles.

1.2.2 Codage

Comme tout signal numérique destiné à être traité par un ordinateur, les images sont codées en bits. Le nombre de bits nécessaires, en moyenne, pour coder un pixel, définit le bitrate, exprimé en bits par pixel (bpp). Par exemple, pour une image en 256 niveaux de gris, chaque pixel est codé sur 8 bits ($2^8 = 256$). Il faut distinguer ce bitrate de celui donné en bits par seconde qui exprime la bande passante d'un canal de transmission. En transmission progressive, le but est de reconstruire l'image en plusieurs étapes intermédiaires,

en augmentant progressivement le bitrate ; et l'objectif en compression d'image est de réduire ce nombre le plus possible, tout en conservant une image compressée de la meilleure qualité possible (un PSNR élevé). Pour cela, les techniques de codage existantes cherchent à simplifier les informations contenues dans l'image, ou à réorganiser l'information afin de pouvoir l'encoder plus efficacement.

La théorie de l'information de Shannon propose un calcul statistique, qui permet d'évaluer le bitrate de l'image si un codage optimal était utilisé. La grandeur correspondante, l'entropie, est notée H :

$$H = -\sum_{i=0}^{255} p_i \ln_2(p_i)$$

où p_i est la probabilité d'apparition du niveau de gris i, en fait son estimation, *i.e.* le nombre de pixels ayant ce niveau i divisé par le nombre total de pixels de l'image.

Ce calcul permet d'estimer la taille de l'image après compression et codage, et de la comparer avec sa taille originale pour évaluer la compression réalisée de façon générale, sans avoir à développer une méthode d'encodage spécifique.

Dans ce chapitre, nous avons présenté le contexte historique et théorique du théorème de superposition, ce qui nous a permis de constater que les contributions mathématiques étaient nombreuses. Cependant, il y a peu d'applications proposées pour exploiter la décomposition obtenue par le théorème de superposition, ce qui peut s'expliquer par le fait que la construction de la décomposition reste un problème ardu. En effet, peu d'algorithmes permettent d'obtenir des fonctions monodimensionnelles pouvant être observées et manipulées. Pourtant, la décomposition proposée par le théorème de superposition est attractive, en proposant de décomposer un signal multidimensionnel, complexe, en plusieurs signaux monodimensionnels, simples. Notre travail a donc été d'étudier les algorithmes permettant de décomposer une image en fonctions monodimensionnelles suivant le théorème de superposition, puis de proposer des applications de cette décomposition au traitement d'image. Dans les chapitres suivants, nous présentons ainsi des approches originales (la modification et l'application d'algorithmes de décomposition des images par le théorème de superposition) pour répondre à des problèmes classiques de traitement d'images : la compression et la transmission progressive.

Chapitre 2

Implémentations

Dans le chapitre précédent, nous avons présenté le théorème de superposition de Kolmogorov et les différentes contributions proposées depuis sa présentation en 1957, notamment pour la construction des fonctions monodimensionnelles. Dans notre étude de la décomposition d'images par le théorème de superposition, nous avons tout d'abord implémenté l'algorithme proposé par Sprecher. Nous nous sommes intéressés en premier à cet algorithme parce que la décomposition proposée est fidèle au théorème, et son énoncé et son implémentation sont particulièrement accessibles. Nous avons ensuite implémenté l'algorithme d'Igelnik et Parikh pour pouvoir bénéficier d'une autre décomposition, plus ajustable, et disposant d'une représentation analytique des fonctions internes et externes. Très rapidement, nous avons observé que l'algorithme de Sprecher ne permettait pas une aussi grande flexibilité alors que celui d'Igelnik, notamment grâce à la représentation continue des fonctions internes et externes, ouvrait la porte à de plus amples adaptations. C'est la raison pour laquelle nos contributions présentées dans les chapitres 3 et 4 s'appuient sur des modifications de l'algorithme d'Igelnik. Nous présentons dans ce chapitre ces deux algorithmes et leur application à la décomposition d'images en niveaux de gris.

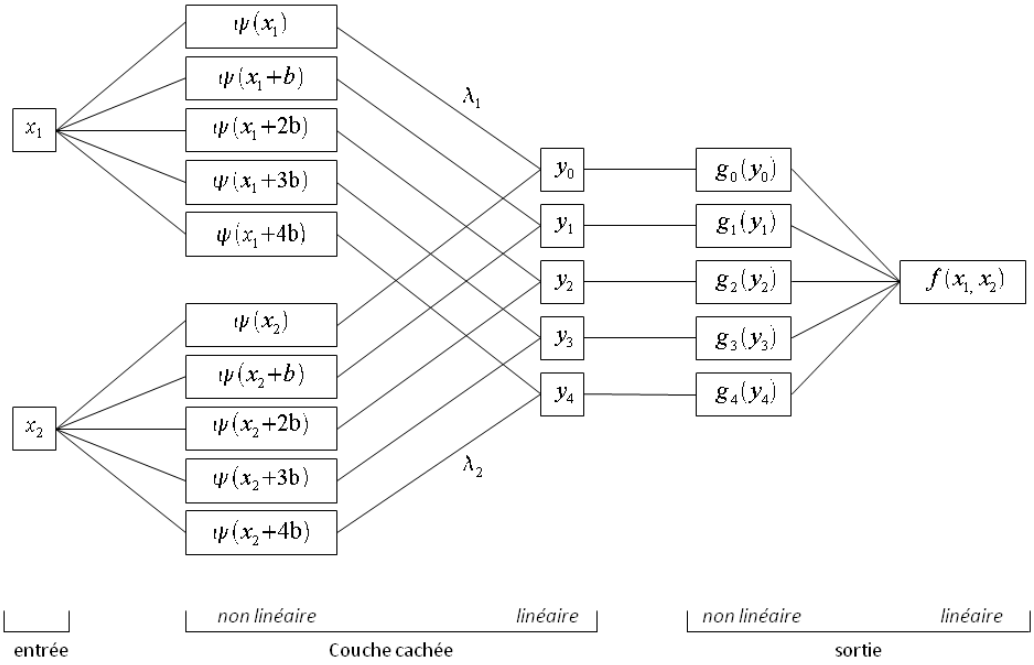


Figure 2.1 – Illustration de l'analogie entre le TSK et un réseau de neurones à une couche cachée.

2.1 L'algorithme de Sprecher

2.1.1 Introduction

L'algorithme de Sprecher s'appuie sur la formulation du théorème de Kolmogorov sous la forme suivante :

$$\begin{cases} f(x_1, ..., x_d) = \sum_{n=0}^{2d} g_n\Big(\xi(x_1 + bn, ..., x_d + bn)\Big) \\ \\ \xi(x_1 + bn, ..., x_d + bn) = \sum_{i=1}^{d} \lambda_i \psi(x_i + bn) \end{cases} \tag{2.1}$$

Cette formulation peut être interprétée comme une représentation de la fonction multi-dimensionnelle par un réseau de neurones à une couche cachée comme l'a montré Hecht-Nielsen et comme illustré en Figure 2.1.

Sprecher a proposé un algorithme exact, à une précision donnée, pour la construction des fonctions internes et externes dans [Sprecher, 1996] et [Sprecher, 1997], respectivement. Comme la fonction ψ initialement définie par Sprecher pour construire ξ est discontinue pour certaines valeurs d'entrée, nous utilisons la fonction ψ proposée par Braun et Griebel dans [Braun and Griebel, 2009] à la continuité et monotonie démontrées, qui corrige les problèmes de discontinuité aux extrémités des intervalles de la fonction ψ introduite par Sprecher. Le reste de l'algorithme de Sprecher n'est pas modifié, *i.e.* les fonctions monodimensionnelles internes et externes restent évaluées à une précision donnée pour n'importe quelle coordonnée du domaine de définition.

Définition 1 (Notations)

- d *est la dimension,* $d \geqslant 2$.
- m *est le nombre de couches de fonctions,* $m \geqslant 2d$.
- γ *est la base des variables* x_i, $\gamma \geqslant m + 2$.
- $b = \frac{1}{\gamma(\gamma-1)}$ *est la translation entre deux couches de pavage.*
- $\lambda_1 = 1$ *et pour* $2 \leqslant i \leqslant d$, $\lambda_i = \sum_{r=1}^{\infty} \frac{1}{\gamma^{(i-1)(d^r-1)/(d-1)}}$ *sont les coefficients de la combinaison linéaire qui sert d'argument aux fonctions* g_n.

2.1.2 Algorithme

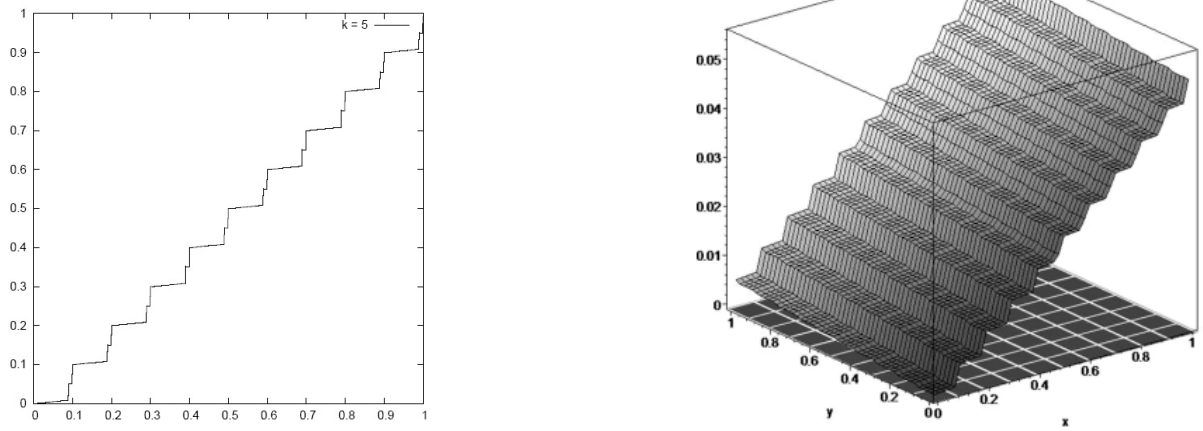
Les travaux de Sprecher portent sur la calculabilité de la fonction ξ et la définition de ψ. Plus précisément, la construction de ψ et la structure de l'algorithme sont basées sur la décomposition des nombre réels dans la base γ ; *i.e.* on peut écrire tout nombre décimal (noté d_k) entre $[0, 1]$ à k décimales sous la forme :

$$d_k = \sum_{r=1}^{k} i_r \gamma^{-r}, \text{ et } d_k^n = d_k + n \sum_{r=2}^{k} \gamma^{-r} \tag{2.2}$$

En utilisant les d_k définis en équation 2.2, Braun *et al.* définissent la fonction ψ par :

$$\psi_k(d_k) = \begin{cases} d_k & pour\ k = 1, \\ \psi_{k-1}(d_k - \frac{i_k}{\gamma^k}) + \frac{i_k}{\gamma^{\frac{d^k-1}{d-1}}} & pour\ k > 1 et\ i_k < \gamma - 1, \\ \frac{1}{2}(\psi_k(d_k - \frac{1}{\gamma^k}) + \psi_{k-1}(d_k + \frac{1}{\gamma^k})) & pour\ k > 1 et\ i_k = \gamma - 1. \end{cases} \quad (2.3)$$

La figure 2.2(a) représente le graphe de la fonction ψ sur l'intervalle $[0, 1]$. La fonction ξ est calculée par combinaison linéaire des réels λ_i et de la fonction ψ appliquée à chaque composante x_i de la valeur d'entrée. La figure 2.2(b) représente la fonction ξ sur l'espace $[0, 1]^2$.

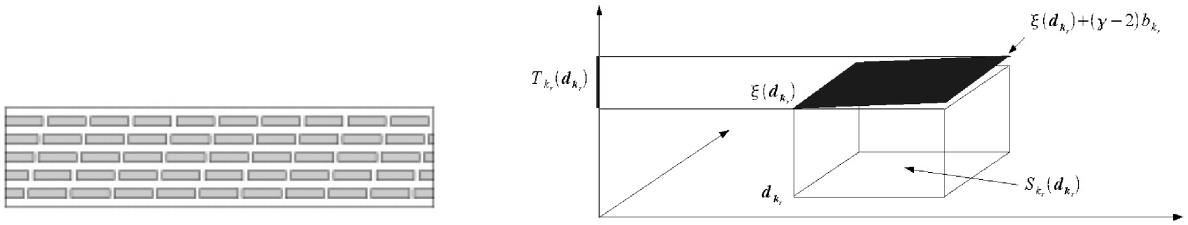


(a) (b)

Figure 2.2 – (a) Graphe de la fonction ψ pour la base $\gamma = 10$, d'après [Braun and Griebel, 2009]. (b) Fonction de hashage ξ pour $d = 2$ et $\gamma = 10$, d'après [Brattka, 2004].

Sprecher a montré que chacun des intervalles disjoints I était associé à un unique intervalle $\psi(I)$. Cette propriété de séparation est utilisée pour obtenir des intervalles définissant un pavage disjoint de $[0, 1]$. Cette méthode est étendue à un espace d-dimensionnel en effectuant le produit cartésien des intervalles I. Afin de couvrir tout l'espace, le pavage est translaté plusieurs fois d'une constante b, ce qui produit les différentes couches finales du pavage. On obtient ainsi $2d + 1$ couches, le pavage original constitué par des hypercubes disjoints ayant des images disjointes par la combinaison linéaire des fonctions ψ, et $2d$

pavages translatés d'une constante b le long de chaque dimension. La figure 2.3(a) représente une coupe du pavage complet pour un espace $2D$: on voit les $2d + 1 = 5$ pavages différents, décalés par b, qui se superposent.



(a) (b)

Figure 2.3 – (a) Vue en coupe d'un pavage dans le cas $2D$ avec $\gamma = 10$ (5 couches différentes), d'après [Brattka, 2004]. (b) La fonction ξ met en correspondance chaque pavé avec un intervalle T_k dans $[0, 1]$.

Dans le cas de la décomposition de fonctions bivariées, un hypercube est associé à un couple $\mathbf{d_{k_r}} = (d_{k_r 1}, d_{k_r 2})$. L'hypercube $S_{k_r}(\mathbf{d_{k_r}})$ est associé à un intervalle de valeur $T_{k_r}(\mathbf{d_{k_r}})$ par la fonction ξ, comme l'illustre la figure 2.3(b).

Une fois les fonctions internes ψ et ξ calculées, il reste à évaluer les fonctions externes g_n. Cependant, ces fonctions ne peuvent pas être déterminées directement et ne peuvent seulement être calculées que par un processus itératif. Sprecher propose la construction de r fonctions g_n^r dont la somme converge vers la fonction externe g_n. L'algorithme construit itérativement en trois étapes une fonction externe g_n^r. La fonction f_r définit l'erreur d'approximation, qui tend vers 0 quand r croît. L'algorithme est initialisé avec $f_0 = f$ et $r = 1$.

Première étape : détermination de la précision et construction du pavage

Avant tout calcul, nous déterminons la précision k_r pour l'étape courante. Pour deux coordonnées x_i et x_i' de deux vecteurs différents, appartenant à la même dimension i et espacées d'une distance donnée, la distance des images des deux vecteurs $\mathbf{x}$ et $\mathbf{x}'$ produites par f doit être inférieure au N^{e} de l'oscillation de f, *i.e.* :

$$si \left|x_i - x_i'\right| \leqslant \frac{1}{\gamma^{k_r}}, \left|f_{r-1}(x_1, ..., x_d) - f_{r-1}(x_1', ..., x_d')\right| \leqslant \epsilon \left\|f_{r-1}\right\|.$$

Une fois k_r déterminé, le pavage $d^n_{k_r1}, ..., d^n_{k_rd}$ est calculé par :

$$\forall i \in [\![1, d]\!],\ d^n_{k_ri} = d_{k_ri} + n\sum_{r=2}^{k_r} \frac{1}{\gamma^r}$$

Deuxième étape : évaluation des fonctions internes ψ et ξ

Pour n de 0 à m, déterminer $\psi(d^n_{k_r})$ et $\xi(d^n_{k_r1}, ..., d^n_{k_rd})$ avec les équations 2.2 et 2.3.

Troisième étape : détermination de l'erreur d'approximation

$\forall n \in [\![0, m]\!]$, calculer :

$$g^r_n \circ \xi(x_1 + bn, ..., x_d + bn) = \frac{1}{m+1} \sum_{d^n_{k_r1}, ..., d^n_{k_rd}} f_{r-1}\Big(d_{k_r1}, ..., d_{k_rd}\Big)\theta_{d^n_{k_r}}\Big(\xi(x_1 + bn, ..., x_d + bn)\Big)$$

où θ est définie dans la définition 2. Ensuite, calculer :

$$f_r(x_1, ..., x_d) = f(x_1, ..., x_d) - \sum_{n=0}^{m}\sum_{j=1}^{r} g^j_n \circ \xi(x_1 + bn, ..., x_d + bn)$$

A la fin de la r^{e} itération, le résultat de l'approximation de f est donné par la somme des $m+1$ couches de r fonctions g^r_n déterminées précédemment :

$$f \approx \sum_{n=0}^{m}\sum_{j=1}^{r} g^j_n \circ \xi(x_1 + bn, ..., x_d + bn).$$

Définition 2 *Définition de la fonction θ utilisée pour le calcul de la fonction g^r_n.*

$$\begin{aligned} &\theta_{d^n_k}(y_n) = \sigma\Big(\gamma^{\frac{d^{k+1}-1}{d-1}}\Big(y_n - \xi(d^n_k)\Big) + 1\Big) - \sigma\Big(\gamma^{\frac{d^{k+1}-1}{d-1}}\Big(y_n - \xi(d^n_k) - (\gamma-2)c_k\Big)\Big), \\ &\textit{avec } y_n = \xi(x_1 + bn, ..., x_d + bn) \end{aligned} \tag{2.4}$$

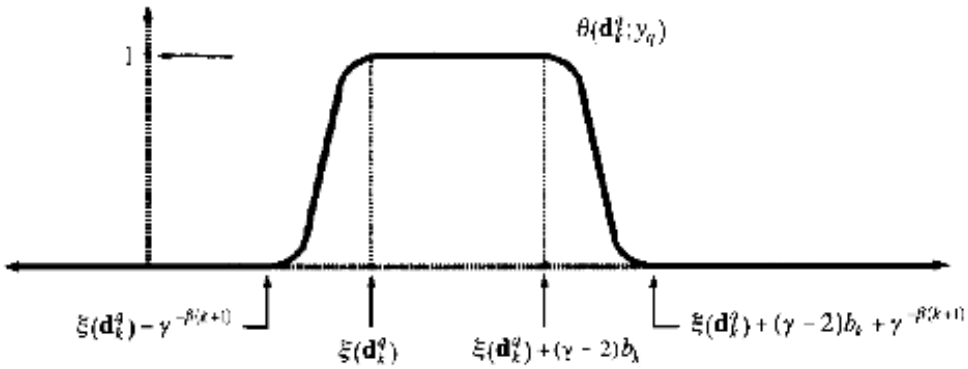


Figure 2.4 – Représentation de la fonction θ, d'après [Sprecher, 1997].

où σ *est une fonction continue définie arbitrairement et vérifiant :* $\begin{cases} \sigma(x) \equiv 0, \text{ pour } x \leqslant 0 \\ \sigma(x) \equiv 1, \text{ pour } x \geqslant 1 \end{cases}$,
et $c_k = \sum_{r=k+1}^{\infty} \frac{1}{\gamma^{\frac{d^r-1}{d-1}}} \sum_{p=1}^{n} \lambda_p$.

La Figure 2.4 illustre la fonction θ. Cette fonction a une valeur constante sur un hypercube, et agit comme une fonction de mélange pour les fonctions externes lorsque les coordonnées considérées sont situés entre deux hypercubes.

Remarque 1 *L'algorithme est très proche de l'énoncé du théorème, puisque la fonction interne est identique pour toute fonction* f *et que seules les fonctions externes sont adaptées à la fonction multidimensionnelle décomposée. Ces fonctions externes sont construites itérativement en prenant en compte les erreurs générées par les itérations antérieures. l'oscillation de la fonction d'erreur* f_r *tend vers 0 quand* r *croit, par conséquent, plus le nombre d'itérations est important, meilleure est l'approximation de la fonction* f *par les fonctions* g_n.

2.1.3 Application de la décomposition sur des images

Nous présentons ici les résultats de la décomposition appliquée à des images en niveaux de gris, pouvant être vues comme des fonctions bidimensionnelles $f(x, y) = I(x, y)$. La figure 2.5 représente deux couches issues de l'approximation obtenue après une et deux

itérations de l'algorithme de Sprecher. La somme de toutes ces couches donne l'approximation de l'image originale par l'algorithme. Les pixels blancs visibles sur les images 2.5(b) et 2.5(e) correspondent aux valeurs négatives de la fonction externe. La figure 2.6 montre deux reconstructions d'une même image de départ après une et deux itérations de l'algorithme. Les couches issues de la décomposition sont très similaires : chaque couche correspond à une fraction d'un sous-échantillon de la fonction f, légèrement translatée par un multiple de la valeur b. Dans le cas de fonctions bivariées, on observe que la reconstruction tend très rapidement vers l'image originale. On observe d'ailleurs très peu de différence entre la première et la seconde approximation sur les figures 2.6(b) et 2.6(c).

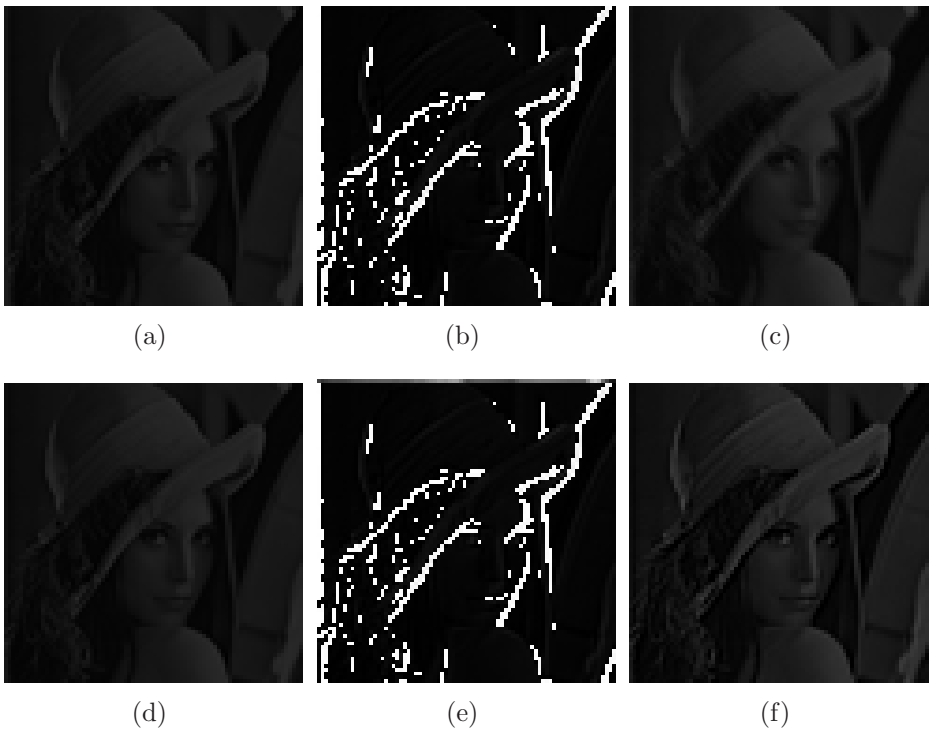


Figure 2.5 – (a) et (b) Première couche ($n = 0$) après une et deux itérations ($r = 1, r = 2$) respectivement. (c) Somme de (a) et (b), reconstruction partielle pour la première couche. (d) et (e) La dernière couche ($n = 5$) après une et deux itérations ($r = 1, r = 2$) respectivement. (f) Somme de (d) et (e), reconstruction partielle pour la dernière couche.

2.1.4 Conclusion

L'algorithme proposé par Sprecher construit des fonctions internes et externes proches de la formulation du théorème : pour toute fonction f à décomposer, une fonction interne

(a) (b) (c)

Figure 2.6 – (a) Image originale. (b) et (c) Reconstruction après une et deux itérations respectivement.

unique est utilisée, et seules les fonctions externes sont adaptées. Nous avons ensuite appliqué cet algorithme à la décomposition d'images en niveau de gris, considérées comme la représentation discrète d'une fonction bivariée. Nos résultats montrent que l'algorithme converge rapidement vers l'image originale. Cependant, le principal inconvénient de l'algorithme de Sprecher réside dans sa rigidité : les fonctions internes ψ ne peuvent pas être modifiées sans modifier la construction des fonctions externes. Or, dans la perspective d'étudier les possibles applications de la décomposition proposée par Kolmogorov, un algorithme flexible et facilement modifiable offrira davantage de possibilités d'adaptation. C'est pourquoi nous avons ensuite étudié l'algorithme d'Igelnik et Parikh, qui ont proposé dans [Igelnik and Parikh, 2003] un algorithme pour approcher les fonctions internes et externes, et qui permet en outre de disposer d'une représentation paramétrique continue de ces fonctions. Il permet notamment de faire varier le nombre de couches de pavage, de générer plusieurs fonctions internes, et de mettre à profit la continuité des fonctions pour différentes tâches d'interpolation et/ou de simplification. De plus, les approximations et définitions des fonctions monodimensionnelles sont indépendantes et modifiables en intervenant seulement localement dans l'algorithme.

2.2 L'algorithme d'Igelnik et Parikh

2.2.1 Introduction

L'algorithme d'approximation proposé par Igelnik et Parikh dans [Igelnik and Parikh, 2003] appelé Kolmogorov Spline Network (KSN), propose une décomposition qui ne représente plus qu'une approximation de la fonction multidimensionnelle f, en échange d'une plus grande flexibilité dans le nombre de fonctions monodimensionnelles et de leur construction. Igelnik et Parikh ont conservé le principe de pavage en utilisant plusieurs fonctions internes (une par couche), et le nombre de couples fonctions internes/externes (associés à chaque couche de pavage) est variable, la convergence de l'algorithme étant obtenue en augmentant ce nombre (voir démonstration dans [Igelnik and Parikh, 2003]). L'équation 1 est remplacée par l'approximation suivante :

$$\begin{cases} f(x_1, ..., x_d) \simeq \sum_{n=1}^{N} a_n g_n\Big(\xi_n(x_1, ..., x_d)\Big) \\ \\ \xi_n(x_1, ..., x_d) = \sum_{i=1}^{d} \lambda_i \psi_{ni}(x_i) \end{cases} \tag{2.5}$$

Bien qu'une discussion sur les principales différences entre l'algorithme de Sprecher et celui d'Igelnik soit proposée en section 2.2.6, nous pouvons déjà observer les points suivants :

- ψ_{ni} a deux indices i et n : les fonctions internes ψ_{ni}, indépendantes de la fonction f, sont générées aléatoirement pour chaque dimension i et chaque couche n.
- les fonctions ψ_{ni} sont échantillonnées régulièrement puis interpolées par des splines cubiques.
- La somme des fonctions externes g_n est pondérée par des coefficients a_n.

2.2.2 Algorithme

La première étape consiste en la définition d'un pavage disjoint sur le domaine de définition $[0, 1]^d$ de la fonction multidimensionnelle f. Afin de couvrir tout l'espace, plusieurs

autres couches de pavages sont générées en translatant la première couche, comme illustré sur la Figure 2.8(a). Pour une couche de pavage d'indice n donnée, d fonctions internes ψ_{ni} sont générées aléatoirement (une par dimension), indépendamment de la fonction f. Les fonctions ψ_{ni} sont échantillonnées par m points qui sont ensuite interpolés par des splines cubiques. La combinaison convexe de ces fonctions internes ψ_{ni} avec les réels λ_i constitue l'argument de la fonction externe g_n. Enfin, la fonction externe g_n associée à cette couche de pavage est construite en utilisant les valeurs de la fonction f aux centres des hypercubes. Un d+1-uplet composé de d fonctions internes et d'une fonction externe $\left(\psi_1, ..., \psi_d, g\right)_n$ est donc associé avec chaque couche de pavage n. L'ensemble constitué de tous les paramètres définissant les fonctions internes et externes est appelé un réseau. Les approximations partielles obtenues avec chaque couche du réseau sont sommées pour approcher la fonction f. Pour optimiser la convergence du réseau vers la fonction f, chaque couche est pondérée par des coefficients a_n et les paramètres des fonctions internes (certains points d'échantillonage) sont optimisés. La Figure 2.7 illustre ces différentes étapes pour un réseau constitué de cinq couches de pavages.

Le pavage est constitué d'hypercubes H_n, obtenus en calculant le produit cartésien des intervalles $I_n(j)$, définis ci-après.

Définition 3

$$\forall n \in [\![1, N]\!], j \in \left\{\mathbb{N} \cup \{-1\}\right\},\ I_n(j) = \left[(n-1)\delta + (N+1)j\delta, (n-1)\delta + (N+1)j\delta + N\delta\right]$$

Par construction, les intervalles $I_n(j)$ sont de longueur $N\delta$. δ, qui correspond à la distance entre deux intervalles consécutifs, est tel que les oscillations de la fonction f sur chaque hypercube H_n soient inférieures à $\frac{1}{N}$. Pour l'application au traitement d'images, nous étudions le cas de fonctions bidimensionnelles, donc les hypercubes H_n sont des carrés. Chaque intervalle a pour indice j, qui identifie sa position sur une dimension, et dont les valeurs sont définies telles que les intervalles $I_n(j)$ intersectent l'intervalle $[0, 1]$, comme illustré sur la Figure 2.8(b). D'après cette condition, on observe que pour des valeurs de N

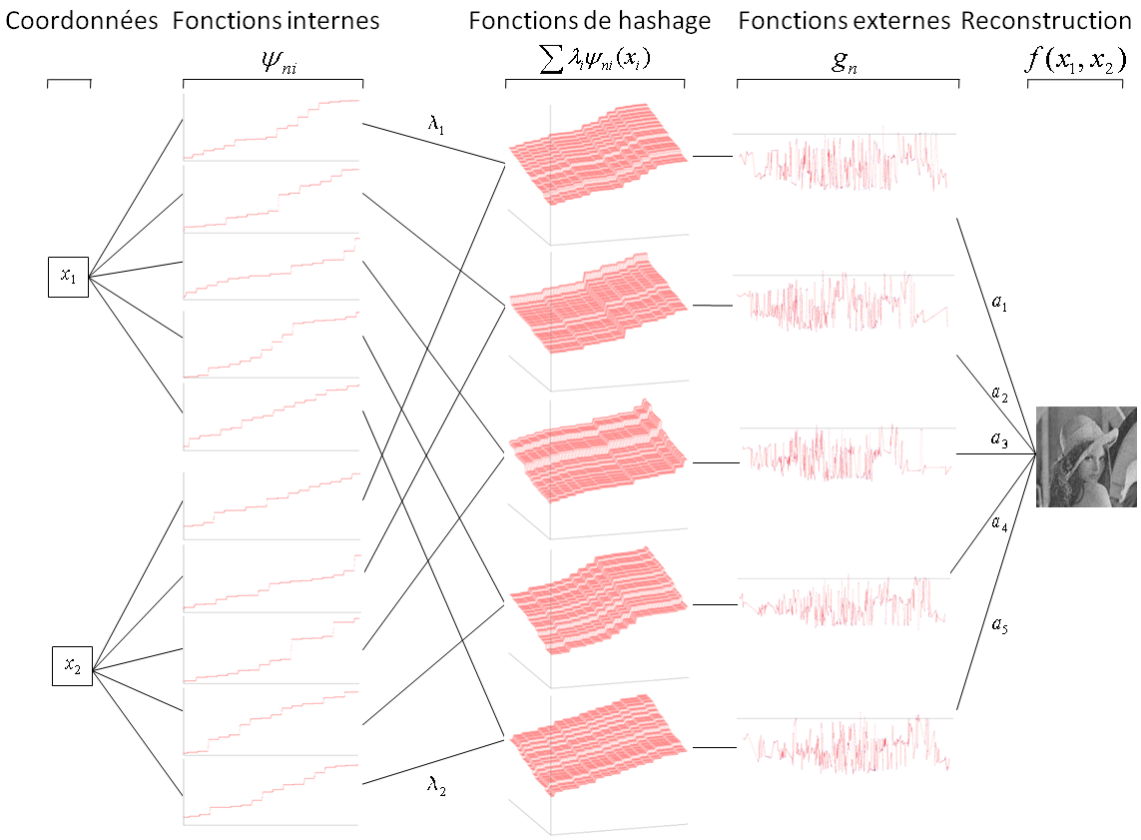


Figure 2.7 – Réseau constitué de cinq couches de pavage. Première étape : à chaque dimension de coordonnées x_i et pour chaque couche de pavage n est associée une fonction interne ψ. Deuxième étape : pour chaque couche n donnée, la combinaison convexe des fonctions internes est utilisée comme argument de la fonction de hashage ξ. Troisième étape : Les fonctions externes sont construites à partir des images de la fonction ξ associée et de la fonction f. Quatrième étape : les approximations obtenues avec chaque couche sont sommées, avec une pondération a_n, pour reconstruire l'image.

usuelles, le premier intervalle est obtenu avec $j = 0$ ou $j = -1$, selon la couche considérée :

$$\forall n \in [\![1, N]\!],\ (n-1)\delta + (N+1)j\delta + N\delta \geqslant 0 \ \Leftrightarrow\ j \geqslant \frac{1-n-N}{N+1} \tag{2.6}$$

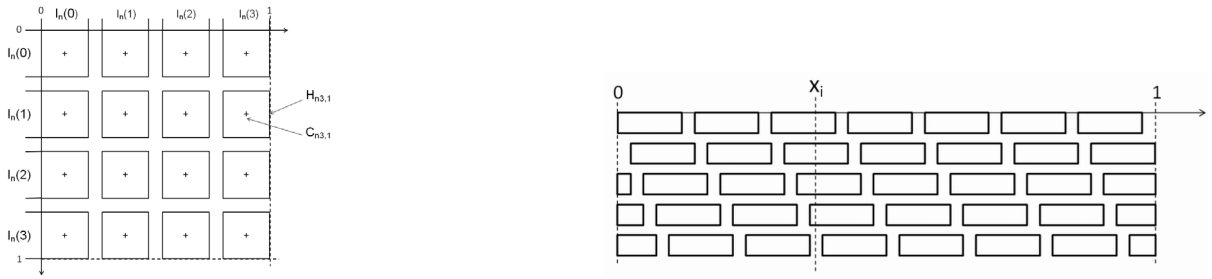
On remarque que l'indice j est fonction de n, l'indice de la couche considérée, et que cette fonction est strictement décroissante. Autrement dit, la valeur minimum de j est obtenue pour $n = N$. $\frac{1-n-N}{1+N}$ devient $\frac{1-2N}{1+N} = \frac{3}{N+1} - 2 > -2$ puisque $N > -1$.

La taille des pavés est constante pour chaque couche, et toutes les couches de pavage sont translatées les unes par rapport aux autres, afin d'assurer que tout point x de $[0, 1]^d$ est

couvert par au moins $N-1$ hypercubes :

$$\forall x \in [0,1]^d,\ si\ \exists i \in [\![1,d]\!]\ et\ \nexists n_p \in [\![1,N]\!],\ x_i \in I_{n_p}\ alors\ \forall n \in [\![1,N]\!] \backslash \{n_p\},\ x_i \in I_n$$

Cette construction est illustrée par la Figure 2.8(a).



(a) (b)

Figure 2.8 – (a) Produit cartésien des intervalles I_n définissant un pavage disjoint d'hypercubes H_n et de centres C_n. (b) Vue en coupe de la superposition des couches de pavages disjoints translatées.

2.2.3 Construction des fonctions internes ψ_{ni}

Sur chaque intervalle I_n, les fonctions internes ψ_{ni} ont une valeur constante y_{nij} attribuée aléatoirement, mais en vérifiant que lors du calcul de la combinaison linéaire dans la fonction ξ_n, les réels obtenus soient uniques. Les valeurs y_{nij} définissent les valeurs des paliers de la fonction ψ_{ni}, comme l'illustre la figure 2.9(a). Les intervalles I_n sont disjoints, donc les paliers le sont également. Pour construire une fonction interne continue, des points d'échantillonnage sont générés en utilisant les valeurs des paliers y_{nij} et des valeurs aléatoires entre deux paliers consécutifs. Ces points sont ensuite interpolés par des splines cubiques.

Plus précisément, chaque fonction ψ_{ni} est définie comme suit :

- Générer un ensemble de j nombres distincts y_{nij}, compris entre Δ et $1-\Delta$, $0 < \Delta < 1$. Δ est choisi tel que l'amplitude des oscillations de la spline cubique interpolante de ψ_{ni} sur l'intervalle δ soit inférieure à Δ.
- Les nombres réels y_{nij} sont triés, *i.e.* : $y_{nij} < y_{nij+1}$. L'image de l'intervalle $I_n(j)$ par la fonction ψ_{ni} est y_{nij}.

– Les valeurs de j sont détaillées dans la définition 3. $j \in \{\mathbb{N} \cup \{-1\}\}$.

Ensuite, ψ_{ni} est échantillonnée régulièrement. Les points d'échantillonnage obtenus sont de deux types : les points situées sur les plateaux (correspondant aux images des intervalles $I_n(j)$) appartiennent à un ensemble noté $\mathscr{M}$; et les points situés entre les plateaux, *i.e.* dont l'abscisse est entre deux intervalles consécutifs $I_n(j)$ et $I_n(j+1)$, appartiennent a un ensemble noté $\mathscr{M}'$, et ont une ordonnée aléatoire comprise dans l'intervalle $]y_{nij}, y_{nij+1}[$, comme illustré sur la Figure 2.9(b). Ces points sont ensuite interpolés par une spline cubique pour obtenir une fonction croissante continue, comme l'illustre la Figure 2.9(c). L'algorithme utilisé pour la construction de splines cubiques passant par des points régulièrement espacés est présenté dans [Moon, 2001]. L'ordre 3 de ces splines assure la convergence de l'algorithme, comme démontré dans [Igelnik and Parikh, 2003]. Les coordonnées des points de $\mathscr{M}'$ sont optimisées par une approche stochastique pendant la construction du réseau décrite dans [Igelnik and Parikh, 2003], ce qui permet d'améliorer la précision de la reconstruction et la convergence vers la fonction f.

Une fois que les fonctions ψ_{ni} sont construites, l'argument des fonctions externes g_n, *i.e.* les fonctions ξ_n, peut être calculé. Sur les hypercubes $H_{nj_1,\dots,j_d}$, cette valeur est constante et on a :

$$p_{nj_1,\dots,j_d} = \sum_{i=1}^{d} \lambda_i y_{nij_i}$$

Lors de la construction des fonctions ψ_{ni}, chaque nombre aléatoire y_{nij_i} vérifie que les valeurs générées $p_{nj_1,\dots,j_d}$ sont toutes uniques, $\forall i \in [\![1,d]\!], \forall n \in [\![1,N]\!], \forall j \in \{\mathbb{N} \cup \{-1\}\}$. Les nombres réels λ_i sont choisis linéairement indépendants, strictement positifs, et tels que $\sum_{i=1}^{d} \lambda_i \leqslant 1$. La figure 2.10 illustre la construction de la fonction ξ_n, obtenue par combinaison linéaire des constantes λ_i et des fonctions internes ψ_{ni}. L'intervalle contenant les images des fonctions ξ_n (*i.e.* l'intervalle de définition de la fonction externe associée g_n) est différent pour chaque couche n :

$$\mathscr{D}_{g_n} = [min_{j_i}(p_{nj_1,\dots,j_d}), max_{j_i}(p_{nj_1,\dots,j_d})] \subset [\Delta, 1-\Delta]$$

Le vecteur des coordonnées du centre de l'hypercube $H_{nj_1,\dots,j_d}$ est noté $C_{nj_1,\dots,j_d}$.

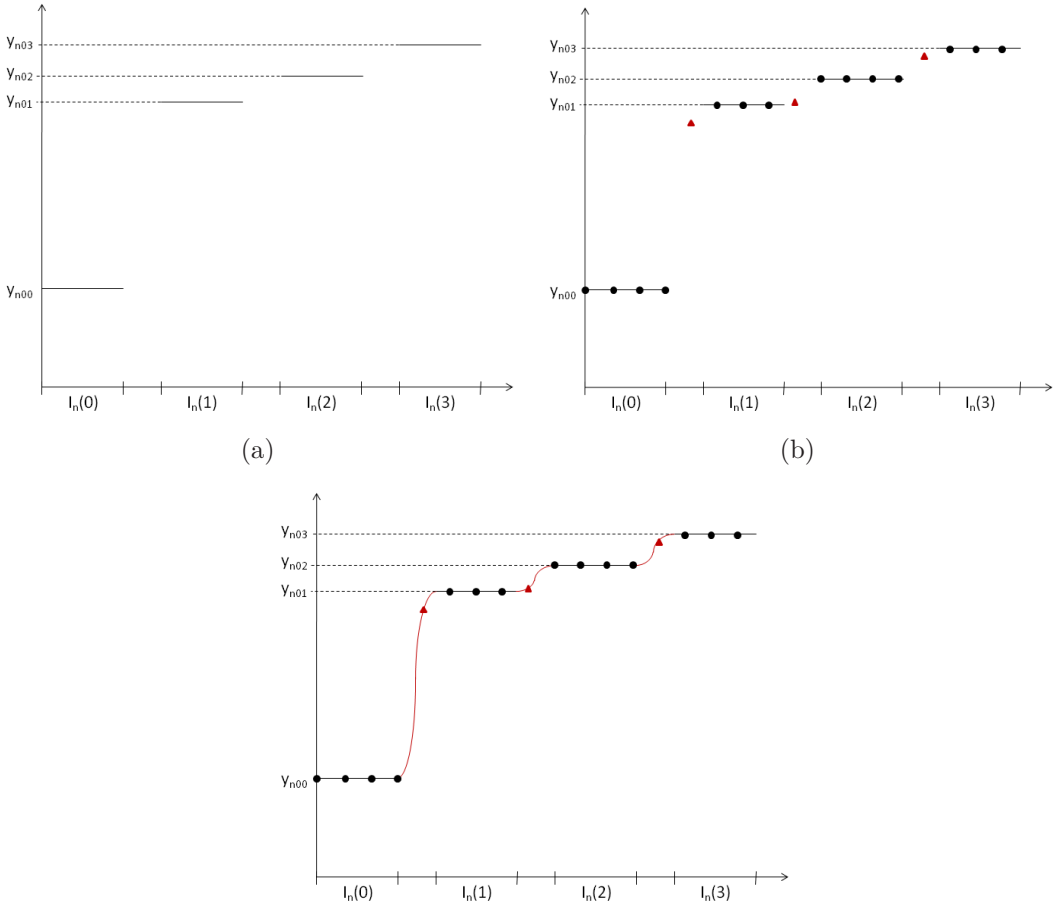


Figure 2.9 – (a) Répartition aléatoire des hauteurs des paliers de la fonction ψ_{ni}. (b) Echantillonnage régulier de la fonction ψ_{ni} : l'ordonnée des points se trouvant entre deux paliers ($\in \mathscr{M}'$) est attribuée aléatoirement et bornée par les hauteurs des paliers adjacents. (c) Interpolation des points d'échantillonage par des splines cubiques.

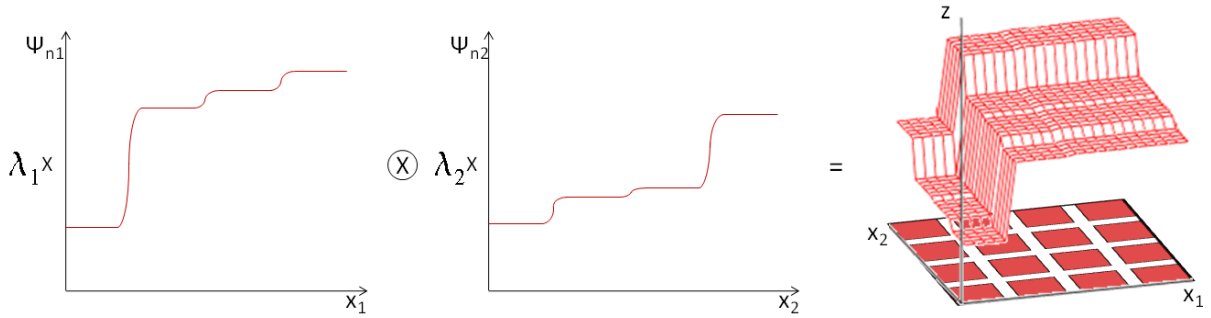


Figure 2.10 – Exemple d'une fonction ξ_n, construite par combinaison linéaire des constantes λ_i et des fonctions internes ψ_{ni}.

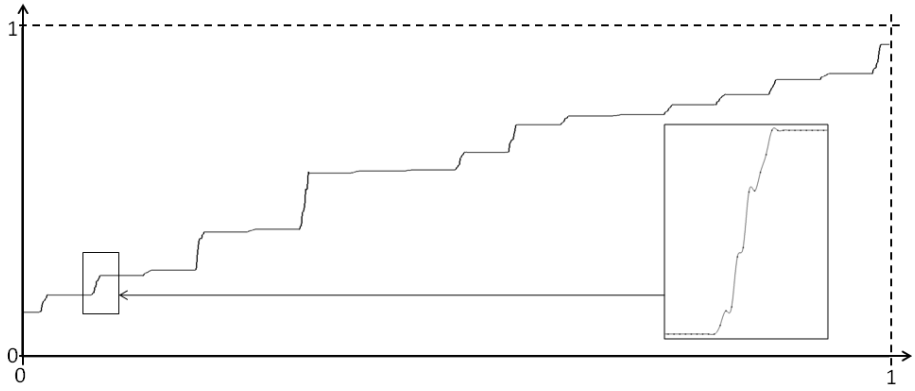


Figure 2.11 – Exemple d'une fonction ψ_{ni} échantillonnée par 500 points, ensuite interpolés par une spline cubique.

2.2.4 Construction des fonctions externes g_n

Une fonction externe g_n est définie pour chaque couche de pavage d'indice n. Tout d'abord, un ensemble de points $\mathscr{S}$ est créé. L'abscisse de chaque point correspond à l'image de la fonction ξ_n associée, *i.e.* les réels $p_{nj_1,\ldots,j_d}$ qui identifient de façon unique l'hypercube $H_{nj_1,\ldots,j_d}$. L'ordonnée de chaque point correspond à l'image de la fonction f au centre de l'hypercube $C_{nj_1,\ldots,j_d}$. Enfin, les points de l'ensemble $\mathscr{S}$ sont reliés par des splines de degré neuf et des droites pour obtenir une représentation paramétrique continue de la fonction externe.

Plus précisément, les fonctions g_n sont construites comme suit :

– Pour chaque réel $t = p_{nj_1,\ldots,j_d}$, l'ordonnée $g_n(t)$ est égale au N^{e} de la valeur de la fonction f au centre de l'hypercube $H_{nj_1,\ldots,j_d}$:

$$g_n(p_{nj_1,\ldots,j_d}) = \frac{f((x_{C_{nj_1,\ldots,j_d}}, y_{C_{nj_1,\ldots,j_d}}))}{N} \tag{2.7}$$

Ces points sont notés $A_k = (t, g_n(t)), k \in \mathbb{N}$. La figure 2.12 illustre la création de cet ensemble de points sur un exemple. Une image 6×6 pixels est la fonction bivariée à décomposer, et la fonction ξ précédemment générée est utilisée pour construire l'ensemble $\mathscr{S}$.

– La définition des fonctions g_n est étendue pour tous les $t \in \mathscr{D}_{g_n}$ comme suit :
  – Deux points B_k et B'_k sont placés aléatoirement dans le voisinage de A_k, tels que $x_{B_k} < t < x_{B'_k}$. Le placement des points B_k et B'_k dans le voisinage de A_k doit conserver l'ordre des points : $..., B'_{k-1}, B_k, A_k, B'_k, B_{k+1}, ...$; *i.e.* , la distance entre B_k et A_k ou A_k et B'_k doit être strictement inférieure à la moitié de la distance entre deux points consécutifs A_k et A_{k+1}.
  – Les points B'_k et B_{k+1} sont reliés par une droite de pente r.
  – Les points A_k et B'_k sont reliés par une spline de degré neuf, notée s, telle que :
    – $s(t_{A_k}) = g_n(t_{A_k})$,
    – $s(t_{B'_k}) = g_n(t_{B'_k})$,
    – $s'(t_{B'_k}) = r$, et
    – $s^{(2)}(t_{B'_k}) = s^{(3)}(t_{B'_k}) = s^{(4)}(t_{B'_k}) = 0$.

    Les points B_k et A_k sont reliés de façon similaire par une spline de degré neuf. Les conditions de raccordement aux points A_k des deux splines de degré neuf donnent les conditions restantes à la définition des splines.

  Cette construction, illustrée par la Figure 2.13, assure la continuité de la fonction externe g_n et la convergence de l'approximation vers la fonction f comme démontré dans [Igelnik and Parikh, 2003].

On peut noter que les fonctions monodimensionnelles, bien que continues, sont exactement définies par des ensembles de points finis : soit les points de l'ensemble $\mathscr{S}$ pour les fonctions externes, soit les points des ensembles $\mathscr{M}$ et $\mathscr{M}'$ pour les fonctions internes. Cela permet l'utilisation de fonctions continues sans problème de stockage ou d'échantillonnage : les courbes connectant ces points peuvent être calculées pour n'importe quelles coordonnées, puisqu'il s'agit de splines ou de droites.

2.2.5 Optimisation stochastique du réseau

On peut remarquer qu'entre deux pavés consécutifs, les valeurs de la fonction interne ne dépendent pas de la fonction f. Pour diminuer l'erreur d'approximation du réseau, plusieurs paramètres définissant les fonctions monodimensionnelles doivent être optimisés selon une méthode stochastique (appelée "ensemble approach", voir [Igelnik et al., 1999] et [Igelnik et al., 2001]). Ces paramètres correspondent aux poids a_n associés aux couches et aux positions des points d'échantillonnage de $\mathscr{M}'$, *i.e.* les points d'échantillonnage des fonctions internes ψ_{ni} situés entre deux paliers consécutifs. Pour optimiser le réseau, trois ensembles de points sont construits à partir des valeurs de f : un ensemble d'entrainement $\mathscr{D}_T$, un ensemble de généralisation $\mathscr{D}_G$, et un ensemble de validation $\mathscr{D}_V$.

Les N couches qui constituent le réseau sont construites successivement. Pour ajouter une nouvelle couche au réseau, K couches candidates sont générées avec les même paliers y_{nij} et ajoutées au réseau existant, ce qui définit K nouveaux réseaux potentiels. Ainsi, deux couches sont différentes car l'ensemble des points choisis aléatoirement $\mathscr{M}'$ situés entre deux intervalles consécutifs est différent. La couche contenue dans le réseau donnant la plus petite erreur quadratique moyenne pour l'ensemble des points de l'ensemble de généralisation $\mathscr{D}_G$ est conservée. Les poids a_n sont calculés pour minimiser la différence entre l'approximation donnée par le réseau et les valeurs de la fonction f pour les points d'entrainement de l'ensemble $\mathscr{D}_T$. L'algorithme est itéré jusqu'à ce que N couches soient construites. L'erreur de validation du réseau final est calculée en utilisant l'ensemble de validation $\mathscr{D}_V$, *i.e.* , en calculant les approximations fournies par le réseau pour tout point de $\mathscr{D}_V$. Le diagramme de la construction complète du réseau est présenté Figure 2.14.

Pour calculer les coefficients a_n ,*i.e.* les poids de chaque couche, la différence entre f et son approximation $\tilde{f}$ donnée par le réseau doit être minimisée :

$$\|Q_n a_n - t\|, \text{ en notant } t = \begin{bmatrix} f(x_{1,1}, ..., x_{d,1}) \\ ... \\ f(x_{1,P}, ..., x_{d,P}) \end{bmatrix} \tag{2.8}$$

où Q_n est une matrice de vecteurs colonnes $q_k, k \in [\![0, n]\!]$ qui correspondent à l'approximation $(\tilde{f})$ de la k^e couche pour l'ensemble de points $\Big((x_{1,1}, ..., x_{d,1}), ..., (x_{1,P}, ..., x_{d,P})\Big)$ de $\mathscr{D}_T$:

$$Q_n = \left[\begin{bmatrix} \tilde{f_0}(x_{1,1}, ...x_{d,1}) \\ ... \\ \tilde{f_0}(x_{1,P}, ...x_{d,P}) \end{bmatrix}, ..., \begin{bmatrix} \tilde{f_n}(x_{1,1}, ...x_{d,1}) \\ ... \\ \tilde{f_n}(x_{1,P}, ...x_{d,P}) \end{bmatrix} \right]$$

Un algorithme pour le calcul de la pseudo-inverse $Q_n^{-1}t = a_n$ tenant compte des spécificités de l'algorithme (comme par exemple la construction de la matrice Q_n ligne par ligne) est proposée par Igelnik dans [Igelnik et al., 1999]. Le coefficient a_l du vecteur colonne $(a_0, ..., a_n)^T$ correspond au poids associé à la couche l, $l \in [\![0, n]\!]$. On peut noter que pour initialiser l'algorithme de calcul de la pseudo-inverse, la matrice Q_n est initialisée avec une première ligne constante, ce qui se traduit par un poids a_0 supplémentaire, qui n'est associé à aucune couche construite à partir d'un pavage dans le réseau.

2.2.6 Comparaison des approches de Sprecher et Igelnik

La principale différence se situe au niveau du principe même de l'algorithme : pour Sprecher, l'algorithme permet de calculer la valeur exacte des fonctions monodimensionnelles à une précision donnée sur tout le domaine de définition $[0, 1]$. Néanmoins, de par la nature récursive de l'algorithme, il n'est pas possible d'obtenir une représentation analytique, complète et détaillée de la fonction g ni de séparer la construction de la fonction interne et de la fonction externe. Dans le cadre d'une étude des applications possibles du schéma de décomposition, en plus de pouvoir étudier les images issues de la décomposition, il est souhaitable de pouvoir extraire les fonctions monodimensionnelles afin de les étudier ou de les traiter. En effet, en traitement d'images, de nombreux outils sont issus du traitement de signal $1D$. Deux approches peuvent alors être considérées : soit les outils $1D$ peuvent être étendus à la $2D$, ou soit l'image est convertie en signal $1D$, généralement par concaténation des lignes et/ou des colonnes. Dans ce deuxième cas, la décomposition proposée

par le TSK est radicalement différente et offre des perspectives intéressantes, par exemple pour l'application d'outils $1D$ qui ne peuvent être étendus en $2D$.

Sprecher *et al.* ont montré dans [Sprecher and Draghici, 2002] qu'une courbe de remplissage de l'espace (fractale) peut être obtenue à partir d'un mappage de la fonction ξ. La Figure 2.15 représente cette courbe de remplissage. Elle relie tout vecteur de l'espace $[0,1]^d$ dont les coordonnées sont des d_k (tels que définit dans la définition 2.2), d'où la nature fractale de la courbe, qui se répète à chaque niveau de précision k. Autrement dit, pour un espace $2D$, chaque couple $(d_{k_r x}, d_{k_r y})$ ($k_r = 2$ dans la Figure 2.15) est associé à une valeur réelle dans $[0,1]$. En parcourant ces réels dans l'ordre croissant, on obtient alors la courbe de remplissage de l'espace.

Cette propriété est très séduisante dans le cas de la décomposition d'une image : grâce à la décomposition du TSK, il serait possible de convertir une image en signaux $1D$ tout en contrôlant l'ordre de parcourt des pixels, sans tenir compte des relations de voisinage initiales. Cela nécessiterait néanmoins de modifier en profondeur l'algorithme de Sprecher, puisque dans sa version originale, la construction de la fonction interne, qui contrôle le balayage, est indépendante de la fonction f (l'image à décomposer) et imbriquée avec la construction des fonctions externes.

L'approche proposée par Igelnik et Parikh, offre la flexibilité requise pour ces modifications, mais permet seulement une approximation des fonctions monodimensionnelles. Dans cette approche, il devient possible de modifier le nombre de couches, la taille du pavage, l'orientation du balayage et la construction des fonctions monodimensionnelles ; sans augmenter la complexité des calculs. Nos observations montrent que la complexité est principalement liée à la densité du pavage. La densité du pavage détermine le nombre de points nécessaires à l'échantillonnage des fonctions internes, ce qui détermine la durée de construction d'une couche. La densité du pavage est, dans le pire des cas, égale à la résolution de l'image. Le nombre de couches total N ou le nombre de couches candidates K font ensuite augmenter ce temps linéairement. Pour une image de résolution $R \times R$ pixels, la complexité de l'algorithme est donc de l'ordre de $O(N \times K \times R^d)$. D'un point de vue pratique, la décomposition d'une image de résolution 200×200 pixels par un réseau

de 10 couches avec un pavage de 120×120 pavés par couche requiert environ 2 minutes avec un processeur à 3 GHz. Il faut noter que l'algorithme, programmé en C/C++, n'a pas fait l'objet d'optimisation poussée, ou de parallélisation des calculs (multithreads ou exécution sur GPU).

Il est possible de construire des courbes de balayages pour les fonctions d'Igelnik. Il s'agit en fait d'une courbe de remplissage, mais qui est définie pour un espace discret seulement. Une courbe différente est obtenue pour chaque couche puisqu'une nouvelle fonction ξ_n est construite pour chaque couche. Par ailleurs, chaque fonction ξ_n est constante sur chaque pavé, ce qui introduit une indétermination dans les courbes de remplissage : différents couples $(d_{k_r x}, d_{k_r y})$ d'un même hypercube ayant la même image par la fonction ξ_n. Les Figure 2.16 et Figure 2.17 représentent deux vues d'une même fonction de remplissage. Dans la vue $2D$, les carrés contenant les courbes bleues correspondent aux paliers de la fonction ξ_n.

De plus, dans le KSN, les constantes λ_i peuvent être choisie librement. Ces constantes permettent de contrôler globalement l'orientation du balayage. La Figure 2.18 présente trois fonctions de balayage utilisant différentes valeurs de λ_1, λ_2 : (a) est obtenue avec un poids presque égal pour chaque dimension ($\lambda_1 \simeq 0.5, \lambda_2 = 0.4$), (b) avec $\lambda_1 \simeq 0.95, \lambda_2 = 0.04$, et (c) avec $\lambda_1 \simeq 0.95$ et $\lambda_2 = 0.00004$, ce qui se rapproche du balayage ligne par ligne classique. (d)(e)(f) représentent un exemple de fonctions externes obtenues pour ces balayages : les λ_i ne modifient pas seulement le balayage, mais aussi la répartition spatiale dans les fonctions externes. Par contre, le contrôle local du balayage reste un problème ouvert. La construction des fonctions ψ_{ni} est indépendante de l'image, puisque les paliers sont générés aléatoirement. Autrement dit, toutes les informations de l'image sont contenues dans les fonctions externes. Pour répartir les informations de l'image, il s'agit donc de lier la hauteur des paliers aux valeurs des pixels. Toutefois, s'il est possible de construire la fonction ξ_n pour un balayage donné, comment déterminer ensuite les fonctions internes et les λ_i permettant d'obtenir cette fonction ξ_n ? De même, comment déterminer la quantité de données devant faire partie des fonctions internes et des fonctions externes ? En effet, on peut imaginer un balayage de l'image par ordre croissant de niveaux de gris, où la fonction externe est alors une droite, cette fois indépendante de la fonction f. Evidemment, cette

dernière question dépend de l'application envisagée. Dans les applications présentées dans ce mémoire, nous supposons que les fonctions internes sont indépendantes de la fonction f.

2.2.7 Application à la décomposition d'image

Comme pour l'algorithme de Sprecher, nous présentons les résultats de la décomposition d'images en niveaux de gris. La Figure 2.19 présente les reconstructions partielles obtenues pour chacune des six couches composant le réseau utilisé dans cet exemple. La somme pondérée de ces couches donne l'approximation de l'image originale par l'algorithme, cf. Figure 2.19(f). La précision de la reconstruction dépend de la taille du pavage. Dans cet exemple, un pavé couvre en moyenne 4 pixels, mais tous les pixels de l'image sont utilisés pour construire le réseau à cause de la translation du pavage.

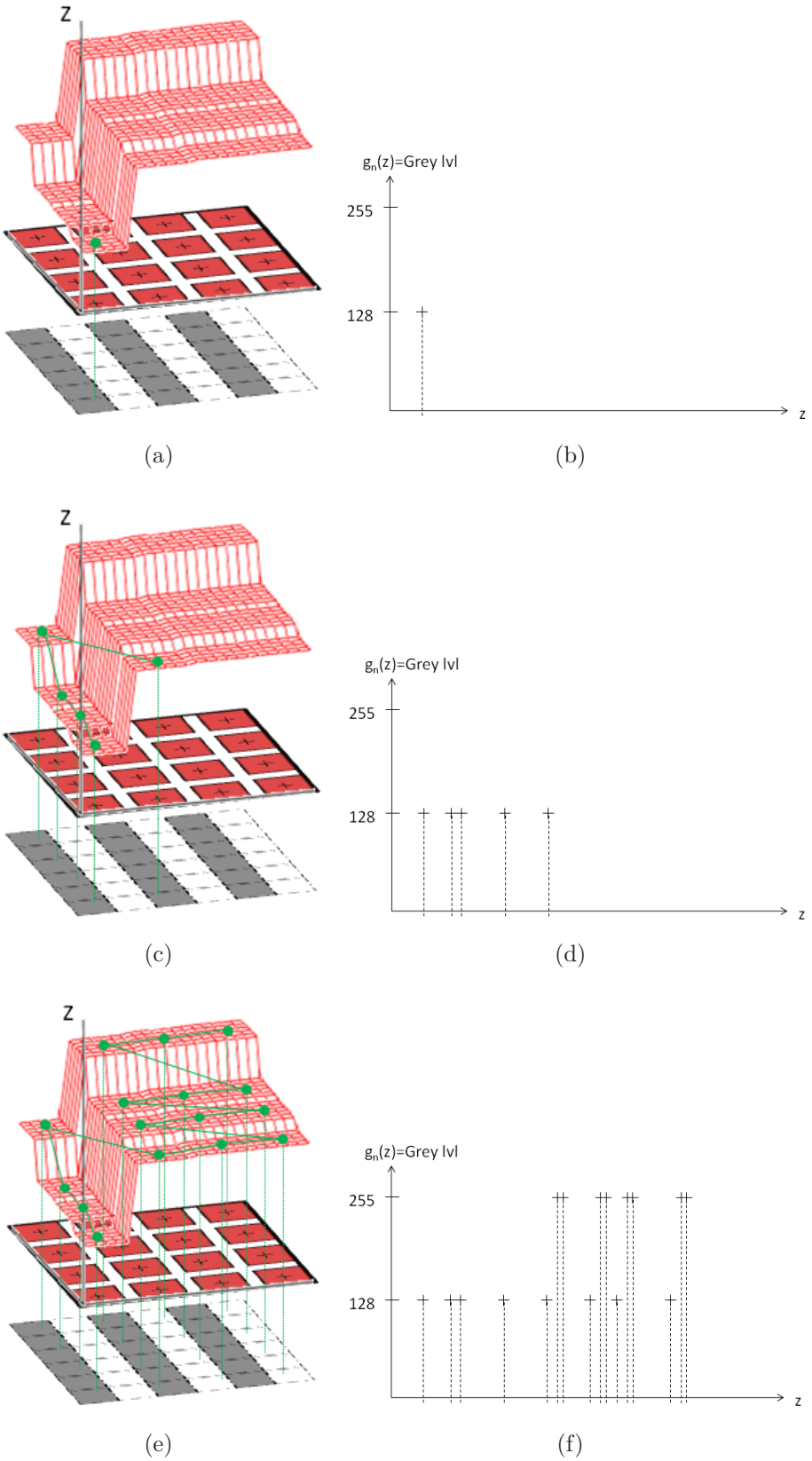


Figure 2.12 – Génération des points de l'ensemble $\mathscr{S}$ de g_n. En commençant par le palier le plus bas (a), on construit un point dans g_n : son ordonnée correspond à la valeur du pixel situé au centre de l'hypercube, et son abscisse est la valeur du palier associé à cet hypercube (b). Puis on répète l'opération pour tous les autres hypercubes (c-f).

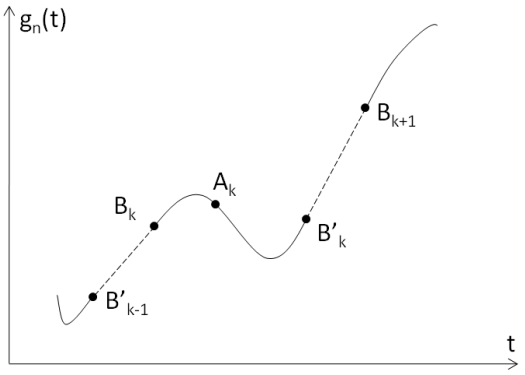


Figure 2.13 – Graphe de g_n. Les points B, A et B' sont reliés par une spline de degré neuf. Les points B' et B sont connectés par des droites.

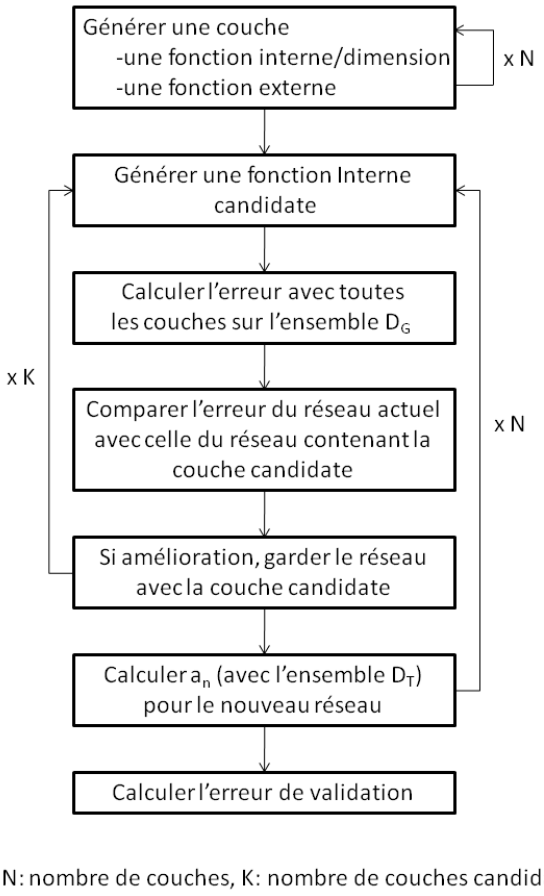


Figure 2.14 – Principales étapes de construction du réseau.

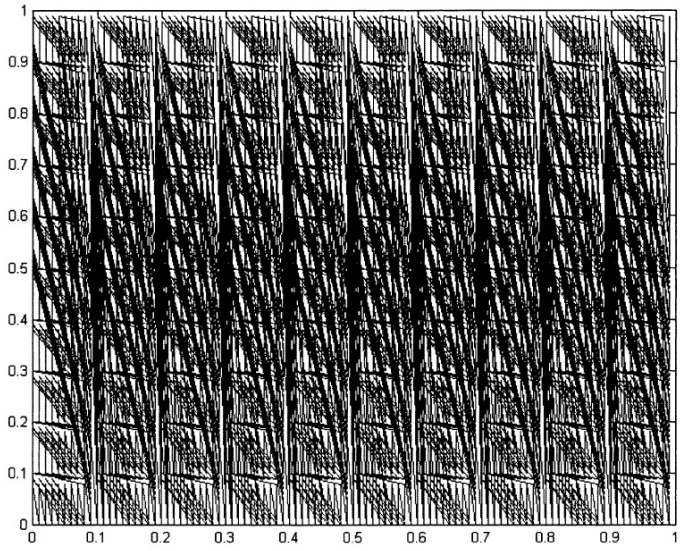


Figure 2.15 – Courbe de remplissage de l'espace associée à la fonction ξ de l'algorithme de Sprecher, d'après [Sprecher and Draghici, 2002].

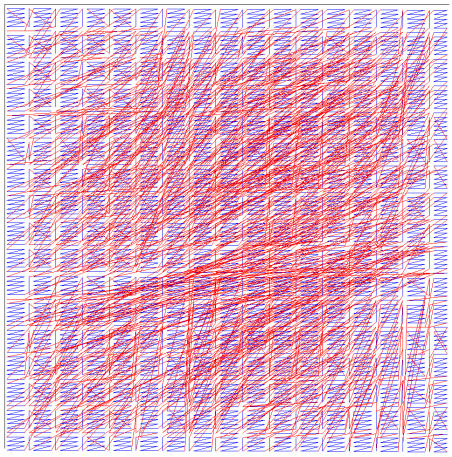


Figure 2.16 – Vue du dessus d'une fonction de balayage du KSN.

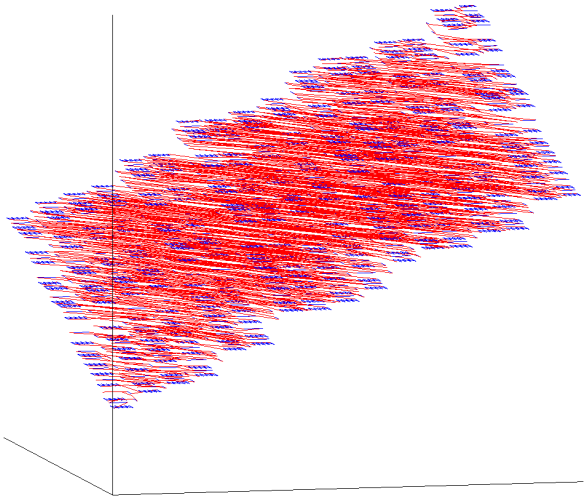
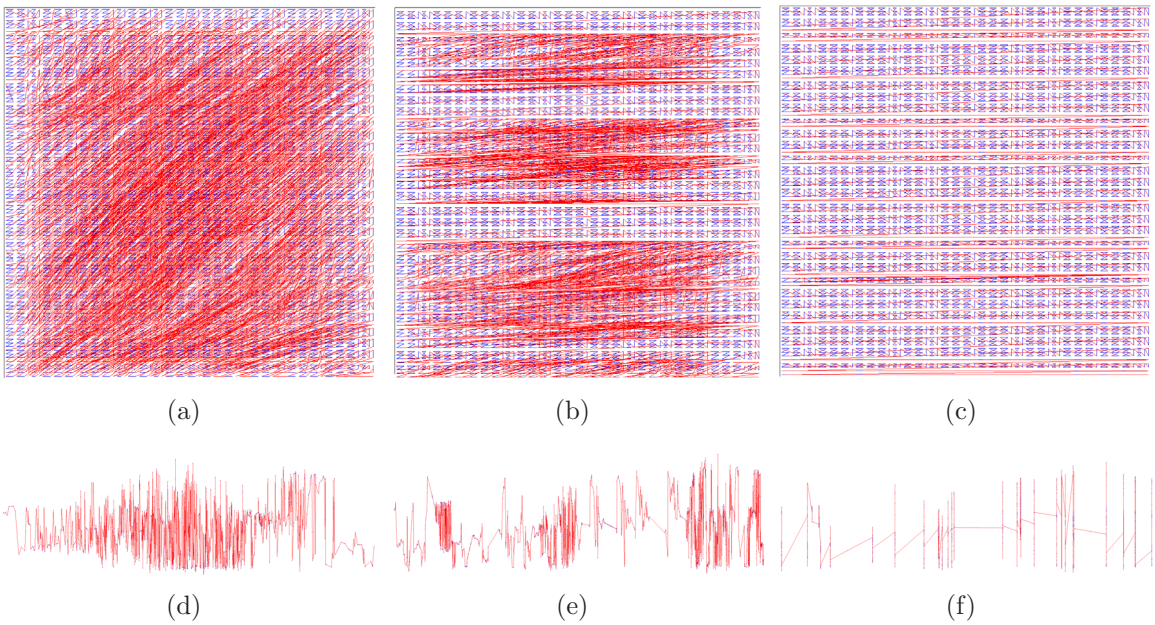


Figure 2.17 – Vue 3D d'une fonction de balayage du KSN.



(a) (b) (c)

(d) (e) (f)

Figure 2.18 – Courbes de balayage : (a) $\lambda_1 \simeq 0.55, \lambda_2 = 0.4$, (b) $\lambda_1 \simeq 0.95, \lambda_2 = 0.04$, et (c) $\lambda_1 \simeq 0.95, \lambda_2 = 0.00004$. Et (d)(e)(f), respectivement, un exemple de fonctions externes associées.

(a) (b) (c)

(d) (e) (f)

(g)

Figure 2.19 – Reconstruction de Lena, (a-f) sont les reconstructions partielles associées aux couches du réseau, (g) est l'image reconstruite finale par la somme pondérée des couches.

2.2.8 Conclusion

L'algorithme proposé par Igelnik permet de construire un réseau de fonctions monodimensionnelles selon le schéma du TSK formulé par Kolmogorov, cf. équation 1.1. Nous avons appliqué cet algorithme à la décomposition d'images en niveaux de gris. La qualité de la reconstruction dépend de la taille du pavage sélectionné, comme nous l'étudions en détail dans le chapitre 3. En effet, par rapport à l'approche proposée par Sprecher, celle d'Igelnik permet de modifier la définition des fonctions monodimensionnelles directement. Par ailleurs, il est facile d'extraire et de représenter les fonctions monodimensionnelles (elles sont définies de façon paramétrique), ce qui permet d'observer comment varient ces fonctions en fonction des images et des différents paramètres, tels que la taille du pavage.

Dans ce chapitre, nous avons présenté les deux principaux algorithmes de représentation de fonction multivariées issus du TSK, et nous avons illustré leurs fonctionnements sur des images en niveaux de gris. Pour résumer, l'algorithme de Sprecher est plus à facile à implémenter que celui d'Igelnik, et correspond exactement au théorème. Les résultats montrent une convergence très rapide de l'algorithme vers la fonction multidimensionnelle. Néanmoins, afin de pouvoir manipuler facilement les fonctions monodimensionnelles, nous avons choisi de travailler à partir de l'algorithme d'Igelnik.

Chapitre 3

Compression d'images

Nous proposons une version modifiée de l'algorithme d'Igelnik pour la compression des images en niveaux de gris. Cette décomposition permet d'obtenir une représentation originale de l'image par des fonctions monodimensionnelles continues, qui supprime la relation de voisinage entre les pixels, mais offre en contrepartie plusieurs avantages pour la compression. Par exemple, il devient possible d'utiliser des outils et méthodes dans cet espace $1D$, ou encore de simplifier les fonctions monodimensionnelles en tirant parti de la continuité. Par ailleurs, notre approche repose sur la combinaison de plusieurs techniques pour améliorer les résultats, y compris à faibles bitrates incluant la décomposition en ondelette, la simplification et l'interpolation des fonctions monodimensionnelles, et l'intégration de ce schéma comme une étape supplémentaire du schéma de compression JPEG 2000, afin d'illustrer le potentiel de notre approche sur un standard performant et reconnu.

Nous présentons dans ce chapitre les caractéristiques de la décomposition en fonctions monodimensionnelles pouvant être utilisées pour compresser une image. Nous détaillons ensuite notre approche, qui associe transformée en ondelettes et décomposition en fonctions monodimensionnelles. Enfin, nous présentons l'intégration de notre approche au schéma JPEG 2000 et les résultats obtenus avec cette compression JPEG 2000 modifiée.

3.1 Etat de l'art

3.1.1 Méthodes de compression

La compression est un outil très répandu et utilisé par divers formats de données multimédia, telles que la musique, la vidéo, et les images. Parmi les formats les plus connus, on peut citer notamment : mp3, jpg ou encore zip. La compression cherche à réduire la redondance dans les données et peut être effectuée avec ou sans perte. La compression sans perte est souvent préférée à une compression avec perte pour l'archivage d'images de haute qualité - telles que des images médicales ou des schémas techniques, car une compression avec perte introduit des artéfacts lorsque le taux de compression augmente (pour des bitrates faibles). Néanmoins, les méthodes avec pertes sont appropriées pour la compression d'images naturelles, pour lesquelles une réduction substantielle du bitrate peut être obtenue au prix d'une perte de la qualité acceptable, c'est-à-dire sans dégradation visible à l'œil nu.

Salomon a présenté dans [Salomon, 2007] un état de l'art détaillé des méthodes de compression, parmi lesquelles ont peut notamment citer, pour les méthode de compression sans perte :

- **Les codages entropiques.** Ce type de codage utilise les propriétés statistiques de l'image pour associer les codes les plus courts aux symboles les plus fréquents. Parmi les codages entropiques les plus connus, on trouve le code de Huffman (qui repose sur la création d'un arbre) et le codage arithmétique (qui associe un intervalle dont la taille dépend de la fréquence du symbole).
- **Les codages par dictionnaires.** Ces approches reposent sur la construction d'un dictionnaire pour associer le code aux symboles. Le plus connu est celui proposé par Lempel-Ziv, qui a donné naissance à de nombreux dérivés (LZ78, LZMA, *etc*). L'algorithme ne considère plus les données symbole par symbole, mais par mots, qui peuvent être constitués de plusieurs symboles. Là encore, les mots les plus fréquents ont les codes les plus courts.

- **Les codages prédictifs.** La valeur de chaque symbole est prédite à partir de ceux précédemment codés, et seul l'écart entre la valeur prédite et la valeur réelle est quantifié puis codé. L'écart étant en général faible, sa représentation nécessite moins de bits que le symbole lui même.
- **Le codage par plages ("Run-length encoding").** Ce codage s'applique essentiellement pour les documents en noir et blanc : au lieu de coder les pixels séparément sur un bit, on code le nombre de pixels identiques consécutifs.
- **Les techniques hybrides.** Il s'agit des méthodes qui associent plusieurs approches pour être plus efficaces. La méthode connue sous le nom de "Deflate", par exemple, est utilisé pour le format "png", et associe un codage entropique (Huffman) à un codage par dictionnaire (LZ77).

La compression sans perte demeure, sauf cas particulier, moins performante que la compression avec pertes. De plus, parmi les approches citées, la plupart n'ont pas été développées spécifiquement pour les images. En général, l'image est convertie en un signal à une dimension par une lecture ligne par ligne.

Les méthodes de compression avec pertes incluent :

- **Le codage scalaire** Cette méthode découle du procédé permettant d'approcher un signal continu par des valeurs d'un ensemble discret de taille limitée. Un quantifieur scalaire peut se définir comme une fonction qui associe à un intervalle de valeurs de l'espace de départ une seule valeur de l'espace d'arrivée. La forme typique d'un quantifieur est une fonction en escalier. Ce procédé est en général associé à d'autres opérations, comme une transformée en ondelettes. La quantification permet alors d'ajuster la précision du codage des coefficients.
- **Le codage vectoriel** Cette méthode est une généralisation du codage scalaire. Il s'agit de compresser non plus les symboles un par un, mais par blocs. Les blocs sont comparés à un dictionnaire, et le bloc à coder est remplacé par l'adresse du mot du dictionnaire le plus proche. Cette approche permet d'obtenir des résultats au moins aussi bon qu'un codage scalaire, mais elle est plus coûteuse en temps de calcul.
- **Les transformations dans l'espace de couleurs.** Une image couleur est généralement codée sur trois canaux (Rouge, Vert, Bleu). Ces approches consistent à exploiter

les propriétés de la vision humaine, qui ne perçoit pas les couleurs de manière uniforme, pour transposer les canaux RGB dans un autre espace (par exemple YUV dans JPEG). Il existe de nombreuses contributions, comme par exemple [Papamarkos et al., 2002], où les auteurs proposent une méthode de réduction des couleurs adaptatives.

– **Chroma subsampling.** Cette approche consiste à coder les informations de couleur à une résolution inférieure à celle de l'image. Cette méthode est utilisée par JPEG, après une transposition des couleurs RGB dans l'espace YCbCr (luminosité, Bleu, Rouge), et autorise une quantification plus importante pour les canaux Cb et Cr. Ce type d'approche est également utilisé pour la compression de vidéos, comme dans [Chen et al., 2009].

– **La compression fractale.** Le principe de cette compression, décrit en détails dans [Barnsley et al., 1993], est de découper l'image en blocs, puis de trouver pour chacun de ces blocs un motif simple, qui après plusieurs transformations affines (rotations, translation, étirement) permet d'approcher le bloc le plus fidèlement possible. Parmi les contributions récentes, on peut noter [Distasi et al., 2005], qui propose une optimisation de la phase d'encodage de l'image, mais qui reste coûteuse en temps de calcul comparée aux autres méthodes de compression.

Toutefois, la plupart des approches récentes et utilisées dans les standards tels que JPEG ou JPEG 2000, reposent sur une transformation permettant de transférer l'énergie des pixels de l'image sur seulement quelques coefficients, comme une transformée en ondelettes ou en cosinus discret (DCT). En fonction de la précision du codage de ces coefficients, ces approches peuvent amener à des compressions avec ou sans perte. Parmi les méthodes les plus connues, on trouve :

– **SPIHT (Set Partitioning In Hierarchical Trees).** Cette méthode a été proposée par Said ([Said and Pearlman, 1996]) et utilise la corrélation entre les coefficients d'ondelette à travers les différents niveaux de décomposition pour proposer une compression efficace avec ou sans perte.

– **JPEG 2000.** Ce standard repose également sur une décomposition en ondelettes, et offre une compression avec ou sans perte également. Nous décrivons plus précisément ce standard dans la section suivante, puisqu'il est utilisé dans nos travaux.

– **Bandelets.** Plus récemment, Le Pennec et Mallat, dans [Le Pennec and Mallat, 2005], ont proposé une décomposition de l'image selon une base dont les vecteurs sont orientés selon les variations régulières des niveaux de gris. Cette technique encore peu répandue est plus performante que la décomposition en ondelettes.

3.1.2 Le standard JPEG 2000

La standard JPEG 2000 permet de compresser avec ou sans perte des images ayant une ou plusieurs composantes. De plus, il propose des fonctionnalités supplémentaires telles que la reconstruction progressive de l'image en résolution spatiale ou en précision, de très bonnes performances à bas-débit (pas d'artéfact comme le standard JPEG par exemple), et une bonne résistance aux erreurs de transmission.

Nous décrivons brièvement les principales étapes d'encodage et de décodage de l'algorithme (illustrées par la Figure 3.1), et nous invitons le lecteur à consulter le livre de [Taubman and Marcellin, 2001] et les articles de [Skodras et al., 2001] pour une description détaillée, et [Adams, 2001] pour les détails de l'implémentation.

– La première étape concerne la transformation des composantes de l'image, vers l'espace YCbCr pour la compression avec pertes, ou vers YUV pour la compression avec ou sans perte.
– Ensuite, chacune de ces tuiles est décomposée en utilisant une transformée en ondelettes discrète (DWT), avec plusieurs niveaux hiérarchiques (optionnel). Deux familles d'ondelettes peuvent être utilisées : les ondelettes 5-3 de LeGall pour la compression avec ou sans perte (transformation réversible) et les ondelettes 9-7 de Daubechies pour la compression avec perte (transformation irréversible).
– Les coefficients sont préalablement groupés, dans chaque sous-bande, en blocs rectangulaires (code-blocks) de taille 64×64 ou 32×32. Puis chaque bloc est quantifié et encodé en plans de bits par un codage arithmétique adaptatif.
– Enfin, le flot de bits final est composé, en ajoutant un en-tête décrivant les différents paramètres utilisés et l'ordre dans lequel les code-blocks sont placés (pour les options de reconstruction progressive).

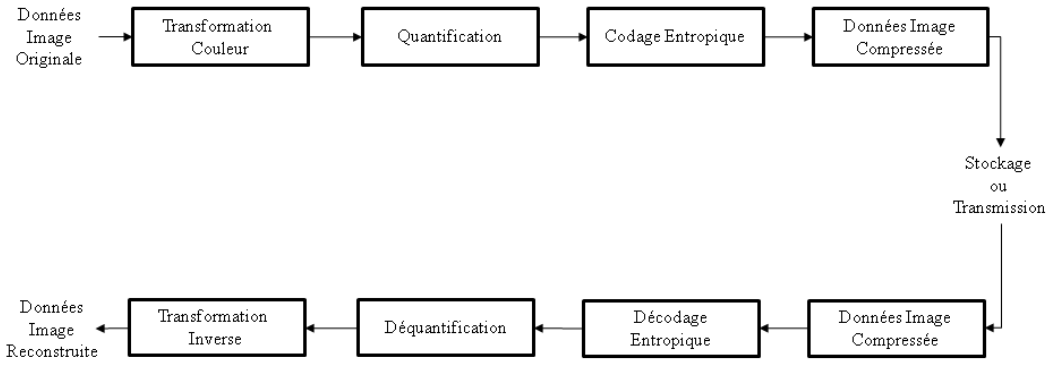
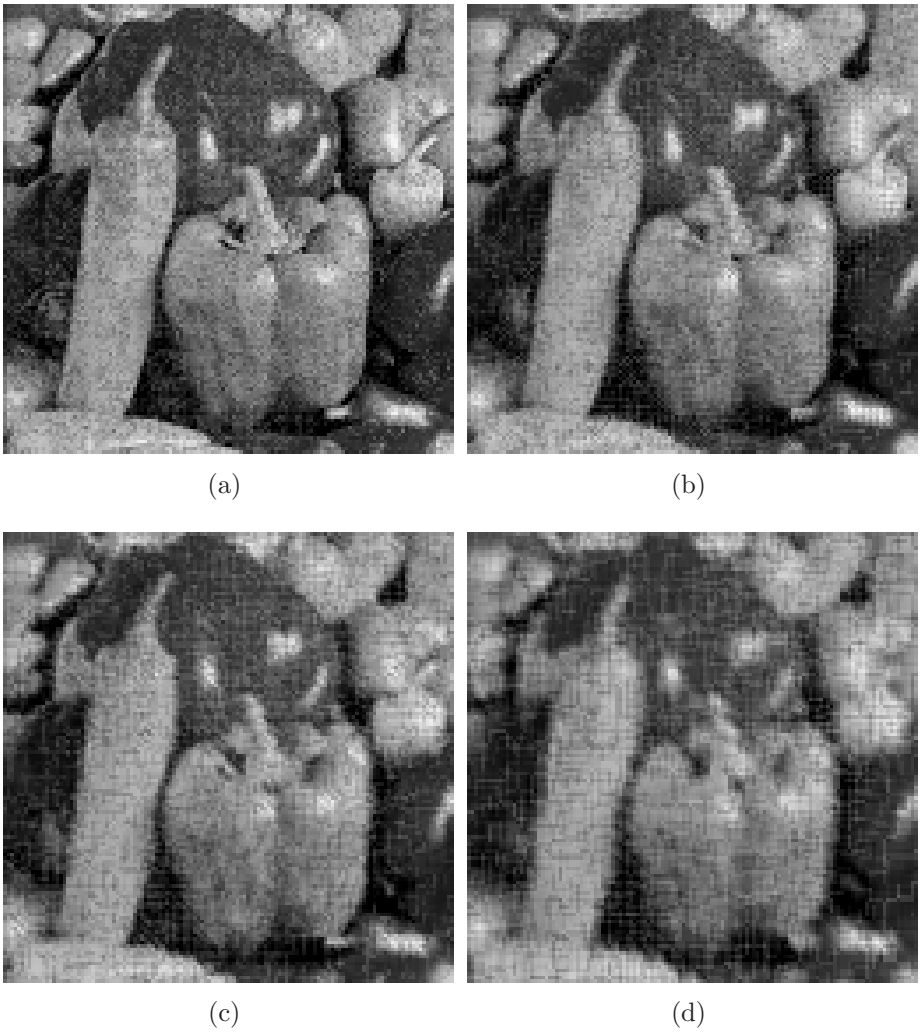


Figure 3.1 – Principales étapes de la compression par JPEG 2000.

3.2 Compression avec KSN

Dans cette section, nous présentons l'approche utilisée pour réduire la quantité d'information nécessaire pour la construction des fonctions externes. Pour cela, deux approches distinctes sont utilisées : la première intervient dès la construction du réseau et la deuxième se base sur une simplification de la décomposition obtenue. En changeant les paramètres δ et N, la taille du pavage peut être modifiée, *i.e.* , le nombre de pavés par couche. La taille des pavés détermine directement le nombre de pixels de l'image originale utilisés pour la construction du réseau, puisque les valeurs des pixels situés au centre des pavés sont utilisées pour la construction des fonctions externes g_n. Diminuer le nombre de pixels utilisés dans les fonctions externes permet d'utiliser seulement une partie des pixels de l'image. Dans ce cas, il s'agit d'un sous-échantillonnage régulier de l'image, avec une grille fixe.

La Figure 3.2 illustre les images obtenues par reconstruction avec un réseau utilisant entre 100% et 25% des pixels de l'image originale, de résolution 128×128. Il apparaît clairement que la qualité décroit très rapidement à cause de l'apparition d'artéfacts, qui sont dûs aux erreurs d'approximation entre les pavés. La Figure 3.3 confirme que l'interpolation effectuée par notre décomposition est moins performante que les méthodes d'interpolation classiques. Notre approche offre des performances équivalentes à l'interpolation par les plus proches voisins, mais l'interpolation bi-cubique améliore le PSNR de l'image reconstruite d'au moins 4dB.



(a) (b)

(c) (d)

Figure 3.2 – Reconstructions à partir de 100%(a), 75%(b), 50%(c), et 25%(d) des pixels de l'image originale.

Ces résultats montrent que changer seulement la densité du pavage n'offre pas des taux de compression intéressants : le PSNR de l'image reconstruite est inférieure à 30dB dès que le taux de compression est supérieur à 15%. Cela illustre que la nouvelle relation de voisinage créée par la réorganisation dans les fonctions externes n'est pas exploitée correctement ou plutôt qu'elle n'est pas intéressante en l'état actuel, ce qui est lié au caractère aléatoire du balayage. Pour améliorer ces résultats, deux approches sont possibles : contrôler le balayage afin d'améliorer l'interpolation en créant des relations de voisinage dans les fonctions externes dépendantes de l'image, ou sélectionner les pixels à enlever non plus selon une grille fixe, mais *a posteriori*, en observant leur importance pour la construction des fonctions externes. Nous avons montré dans le chapitre 2 que le contrôle

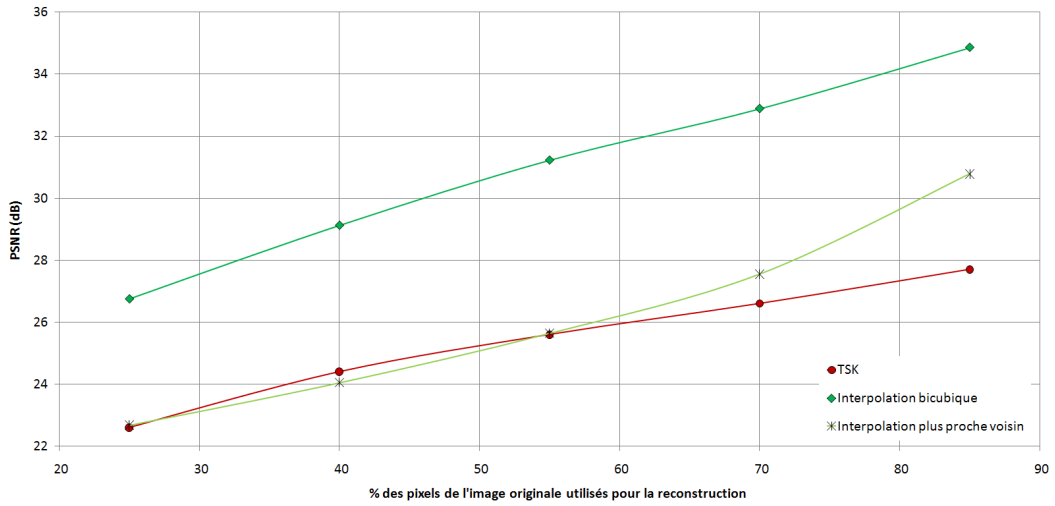


Figure 3.3 – PSNR de l'image reconstruite en fonction du nombre de pixels utilisés pour construire les fonctions externes.

global du balayage était possible, toutefois, le contrôle local du balayage en tenant compte des spécificités de l'image (contours par exemple) reste un problème ouvert. Nous avons donc exploré la deuxième solution, consistant à sélectionner les pixels importants pour la construction du réseau et à "enlever" ceux pouvant être reconstruit sans erreur ou avec une erreur faible. De plus, nous effectuons la décomposition non plus sur l'image directement, mais sur les images de détails obtenues par une transformée en ondelettes. Cela présente deux avantages : l'algorithme est plus performant pour décomposer et reconstruire des données éparses ; et les images de détails étant orientées, nous pouvons mettre à profit le contrôle global du balayage.

3.2.1 Approche ondelettes

La décomposition en ondelettes a été abordée dans de nombreux articles et livres, autant pour présenter les aspects mathématiques que les intérêts de ce type d'approche, notamment pour la compression. Le lecteur intéressé trouvera dans [Mallat, 1998] une présentation complète de cet outil. Pour nos travaux, nous avons utilisé une base d'ondelettes séparable, ce qui permet de filtrer les lignes et les colonnes de l'image séparément. Des signaux de détails et d'approximations sont obtenus et définis à partir de la relation

récursive :

$$\begin{aligned} A_{i-1}x &= A_i x + D_i x \\ &= \sum_{n=-\infty}^{+\infty} \langle x, \Phi_{i,n} \rangle \Phi_{i,n} + \sum_{n=-\infty}^{+\infty} \langle x, \Psi_{i,n} \rangle \Psi_{i,n} \end{aligned}$$

où la suite $\langle x, \Phi_{i,n} \rangle$ est constituée des coefficients d'échelle, et $\langle x, \Psi_{i,n} \rangle$ des coefficients d'ondelette. L'image est divisée en 4 sous-images, où une sous-image contient les basse-fréquences et trois contiennent les hautes-fréquences. Cette décomposition est récursive, ainsi, l'image d'approximation peut à nouveau être re-décomposée en 4 sous-images selon le même schéma, comme l'illustre la Figure 3.4

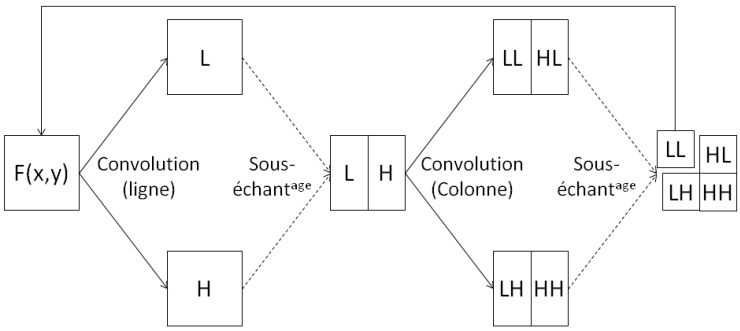
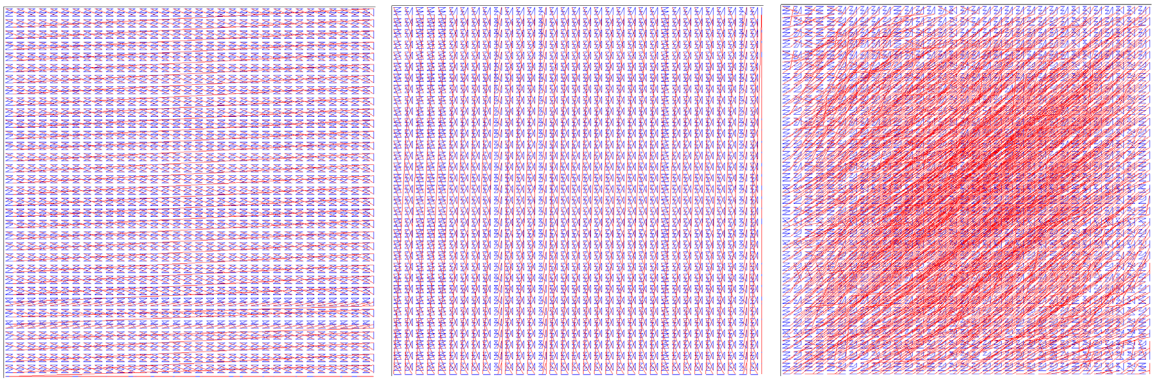


Figure 3.4 – Schéma de décomposition multiresolution.

Notre approche consiste à décomposer les images de détails (*i.e.* les sous-images contenant les hautes-fréquences) avec un pavage dense, et en utilisant le contraste limité, la sparsité et la continuité de cette décomposition, à réduire le nombre de coefficients nécessaires à la construction des fonctions externes. La décomposition des images de détails conduit à des fonctions externes pouvant être simplifiées en utilisant deux approches. Ces deux approches sont appliquées pendant la construction des fonctions externes, après la génération des points de l'ensemble $\mathscr{S}$ (voir la section 2.2.4). Une fois les fonctions externes simplifiés, il est possible de déterminer les coefficients nécessaires à la construction du réseau, et ceux pouvant être approchés, qui peuvent alors être simplifiés, *i.e.* enlevés des images de détails. De plus, pour optimiser la décomposition, le sens de balayage global est ajusté avec les constantes λ_1 et λ_2 pour correspondre à l'orientation des images de détails,

comme illustré sur la Figure 3.5 : ligne par ligne (a) pour les détails horizontaux, colonne par colonne (b) pour les détails verticaux et (c) pour les détails diagonaux.



(a) (b) (c)

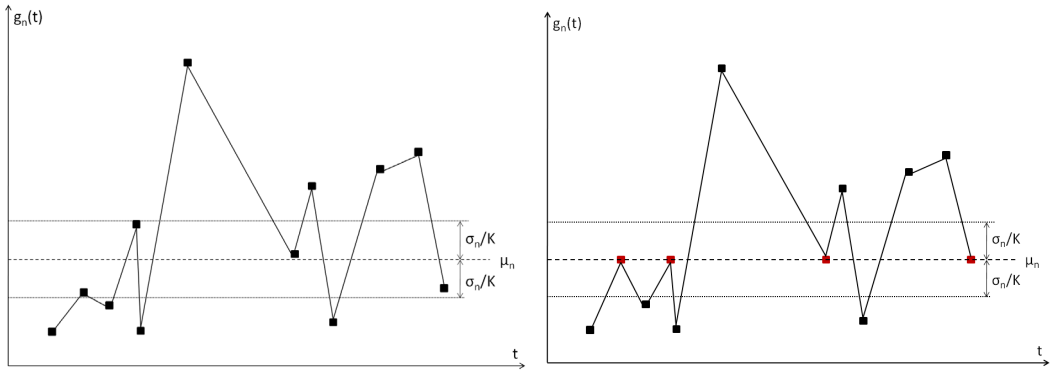
Figure 3.5 – Courbes de balayage : (a) $\lambda_1 = 0.00004, \lambda_2 \simeq 0.95$, (b) $\lambda_1 \simeq 0.95, \lambda_2 = 0.00004$, et (c) $\lambda_1 \simeq 0.55, \lambda_2 = 0.4$.

la première simplification consiste à remplacer par la valeur moyenne de la fonction externe les coefficients appartenant à la bande centrée autour cette valeur moyenne. Pour chaque fonction externe g_n de valeur moyenne μ_n et d'écart type σ_n, les valeurs $g_n(p_{nj_1,j_2})$ sont remplacées par μ_n lorsqu'elles vérifient la condition :

$$K \in \mathbb{N}^*, |g_n(p_{nj_1,j_2}) - \mu_n| < \frac{\sigma_n}{K} \tag{3.1}$$

On rappelle que $g_n(p_{nj_1,j_2})$ correspond à la valeur de g_n sur l'hypercube H_{nj_1,j_2}, qui est égale à une fraction du coefficient d'ondelettes présent au centre de cet hypercube. La constante K peut être modifiée : une valeur faible augmente l'intervalle des valeurs pouvant être simplifiées. Avec cette simplification, les coefficients situés aux centres des hypercubes ne sont plus nécessaires à la construction du réseau. Cette simplification est illustrée sur la Figure 3.6.

La seconde simplification utilise la continuité des fonctions externes. La seconde simplification est appliquée après la simplification par le critère de la valeur moyenne (donc également après la construction avec les valeurs correspondants aux centres des hypercubes (ensemble $\mathscr{S}$)). Certaines valeurs $g_n(p_{nj_1,j_2})$ peuvent être approchées par interpolation linéaire. Les coefficients utilisés pour construire ces valeurs ne sont alors plus nécessaires. Plus précisément, pour chaque point d'abscisse $t_0 = p_{nj_1,j_2}$, les pentes de deux droites sont



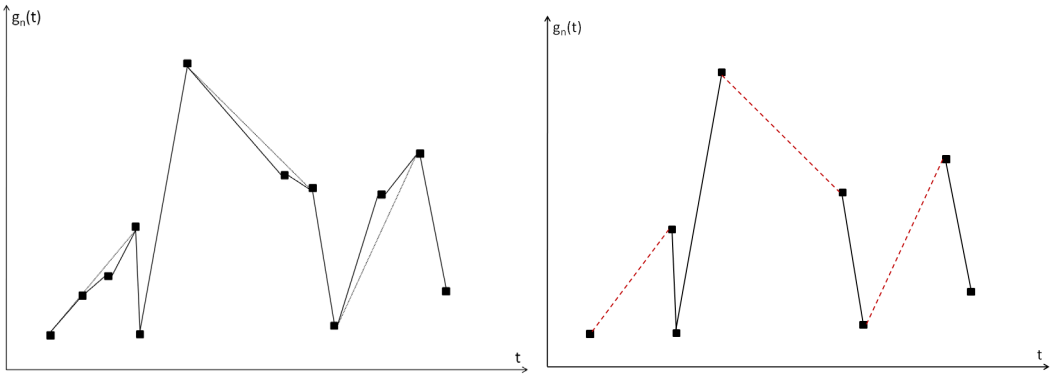
(a) (b)

Figure 3.6 – Simplification des fonctions externes en utilisant le critère de la valeur moyenne. (a) fonction originale, et (b) après simplification.

calculées, la première reliant t_0 au point précédent (d'abscisse t_{-1}) : $r_0 = \frac{g(t_0)-g(t_{-1})}{t_0-t_{-1}}$; et la seconde reliant t_0 au point suivant (d'abscisse t_1) : $r_1 = \frac{g(t_1)-g(t_0)}{t_1-t_0}$. En calculant le ratio de ces deux pentes, le point t_0 peut être enlevé lorsqu'il est aligné avec les points t_{-1} et t_1 :

$$\epsilon \in \mathbb{R}, \frac{r_0}{r_1} = 1 \pm \epsilon. \tag{3.2}$$

La constante ϵ peut être modifiée : une valeur importante augmente le nombre de points simplifiés. La Figure 3.7 illustre cette simplification : les points vérifiant le critère d'alignement peuvent être enlevés puis interpolés (ligne en pointillés).

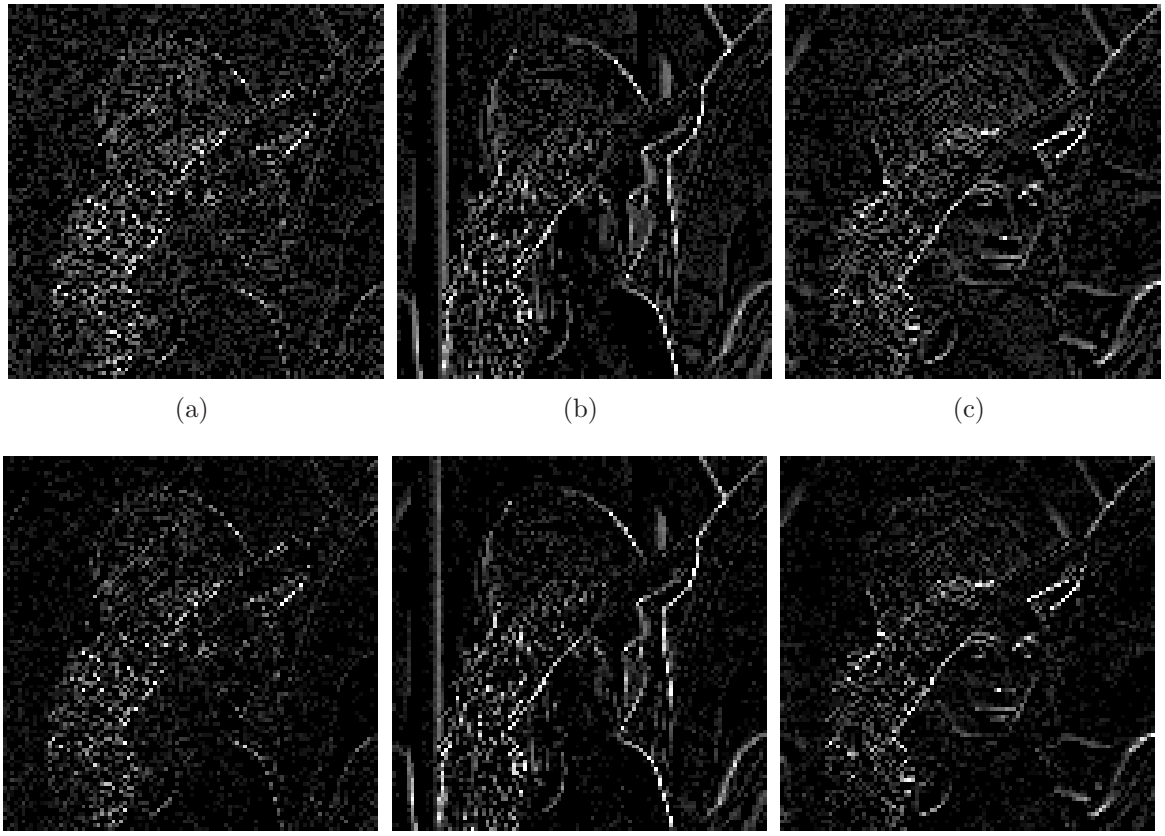


(a) (b)

Figure 3.7 – Simplification des fonctions externes en utilisant le critère d'alignement. (a) fonction originale, et (b) après simplification.

Pour mesurer les résultats obtenus avec notre approche, nous avons limité dans un premier temps nos tests à une décomposition à un niveau par les ondelettes de Haar. La Figure 3.2.1 montre les sous-images haute-fréquence de la décomposition originale par les ondelettes de Haar, et les reconstructions de ces sous-images après décomposition et simplifications dans l'espace des fonctions monodimensionelles. Encore une fois, nous comparons cette approche avec les techniques d'interpolations classiques. Les images de détails sont réduites puis reconstruites par des interpolations bi-cubiques et plus proche voisin. Les PSNR des images reconstruites sont présentés sur la Figure 3.9. On observe que le PSNR des images reconstruites avec notre approche est supérieur à 30dB pour un taux de compression allant jusqu'à 60%. La combinaison de la décomposition par le KSN avec une décomposition en ondelettes fournit de meilleurs résultats que les approches d'interpolations classiques, même pour des taux de compression élevés. La Figure 3.10 présente le taux de compression et le PSNR de l'image après reconstruction pour cinq images. Le PSNR des images reconstruites est supérieur à 40dB pour un taux de compression allant jusqu'à 40%.

La Figure 3.11 présente les images obtenues pour trois simplifications des fonctions externes correspondant à un taux de compression élevé, moyen et faible. On observe que les images ne comportent pas d'artéfacts. La répartition irrégulière des mesures est due aux méthodes de simplification : la simplification des fonctions externes est dépendante de l'image, donc le taux de compression obtenu l'est également. Afin de pouvoir comparer notre approche aux méthodes de compression existantes, nous avons choisi d'intégrer notre méthode dans JPEG 2000, mais avant de détailler les modifications apportées, nous étudions le fonctionnement et les caractéristiques de la simplification que nous proposons.



(a) (b) (c)

(d) (e) (f)

Figure 3.8 – (a), (b), et (c) sont les images de détails obtenues après une décomposition en ondelettes de Haar. (d), (e), et (f) correspondent aux images de détails reconstruites après décomposition par le KSN et simplification des fonctions externes g_n.

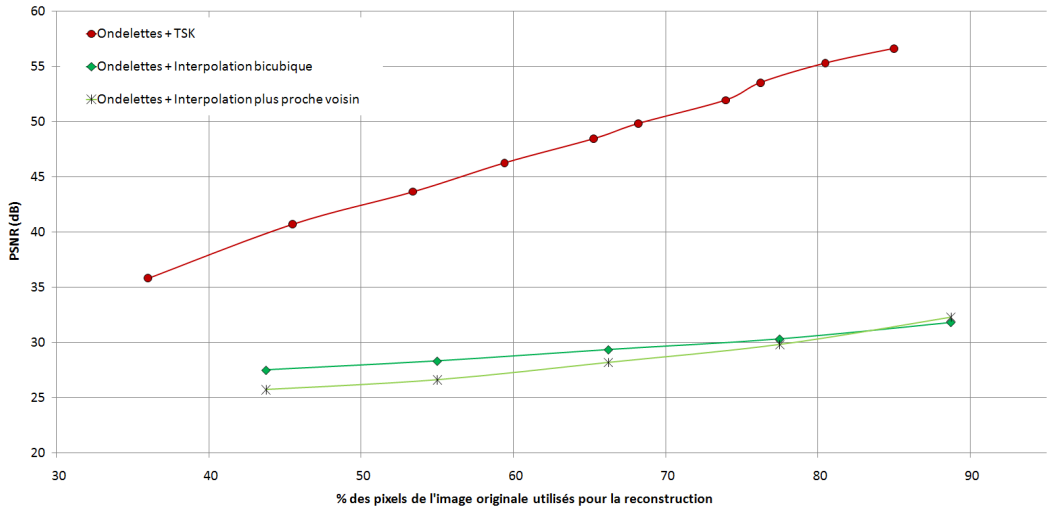


Figure 3.9 – PSNR des images reconstruites en fonction du nombre de coefficients utilisés pour construire les fonctions externes.

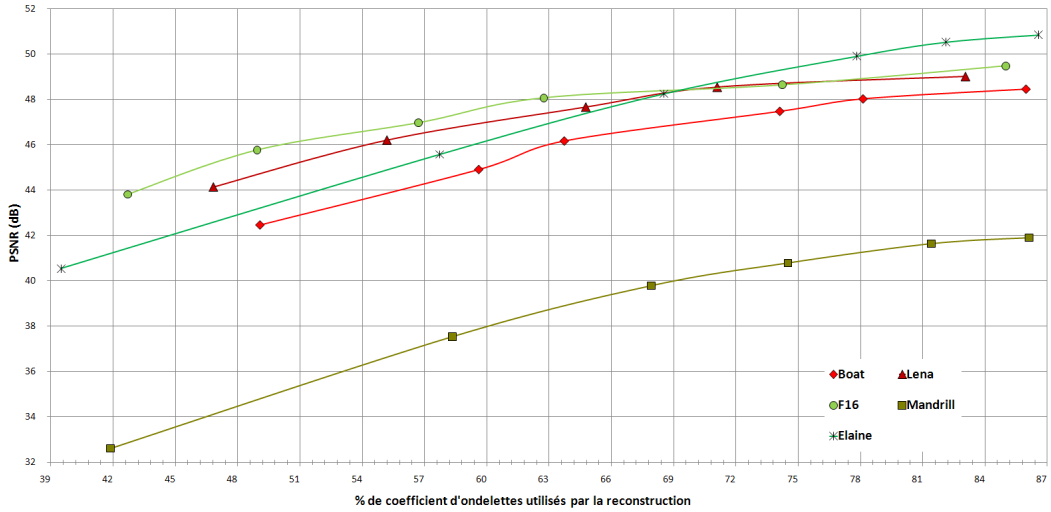


Figure 3.10 – PSNR des reconstructions de cinq images différentes en fonction du pourcentage de coefficients d'ondelettes utilisés pour la construction.

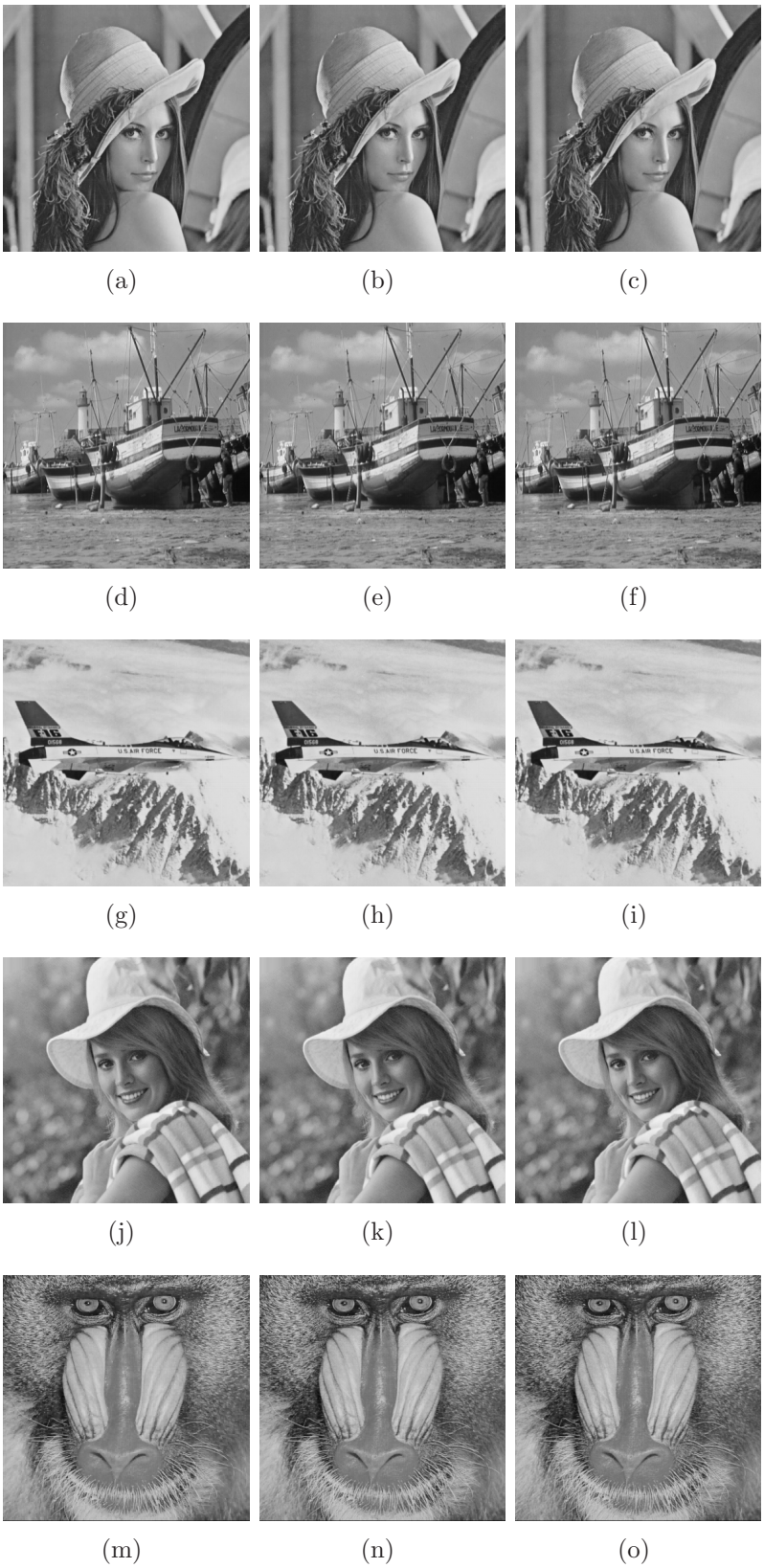


Figure 3.11 – Reconstructions de Lena (a)(b)(c), Boat (d)(e)(f), F16 (g)(h)(i), Elaine (j)(k)(l), et Mandrill (m)(n)(o), utilisant 45%, 65% and 85% des coefficients d'ondelettes (respectivement).

3.2.2 Fonctionnement détaillé

Nous détaillons dans cette section le comportement et les caractéristiques de notre approche combinant ondelettes et décomposition avec le KSN. Pour cela, nous étudions l'influence des simplifications réalisées dans l'espace des fonctions monodimensionnelles sur les images de détail. Le tableau 3.1 présente l'entropie d'une image de détails avant et après la simplification de coefficients. Avec le critère de simplification fixé à $K = 4$ (K est défini dans l'équation 3.1), ce qui représente un bon compromis entre simplification et qualité de reconstruction, on observe que l'entropie est légèrement plus faible après simplification. Le nombre total de coefficients à coder est réduit en supprimant de la redondance entre les informations, les coefficients restants sont donc tous différents, mais moins nombreux. Avec le critère de simplification le plus large, $K = 2$, l'entropie est diminuée ainsi que le nombre de coefficients, mais cette simplification est trop agressive pour être utilisable.

	originale			simplifiée avec K=4			simplifiée avec K=2		
	hh	hl	lh	hh	hl	lh	hh	hl	lh
entropie	4.14	3.44	3.69	3.43	2.99	3.30	2.18	2.14	2.44
nb. de coef.	36864	36864	36864	20064	20256	22401	11588	9868	13301

TABLE 3.1 – Mesures comparatives de l'entropie d'une image de détails, avant et après simplification.

Nous illustrons à présent l'influence de la simplification sur les fonctions monodimensionnelles. Pour cela, nous mesurons l'entropie des fonctions externes. On simule l'encodage des fonctions externes en utilisant une quantification vectorielle après les avoir simplifiées avec les critères de continuité et de valeur moyenne. Nous comparons ces résultats avec le signal $1D$ obtenu avec le balayage ligne par ligne de la même image de détail et encodé avec le même codage vectoriel. Nous convertissons et encodons avec la même méthode l'image de détail, après avoir enlevé 50% des coefficients par une interpolation cubique. Ces résultats sont présentés dans le tableau 3.2. On observe que l'entropie moyenne d'un vecteur est plus élevée pour les fonctions externes. Autrement dit, les simplifications appliquées aux fonctions monodimensionnelles diminuent la redondance, ce qui augmente l'entropie, tout en conservant seulement les données nécessaires à la construction du réseau. Toutefois, le nombre total de bits requis pour coder une fonction est plus faible que

pour les deux autres signaux, car le nombre total de vecteurs à coder est beaucoup plus faible. Cette sélection "intelligente" des coefficients offre une reconstruction de meilleure qualité, contrairement à une interpolation classique qui sous-échantillonne l'image selon une grille fixe.

	ligne par ligne	interp. cubique	fct. ext. KSN
nb. vecteurs	2129	1290	314
entropie moyenne	2.43	3.09	3.00
écart type	0.51	0.30	0.27
entropie totale	5182	3986	943

TABLE 3.2 – Mesures comparatives de l'entropie d'une image de détails encodée par quantification vectorielle.

Pour étudier la simplification plus en détail, il faudrait déterminer, dans le cas général, les caractéristiques des coefficients des images de détail qui sont simplifiés. Les coefficients simplifiés étant ceux inutilisés par toutes les couches, il y a deux difficultés principales à résoudre pour cette étude théorique :

– En fonction de la densité du pavage, une couche du réseau contient, avant simplification, seulement une partie des coefficients. Les couches de pavage étant construites par translation, les pixels lus par les couches sont différents. Il peut toutefois y avoir des valeurs en commun, si la translation δ est peu importante et/ou si les pavés sont petits (ce qui dépend de N et δ).
– Les simplifications dépendent de l'ordre des coefficients dans la fonction externe (pour la simplification par critère d'alignement) et de la valeur moyenne de la fonction externe : l'ordre des coefficients dépend du balayage. La valeur moyenne dépend des coefficients situés aux centres des hypercubes.

Une des perspectives de recherche envisagée serait de conduire l'étude sur une modélisation de l'image de détails par une mixture de gaussiennes, en simplifiant également le comportement de l'algorithme pour la décomposition. Par exemple, on pourrait assimiler le balayage à un parcours ligne par ligne, colonne par colonne, et diagonal pour les images de détails respectives, et se placer dans le cas où le pavage est suffisamment dense pour que chaque couche contienne tous les coefficients, ce qui permet de s'affranchir de la première difficulté citée précédemment.

3.3 Compression avec JPEG 2000

Pour permettre une comparaison entre notre approche et les méthodes de compression existantes, nous avons intégré notre méthode comme une nouvelle étape dans le schéma de compression et de décompression de JPEG 2000. Il s'agit d'intégrer l'approche présentée dans la section 3.2.1 de décomposition et de compression des images de détails dans JPEG 2000. Pour cela, nous ajoutons une étape supplémentaire après la transformée en ondelettes effectuée par JPEG 2000 et avant la quantification et l'encodage de ces coefficients, sans modifier les autres parties du schéma de compression de JPEG 2000. La Figure 3.13 montre le diagramme de fonctionnement de JPEG 2000 et l'intégration de notre approche. Comme on peut le voir, le décodage doit également être modifié afin de reconstruire les coefficients simplifiés pendant la compression. Cette approche cherche à montrer la faisabilité et le potentiel de la décomposition obtenue par le KSN pour la compression, mais demeure expérimentale : pour pouvoir rendre compatible notre approche avec l'encodage de JPEG 2000, les images de détails doivent être adaptées. En effet, après décomposition et simplification avec le KSN, on obtient une image avec des "trous", là où des coefficients ont été simplifiés (voir la Figure 3.12(a)). Une telle "image" ne peut pas être encodée directement dans JPEG 2000, qui requiert des valeurs pour tous les pixels. Afin de rendre ces images compatibles, nous avons choisi de modifier l'organisation des coefficients pour diminuer le nombre de code-blocks à traiter. Plus précisément, les sous-images obtenues par décomposition en ondelettes sont découpées en pavés de taille 64×64 ou 32×32 pixels, puis chacun de ces code-blocks est codé indépendamment puis rassemblé dans le flot de bits final. En réorganisant les coefficients dans les images de détails après simplification, la quantité de code-blocks à encoder pour chaque sous-images est réduite. Les coefficients sont pour cela rassemblés pour tenir dans le moins de code-blocks possible, ainsi le nombre de code-blocks de l'image totale est réduit, de même que sa taille. Pour réorganiser les coefficients, une table binaire de la taille des sous-images est utilisée pour mémoriser, à chaque position, si un pixel a été simplifié. Les coefficients restants sont lus ligne par ligne pour obtenir une représentation $1D$ (Figure 3.12(b)) et réorganisés pour remplir les sous-images bloc par bloc (Figure 3.12(c)). Cette ré-organisation limite les avantages de notre approche : comme nous l'avons montré dans la section 3.2.2, l'entropie de l'image une fois

simplifiée augmente, autrement dit, une fois découpée en code-blocks, ceux-ci seront plus volumineux une fois codés. Pour réellement proposer une compression, il faut compenser l'augmentation de taille de ces code-blocks en réduisant suffisamment le nombre total de code-blocks à coder. De plus, lorsque l'image devient petite, cette contrainte implique une simplification des coefficients plus importante (et donc une reconstruction de moins bonne qualité), car la taille minimum d'un code-block est de 32×32 coefficients.

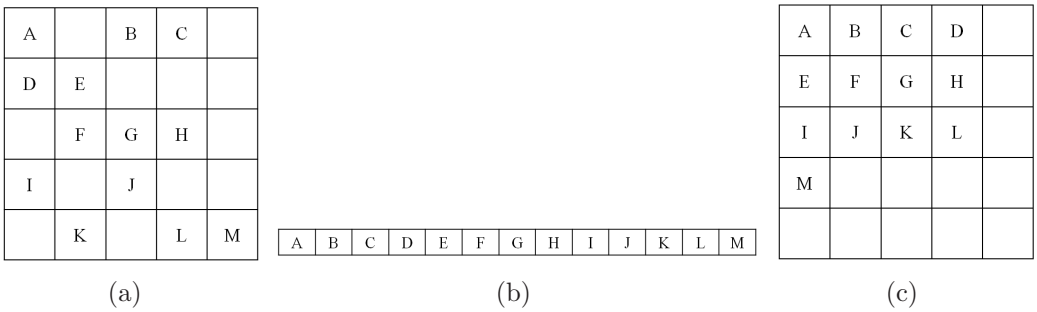


Figure 3.12 – (a) Image de détails après simplification. (b) Representation $1D$. (c) Image de détails "défragmentée".

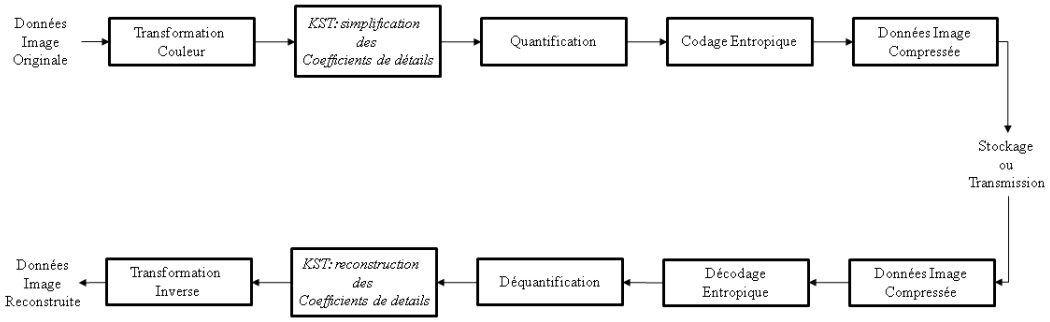


Figure 3.13 – Schéma de JPEG 2000, incluant la décomposition par le KSN.

La Figure 3.14 présente les résultats obtenus pour la compression avec JPEG 2000 d'une image de 384×384 pixels, avec et sans utiliser notre étape additionnelle, pour les ondelettes 5-3 de LeGall (Figure 3.14(a)) et 9-7 de Daubechies (Figure 3.14(b)). Pour effectuer ces mesures, nous avons fixé la taille des code-blocks à 32×32 et limité la décomposition en ondelettes à un et deux niveaux hiérarchiques. En effet, pour une image de cette taille, au delà de deux niveaux, les images obtenues sont trop petites pour être séparées en plusieurs code-blocks.

Avec un seul niveau de décomposition, pour un bitrate inférieur à 1bpp, JPEG 2000 ne quantifie presque aucun coefficient dans les images de détails. Les images de détails

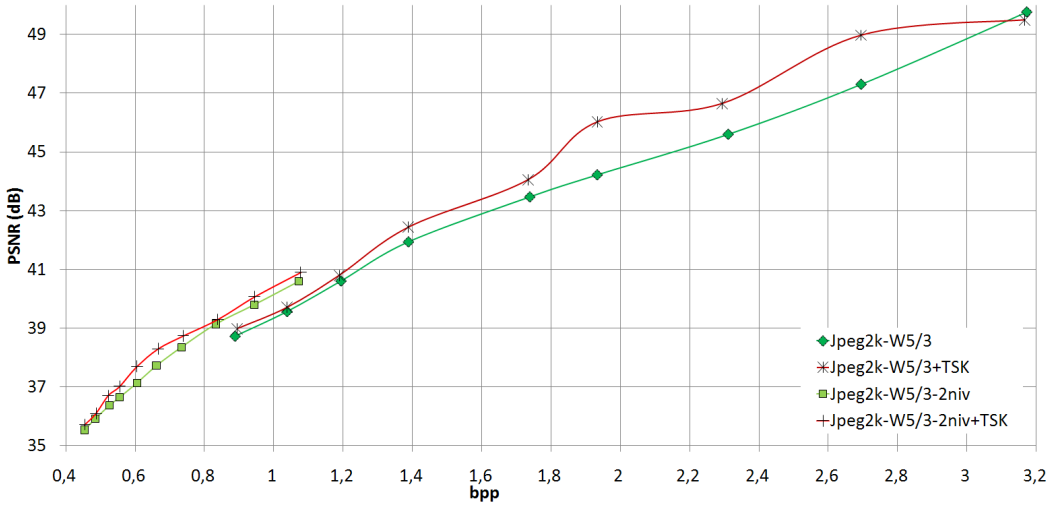
contiennent alors presque uniquement des coefficients nuls, à partir desquels notre algorithme ne peut pas reconstruire à la fois les valeurs qu'il a simplifiées et celles simplifiées par JPEG 2000. C'est ce qui explique que les PSNR de l'image reconstruite avec ou sans notre approche convergent et décroissent rapidement.

Avec deux niveaux de décomposition, notre approche améliore toujours le PSNR, mais marginalement : les erreurs introduites par notre approche dans le premier niveau de décomposition sont propagées au niveau supérieur, et le nombre de code-blocks à coder reste trop important compte tenu de la qualité de reconstruction obtenue.

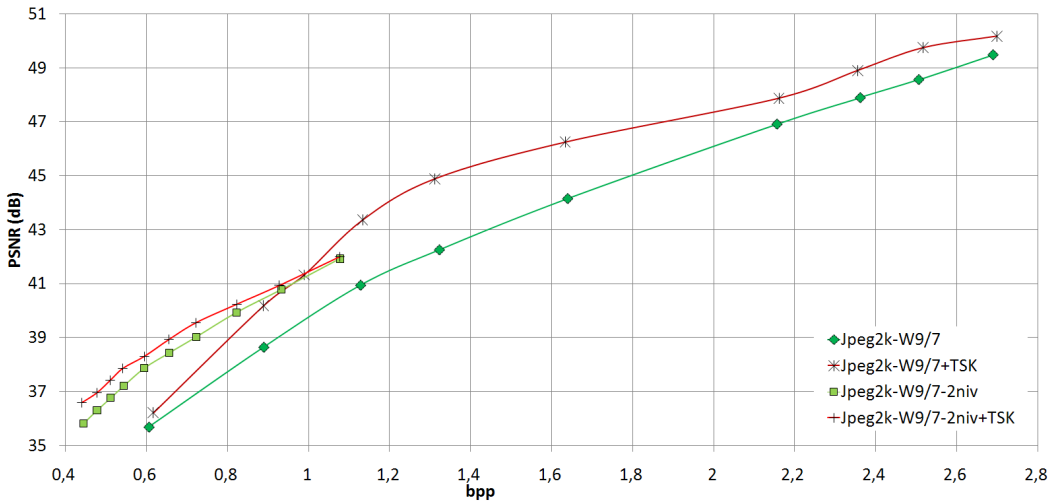
Les résultats présentés dans la Figure 3.14 sont repris dans les tableaux 3.3 et 3.4 pour les ondelettes de LeGall, et les tableaux 3.5 et 3.6 pour les ondelettes de Daubechies. Ces tableaux détaillent les paramètres de simplification utilisés pour les fonctions externes et les taux de compression fixés dans JPEG 2000, ainsi que les bitrates obtenus et les PSNR mesurés lors de la reconstruction, pour un et deux niveaux de décomposition en ondelettes. "def" signifie que le ratio de compression par défaut de JPEG 2000 a été utilisé, ce qui permet d'obtenir une compression sans perte avec les ondelettes de LeGall, et le niveau de précision maximum avec les ondelettes de Daubechies. Dans les tableaux 3.5 et 3.6, deux critère de simplification K sont utilisés : les images des détails diagonaux et verticaux sont simplifiées avec $K = 4$ et les horizontaux avec $K = 2$. On peut voir que le critère de simplification K n'est plus modifié lors de l'utilisation de ratio de compression dans JPEG 2000 ou de transformée en ondelettes à plusieurs niveaux. Il est fixé afin d'obtenir le meilleur ratio entre le nombre de code-blocks à encoder et la qualité de reconstruction. Ces résultats montrent que globalement, malgré la contrainte de la ré-organisation des coefficients, notre approche améliore la compression de JPEG 2000.

Pour les bitrates élevés, le même phénomène que sur le premier niveau de décomposition en ondelettes dans l'exemple précédent se produit : la simplification minimum des fonctions internes réduit déjà significativement la qualité de la reconstruction, sans retirer assez de coefficients pour obtenir une compression plus intéressante que JPEG 2000 seul avec un ratio de compression plus élevé. Par exemple, sur la Figure 3.14(a), le point correspondant à un bitrate de 3.17bpp a un PSNR similaire à JPEG 2000 avec un taux de compression plus élevé et sans l'étape supplémentaire du KSN. Dans tous les autres cas, à bitrate identique, la combinaison de la compression de JPEG 2000 avec l'étape de

simplification par KSN offre de meilleurs résultats qu'une simplification plus importante des fonctions monodimensionnelles avec un ratio JPEG 2000 plus faible. Ce phénomène est illustré par les "creux" dans les courbes (à 1.93bpp pour les ondelettes de LeGall et 2.20bpp pour les ondelettes de Daubechies) correspondants à des mesures obtenues sans compression de JPEG 2000 et en utilisant un taux de simplification plus élevé dans le KSN.



(a)



(b)

Figure 3.14 – Comparaison entre JPEG 2000 avec et sans l'étape KSN, (a) pour les ondelettes 5-3 de LeGall, et (b) pour les ondelettes 9-7 de Daubechies.

bitrate(bpp)	ratio JPEG	K (simplif. val. moyenne)	PSNR(dB)
0.89	8	4	38.98
1.04	7	4	39.71
1.19	6	4	40.80
1.39	5	4	42.44
1.73	4	4	44.05
2.29	3	4	46.02
1.93	def	2	46.24
2.69	def	4	48.97
3.17	def	6	49.49

TABLE 3.3 – PSNR de l'image reconstruite après compression dans JPEG 2000 en utilisant les ondelettes 5-3 de LeGall et le KSN. "def" signifie que le ratio de compression par défaut a été utilisé (qui permet d'obtenir une compression sans-perte en utilisation normale).

bitrate(bpp)	ratio JPEG	K (simplif. val. moyenne) niv. 1	K (simplif. val. moyenne) niv. 2	PSNR(dB)
0,45	15	4	4	35,72
0,49	14	4	4	36,10
0,52	13	4	4	36,71
0,55	12	4	4	37,02
0,60	11	4	4	37,70
0,67	10	4	4	38,30
0,74	9	4	4	38,74
0,84	8	4	4	39,30
0,94	7	4	4	40,07
1,08	6	4	4	40,90

TABLE 3.4 – PSNR de l'image reconstruite après compression dans JPEG 2000 en utilisant les ondelettes 5-3 de LeGall avec deux niveaux de décomposition et le KSN. "def" signifie que le ratio de compression par défaut a été utilisé (qui permet d'obtenir une compression sans-perte en utilisation normale).

La Figure 3.15 illustre les images reconstruites après compression dans JPEG 2000, avec et sans notre approche. On observe que notre méthode n'introduit pas d'artéfact à la compression JPEG 2000, tout en améliorant légèrement le PSNR.

bitrate(bpp)	ratio JPEG	K (simplif. val. moyenne)	PSNR(dB)
0.62	12	4&2	36.21
0.89	8	4&2	40.18
0.99	7	4&2	41.32
1.13	6	4&2	43.36
1.31	5	4&2	44.88
1.63	4	4&2	46.24
2.16	3	4&2	47.87
2.35	def	4&2	48.90
2.52	def	4	49.74
2.7	def	6	50.17

TABLE 3.5 – PSNR de l'image reconstruite après compression dans JPEG 2000 en utilisant les ondelettes 9-7 de Daubechies et le KSN. "def" signifie que le ratio de compression par défaut a été utilisé (qui permet).

bitrate(bpp)	ratio JPEG	K (simplif. val. moyenne) niv. 1	K (simplif. val. moyenne) niv. 2	PSNR(dB)
1,08	6	4&2	4&2	42,00
0,93	7	4&2	4&2	40,93
0,82	8	4&2	4&2	40,23
0,72	9	4&2	4&2	39,55
0,65	10	4&2	4&2	38,93
0,59	11	4&2	4&2	38,30
0,54	12	4&2	4&2	37,86
0,51	13	4&2	4&2	37,42
0,48	14	4&2	4&2	36,95
0,44	15	4&2	4&2	36,59

TABLE 3.6 – PSNR de l'image reconstruite après compression dans JPEG 2000 en utilisant les ondelettes 9-7 de Daubechies avec deux niveaux de décomposition et le KSN. "def" signifie que le ratio de compression par défaut a été utilisé (qui permet).

(a) (b)

(c) (d)

Figure 3.15 – (a)(b) Résultats obtenus avec JPEG 2000 (ondelettes de LeGall 5-3) et notre méthode, à des bitrates de 2.7bpp et 1.3bpp. (c)(d) Résultats obtenus avec JPEG 2000 (ondelettes de Daubechies 9-7) et notre méthode, à des bitrates de 2.7bpp et 1.3bpp

3.4 Conclusion

Nous avons présenté l'adaptation de la décomposition en réseau de splines à la compression d'image. Pour cela, nous avons tout d'abord appliqué l'algorithme du KSN à la représentation d'images d'ondelettes, ce qui a montré des résultats prometteurs. Les images de détails verticaux, horizontaux et diagonaux sont décomposées en fonctions monodimensionnelles, puis des simplifications sont effectuées dans cet espace $1D$, dont les résultats sont des images de détails comportant moins de coefficients. Nous avons montré que cette interpolation originale, offrait, dans l'espace ondelettes, des résultats bien supérieurs aux méthodes d'interpolation classiques, comme l'illustre la Figure 3.9, où l'on note clairement un gain d'au moins 10dB. Nous avons ensuite illustré cette approche sur plusieurs images naturelles, et étudié l'influence des paramètres de simplification sur l'entropie des images de détails issus de la décomposition en ondelettes. Enfin, pour illustrer le potentiel de cette approche, nous avons inclus cette étape de décomposition/simplification dans le schéma JPEG 2000. Les résultats obtenus montrent une amélioration de l'efficacité de la compression, malgré les limites de cette association, puisque l'encodage de JPEG 2000 ne peut tirer complètement avantage de notre simplification. En effet, les coefficients des images de détails, après simplification, doivent être réorganisés pour former des images de détails plus petites mais plus complexes, et dont les coefficients sont décorrélés.

Cette limitation est une des principales perspectives d'amélioration : puisque la connectivité locale des pixels est perdue lors de la décomposition, toute méthode de compression utilisant une représentation par blocs ne peut être optimale. Il faut donc développer de nouveaux algorithmes de codage, adaptés à notre décomposition : soit en tirant parti des images de détails à "trous" obtenues, soit en cherchant à coder directement les fonctions externes.

Sans redévelopper de nouvelles approches de compression, l'approche actuelle peut encore être optimisée. Notamment, les paramètres de simplification ne sont pas forcément optimaux pour tous les ratios de compression. On pourrait également modifier la réorganisation des coefficients par blocs pour tirer parti des spécificités du codage de JPEG 2000 en créant des code-blocks de coefficients plus homogènes.

Enfin, les aspects théoriques et statistiques de cette approche sont encore mal connus et pourraient ouvrir sur de nouvelles applications ou optimisations pour la décomposition des images. L'algorithme abandonne l'organisation spatiale des pixels en échange d'une représentation monodimensionnelle et flexible. En contrôlant le balayage, il serait alors possible d'obtenir des informations sur le signal multidimensionnel à la fois par les fonctions internes et par les fonctions externes. Ces données seraient profitables à toutes les méthodes de traitement ayant besoin d'un signal monodimensionnel et qui convertissent pour cela l'image en la parcourant ligne par ligne, en perdant une partie des informations au passage (le voisinage par exemple).

Chapitre 4

Transmission

Si la compression permet de transmettre efficacement une image en réduisant sa taille, il est parfois intéressant d'augmenter par la suite la qualité de l'image transmise, autrement dit de transmettre progressivement l'image. Nous présentons une version modifiée de l'algorithme d'Igelnik pour la transmission progressive d'images en faisant varier la taille des pavés pour chaque couche. L'image est décomposée en fonctions monodimensionnelles qui peuvent être transmises progressivement, en plusieurs étapes. A chacune de ces étapes intermédiaires, une reconstruction basse-résolution de l'image est calculée. Chaque fonction ajoute de nouvelles données à celles précédemment transmises, ainsi, l'image est progressivement reconstruite jusqu'à sa résolution initiale, sans erreur. Nous présentons dans la section suivante l'aspect progressif totalement adaptable, puis nous présentons dans la deuxième moitié de ce chapitre l'autre principal avantage de notre approche : sa résistance au bruit.

4.1 Transmission Progressive

4.1.1 Etat de l'art

Si une image est transmise progressivement ligne par ligne, ou colonne par colonne, il faut en général attendre qu'une grande partie de l'image ait été transmise avant de pouvoir la reconnaitre, autrement dit il faut déjà transmettre beaucoup de données, ce qui peut être long sur des réseaux dont le débit est faible. Avec une compression traditionnelle, il faut alors choisir si on souhaite transmettre une image rapidement (de petite taille, donc fortement compressée) ou de haute qualité (faiblement compressée, donc plutôt lourde). Il peut donc être intéressant de développer des méthodes permettant d'améliorer la qualité de l'image reçue en fonction du canal de transmission : c'est le but de la transmission progressive (Progressive Image Transmission, PIT). Plutôt que de transmettre une image haute résolution séquentiellement, une approximation de l'image est transmise et converge progressivement vers l'image originale. Dans [Chee, 1999], Chee propose de classifier les méthodes de transmissions progressives en 4 catégories (non exclusives) :

a. **Les méthodes par approximations successives ("Successive Approximation Method")**

   La qualité de l'image est progressivement augmentée en améliorant la précision de l'encodage des données. Par exemple, dans le cas où l'image a été encodée avec un codage vectoriel, la transmission commence alors avec les bits de poids forts puis continue jusqu'aux bits de poids faibles. Cette méthode peut également s'appuyer sur une transformée en ondelettes.

   – **Plan de Bits ("Bit-Plane").** Cette méthode est la plus simple des méthodes de transmission progressive : les pixels de l'image sont directement quantifiés. Les 256 niveaux d'intensité peuvent donc être transmis en 8 étapes. La transmission des bits s'effectue du bit de poids le plus fort vers le bit de poids le plus faible. Son principal inconvénient est un débit minimum de 1bpp pour chaque couche.

- **Quantification vectorielle ("Vector Quantization").** Les mots du code sont triés par similarité, en utilisant un arbre ou un dictionnaire. Selon la méthode de transmission des mots du code, il est possible d'obtenir une reconstruction progressive en précision ou en résolution. Soit les mots sont transmis complètement seulement dans certaines zones de l'image, soit les mots sont progressivement reconstruits en partant de certains mots simples et en parcourant l'arbre ou le dictionnaire pour aller vers les mots plus complexes.
- **Codage en banc de filtres ("Filter-Bank Coders").** L'image est décomposée en ondelettes, puis les coefficients sont codés sous forme d'arbre hiérarchique. Il est possible d'interrompre la transmission à n'importe quel moment et d'effectuer une reconstruction à ce stade. De nombreux travaux portent sur ce type d'approche, où les bits des coefficients sont transmis progressivement, comme dans [Chang et al., 2008] ou [Garcia et al., 2005], ou comme pour SPIHT (avec un algorithme pour la correction d'erreur : [Hwang et al., 1999]) et JPEG 2000. Ce type d'approche permet également de compresser l'image transmise. L'objectif est de reconstruire l'image à la meilleure précision possible pour un bitrate donné. Avec ces méthodes, il est toutefois nécessaire de reconstruire toute l'image à chaque étape intermédiaire pour prendre en compte les nouveaux coefficients.
- **Codage par blocs ("Block-Transform Coders").** Le principe est similaire à celui exposé précédemment, mais une transformée en cosinus discrète (DCT) est réalisée à la place d'une transformée en ondelettes. Cette approche est implémentée dans JPEG notamment. En plus du coût de chaque reconstruction intermédiaire dû au fait que chaque bloc doit être recalculé en fonction des nouveaux coefficients, il peut aussi y avoir des artéfacts (effets de blocs) pour des bitrates faibles.

b. **Les méthodes par séquence de transmission ("Transmission-Sequence-based Coding")**

Les données sont d'abord séparées en différents groupes, qui sont ensuite ordonnés selon leur importance avant d'être transmis. La difficulté est d'équilibrer la quantité d'information à ajouter aux données pour les identifier afin de les réordonner : si toutes les données sont identifiées séparément, alors le surcoût par rapport à l'image

originale est trop important ; et sinon, la reconstruction risque de ne pas être assez progressive pour présenter un intérêt.

- **"Block-Transform Coders" et "Filter-Bank Coders".** Les blocs issus de la DCT de l'image ou les coefficients de la transformée en ondelettes sont réordonnés, soit de façon systématique (indépendamment de l'image), soit en fonction de l'image ou de paramètres statistiques. Par rapport à une méthode par approximations successives, où les coefficients sont affinés à chaque itération, il s'agit cette fois de réordonner les coefficients avant de les transmettre, par exemple par fréquence.
- **Codage par blocs de l'espace ("Block-Based Spatial Domain Coders").** L'image est segmentée en blocs, pouvant être identifiés et ordonnés en fonction de leur contenu : la variance entre les pixels, les contours, la texture, ... La principale difficulté résidant précisément dans la reconnaissance des différents types de blocs.
- **Codage par segmentation ("Segmented Image Coders").** L'image est segmentée en régions homogènes. Cette approche permet de conserver les contours, même à bas débits, et offre également des taux de compression intéressants, mais le temps de calcul reste important et les performances sont comparables à celles obtenues avec les transformées en ondelettes et DCT.

c. Les méthodes de codage résiduel multi-échelles ("Multistage Residual Coding")

L'image est encodée grossièrement puis transmise. Une image d'erreur résiduelle est obtenue en soustrayant l'image reconstruite de l'image originale. Cette "image d'erreur" est encodée à son tour puis transmise. Le codage utilisé peut changer entre chaque itération. L'erreur entre l'image originale et l'image reconstruite diminue à chaque itération et plus la transmission progresse, plus l'entropie de l'image résiduelle décroit. Deux codeurs ont été principalement étudiés dans la littérature :

- **Codage par quantification vectorielle ("VQ-Based Coders").**
- **"Block-Transform Coders".** A noter que cette transformée fonctionne mieux sur les images naturelles, et par conséquent est peu efficace pour coder l'image d'erreur résiduelle.

d. Les méthodes par codage hiérarchique ("Hierarchical Coders")

L'image est séparée en niveaux hiérarchiques. Contrairement à la décomposition obtenue avec une approche de PIT par segmentation ou basée sur la séquence, il s'agit de niveaux de décomposition de l'image, qui peuvent être traités indépendamment. Une image intermédiaire peut être reconstruite à chaque niveau de décomposition, ce qui est utile dans le cas d'un environnement où différents types d'affichages (avec différentes résolutions) cohabitent.

- **Codage Hiérarchique ("Nonresidual Hierarchical Coders").** L'image est subdivisée récursivement jusqu'à obtenir des régions vérifiant un critère d'homogénéité. La structure hiérarchique est souvent un arbre (binaire, quadtree, ...) et peut être codée indépendamment des régions. Les reconstructions intermédiaires sont généralement peu intéressantes du fait de la simplification de nombreux pixels par une seule valeur.
- **Codage Multi-échelles ("Residual Multiscale Coders").** En plus d'utiliser une structure hiérarchique, ces méthodes incluent la construction d'une image résiduelle entre chaque niveau dans le but d'optimiser l'efficacité du codage ou les reconstructions intermédiaires.
- **"Filter-Bank Coders".** Une reconstruction est associée à chaque niveau de décomposition. La transmission s'effectue alors à partir de l'image basse-résolution, puis en ajoutant un niveau de détail à chaque étape.

Comme on peut le voir, il existe de nombreuses approches : certaines approches se focalisent sur la compression de l'image, tandis que d'autres permettent plus de flexibilité dans les reconstructions intermédiaires, permettent un codage/décodage rapide de l'image ou encore effectuent des traitements avant transmission pouvant être réutilisés par la suite (la segmentation dans le cas d'un codage par segmentation par exemple). Le critère de comparaison ne peut donc pas se réduire seulement à la qualité de l'image reconstruite. Notre approche se focalise sur la reconstruction de l'image en augmentant progressivement la résolution comme le permettent les approches par approximations successives (JPEG 2000, codage vectoriel). Elle partage également les avantages des codeurs multi-échelles hiérarchiques, *i.e.* , dans un environnement de dispositifs d'affichage non-homogène, l'image

peut être lue à différentes résolutions sans réencodage et stockée sous une seule forme ; de plus l'image finale est exactement reconstruite. Plus généralement, elle est similaire à l'approche dite "naïve", qui consiste à transmettre l'image un pixel sur quatre, puis un pixel sur deux et ainsi de suite. Néanmoins, notre algorithme est caractérisé par sa flexibilité : n'importe quel nombre d'image intermédiaires peut être transmis, à n'importe quelle résolution, et avec une quantité de données transmises constante pour reconstruire l'image originale sans erreur. De plus, avec cet algorithme, les données originales sont converties en données monodimensionnelles avant la transmission ce qui implique que n'importe quelles données de dimensions quelconques peuvent être décomposées et transmises progressivement. Enfin, grâce à la nature de la décomposition par le KST qui fait perdre l'information de voisinage, notre approche est beaucoup moins sensible à la perte de paquets.

4.1.2 Transmission progressive avec KSN

Nous présentons dans cette partie les modifications apportées à l'algorithme d'Igelnik et Parikh pour permettre un schéma de transmission progressive (en résolution) d'images. L'idée générale consiste à construire un pavage multirésolution tout en maintenant la cohérence entre les fonctions internes et les fonctions externes. Plus précisemment, la principale modification concerne la construction des fonctions internes, afin qu'une fonction externe à une résolution donnée puisse être construite à partir des fonctions externes des résolutions inférieures.

Tout d'abord, la dernière couche du réseau (la couche d'indice $n, n = N$) est construite avec un pavage haute densité contenant un pixel par pavé et ses fonctions monodimensionnelles associées, ce qui correspond à la décomposition exacte de l'image. Les couches intermédiaires (les couches d'indices $n, n < N$) sont ensuite générées en utilisant des pavés plus gros, tels que les fonctions externes obtenues soient des sous-échantillonnages de la fonctions externe générée pour le pavage haute densité. Autrement dit, un pavage est généré par couche avec une densité croissante, *i.e.* la taille des pavés diminue quand l'indice n de la couche croît. Pour obtenir ce sous-échantillonnage de la fonction externe, les fonctions internes associées aux pavages intermédiaires doivent utiliser les valeurs des paliers générés

dans les fonctions internes du pavage haute densité : le réel associé à toute position de l'espace $2D$ doit être le même pour toutes les couches dont un hypercube recouvre cette position. Ainsi, avec la fonction externe correspondant à chaque couche intermédiaire, une reconstruction partielle de l'image est obtenue, et la résolution de l'image reconstruite augmente progressivement jusqu'à reconstruire parfaitement l'image originale. De plus, la quantité de données à transmettre peut être ajustée en modifiant la densité du pavage et le nombre des couches intermédiaires. La Figure 4.1 présente un exemple d'un réseau constitué de cinq couches de pavages.

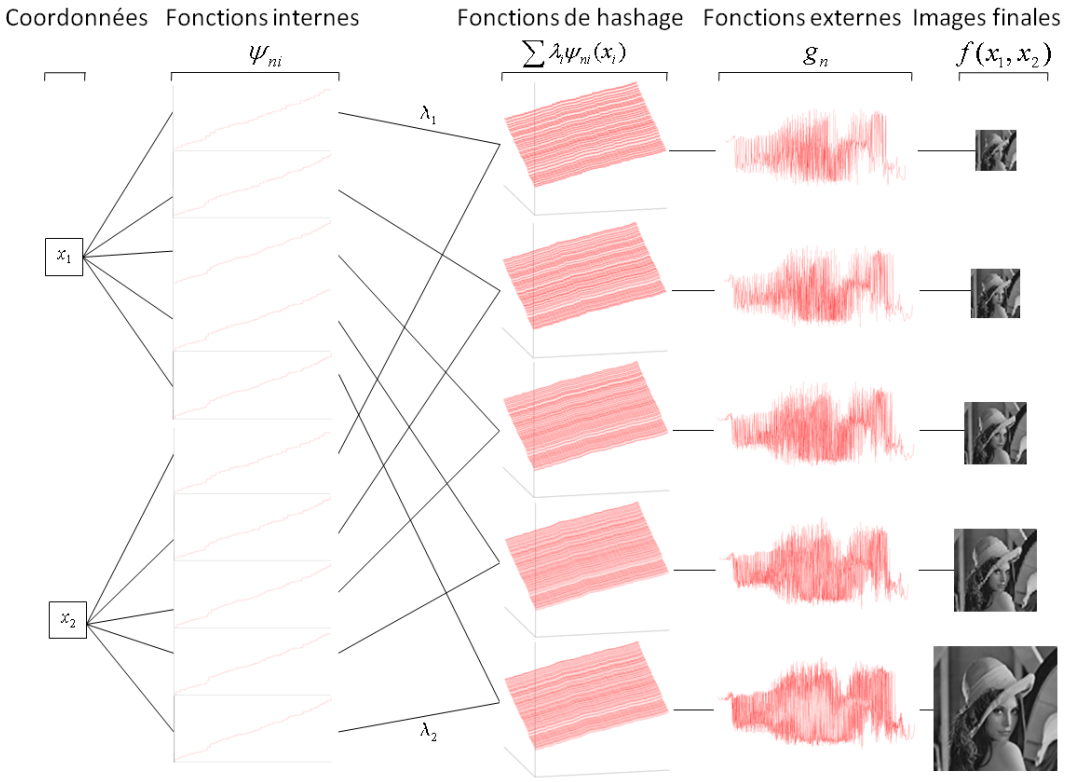


Figure 4.1 – KSN modifié pour la transmission progressive, constitué de cinq couches de pavage. La fonction externe est itérativement reconstruite en ajoutant des points à celle de la couche précédente, et les fonctions internes sont ajustées en ajoutant des paliers à ceux des couches précédentes lorsque la densité du pavage augmente.

La définition du pavage (eq. 3) est modifiée pour prendre en compte l'indice de la couche (n). Le pavage est quant à lui toujours obtenu en réalisant le produit cartésien des intervalles.

$$\forall n \in [\![1, N]\!],\ j \in \left\{\mathbb{N} \cup \{-1\}\right\},\ I_n(j) = \left[(N+1)j\delta_n, (N+1)j\delta_n + N\delta_n\right]$$

On note que δ a été remplacé par δ_n, ce qui signifie que la distance entre entre les hypercubes ainsi que leur taille varient pour chaque couche. La translation des couches les unes par rapport aux autres est compensée par le changement de la taille du pavage, et dans tous les cas, la dernière couche contient tous les pixels de l'image. Le terme $(n-1)\delta$ présent dans la définition originale et correspondant à la translation n'est donc plus nécessaire. Avec cette définition, la taille des intervalles peut être modifiée pour chaque couche, en vérifiant :

$$\forall n_1, n_2 \in [\![1, N]\!],\ n_1 < n_2 \Rightarrow \delta_{n_1} > \delta_{n_2}$$

La densité augmente donc avec le nombre de couches, la dernière couche étant toujours la couche haute densité décomposant exactement l'image. La Figure 4.2 montre un exemple de superposition de couches vérifiant cette nouvelle définition.

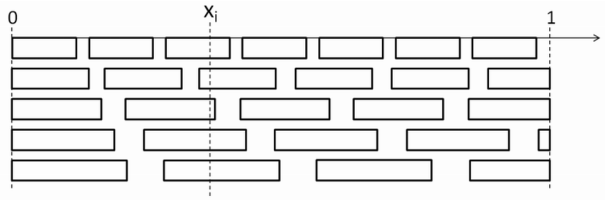


Figure 4.2 – Superposition de pavages disjoints, dont la densité varie en fonction de la couche.

4.1.2.1 Construction des fonctions internes ψ_{ni}

Au vu des modifications apportées à la construction du pavage, la taille des paliers des fonctions ψ_{ni} va diminuer lorsque n augmente. Les fonctions internes associées à la couche N sont construites en premier, de telle façon qu'il y ait un palier par pixel. Les fonctions internes associées aux couches intermédiaires $n, n < N$, sont générées en sous-échantillonnant les fonctions internes de la couche N : à chaque intervalle $I_n(j)$, on associe le réel $y_{Nij'}$ (image de l'intervalle $I_n(j)$ par la fonction ψ_{ni}) correspondant à un palier de ψ_{Ni}. j' est tel que le centre de l'intervalle $I_n(j)$ appartienne à l'intervalle $I_N(j')$:

$$\begin{aligned} &\forall n \in [\![1, N-1]\!], \forall j \in \left\{\mathbb{N} \cup \{-1\}\right\}, \\ &si\ \exists j' \in \left\{\mathbb{N} \cup \{-1\}\right\}\ tel\ que\ (N+1)j\delta_n + \frac{\delta_n}{2} \in I_n(j'),\ alors\ y_{nij} = y_{Nij'} \end{aligned} \tag{4.1}$$

On peut déduire de cette condition et en observant la Figure 4.2 que lorsque la taille des pavés change rapidement d'une couche à l'autre, tous les points de la fonction externe précédent ne sont pas nécessairement utilisés par la dernière couche transmise. Par contre, tous les points transmis sont utilisés dans la dernière couche pour la reconstruction haute résolution. Par ailleurs, aucun point n'est transmis en double puisque par construction, les hauteurs des paliers de la fonction ξ_n sont toutes différentes.

La suite de l'algorithme pour la construction des fonctions internes n'est pas modifiée : les fonctions sont échantillonnées régulièrement, et ces points sont ensuite interpolés par des splines cubiques.

4.1.2.2 Construction des fonctions externes g_n

La seule modification pour la construction des fonctions externes concerne la valeur attribuée au centre des hypercubes. Autrement dit, l'équation 2.7 est remplacée par

$$\forall i \in [\![1, d]\!], \forall n \in [\![1, N]\!], \forall j \in \left\{\mathbb{N} \cup \{-1\}\right\},\ g_n(p_{nj_1,\dots,j_d}) = f((x_{C_{nj_1,\dots,j_d}}, y_{C_{nj_1,\dots,j_d}})) \tag{4.2}$$

Les valeur de la fonction g_n sont égales aux valeurs de la fonction f aux centres des hypercubes, et non plus au N^{e} comme dans le schéma original.

Le reste de l'algorithme, notamment la définition de g_n pour $\forall t \in [0, 1]$, n'est pas modifié.

Avec cette nouvelle construction, chaque couche permet de calculer une approximation de l'image à une résolution donnée, et la contribution de chaque couche se traduit par un ajout de points dans la fonction externe. De plus, les fonctions internes sont construites en sous-échantillonnant la décomposition exacte de l'image, l'étape d'optimisation est donc supprimée.

La Figure 4.3 illustre la construction progressive de la fonction externe pour un réseau à trois couches. La première couche (4.3(a)) contient trois points. Avec la deuxième couche ((4.3(b))), quatre points (triangles) sont ajoutés aux points déjà existants de la fonction externe (cercles). Enfin, avec la troisième et dernière couche (4.3(c)), 5 points (carrés)

sont ajoutés à la fonction externe. On peut noter que les nouveaux points sont ajoutés à n'importe quelle position dans la fonction externe, et pas nécessairement entre des points déjà transmis.

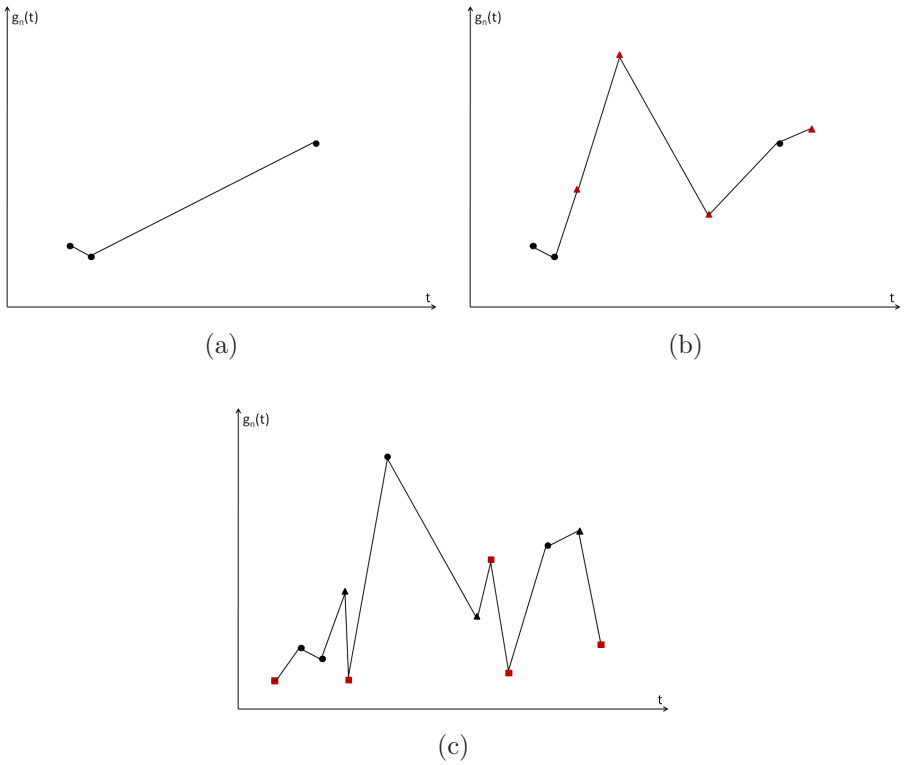


Figure 4.3 – Construction progressive de la fonction externe. (a) Fonction externe obtenue avec les données de la première couche. (b) Fonction externe obtenue avec les données de la première et de la deuxième couche (triangles). (c) Fonction externe obtenue avec les données des trois couches (carrés = données de la dernière couche).

Pour transmettre l'image progressivement, il faut (et il suffit de) transmettre les fonctions monodimensionnelles composant le réseau, obtenues lors de la décomposition de l'image :

– Les fonctions internes sont générées aléatoirement et n'ont donc pas besoin d'être transmises. Pour justifier cela, il y a deux possibilités : soit l'émetteur et le receveur ont le même générateur de nombres aléatoires ; soit on suppose avoir construit les fonctions internes pour le pavage haute résolution une fois pour toute, puisque que comme dans l'énoncé original du TSK 1, elles sont indépendantes de l'image.
– Les fonctions externes sont continues par construction. Toutefois, il n'y pas de problème de discrétisation, car elles sont complètement définies par les points de l'ensemble $\mathscr{S}$. Ces points sont obtenus pour les valeurs aux centres des hypercubes. En sachant que la

couche de densité maximum est telle que chaque pixel soit couvert par un hypercube, le nombre de points de l'ensemble $\mathscr{S}$ à transmettre est le même que le nombre de pixels de l'image originale.

4.1.3 Résultats

Chaque couche d'indice $n, n < N$, du réseau est constituée d'un pavage généré à partir des paramètres N (le nombre de couches total du réseau) et δ_n (l'intervalle entre deux intervalles consécutifs et l'unité de longueur d'un intervalle, la longueur étant $N \times \delta_n$). Le nombre de couches total(N) détermine le nombre de reconstructions intermédiaires et peut être ajusté librement. Un δ_n peut être choisi pour chaque couche, ce qui permet d'adapter les résolutions des reconstructions intermédiaires.

Nous illustrons notre approche avec plusieurs exemples d'images transmises progressivement. La Figure 4.4 illustre la transmission progressive en cinq étapes pour une image de 200×200 pixels. La Figure 4.5 présente une décomposition similaire pour les images Peppers et Mandrill. Dans tous les cas, on peut voir que l'image finale ne présente pas d'artéfacts et que la décomposition est indépendante de l'image considérée.

La Figure 4.6 présente deux exemples de transmissions progressives, avec 2 et 6 reconstructions intermédiaires. La première transmission utilise un réseau à trois couches : une première image de 10×10 pixels est construite avec la première couche, puis 100×100 pixels avec la deuxième, et finalement la troisième couche permet de reconstruire l'image à sa résolution d'origine. La seconde transmission utilise un réseau à 7 couches. Dans les deux cas, l'image finale est identique à l'image initiale, mais la quantité de données transmise à chaque étape a changé, de deux grands ensembles de points pour la première transmission à huit petits ensembles de tailles presque identiques pour la deuxième.

Le tableau 4.1 présente le nombre de points ajoutés à la fonction externe pour chaque couche, dans le cas d'une transmission progressive d'une image de 200×200 pixels. On peut noter que le nombre de points contenus dans les fonctions externes augmente après chaque couche transmise et avec la résolution de l'image reconstruite, mais le nombre de

points total reste constant et dépend de la résolution de l'image à transmettre. Les 500 points supplémentaires par rapport au nombre de pixels de l'image sont dûs à l'algorithme qui utilise parfois les pixels du bord de l'image plusieurs fois.

	Nombre de points ajoutés dans la fonction externe										
N	36×36	40×40	44×44	50×50	57×57	67×67	80×80	100×100	133×133	200×200	total
10	1.4	1.6	1.9	2.3	2.8	3.5	4.6	6.1	7.8	8.5	40.5
9	/	1.7	1.9	2.4	2.8	3.8	4.8	6.3	8.0	8.8	40.5
8	/	/	2.0	2.5	3.0	3.9	4.7	6.6	8.5	9.3	40.5
7	/	/	/	2.6	3.2	3.8	5.1	7.0	9.5	9.3	40.5
6	/	/	/	/	3.4	4.0	5.5	7.0	10.5	10.1	40.5
5	/	/	/	/	/	4.6	5.8	7.6	11.3	11.2	40.5
4	/	/	/	/	/	/	6.6	8.6	11.3	14.0	40.5
3	/	/	/	/	/	/	/	10.2	13.6	16.7	40.5

TABLE 4.1 – Influence des reconstructions intermédiaires sur le nombre de points ($\times10^3$) à transmettre pour des réseaux contenant jusqu'à 10 couches et pour une image de 200×200 pixels.

4.2 Résistance au bruit

Comme nous l'avons vu dans la section 4.1.1, il existe de nombreux types de transmissions progressives, notamment caractérisées par leur flexibilité, leur complexité algorithmique, et/ou leur qualité de reconstruction. Le critère de comparaison communément admis est le ratio qualité de reconstruction/débit. Toutefois, la plupart de ces approches sont basées sur une approche multirésolution, et transmettent progressivement les bits correspondant à l'encodage des coefficients, ce qui permet de construire des images intermédiaires avec une précision croissante mais à la résolution de l'image d'origine. Dans notre approche, la transmission progressive est effectuée dans l'espace des fonctions monodimensionnelles, donc les images intermédiaires sont construites à des résolutions inférieures à l'image originale, en utilisant les données déjà transmises. Par conséquent, on ne peut pas comparer ces reconstructions intermédiaires à l'image originale directement. Plus précisément, il serait nécessaire d'augmenter la résolution des images intermédiaires, et donc d'introduire une méthode d'interpolation, ce qui pose problème : l'évaluation de la qualité porterait alors sur la méthode d'interpolation utilisée et non plus sur notre approche. Il serait également possible d'utiliser l'algorithme pour reconstruire l'image à sa résolution d'origine, toutefois,

l'interpolation obtenue par la lecture des fonctions externes à un échantillonnage différent de celui des pavages contenus dans le réseau est mauvaise. En effet, la construction des fonctions internes est indépendante de l'image, autrement dit la relation de voisinage des pixels dans la fonction externe est totalement aléatoire.
Un critère de comparaison commun à notre approche et aux approches existantes est la résistance au bruit, c'est pourquoi nous proposons dans cette section d'étudier la résistance au bruit de notre approche.

4.2.1 Etat de l'art

Dans la littérature, la robustesse de la transmission est généralement évaluée en simulant un canal de transmission. La problématique n'est pas nécessairement liée celle de la transmission progressive, les simulations portant généralement sur des transmissions à un bitrate donné. Les canaux couramment utilisés pour la simulation peuvent se séparer en trois catégories :

- **Binary Symmetric Channel**. Il s'agit d'un canal très répandu pour l'étude des codes détecteurs et correcteurs d'erreurs. Ce canal simule l'échange de certains bits du message pendant la transmission avec une probabilité fixée. Dans [Stankovic et al., 2003], les auteurs utilisent ce canal pour étudier les performances d'une méthode temps-réel de protection des erreurs pour deux encodeurs (SPIHT et JPEG2000). Leur approche s'adapte en fonction du débit et permet d'améliorer la quantité de bits corrects reçus lorsque le bitrate est inférieur à celui de l'encodage optimal de l'image.
- **Packet-Loss Channel**. Les données transmises sont regroupées sous forme de paquets, ayant une probabilité d'être altérés ou perdus. La taille des paquets et le type de données transmises permet de simuler des canaux de différents débits. Dans [Li and Cai, 2007], [Kim et al., 2003], et [Grangetto et al., 2004], différentes approches sont proposées pour la transmission d'images encodées avec JPEG 2000 et SPIHT sur des canaux de type Packet-Loss.
  - Dans [Li and Cai, 2007], les auteurs adaptent un algorithme de correction d'erreurs pour la transmission progressive à JPEG 2000, afin de décoder de façon optimale des données endommagées pendant la transmission.

  - [Kim et al., 2003] s'appuient sur la transmission de plan de bits obtenus par compression avec SPIHT et JPEG 2000. Il s'agit d'anticiper la perte de données en ajoutant de la redondance lors de la transmission.
  - [Grangetto et al., 2004] proposent un algorithme visant à assurer une qualité de reconstruction minimum spécifiée par l'utilisateur pour les images encodées avec JPEG 2000 et SPIHT.
  - Dans [Charfi et al., 2003], les auteurs abordent la transmission sur des canaux de types "Binary Symmetric" et "Packet-Loss" avec des contraintes de temps-réel. Leurs contributions portent sur l'amélioration et l'optimisation des codeurs ondelettes JPEG 2000 et SPIHT pour la correction des erreurs après réception, mais sans prendre en compte une transmission progressive.

  Dans toutes ces approches, il s'agit de compléter des algorithmes de compression existants afin d'ajouter les fonctionnalités de contrôle et de corrections des erreurs.
- **Bit-Error Channel**. Ce canal permet de simuler des interférences pendant une transmission sur un canal physique : les données transmises sont altérées avec un bruit blanc gaussien. La fidélité des données après transmission est évaluée à un bitrate donné simulant le débit de transmission maximum du canal. Ce canal peut être combiné avec des pertes de paquets, par exemple pour simuler une connexion WiFi.
  - Un cas de canal mixte Bit-Error et Packet-Loss est étudié dans [Cosman et al., 2000]. L'approche repose sur un codage hiérarchique de transformée en ondelettes. Les auteurs proposent une méthode hybride empruntant à deux approches classiques : ajouter des données supplémentaires lors de la transmission et modifier la structure de ces données.
  - Dans [Chande and Farvardin, 2000], les auteurs présentent également une approche pour la transmission sur des canaux de types Bit-Error Channel ou Packet-Loss Channel (mais pas les deux à la fois). Ils proposent un algorithme permettant de sélectionner et adapter des codes correcteurs existants pour un bitrate donné, et ceci pour des gammes de bitrates intermédiaires si les codes le permettent.

Les approches développées dans la littérature fonctionnent généralement à partir d'une méthode de transmission existante à laquelle est ajoutée une méthode permettant de détecter et de restaurer les erreurs. Pour les méthodes cherchant à développer le contrôle et

la correction des erreurs pour la transmission progressive, la problématique réside alors dans l'adaptation du code pour des bitrates variables, afin de conserver de bonnes performances dans le plus de configurations possibles. Notre approche est différente dans le sens où la résistance aux erreurs découle de la nature de la décomposition en $1D$. Les pixels de l'image sont réordonnés aléatoirement ce qui permet notamment de réduire l'impact d'erreurs consécutives. Il n'est pas nécessaire d'ajouter de la redondance aux données à transmettre ou d'effectuer des calculs préliminaires à la transmission en dehors de la décomposition.

4.2.2 Résistance de la transmission progressive par KSN

Notre approche offre peu d'intérêt lors de la transmission sur des canaux de types "Binary Symmetric" et "Bit-Error" par rapport à une transmission de l'image pixel par pixel. En effet, toute modification sur les points de la fonction externe se répercute sur la valeur des pixels lors de la reconstruction de l'image. Néanmoins, dans le cas où le parcours de l'image serait adapté, il est alors possible d'envisager des relations connues entre les points dans la fonction externe, ce qui permettrait de contrôler et de restaurer les données. Si le parcours de l'image est tel que l'organisation des points de la fonction externe se répète selon un même motif, il est même possible de coder la fonction externe avec un codage vectoriel de façon efficace.

Pour démontrer la tolérance aux erreurs de notre approche, nous simulons la transmission des points de la fonction externe par un canal de type packet-loss. Les paquets perdus sont remplacés par du bruit (valeurs aléatoires de niveaux de gris).
Dans la littérature, ainsi que dans les différents protocoles de transmission, la taille des paquets est très variable. Dans notre approche, la taille des paquets ne modifie pas le fonctionnement de l'algorithme, seul le nombre total de points perdus a un impact : plus il y a de points perdus, plus l'image reconstruite contiendra des pixels à la valeur aléatoire. Nous illustrons le comportement de notre algorithme pour différentes tailles de paquets et différentes probabilités de pertes. La résolution de l'image à transmettre sera toujours fixée à 200×200 pixels pour permettre une meilleure comparaison entre tous les résultats,

mais notre approche est applicable à n'importe quelle autre résolution. La qualité de la reconstruction est évaluée par un calcul de PSNR. On remarque que ce calcul ne traduit pas nécessairement la cohérence spatiale de l'image après reconstruction, c'est pourquoi nous illustrons les résultats importants par les images obtenues.

La Figure 4.7 montre un exemple de reconstruction obtenue après la perte de plusieurs paquets. Les paquets contiennent 50 points et la probabilité d'une perte est de 5%. Dans cet exemple, 3 paquets ont été perdus pendant la transmission de la première couche, 9 pendant la seconde, 6 pendant la troisième, 12 pendant la quatrième et enfin 7 pendant la dernière. Au total 1850 points sont manquants dans la fonction externe. On observe que les erreurs sont réparties sur toute l'image, ce qui signifie que l'application d'une méthode de débruitage simple (tel qu'un filtre médian) permet de restaurer l'aspect global de l'image.

Le tableau 4.2 offre une vue synthétique de la qualité des reconstructions après restauration avec un filtre médian 3×3. On observe que notre approche est très peu sensible à la taille des paquets ou aux taux de pertes inférieurs à 10% : la qualité de la reconstruction est toujours maintenue à environ 27dB. La figure 4.8 illustre la meilleure reconstruction à 28.28dB et la moins bonne à 23.71dB. Dans les deux cas l'aspect global de l'image est préservé.

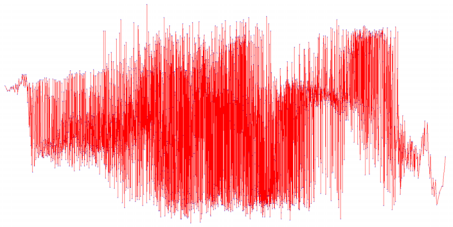
Taille d'un paquet(points)	Probabilité de perte d'un paquet				
	0.01	0.02	0.05	0.1	0.2
50	28.28	28.26	27.63	27.32	25.38
100	28.27	28.12	27.52	27.24	24.53
200	28.03	27.98	27.76	27.38	23.71

TABLE 4.2 – PSNR (dB) de l'image après restauration et transmission en fonction de la taille des paquets (points) et de la probabilité de perdre un paquet.

La Figure 4.9(a) montre la reconstruction en utilisant des paquets de 2000 points lors de la transmission. La reconstruction après un filtrage médian est proposée sur la Figure 4.9(b). On peut noter qu'aucune zone de l'image n'est perdue, par contre les détails sont moins précis, à cause du filtrage. A titre de comparaison, si l'image était transmise ligne par ligne, la perte d'un de ces paquets serait équivalente à la perte de 10 lignes consécutives (voir 4.9(c)). La tentative de restauration avec un filtrage médian est présentée Figure 4.9(d).

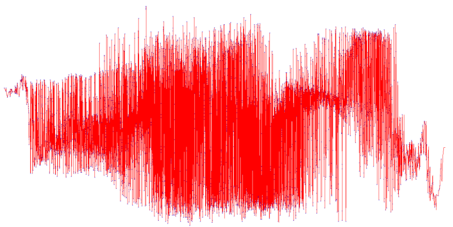
Cette fois, il n'est pas possible de restaurer les données contenues dans ces 10 lignes, même en considérant un filtre de plus grande taille.

La Figure 4.10 présente la transmission de notre image dans le cas le plus défavorable. La taille des paquets est de 100 points et tous les paquets contenant la première couche ne sont pas transmis. Par la suite, un paquet est perdu pendant la transmission de chaque couche. Contrairement à transmission par plan de bits, où la perte du plan des bits de poids forts empêche de reconstruire une image, tous les paquets ont la même importance dans notre approche.



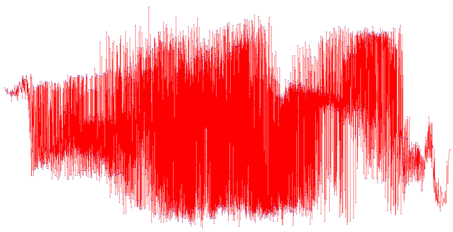
(a) g_1

(b)
40×40



(c) g_2

(d)
50×50

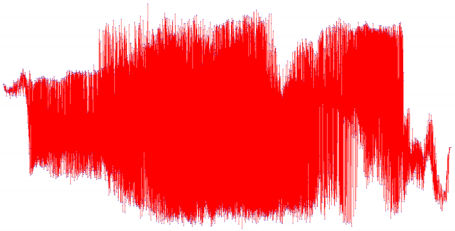


(e) g_3

(f) 67×67

(g) g_4

(h) 100×100



(i) g_5

(j) 200×200

Figure 4.4 – (a-b)(c-d)(e-f)(g-h)(i-j) Les fonctions externes et leurs reconstructions intermédiaires associées pour les cinq couches du réseau (respectivement).

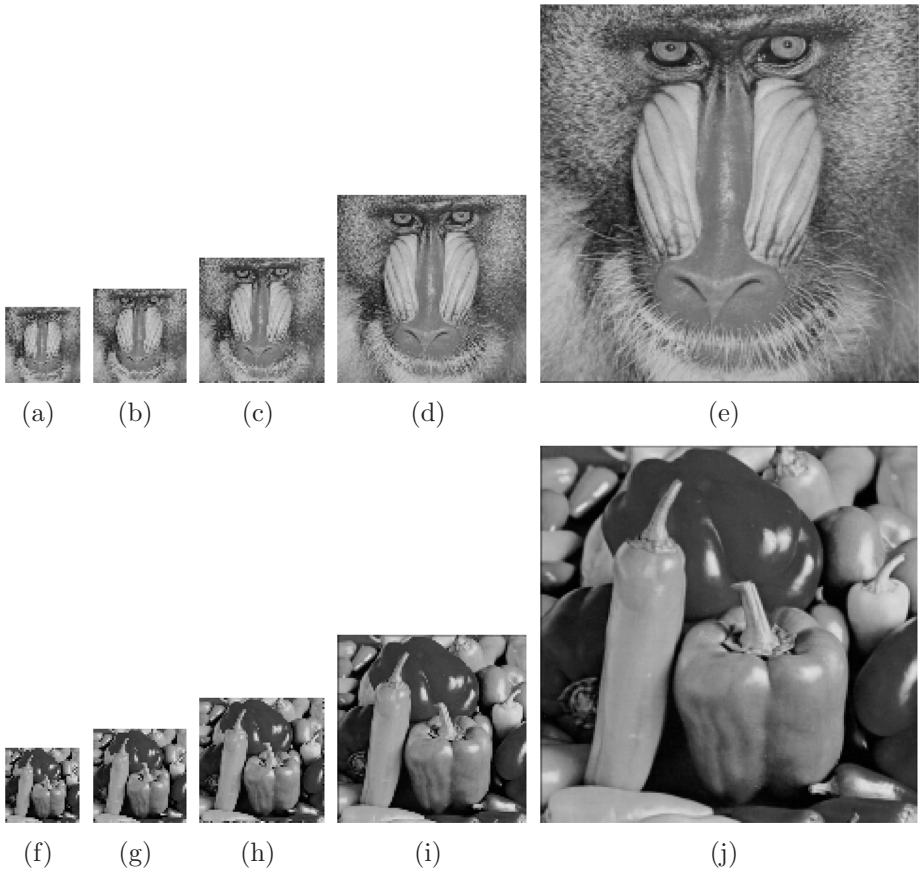
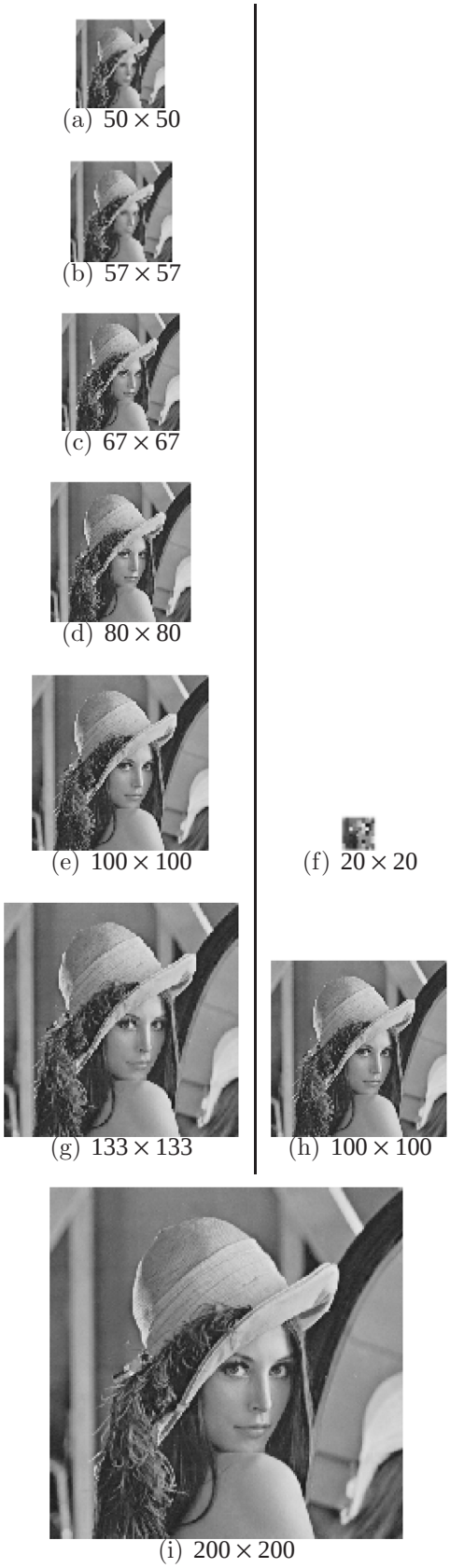
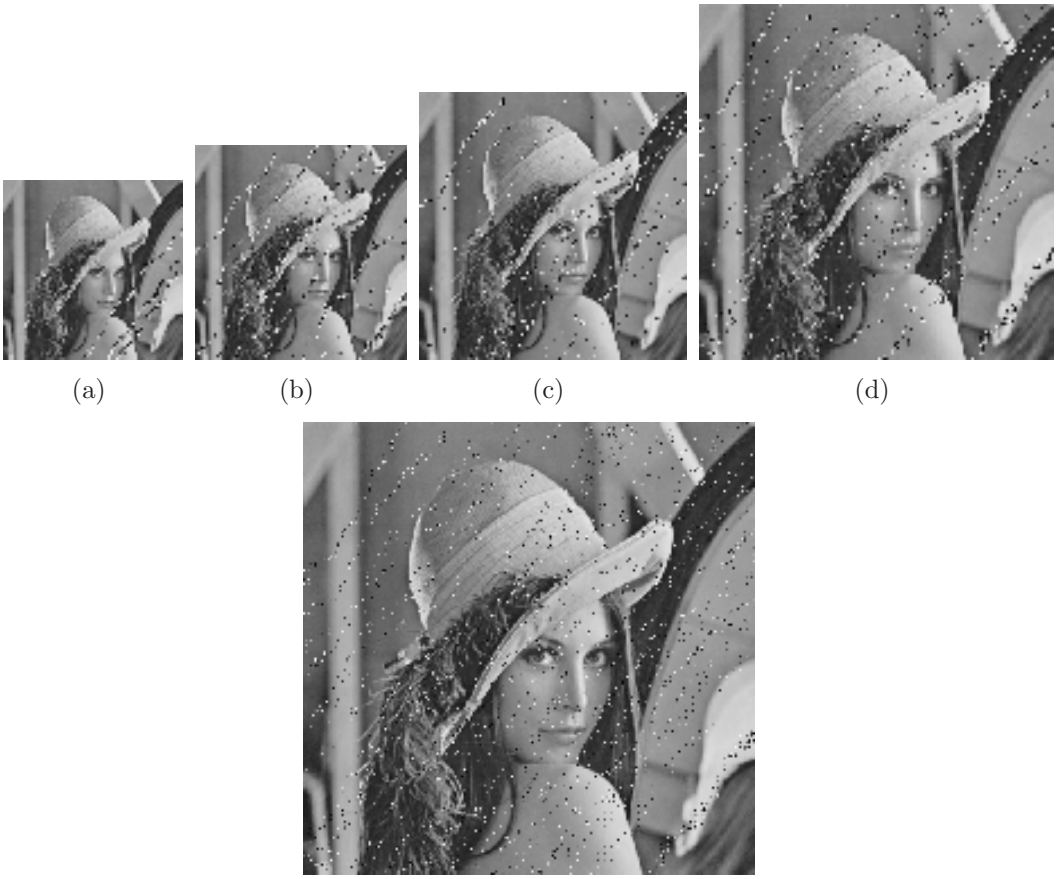


Figure 4.5 – Reconstruction progressive de 2 images par 4 niveaux de transmission intermédiaires : Mandrill (a-e) et Peppers (f-j).



(a) 50×50 (b) 57×57 (c) 67×67 (d) 80×80 (e) 100×100 (f) 20×20 (g) 133×133 (h) 100×100 (i) 200×200

Figure 4.6 – Construction progressive de Lena (i) avec 6 (a-g), ou 2 (f-h) étapes intermédiaires.



(a) (b) (c) (d)

(e)

Figure 4.7 – Reconstruction progressive de Lena avec pertes de données. 3, 9, 6, 12, 7 paquets perdus pour les couches (a)(b)(c)(d)(e) respectivement. Chaque paquet contient 50 points.

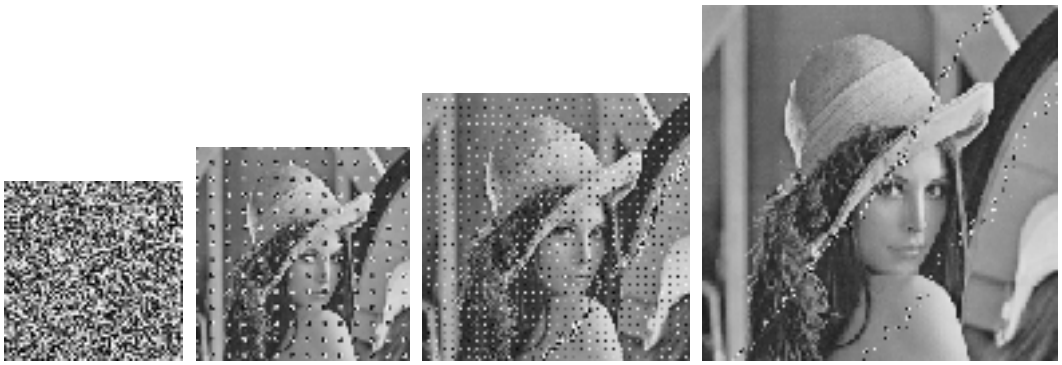
(a) (b)

Figure 4.8 – Restauration de Lena par un filtre médian 3×3 après transmission (a) par paquets de 50 points et 1% de pertes, (b) par paquets de 200 points et 20% de pertes.

(a) (b)

(c) (d)

Figure 4.9 – (a) Reconstruction après la perte 2 paquets, contenant chacun 2000 points. (b) Restauration de (a) avec un filtre médian 3×3. (c) Perte équivalente à (a) (2 fois 2000 pixels consécutifs perdus), mais dans le cas d'une transmission ligne par ligne. (d) Filtrage de (c) avec un filtre médian 3×3.



(a) (b) (c) (d)

(e) (f)

Figure 4.10 – (a-e) Reconstruction progressive de Lena avec pertes de paquets : la première couche (a) n'est pas transmise, un paquet de 100 points est perdu pendant la transmission de chaque couche restante (b-e). (f) Restauration de (e) à l'aide d'un filtre médian 3×3.

4.3 Conclusion

Nous avons présenté une version modifiée de l'algorithme d'Igelnik pour la transmission progressive (en résolution) des images en niveaux de gris. Cet algorithme décompose les fonctions multidimensionnelles en un ensemble (ou réseau) de splines. En adaptant l'algorithme pour pouvoir modifier la densité du pavage de chaque couche indépendamment, il est possible de reconstruire une image progressivement jusqu'à sa résolution initiale. De plus, un nombre quelconque d'images intermédiaires peut être construit, à n'importe quelles résolutions inférieures à l'image d'origine. Dans tous les cas, la quantité de données à transmettre est constante et dépend uniquement de la résolution de l'image originale. Enfin, la reconstruction finale est fidèle à l'image originale, sans erreur ni artéfact. Notre approche est caractérisée par sa robustesse aux erreurs de transmission dans le cas d'un canal avec pertes de paquets : quels que soient le nombre et la taille des paquets, les erreurs sont diffusées sur toute l'image. En effet, les pixels de l'image sont réordonnés aléatoirement dans les fonctions externes transmises. Autrement dit, les erreurs survenues durant la transmission ne sont pas concentrées localement, donc l'aspect général peut être restauré par des techniques de débruitage classiques, au prix d'une perte de détails.

Nous avons illustré notre approche pour la transmission adaptable et progressive d'images, comme des fonctions $2D$. L'élément clé de cette approche réside dans la transmission progressive, qui est effectuée dans l'espace $1D$ des fonctions externes. Plus généralement, notre algorithme permet de décomposer tout signal nD en fonctions $1D$. Autrement dit, il est possible d'imaginer l'extension de cette approche dans des dimensions supérieures, par exemple pour la transmission de maillages $3D$ texturés. Par ailleurs, le balayage de l'image induit par les fonctions internes est nécessaire à la reconstruction des données de la fonction externe. On peut donc imaginer une transmission cryptée où la clé serait constituée des paliers des fonctions internes qui pourraient également servir de signature. Enfin, ce cryptage peut être combiné avec les modifications proposées ici pour la transmission progressive, afin d'obtenir une transmission progressive et sécurisée de signaux multidimensionnels.

Conclusion générale et perspectives

Conclusion

Nous avons présenté deux adaptations de l'algorithme proposé par Igelnik et Parikh, pour la compression d'image et pour la transmission progressive. Nous avons montré que la combinaison de la décomposition d'images de transformée en ondelettes en fonctions monodimensionnelles, suivie de simplifications dans cet espace $1D$, permet de compresser efficacement ces images. Nous avons également présenté l'intégration de ce schéma dans JPEG 2000. Bien qu'expérimentale et non optimale, notamment à cause des contraintes imposées par JPEG 2000, cette méthode permet d'améliorer le PSNR de l'image compressée, même à bas bitrates.
La deuxième application proposée repose sur la transmission progressive des fonctions monodimensionnelles. Ces fonctions permettent de reconstruire progressivement l'image en augmentant sa résolution. Cette approche permet une reconstruction avec un nombre quelconque d'étapes intermédiaires dans le cas d'une transmission progressive ; ou l'accès à l'image à plusieurs niveaux de résolutions simultanément dans le cas d'un serveur avec des clients ayant des résolutions d'affichages hétérogènes. De plus, dans le cas de la transmission de ces informations sur un canal avec des pertes de paquets, nous avons montré qu'il était possible de restaurer l'aspect global de l'image.

Ces travaux ont fait l'objet de cinq publications dans des conférences internationales et dans une revue internationale. Par ailleurs, un article dans une deuxième revue internationale est en cours de soumission. Le détail de ces publications se trouve en annexe A.

L'originalité de notre travail repose sur le fait que les traitements (simplification, transmission) sont effectués dans un espace $1D$, qui est complètement différent des approches existantes reposant sur un voisinage des pixels dans l'image ou sur une décomposition sur des bases de fonctions. En effet, les relations de voisinage sont remplacées par un ordre aléatoire de lecture des données multidimensionnelles. Comme illustré précédemment, le balayage de l'image défini par les fonctions internes est différent d'une lecture de l'image ligne par ligne, ce qui fait à la fois la force et la faiblesse de notre approche. En effet, l'image est convertie en fonctions monodimensionnelles, ce qui ouvre des perspectives nouvelles au niveau des traitements de signaux multidimensionnels complexes, mais en contrepartie, la relation de voisinage est perdue. Cette opposition se retrouve dans les deux principaux axes de recherche qui se dégagent : on peut imaginer de nouvelles applications qui profiteraient d'un balayage adapté aux signaux multidimensionnels, ou de nouveaux développements qui exploitent justement le caractère aléatoire du balayage de l'algorithme actuel.

Perspectives

L'aspect compression que nous avons présenté pourrait être développé et amélioré selon plusieurs axes. Tout d'abord, les simplifications des fonctions monodimensionnelles restent simples, et des simplifications plus efficaces pourraient être proposées. Par ailleurs, l'intégration dans JPEG 2000 reste non optimale, puisque les données obtenues après nos simplifications sont différentes d'images d'ondelettes classiques. Les simplifications effectuées dans l'espace des fonctions monodimensionnelles se traduisent par des coefficients retirés des images de détails, et pour que ces images puissent être encodées par JPEG 2000, il est nécessaire de ré-organiser les coefficients. Cette approche profiterait d'une réorganisation des coefficients cherchant à former des code-blocks de coefficients plus homogènes afin d'optimiser l'encodage par JPEG 2000.

Une autre perspective de recherche consiste à développer une méthode de codage adaptée aux fonctions externes. Ce codage pourrait par exemple prendre en compte la redondance des données entre fonctions externes. En effet, il est préférable d'utiliser un pavage dense

pour décomposer les images, quitte à simplifier les fonctions monodimensionnelles par la suite, plutôt que d'utiliser de plus gros pavés. Autrement dit, chaque couche contient une grande partie des données de l'image, données qui sont donc redondantes entre les couches. Il serait également intéressant de générer un balayage dépendant de l'image afin de pouvoir répartir les informations entre les fonctions internes et externes. Avec le schéma de décomposition actuelle, les informations sont totalement contenues dans les fonctions externes, puisque le balayage dépend des fonctions internes. En construisant un balayage dépendant de l'image, il serait alors possible de construire des fonctions externes d'aspect moins bruité, par exemple en parcourant localement les zones homogènes de l'image. Dans un premier temps, pour concevoir un tel balayage, il pourrait être nécessaire d'abandonner certaines contraintes du théorème, telles que la continuité des fonctions internes. Cette perspective en rejoint ainsi une autre, concernant l'adaptation des algorithmes présentés à la décomposition de données discrètes. En tirant parti de la nature discrète du signal multidimensionnel, il pourrait être possible d'optimiser et/ou de simplifier les algorithmes de décomposition existants.

L'application à la transmission progressive, au contraire, profite du caractère aléatoire du balayage, qui permet de restaurer l'aspect global de l'image même lorsque les fonctions externes sont endommagées pendant la transmission. De plus, dans le schéma original, les fonctions internes sont nécessaires à la réorganisation des données contenues dans les fonctions externes. Ainsi, on peut imaginer un algorithme de cryptage dont la clé serait constituée des paliers des fonctions internes. Les données envoyées seraient les valeurs des fonctions externes, impossibles à replacer sans les fonctions internes. Ce cryptage peut de plus être combiné à l'approche de transmission progressive proposée. Toujours dans le domaine de la sécurité, il serait possible d'adapter notre schéma de transmission des fonctions externes pour du Tatouage (Watermarking). En décomposant un tatouage comme une image classique, les fonctions externes obtenues pourraient être mélangées aux fonctions externes issues de la décomposition de l'image tatouée. Toutefois, cette approche suppose de transmettre les fonctions externes, ce qui ramènent à une des perspectives évoquée précédemment, à propos du développement d'un codage adapté aux fonctions externes.

Enfin, nous avons appliqué le théorème de superposition à la décomposition d'images, *i.e.* des fonctions bivariées, mais on peut imaginer étendre cette décomposition et ses propriétés pour représenter des signaux en 3 dimensions ou plus (par exemple de la vidéo ou un maillage $3D$ texturé), ce qui ouvre de nouvelles perspectives de transmission et de traitement. Par exemple, une vidéo pourrait être stockée sous sa décomposition monodimensionnelle, et ensuite reconstruite, selon les besoins, à différentes résolutions, avec un nombre variable d'images par seconde, le tout sans calcul supplémentaire de ré-échantillonage. De plus, cette approche resterait compatible avec le cryptage par les fonctions internes évoqué précédemment, ou du watermarking.

Bibliographie

[Adams, 2001] Adams, M. D. (2001). The jpeg-2000 still image compression standard. Technical report, University of British Columbia.

[Barnsley et al., 1993] Barnsley, M. F., Hurd, L., and Anson, L. (1993). *Fractal image compression*. AK peters Massachusetts.

[Beghdadi and Pesquet-Popescu, 2003] Beghdadi, A. and Pesquet-Popescu, B. (2003). A new image distortion measure based on wavelet decomposition. In *Proceedings of International Symposium on Signal Processing and Its Applications*, volume 1, pages 485–488. IEEE.

[Bodyanskiy et al., 2005] Bodyanskiy, Y., Kolodyazhniy, V., and Otto, P. (2005). Neuro-fuzzy kolmogorov's network for time series prediction and pattern classification. *KI 2005 : Advances in Artificial Intelligence*, pages 191–202.

[Brattka, 2004] Brattka, V. (2004). Du 13-ième problème de Hilbert à la théorie des réseaux de neurones : aspects constructifs du théorème de superposition de Kolmogorov. In *L'héritage de Kolmogorov en mathématiques*, pages 241–268. Éditions Belin, Paris.

[Braun and Griebel, 2009] Braun, J. and Griebel, M. (2009). On a constructive proof of kolmogorov's superposition theorem. *Constructive approximation*, 30(3) :653–675.

[Bryant, 2008] Bryant, D. (2008). *Analysis of Kolmogorov's Superposition Theorem and its Implementation in Applications with Low and High Dimensional Data*. PhD thesis, University of Central Florida Orlando, Florida.

[Chande and Farvardin, 2000] Chande, V. and Farvardin, N. (2000). Progressive transmission of images over memoryless noisy channels. *IEEE Journal on Selected Areas in Communications*, 18(6) :850–860.

[Chang et al., 2008] Chang, C.-C., Li, Y.-C., and Lin, C.-H. (2008). A novel method for progressive image transmission using blocked wavelets. *International Journal of Electronics and Communication*, 62(2) :159–162.

[Charfi et al., 2003] Charfi, Y., Hamzaoui, R., and Saupe, D. (2003). Model-based real-time progressive transmission of images over noisy channels. *Proceedings of IEEE WCNC*, 2 :784–789.

[Chee, 1999] Chee, Y.-K. (1999). Survey of progressive image transmission methods. *International Journal of Imaging Systems and Technology*, 10(1) :3–19.

[Chen et al., 2009] Chen, H., Sun, M., and Steinbach, E. (2009). Compression of bayer-pattern video sequences using adjusted chroma subsampling. *IEEE Trans. Cir. and Sys. for Video Technol.*, 19(12) :1891–1896.

[Chen and Lin, 2005] Chen, S. and Lin, J. (2005). Fault-tolerant and progressive transmission of images. *Pattern recognition*, 38(12) :2466–2471.

[Cheng and Li, 1996] Cheng, H. and Li, X. (1996). On the application of image decomposition to image compression and encryption. In *Communications and Multimedia Security II, IFIP TC6/TC11 Second Joint Working Conference on Communications and Multimedia Security, CMS*, volume 96. Citeseer.

[Coppejans, 2004] Coppejans, M. (2004). On kolmogorov's representation of functions of several variables by functions of one variable. *Journal of Econometrics*, 123(1) :1–31.

[Cosman et al., 2000] Cosman, P., Rogers, J., Sherwood, P., and Zeger, K. (2000). Combined forward error control and packetized zerotree wavelet encoding for transmission of images over varying channels. *IEEE Transactions on Image Processing*, 9(6) :982–993.

[Distasi et al., 2005] Distasi, R., Nappi, M., and Riccio, D. (2005). A range/domain approximation error-based approach for fractal image compression. *IEEE Transactions on Image Processing*, 15(1) :89–97.

[Furht et al., 2005] Furht, B., Muharemagic, E., and Socek, D. (2005). *Multimedia Encryption and Watermarking*. Springer.

[Garcia et al., 2005] Garcia, J. A., Rodriguez-Sánchez, R., and Fdez-Valdivia, J. (2005). Emergence of a region-based approach to image transmission. *Optical Engineering*, 44 :067004.

[Girosi and Poggio, 1989] Girosi, F. and Poggio, T. (1989). Representation properties of networks : Kolmogorov's theorem is irrelevant. *Neural Computation*, 1(4) :465–469.

[Grangetto et al., 2004] Grangetto, M., Magli, E., Olmo, G., and di Elettronica, D. (2004). Ensuring quality of service for image transmission : hybrid loss protection. *IEEE Transactions on Image Processing*, 13(6) :751–757.

[Grattan-Guinness, 2000] Grattan-Guinness, I. (2000). A sideways look at Hilbert's twenty-three problems of 1900. *Notices-American Mathematical Society*, 47(7) :752–757.

[Hecht-Nielsen, 1987] Hecht-Nielsen, R. (1987). Kolmogorov's mapping neural network existence theorem. *Proceedings of the IEEE International Conference on Neural Networks III, New York*, pages 11–13.

[Hilbert, 1900] Hilbert, D. (1900). Mathematische probleme. vortrag, gehalten auf dem internationalen mathematiker-kongreß zu paris 1900. *Nachrichten von der K "oniglichen Gesellschaft der Wissenschaften zu G "ottingen*, pages 253–297.

[Hwang et al., 1999] Hwang, W.-J., Hwang, W.-L., and Lu, Y.-C. (1999). Layered image transmission based on embedded zero-tree wavelet coding. *Optical Engineering*, 38(8) :1326–1334.

[Igelnik, 2009] Igelnik, B. (2009). Kolmogorov's spline complex network and adaptive dynamic modeling of data. *Complex-Valued Neural Networks : Utilizing High-Dimensional Parameters*, pages 56–78.

[Igelnik et al., 1999] Igelnik, B., Pao, Y.-H., LeClair, S. R., and Shen, C. Y. (1999). The ensemble approach to neural-network learning and generalization. *IEEE Transactions on Neural Networks*, 10 :19–30.

[Igelnik and Parikh, 2003] Igelnik, B. and Parikh, N. (2003). Kolmogorov's spline network. *IEEE transactions on neural networks*, 14(4) :725–733.

[Igelnik et al., 2001] Igelnik, B., Tabib-Azar, M., and LeClair, S. R. (2001). A net with complex weights. *IEEE Transactions on Neural Networks*, 12(2) :236–249.

[Ismailov, 2008] Ismailov, V. (2008). On the representation by linear superpositions. *Journal of Approximation Theory*, 151(2) :113–125.

[Kim et al., 2003] Kim, J., Mersereau, R. M., and Altunbasak, Y. (2003). Error-resilient image and video transmission over the internet using unequal error protection. *IEEE Transactions on Image Processing*, 12(2) :121–131.

[Kolmogorov, 1957] Kolmogorov, A. N. (1957). On the representation of continous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademiia Nauk SSSR*, 14(5) :953–956.

[Kolmogorov, 1963] Kolmogorov, A. N. (1963). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *American Mathematical Society Translation*, 28 :55–59.

[Kolodyazhniy and Bodyanskiy, 2004] Kolodyazhniy, V. and Bodyanskiy, Y. (2004). Fuzzy kolmogorov's network. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 764–771. Springer.

[Köppen, 2002] Köppen, M. (2002). On the training of a Kolmogorov Network. *Lecture Notes in Computer Science, Springer Berlin*, 2415 :140–145.

[Köppen and Yoshida, 2005] Köppen, M. and Yoshida, K. (2005). Universal representation of image functions by the sprecher construction. *Soft Computing as Transdisciplinary Science and Technology, Springer Berlin*, 29 :202–210.

[Kurková, 1991] Kurková, V. (1991). Kolmogorov's theorem is relevant. *Neural Computation*, 3(4) :617–622.

[Kurková, 1992] Kurková, V. (1992). Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5(3) :501–506.

[Lagunas et al., 1993] Lagunas, M. A., Pérez-Neira, A., Nájar, M., and Pagés, A. (1993). The Kolmogorov Signal Processor. *Lecture Notes in Computer Science*, 686 :494–512.

[Lamarque and Robert, 1996] Lamarque, C. and Robert, F. (1996). Image analysis using space-filling curves and 1d wavelet bases. *Pattern Recognition*, 29(8) :1309–1322.

[Le Pennec and Mallat, 2005] Le Pennec, E. and Mallat, S. (2005). Sparse geometric image representations with bandelets. *IEEE Transactions on Image Processing*, 14(4) :423–438.

[Leni et al., 2008a] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2008a). Kolmogorov superposition theorem and its application to multivariate function decompositions and image representation. In *proceedings of IEEE conference on Signal-Image Technology & Internet-Based System*, pages 344–351. IEEE Computer Society Washington, DC, USA.

[Leni et al., 2008b] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2008b). Théorème de superposition de kolmogorov et son application à la décomposition de fonctions multivariées. In *proceedings of MajecSTIC*.

[Leni et al., 2009a] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2009a). Kolmogorov superposition theorem and its application to wavelet image decompositions. *Electronic Imaging - Wavelet Applications in Industrial Processing VI, Proceedings of the SPIE*, 7248 :724804–724804–12.

[Leni et al., 2009b] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2009b). Kolmogorov superposition theorem and wavelet decomposition for image compression. *Lecture notes on Computer Science*, 5807 :43–53.

[Leni et al., 2009c] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2009c). A novel approach for image sweeping functions using approximating scheme. In *proceedings of Quality Control by Artificial Vision*.

[Leni et al., 2010a] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2010a). Kolmogorov superposition theorem and wavelets for image compression. *Wavelet Applications in Industrial Processing VII, Proceedings of the SPIE*, 7535(1) :753502–753510.

[Leni et al., 2010b] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2010b). New adaptive and progressive image transmission approach using function superpositions. *Optical Engineering*, 49(9) :097001–097011.

[Leni et al., 2011] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2011). The kolmogorov spline network for image processing. In Igelnik, B., editor, *Computational Modeling and Simulation of Intellect : Current State and Future Perspectives*. IGI Global.

[Li and Cai, 2007] Li, X. and Cai, J. (2007). Robust transmission of jpeg2000 encoded images over packet loss channels. *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 947–950.

[Liu et al., 2004] Liu, Y., Wang, Y., Zhang, B., and Wu, G. (2004). Ensemble algorithm of neural networks and its application. In *Proceedings of International Conference on Machine Learning and Cybernetics*, volume 6, pages 3464–3467.

[Liu and Zalik, 2005] Liu, Y. K. and Zalik, B. (2005). An efficient chain code with huffman coding. *Pattern Recognition*, 38(4) :553–557.

[Lorentz, 1962] Lorentz, G. (1962). Metric entropy, widths, and superpositions of functions. *American Mathematical Monthly*, 69(6) :469–485.

[Lorentz, 1976] Lorentz, G. G. (1976). The 13-th problem of hilbert. In Browder, F., editor, *Mathematical developments arising from Hilbert problems*, pages 419–430. American Mathematical Society.

[Lu et al., 2002] Lu, H., Wang, J., Kot, A., and Shi, Y. (2002). An objective distortion measure for binary document images based on human visual perception. In *Proceedings of International conference on Pattern Recognition*, volume 16, pages 239–242.

[Mallat, 1998] Mallat, S. (1998). *A wavelet tour of signal processing*. Academic Press, San Diego.

[Moon, 2001] Moon, B. (2001). An explicit solution for the cubic spline interpolation for functions of a single variable. *Applied Mathematics and Computation*, 117 :251–255.

[Nakamura et al., 1993] Nakamura, M., Mines, R., and Kreinovich, V. (1993). Guaranteed intervals for kolmogorov's theorem (and their possible relation to neural networks). *Interval Computations*, 3 :183–199.

[Nees, 1994] Nees, M. (1994). Approximative versions of Kolmogorov's superposition theorem, proved constructively. *Journal of Computational and Applied Mathematics*, 54(2) :239–250.

[Ozturk and Sogukpinar, 2004] Ozturk, I. and Sogukpinar, I. (2004). Analysis and comparison of image encryption algorithms. In *ICIT 2004 : International Conference on Information Technology, Istanbul*, pages 38–42. World Scientific and Technological Research Society, Turkey,.

[Papamarkos et al., 2002] Papamarkos, N., Atsalakis, A. E., and Strouthopoulos, C. P. (2002). Adaptive color reduction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 32(1) :44–56.

[Pednault, 2006] Pednault, E. (2006). Transform regression and the kolmogorov superposition theorem. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, pages 35–46. Society for Industrial Mathematics.

[Perez-Freire et al., 2006] Perez-Freire, L., Comesana, P., Troncoso-Pastoriza, J., and Perez-Gonzalez, F. (2006). Watermarking security : a survey. *Transactions on Data Hiding and Multimedia Security I*, pages 41–72.

[Said and Pearlman, 1996] Said, A. and Pearlman, W. (1996). A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE transactions on Circuits and Systems for Video Technology*, 6(3) :243–251.

[Salomon, 2007] Salomon, D. (2007). *Data compression : the complete reference*. Springer-Verlag New York Inc.

[Skodras et al., 2001] Skodras, A., Christopoulos, C., and Ebrahimi, T. (2001). The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18 :36–58.

[Sprecher, 1965] Sprecher, D. (1965). On the structure of continuous functions of several variables. *Transactions of the American Mathematical Society*, 115(3) :340–355.

[Sprecher, 1972] Sprecher, D. (1972). An improvement in the superposition theorem of Kolmogorov. *Journal of Mathematical Analysis and Applications*, 38(1) :208–213.

[Sprecher, 1996] Sprecher, D. A. (1996). A numerical implementation of Kolmogorov's superpositions. *Neural Networks*, 9(5) :765–772.

[Sprecher, 1997] Sprecher, D. A. (1997). A numerical implementation of Kolmogorov's superpositions II. *Neural Networks*, 10(3) :447–457.

[Sprecher and Draghici, 2002] Sprecher, D. A. and Draghici, S. (2002). Space-filling curves and Kolmogorov superposition-based neural networks. *Neural Networks*, 15(1) :57–67.

[Stankovic et al., 2003] Stankovic, V., Hamzaoui, R., Charfi, Y., and Xiong, Z. (2003). Real-time unequal error protection algorithms for progressive image transmission. *IEEE Journal on Selected Areas in Communications*, 21(10) :1526–1535.

[Stockhammer et al., 2002] Stockhammer, T., Jenkac, H., and Weiß, C. (2002). Feedback and error protection strategies for wireless progressive video transmission. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6) :465–482.

[Taubman and Marcellin, 2001] Taubman, D. and Marcellin, M. (2001). *JPEG2000, Image Compression Fundamentals, standards and practice*. Kluwer Academic Publishers.

[Vecci et al., 1998] Vecci, L., Piazza, F., and Uncini, A. (1998). Learning and approximation capabilities of adaptive spline activation function neural networks. *Neural Networks*, 11(2) :259–270.

[Vitushkin, 2004] Vitushkin, A. G. (2004). On Hilbert's thirteenth problem and related questions. *Russian Mathematical Surveys*, 59(1) :11–27.

Table des figures

Liste des tableaux

Annexe A

Liste des publications

Articles revues internationales

Soumis

– "Kolmogorov Superposition Theorem and JPEG 2000 for image compression", Pierre-Emmanuel LENI, Yohan FOUGEROLLE, Frédéric TRUCHETET

Paru

– [Leni et al., 2010b] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2010b). New adaptive and progressive image transmission approach using function superpositions. *Optical Engineering, 49(9) :097001-097011.*

Chapitre de livre

– [Leni et al., 2011] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2011). The Kolmogorov spline network for image processing. In *Igelnik, B., editor, Computational Modeling and Simulation of Intellect : Current State and Future Perspectives. IGI Global.*

Conférences internationales

– [Leni et al., 2008a] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2008a). Kolmogorov superposition theorem and its application to multivariate function decompositions and image representation. In *proceedings of IEEE conference on Signal-Image Technology & Internet-Based System, pages 344-351. IEEE Computer Society Washington, DC, USA.*

– [Leni et al., 2009a] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2009a). Kolmogorov superposition theorem and its application to wavelet image decompositions. *Electronic Imaging - Wavelet Applications in Industrial Processing VI, Proceedings of the SPIE, 7248 :724804-724804-12.*

– [Leni et al., 2009b] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2009b). Kolmogorov superposition theorem and wavelet decomposition for image compression. *Lecture notes on Computer Science, 5807 :43-53.*
– [Leni et al., 2009c] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2009c). A novel approach for image sweeping functions using approximating scheme. In *proceedings of Quality Control by Artificial Vision.*
– [Leni et al., 2010a] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2010a). Kolmogorov superposition theorem and wavelets for image compression. *Wavelet Applications in Industrial Processing VII, Proceedings of the SPIE, 7535(1) :753502-753510.*

Conférence nationale

– [Leni et al., 2008b] Leni, P.-E., Fougerolle, Y. D., and Truchetet, F. (2008b). Théorème de superposition de kolmogorov et son application à la décomposition de fonctions multivariées. In *proceedings of MajecSTIC.*