Université Paris Diderot (Paris 7)

Ecole doctorale
Sciences Mathématiques de Paris-Centre

Thèse présentée pour le titre de
Docteur en Sciences
Spécialité Traitement d'Images

Parcimonie et ses applications à des données de flot optique

Arnaud Woiselle

Directeur de thèse **Jean-Luc Starck**
Co-directeur **Jalal Fadili**

Soutenue le 23 Novembre 2010

Jury

Michael Elad	Technion, Israel Institute of Technology	Rapporteur
Fionn Murtagh	University of London	Rapporteur
Jean-Luc Starck	CEA/IRFU/SEDI-Service d'Astrophysique	Jury
Jalal Fadili	GREYC ENSICAEN	Jury
Rachid Deriche	INRIA/ATHENA, Sophia Antipolis	Jury
Gabriel Peyre	CEREMADE, Univ. Paris Dauphine	Jury
Marc Bousquet	SAFRAN/Sagem	Jury

Remerciements

Je remercie sincèrement Fionn Murtagh et Michael Elad, qui m'ont fait l'honneur d'être rapporteurs de cette thèse, pour la relecture de ce manuscrit et pour leurs remarques et critiques.

Rachid Deriche et Gabriel Peyre reçoivent aussi toute ma gratitude pour avoir accepté de participer à ma soutenance en tant que membres du jury.

Je tiens à exprimer ma profonde gratitude envers mon directeur de thèse Jean-Luc Starck, pour m'avoir accueilli dans son équipe, avoir été présent pour répondre à mes nombreuses questions, avoir partagé ses idées et son expertise, ainsi que pour la liberté qu'il m'a laissée au long de cette thèse.

Je remercie également mon co-directeur Jalal Fadili, qui a apporté sa rigueur et ses connaissances théoriques, et m'a permis de présenter mes travaux dans son laboratoire.

Je remercie tous les gens du Service d'Astrophysique et du SEDI du CEA Saclay, avec lesquels j'ai passé plus de deux ans de ma thèse, et plus particulièrement les autres thésards et post-docs pour l'ambiance du labo et les entraides mutuelles.

Je souhaite également remercier les personnes de Sagem avec lesquelles j'ai pu travailler pendant cette thèse, dont mon encadrant Joël Budin et mes responsables Jean-Marc Champeyroux, François Coursaget, ainsi que Marc Bousquet, Michel Broekaert et Daniel Duclos pour les discussions constructives, et car ils m'ont soutenu et ont cru en ces travaux en décalage par rapport au quotidien d'une entreprise, ainsi que les membres de l'équipe traitement d'images études amonts du CE CTP.

Merci à Luis Garrido et Romain Teyssier pour nous avoir fourni respectivement les séquences de désentrelacement de la section 4.5 (voir [4]) et les simulations λCDM [159] des sections 3.5 et 3.4.

Enfin je remercie ma famille et mes amis qui m'ont soutenu au long de ces trois ans.

Résumé

Cette thèse adresse la résolution de problèmes inverses par l'utilisation de transformées parcimonieuses tridimensionnelles. Nous présentons tout d'abord les systèmes d'imagerie infrarouge et les problèmes liés aux bruits et aux données manquantes, puis nous faisons un court état de l'art des techniques industrielles de traitement d'images. Nous introduisons ensuite le concept de parcimonie et présentons les algorithmes et représentations classiques de ce domaine. Le coeur de ce travail de recherche est le développement de nouvelles représentations 3D, qui sont bien adaptées à la représentation des surfaces et des filaments dans un volume 3D. Deux nouvelles transformées sont proposées, ainsi que la modification d'une troisième, les curvelets rapides, en vue de réduire sa redondance. Ces outils sont alors appliqués à la résolution de divers problèmes inverses comme le débruitage, l'inpainting ou le désentrelacement. Des expériences numériques montrent la qualité des résultats. Finalement, nous proposons quelques applications concrètes utilisant des transformées parcimonieuses au traitement de vidéos infrarouges, telles que la détection de petites cibles, ou la déconvolution de flou de bougé.

Abstract

This manuscript addresses inverse problems using 3D sparsifying transforms in the context of infrared video. We first survey the infrared imaging system and the problems associated such as noise and missing data, and make a short state of the art on industrial imaging processing technics. Then we introduce the concept of sparsity and present common algorithms and representations used in this domain. The core of this research is the development of new 3D representations, which are well adapted to representing surfaces or filaments in a 3D volume. Two innovative transforms are proposed, and an existing one – the fast curvelet transform – is modified to better suit the applications by reducing its redundancy. These tools are used in many inverse problems such as denoising, inpainting and video deinterlacing, and prove to be very useful and competitive. Finally, we propose a few concrete applications of sparse transforms to infrared video processing, such as small targets detection or motion blur deconvolution.

Table des matières

Introduction

La sécurité de la France est aujourd'hui en partie due à la dissuasion nucléaire, mais aussi à sa capacité à détecter les menaces qui pourraient se présenter, et cela à l'aide de dispositifs optroniques les plus performants possibles. Par ailleurs, lorsque l'envoi de soldats est nécessaire, et que leur vie est en jeu, il convient de les doter d'outils à la pointe de la technologie, dont des caméras observant dans le domaine visible, mais aussi dans l'infrarouge pour augmenter leur conscience de l'environnement. Les systèmes optiques et les capteurs sont maintenant très perfectionnés, et le traitement de l'image peut permettre d'augmenter significativement les performances de ces systèmes. En effet, il permet d'une part de restituer au mieux le signal enregistré, mais il donne aussi une assistance à l'analyse de la scène et peut servir d'outil d'aide à la décision, ou signaler des points d'intérêt dans la scène.

Le traitement d'images est une discipline extrêmement large, en relation étroite avec les mathématiques, et de nombreuses approches sont possibles dépendant des contraintes de performance ou de puissance que l'on pose au système, aussi bien que des a priori sur ce que l'on veut détecter ou étudier. Un domaine récent du traitement d'images est la parcimonie, selon laquelle certains signaux ont une représentation simple (creuse, parcimonieuse) dans un espace donné. Ce concept a donné naissance à de nombreux algorithmes permettant de résoudre toutes sortes de problèmes en traitement d'images, mais dont les performances restent assujetties à l'utilisation d'un domaine dans lequel les données – les images dans notre cas – sont effectivement parcimonieuses.

Les flux vidéos, qui sont au coeur de cette thèse, sont des données tridimensionnelles, et il faut par conséquent trouver des espaces de représentation dans lesquels de telles données sont parcimonieuses. Une transformation classique est la transformée en ondelettes[107], qui fournit des fonctions multi-échelles isotropes et localisées, mais ne s'adapte pas bien aux singularités anisotropes des images (e.g. les contours). De nombreuses autres représentations ont été développées pour représenter les signaux 2-D [90, 17, 25, 18, 43, 48], s'adaptant mieux à ces structures anisotropes, ainsi que des méthodes d'apprentissage [124, 1]. Cette thèse s'intéresse à la construction de transformations tridimensionnelles [98, 30] adaptées aux singularités qui apparaissent lorsque l'on considère une vidéo comme un volume 3-D, comme les lignes ou les surfaces, et à leur application à la résolution de divers problèmes inverses en traitement d'image, comme le débruitage, l'inpainting ou la déconvolution.

Structure du manuscrit

Le chapitre premier présente la problématique liée à l'acquisition et au traitement de données vidéo infrarouge, et décrit les divers traitements d'image utilisés au sein de l'équipe dans laquelle j'étais intégré au long de cette thèse, au sein de la société Sagem du groupe Safran. En effet, il faut modéliser la chaîne de formation d'image et les traitements qui y sont présents au niveau du capteur (correction de gain et d'offset), pour pouvoir par la suite traiter des problèmes plus complexes de restauration, et comprendre leur influence sur les traitements de plus haut niveau qui peuvent suivre comme la détection d'objets mobiles.

Le second chapitre introduit et illustre le concept de parcimonie, puis présente divers algorithmes de décomposition dans les dictionnaires redondants [53, 32, 110, 122], et finalement décrit les transformations classiques qui rendent certains types de données parcimonieuses, qu'elles soient adaptatives [93, 125, 108] ou non [25, 18, 43].

Le troisième chapitre introduit la première contribution majeure de cette thèse, à savoir les transformées en curvelets tridimensionnelles première génération [167], qui sont des représentations nouvelles permettant de représenter efficacement les signaux contenant des structures surfaciques ainsi que linéiques dans un volume tri-dimensionnel. Cette dernière structure de dimension 1 ayant un intérêt particulier étant donné qu'aucune autre représentation actuelle ne permet de représenter efficacement de telles structures à toutes les échelles.

Le chapitre 4 présente une nouvelle implémentation [168] des curvelets seconde génération ayant une redondance réduite d'un facteur supérieur à deux par rapport aux curvelets rapides de Ying, Demanet et Candès [170]. Ces deux chapitres introduisent des nouvelles représentations, et sont accompagnés d'expériences montrant leur applicabilité et utilité pour résoudre divers problèmes inverses comme le débruitage ou l'inpainting.

Le dernier chapitre se focalise sur les applications à la vidéo, et plus particulièrement la vidéo infrarouge à l'aide de curvelets bi-dimensionnelles et tridimensionnelles, avec des résultats marquants en débruitage, déconvolution, inpainting ou la séparation de composantes.

Une conclusion générale sur les travaux réalisés lors de cette thèse ainsi que les perspectives terminent le corps de ce mémoire.

L'annexe A contient un rapport technique concernant une comparaison de quelques algorithmes de minimisation pour une application à de l'échantillonnage compressif en Fourier, et les codes développés dans ce contexte ont été utilisés par une équipe d'astronomes travaillant sur le radio-télescope SKA, menant à la publication [96]. L'annexe B présente le logiciel développé pour Sagem qui contient la plupart des outils développés pendant cette thèse, et dont une certaines fonctions sont distribuées en ligne sur `http ://www.morphologicaldiversity.org`.

Liste des tableaux

Table des figures

Notations

Ensembles
- $\mathbb{N}$ l'ensemble des entiers naturels,
- $\mathbb{Z}$ l'ensemble des entiers relatifs,
- $\mathbb{R}$ l'ensemble des réels,
- $\mathbb{C}$ l'ensemble des complexes,
- $|\mathbb{E}|$ le cardinal de l'ensemble $\mathbb{E}$
- $\mathbb{E}^* = \mathbb{E} \setminus 0_E$,
- $\mathbb{E}_+ = \{x \in \mathbb{E} : x \geq 0_E\}$.

Fonctions
- $f(t)$ une fonction à variable continue de $t \in \mathbb{R}$ dans $\mathbb{R}$,
- $f[t]$ une fonction à variable discrète de $t \in \mathbb{Z}$ dans $\mathbb{R}$,
- I une fonction de $\mathbb{A} \subset \mathbb{R}^2$ dans $\mathbb{C}$, dite *image*,
- $\hat{f}, \hat{I}$ les transformées de Fourier de f et I,
- $\mathcal{TF}^{1d} I$ la transformée de Fourier 1-D sur chaque colonne de I,
- $\mathbf{Rad} I$ la Transformée de Radon de I,
- $\mathcal{W}I$, $\mathcal{R}I$, $\mathcal{C}I$ respectivement les Transformées en Ondelettes, Ridgelets, et Curvelets de I,
- $\check{f}$ la décimation de f, soit $\check{f}[n] = \begin{cases} f[p] \text{ pour } n = 2p \\ 0 \text{ pour } n = 2p+1 \end{cases}$,
- $L_2(\Omega)$ l'espace des fonctions continues de carré intégrable sur un domaine continu Ω,
- $\ell_2(\Omega)$ l'espace des fonctions discrètes de carré intégrable sur un domaine continu Ω,
- f^* le conjugué de f,
- $\overline{f}(t) = f^*(-t)$,
- $\tilde{f}$ une estimée ou transformée de f, définie lors de l'emploi de la notation.

Matrices et normes
- $\mathbf{M}^T$ la matrice transposée de $\mathbf{M}$,
- $\mathbf{M}^*$ l'adjoint de $\mathbf{M}$,
- $\mathbf{M}^+$ le pseudo-inverse de $\mathbf{M}$,
- $|x|$ le module de $x \in \mathbb{C}$,
- $\|\cdot\|_0$ la pseudo-norme l^0, le nombre d'éléments non nuls,
- $\|\cdot\|_p$ la norme l^p pour $p > 0$ avec l'extension à $p = \infty$,
- $\|\cdot\|_{w\ell_p}$ la norme l^p-faible,
- $\|\cdot\|_{TV}$ la semi-norme de la variation totale discrète,
- $x_+ = \max(x, 0)$ et $x_- = \min(x, 0)$ les parties positives et négatives de x.

Opérateurs
- $*$ le produit de convolution en continu,

- $\circledast$ le produit de convolution circulaire,
- $\odot$ le produit terme à terme,
- $\otimes$ le produit de Kronecker,
- ∇ gradient discret d'une fonction ou image.

Chapitre 1

Le traitement d'images IR

1.1 Contexte

Cette thèse s'est inscrite dans le cadre d'un contrat CIFRE avec la Sagem – société du groupe SAFRAN, – un des leaders mondiaux en optronique et avionique, et plus précisément dans sa branche Optronique et Défense qui crée des produits de haute technologie pour le civil et le militaire, aussi bien pour le naval que l'aéroporté ou le terrestre pour diverses applications critiques. Sagem conçoit et produit des caméras dans le visible et l'infrarouge, des périscopes et boules gyro-stabilisées, et offre des solutions à la pointe de la technologie.
Les algorithmes et outils acquis ou développés dans le cadre de cette thèse ont pour vocation d'être un jour portés sur les produits Sagem. Si ces produits sont variés, pouvant être embarqués (caméras sur des chars, hélicoptères, drones, etc.) ou portatifs (jumelles, viseurs d'armes, etc.), leurs applications le sont tout autant, allant de la veille maritime et aéroportée au guidage de missiles, en passant par l'optronique terrestre et l'équipement des fantassins.
Nous allons d'abord effectuer un tour d'horizon concernant l'environnement physique dans lequel seront implantés les algorithmes, c'est à dire la chaîne optique et les propriétés de l'infrarouge, puis présenter certains des algorithmes de traitement d'images implémentés ou à l'étude pour les caméras thermiques portables.

1.2 Chaîne image

1.2.1 Environnement

Afin de bien caractériser les performances des produits, il faut connaître l'environnement dans lequel ils opèrent, c'est à dire caractériser les objets et le chemin de l'objet au détecteur. En imagerie un objet est caractérisé par ce qu'on appelle sa ***signature***. En infrarouge, la signature thermique représente la température de l'objet telle qu'elle serait mesurée par un capteur (caméra thermique) donné. Pour pouvoir effectuer des traitements adaptés et pouvoir reconnaître les objets, il faudrait faire étudier les signatures des cibles (êtres humains, véhicules - chars, mines, etc.) et des fonds (ville, campagne, forêt, désert, mer, neige, etc.) en fonction de l'environnement notamment du climat et du temps (humidité, température, etc.). Cela peut être fait par deux méthodes : soit par mesures réelles, soit par simulation. Les mesures réelles de signatures nécessitant énormément de temps et de ressources sont rarement effectuées. Les simulations quant à elles sont faites à l'aide de logiciels générant par exemple l'image d'un objet donné sur un fond donné en spécifiant la

géométrie des objets, les sources de chaleur, les matériaux, et toute autre information sur le milieu. Le logiciel tient alors compte de la thermodynamique et des rayonnements induits pour générer une image. Cependant, le problème majeur est que le client accepte et valide l'outil de simulation, ce qui pousse à rechercher des critères standards pour caractériser les performances d'un produit d'imagerie.
Les produits pouvant être utilisés à des fins très variées, telles la veille, l'observation ou la conduite de tir, et dans tout type d'environnement, et les cibles étant difficiles à caractériser (elles sont en général plus chaudes que le fond mais ce c'est pas toujours le cas), un critère d'efficacité est utilisé. La caractérisation se fait sur des cibles normalisées, ayant une différence de température ΔT donnée avec le fond, ainsi que sur des mires. La distance à laquelle on détecte les mires, ou où l'on peut compter le nombre de barres de celle-ci définit la ***portée***. On détermine ainsi trois portées :

- la détection : on voit quelque chose ;
- la reconnaissance : on sait ce que c'est, un char par exemple ;
- l'identification : on connaît le modèle du char.

Les spécifications permettant de calculer cette portée sont basées sur le critère de Johnson, une série de tests psychovisuels effectués il y a une cinquantaine d'années, et constituent aujourd'hui le standard d'évaluation. Cependant, ils ont été effectués en noir et blanc et leur équivalent couleur n'existe pas, posant un problème de plus pour les nouveaux équipements.

1.2.2 Optique

Les optiques sont généralement différentes pour chaque utilisation et bande spectrale (IR, visible, télémétrie). Même si des optiques multi-bandes ont déjà été testées sur des produits, elles se sont révélées trop onéreuses du fait du cahier des charges trop strict une fois les besoins des différentes applications sommés. Aujourd'hui, seul un produit Sagem possède une optique commune pour la voie jour qui sert aussi de réception à la télémétrie.
Les diverses aberrations optiques telles les aberrations sphériques, chromatiques, ou la coma sont en grande partie corrigées par l'utilisation de matériaux à forte constringence (i.e. des matériaux dont la variation d'indice de réfraction avec la longueur d'onde est faible) et par l'utilisation de plusieurs lentilles accolées dont les défauts se compensent. Dans certains cas, on utilise des dispositifs diffractants pour corriger les forts défauts de chromatisme.
La diversité des matériaux permet aujourd'hui l'utilisation d'objectifs athermalisés dans le visible comme en infrarouge, c'est à dire une combinaison de matériaux optiques dont les propriétés se compensent lorsque la température varie. Cela permet de garder le plan image sur le capteur indépendamment de la température. Cette technologie donne une grande stabilité d'axe, une plus longue durée de vie, et grâce à l'absence de parties mobiles et d'électronique pour effectuer la mise au point, un gain important en volume et en masse. Lorsque les contraintes de masse sont importantes, on peut aussi réduire le poids de la coque d'environ 30% en utilisant du magnésium au lieu de l'aluminium.
Les matériaux optiques utilisés dans le visible sont très variés, avec une importance pour les difluoromethylenes CF_2, utilisés aussi en IR. Cependant, pour l'IR, le germanium et le silicium sont les plus utilisés, avec le calcium fluoride C_aF_2. Un autre matériau intéressant est le gazir, qui est indispensable pour les objectifs athermiques, mais est très fragile et difficile à usiner. Les parties très exposées, notamment les dômes extérieurs sur les caméras

embarquées, sont quant à elles faites en saphir.

1.2.3 Capteurs

Technologie

Les capteurs utilisés ont des résolutions allant de 384×288 (voir moins en IR) à 1280×1024 (voir même de la HD) et utilisent la technologie CMOS. Les bandes observées vont du visible à l'IR thermique (longueur d'onde de 3 à $15\mu m$), et certaines caméras fonctionnent dans le visible en intensification de lumière. Si la plupart des capteurs sont aujourd'hui sous forme de matrices, en infrarouge refroidi dans la bande spectrale $3 - 5\mu m$, et en refroidi comme en non refroidi dans la bande $8 - 12\mu m$, d'anciens produits toujours en vente comme les caméras IRIS utilisent une barrette de capteurs associée à un miroir de balayage. Selon les produits, on utilise des capteurs couleurs ou en niveaux de gris, sachant que les capteurs couleurs sont généralement des matrices à filtres de Bayer, donc n'échantillonnent pas toutes les couleurs avec la même résolution (deux pixels verts pour un rouge et un bleu en général) et engendrent des résolutions inférieures.
Pour ce qui est de l'IR, la gamme de produits se scinde en deux classes de qualité différentes, selon que le détecteur est refroidi ou non. Dans le cas des détecteurs non refroidis, on utilise des détecteurs thermiques, dits bolométriques, qui voient leur conductivité changer en fonction de la température, donc du flux incident. Leur utilisation augmente l'importance du traitement d'image car la réponse des pixels n'est pas du tout homogène sur cette technologie, et la température ambiante importante crée un flux parasite beaucoup plus important que le flux utile (de l'ordre de quinze fois plus grand). Ces détecteurs sont parfois stabilisés en température par effet Pelletier. En refroidi, les détecteurs sont des semi-conducteurs à base d'indium et d'antimoine (InSb) et se voient placés à l'intérieur de cryostats à une température inférieure à $80K$. Dans ce cas, les détecteurs fonctionnent comme des compteurs de photons et sont donc bien plus précis (avec une très bonne capacité quantique, presque chaque photon incident est transformé en électron). De plus, sur les matrices refroidies, toutes les valeurs des pixels sont lues simultanément (enfin toutes enregistrées dans des capacités puis lues), alors qu'en non refroidi, les lignes sont lues une à une d'où des effets visuels désagréables lors des mouvements du porteur.

1.2.4 Restitution

Les équipements portés par les soldats tels les jumelles doivent avoir une optique de restitution de faibles poids et volume, on utilise donc un écran OLED au plan focal d'un prisme. Pour permettre au plus grand nombre une utilisation optimale malgré des faibles problèmes de vision, l'optique est réglée à -0.6 dioptries ; ainsi, l'œil d'un utilisateur ayant une bonne vision ne fait que peu d'effort pour accommoder, et il reste possible pour quelqu'un ayant une faible myopie d'utiliser cet équipement de tête.
Le bloc de restitution comprend de l'électronique (pour les réglages sur l'affichage), le *μ-display*, l'optique, et la mécanique permettant les réglages lors de l'assemblage. L'écran OLED a une résolution de 800×600 en couleurs, avec des pixels de $15\mu m$ de côté.
Pour le matériel de taille plus conséquente ou aéroporté, l'affichage est déporté vers des moniteurs LCD, en respectant les standards de transmission et codage de l'organisme utilisateur du produit (OTAN par exemple). Dans l'optique des réductions de coût, certains produits commencent à voir apparaître des lentilles en plastique, mais cela reste rare.

1.3 Problèmes liés à l'acquisition d'images

1.3.1 Défauts capteurs

Les matrices de pixels possèdent des défauts dus à la fabrication, et elles sont systématiquement toutes testées dès la sortie d'usine sur un banc de montage provisoire. Chaque capteur est placé devant un corps noir dont on fait une séquence d'images, et on en extrait les caractéristiques de chaque pixel en termes de gain, d'offset et de bruit. On obtient alors une carte caractérisant la matrice, et suivant des critères sur l'ensemble de ces trois composantes, chaque pixel est déclaré *accepté* ou *refusé*. Les figures 1.1, 1.2 et 1.3 représentent la réponse d'un capteur et la carte de pixels classés *refusés* ainsi que le motif du refus, c'est à dire la ou les composantes (gain, offset, bruit) ne respectant pas les critères d'admissibilité. Le capteur de la figure 1.1 est déclaré *accepté*, alors que ceux des figures 1.2 et 1.3 sont refusés respectivement pour clivage et clusters de pixels morts.
La carte des pixels morts, ayant un offset important, trop de bruit, ou toute autre anomalie importante est créée et enregistrée dans l'électronique du capteur, et servira de base aux traitements de restauration : les valeurs de ces pixels ne seront pas prises en compte, mais remplacées avant tout autre traitement. Cette opération de substitution des pixels morts est actuellement effectuée comme décrit dans la section 1.4.2, mais a pour vocation d'être remplacée par les résultats de cette thèse, par inpainting basé sur la parcimonie et la diversité morphologique dans la vidéo.

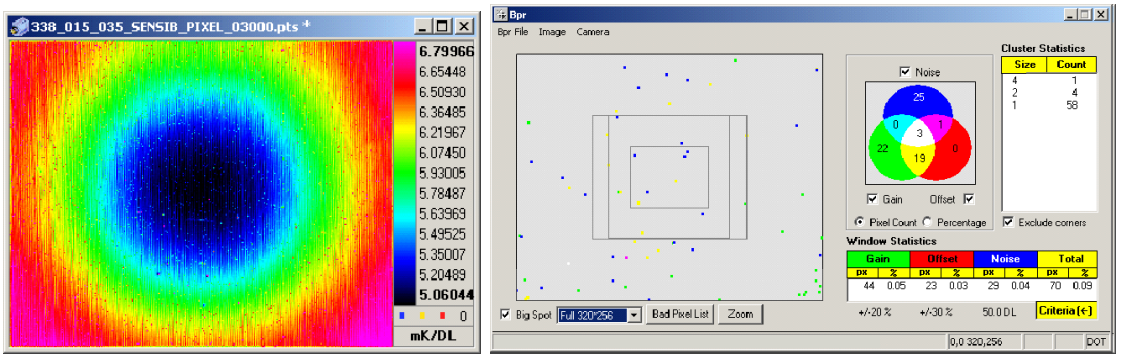


FIG. 1.1 – Gauche : Image de la sensibilité d'un capteur matriciel obtenue en le mettant sous le rayonnement d'un corps noir. Droite : Carte des pixels défectueux d'un capteur, avec classification en couleurs selon le défaut : Gain, Offset, Bruit.

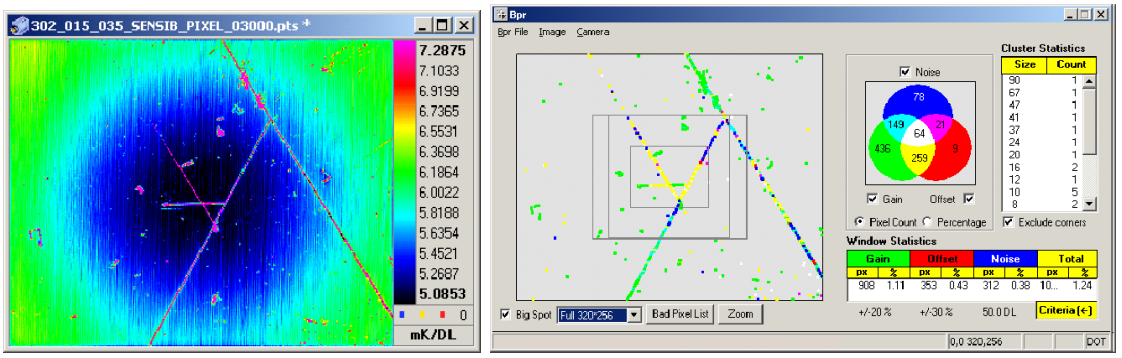


FIG. 1.2 – Cas d'un capteur refusé pour cause de clivage. Gauche : Image de sensibilité. Droite : Carte des pixels défectueux.

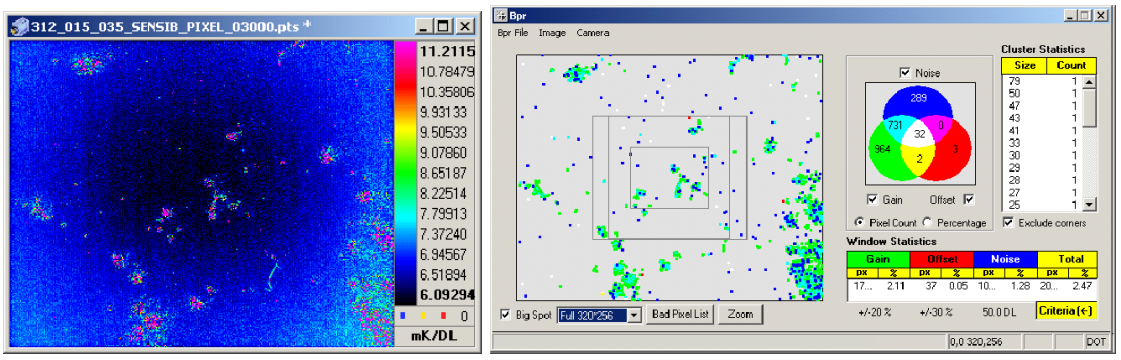


FIG. 1.3 – Cas d'un capteur refusé pour cause de clusters défectueux. Gauche : Image de sensibilité, droite : Carte des pixels défectueux.

1.3.2 Bruits

Le bilan des bruits qui interviennent dans la chaîne d'acquisition est le suivant :

- En faible flux, la nature même de la lumière en tant que photons fait apparaître le bruit de photon [135] qui suit une loi de Poisson de paramètre λ =nombre moyen de photons capturés par seconde. Lorsque le flux est fort, ou le temps d'intégration long, ce bruit se rapproche d'un bruit gaussien de moyenne et variance λ par le théorème central limite.
- La présence de la résistance bolométrique dans le circuit de lecture induit un bruit thermique dit bruit Johnson. La variance sur la tension mesurée est donnée par la formule de Nyquist. Il s'agit d'un bruit blanc gaussien additif.
- Le bruit de scintillation , ou "bruit en $1/f$" proviendrait des variations de l'intensité et du courant dans les circuits, dues aux variations des résistances. Il s'agit d'un bruit temporel, décorrélé spatialement, et ayant une densité spectrale de puissance en $1/f^n$. Ce bruit apparait donc comme un fort bruit spatial évoluant lentement avec le temps.
- Dans le cas de l'IR, le boîtier et l'optique sont en général à la température ambiante donc émettent un rayonnement parasite sur le capteur.
- Le bruit ROIC, *read-out integrated circuit*. Il s'agit d'un bruit de lignage/colonage, blanc ou en $1/f$ [95].

1.3.3 PSF

La réponse impulsionnelle (notée PSF, fonction d'étalement du point) de l'optique est connue et est considérée en deux dimensions, homogène dans le plan perpendiculaire à l'axe optique. Cependant, la tendance actuelle étant à la réduction des coûts de production, il convient d'obtenir les mêmes performances avec de l'optique de basse qualité mais en améliorant les traitements. Par exemple, la prise en compte d'une PSF variant spatialement est envisageable dans l'avenir.

1.3.4 Echantillonnage

D'après le théorème d'échantillonnage de Shannon-Nyquist, pour pouvoir reconstruire un signal 1-D après échantillonnage à une fréquence f_e la partie basses fréquences ($0 \leq f \leq f_e/2$) d'un signal, il faut couper les fréquences supérieures à $f_e/2$ pour prévenir contre le repliement de spectre. Dans le cas d'un système d'acquisition optique (on ne considérera

qu'une dimension, la symétrie de révolution étant supposée), cette coupure peut être faite par l'optique, en ajustant la taille de la tâche de diffraction de manière à ce que sa MTF (Modulation Transfert Function) coupe les fréquences supérieures à $f_e/2$. Cependant, si on coupe effectivement ces fréquences, notre résolution sera limitée par f_e même si on sait faire de l'optique permettant l'accès à des fréquences plus élevées. On peut alors choisir de laisser passer les hautes fréquences (i.e. utiliser une meilleure optique) mais on subit le repliement de spectre. Une méthode pour s'en affranchir est d'effectuer de la super-résolution, et donc d'augmenter virtuellement la fréquence d'échantillonnage, dans l'idéal jusqu'à la fréquence de coupure de l'optique. Un compromis entre la possibilité d'effectuer de la super-résolution et la qualité de l'optique est donc à trouver.
L'utilisation de capteurs couleurs et la difficulté de les fabriquer à la même résolution que les capteurs monochromes ("noir et blanc") sans subir une baisse conséquente du rapport signal sur bruit (SNR ou RSB), pousse à l'usage des matrices de Bayer [5] (les lignes de pixels sont acquises alternativement VBVB... et RVRV...). L'image obtenue est alors interpolée pour reconstruire une image RVB à la résolution du capteur comme si il était trichrome, ce qui peut créer des artéfacts colorés dans les zones à fort gradient. Se référer à [2, 75] pour une revue des algorithmes de démosaïcage.

1.4 Le traitement d'images IR

Les traitements effectués au sein de Sagem ont plusieurs objectifs. Dans un premier temps, il s'agit d'améliorer la qualité de la vidéo pour l'utilisateur, à savoir ajuster la dynamique puis diminuer le bruit, combler les pixels morts, augmenter la résolution, ou stabiliser l'image.
Une seconde phase consiste ensuite à en extraire de l'information automatiquement, comme détecter les objets mobiles.

1.4.1 Méthodes d'adaptation de la dynamique

L'adaptation de la dynamique peut se faire à différents niveaux sur un produit. D'un point de vue analogique avant la numérisation, puis lors des traitements sur l'image digitalisée.

Traitements analogiques

Afin d'avoir le meilleur SNR, le temps d'intégration du capteur est optimisé, c'est à dire qu'on le prend le plus grand possible sans saturer les capteurs. Le courant de sortie varie donc, et il faut appliquer un gain analogique entre le capteur et le CAN (Convertisseur Analogique Numérique) afin d'adapter le signal à la dynamique du CAN. Ce genre de contrôle de gain doit être asservi et le gain dépend des statistiques de l'image (de type min, max, $\max - \min$), qui dépendent eux-mêmes du temps d'intégration. Afin d'éviter les effets de pompage (*flicker*, oscillation rapide de la luminance des images), il convient par exemple de filtrer ces statistiques avant de calculer les gains.

Traitements numériques

D'un point de vue numérique, un ajustement de la dynamique est aussi effectué [111]. Un traitement linéaire est effectué dans la plupart des applications, afin d'ajuster la dynamique de l'image à celle de l'imageur par une correction linéaire. Ainsi, l'image obtenue a

la dynamique maximale à Q niveaux. Cette transformation se fait en associant à chaque pixel i de niveau p_i le niveau

$$q_i = L(p_i) = (Q-1)\frac{p_i - \min_j(p_j)}{\max_j(p_j) - \min_j(p_j)}.$$

Dans la pratique, on ne prend pas les maximum et minimum réels de l'image, mais on utilise les estimateurs robustes associés (e.g. après avoir exclu des statistiques les quelques pixels à valeurs extrêmes).

Egalisation d'histogramme

Un autre traitement classique pour améliorer la dynamique visuelle d'une image est l'égalisation d'histogramme. Elle consiste en une transformation croissante de l'histogramme normalisé $H(p)$ d'une image à N pixels, afin de le rendre le plus plat possible.

$$H(p) = \frac{1}{N}\sum_{i=0}^{p} n_i,$$

où n_i est le nombre de pixels ayant la valeur i. Cette opération a donc pour but que l'image transformée aie autant de pixels dans chaque niveau, donc d'utiliser toute la dynamique tout en utilisant plus de niveaux pour détailler les zones à forte population. Cette transformation associe aux pixels de niveau p un nouveau niveau q tel que

$$q = E(p) = \frac{(Q-1)}{1 - H(p_0)}\sum_{i=0}^{p} H(i) - H(p_0),$$

en notant p_0 le premier niveau occupé dans H, et Q le nombre de niveaux de l'image de sortie. Le problème de cette méthode est que par principe elle attribue des niveaux de gris proportionnellement à la population en entrée. La figure 1.4 montre l'effet d'une égalisation d'histogramme sur une image infrarouge à forte dynamique, et contenant un point chaud, et la figure 1.5 illustre un cas où l'égalisation échoue à rendre les objets plus visibles.

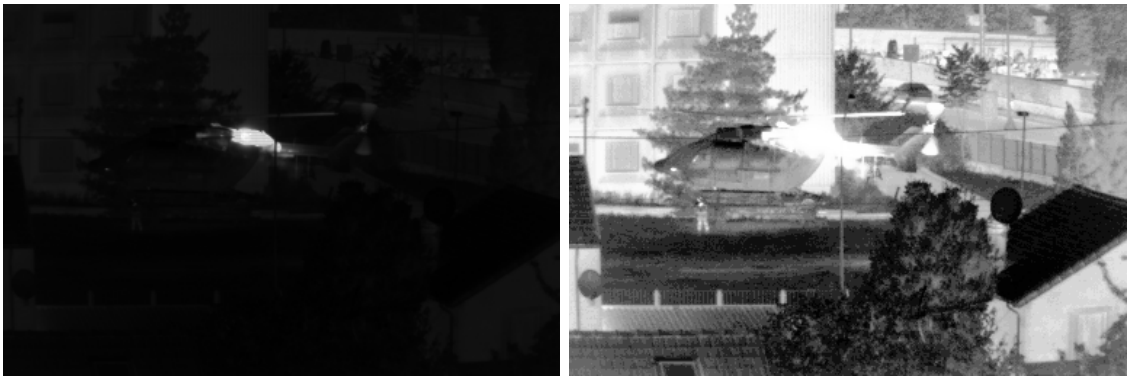


FIG. 1.4 – A gauche, une image infrarouge d'un hélicoptère au décollage, après avoir effectué un ajustement linéaire. A droite, la même image ayant subi une égalisation d'histogramme.

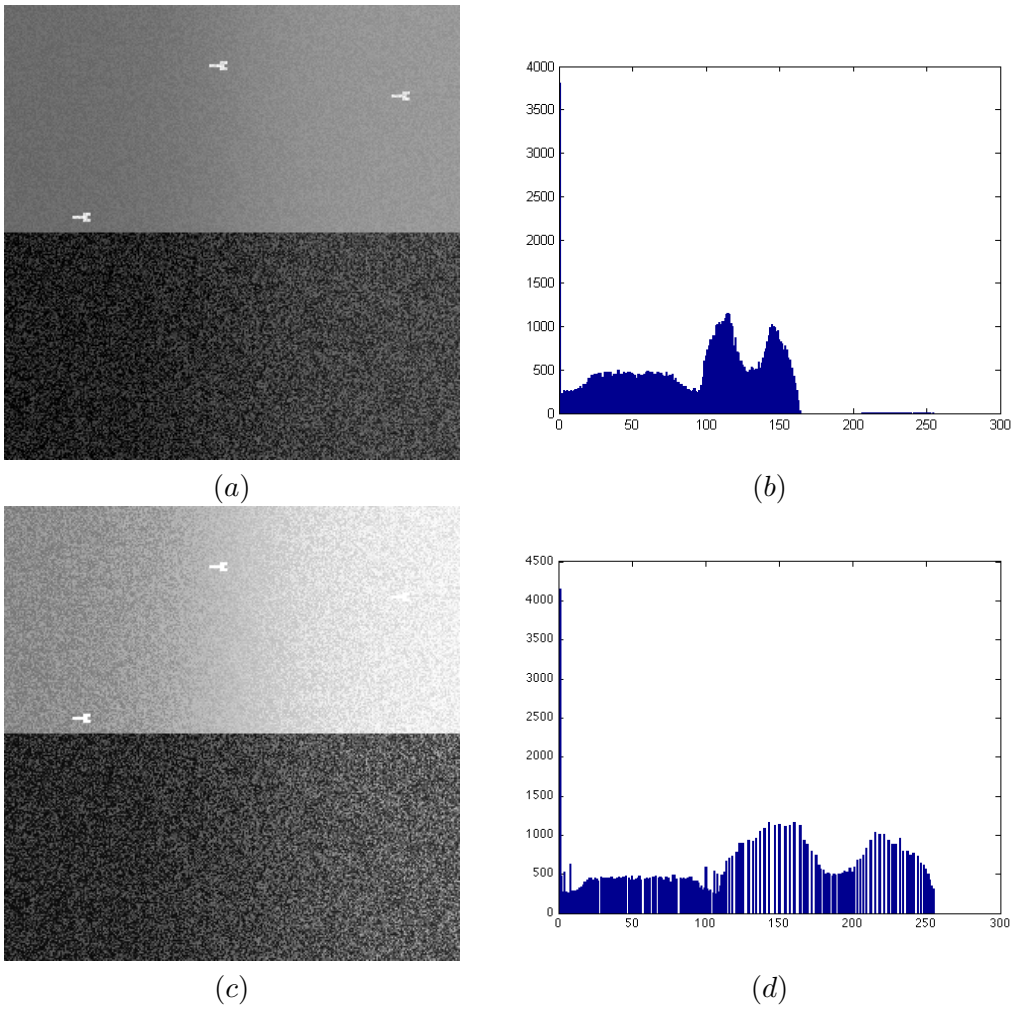


FIG. 1.5 – Illustration de la limite de l'égalisation d'histogramme. (a) et (b) : une image contenant des objets bien visibles dans la partie haute, et son histogramme. (c) et (d) : résultats de l'égalisation d'histogramme.

Egalisation contrainte

Pour palier aux problèmes de compression des niveaux peu peuplés de l'égalisation d'histogramme, on peut décider d'égaliser non pas l'histogramme H de l'image, mais un histogramme booléen H_b servant de marqueur des niveaux suffisamment peuplés, c'est à dire dont le remplissage est supérieur à un seuil fixé ω :

$$H_b(p) = \begin{cases} 1 \; si \; H(p) \geq \omega, \\ 0 \; si \; H(p) < \omega. \end{cases}$$

Cela a pour effet de combler les vides de l'histogramme. La figure 1.6) montre l'application d'un tel traitement à une image réelle ayant très peu de dynamique, et étant très bruitée. On peut aussi utiliser une méthode mixte entre l'égalisation $p \rightarrow E(p)$ et l'ajustement linéaire $p \rightarrow L(p)$, c'est à dire $T(p) = \rho E(p) + (1-\rho)L(p), \rho \in [0,1]$.

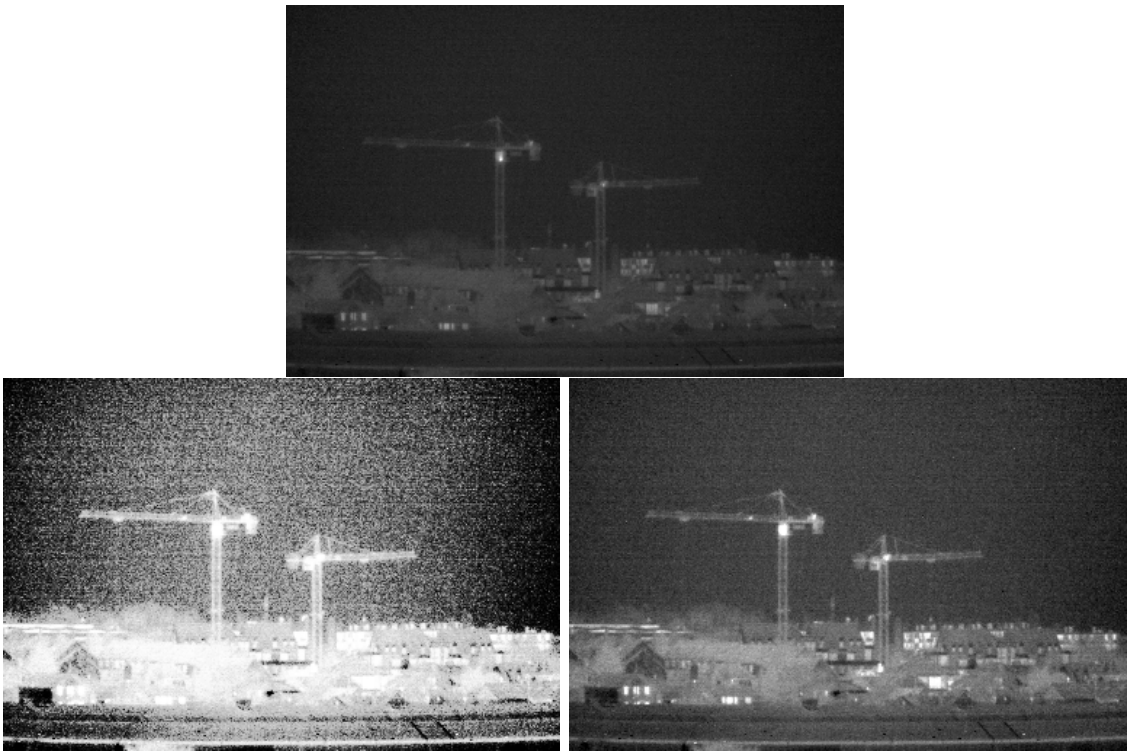


FIG. 1.6 – Haut : ajustement linéaire d'une image urbaine infrarouge d'une grue, ayant un niveau de bruit élevé par rapport à la dynamique ; Gauche : égalisation d'histogramme, le bruit est amplifié ; Droite : équirépartition, le contraste est amélioré dans la zone utile sans augmenter le bruit. Un seuil de 5 a été ici utilisé, car une partie de la dynamique était couverte à faible densité.

Un traitement local : l'uniformisation de la variance

Les traitements présentés ci-dessus étaient tous des traitements globaux de l'image, qui étaient construits à l'aide d'une transformation de la LUT (*Look Up Table*). Ils présentent des défauts, notamment dans le cas de petits objets à faible dynamique, comme montré dans les exemples précédents, et un algorithme de traitement local est alors préférable. Un algorithme d'uniformisation de la variance [94, 87] est utilisé, et permet d'uniformiser la moyenne et la variance locale de l'image.
Ce traitement nécessite donc un filtre passe-bas spatial W, par exemple un filtre triangulaire, qui est intéressant car séparable et apodisé. A partir de là, il faut calculer les moyenne et la variance locales de l'image $I[n_1, n_2]$:

$$\widehat{\mathrm{Moy}}_I[n_1, n_2] = (I * W)[n_1, n_2],$$

$$\widehat{\mathrm{Var}}_I[n_1, n_2] = ((I - \widehat{\mathrm{Moy}}_I)^2 * W)[n_1, n_2].$$

L'image traitée est finalement l'image centrée localement par les moyennes locales et normalisée par la variance, à laquelle on force une moyenne M_0 et une variance V_0 prédéfinies :

$$I_{\mathrm{filt}}[n_1, n_2] = \sqrt{\frac{V_0}{\widehat{\mathrm{Var}}_I[n_1, n_2]}}(I[n_1, n_2] - \widehat{\mathrm{Moy}}_I[n_1, n_2]) + M_0.$$

L'image est bien sûr saturée pour les points dont les valeurs de sortie seraient en dehors de la dynamique souhaitée. Le point clé du filtrage réside dans le choix de la fenêtre, qui doit être lisse pour éviter les discontinuités, assez grand pour diluer les artéfacts, et assez petit pour révéler les informations de faible contraste. Il est possible de trouver le meilleur compromis en étudiant le biais et la variance de différentes fenêtres et d'adapter leur taille [133], mais pour des contraintes d'implémentation et d'exécution en temps réel, un filtre triangulaire de taille fixée empiriquement est utilisé. Afin d'éviter une sur-amplification des zones constantes de l'image, une limite de gain est définie par $G_{\max} = 1/V_{\min}$. Cette limite permet d'une part de ne pas trop amplifier les zones homogènes, que l'on définit comme ayant une variance inférieure à σ_h^2, où σ_h est une proportion donnée de la dynamique de la scène, et d'autre part de ne pas amplifier le bruit, de variance σ_b^2, soit par exemple

$$V_{\min} = \max\left(\sigma_h, \sigma_b\right)^2 .$$

A noter que l'on peut aussi appliquer ce traitement plusieurs fois de suite à plusieurs échelles afin d'améliorer le rendu final. Les figures 1.7 et 1.8 montrent les résultats sur des données simulées puis réelles sur des images fortement contrastées ou à faible dynamique.

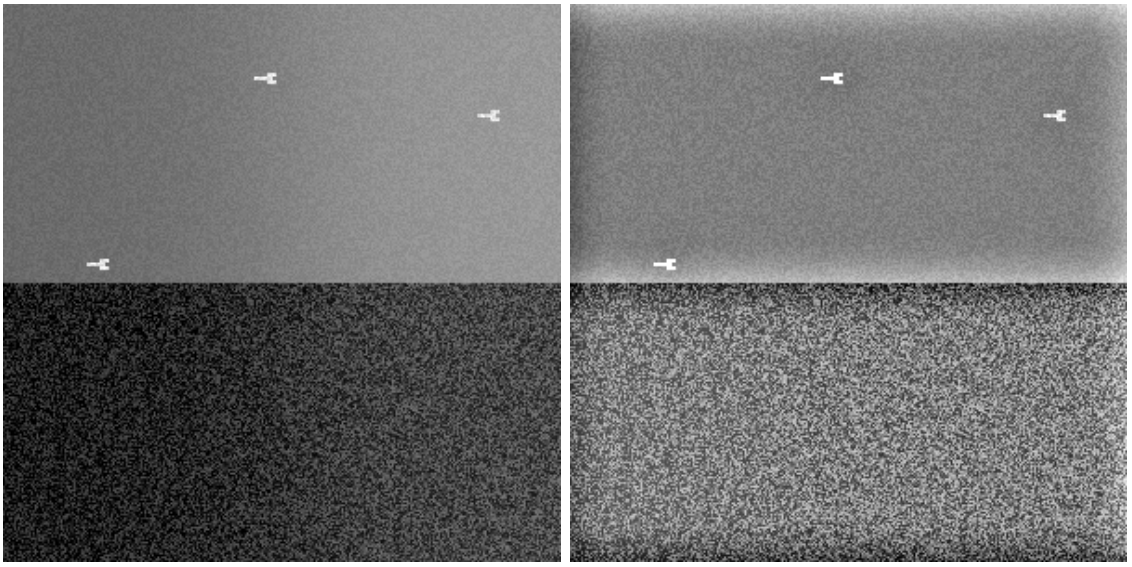


FIG. 1.7 – A gauche, l'image synthétique précédemment utilisée, ayant un niveau de bruit très fort dans la partie basse, et des objets dans la partie haute. A droite, cette image après filtrage par uniformisation de la variance. Les dégradés ont disparu, les cibles sont plus contrastées, le bruit de la partie haute n'a pas été amplifié, et celui du bas l'a été car son étalement (100 niveaux sur 256) est trop important, il est donc considéré comme de l'information.

1.4.2 Correction des pixels défectueux

Cette opération, de par sa nécessité préalable aux traitements suivants, est effectuée très en amont en traitement d'image, simultanément à la calibration des gains et offsets de l'image. Elle correspond à l'affectation d'une valeur plausible aux pixels défectueux. Les zones défectueuses étant généralement de faibles surfaces (pas de blocs manquants), et chaque pixel manquant ayant des voisins connus, on peut remplir les valeurs manquantes par filtrage médian, c'est à dire par la valeur médiane sur un voisinage donné, généralement

FIG. 1.8 – Haut : L'image précédente de l'hélicoptère après égalisation d'histogramme (il s'avère que cette image a un meilleur rendu qu'avec égalisation à cause de la forte dynamique du point chaud), et après l'algorithme d'uniformisation de la variance : la dynamique de rendu est utilisée dans toutes les zones de l'image, ce qui permet d'améliorer le contraste des zones sombres, ainsi que de distinguer l'intérieur du point chaud. Bas : L'image de la grue, après égalisation d'histogramme, et après uniformisation de la variance : la dynamique est améliorée sur les maisons (comme avec l'égalisation bien paramétrée), mais surtout le bruit n'a pas été amplifié.

les pixels valides parmi les huit pixels voisins.

1.4.3 Estimation de mouvement

La plupart des autres traitements se placent en aval d'une étude du mouvement global de la scène [7], qui utilise un modèle de déplacement de la scène dans l'image. Dans la suite, on appellera flot optique le mouvement apparent de la scène dans une vidéo. On modélise le déplacement des pixels de l'image, selon un modèle de mouvement, sa complexité allant d'une simple translation à l'homographie. lorsque le mouvement est de faible amplitude, l'équation du flot optique stipule que l'intensité d'un point de la scène change peu lorsque cette dernière subit un déplacement faible,

$$I(x, y, t) = I(x + dx, y + dy, z + dz)$$

On suppose que l'on peut faire un développement en série de Taylor de l'intensité observée. On obtient donc l'équation de contrainte du mouvement apparent

$$\nabla I \cdot \mathcal{V} + I_t = 0, \tag{1.1}$$

où I_t est la dérivée temporelle de l'image, ∇ désigne le gradient spatial, et $\mathcal{V}$ le champ de vitesses, ou flot optique. Cette équation ne suffit pas à estimer le flot optique $\mathcal{V}$, car il ne peut être estimé que dans la direction du gradient (il s'agit du problème d'ouverture). Il existe plusieurs méthodes pour résoudre le problème, par exemple la méthode de Hord et Schunck [84], qui est une méthode itérative cherchant à minimiser les variations spatiales de $\mathcal{V}$, par le terme de régularisation $\|\nabla\mathcal{V}_x\|_2^2 + \|\nabla\mathcal{V}_y\|_2^2, \mathcal{V} = [\mathcal{V}_x, \mathcal{V}_y]^T$. Une autre approche, non itérative, est celle de Lucas-Kanade [100], qui suppose que le champ de vitesses est localement constant sur un voisinage Ω_a du point a considéré, et le déplacement n'est estimé que si le gradient varie suffisamment en terme de direction. L'énergie à minimiser est

$$J_{LK}(a) = \sum_{b\in\Omega_a} W_a(b)^2(\nabla I \cdot \mathcal{V} + I_t)^2,$$

où W_a désigne une fenêtre – gaussienne par exemple – pondérant le voisinage de a. Si on recherche un mouvement global (c.à.d. $\mathcal{V}$ correspondant à un mouvement unique sur toute l'image) on peut réduire les calculs en ne calculant $\mathcal{V}$ que sur N points significatifs de l'image, ayant un gradient spatial local à la fois fort et sur plusieurs directions. Le nombre d'inconnues recherchées dépend du modèle de mouvement considéré, qu'il s'agisse d'une translation pure, que l'on ajoute une rotation ou que l'on considère un mouvement affine. Dans le cas d'un mouvement plus compliqué de la scène, lorsqu'il y a beaucoup d'objets mobiles ou lorsque la scène contient des structures à des distances fortement variables (par rapport à la distance d'observation), le mouvement ne peut plus être modélisé globalement, et il faut en faire une estimation locale [85]. On doit donc résoudre 1.1 en tout point de la scène. Afin de pouvoir résoudre le problème en tout point et non uniquement dans les zones ayant suffisamment de variation du gradient, on ajoute les équations $\varepsilon\mathcal{V}^n = \varepsilon\bar{\mathcal{V}}^{n-1}$, avec $\bar{\mathcal{V}}$ une version lissée de $\mathcal{V}$, qui permettent, avec un poids ε dépendant du gradient local, de régulariser le problème lorsque l'on le résout itérativement par moindres carrés. Lorsque le mouvement est trop important, on utilise cette méthode sur des versions multi-échelles filtrées [10] de notre image en recalant les échelles les unes sur les autres jusqu'à la plus fine, en utilisant les résultats précédents comme initialisation pour les échelles suivantes, ce qui accélère considérablement le temps de calcul, jusqu'à l'échelle la plus fine qui donne le mouvement dense.

1.4.4 Stabilisation

L'application directe de la connaissance du mouvement est la stabilisation de l'image [116], ce qui améliore le confort de l'utilisateur. Une fois les vitesses de déplacement connues, elles sont intégrées afin de connaître le vecteur déplacement total à appliquer pour recaler l'image par rapport à une image de référence. Dans le cas d'un mouvement global translation+rotation, le champ de vitesses $\mathcal{V}$ se décompose en une translation $\mathcal{T}$ et une rotation $\mathcal{R}$. A partir de $(\mathcal{T}^{(n)}, \mathcal{R}^{(n)}), [\mathcal{T}_x^{(n)}, \mathcal{T}_y^{(n)}]^T$ pour l'image courante et du vecteur déplacement $(\mathbf{T}^{(n)}, \mathbf{R}^{(n)}), \mathbf{T}^{(n)} = [\mathbf{T}_x^{(n)}, \mathbf{T}_y^{(n)}]^T$ de l'image précédente par rapport à l'image référence, on en déduit le nouveau vecteur déplacement pour l'image courante, avec un pas temporel unitaire :

$$\begin{aligned}
\mathbf{T}_x^{(n)} &= \mathcal{T}_x^{(n)} + \mathbf{T}_x^{(n-1)} \cdot \cos(\mathcal{R}^{(n)}) + \mathbf{T}_x^{(n-1)} \cdot \sin(\mathcal{R}^{(n)}),\\
\mathbf{T}_y^{(n)} &= \mathcal{T}_y^{(n)} + \mathbf{T}_y^{(n-1)} \cdot \cos(\mathcal{R}^{(n)}) - \mathbf{T}_y^{(n-1)} \cdot \sin(\mathcal{R}^{(n)}),\\
\mathbf{R}^{(n)} &= \mathcal{R}^{(n)} + \mathbf{R}^{(n-1)},
\end{aligned}$$

On ajoute alors un paramètre de stabilisation α compris entre 0 (aucune stabilisation) et 1 (stabilisation totale, le champ affiché reste le champ de départ). Il s'agit d'un filtrage

passe bas du déplacement au premier ordre, c'est à dire que le déplacement (on considère ici uniquement l'axe x) total à appliquer à l'image courante n'est pas $\mathbf{T}^{(n)}$, mais $\mathbf{T}^{(n)}_{\text{filt}} = \mathbf{T}^{(n-1)} + \alpha \mathcal{T}_x^{(n)}$. Cela a pour effet de corriger les mouvements haute fréquence pour pouvoir balayer la scène lentement sans subir les vibrations du porteur.
De plus, un décalage maximal $\mathbf{T}_{\max}$ (retard spatial) par rapport au déplacement $\alpha = 1$ est défini, au delà duquel la stabilisation *décroche* et le déplacement sature à sa valeur maximale $\mathbf{T}^{(n)}$ pour permettre le suivi de l'observation, ce qui évite à l'utilisateur de désactiver manuellement la stabilisation lorsqu'il souhaite effectuer des mouvements amples :

$$\begin{cases} \mathbf{T}^{(n)}_{\text{filt}} & \text{si } |\mathbf{T}^{(n)} - \mathbf{T}^{(n)}_{\text{filt}}| < \mathbf{T}_{\max}, \\ \mathbf{T}^{(\text{n})} & \text{sinon.} \end{cases}$$

Une simulation a été effectuée, et la figure 1.9 montre le déplacement que l'observateur souhaite effectuer avec sa caméra, le mouvement réel dû à ses tremblements, et la trajectoire stabilisée. Pour éviter d'avoir des bords noirs lorsque l'on stabilise une séquence, une image de fond – mémoire – est calculée à l'aide des images précédentes à laquelle on applique la même transformation.
Afin d'effectuer ce recalage de l'image rapidement, une lecture des pixels décalée de la partie entière de la translation est effectuée pour compenser le mouvement à l'entier près. La partie décimale du décalage est alors prise en compte dans l'interpolation de l'image qui peut maintenant se faire sur un voisinage restreint.

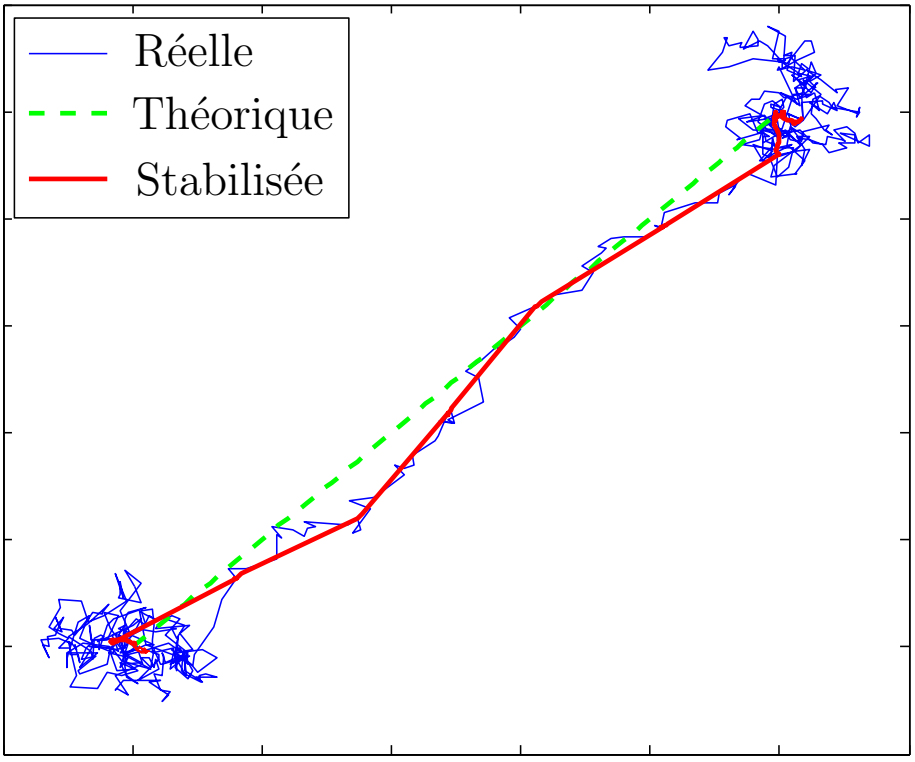


FIG. 1.9 – Effet de la stabilisation filtrée sur une ligne de visée synthétique. le déplacement théorique est en tirets : on observe un point fixe, puis on se déplace vers un autre. Le déplacement réel dû aux vibrations du porteur est en trait plein fin, et le déplacement stabilisé est en trait plein épais. On remarque le décrochage de la stabilisation dans les forts déplacements pour permettre le suivi.

1.4.5 Débruitage et calibrage sur scène

Maintenant que l'image est stabilisée, plusieurs autres traitements deviennent possibles tels la post-accumulation, la détection d'objets mobiles (cf. section 1.4.6), le calibrage sur scène et la détection des pixels défectueux.
La post-accumulation [86] de paramètre $\gamma \in [0,1]$ consiste à effectuer un filtrage passe-bas sur une séquence d'images recalées dans le même repère, $I_{\mathrm{PA}}^{(n)} = I_{\mathrm{PA}}^{(n-1)} + \gamma(I^{(n)} - I_{\mathrm{PA}}^{(n-1)})$ pour réduire le niveau du bruit dans les zones où l'on suppose que la scène n'a pas changé (i.e. absence d'objets mobiles).
La réponse des capteurs IR n'étant pas uniforme ni constante dans le temps, il est impératif de calibrer les appareils à chaque utilisation. Cela est dû entre autres à des défauts de fabrication des matrices photosensibles, et au flux parasite qui dépend notamment de la température de l'atmosphère et de celle du boîtier dans lequel est fixé le capteur. Une partie de la calibration est déterministe, dépend du temps d'intégration et de la température et peut être calculée avec une modélisation photométrique. Cependant cette correction n'est pas suffisante et il faut ajouter un traitement en aval. La calibration fine s'effectuait avant sur le site d'utilisation, et devait parfois être refaite plusieurs fois en attendant que le système atteigne un équilibre. Elle était longue et perturbait l'utilisation des produits, l'utilisateur devant par exemple effectuer une défocalisation pour obtenir une image plate afin de recalibrer l'appareil. On a donc créé une méthode pour les calibrer automatiquement pendant l'utilisation [16]. Ce calibrage sur scène consiste à utiliser le fait que le champ observé par la caméra change, et donc que plusieurs pixels identifiés par la détection de vitesses, *voient* un même point de la scène. On peut donc mesurer la différence de réponse de plusieurs pixels et les uniformiser. En notant $I(x,y,t)$ la scène à l'instant t, (dx,dy) le vecteur déplacement de (x,y) entre t et $t' = t+dt$, on peut écrire l'équation du flot optique

$$I(x,y,t) = I(x+dx, y+dy, t').$$

En notant $O(x,y)$ les offsets individuels supposés constants sur la durée du calibrage, $n(x,y,t)$ les bruits à chaque instant, on obtient le signal réel $s(x,y,t) = I(x,y,t)+O(x,y)+n(x,y,t)$. On s'intéresse alors à la différence de réponse entre deux pixels (x,y) et $(x+dx, y+dy)$ observant le même point de la scène aux instants respectifs t et t' :

$$\begin{aligned} &s(x,y,t) - s(x+dx, y+dy, t') \\ &= O(x,y) - O(x',y',t') + n(x,y,t) - n(x',y',t'). \end{aligned} \tag{1.2}$$

Intégrer $O(x'(x,y,t), y'(x,y,t))$ sur le temps $t \in T$ revient à l'intégrer sur l'ensemble des positions $W(x,y) = \{(x'(x,y,t), y'(x,y,t)) | t \in T\}$ pondérées par un poids $p(x,y)$ qui prend en compte les interpolations et le fait qu'un point de la scène puisse être visité plusieurs fois par le vecteur déplacement. Le bruit étant supposé stationnaire – donc ergodique – et à moyenne nulle, on peut donc réécrire l'équation (1.2) en moyennant sur l'intervalle T :

$$\begin{aligned} &\langle s(x,y,t) - s(x+dx, y+dy, t') \rangle_{t \in T} \\ &= O(x,y) - \langle p(x,y) O(x',y',t') \rangle_{(x,y) \in W(x,y)}, \end{aligned} \tag{1.3}$$

en notant ici $\langle \cdot \rangle_\Omega$ la moyenne statistique sur l'ensemble Ω, et en remarquant qu'il ne s'agit pas en réalité d'une égalité, mais que le bruit a été réduit d'un facteur $\sqrt{T}$. On corrige donc les pixels en retranchant la quantité (1.3), ce qui uniformise les offsets à l'échelle de la taille de $W(x,y)$. On peut augmenter cette étendue, en effectuant des chaînes de propagation, c'est à dire en prenant en compte tous les pixels dont une suite de vecteurs vitesse le fait passer au point (x,y). On peut ainsi apparier des pixels éloignés dans l'image, et on effectue

cette opération pour toutes les trajectoires de la séquence.
Dans le même temps, lorsque l'offset calculé d'un pixel dépasse un niveau de bruit donné (caractéristique du détecteur) dans une zone uniforme de l'image, on identifie ce pixels comme ayant un comportement aberrant et on l'ajoute dans la cartographie des pixels défectueux : sa valeur ne sera plus utilisée pour les traitements, mais on lui associera une valeur par la méthode de gestion des pixels morts choisie.

1.4.6 Détection d'objets mobiles

L'algorithme d'Aide à la Détection d'Objets Mobiles (ADOM) a pour but de mettre en valeur dans l'image (couleur ou noir et blanc), les éléments mobiles, pour des produits telles les jumelles IR multifonctions dont l'image est stabilisée.
Premièrement, on gère la détection à proprement parler, puis la visualisation qui combine les données de l'image et celles du mouvement. Pour conserver l'indépendance des parties, la détection renvoie deux images 8 bits pour le plan image et le plan des objets mobiles. La détection s'appuie sur la différence entre l'image courante et l'image stabilisée précédente (image de référence construite avec plusieurs images passées). On applique un seuillage à cette image de différence pour obtenir le masque des objets mobiles. Le seuil est défini comme le maximum entre le niveau de bruit du capteur et le gradient local spatial de l'image. Pour une implémentation rapide, le gradient spatial est défini comme le maximum des gradients discrets par différences finies décalées à droite et centrées appliqués sur les deux directions. Afin de conserver l'ADOM dans son domaine de fonctionnement, un critère de désactivation automatique est défini par un booléen qui s'active lors de la déstabilisation (ou décrochage de la stabilisation), lors d'une défocalisation, ou lorsqu'un nombre d'objets mobiles limite a été atteint. Afin de réduire la perturbation due à la présence d'objets mobiles, les calculs de la différence ne sont effectués que sur un masque M où l'on a marqué les zones fixes.
Les objets mobiles détectés peuvent être visualisés en couleurs en affichant l'image I en niveaux de gris, et les zones mobiles en couleurs. On peut aussi rester sur un afficheur en niveaux de gris, en faisant clignoter les zones détectées comme mobiles.
Le masque d'objets mobiles est réutilisé directement dans la post-accumulation pour le débruitage, qui n'est activée que pour les zones stationnaires. Par ailleurs, on peut améliorer le taux de détection et le taux de fausse alarme en ajoutant un suivi des objets.

1.4.7 Zooms numériques

Un zoom numérique est effectué par interpolation d'une zone de l'image sur une grille plus fine. Les zooms classiques sont les zooms discrets, c'est à dire des multiplications par un facteur entier. Mais il en existe d'autres tels que le zoom continu, ou le zoom bicapteur qui utilise une image grand champ et une petit champ.

Zoom discret

Il s'agit d'un zoom ayant un facteur entier, il est effectué par interpolation avec un noyau d'interpolation défini à l'avance, et dont on utilise toujours les mêmes points pour interpoler. On peut donc stocker les coefficients d'interpolation de chaque niveau de zoom ($\times 2$,$\times 4$,$\times 8$, par exemple) dans une LUT mémoire, et le calcul s'effectuera rapidement.

Zoom continu

Cette fois, le facteur de zoom est un réel, ou du moins prend de nombreuses valeurs entre deux entiers consécutifs, et il n'est plus possible d'enregistrer les valeurs des coefficients d'interpolation pour chaque niveau de zoom. On pourrait choisir de calculer la valeur de la fonction d'interpolation en temps réel, mais la puissance de calcul embarquée dans les produits n'est pas forcément suffisante, on préfère donc plutôt enregistrer directement les valeurs de la fonction d'interpolation sur une grille assez fine pour le zoom maximal considéré.
L'interpolation est une convolution discrète construite ici avec un filtre séparable, produit tensoriel de filtres 1-D. Les interpolations sont donc effectuées sur un axe puis sur l'autre, ce qui est plus rapide, et permet de n'enregistrer que les coefficients d'une fonction d'interpolation mono-dimensionnelle. Les fonctions d'interpolations sont choisies comme le meilleur compromis entre le facteur de zoom nécessaire et de la charge calculatoire associée. Pour un zoom inférieur à deux, une interpolation bilinéaire peut être suffisante visuellement, mais au delà on utilise des splines dont on augmente le degré avec le zoom.
Lorsque le zoom est continu, il existe différentes manières de passer d'un facteur à un autre plus élevé. On peut le faire grossir linéairement, ce qui donne l'impression que le zoom ralentit, ou exponentiellement, ce qui correspond à un grandissement constant, comme appliquer un zoom fixe à intervalles de temps constants.

1.4.8 Déconvolution

Lors de l'acquisition d'une scène par un capteur derrière un dispositif optique, l'image subit des dégradations (cf. sections 1.2.2 et 1.2.3) dues à la fonction d'étalement du point (PSF) globale du système, et qui peuvent être réduites par déconvolution. Le modèle de dégradation est une convolution de la scène s par la PSF h du système et l'ajout de bruit n supposé gaussien :

$$I(x, y) = s * h(x, y) + n(x, y). \tag{1.4}$$

Plusieurs approches sont possibles [166, 129, 101, 71, 119] pour estimer s : une méthode directe comme Wiener régularisé par ondelettes ou paquets d'ondelettes, itérative comme Richardson-Lucy, ou variationnelle. Par la suite, on notera $\tilde{\cdot}$ les estimées.

Filtrage de Wiener

Le filtre de Wiener [166] consiste à trouver l'estimateur linéaire qui minimise l'erreur quadratique moyenne $\|s - \tilde{s}\|^2$. La solution bien connue se calcule dans le domaine de Fourier et fournit le filtre inverse $\hat{g}$ en Fourier par

$$\hat{g} = \frac{\hat{h}^*}{|\hat{h}|^2 + \frac{\langle|\hat{n}|^2\rangle}{\langle|\hat{s}|^2\rangle}},$$

où $\hat{\cdot}$ représente la transformée de Fourier, et $\langle|\hat{\cdot}|^2\rangle$ la densité spectrale de puissance. Cela correspond à un filtre inverse régularisé par le rapport signal à bruit local dans le domaine spectral, et est aussi la solution du maximum a posteriori avec des a priori gaussiens indépendants sur le bruit et le signal. D'autres régularisations peuvent s'avérer utiles, car d'une part on ne connaît généralement pas la densité spectrale de puissance du signal, et d'autre part lorsque le flou est important, le filtre de Wiener crée des artéfacts importants dans les zones à fort gradient. On peut alors remplacer le SNR local en fréquence $\frac{\langle|\hat{n}|^2\rangle}{\langle|\hat{s}|^2\rangle}$ par un scalaire indépendant de la fréquence, ce qui accélère le calcul. Cependant, le choix de

cette constante est très sensible, car si elle est choisie trop grande ou trop petite, l'image ne sera pas déconvoluée ou le bruit sera amplifié. En pratique, ce coefficient est estimé sur l'image dégradée, soit globalement, soit sur un voisinage du point considéré. Une des méthodes consiste à considérer que puisque le bruit et le signal inconnu sont indépendants, leurs variances s'ajoutent, et donc $\langle|\hat{s}|^2\rangle \approx (\langle|\hat{I}|^2\rangle - \langle|\hat{n}|^2\rangle)_+$, en notant $(x)_+$ la partie positive de x.
Suivant la décomposition en wavelet-vaguelette de D.L. Donoho [45], le résultat du filtrage de Wiener peut être amélioré en le combinant à un débruitage par paquets d'ondelettes, dont un cas particulier est d'utiliser les ondelettes miroir de Khalifa *et al.* [88] qui sont adaptées à la réponse fréquentielle de l'inverse de l'opérateur de convolution. En effet, on peut effectuer un filtrage de Wiener avec un paramètre de régularisation plus faible que précédemment, ce qui restaure mieux les hautes fréquences mais amplifie le bruit, puis transformer le résultat dans une base de paquets d'ondelettes et effectuer le débruitage du bruit coloré par compression des coefficients (seuillage doux par exemple). La figure 1.10 montre le résultat d'un filtre de Wiener et du débruitage en paquets d'ondelettes.

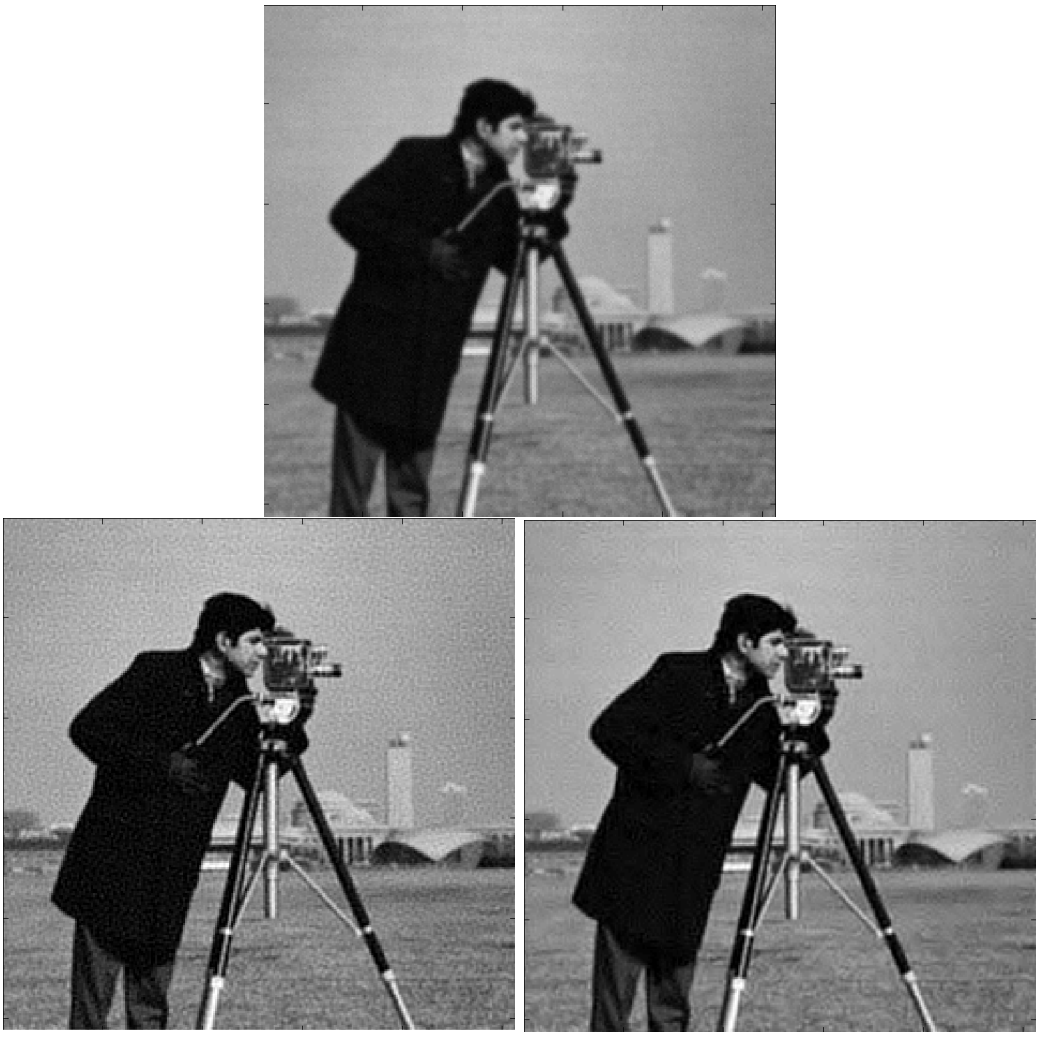


FIG. 1.10 – En haut : l'image *cameraman* corrompue par une PSF gaussienne de variance 1 et un bruit additif gaussien de variance 3.3 (BSNR de 25) ; à gauche : le résultat d'un filtrage de Wiener ; à droite : après débruitage par paquets d'ondelettes.

Algorithme de Richardson-Lucy

Richardson et Lucy [129, 101] ont proposé un algorithme itératif pour résoudre le problème de déconvolution, en cherchant la solution correspondant au maximum de vraisemblance sous un bruit de Poisson :

$$\hat{s} \in \arg\min_s \int_\Omega I(x,y) - (h * s)(x,y) \log\left((h * s)(x,y)\right) dxdy,$$

où Ω est le domaine de définition de l'image. L'itération principale de l'algorithme est la suivante :

$$s^{n+1}(\xi,\eta) = s^n(\xi,\eta) \int_\Omega \frac{I(x,y)}{s_h^n(x,y)} h(x-\xi, y-\eta) dxdy,$$

où

$$s_h^n(x,y) = \int_\Omega s^n(\xi,\eta) h(x-\xi, y-\eta) d\xi d\eta.$$

On initialise cet algorithme avec $s^0 = I$ ou une version lissée. Le problème principal de cet algorithme est qu'en présence de bruit gaussien, il est instable, et la régularisation se fait en stoppant la déconvolution après un nombre d'itérations à estimer en fonction du bruit et du contenu fréquentiel de l'image.

Approches variationnelles

Ces approches correspondent à chercher le minimum d'une fonction de coût comprenant un terme d'attache aux données, métrique liant la solution aux données par le biais du problème direct, et un terme de régularisation, mesure de la régularité, ou correspondance au modèle que l'on impose à la solution.
Le terme d'attache aux donnée, dans un cas où les bruits sont gaussiens, est l'écart asymptotique entre la donnée I et la solution courante $\tilde{s}$ telle qu'elle serait observée par h :

$$F_{\text{data fidelity}} = \|I - h * \tilde{s}\|_2^2.$$

Selon les a priori que l'on a sur la solution, on peut utiliser différents termes de régulation. Si l'image recherchée est lisse, on peut par exemple utiliser la régularisation de Tikhonov qui minimise $F_T = \|D^{(m)}\tilde{s}\|_2^2$, où $D^{(m)}$ est un opérateur de différentiation d'ordre m. Si $m = 0$, on obtient l'identité, qui correspond à chercher la solution d'énergie minimale, si $m = 1$ et $m = 2$, on obtient le gradient et le laplacien, qui pénalisent les variations de la solution et donc favorisent une solution lisse. Mais cette régularisation pénalise les contours dans les images, qui sont des bords francs à composantes haute fréquence. Une autre régularisation est la variation totale (TV), et la fonction associée est $F_{TV} = \|\nabla\tilde{s}\|_1$, qui tend à créer des images constantes par morceaux. D'autres régularisations plus générales comme celle de Geman et Reynolds [71] sont aussi utilisées, et donnent les meilleurs résultats.

1.4.9 Super-résolution

La super-résolution, qui permet à partir d'une séquence d'images basse résolution d'obtenir une image haute résolution, combine toutes les briques de traitement précédentes. Voir [14] et [63] pour des revues de l'état de l'art.
On considère le modèle de dégradation suivant : Soit z_i la séquence inconnue d'images haute résolution, H et P des opérateurs de convolution correspondant respectivement à

l'optique et à la surface d'un pixel basse résolution, et D un opérateur de décimation. La séquence observée I_i est donnée par

$$I_i = DPHz_i.$$

Les opérateurs P et H peuvent être absorbés par un seul opérateur de convolution G regroupant les deux PSF de H et P. Dans l'hypothèse classique où la scène z_i reste identique à $z = z_0$ à une transformation linéaire L près, le plus souvent un déplacement, on peut écrire

$$I_i = DGL_iz.$$

La résolution de ce problème inverse Dans le cas simplifié où l'opérateur L_i est une translation, L_i commute avec G. De même, si L_i contient aussi une rotation, et si la PSF associée à G est isotrope, les opérateurs commutent aussi. Cependant, G contenant la convolution par le pixel, généralement carré ou rectangulaire, il n'est pas isotrope, donc faire commuter les opérateurs est une approximation.
Sous cette hypothèse, la première étape est donc d'effectuer un décalage des images et de les fusionner sur une grille plus fine (shift and add). Les premières méthodes [89] effectuaient une interpolation de ces échantillons répartis de manière irrégulière pour obtenir la première estimée de l'image super-résolue, puis d'utiliser le problème direct pour simuler des observations basse résolution et d'utiliser les erreurs pour corriger l'image haute résolution. Les approches variationnelles permettent plus de flexibilité, les termes d'attache aux données ayant évolué de l'attache des moindres carrés produisant trop d'artéfacts à des attaches plus robustes [62] en norme ℓ_1, et les régularisations ont aussi évolué de celle de Tikhonov à des régularisations de type variation totale conservant les discontinuités des images. Ces approches itératives étant très consomptives en termes de ressources, nous nous sommes tournés vers des simplifications où le problème de super-résolution est séparé en un recalage des images sur un même référentiel suivi d'une interpolation et déconvolution itérative, comme proposé dans [64].
Plus récemment, des nouvelles méthodes utilisant les moyennes non locales [127], la régularité géométrique dans des dictionnaires adaptatifs [172] ont été développées, mais n'ont pas encore étudiées du point de vue de leur applicabilité en temps réel sur des produits portables.

1.5 conclusion

Ce chapitre a présenté rapidement la problématique liée à l'infrarouge et les traitements implémentés ou à l'étude pendant ce travail de thèse, et montre que malgré le souhait d'utiliser des techniques de restauration et traitement au niveau de l'état de l'art de la recherche publique, les contraintes d'exécution en temps réel sur des produits portables à faible consommation limitent souvent les possibilités de portage des algorithmes, et que des solutions plus simples sont souvent choisies. D'autre part, le domaine de la parcimonie n'est pas encore un acquis dans le monde industriel, en dehors d'une utilisation récente des transformées en ondelettes, et ce travail de thèse permettra d'apporter ce domaine à cette communauté.

Chapitre 2

Parcimonie

2.1 Introduction

Un signal discret est généralement représenté comme une suite d'échantillons, mais on sait qu'il existe de nombreuses représentations équivalentes comme par exemple le domaine fréquentiel. On change d'espace car la nouvelle représentation permet de scruter le signal sous un autre angle en en révélant des propriétés inaccessibles dans le domaine original.
Le but de certaines représentations est de faire ressortir l'information utile par rapport à l'environnement du signal. Pour atteindre cet objectif, il faut pouvoir représenter le signal dans un espace où il n'a que peu de coefficients non nuls, de manière à concentrer son énergie quand le bruit, décorrélé de tout continuera à étaler son énergie. La mesure du nombre de coefficients non nuls est appelée la ***parcimonie*** du signal dans l'espace considéré. Un signal est dit (strictement) parcimonieux si son nombre de coefficients non nuls est très faible face à la longueur de sa représentation.
On pourrait se demander si les signaux naturels, comme les images dites *naturelles*, peuvent effectivement être représentées avec peu de coefficients, ce qui revient à poser la question de leur compressibilité. On peut simplement observer que parmi la multitude de signaux possibles, les signaux et images réels ne représentent qu'une partie infime : quelle est la probabilité d'obtenir une image qui ressemble à autre chose qu'un bruit avec un générateur aléatoire de bruit uniforme ou gaussien ? Quasiment nulle. L'ensemble des images qui nous intéressent est extrêmement restreint, à faible entropie, et par conséquent les images naturelles sont fortement compressibles.
Depuis une quinzaine d'années, le concept de parcimonie a été étudié d'un point de vue théorique et utilisé dans de nombreuses applications du traitement du signal et des images, et la ***parcimonie*** d'un signal ou d'une image dans un domaine donné est devenu un critère de régularisation classique, et peut être liée à la régularité de la dite image.

2.2 Dictionnaire

Dans le paradigme de la parcimonie, on appelle ***dictionnaire*** $\Phi = (\boldsymbol{\phi}_i)_{i\in\{1,\cdots,P\}}$ la matrice contenant l'ensemble des fonctions prototypes $\boldsymbol{\phi}_i$ avec lesquelles on modélise le signal, et chacune de ces fonctions – appelée ***atome*** – est une colonne de Φ. Un exemple simple de dictionnaire est une base orthonormée, comme l'identité (base de diracs) ou Fourier, mais un dictionnaire peut contenir plus d'atomes que la longueur du signal, on parle alors de dictionnaire redondant.

Un signal $\mathbf{x} \in \mathbb{R}^N$ est représenté dans le dictionnaire $\Phi \in \mathbb{R}^{N \times P}, N \leq P$ par une superposition linéaire des atomes ϕ_i selon le modèle génératif linéaire

$$\mathbf{x} = \Phi\boldsymbol{\alpha} = \sum_{i=1}^{i=P} \boldsymbol{\alpha}_i \boldsymbol{\phi}_i, \boldsymbol{\alpha} \in \mathbb{R}^P.$$

On porte alors un a priori synthèse. Si $P > N$, le dictionnaire Φ est redondant et ses atomes sont linéairement dépendants. Il n'existe donc pas en général de représentation unique d'un signal dans Φ. Considérons maintenant un signal parcimonieux dans Φ, c'est à dire représenté par un vecteur $\boldsymbol{\alpha}$ avec peu de composantes non nulles

$$\mathbf{x} = \Phi\boldsymbol{\alpha} = \sum_{i \in \text{supp}(\boldsymbol{\alpha})} \alpha[i]\boldsymbol{\phi}_i, \quad \text{avec} \quad \text{card}(\boldsymbol{\alpha}) \ll P. \tag{2.1}$$

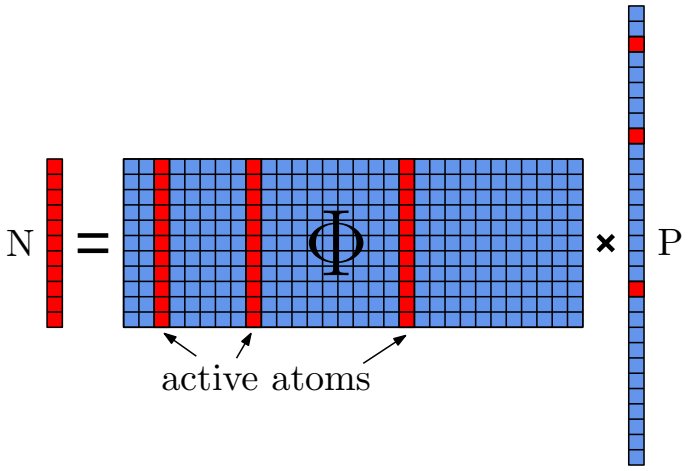


FIG. 2.1 – Illustration de la synthèse d'un signal parcimonieux de taille N dans un dictionnaire Φ redondant de taille $N \times P, P > N$ à partir de son vecteur de représentation $\boldsymbol{\alpha}$ de longueur P : $\mathbf{x} = \Phi\boldsymbol{\alpha}$. Le signal $\mathbf{x}$ n'est pas creux, mais son vecteur de représentation n'a que trois coefficients non nuls, correspondant aux trois atomes/fonctions en surbrillance dans la matrice Φ. Un exemple concret est donné en figure 2.2.

Cependant, les signaux réels sont rarement strictement parcimonieux, mais vérifient généralement l'hypothèse plus souple de compressibilité.

2.2.1 Compressibilité

Un signal $\mathbf{x}$ est dit ***compressible*** si la liste ordonnée de ses coefficients dans un dictionnaire Φ décroît comme une loi de puissance. Pour exprimer la compressibilité formellement, il faut considérer les signaux de dans une boule ℓ_p-faible de rayon donné. La norme ℓ_p-faible est définie pour $0 < p < \infty$ par

$$\|\boldsymbol{\alpha}\|_{w\ell_p} = \sup_{\epsilon > 0} \epsilon^p \left|\{i \text{ tel que } |\boldsymbol{\alpha}[i]| > \epsilon\}\right|.$$

Considérons donc un signal $\mathbf{x} \in w\ell_p(C)$ dont on ordonne la liste des coefficients $(\boldsymbol{\alpha}_i)_{i=1\cdots P}$: $\mathbf{x} = \Phi\boldsymbol{\alpha}$ par valeur absolue décroissante $(\boldsymbol{\alpha}_i^{\text{ord}})_{i \in \{1,\cdots,P\}}$: $|\boldsymbol{\alpha}_1^{\text{ord}}| \geq \cdots \geq |\boldsymbol{\alpha}_P^{\text{ord}}| \geq 0$. Alors

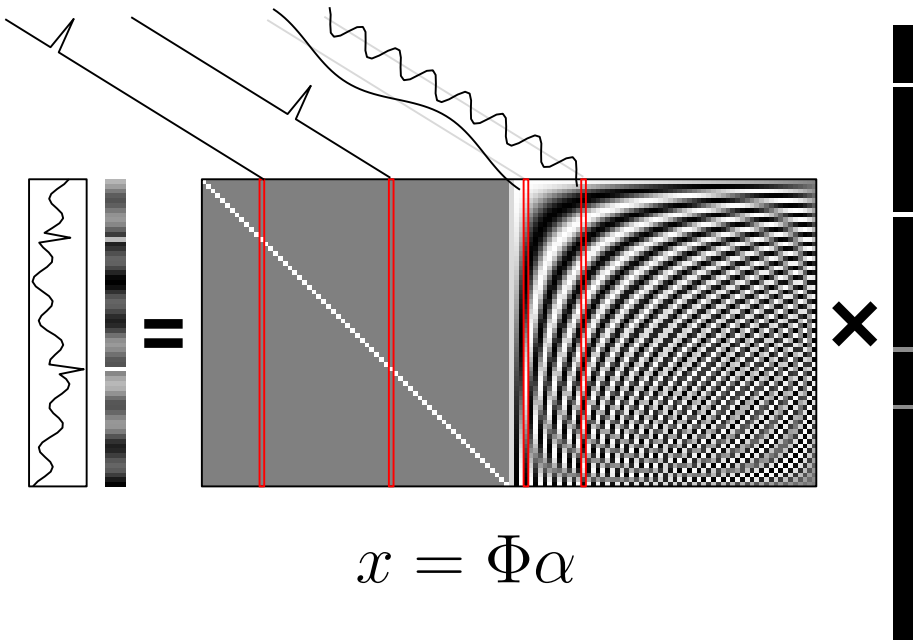


FIG. 2.2 – Exemple d'un signal parcimonieux dans le dictionnaire $\Phi = [I\ F]$ qui est la concaténation de l'identité et de Fourier. Le signal exemple a quatre coefficients non nuls dans ce dictionnaire, alors que sa représentation serait pleine en Fourier comme dans l'espace direct.

pour tout $i \in [1, P]$

$$|\boldsymbol{\alpha}_i^{\text{ord}}| \leq C i^{-1/p}. \tag{2.2}$$

Sous cette condition, étant donné un nombre d'atomes M, la meilleure approximation à M termes $\mathbf{x}_M$ de $\mathbf{x}$ est obtenue simplement en gardant les M premiers termes de la représentation ordonnée

$$\mathbf{x}_M = \sum_{i=1}^{M} \boldsymbol{\alpha}_i^{\text{ord}} \boldsymbol{\phi}_i, \tag{2.3}$$

et l'erreur d'approximation obéit elle aussi à une loi de puissance, dans le cas $p \leq 2$

$$\|\mathbf{x} - \mathbf{x}_M\|_2 \leq C(2/p - 1)^{-1/2} M^{1/2-1/p}. \tag{2.4}$$

Cette notion de compressibilité est extrêmement importante, car elle détermine notre capacité à représenter le signal au mieux, et donc à compresser, débruiter, etc. Les signaux réels ne sont en général pas strictement parcimonieux, mais sont considérés comme compressibles.

2.2.2 Quel dictionnaire choisir ?

D'après ce qui a été dit précédemment, si on veut que le signal (ou la classe de signaux) que l'on observe soient bien représentés, il faut un dictionnaire qui leur soit adapté. On veut donc mettre dans le dictionnaire toutes les formes ou structures qui apparaissent dans nos données. Cependant, les signaux réels étant très variés il faudrait construire un dictionnaire extrêmement long et le calcul de la représentation d'un signal dans un tel dictionnaire serait difficile en un temps raisonnable. Il est donc préférable d'avoir un dictionnaire sur

lequel on sache appliquer une transformée (i.e. Φ^T) rapide, et qui permette aussi une synthèse rapide (i.e. Φ) d'un signal à partir de ses coefficients. Des exemples simples sont les bases, par exemple la base standard ou Fourier. De nombreuses autres transformées ont été développées qui permettent d'approximer les images, comme les ondelettes [107] ou les curvelets [25], et forment des dictionnaires à transformée et reconstruction rapide. Les sections suivantes présentent ces décompositions, qui sont au cœur de cette thèse, et qui sont définies sur des images ou des données tridimensionnelles. Mais ces représentations fixes ne sont pas toujours adaptées aux données que l'on souhaite traiter, on peut alors choisir de créer un dictionnaire adapté à nos données. Il s'agit de l'apprentissage du dictionnaire, qui est rapidement introduit dans la section 2.8.
Le dictionnaire choisi ne dispose pas forcément d'opérateurs de transformation et reconstruction rapide, et dans le cas classique d'un dictionnaire redondant, la représentation d'un signal n'est en général pas unique. Il faut donc définir des procédures pour décomposer un signal dans un dictionnaire, ce qui est l'objet de la section suivante.

2.3 Décomposition d'un signal dans un dictionnaire redondant

Soit $\Phi \in \mathbb{R}^{N\times P}$ un dictionnaire constitué de P atomes $\boldsymbol{\phi}_i \in \mathbb{R}^N$ normalisées en norme ℓ_2 formant une famille génératrice de $\mathbb{R}^N$. On souhaite décomposer un signal $\mathbf{x}$ sur le dictionnaire redondant Φ, en tant que combinaison linéaire de ses atomes $\boldsymbol{\phi}_i$

$$\mathbf{x} = \Phi\boldsymbol{\alpha} = \sum_i \boldsymbol{\alpha}_i \boldsymbol{\phi}_i. \tag{2.5}$$

Chercher la représentation la plus parcimonieuse de $\mathbf{x}$ dans Φ correspond à résoudre le problème d'optimisation suivant

$$(P_0): \quad \min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \text{ tel que } \mathbf{x} = \Phi\boldsymbol{\alpha}. \tag{2.6}$$

Ce problème (P_0) étant un problème NP-difficile, il n'est en pas soluble en un temps de calcul polynomial. Donoho [53] a proposé de résoudre le problème approché (P_1) suivant, une version convexifiée de (P_0), en substituant la pseudo norme $\|.\|_0$ à la norme $\|.\|_1$:

$$(P_1): \quad \min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_1 \text{ tel que } \mathbf{x} = \Phi\boldsymbol{\alpha}. \tag{2.7}$$

Ce problème, appelé poursuite de base (Basis Pursuit) [32, 33], n'est évidemment pas équivalent au problème (P_0), mais il a été montré que sous certaines conditions de parcimonie, la solution de (P_0) pouvait être trouvée en résolvant (P_1). Il s'agit de l'identifiabilité ℓ_1.

Un outil fondamental permettant entre autres de quantifier le domaine d'équivalence des problèmes (P_0) et (P_1) est la cohérence d'un dictionnaire. Comme la définition suivante le montre, il s'agit d'une mesure de la corrélation intrinsèque du dictionnaire.
La ***cohérence*** μ ou d'un dictionnaire Φ quelconque [72, 53] est définie par

$$\mu(\Phi) = \max_{i\neq j} |\langle \boldsymbol{\phi}_i, \phi_j\rangle|, \ \phi_k \in \Phi.$$

Pour une BON la cohérence est naturellement nulle, et on peut montrer que pour tout dictionnaire Φ redondant,

$$\mu(\Phi) \geq 1/\sqrt{N}.$$

Un dictionnaire redondant Φ est ***parfaitement incohérent*** lorsque $\mu(\Phi) = 1/\sqrt{N}$. Cette égalité est atteinte par exemple pour l'union de l'identité et du dictionnaire de Fourier. Lorsque Φ est une trame, $\mu(\Phi) \geq \sqrt{\frac{P-N}{N(P-1)}}$, et l'égalité est atteinte pour les trames ajustés équi-angulaires [157].

Identifiabilité avec la cohérence Les auteurs de [53, 15, 65] ont montré qu'une condition suffisante pour avoir l'identifiabilité ℓ_1 d'un signal $\mathbf{x} \in \mathbb{R}^N$ K-parcimonieux est que

$$\|\boldsymbol{\alpha}\|_0 = K < \frac{1}{2}(1 + \mu(\Phi)^{-1}). \tag{2.8}$$

Ce type de bornes assure aussi la stabilité à un bruit borné et la robustesse à la compressibilité, où la constante 1/2 se voit remplacée par 1/4 [52]. Ce critère est issu d'une analyse au pire cas, et l'identifiabilité est en général obtenue pour une parcimonie de l'ordre de $O(\sqrt{N})$.
Un autre outil d'étude de l'unicité de la solution de (P_0) est le spark [51] d'un dictionnaire, défini comme la taille du plus petit sous ensemble linéairement dépendant de celui-ci, soit $Z(\Phi) = \min_{x \in \ker(\Phi), x \neq 0} \|x\|_0$. Mais le calcul de cette quantité est aussi complexe que la résolution de (P_0), et de plus il n'est pas adapté à l'extension aux cas bruités [52].

Contrainte d'isométrie restreinte Un dictionnaire Φ vérifie la contrainte d'isométrie restreinte (RIP) [28] de constante δ_K si pour tout signal $x = \Phi\boldsymbol{\alpha}$ K-parcimonieux dans Φ

$$(1 - \delta_K)\|\boldsymbol{\alpha}\|_2^2 \leq \|\Phi\boldsymbol{\alpha}\|_2^2 \leq (1 + \delta_K)\|\boldsymbol{\alpha}\|_2^2. \tag{2.9}$$

Des contraintes sur la constante RIP δ_K (ou plus souvent δ_{2K}) permettent d'obtenir l'identifiabilité uniformément sur tous les signaux K-parcimonieux, la stabilité au bruit et la robustesse aux signaux compressibles. Actuellement, la contrainte la plus forte [67] est $\delta_{2K} < 2(3 - \sqrt{2})$, sachant qu'on peut montrer [38] que pour $\delta_{2K} > 1/\sqrt{2}$ il existe au moins un vecteur K-parcimonieux non ℓ_1-identifiable, même sans bruit.
D'autres critères utilisent le support de la solution, comme l'ERC [161] (Exact Recovery Coefficient), et le signe [70], ou des propriétés topologiques [50, 56], permettent d'obtenir des bornes d'identifiabilité plus ou moins fines. Certains de ces critères sont par ailleurs stables au bruit.

Les sections suivantes présentent des méthodes permettant d'obtenir une solution $\boldsymbol{\alpha}$ parcimonieuse, au sens de ℓ^0 ou ℓ^1.

2.3.1 Matching Pursuit (MP)

Mallat et Zheng [110] ont introduit à la communauté du traitement du signal un algorithme glouton de poursuite très similaire à la *projection pursuit* [68] en statistiques, qu'ils appellent Matching Pursuit, pour approximer la décomposition idéale (P_1). Il s'agit d'un algorithme glouton qui, au lieu de chercher la solution globale recherche successivement la solution optimale d'un sous-problème, à savoir la meilleure approximation à un terme du signal.

On recherche une décomposition approximative du signal $\mathbf{x}$ sur un dictionnaire Φ, soit

$$\mathbf{s} = \sum_{i=0}^{m-1} \boldsymbol{\alpha}_i \boldsymbol{\phi}_i + R^{(m)},$$

où $R^{(m)}$ représente l'erreur résiduelle à l'étape m.
A partir d'une approximation initiale $\mathbf{x}^{(0)}$ de $\mathbf{x}$, et d'un résidu $R^{(0)} = \mathbf{x}$, on construit une famille d'approximations pas à pas. A chaque itération k, on recherche dans le dictionnaire l'atome d'indice γ_k qui a le plus grand produit scalaire avec le résidu $R^{(k-1)}$, et on ajoute sa contribution à l'estimé précédent $\mathbf{x}^{(k)}$. On obtient ainsi

$$\begin{aligned} & \mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \boldsymbol{\alpha}_k \boldsymbol{\phi}_{\gamma_k}, \\ \text{et} \quad & R^{(k)} = R^{(k-1)} - \boldsymbol{\alpha}_k \boldsymbol{\phi}_{\gamma_k}, \\ \text{avec} \quad & \boldsymbol{\alpha}_k = \langle R^{(k-1)}, \boldsymbol{\phi}_{\gamma_k} \rangle. \end{aligned}$$

Lorsque le dictionnaire est orthogonal, la méthode fonctionne parfaitement, et si le signal est construit avec seulement n composantes, l'algorithme s'arrête après n itérations. Cependant, bien que MP fonctionne généralement bien, il existe des exemples [32, 41] de signaux parcimonieux qui la mettent en défaut et lui font produire des vecteurs de représentation non creux.

2.3.2 Orthonormal Matching Pursuit (OMP)

Un raffinement de MP [122] consiste à rajouter une étape d'orthogonalisation pour obtenir à chaque itération m la meilleure approximation avec les atomes déjà choisis,

$$\min_{\{\boldsymbol{\alpha}_i\}, i \in [1,m]} \left\| \mathbf{x} - \sum_{i=1}^{m} \boldsymbol{\alpha}_i \boldsymbol{\phi}_{\gamma_i} \right\|_2^2,$$

en prenant en compte les m termes qui ont été trouvés par l'algorithme MP. On construit ainsi une solution pour laquelle le résidu est orthogonal à tous les vecteurs utilisés.

2.3.3 Basis Pursuit (BP)

Face à la difficulté que peuvent avoir les algorithmes gloutons de poursuite avec certains signaux ayant des composantes dans des bases différentes, pour lesquels l'algorithme perd beaucoup à corriger les erreurs dues aux approximations de ses premiers termes, Chen, Donoho et Saunders [33, 32] ont développé une méthode appelée Basis Pursuit qui résout le problème (P_1). Il s'agit d'un problème convexe, et la norme ℓ^1 n'étant pas différentiable en 0, elle favorise la parcimonie de la solution.
Soit $\lambda > 0$, on note $(P_{1,\lambda})$ le problème régularisé associé

$$(P_{1,\lambda}) : \quad \min_{\boldsymbol{\alpha}} \tfrac{1}{2} \|\mathbf{x} - \Phi\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1.$$

Ce problème est soluble par des éléments d'optimisation convexe non lisse. Soit $\Gamma_0(\mathcal{H})$ la classe des fonctions propres semi-continues inférieurement de l'espace séparable de Hilbert $\mathcal{H}$ vers $]?\infty, +\infty]$. Pour tout $f \in \Gamma_0(\mathcal{H})$ et $x \in \mathcal{H}$, la fonction $y \to f(y) + \frac{1}{2}\|y - x\|_2^2$ atteint son minimum en un point unique noté $\text{prox}_f(x)$, défini par Moreau [115]

$$\text{prox}_f(x) = \arg\min_{y \in \mathcal{H}} f(y) + \frac{1}{2}\|y - x\|_2^2.$$

Dans le cas plus général où l'on considère la somme de deux fonctions F_1 et F_2, avec F_2 à gradient κ-lipschitz, on peut utiliser le schéma de Douglas/Peaceman-Rachford [97], qui pour $\gamma > 0$ et $\mu_t \in (0, 2)$ définit la classe d'itérations

$$x^{(t+1)} = \left(1 - \frac{\mu_t}{2}\right) x^{(t)} + \frac{\mu_t}{2} \operatorname{rprox}_{\gamma F_1}\left(\operatorname{rprox}_{\gamma F_2}(x^{(t)})\right),$$

où $\operatorname{rprox}_{\mathrm{F}} = (2\operatorname{prox}_F - I)$. Ce schéma est stable au bruit, et converge dès lors que les erreurs sont de normes sommables [37]. Dans le cas considéré, $F_1 = \|\cdot\|_1$, d'opérateur proximal le seuillage doux, et $F_2 = \|y - \Phi x\|_2^2$.
Une autre approche est d'utiliser l'algorithme Block Coordinate Relaxation (BCR) proposé par Sardy *et al.* [136], qui s'applique lorsque le dictionnaire Φ est une concaténation de M bases orthonormées, $\Phi = [\Phi_1, \cdots, \Phi_M]$. Il met à jour la valeur estimée de $\boldsymbol{\alpha}$ en minimisant exactement la fonction objective du problème $(P_{1,\lambda})$ pour un bloc de coefficients, laissant les autres constants. Le coeur de l'algorithme repose sur la solution de seuillage doux dans le cas où les $\Phi_m, m = 1 \cdots M$ sont unitaire, c'est à dire

$$\tilde{\boldsymbol{\alpha}}_\lambda = \mathcal{ST}_\lambda(\Phi_m^T \mathbf{s}) \ , \quad \mathcal{ST}_\lambda(\mathbf{x}) = \frac{\mathbf{x}}{|\mathbf{x}|}(|\mathbf{x}| - \lambda)_+.$$

L'algorithme 1 suivant décrit les étapes du BCR.

Algorithm 1: Algorithme de BCR

Data: Un signal $\mathbf{s} \in L^2(\mathbb{R})$.
Input: Un dictionnaire $\Phi = [\Phi_1, \cdots, \Phi_M]$, $\forall m = 1 \cdots M, \Phi_m$ est une BON. Une approximation initiale $\boldsymbol{\alpha}$ de $\mathbf{s}$ dans Φ.
begin

1. Sélectionner un bloc $m \in [\![1, M]\!]$,
2. séparer $\boldsymbol{\alpha}$ en $\boldsymbol{\alpha}_m$ et $\boldsymbol{\alpha}_{-m}$ correspondant aux vecteurs de Φ_m et aux autres $(\Phi_{-m} = \{\Phi^i\}_{i \neq m})$,
3. calculer le résidu $\mathbf{v}_m = \mathbf{s} - \Phi_{-m}\boldsymbol{\alpha}_{-m}$,
4. améliorer $\boldsymbol{\alpha}_m$ en résolvant le problème marginal suivant par seuillage doux,

$$\boldsymbol{\alpha}_m = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{s} - \Phi_m\boldsymbol{\alpha}_m\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1,$$

5. boucler tant que le critère de convergence n'est pas atteint.

end
Result: Une décomposition parcimonieuse $\boldsymbol{\alpha}$ de f dans Φ.

Sous le critère sur la cohérence (2.8), BP comme OMP retrouvent la solution de (P_1), ainsi que sous d'autres critères d'identifiabilité tels que $\mathrm{ERC}(\Phi) > 0$ [161]. Des contraintes légèrement plus faibles sur la cohérence existent pour OMP [70] comme pour BP [51, 15, 73]. Une version plus générale de la cohérence, la fonction de Babel [160] μ_1 de Φ, donne aussi une condition suffisante pour que BP et OMP retrouvent la solution m-parcimonieuse de (P_1), à savoir $\mu_1(m-1) + \mu_1(m) < 1$, ou $\mu_1(m) < 1/2$. Finalement, dans le cas des matrices de mesure aléatoires gaussiennes, Tropp et Gilbert [162] ont montré que OMP peut retrouver un signal m-parcimonieux de longueur N avec seulement $\mathcal{O}(m \ln N)$ mesures.

2.3.4 Analyse en Composantes Morphologiques (MCA)

Alors que la plupart des méthodes précédentes permettent de décomposer un signal dans un dictionnaire quelconque, la méthode proposée par Starck *et al.* [151] d'Analyse en Composantes Morphologiques (MCA) propose de décomposer un signal dans un dictionnaire formé de la concaténation de K sous-dictionnaires à décomposition et reconstruction rapide comme ceux présentés dans ce chapitre.
Un problème intéressant en traitement du signal comme de l'image est la séparation de sources dans le cas où on n'observe que le mélange de toutes les sources simultanément. Dans le cas de K sources additionnées dans un signal de dimension N, on veut retrouver le signal associé à chacune de ces sources, soit $N \times K$ inconnues pour seulement N données. Cependant, sous certaines hypothèses de parcimonie des sources, et de décorrélation de celles-ci, on peut trouver la solution de ce problème.
Supposons que l'on observe un mélange linéaire bruité de K sources

$$\mathbf{x} = \sum_{k=1}^{K} \mathbf{x}_k + n, \quad n \sim \mathcal{N}(0, \sigma^2). \tag{2.10}$$

L'analyse en composantes morphologiques suppose que chaque source k soit parcimonieuse dans un dictionnaire Φ_k, et de plus que ces sources ne soient parcimonieuses dans aucun des autres dictionnaires $\Phi_{l \neq k}$. Cette dernière condition peut être interprétée comme une décohérence des dictionnaires.

$$\mathbf{x} = \sum_{k=1}^{K} \Phi_k \boldsymbol{\alpha}_k + n, \quad n \sim \mathcal{N}(0, \sigma^2). \tag{2.11}$$

L'algorithme de MCA (Algorithme 2) recherche la représentation la plus parcimonieuse du mélange dans le dictionnaire $\Phi = [\Phi_1, \cdots, \Phi_K]$, concaténation des sous-dictionnaires Φ_k, ce qui correspond à minimiser l'équation

$$\lambda \sum_{k=0}^{K} \|\boldsymbol{\alpha}_k\|_p^p + \frac{1}{2} \|\mathbf{x} - \sum_{k=0}^{K} \Phi_k \boldsymbol{\alpha}_k\|_2^2, \tag{2.12}$$

où la (pseudo) norme $\ell_p, 0 \leq p \leq 1$ force la parcimonie de la solution. L'idée derrière l'algorithme de MCA est d'utiliser le fait qu'en fixant $\boldsymbol{\alpha}_k, k \neq m$, une solution peut être trouvée appliquant un seuillage $\mathbf{r}_m = \mathbf{x} - \sum_{k \neq m} \Phi_k \boldsymbol{\alpha}_k$ dans Φ_m. Pour obtenir une séparation entre les différents dictionnaires, il ne faut pas garder tout le résidu dans Φ_m, car il contient initialement tout le signal. Il convient d'appliquer un seuil élevé, tel qu'uniquement des coefficients réellement associés à Φ_m soient gardés. Ceci peut être atteint en initialisant le seuil à $\max_k \|\Phi_k^T \mathbf{x}\|_\infty$, puis en le faisant descendre suffisamment lentement en bouclant sur les différents dictionnaires. Ainsi, si le seuil décroît suffisamment lentement et si les dictionnaires sont suffisamment décorrélés, à chaque itération chaque dictionnaire va récupérer un groupe de coefficients associés aux structures les plus fortes du résidu dans ce dictionnaire, qui ne correspondront à aucune structure dans les dictionnaires concurrents. Les garanties théoriques de la convergence de MCA en l'absence de bruit ont été étudiées avec la stratégie MOM (Mean of Max) dans [11, 13].

Algorithm 2: Analyse en Composantes Morphologiques

Data: Un signal observé g contaminée par un bruit de variance σ^2.
Input: Un dictionnaire $\Phi = [\Phi_1, \cdots, \Phi_K]$ composé de K sous-dictionnaires, un nombre d'itérations T_{iter} et un seuil final τ (e.g. 3).
begin
  Initialiser les composantes $f_k^{(0)} = 0, k = 1, \cdots, K$.
  Initialiser le résidu $r^{(0)} = g$.
  Initialiser le seuil : $k^\star = \arg\max_k \|\Phi_k^{\mathrm{T}} g\|_\infty$, soit $\lambda_0 = \max_{k \neq k^\star} \|\Phi_k^{\mathrm{T}} g\|_\infty$.
  for $t = 1$ **to** T_{iter} **do**
    for $k = 1$ **to** K **do**
      Calculer les résidus marginaux $r_k^{(t)} = r^{(t-1)} + f_k^{(t-1)}$.
      Mettre à jour (MAJ) les coefficients de la k^{ieme} composante en seuillant $\alpha_k^{(t)} = \text{Thresh}_{\lambda_{t-1}} \left(\Phi_k^{\mathrm{T}} r_k^{(t)} \right)$.
      MAJ la k^{ime} composante $f_k^{(t)} = \Phi_k \alpha_k^{(t)}$.
    MAJ la solution $f^{(t)} = \sum_{k=1}^K f_k^{(t)}$.
    MAJ le résidu $r^{(t)} = g - f^{(t)}$.
    MAJ le seuil $\lambda_t = \lambda_0 - t \left(\lambda_0 - \tau\sigma \right) / T_{\text{iter}}$.
end
Result: Les composantes f_k de f dans Φ_k

2.4 Ondelettes

Cette section présente les ondelettes, tout d'abord dans leur version continue unidimensionnelle, puis discrète avec les différentes constructions possibles et l'extension aux signaux bidimensionnels.

2.4.1 Ondelettes continues

D'après la définition de Morlet et Grossmann [74], une ondelette ψ est une fonction de $L^2(\mathbb{R})$ de moyenne nulle, normalisée à $\|\psi\|_{L_2(\mathbb{R})} = 1$ et décroissant suffisamment vite à l'infini, ou plus formellement vérifiant la condition d'admissibilité suivante

$$C_\psi = \int_0^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega < +\infty.$$

On définit la famille d'atomes temps-fréquence $\psi_{u,s}$ associés par dilatation d'un facteur s et translation de u

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}} \psi \left(\frac{t-u}{s} \right), \bar{\psi}_s(t) = \psi_{s,0}^*(-t)$$

qui étudie les structures de taille s au voisinage de u.
La transformée en ondelettes continues d'une fonction $f \in L^2(\mathbb{R})$ est

$$\mathcal{W}f(u,s) = \int_{\mathbb{R}} f(t) \psi_{u,s}^*(t) dt = f * \bar{\psi}_s(u), \tag{2.13}$$

et la reconstruction de f à partir de ses coefficients $\mathcal{W}f$ se fait par

$$f(t) = \frac{1}{C_\psi} \int_0^{+\infty} \int_{-\infty}^{+\infty} \mathcal{W}f(u,s)\psi_{u,s}(t)du\frac{ds}{s^2}.$$

Si on ne calcule $\mathcal{W}f$ que jusqu'à une échelle limite s_0, on a besoin d'ajouter l'information contenue dans les échelles $s > s_0$ pour pouvoir reconstruire f. On construit donc une fonction ϕ de phase arbitraire et qui prendra en compte l'énergie perdue pour $s > s_0$. Pour cela, on définit une fonction d'échelle qui agrège les ondelettes d'échelle supérieure à 1 :

$$|\hat{\phi}(\omega)|^2 = \int_1^{+\infty} |\hat{\psi}(s\omega)|^2 \frac{ds}{s} = \int_\omega^{+\infty} \frac{|\hat{\psi}(\xi)|^2}{\xi} d\xi.$$

On définit donc la famille de fonctions d'échelles par

$$\phi_{u,s}(t) = \frac{1}{\sqrt{s}}\phi\left(\frac{t-u}{s}\right), \bar{\phi}_s(t) = \phi^*_{s,0}(-t)$$

et l'approximation basses fréquences de f est

$$Lf(u,s) = f * \bar{\phi}_s(u).$$

Ainsi on peut reconstruire f exactement en utilisant les coefficients d'ondelettes conjointement aux coefficients d'approximation basse fréquence suivants

$$\begin{aligned} f(t) &= \frac{1}{C_\psi} \int_0^{s_0} \int_{-\infty}^{+\infty} \mathcal{W}f(u,s)\psi_{u,s}(t)du\frac{ds}{s^2} + \frac{1}{C_{\psi s_0}} \int_{-\infty}^{+\infty} Lf(u,s_0)\phi_{u,s_0}(t)du \\ &= \frac{1}{C_\psi} \int_0^{s_0} \mathcal{W}f(\cdot,s) * \psi_s(t)\frac{ds}{s^2} + \frac{1}{C_\psi s_0} Lf(\cdot,s_0) * \phi_{s_0}(t). \end{aligned} \tag{2.14}$$

Un exemple de décomposition en ondelettes continues d'un signal 1D est montré en figure 2.3.

2.4.2 Exemples d'ondelettes classiques

Ondelette de Haar

L'approximation multi-échelle de Haar est constante par morceaux et correspond à la fonction d'échelle $\phi = \mathbf{1}_{[0,1]}$ et d'ondelette

$$\psi(t) = \begin{cases} -1 & \text{pour } 0 \leq t < 1/2, \\ 1 & \text{pour } 1/2 \leq t < 1, \\ 0 & \text{ailleurs.} \end{cases}$$

C'est l'ondelette orthogonale au support le plus petit, mais elle n'a qu'un seul moment nul, la moyenne, le moment d'ordre $k \in \mathbb{N}$ (non centré) étant défini par

$$\text{moment}_k(\psi) = \int_{\mathbb{R}} t^k \psi(t)dt.$$

Elle est peu utilisée en tant qu'ondelette car elle crée de forts artéfacts en marches d'escalier, surtout dans le cas orthogonal, mais est très utile dans de nombreuses applications de par son lien étroit avec le gradient.

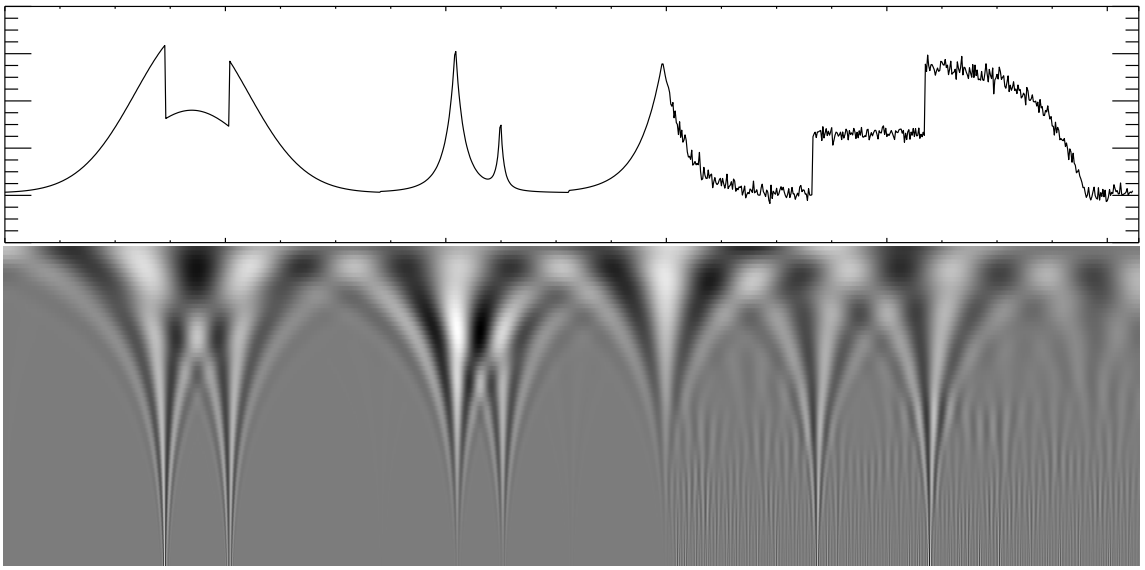


FIG. 2.3 – Scalogramme : décomposition en ondelettes continues d'un signal 1D. Sur l'image du bas, l'échelle va de la plus fine en bas de l'image, aux grandes structures du signal pour la dernière ligne. Le paramètre d'échelle varie continûment, par opposition aux ondelettes discrètes classiques présentées plus loin, pour qui l'échelle varie généralement de manière dyadique.

Ondelette de Shannon

L'ondelette de Shannon correspond à une fonction d'échelle qui est un passe-bas parfait : $\hat{\phi} = \mathbf{1}_{[-\pi,\pi]}$. L'ondelette associée est

$$\psi(t) = \mathrm{sinc}\,(2\pi(t-1/2)) - \mathrm{sinc}\,(\pi(t-1/2))\,.$$

Il s'agit de l'ondelette sinus cardinal, qui est à support compact parfait (passe-bande idéal) en Fourier mais oscille trop dans le domaine original, ce qui a conduit à la construction de l'ondelette de Meyer suivante, qui est à support compact en Fourier mais avec une décroissance douce.

Ondelette de Meyer

On définit l'ondelette *mère* de Meyer [91] dans le domaine fréquentiel par :

$$\hat{\psi}(\xi) = \begin{cases} \exp^{-i\pi\xi}\sin(\frac{\pi}{2}\nu(3|\xi|-1)), & \text{si } 1/3 < |\xi| \leq 2/3 \\ \exp^{-i\pi\xi}\cos(\frac{\pi}{2}\nu(\frac{3}{2}|\xi|-1)), & \text{si } 2/3 < |\xi| \leq 4/3 \\ 0 & \text{ailleurs.} \end{cases}$$

où ν est une fonction lisse allant de 0 à 1 sur $[0,1]$, telle que :

$$\nu(x) = \begin{cases} 1-\nu(1-x) & \text{si } x \in [0,1] \\ 0 & \text{si } x \leq 0 \\ 1 & \text{si } x \geq 1. \end{cases}$$

La fonction d'échelle associée (ondelette *père*) est définie en Fourier en utilisant la même

fonction ν par :

$$\hat{\phi}(\xi) = \begin{cases} 1, & \text{if } |\xi| \leq 1/3 \\ \cos(\frac{\pi}{2}\nu(3|\xi|-1)), & \text{if } 1/3 < |\xi| \leq 2/3 \\ 0 & \text{ailleurs} \end{cases}$$

Ces ondelettes sont C^∞ car leurs transformées de Fourier sont à support compact, mais ont une décroissance en $1/(1+M)^{n+1}$ si le filtre associé est C^n, ce qui limite les rebonds dans le domaine original.

Ondelette de Battle-Lemarié

Elles sont construites sur l'approximation multi-résolution par des fonctions splines. La fonction d'échelle et son filtre associé s'écrivent en Fourier

$$\hat{\phi}(\omega) = \frac{\exp(-i\epsilon\omega/2)}{\omega^{m+1}\sqrt{S_{2m+2}(\omega)}}, \qquad \epsilon = \begin{cases} 1 & \text{si } m \in 2\mathbb{N}, \\ 0 & \text{sinon} \end{cases},$$

$$S_m(\omega) = \sum_{k\in\mathbb{Z}} \frac{1}{(\omega+2k\pi)^m},$$

et l'ondelette associée est

$$\hat{\psi}(\omega) = \frac{\exp(-i\omega/2)}{\omega^{m+1}} \sqrt{\frac{S_{2m+2}(\omega/2+\pi)}{S_{2m+2}(\omega)S_{2m+2}(\omega/2)}}.$$

Ces ondelettes ont une décroissance exponentielle, mais sont moins régulières que les ondelettes de Meyer. La figure 2.4 montre une ondelettes de Battle-Lemarié ayant six moments nuls.

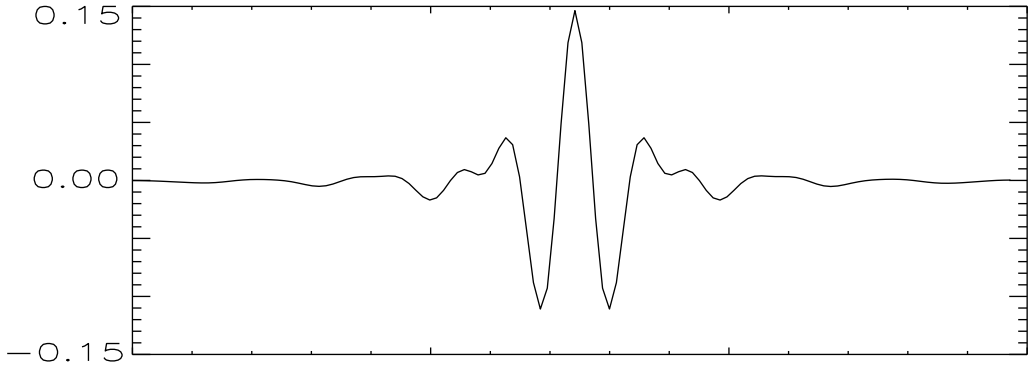


FIG. 2.4 – Ondelette de Battle-Lemarié à six moments nuls.

2.4.3 Ondelettes discrètes

Comme le montrent les équations (2.13) et (2.14), le transformée en ondelettes et sa reconstruction peuvent être effectuées par des convolutions et sommations. On note $f[n], n \in \{0, \cdots, N-1\}$ le signal discret associé au signal continu f défini sur $[0,1]$ dilaté d'un facteur N, échantillonné avec un pas unitaire : $f[n] = f^{\text{dilate}}(n)$. Les échelles considérées sont $s = a^j$, donc séparées par un facteur de dilatation constant usuellement $a = 2^{1/\nu}$, ce qui permet d'avoir ν échelles par octave.
Soit ψ une ondelette à support dans $[-K/2, K/2]$. L'ondelette discrète dilatée ψ_j est définie

par

$$\psi_j[n] = \frac{\sqrt{C_\psi}}{\sqrt{a^j}} \psi\left(\frac{n}{a^j}\right).$$

Pour une implémentation rapide, la transformée en ondelettes discrètes se fait en utilisant la FFT (transformée de Fourier rapide), qui implique des conditions de bord périodiques et qui permet une implémentation par simple convolution par le filtre d'ondelette

$$\mathcal{W}f[n, a^j] = \sum_{m=0}^{N-1} f[m]\psi_j^*[m-n] = f \circledast \overline{\psi}_j[n].$$

Pour les échelles supérieures à a^J, on convolue par la fonction d'échelle échantillonnée

$$\phi_J[n] = \sqrt{\frac{C_\psi}{\ln a}} \frac{1}{\sqrt{a^J}} \phi\left(\frac{n}{a^J}\right),$$

$$Lf[n, a^J] = f \circledast \overline{\phi}_J[n].$$

En notant a^I l'échelle la plus fine, la reconstruction se fait donc par discrétisation de l'équation (2.14), soit

$$f[n] \approx \sum_{j=I}^{J} \mathcal{W}f[\cdot, a^j] \circledast \psi_j[n] + Lf[\cdot, a^J] \circledast \phi_J[n],$$

l'approximation [107] étant d'autant meilleure que le nombre d'échelles par octave augmente.

2.4.4 Bases orthogonales d'ondelettes et bancs de filtres CMF

L'analyse multi-résolution, introduite par Mallat et Meyer [104, 105], considère une séquence de sous-espaces fermés imbriqués $\{V_j\}_{j\in\mathbb{Z}}$ de $L_2(\mathbb{R})$

$$L_2(\mathbb{R}) \supset \cdots \supset V_{j-1} \supset V_j \supset \cdots \supset \{0\},$$

vérifiant l'autosimilarité en temps et en échelle

$$\forall (j,k) \in \mathbb{Z}^2 \ , \ f(t) \in V_j \Leftrightarrow f(t - 2^j k) \in V_j,$$

$$\forall j \in \mathbb{Z} \ , \ f(t) \in V_j \Leftrightarrow f(t/2) \in V_{j+1},$$

ainsi que les conditions aux limites suivantes

$$\lim_{j\to\infty} V_j = \bigcap_{j\in\mathbb{Z}} V_j = \{0\},$$

$$\lim_{j\to-\infty} V_j = \text{Adhérence}\left(\bigcup_{j\in\mathbb{Z}} V_j\right) = L^2(\mathbb{R}),$$

et finalement l'existence d'une fonction $\theta \in L_2(\mathbb{R})$ telle que $\{\theta(t-n)\}_{\in\mathbb{Z}}$ soit une base de Riesz [36] de V_0. L'approximation à l'échelle 2^j d'une fonction f est alors sa projection sur V_j.
A la base de Riesz $\{\theta(t-n)\}_{\in\mathbb{Z}}$ on peut associer une fonction ϕ appelée fonction d'échelle,

qui génère la famille $\{\phi_{j,n}\}_{n\in\mathbb{Z}}$ suivante, qui est une base orthonormée (BON) de V_j,

$$\phi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\phi\left(\frac{t-n}{2^j}\right). \tag{2.15}$$

La famille $\{\phi_{j,n}\}_{n\in\mathbb{Z}}$ étant une BON de V_j, on peut effectuer la décomposition

$$\frac{1}{2}\phi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} h[n]\phi(t-n), \tag{2.16}$$

avec

$$h[n] = \left\langle \frac{1}{2}\phi\left(\frac{t}{2}\right), \phi(t-n)\right\rangle, \tag{2.17}$$

soit en Fourier

$$\hat{h}(\omega) = \frac{\hat{\phi}(2\omega)}{\hat{\phi}(\omega)}. \tag{2.18}$$

Le filtre ainsi défini est un filtre miroir conjugué (CMF), c'est à dire qu'il vérifie la condition de quadrature

$$|\hat{h}(\omega)|^2 + |\hat{h}(\omega+\pi)|^2 = 1, \tag{2.19}$$

et $\hat{h}(0) = 1$.
On définit l'espace de détail W_j comme le complémentaire de V_j dans V_{j-1} : $V_{j-1} = V_J \oplus W_j$. Soit ψ la fonction dont la transformée de Fourier vérifie

$$\hat{\psi}(\omega) = \frac{1}{\sqrt{2}}\hat{g}\left(\frac{\omega}{2}\right)\hat{\phi}\left(\frac{\omega}{2}\right), \tag{2.20}$$

avec

$$\hat{g}(\omega) = e^{-\imath\omega}\hat{h}^*(\omega+\pi), \tag{2.21}$$

ou encore

$$\hat{g}(\omega) = \frac{\hat{\psi}(2\omega)}{\hat{\phi}(\omega)}\ , \qquad g[n] = \frac{1}{2}\left\langle \psi\left(\frac{t}{2}\right), \phi(t-n)\right\rangle. \tag{2.22}$$

La famille $\{\psi_{j,n}\}_{n\in\mathbb{Z}}$ à j fixé forme une BON de W_j, et par conséquent sur l'ensemble des échelles, $\{\psi_{j,n}\}_{(j,n)\in\mathbb{Z}^2}$ forme une base de $L_2(\mathbb{R})$.

2.4.5 Transformée en ondelettes orthogonales rapide, FWT

La transformée en ondelettes rapide, à facteur de dilatation dyadique se calcule en utilisant le banc de filtres (h, g) associé aux fonctions d'échelle et d'ondelette définis précédemment. On veut calculer une approximation de f à l'échelle $j+1$ en le projetant sur le sous-espace d'approximation V_j engendré par les fonctions d'échelle $\phi_{j+1,p}$, à partir du filtre h associé à ϕ vérifiant (2.18) :

$$\begin{aligned}
a_{j+1}[p] &= \langle f, \phi_{j+1,p}\rangle && (2.23)\\
&= \left\langle f, \sum_n \langle\phi_{j+1,p}, \phi_{j,n}\rangle\, \phi_{j,n}\right\rangle = \left\langle f, \sum_n h[n-2p]\phi_{j,n}\right\rangle && (2.24)\\
&= \sum_n h[n-2p]\,\langle f, \phi_{j,n}\rangle = \sum_n h[n-2p]a_j[n] && (2.25)\\
&= a_j * \overline{h}[2p]. && (2.26)
\end{aligned}$$

De même la projection sur le sous-espace de détail W_j muni de la BON $\{\psi_{j+1,p}\}_p$ en utilisant le filtre g vérifiant (2.22) se calcule par

$$d_{j+1}[p] = a_j * \overline{g}[2p]. \tag{2.27}$$

La reconstruction s'effectue par :

$$\begin{aligned} a_j[p] &= \sum_n h[p-2n]a_{j+1}[n] + g[p-2n]d_{j+1}[n] && (2.28)\\ &= \check{a}_{j+1} * h[p] + \check{d}_{j+1} * g[p], && (2.29) \end{aligned}$$

en notant $\check{a}$ le signal discret a dans lequel on a inséré des zéros entre chaque valeur :

$$\check{a}[n] = \begin{cases} a[p] & \text{si } n = 2p, \\ 0 & \text{si } n = 2p+1. \end{cases} \tag{2.30}$$

2.4.6 Bancs de filtres

L'implémentation de la FWT à l'aide des formules ci-dessus (2.26, 2.27, 2.29) correspond à l'algorithme pyramidal de Mallat [104], et se fait en utilisant des bancs de filtres sous-échantillonnés, c'est à dire par convolution avec un filtre passe-bas $\overline{h}$ et un filtre passe-haut $\overline{g}$ puis en décimant d'un facteur deux. Le schéma en figure 2.5 montre le processus de filtrage symétrique, pour la transformation et la reconstruction, qui peut être appliqué plusieurs fois de suite pour décomposer sur un plus grand nombre d'échelles, dans le cas plus général bi-orthogonal présenté dans la section suivante. La figure 2.6 montre une décomposition en ondelettes décimées à quatre échelles d'un signal unidimensionnel.

2.4.7 Ondelettes biorthogonales

Une famille $(u_k, \tilde{u}_k)$ est dite biorthogonale si u_k et $\tilde{u}_k$ sont des bases de Riesz [36, 35] de deux sous espaces V et $\tilde{V}$ inclus dans un espace de Hilbert et si $\forall (i,j) \langle u_i, \tilde{u}_j \rangle = \delta_{ij}$. Ainsi,

$$\forall f \in \tilde{V} \quad \text{on a} \quad f = \sum_j \langle f, u_j \rangle \tilde{u}_j,$$

et

$$\forall f \in \tilde{V} \quad \text{on a} \quad f = \sum_j \langle f, \tilde{u}_j \rangle u_j.$$

Bancs de filtres

De même que lorsque l'on travaille avec des *filtres miroirs conjugués* (cf. section 2.4.4), on peut décomposer un signal a_j et le reconstruire à l'aide des filtres g, h et de leurs duaux $\tilde{g}, \tilde{h}$

$$\begin{aligned} a_{j+1}[n] &= a_j * \overline{h}[2n], \\ d_{j+1}[n] &= a_j * \overline{g}[2n], \\ a_j[n] &= \check{a}_{j+1} * \tilde{h}[n] + d_{j+1} * \tilde{g}[n], \end{aligned}$$

où le banc de filtres $(h, g, \tilde{h}, \tilde{g})$ satisfait les relations de reconstruction exacte si et seulement si

$$\begin{cases} \hat{h}^*(\omega+\pi)\hat{\tilde{h}}(\omega) + \hat{g}^*(\omega+\pi)\hat{\tilde{g}}(\omega) = 0 \\ \hat{h}^*(\omega)\hat{\tilde{h}}(\omega) + \hat{g}^*(\omega)\hat{\tilde{g}}(\omega) = 1. \end{cases}$$

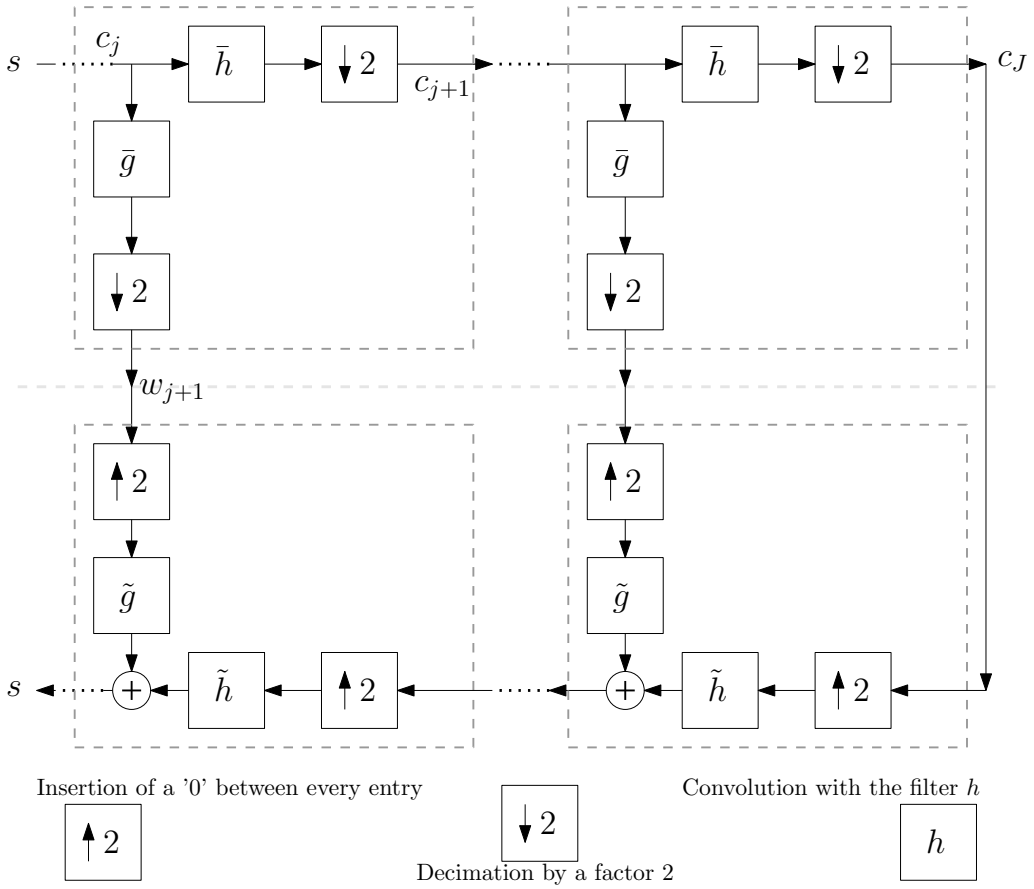


FIG. 2.5 – Schéma montrant l'algorithme pyramidal de Mallat pour calculer la transformée en ondelettes orthogonale unidimensionnelle. Le signal c_j est séparé en deux composantes avec les filtres $\bar{g}$ et $\bar{h}$, puis décimé. On obtient ainsi les coefficients d'échelle fine w_{j+1} et l'approximation d'échelle suivante c_{j+1}. Le bloc en dessous effectue le processus inverse, en insérant des zéros entre les échantillons et en appliquant les filtres de reconstruction associés. Le processus peut être répété plusieurs fois sur la sortie c_{j+k} pour obtenir une décomposition avec plus d'échelles w_{j+k} et une approximation finale c_J. La reconstruction se fait suivant la partie basse du schéma, en utilisant les filtres $\tilde{h}$ et $\tilde{g}$.

Dans ce cas, on appelle $(h, \tilde{h})$ des *filtres à reconstruction parfaite*. Si en plus $h = \tilde{h}$, ce sont des *filtres miroirs conjugués*.

2.4.8 Ondelettes 2-D

Les différents types de transformée en ondelettes présentés précédemment sont extensibles en dimension quelconque en les appliquant de manière séquentielle et séparable sur les différentes dimensions du signal. En deux dimensions, on convolue les lignes de l'image a_j par $\overline{h}$ et $\overline{g}$, puis on sous-échantillonne les images obtenues. On effectue alors ces deux mêmes opérations sur les colonnes des imagettes obtenues et on sous-échantillonne à nouveau. On obtient ainsi une transformée $(a_{j+1}, d^1_{j+1}, d^2_{j+1}, d^3_{j+1})$, où les des sous-bandes de détails d^1, d^2 et d^3 correspondent respectivement aux orientations horizontale (haute fréquence en y, basse en x), verticale (le contraire), et diagonale (hautes fréquences en x et y). En suivant les notations de la section 2.4.5, et en notant $p = (p_1, p_2) \in \mathbb{N}^2$ et $xy[p] = x[p_1]y[p_2]$, la

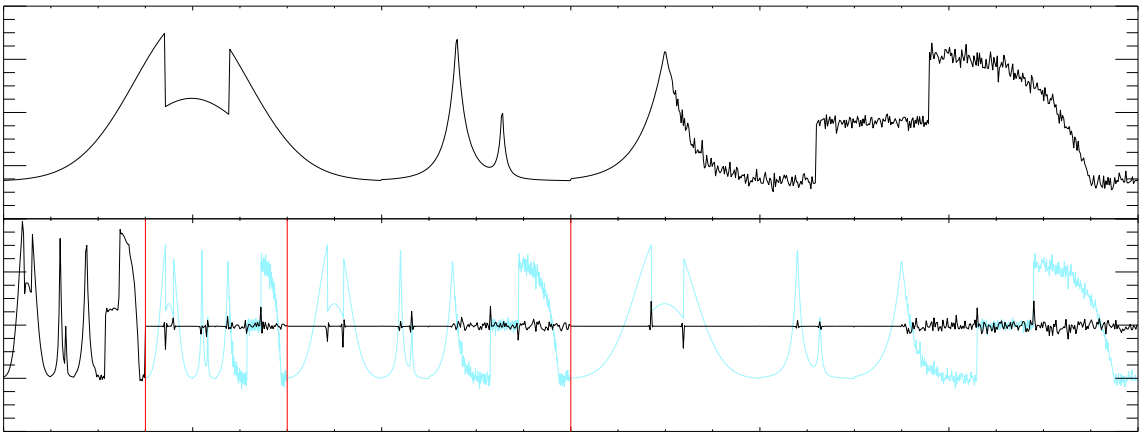


FIG. 2.6 – Haut : Signal lisse par morceaux contaminé par un bruit additif gaussien sur sa dernière moitié. Bas : sa décomposition en ondelettes orthogonales. Il y a autant de coefficients que d'échantillons dans le signal, et chaque échelle fait la moitié de la taille de la suivante. Le signal original compressé est affiché en clair derrière les coefficients pour les visualiser par rapport aux singularités.

transformée s'effectue par filtrage et décimation,

$$\begin{aligned} a_{j+1}[p] &= a_j * \overline{hh}[2p] \\ d^1_{j+1}[p] &= a_j * \overline{h}\overline{g}[2p] \\ d^2_{j+1}[p] &= a_j * \overline{g}\overline{h}[2p] \\ d^3_{j+1}[p] &= a_j * \overline{gg}[2p]. \end{aligned}$$

La reconstruction s'obtient par

$$a_j[p] = \check{a}_{j+1} * hh[p] + \check{d}^1_{j+1} * hg[p] + \check{d}^2_{j+1} * gh[p] + \check{d}^3_{j+1} * gg[p],$$

$\check{a}$ étant défini en (2.30). La figure 2.7 montre la transformée en ondelettes à six échelles de l'image *lena*.

2.4.9 Ondelettes non décimées

Un des défauts des bases ondelettes orthogonales séparables est leur non invariance par translation. Si on dispose de l'espace mémoire suffisant, et de temps pour le calcul, on peut ne pas décimer les différentes échelles et ainsi obtenir une transformée invariante par translation. Le problème est que contrairement aux ondelettes décimées, dont le nombre de coefficients est identique au nombre d'échantillons, dans le cas non décimé on obtient de la redondance. Pour un signal unidimensionnel, la redondance sera égale au nombre d'échelles de décomposition plus une pour l'approximation basse résolution. Une façon simple d'effectuer une transformée sur un signal de dimension supérieure est de considérer une transformation séparable, donc d'appliquer la transformée 1D de manière séquentielle sur chacune des dimensions. Ainsi, en deux dimensions nous obtenons une redondance de $3J+1$, J étant le nombre de décompositions. En dimension d quelconque la redondance des ondelettes séparables non décimées pour une décomposition sur J échelles est $J(2^d-1)+1$. Il existe plusieurs méthodes pour réaliser une transformée non décimée. On peut effectuer plusieurs transformées décimées, après avoir appliqué un décalage au signal, ou on peut

FIG. 2.7 – Image *lena* et sa transformée en ondelettes orthogonales sur six échelles : cinq échelles fines à trois bandes par échelle ($g_x h_y$, $h_x g_y$ et $h_x h_y$), et l'approximation résultante. A noter que la transformée est montrée en échelle logarithmique pour plus de visibilité des faibles coefficients.

utiliser des filtres modifiés par insertion de zéros, c'est l'idée de l'algorithme à trous.

L'algorithme *à trous*

Proposé par Holdschneider *et al.* [83], cet algorithme utilise un filtre différent à chaque échelle. En effet, on définit pour tout filtre $h[n]$ le filtre $h_j[n]$ obtenu en insérant entre chaque entrée de h $2^j - 1$ zéros. On peut alors utiliser le schéma classique de décomposition représenté en figure 2.5, dans lequel les filtres de chaque bloc j sont changés par h_j, et où les décimations ont été supprimées. La reconstruction se fait en moyennant les reconstructions des différents filtres. L'algorithme à trous est visible en figure 2.8.

Ondelettes Isotropes

Les propriétés voulues pour construire ces ondelettes sont que le filtre h soit isotrope (i.e. $\hat{h}^*(\omega) = \hat{h}(-\omega) = \hat{h}(\omega)$), que le banc de filtres $(h, g, \tilde{h}, \tilde{g})$ satisfasse la condition de reconstruction exacte, ici donnée en Fourier

$$\hat{h}^* \hat{\tilde{h}} + \hat{g}^* \hat{\tilde{g}} = 1,$$

et finalement que la reconstruction se fasse par sommation des sous-bandes, c'est à dire $\hat{\tilde{g}} = \hat{\tilde{h}} = 1$, ce qui mène à

$$\hat{g}^* = 1 - \hat{h}, \tag{2.31}$$

d'où

$$\frac{1}{2}\psi(x/2) = \boldsymbol{\phi}(x/2) + \frac{1}{2}\boldsymbol{\phi}(x).$$

De la définition de $\hat{g}$ (eqn. 2.31), on remarque que les coefficients d'ondelettes sont obtenus par simple différence de deux fonctions d'échelles consécutives, et la reconstruction se fait par

$$a_0 = a_J + \sum_{j=1}^{J} d_j.$$

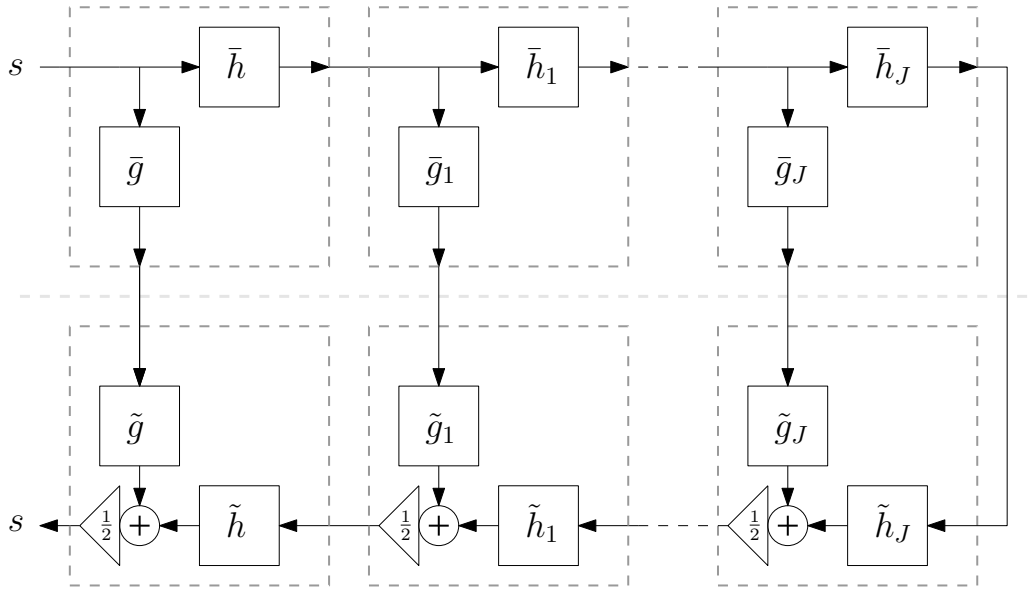


FIG. 2.8 – Schéma montrant la cascade de filtrages à appliquer pour une transformée en ondelettes unidimensionnelle non décimée en utilisant l'algorithme à trous. Pour un filtre x, on définit $x_j[n]$ par insertion de $2^j - 1$ zéros entre chaque entrée de $x[n]$.

Cette construction d'ondelettes par différence de fonctions d'échelles permet de créer des ondelettes bidimensionnelles isotropes, ayant une redondance égale au nombre d'échelles de décomposition plus une pour l'approximation. Utilisée avec une fonction d'échelle B-spline d'ordre 3, elle porte le nom de Starlet [154, 148] du fait de son adéquation particulière aux objets astronomiques. La figure 2.9 montre une transformée en ondelettes 2-D isotropes avec cet algorithme.

2.5 Quelques constructions multi-échelles 2-D non adaptatives

Dans le cas des signaux 1D, Mallat *et al.* [109] ont montré que les ondelettes sont optimales pour représenter les signaux lisses par morceaux ayant un nombre fini de discontinuités. En effet, soit f une fonction C^α par morceaux, l'erreur d'approximation non linéaire à M termes ε_M en projetant sur une famille d'ondelettes ayant $q > \alpha$ moments nuls est définie par

$$\varepsilon_M = \|f - f_M\|_2^2, \quad f_M = \sum_{\mathcal{I}_M} \boldsymbol{\psi}_i \langle f, \boldsymbol{\psi}_i \rangle,$$

où $\mathcal{I}_M$ représente l'ensemble des M indices dont les coefficients $\langle f, \boldsymbol{\psi}_i \rangle$ sont les plus grands dans la base d'ondelettes $\boldsymbol{\psi}_i$. On peut montrer que cette erreur vérifie

$$\varepsilon_M = \mathcal{O}\left(CM^{-2\alpha}\right).$$

Si de plus $f \in BV([0,1])$, espace des fonctions à variations bornées sur $[0,1]$, alors

$$\varepsilon_M = \mathcal{O}\left(C\|f\|_{BV}^2 M^{-2}\right).$$

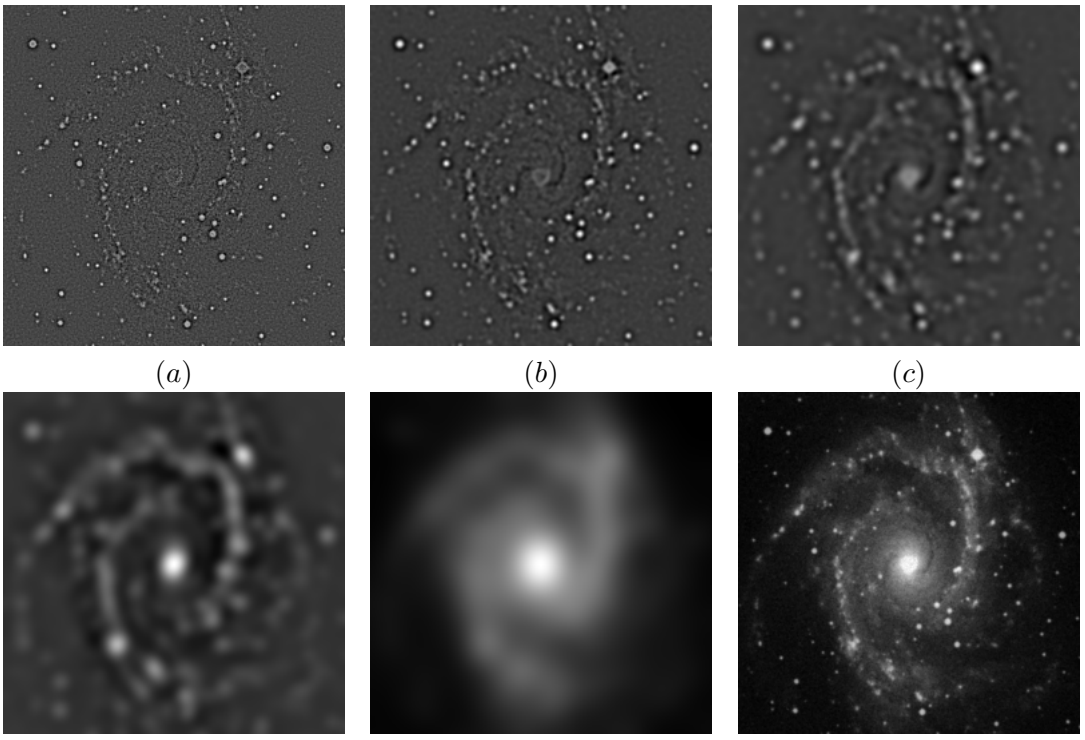


FIG. 2.9 – Sous-bandes de la transformée en ondelettes isotropes à trois échelles d'une image astronomique de la galaxie NGC2997. (a)-(d) sont les sous-bandes de détails d'échelle croissante, (e) est l'approximation, et (f) l'image originale, ou reconstruction par sommation des 5 sous-bandes précédentes.

Cette vitesse de décroissance est optimale dans le sens où elle ne peut pas être améliorée par toute autre approximation non linéaire dans une base orthonormée. Les ondelettes sont donc la BON optimale pour la compression des signaux C^∞ par morceaux, et des signaux à variation bornée.
Les ondelettes sont donc parfaitement adaptées aux signaux 1D, mais quand on passe en deux dimensions, les discontinuités selon des courbes ne sont plus bien représentées par les ondelettes, qui sont principalement isotropes, les ondelettes séparables créent des motifs en damier qui manque de directionnalité. De nombreux travaux ont alors porté sur la construction de BON ou de trames ayant une meilleure sélectivité directionnelle. Les parties suivantes présentent brièvement quelques unes des transformées 2-D non adaptatives qui sont apparues principalement suite au manque de directionnalité des ondelettes classiques, à savoir les ondelettes complexes en arbre dual, les contourlets et les curvelets rapides.

2.5.1 Ondelettes complexes en arbre dual

Afin de palier entre autres au manque de directionnalité des ondelettes décimées, ainsi qu'à leur non invariance par translation, Kingsbury [90, 138] a proposé une construction en ondelettes complexes en arbre dual. Dans le cas unidimensionnel, ces ondelettes complexes $\psi_c^{j,n}(t) = 2^{-j/2}\psi_c(2^{-j}t - n)$, $\psi_c = \psi_r + \imath\psi_i$, avec $\imath = \sqrt{-1}$, ont été construites non pas

en forçant la famille ψ_c à être une base orthogonale, mais ψ_r, ψ_i à être deux bases (bi-)orthogonales au prix d'un facteur 2 en redondance (et 2^d pour l'extension en dimension d).
On note g_r et h_r les filtres passe-haut et passe-bas de la première transformée et g_i et h_i ceux de la deuxième. On peut construire ces quatre filtres vérifiant pour chaque transformée la propriété de reconstruction exacte, et de façon à ce que la transformée globale soit approximativement analytique : on associe l'ondelette ψ_r aux filtres g_r et h_r, et ψ_i à g_i et h_i. On peut construire les filtres de sorte que $\psi_c = \psi_r + \boldsymbol{\imath}\psi_i$ soit approximativement analytique, ce qui correspond à définir ψ_i comme la transformée de Hilbert de ψ_r, et tous ces filtres sont à valeurs réelles. Le banc de filtre correspondant à la transformée est schématisé en figure 2.10. Les coefficients de la transformée complexe sont obtenus en considérant les coefficients des transformées utilisant (g_r, h_r) et (g_i, h_i) comme respectivement les parties réelles et imaginaires. A noter que pour obtenir des ondelettes analytiques avec une bonne approximation, il est nécessaire de modifier légèrement les filtres et le schéma donné en figure 2.10. La reconstruction se fait par les bancs de filtres inverses classiques, et les deux reconstructions sont moyennées, ou on peut répartir le facteur 0.5 en $1/\sqrt{2}$ sur la transformée et la reconstruction pour plus de symétrie. La figure 2.12 montre les parties réelles, imaginaires et les modules des six atomes des ondelettes complexes en arbre dual à une échelle donnée.
Appliquées de manière séparable sur des images ou sur des signaux de dimension su-

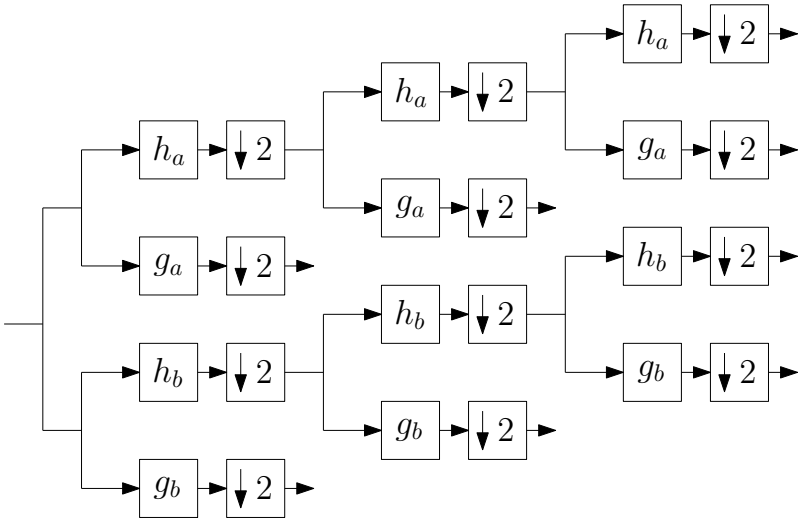


FIG. 2.10 – Schéma montrant l'algorithme d'analyse d'une transformée en ondelettes complexes en arbre dual. h_r et h_i sont les filtres passe-bas, et g_r et g_i les filtres passe-haut.

périeure, ces ondelettes exhibent une séparation angulaire bien supérieure aux ondelettes classiques. En effet, comme le montre la figure 2.11, le fait que ces ondelettes soient analytiques permet d'éviter le schéma en damier au profit de plusieurs ondelettes directionnelles séparées de $\pi/6$. La figure 2.12 montre les parties réelles, imaginaires et les modules des six atomes des ondelettes complexes en arbre dual à une échelle donnée.

2.5.2 Transformée en Ridgelets

Une transformation bien connue en tomographie est la transformée de Radon. C'est l'opération qui est appliquée lorsque l'on fait de la tomographie par transmission, souvent en imagerie médicale, par rayons X par exemple. L'image obtenue est l'intégration le long

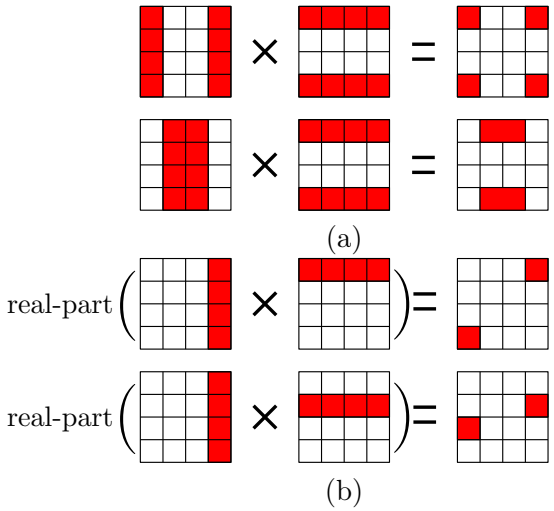


FIG. 2.11 – Schémas idéalisés du support fréquentiel des ondelettes séparables. (a) deux des trois directions dans le cas des ondelettes classiques, et (b) deux des six directions des ondelettes complexes en arbre dual. On constate que la forme analytique des ondelettes complexes en arbre dual permet d'obtenir une séparation directionnelle, et d'éviter le damier créé par la première ligne de (a).

de la ligne de visée de la transmission du matériau.
La transformée de Radon, décrite plus précisément dans la sous-section suivante, transforme les singularités le long des lignes en singularités ponctuelles et est par conséquent bien adaptée pour représenter des lignes dans une image. Mais il lui manque l'aspect multi-échelle que peuvent avoir les discontinuités le long des lignes. L'idée est donc d'appliquer à la suite de cette transformée une transformée en ondelettes, ce qui constitue la transformée en ridgelets, décrite par la suite.

Transformée de Radon

Formellement, la transformée de Radon en deux dimensions est une transformation du plan (x_1, x_2) vers le plan (t, θ), par intégration selon toutes les lignes, indexées par leur angle θ et leur décalage par rapport à l'origine. La transformée de Radon $\mathbf{Rad}\, f$ d'une

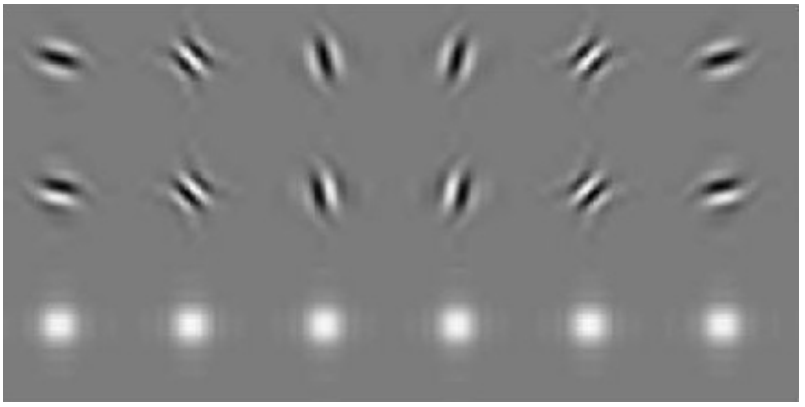


FIG. 2.12 – Exemples d'ondelettes complexes en arbre dual 2-D. De haut en bas, la partie réelle, la partie imaginaire, et le module des six ondelettes d'une échelle donnée.

fonction f est définie pour $(t, \theta) \in \mathbb{R} \times [0, \pi)$ par

$$\mathbf{Rad} f(t, \theta) = \int_{\mathbb{R}^2} f(x_1, x_2) \delta(t - x_1 \cos\theta - x_2 \sin\theta) dx_1 dx_2.$$

Le théorème de projection en Fourier en deux dimensions stipule que la transformée de Fourier de la projection d'une fonction 2-D sur une ligne est égale à une coupe de la transformée de Fourier 2-D de cette fonction passant par l'origine, et parallèle à la ligne sur laquelle la projection a été faite. Cette propriété étant donnée en coordonnées polaires, il y a des problèmes lors du passage au monde discret, et il existe plusieurs implémentations différentes de cette transformée.

En utilisant le théorème de projection pour effectuer la transformée de Radon, on peut construire $\mathbf{Rad} I$ d'une image discrète I de taille $n \times n$ en effectuant une Transformée de Fourrier 2-D de l'image, puis en appliquant à ses coupes radiales des $\mathcal{TF}^{-1}$ 1D

$$\mathbf{Rad} I(t, \theta) = \mathcal{TF}^{-1}_{1D} \left(\hat{I}(k, k\tan\theta), \quad k \in [-n, n[\right),$$

où l'on remplace une grille polaire par son analogue discrète rectopolaire [46] où les pentes sont équi-espacées, et donc θ en variant pas linéairement.

Une autre méthode, proposée par Averbuch *et al.* [3], appelée Fast Slank Stack, redéfinit la transformée de Radon sur une grille cartésienne, en utilisant une image interpolée selon un axe et à laquelle ils appliquent un cisaillement (voir figure 2.17) pour que l'intégration suivant une direction ne nécessite qu'une interpolation 1D et non 2-D.
On note $I(u, v)$ une image sur une grille cartésienne ($-n/2 \leq u, v < n/2$), et on considère la transformée de Radon selon une ligne $x_2 = sx_1 + t$ *plutôt horizontale* (i.e. $|s| < 1$) :

$$\mathbf{Rad} I(y = sx + t, I) = \sum_u \tilde{I}^1(u, su + t)$$

où $\tilde{I}^1$ est un interpolé de I sur le second paramètre par un noyau de Dirichlet D_m. On posera par la suite $s = \tan\theta$ (resp. $s = \cot\theta$) pour les *horizontales* $\theta \in [-\pi/4, \pi/4[$ (resp. *verticales* $\theta \in [\pi/4, 3\pi/4[$). On ne s'intéressera qu'au cas des *horizontales* pour lesquelles les opérateurs seront marqués de l'exposant 1, l'équivalent pour les verticales pouvant être facilement déduit.
On a donc la transformée de Radon (Slant Slack) de $n \times n \to 2n \times 2n$

$$\mathbf{Rad} I(t, \theta) = \sum_u (I(u, \cdot) * D_m)(u, u\tan\theta + t)$$

définie pour $t \in [-n, n[$ et $\theta \in \{\arctan(2l/n) | l \in [-n/2, n/2[\}$, avec une redondance de 4. Il s'agit d'une version algébriquement exacte, géométriquement fidèle, et inversible de la transformée de Radon, mais sa complexité est bien plus grande.

Ridgelets

Une fonction ridgelet de paramètres $\gamma = (a, b, \theta) \in \mathbb{R}^*_+ \times \mathbb{R} \times [0, 2\pi[$ est définie selon Candès [17] par

$$\psi_\gamma = a^{-1/2} \psi \left((x_1 \cos\theta + x_2 \sin\theta - b)/a \right),$$

où $x = (x_1, x_2)$ et ψ est une ondelette (cf. section 2.4) vérifiant la condition d'admissibilité et normalisée ($\int |\hat{\psi}(\xi)|^2 |\xi|^{-2} = 1$).

On peut remarquer que cette fonction est constante selon l'axe $x_1 \cos\theta + x_2 \sin\theta = \textit{constante}$, et est une ondelette sur la direction transversale.
La transformée associée est définie pour les fonctions $f \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ par

$$\langle f, \psi_\gamma \rangle = \mathcal{R}f(a,b,\theta) = \int_{\mathbb{R}^2} f(x)\psi_\gamma^*(x)dx.$$

D'un point de vue algorithmique, la transformée en ridgelets correspond à une transformée de Radon suivie d'une transformée en ondelettes 1D pour chaque direction, ce qui correspond à :

$$\mathcal{R}f(a,b,\theta) = \frac{1}{\sqrt{a}} \int \mathbf{Rad}f(t,\theta)\psi^*\left(\frac{t-b}{a}\right) dt.$$

La figure 2.13 résume les étapes d'une transformée en ridgelets, et la figure 2.14 illustre par un exemple synthétique l'utilité de cette transformée pour débruiter un signal qui est compressible en ridgelets.

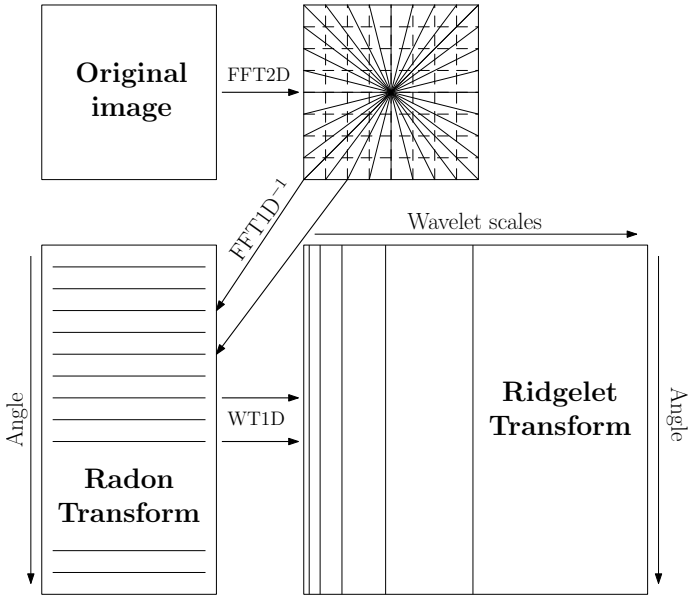


FIG. 2.13 – Schéma montrant les principales étapes d'une transformée en ridgelets en utilisant le théorème de projection de Radon : Transformée de Fourier 2-D, puis extraction des lignes passant par l'origine. Transformée de Fourier inverse puis transformée en ondelettes sur chacune de ces lignes.

2.5.3 Curvelets première génération

Les ridgelets sont bien adaptées pour représenter des lignes à toutes les échelles et orientations traversant l'image de part en part. Cependant, elles ne le sont pas pour des contours dans les images. Lorsque l'on regarde un contour sur une zone suffisamment restreinte, il peut ressembler fortement à une ligne droite. Selon la courbure du contour, il convient de l'estimer par un segment à différentes échelles. C'est l'idée de Candès et Donoho [25] qui proposent la construction en curvelets, plus tard appelées *curvelets-99* ou *curvelets première génération.*
Afin de séparer le signal en bandes spectrales, on utilise un banc de filtres $\forall s \in \mathbb{N}, \Psi_{2s} =$

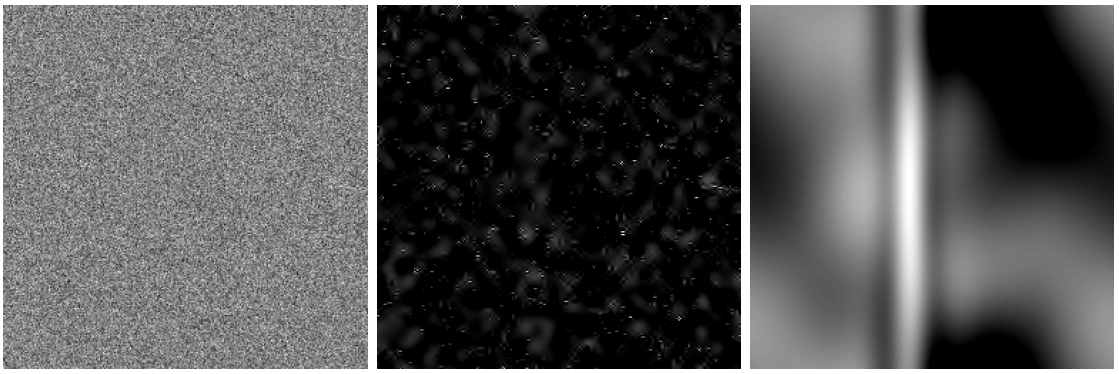


FIG. 2.14 – De gauche à droite : image bruitée (contenant une bande verticale très faible), résultat de débruitage par ondelettes, et résultat en utilisant les ridgelets.

$2^{4s}\Psi(2^{2s}\cdot)$ qui extraient les composantes fréquentielles des bandes $|\xi| \in [2^{2s}, 2^{2s+2}]$, et un filtre passe-bas ψ_0 pour $|\xi| \leq 1$. On obtient donc une partition de l'énergie :

$$|\hat{\psi}_0(\xi)|^2 + \sum_{s\geq 0} |\hat{\Psi}_{2s}(\xi)|^2 = 1.$$

On note $P_0 f = \psi_0 * f$ et $\Delta_s f = \Psi_{2s} * f$.
De même, on effectue une partition spatiale sur des régions Q_s de taille $2^s \times 2^s$

$$Q = Q(s, k_1, k_2) = \left[\frac{k_1}{2^s}, \frac{k_1+1}{2^s}\right[\times \left[\frac{k_2}{2^s}, \frac{k_2+1}{2^s}\right[$$

par des fenêtres lisses w_Q localisées au voisinage de Q, et telles que $\{w_Q\}_{\mathcal{Q}\in\mathcal{Q}_s}$ synthétise une partition de l'unité, i.e. $\sum_{Q\in\mathcal{Q}_s} w_Q^2 = 1$, $\mathcal{Q}_s$ étant la collection des carrés $\mathcal{Q}$ à l'échelle s. Chaque élément de la partition espace-fréquence $w_Q \Delta_s f$ est ensuite re-normalisé sur $[0,1]^2$ par l'opérateur de transport T_Q qui agit sur les fonctions g tel que :

$$T_Q g(x_1, x_2) = 2^s g(2^s x_1 - k_1, 2^s x_2 - k_2),$$

puis subit une transformée en ridgelets. La transformée en curvelets $\mathcal{C}f$ d'une image f est donc

$$\mathcal{C}f = \left\{\mathcal{R}\left((T_Q)^{-1} w_Q \Delta_s f\right) : s \in \mathbb{N}, Q \in \mathcal{Q}_s\right\}.$$

Sous cette définition des curvelets, il faut fusionner les échelles deux à deux afin d'imposer la loi d'échelle parabolique, qui stipule que les curvelets ont une largeur proportionnelle au carré de leur longueur, et qui leur confère des bonnes propriétés d'approximation des images régulières (C^2) au delà de singularités le long de contours C^2 [24]. Les fonctions ainsi obtenues sont localisées en espace et en fréquence, et sont directionnelles. L'implémentation qui fournit les meilleurs résultats [144] diffère par le fait que les échelles ne sont pas regroupées deux à deux, mais que pour obéir à la loi d'échelle parabolique, le nombre de direction n'est doublé qu'une échelle sur deux. Les figures 2.15 et 2.16 montrent respectivement la construction de ces curvelets et les atomes obtenus.

2.5.4 Curvelets rapides seconde génération

Les curvelets-99 se sont montrées très performantes en débruitage [144, 150] comme dans d'autres applications [156, 58], mais souffrent d'un certain nombre de défauts. Tout

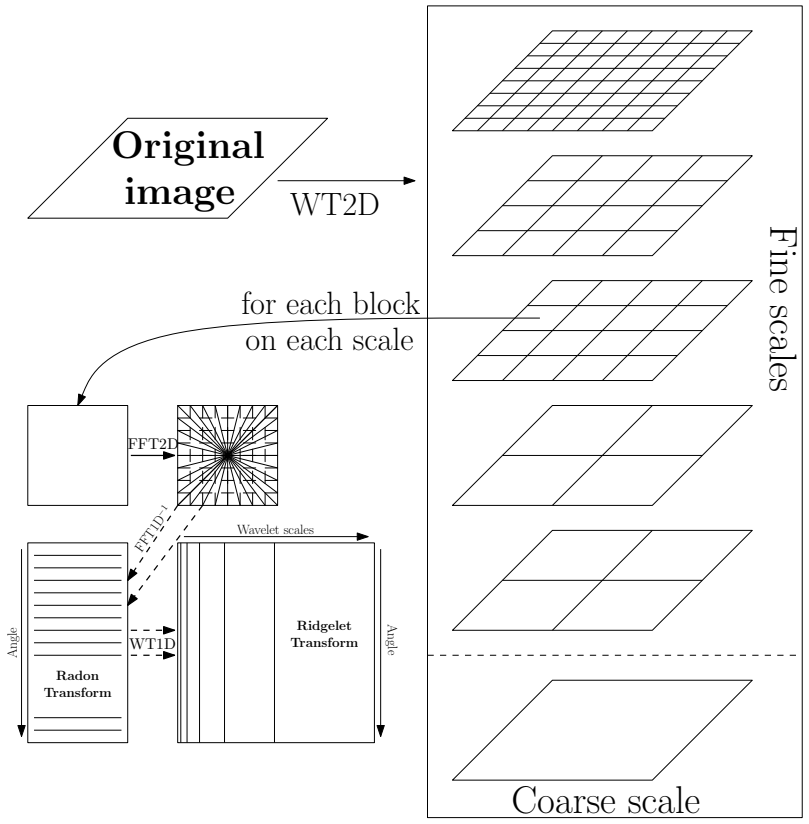


FIG. 2.15 – Schéma illustrant la construction des curvelets-99 : une partition en fréquence (transformée en ondelettes isotropes), puis un pavage respectant la loi d'échelle parabolique (au lieu de fusionner les échelles deux à deux, on peut ne doubler la taille des blocs qu'une échelle sur deux), puis une transformée en ridgelets sur chacun des pavés.

d'abord, leur construction par étapes successives est complexe, et elles possèdent un trop grand nombre de paramètres (deux d'échelle, trois de position, et un d'angle). De plus, elles sont très redondantes (de l'ordre de 16 fois le nombre d'échelles, un facteur 4 des ridgelets, et un autre à cause du recouvrement des blocs du partitionnement spatial), et l'utilisation de ridgelets à toutes les échelles crée des atomes qui obéissent à toute sorte de loi d'échelle. Ces revers ont conduit à une construction d'un autre type de curvelets, ayant seulement trois paramètres, et avec une redondance plus faible.

Cette nouvelle transformée, introduite par Candès *et al.* [18], s'implémente comme un pavage de l'espace de Fourier, considéré comme $[-1,1]^2$. On définit tout d'abord la transformée continue à l'aide des fenêtres V et W, respectivement radiale et angulaire, qui agiront dans le domaine de Fourier :

$$W(r), r \in [1/2, 2] \text{ et } V(t), t \in [-1, 1].$$

Ces fenêtres satisfont les conditions d'admissibilité suivantes, et forment ainsi une partition de l'espace de Fourier :

$$\begin{aligned} \sum_{j\in\mathbb{Z}} W^2(2^j r) &= 1, \qquad r \in \mathbb{R}_+^* \\ \sum_{l\in\mathbb{Z}} V^2(t-2l) &= 1, \qquad t \in \mathbb{R}. \end{aligned} \tag{2.32}$$

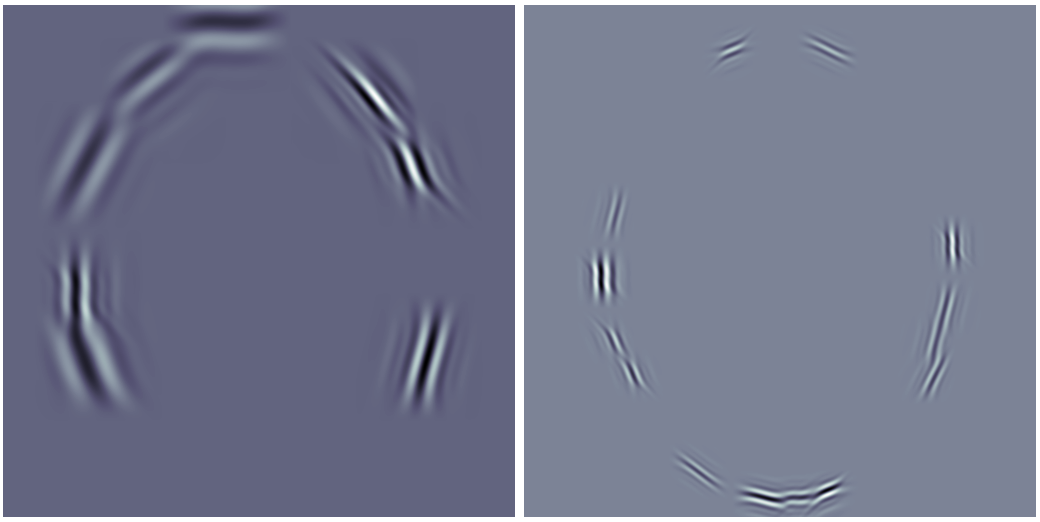


FIG. 2.16 – Exemples d'atomes de curvelet-99 à différentes échelles, positions, et orientations.

Pour toute échelle $j \geq j_0$, on définit une fenêtre fréquentielle U_j par

$$U_j(r,\theta) = 2^{-3j/4} W(2^{-j}r) V(2^{\lfloor j/2 \rfloor}\theta/2\pi).$$

On peut ainsi définir la curvelet à l'échelle j dans le domaine de Fourier,

$$\hat{\psi}_j(\omega) = U_j(\omega),$$

les autres pouvant en être déduites par rotations et translations. La curvelet à l'échelle 2^{-j}, d'orientation $\theta_l = 2\pi 2^{-\lfloor j/2 \rfloor} l, l = 0 \cdots 2^{\lfloor j/2 \rfloor} - 1$ et de position $x_k^{(j,l)} = R_{\theta_l}^{-1}(k_1 2^{-j}, k_2 2^{-j/2})$ est donc définie par

$$\psi_{j,l,k}(x) = \psi_j(R_{\theta_l}(x - x_k^{(j,l)}))$$

avec $R_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$.

La décomposition en curvelets est maintenant obtenue par simple produit scalaire sur cette famille de fonctions : $c(j,l,k) = \langle f, \psi_{j,l,k} \rangle$ qui peut être défini dans le domaine fréquentiel par le théorème de Parseval

$$c(j,l,k) = \frac{1}{(2\pi)^2} \int_{[-1,1]^2} \hat{f}(\omega) U_j(R_{\theta_l}\omega) e^{\imath \langle x_k^{(j,l)}, \omega \rangle} d\omega.$$

On définit ensuite la curvelet à l'échelle grossière en introduisant un filtre passe-bas W_0 vérifiant la partition uniforme de l'unité

$$|W_0(r)|^2 + \sum_{j \geq 0} |W_0(2^{-j}r)|^2 = 1.$$

On construit donc les curvelets d'échelle à partir de la curvelet père

$$\hat{\phi}_{j_0}(\omega) = 2^{-j_0} W_0(2^{-j_0}\sqrt{\omega_1^2 + \omega_2^2}),$$

$$\phi_{j_0,k}(x) = \phi_{j_0}(x - 2^{-j_0}k).$$

Propriétés

- Les curvelets ainsi définies forment une trame ajustée, c'est à dire que toute fonction $f \in L^2(\mathbb{R}^2)$ se décompose selon $f = \sum \langle f, \psi_{j,l,k} \rangle \psi_{j,l,k}$ et vérifie la conservation d'énergie (identité de Parseval) lorsque l'on inclue les éléments des ondelettes pères.
- A l'échelle 2^{-j}, les curvelets ψ_j sont à décroissance rapide en dehors d'un rectangle de longueur $2^{-j/2}$ et largeur 2^{-j},
- Elles ont un nombre infini de moments nuls, par leur support éloigné de l'origine en Fourier.

La transformée discrète

Pour le passage en discrèt, il faut changer quelque peu la forme des fenêtres sur lesquelles vivent les curvelets pour les adapter à une grille cartésienne. Ainsi, le pavage du plan fréquenciel en anneaux circulaires concentriques (filtres passe-bande) se transforme en couronnes cartésiennes concentriques lorsque l'on applique les filtres sur les deux directions ω_1 et ω_2 indépendamment.
Soit ϕ une fonction à support dans $[-2, 2]$ et valant 1 sur $[-1/2, 1/2]$. On pose $\Phi_j(\omega) = \phi(2^{-j}\omega_1)\phi(2^{-j}\omega_2)$. On définit la famille de filtres passe-bande

$$W_j^D(\omega) = \sqrt{\Phi_{j+1}^2(\omega) - \Phi_j^2(\omega)}, j \geq 0$$

qui vérifie $\Phi_0^2(\omega) + \sum_{j\geq 0} {W_j^D}^2(\omega) = 1$.
On peut conserver la même fonction V qu'en continu, à support dans $[-1, 1]$, vérifiant la condition d'admissibilité (2.32) et définir

$$V_j(\omega) = V(2^{\lfloor j/2 \rfloor}\omega_2/\omega_1),$$

et la fenêtre fréquentielle cartésienne associée

$$U_j^D(\omega) = W_j^D(\omega)V_j(\omega).$$

U_j^D isole donc les fréquences au voisinage du polygone (portion d'un secteur angulaire)

$$\left\{(\omega_1, \omega_2) : 2^j \leq \omega_1 \leq 2^{j+1}, -2^{j/2} \leq \omega_2/\omega_1 \leq 2^{j/2}\right\}.$$

Pour paver tout l'espace, il reste à décaler cette fonction non pas à angles équi-espacés comme dans le domaine continu, mais à pentes équi-espacées, ce qui revient à remplacer la rotation R_θ par un cisaillement

$$S_\theta = \begin{pmatrix} 1 & 0 \\ -\tan\theta & 1 \end{pmatrix}.$$

pour des angles compris entre $-\pi/4$ et $\pi/4$. Les pentes à parcourir sont donc dans $[-1, 1[$, à savoir $\tan\theta_l = l \cdot 2^{-\lfloor j/2 \rfloor}, l \in [\![-2^{\lfloor j/2 \rfloor}, 2^{\lfloor j/2 \rfloor} - 1]\!]$. Finalement

$$U_{j,l}^D(\omega) = W_j^D(\omega)V_j(S_{\theta_l}(\omega)).$$

On peut faire de même pour le reste de l'espace de Fourier, en appliquant des rotations de $\pi/2$ radians.
Candès *et al.* [18] ont développé deux méthodes pour calculer les coefficients à partir

de la grille pseudo-polaire décrite ci-dessus, différant dans la manière dont les curvelets sont transposées à une échelle et un angle donnés. La transformée peut donc se calculer par transformée de Fourier rapide non équi-espacée (USFFT) ou par repliement circulaire (wrapping) dans le domaine fréquentiel comme expliqué ci-dessous.

Implémentation par repliement (wrapping)

On considère maintenant la translation avant la rotation, ce qui nous donne la curvelet cartésienne suivante et les coefficients associés :

$$\psi^D_{j,l,k}(x) = \psi^D_j(S^T_{\theta_l}(x-b)),$$

$$c[j,l,k] = \int \hat{f}(\omega) U^D_j(S^{-1}_{\theta_l}\omega) e^{\imath b\cdot\omega} d\omega.$$

Le problème pour calculer la transformée de Fourier, est que les coefficients $U^D_{j,l}[n_1,n_2] = U^D_j(S^{-1}_{\theta_l}\omega)$ ne sont pas dans un rectangle de taille $\sim 2^j \times 2^{j/2}$ aligné avec les axes. $U^D_{j,l}$ est à support dans le parallélogramme $\mathcal{P}_{j,l} = S_{\theta_l}\mathcal{P}_j$. La taille de la fenêtre de calcul importe afin d'obtenir une redondance minimale pour la transformée. On suppose qu'il existe $L_{1,j} \sim 2^j$ et $L_{2,j} \sim 2^{j/2}$ tels que pour toute orientation l, l'ensemble des translations de $\mathcal{P}_{j,l}$ par $L_{1,j}$ et $L_{2,j}$ forme une partition de l'espace. La donnée $d[n_1,n_2] = U^D_{j,l}[n_1,n_2]\hat{f}[n_1,n_2]$ se voit alors calculée par

$$\mathbf{W}d[n_1,n_2] = \sum_{(m_1,m_2)\in\mathbb{Z}^2} d[n_1 + m_1 L_{1,j}, n_2 + m_2 L_{2,j}]$$

$$\mathbf{W}d[n_1 \bmod L_{1,j}, n_2 \bmod L_{2,j}] = d[n_1,n_2].$$

Ainsi, toute l'information contenue dans $\mathcal{P}_{j,l}$ a été ramenée au centre dans le rectangle de taille $L_{1,j} \times L_{2,j}$ par simple ré-indexation des coefficients. L'algorithme 3 résume la transformée en curvelets rapide par repliement.

Algorithm 3: Transformée en curvelet rapide par repliement

Data: Une image f de taille $\mathbf{N} = (N_x, N_y)$.
Input: Nombre d'échelles J, nombre d'angles N_a par côté à l'échelle la plus fine.
begin
1. **Séparation multi-échelle :** appliquer la séparation multi-échelle dans le domaine de Fourier, et obtenir des images de tailles $\mathbf{N}$, $\mathbf{N}/2$, $\cdots$, $\mathbf{N}/2^J$.
2. **Séparation angulaire :**

foreach ***échelle*** $j = 0$ ***à*** J **do**
  foreach ***orientation*** $\mathbf{l} = (l, l')$ **do**
    Multiplier l'image d'échelle j par la fenêtre angulaire $V_{j,\mathbf{l}}$ en Fourier ;
    Repliement : replier le résultat dans un rectangle centré à l'origine de taille minimale $(2^j 3/8 \times 2^{j/2+1}/N_a \times 2^{j/2+1}/N_a)$;
    Appliquer une transformée de Fourier rapide inverse au rectangle pour obtenir les coefficients de curvelet ;

end
Result: La transformée en curvelets rapides de f.

2.5.5 Contourlets

Do et Vetterli [43] ont proposé une construction multi-échelle et directionnelle appelée contourlets, qui sont localisés en espace et en orientation, et obéissent à la loi d'échelle parabolique selon laquelle la largeur (oscillation) des atomes est proportionnelle au carré de leur longueur. Les contourlets sont une implémentation discrète par deux bancs de filtres pour la séparation multi-échelle et angulaire des curvelets (cf. section 2.5.4), dans le sens où elles tentent d'effectuer un pavage similaire de l'espace de Fourier, mais en utilisant un banc de filtres directionnels quand les curvelets sont implémentées directement comme une famille de filtrage passe-bandes fidèles au pavage attendu : cette transformation décompose tout d'abord l'image sur une pyramide Laplacienne, qui la sépare en bandes de fréquences, puis applique à chacune des bandes un banc de filtres directionnels. Ces filtres fonctionnent de la manière suivante : tout d'abord on applique un cisaillement (*shear*) comme le montre la figure 2.17, qui transforme les structures à un angle donné en structures verticales ou horizontales, puis un filtre qui extrait les structures horizontales et verticales. Cette décomposition fournit des fonctions allongées dans une direction, et dont on peut choisir le nombre de directions. Il y a cependant quelques différences avec les curvelets [18]. Tout d'abord les contourlets ne sont pas fidèles aux curvelets car la plupart des filtres ne sont pas nettement localisés en fréquence, ce qui fait que les fonctions sont moins lisses dans la longueur, et plus oscillantes perpendiculairement. De plus, les contourlets ne sont pas supportées par des théorèmes d'approximation comme les curvelets. La figure 2.18 montre des atomes à différentes échelles.

On peut aussi mentionner une construction multi-échelle proposée par Guo *et al.* [77, 76],

FIG. 2.17 – Exemple d'image et de sa version cisaillée (sheared) de 30°. Les structures à 30° sont maintenant verticales.

appelée shearlets. Cette représentation utilise des couronnes cartésiennes concentriques et des cisaillements pour obtenir un pavage de Fourier similaire aux curvelets, mais ayant une direction privilégiée, verticale ou horizontale.

2.6 Constructions 2-D adaptatives

Cette section présente quelques transformées multi-échelles 2-D adaptatives structurées sans apprentissage, à savoir les wedgelets et les bandelettes.

2.6.1 Wedgelets

Donoho [48] a proposé une décomposition adaptative qui soit adaptée aux contours lisses dans les images, les wedgelets. Il s'agit d'une représentation optimale des images

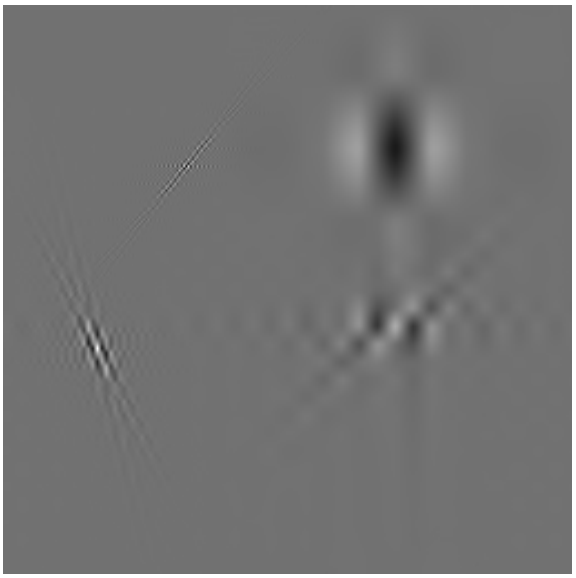


FIG. 2.18 – Atomes de contourlet à différentes échelles et orientations.

appartenant au modèle Horizon [48] des fonctions $\forall(x_1, x_2) \in [0,1] : f(x_1, x_2) = \mathbf{1}_{x_2 > H(x_1)}$, H étant à valeurs sur $[0,1]$. Cette transformée procède à une décomposition dyadique récursive de l'image en carrés dyadiques, à l'aide d'un quadtree, et représente l'image dans chacun de ces carrés par une fonction constante par morceaux avec une discontinuité linéaire. Plus simplement, chaque carré contient deux zones de valeur constante, séparées par une ligne droite. Romberg *et al.* [130] ont ensuite proposé une implémentation rapide en calculant les échelles larges à partir des fines, et en ajoutant une condition d'admissibilité sur les angles des wedgelets, c'est à dire qu'il y a une certaine cohérence, similarité entre les décompositions à différentes échelles, et plus tard Friedrich *et al.* [69] ont proposé une autre implémentation rapide. La figure 2.19 montre un morceau de l'image *cameraman* et l'arbre de wedgelets associé.

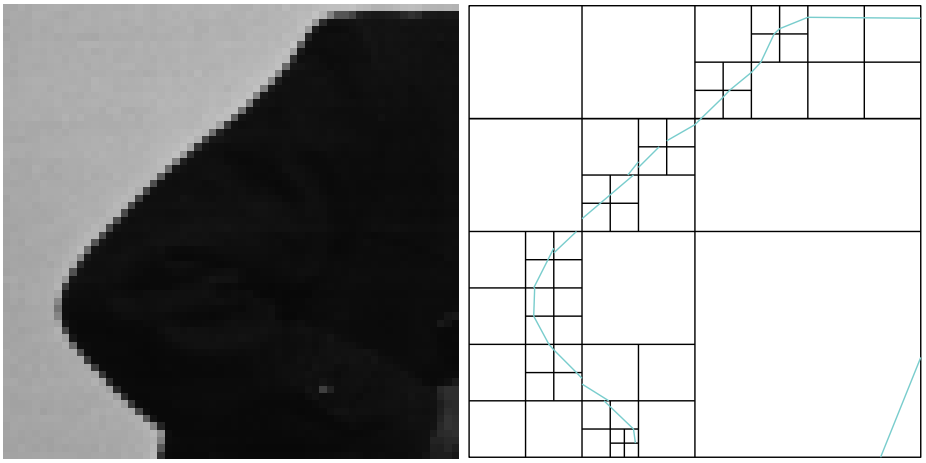


FIG. 2.19 – Extrait de l'image *cameraman* et sa décomposition en wedgelets.

2.6.2 Bandelettes

Créée par Le Pennec, Mallat et Peyré [93, 125], la transformée en bandelettes (seconde génération) est une transformée adaptative de l'image. Elle s'effectue en trois parties principales : tout d'abord l'image subit une transformée en ondelettes orthogonales séparables. Puis chaque sous-bande est segmentée en blocs et sous blocs suivant un quadtree, dans les feuilles duquel on recherche un flot régulier. Une fois la segmentation faite et le flot calculé, les coefficients sont ré-ordonnés dans un vecteur suivant le flot et subissent une transformée en ondelettes unidimensionnelle. Concrètement, la seconde partie peut être de rechercher la direction principale de l'imagette, et de lire les coefficients en considérant cette direction comme l'horizontale. A noter que la transformée originale effectuait un processus similaire, mais décomposait directement l'image et non sa transformée en ondelettes. La figure 2.20 (de Peyre *et al.* [125]) illustre les étapes de la décomposition.

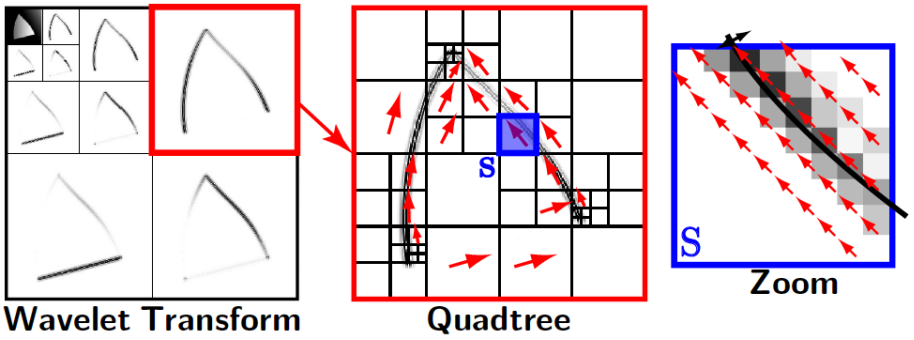


FIG. 2.20 – Une étape de la décomposition en bandelettes. De gauche à droite : la transformée en ondelettes orthogonales d'une image synthétique ; la décomposition en quadtree d'une des sous-bandes et la direction principale de ces imagettes ; illustration du sens de lecture des pixels, parallèlement à la direction principale, avant l'application d'une transformée en ondelettes 1D.

2.6.3 Grouplets

Les grouplets, proposées par Mallat [108] sont une transformée adaptative capable de prendre en compte les régularités géométriques des images, qui peut produire une base orthogonale (grouplets orthogonales) ou une trame (grouplets redondantes). L'image à transformer est séparée en deux grilles disjointes (généralement une colonne sur deux ou une ligne sur deux, selon la direction privilégiée de la donnée), et on associe à chaque point d'une grille un point de l'autre ayant un voisinage similaire, créant ainsi un champ d'association comme défini par Fields *et al.* [66]. Sur ces couples de points est alors appliquée une transformée en ondelettes de Haar, et le processus est répété plusieurs fois de suite sur la version lissée obtenue. Etant adaptée aux images ayant des directions similaires pour une implémentation plus rapide (séparation en lignes ou colones), Mallat a proposé de *grouper les bandelettes*, c'est à dire d'appliquer cette transformée non pas à une image mais à sa décomposition en ondelettes, déjà parcimonieuse et possédant une direction privilégiée. Cette représentation a été utilisée en super-résolution [103] ainsi qu'en synthèse de textures [123].

2.7 Quelques transformées tridimensionnelles

Cette section présente brièvement quelques unes des principales transformées 3-D existantes, dont l'extension naturelle des ondelettes, les surfacelets et les surflets.

2.7.1 Ondelettes

Qu'il s'agisse des ondelettes orthogonales, des ondelettes non décimées, des ondelettes isotropes ou des ondelettes complexes en arbre dual, leur formalisme a été défini directement en dimension quelconque (mais pas forcément la transformée rapide) et donc elles existent naturellement en 3-D. La figure 2.21 illustre les partitions spectrales associées aux transformées en ondelettes orthogonales et isotropes.

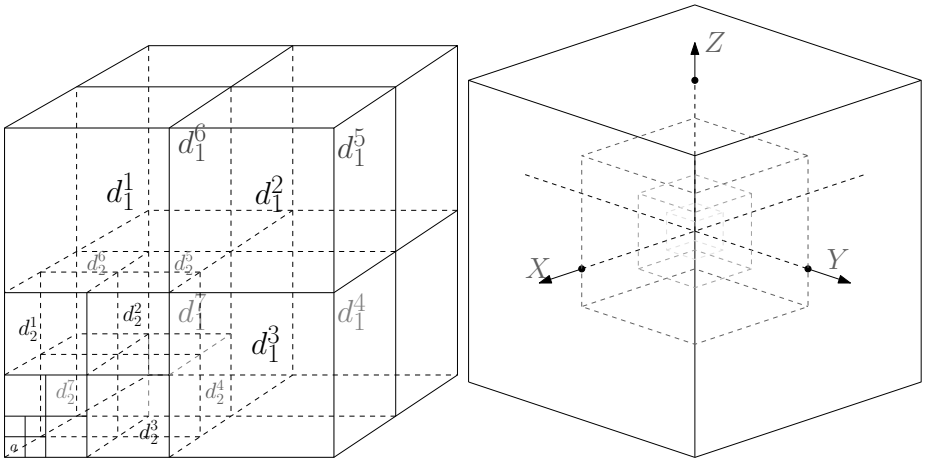


FIG. 2.21 – Pavage de l'espace de Fourier pour deux transformées en ondelettes tridimensionnelles. A gauche avec des ondelettes de Shannon séparables, uniquement sur l'espace $[0, \frac{1}{2}]^3$, et à droite avec des ondelettes de Meyer.

2.7.2 Surfacelets

Afin de construire une représentation ayant des fonctions directionnelles tout en gardant la construction par bancs de filtres, Lu et Do [98] ont proposé les surfacelets. Cette transformée repose sur la combinaison d'une pyramide multi-échelle et d'un banc de filtres directionnel. Les surfacelets produisent un pavage de l'espace de Fourier par des angles d'une manière très proche du pavage des curvelets rapides 3-D, et peuvent être vues comme une extension 3-D des contourlets. Cette transformée a été utilisée en débruitage vidéo [99].

2.7.3 Surflets

Les Surflets [30] sont une extension d-dimensionnelle des wedgelets 2-D [48, 130]. Les surflets sont donc une transformée adaptative qui estime chaque bloc d'une partition spatiale en arbre 2^d-aire de la donnée par deux régions de valeur constante, séparées par une hypersurface polynomiale. Cette décomposition a récemment été utilisée à des fins de compression [31].

2.8 Apprentissage de dictionnaire

Parfois les transformées fixes telles que celles décrites jusqu'à lors ne suffisent pas à bien représenter les données, et on peut décider d'apprendre un dictionnaire adapté aux données, à partir d'une base d'exemples.
Etant donnés N signaux représentatifs $\{\mathbf{y}_i\}_{i=1}^N$ chacun de taille n, on cherche le dictionnaire $\mathbf{D} = [\phi_1, \cdots, \phi_K], \phi_i \in \mathbb{R}^n$ composé de K atomes qui fournisse la représentation la plus parcimonieuse de l'ensemble des signaux $\mathbf{y}_i$. Olshausen *et al.* [121] ont proposé une approche bayésienne en cherchant à minimiser le critère associé à l'estimateur du maximum a posteriori, non-convexe et difficile. Il est

$$(\mathbf{D}, \{\boldsymbol{\alpha}_i\}_i) \in \underset{\{\boldsymbol{\alpha}_i\}_i,\ \mathbf{D}\in\{\forall i, \|\phi_i\|\leq 1\}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \min_{\boldsymbol{\alpha}_i} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1 \right\}, \tag{2.33}$$

en considérant les $\mathbf{y}_i$ comme des patches extraits des images, et la condition $\|\phi_i\| \leq 1$ servant à lever l'ambigüité due à l'invariance par changement d'échelle. Ce problème est généralement résolu par minimisation alternée par relaxation par blocs par rapport à $\mathbf{D}$ et $\{\boldsymbol{\alpha}_i\}_i$ alternativement. On peut séparer le codage des $\mathbf{y}_i$ dans un dictionnaire fixé

$$\min_{\boldsymbol{\alpha}_i} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda\|\boldsymbol{\alpha}_i\|_1, \tag{2.34}$$

qui peut être effectué par des méthodes d'optimisation ℓ_1 comme un seuillage doux itératif [124], et mettre à jour le dictionnaire en résolvant par descente de gradient projeté le problème

$$\min_{\mathbf{D}\in\{\forall i, \|\phi_i\|\leq 1\}} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2. \tag{2.35}$$

D'autres méthodes ont été développées, comme la méthode des directions optimales [59]. En 2006, Aharon *et al.* [1] ont proposé une méthode appelée K-SVD, généralisation de l'algorithme bien connu des K-means. La figure 2.22 montre un dictionnaire de patches obtenu par K-SVD sur l'image *Barbara*.

FIG. 2.22 – Image *Barbara* et le résultat de l'apprentissage par K-SVD à partir des patches 8×8 de l'image.

2.9 Conclusion

Nous avons présenté dans ce chapitre le concept de parcimonie, et les représentations adaptatives ou non, qui permettent de le mettre en application dans le cas des signaux 2-D et 3-D, ainsi que des méthodes de décomposition adaptées à des dictionnaires quelconques comme ceux obtenus par apprentissage. Dans le cas tridimensionnel, les atomes des différentes représentations existantes sont principalement en forme d'assiettes, et c'est ce qui a mené à l'étude présentée dans le chapitre suivant, qui construit un dictionnaire avec des opérateurs rapides d'analyse et de synthèse, pour des atomes linéiques. Un autre problème majeur des transformées 3-D disponibles, est leur redondance, problème abordé dans le chapitre 4.

Chapitre 3

Curvelets 3D première génération

3.1 Introduction

Sparse representations such as wavelets or curvelets have been very successful for 2D image processing. Impressive results were obtained for many applications such as compression (see [31] for an example of Surflet compression ; the new image standard JPEG2000 is based on wavelets rather than DCT like JPEG), denoising [150, 144, 80], contrast enhancement [147], inpainting [58, 61] or deconvolution [155, 156]. Curvelets [144, 26], Bandelets [125] and Contourlets [43] were designed to well represent edges in an image while wavelets are especially efficient for isotropic feature analysis.

In 3D, the separable Wavelet transform (decimated, undecimated, or any other kind) and the Discrete Cosine transform are certainly the most known decompositions [139, 42, 34]. The DCT is mainly used in video compression, but has also been used in denoising [134]. A lot of effort has been made in the last five years to build sparse 3D data representations, which represent better geometrical features contained in the data. The 3D beamlet transform [54] and the 3D ridgelet transform [153] were respectively designed for 1D and 2D features detection. Video denoising using the ridgelet transform was proposed in [29]. The 3D fast curvelet transform [170] consists in paving the Fourier domain with angular wedges in dyadic concentric squares using the parabolic scaling law to fix the number of angles depending on the scale, and has atoms designed for representing surfaces in 3D. The Surflet transform [30] – a d-dimensional extension of the 2D wedgelets [48, 130] – has recently been studied for compression purposes [31]. Surflets are an adaptive transform estimating each cube of a quad-tree decomposition of the data by two regions of constant value separated by a polynomial surface. Another possible representation uses the Surfacelets developed by Do and Lu [98]. It relies on the combination of a Laplacian pyramid and a d-dimensional directional filter bank. Surfacelets produce a tiling of the Fourier space in angular wedges in a way close to the curvelet transform, and can be interpreted as a 3D adaptation of the 2D contourlet transform. This transformation has recently also been applied to video denoising [99].

3D multiscale transforms and cosmological data set

Different statistical measures have been used in the cosmological literature to quantitatively describe the cosmic texture [113], i.e. the large-scale structure of the universe showing intricate patterns with filaments, clusters, and sheet-like arrangements of galaxies encompassing large nearly empty regions, the so-called voids. Wavelets have been used for many years [60, 142, 113], and it has been shown that denoising the galaxy distribu-

tion using wavelet instead of the standard Gaussian filtering, allows us to better preserve structure at different scales, and therefore better constrain our cosmological models [112].

Noise is also a problem of major concern for N-body simulations of structure formation in the early Universe and it has been shown that using wavelets for removing noise from N-body simulations is equivalent to simulations with two orders of magnitude more particles [131, 132].

Finally, 3D walelets, ridgelet and beamlet were also used in order to extract statistical information from galaxy catalogs [153] and compare our data set to simulations obtained from different cosmological models.

Why new transforms ?

These 3D transforms all aim at representing the data using a minimal number of active coefficients, and by construction are better adapted to capture a specific kind of pattern. For the wavelet transform the pattern is smooth and isotropic, while for the DCT it is oscillating in all directions. All previously mentioned 3D transforms, except the beamlet transform, use plate-like functions, useful to represent surfaces in a 3D volume. The beamlet is therefore the only existing geometric decomposition allowing an efficient detection of filaments in 3D. Two relatively different implementations have been proposed. One [54] suffers the lack of any reconstruction algorithm, and the other one [153] is only optimal for the detection of filaments with a specific filament size. This has motivated the design of the FABT transform in biology [9], but this decomposition is also limited, since its optimality is only for filament of a given width. None of these transforms presents the nice scaling properties similar to those of 2D curvelet transform.

This chapter

We propose in this chapter a new transform, the BeamCurvelet transform, which is well adapted for the detection of filaments of different sizes and widths. A minor variation in its construction leads to another transform with plate-like elements, the RidCurvelet transform. Following the notations in [47], a more standard name for these representations would be "Local-k plane Ridgelet Bases in n-D : LRB(k,n)", thus leading respectively to LRB(1,3) and LRB(2,3). Both constructions have interesting scaling properties, and offer exact reconstruction. They can therefore be used for different applications such as denoising or inpainting.

This chapter is organized as follows : In the second and third section, we show how we can extend the 2D curvelet transform to 3D, leading to two new decompositions, the BeamCurvelet and the RidCurvelet. In section 3.4, we investigate different approaches for BeamCurvelet /RidCurvelet denoising. The last section presents applications how these two transforms can be used for inpainting.

3.2 The BeamCurvelet transform

3.2.1 The 2D Curvelet transform

In order to understand our construction for 3D Curvelets, we first recall the simpler but very similar 2D case. For more detailed explanations, see sections 2.5.2 and 2.5.3.

The first generation curvelets [144, 23] were built using the isotropic undecimated wavelet transform [146] and the ridgelet transform [21].

The ridgelet transform is useful for representing global lines crossing an image on its full length the same way the wavelets represent isolated isotropic singularities and shapes. This property is obtained by the Radon transform, which transforms lines into points. A ridgelet function is indexed by a scale parameter a, a position b and an orientation θ. Let $\gamma = (a, b, \theta) \in \mathbb{R}_+^* \times \mathbb{R} \times [0, 2\pi[$. Given a wavelet function ψ, we define a ridgelet ψ_γ by

$$\psi_\gamma = a^{-1/2} \psi \left((x_1 \cos\theta + x_2 \sin\theta - b)/a \right).$$

The ridgelet transform is implemented as a line extraction in Fourier domain, using the projection-slice theorem. The process is shown in Figure 3.1. The main drawback of this

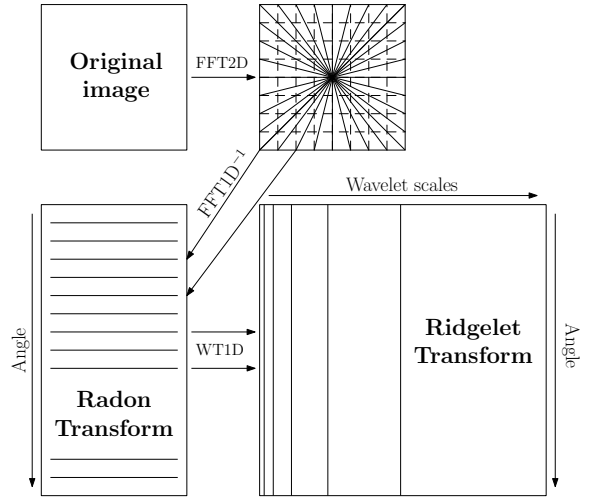


FIG. 3.1 – Scheme showing the main steps of the ridgelet transform in 2D : lines passing through the origin are extracted from the Fourier transform of the image ; a wavelet transform is applied on each of these lines.

transform is that lines span the whole image and thus aren't well localized in space. The idea was then to make it local and multiscale by applying it blockwise on a multiscale isotropic wavelet transform. The key property is a law forcing the size of the blocks from one scale to the next, following a *parabolic scaling* which states that the number of blocks to get in a scale is downsized by a factor four each other scale (the size of the blocks is doubled each other scale on an isotropic undecimated wavelet transform, and divided by two on a decimated one). Figure 3.2 shows the global process described above. The ridgelet transform is implemented through a Radon transform in Fourier domain, followed by a one-dimensional wavelet transform.

As in two dimensions, the 3D first generation Curvelet transforms we develop in this chapter are based on the Radon transform applied to localized blocks of a given size of a 3D spectral decomposition of the data. There are two ways of extending the Radon transform in three dimensions, which lead to the two transforms described below. The first one is obtained by projecting along 2D planes (3D partial Radon transform), which leads to the BeamCurvelets, and the second one by projecting *only* along 1D lines (3D Radon transform), which leads to the RidCurvelets.

3.2.2 The 3D Continuous BeamCurvelet Transform

In order to separate the signal into spectral bands, we use a filter-bank. Let $\mathbb{N}^*$ be the set of strictly positive integers. Given a smooth function $\psi \in L^2(\mathbb{R}^3) : \forall s \in \mathbb{N}^*, \psi_{2s} =$

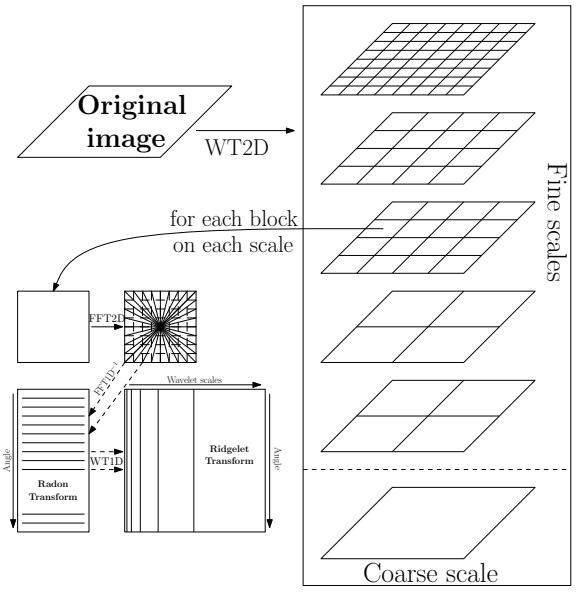


FIG. 3.2 – Scheme showing the main steps of the curvelet-99 transform : the ridgelet transform is applied to each block of each scale of an isotropic 2D wavelet transform.

$2^{6s}\psi(2^{2s}\cdot)$ extracting the frequencies around $|\xi| \in [2^{2s}, 2^{2s+2}]$, and a low-pass filter ψ_0 for $|\xi| \leq 1$. We get a partition of unity :

$$|\hat{\psi}_0(\xi)|^2 + \sum_{s>0} |\hat{\psi}_{2s}(\xi)|^2 = 1 \tag{3.1}$$

Let $P_0 f = \psi_0 * f$ and $\Delta_s f = \psi_{2s} * f$, where $*$ is the convolution product. We can represent any signal f as $(P_0 f, \Delta_1 f, \Delta_2 f, ...)$.
We tile the spatial domain of each scale s with a set $\mathcal{Q}_s$ of regions Q of size 2^s :

$$Q = Q(s, k_1, k_2, k_3) = \left[\frac{k_1}{2^s}, \frac{k_1+1}{2^s}\right[\times \left[\frac{k_2}{2^s}, \frac{k_2+1}{2^s}\right[\times \left[\frac{k_3}{2^s}, \frac{k_3+1}{2^s}\right[\subset [0,1]^3 \tag{3.2}$$

with smooth windows w_Q localized near Q, and verifying $\sum_{Q \in \mathcal{Q}_s} w_Q^2 = 1$, with

$$\mathcal{Q}_s = \left\{ Q(s, k_1, k_2, k_3) | (k_1, k_2, k_3) \in [0, 2^s)^3 \right\}. \tag{3.3}$$

Each element of the frequency-space $w_Q \Delta_s$ is transported to $[0,1]^3$ by the transport operator $T_Q : L^2(Q) \to L^2([0,1]^3)$ applied to $f' = w_Q \Delta_s f$

$$\begin{aligned} T_Q &: L^2(Q) \to L^2([0,1]^3) \\ (T_Q f')(x_1, x_2, x_3) &= 2^{-s} f'\left(\frac{k_1+x_1}{2^s}, \frac{k_2+x_2}{2^s}, \frac{k_3+x_3}{2^s}\right). \end{aligned} \tag{3.4}$$

For each scale s, we have a space-frequency tiling operator g_Q, the output of which lives on $[0,1]^3$

$$g_Q = T_Q w_Q \Delta_s. \tag{3.5}$$

We can apply a 3D Beamlet transform [54, 44] on each block of each scale, by projecting on the beamlet functions :

$$\begin{aligned}\beta_{\sigma,\kappa_1,\kappa_2,\theta_1,\theta_2}(x_1,x_2,x_3) &= \sigma^{-1/2}\phi((-x_1\sin\theta_1 + x_2\cos\theta_1 + \kappa_1)/\sigma, \\ &\quad (x_1\cos\theta_1\cos\theta_2 + x_2\sin\theta_1\cos\theta_2 - x_3\sin\theta_2 + \kappa_2)/\sigma).\end{aligned} \tag{3.6}$$

where σ is the Beamlet scale parameter, (θ_1, θ_2) the orientation parameter and (κ_1, κ_2) the location parameter, which is two dimensional because the beamlet transform integrates the data over one dimension through the partial Radon transform (see section 3.2.3). $\phi \in L^2(\mathbb{R}^3)$ is a smooth function satisfying the following admissibility condition

$$\sum_{s\in\mathbb{Z}} \phi^2(2^s u) = 1, \quad \forall u \in \mathbb{R}^2. \tag{3.7}$$

Finally, the BeamCurvelet transform of a 3D function f is

$$\mathcal{BC}f = \{\langle (T_Q w_Q \Delta_s) f, \beta_{\sigma,\kappa_1,\kappa_2,\theta_1,\theta_2}\rangle : s \in \mathbb{N}^*, Q \in \mathcal{Q}_s\}. \tag{3.8}$$

As we can see, a BeamCurvelet function is parametrized in scale (s, σ), position (Q, κ_1, κ_2), and orientation (θ_1, θ_2). The following sections describe the discretization and the effective implementation of such a transform.

3.2.3 Discretization

For convenience, and as opposed to the continuous notations, the scales are now numbered from 0 to J, from the finest to the coarsest. As seen in the continuous formulation, the transform operates in four main steps.

1. First the frequency decomposition is obtained by applying a 3D wavelet transform on the data with a wavelet compactly supported in Fourier space like the pyramidal Meyer wavelets with low redundancy [149], or using the 3D isotropic *à trou* wavelets.
2. Each wavelet scale is then decomposed in small cubes of a size following the parabolic scaling law, forcing the block size B_s with the scale size N_s according to the formula

$$\frac{B_s}{N_s} = 2^{s/2}\frac{B_0}{N_0}, \tag{3.9}$$

   where N_0 and B_0 are the finest scale's dimension and block size.
3. Then, we apply a partial 3D Radon transform on each block of each scale. This is accomplished by integrating the blocks along lines at every direction and position. For a fixed direction (θ_1, θ_2), the summation gives us a plane. Each point on this plane represents a line in the original cube. We obtain projections of the blocks on planes passing through the origin at every possible angle.
4. At last, we apply a two-dimensional wavelet transform on each Partial Radon plane.

Steps 3 and 4 represent the Beamlet transform of the blocks. The 3D Beamlet atoms aim at representing filaments crossing the whole 3D space. They are constant along a line and oscillate like ϕ in the radial direction. Arranged blockwise on a 3D isotropic wavelet transform, and following the parabolic scaling, we obtain the BeamCurvelet transform.
Figure 3.3 summarizes the beamlet transform, and Figure 3.4 the global BeamCurvelet transform.

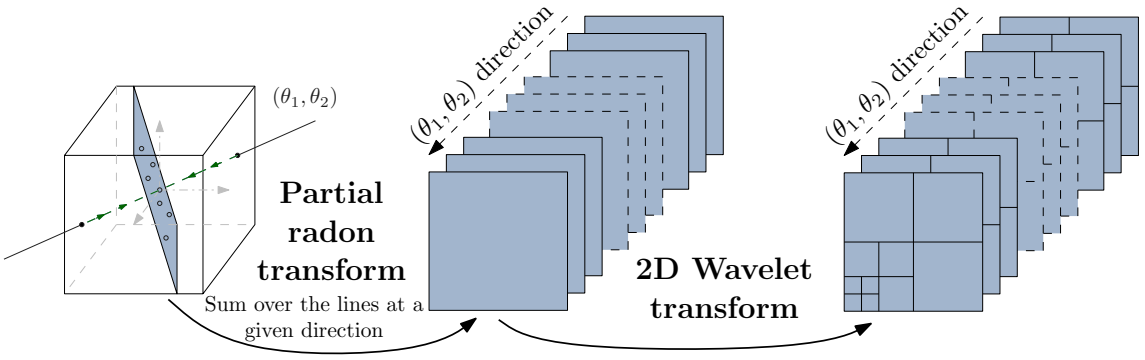


FIG. 3.3 – Schematic view of a 3D Beamlet transform. At a given direction, sum over the (θ_1, θ_2) line to get a $\circ$ point. Repeat over all its parallels to get the dark plane and apply a 2D wavelet transform within that plane. Repeat for all the directions to get the 3D Beamlet transform. See the text (section 3.3) for a detailed explanation and implementation clues.

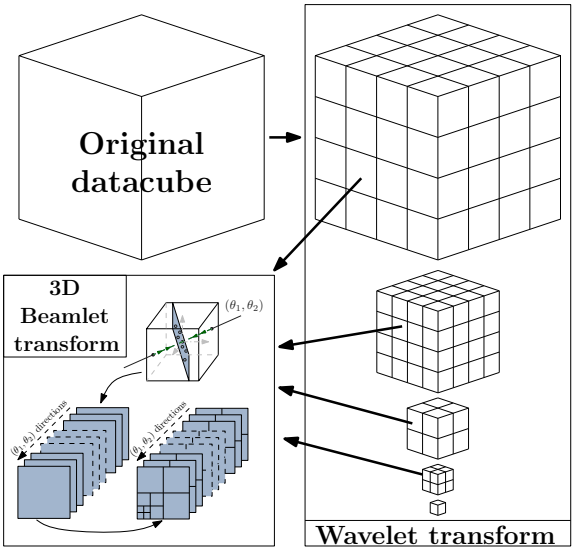


FIG. 3.4 – Global flow graph of a 3D BeamCurvelet transform.

3.2.4 Algorithm summary

As for the 2D Curvelets, the 3D BeamCurvelet transform is implemented effectively in the Fourier domain. Indeed, the integration along the lines (3D partial Radon transform) becomes a simple plane extraction in Fourier space, using the d-dimensional projection-slice theorem, which states that the Fourier transform of the projection of a d-dimensional function onto an m-dimensional linear submanifold is equal to an m-dimensional slice of the d-dimensional Fourier transform of that function through the origin in the Fourier space which is parallel to the projection submanifold. In our case, $d = 3$ and $m = 2$. Algorithm 4 summarizes the whole process.

3.2.5 Properties

As a composition of invertible operators, the BeamCurvelet transform is invertible. As the wavelet and Radon transform are both tight frames, so is the BeamCurvelet transform.

Algorithm 4: The BeamCurvelet Transform

```
Data: A data cube x and an initial block size B
Result: BeamCurvelet transform of x
begin
    Apply a 3D isotropic wavelet transform
    for all scales from the finest to the second coarsest do
        Partition the scale into small cubes of size B
        for each block do
            Apply a 3D FFT
            Extract planes passing through the origin at every angle (θ1, θ2)
            for each plane (θ1, θ2) do
                apply an inverse 2D FFT
                apply a 2D wavelet transform to get the BeamCurvelet coefficients
        if the scale number is even then
            according to the parabolic scaling :
            B = 2B (in the undecimated wavelet case)
            B = B/2 (in the pyramidal wavelet case)
end
```

Given a Cube of size $N \times N \times N$, a cubic block of length B_s at scale s, and $J+1$ scales, the redundancy can be calculated as follows :
According to the parabolic scaling, $\forall s > 0 : B_s/N_s = 2^{s/2}B_0/N_0$. The redundancy induced by the 3D wavelet tansform is

$$R_w = \frac{1}{N^3}\sum_{s=0}^{J} N_s^3, \tag{3.10}$$

with $N_s = 2^{-s}N$ for pyramidal Meyer wavelets (the case used on experiments described in sections 3.4 and 3.5), and thus $B_s = 2^{-s/2}B_0$ according to the parabolic scaling (see equation 3.9).
The partial Radon transform of a cube of size B_s^3 has a size $3B_s^2 \times B_s^2$ to which we apply 2D decimated orthogonal wavelets with no redundancy. There are $(\rho N_s/B_s)^3$ blocks in each scale because of the overlap factor ($\rho \in [1,2]$) in each direction. So the complete redundancy of the transform using the Meyer wavelets is

$$R = \frac{1}{N^3}\sum_{s=0}^{J-1}\left(\rho\frac{N_s}{B_s}\right)^3 3B_s^4 + \frac{N_J^3}{N^3} = 3\rho^3\sum_{i=0}^{J-1} B_s 2^{-3s} + 2^{-3J} \tag{3.11}$$

$$= 3\rho^3 B_0 \sum_{s=0}^{J-1} 2^{-7s/2} + 2^{-3J} \tag{3.12}$$

$$= \mathcal{O}\left(3\rho^3 B_0\right) \quad \text{when } J \to \infty \tag{3.13}$$

$$R(J=1) = 3\rho^3 B_0 + \frac{1}{8} \tag{3.14}$$

$$R(J=\infty) \approx 3.4\rho^3 B_0 \tag{3.15}$$

For a typical block size $B_0 = 17$, we get for $J \in [1, \infty[$:

$$R \in [51.125, 57.8[\quad \text{without overlapping} \tag{3.16}$$
$$R \in [408.125, 462.4[\quad \text{with 50\% overlapping } (\rho = 2). \tag{3.17}$$

3.2.6 Inverse BeamCurvelet Transform

Because all its components are invertible, the BeamCurvelet transform is invertible and the reconstruction error is comparable to machine precision. Algorithm 5 details the reconstruction steps.

Algorithm 5: The Inverse BeamCurvelet Transform

Data: An initial block size B, and the BeamCurvelet coefficients : series of wavelet-space planes indexed by a scale, angles (θ_1, θ_2), and a 3D position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$
Result: The reconstructed data cube
begin
  for ***all scales from the finest to the second coarsest*** **do**
    Create a 3D cube the size of the current scale (according to the 3D wavelets used in the forward transform)
    for ***each block position*** $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$ **do**
      Create a block $\mathcal{B}$ of size $B \times B \times B$
      for ***each plane*** (θ_1, θ_2) ***indexed with this position*** **do**
        – Apply an inverse 2D wavelet transform
        – Apply a 2D FFT
        – Put the obtained Fourier plane to the block, such that the plane passes through the origin of the block with normal angle (θ_1, θ_2)
      – Apply a 3D IFFT
      – Add the block to the wavelet scale at the position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$, using a weighted function if overlapping is involved
    if ***the scale number is even*** **then**
      according to the parabolic scaling :
      $B = 2B$ (in the undecimated wavelet case)
      $B = B/2$ (in the pyramidal wavelet case)
  Apply a 3D inverse isotropic wavelet transform
end

An example of a 3D BeamCurvelet atom is represented in Figure 3.5. The BeamCurvelet atom is a collection of straight smooth segments well localized in space. Across the transverse plane, the BeamCurvelets exhibit a wavelet-like oscillating behavior.

3.3 The 3D RidCurvelet transform

3.3.1 The continuous transform

As referred to in 3.2.2, the second extension of the curvelet transform in 3D is obtained by using the 3D Ridgelet transform [21] instead of the Beamlets. A three-dimensional ridge

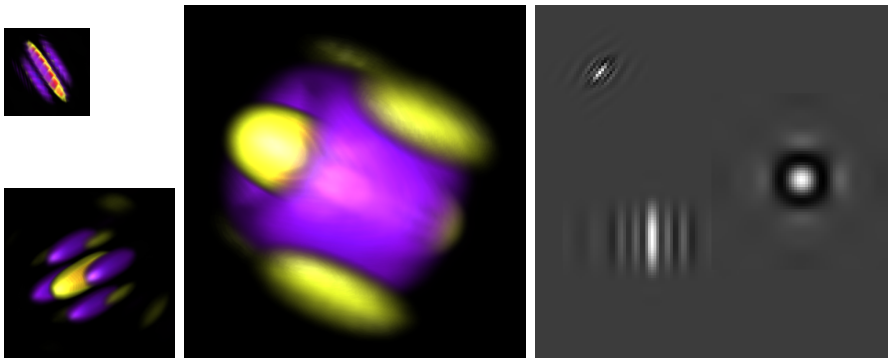


FIG. 3.5 – Examples of a BeamCurvelet atoms at different scales and orientations. These are 3D density plots : the values near zero are transparent, and the opacity grows with the absolute value of the voxels. Positive values are red/yellow, and negative values are blue/purple. The right map is a slice of a cube containing these three atoms in the same position as on the left. The top left atom has an arbitrary direction, the bottom left is in the slice, and the right one is normal to the slice.

function is given by :

$$\rho_{\sigma,\kappa,\theta_1,\theta_2}(x_1,x_2,x_3) = \sigma^{-1/2}\phi\left(\frac{1}{\sigma}(x_1\cos\theta_1\cos\theta_2 + x_2\sin\theta_1\cos\theta_2 + x_3\sin\theta_2 - \kappa)\right), \tag{3.18}$$

where σ and κ are respectively the scale and position parameters, and $\phi \in L^2(\mathbb{R})$ satisfies equation 3.7. The global RidCurvelet transform of a 3D function f is then

$$\mathcal{RC}\, f = \{\langle (T_Q w_Q \Delta_s)\, f, \rho_{\sigma,\kappa,\theta_1,\theta_2}\rangle : s \in \mathbb{N}^*, Q \in \mathcal{Q}_s\}\,. \tag{3.19}$$

3.3.2 Discretization

The discretization is made the same way, the sums over lines becoming sums over the planes of normal direction (θ_1, θ_2), which gives us a line for each direction. The 3D Ridge function is useful for representing planes in a 3D space. It is constant along a plane and oscillates like ϕ in the normal direction. The main steps of the Ridgelet transform are depicted in figure 3.6.

3.3.3 Implementation

The RidCurvelet transform is also implemented in Fourier domain, the integration along the planes becoming a line extraction in the Fourier domain. The overall process is shown in Figure 3.7, and Algorithm 6 summarizes the implementation.

3.3.4 Properties

The RidCurvelet transform forms a tight frame. Additionally, given a 3D cube of size $N \times N \times N$, a block of size-length B_s at scale s, and $J+1$ scales, the redundancy is calculated as follows :
The Radon transform of a cube of size B_s^3 has a size $3B_s^2 \times B_s$, to which we apply a pyramidal 1D wavelet of redundancy 2, for a total size of $3B_s^2 \times 2B_s = 6B_s^3$. There are $(\rho N_s/B_s)^3$

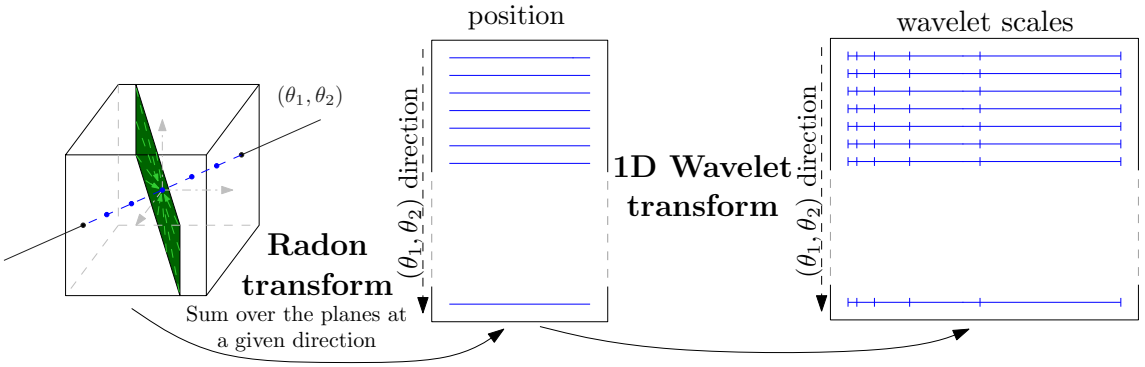


FIG. 3.6 – Overview of the 3D Ridgelet transform. At a given direction, sum over the normal plane to get a • point. Repeat over all its parallels to get the (θ_1, θ_2) line and apply a 1D wavelet transform on it. Repeat for all the directions to get the 3D Ridgelet transform. See the text (section 3.3) for a detailed explanation and implementation clues.

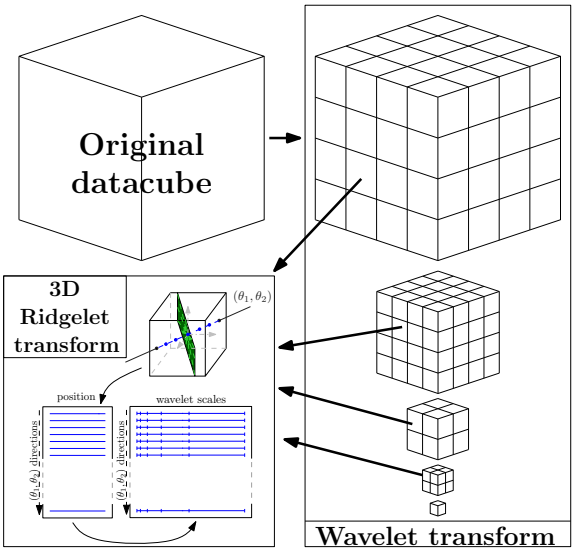


FIG. 3.7 – Global flow graph of a 3D RidCurvelet transform.

blocks in each scale because of the overlap factor ($\rho \in [1, 2]$) in each direction. Therefore, the complete redundancy of the transform using many scales of 3D Meyer wavelets is

$$R = \sum_{s=0}^{J-1} 6B_s^3 \left(\rho \frac{N_s}{B_s}\right)^3 + 2^{-3J} = 6\rho^3 \sum_{s=0}^{J-1} 2^{-3s} + 2^{-3J} \quad (3.20)$$

$$R = \mathcal{O}(6\rho^3) \text{ when } J \to \infty \quad (3.21)$$

$$R(J=1) = 6\rho^3 + 1/8 \quad (3.22)$$

$$R(J=\infty) \approx 6.86\rho^3. \quad (3.23)$$

Algorithm 6: The RidCurvelet Transform

Data: A data cube x and an initial block size B
Result: RidCurvelet transform of x
begin
  Apply a 3D isotropic wavelet transform
  for ***all scales from the finest to the second coarsest*** **do**
    Cut the scale into small cubes of size B
    for ***each block*** **do**
      Apply a 3D FFT
      Extract lines passing through the origin at every angle (θ_1, θ_2)
      for ***each line*** (θ_1, θ_2) **do**
        apply an inverse 1D FFT
        apply a 1D wavelet transform to get the RidCurvelet coefficients
    if ***the scale number is even*** **then**
      according to the parabolic scaling :
      $B = 2B$ (in the undecimated wavelet case)
      $B = B/2$ (in the pyramidal wavelet case)
end

3.3.5 Inverse RidCurvelet Transform

The RidCurvelet transform is invertible and the reconstruction error is comparable to machine precision. Algorithm 7 details the reconstruction steps.

An example of a 3D RidCurvelet atom is represented in Figure 3.8. The RidCurvelet atom is composed of planes with values oscillating like a wavelet in the normal direction, and well localized due to the smooth function used to extract blocks on each wavelet scale.

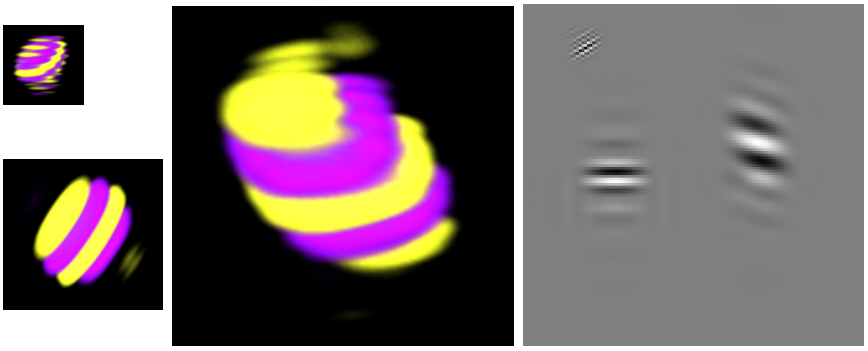


FIG. 3.8 – Examples of RidCurvelet atoms at different scales and orientation. The rendering is similar to that of figure 3.5. The right plot is a slice from a cube containing the three atoms shown here.

3.4 Denoising

3.4.1 Introduction

In sparse representations, the simplest denoising methods are performed by a simple thresholding of the discrete curvelet coefficients. The threshold level is usually taken as

Algorithm 7: The Inverse RidCurvelet Transform

Data: An initial block size B, and the RidCurvelet coefficients : series of wavelet-space lines indexed by a scale, angles (θ_1, θ_2), and a 3D position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$

Result: The reconstructed data cube

begin

- **for** ***all scales from the finest to the second coarsest*** **do**
  - Create a 3D cube the size of the current scale (according to the 3D wavelets used in the forward transform)
  - **for** ***each block position*** $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$ **do**
    - Create a block $\mathcal{B}$ of size $B \times B \times B$
    - **for** ***each line*** (θ_1, θ_2) ***indexed with this position*** **do**
      - Apply an inverse 1D wavelet transform
      - Apply a 1D FFT
      - Put the obtained Fourier line to the block, such that the line passes through the origin of the block with the angle (θ_1, θ_2)
    - Apply a 3D IFFT
    - Add the block to the wavelet scale at the position $(\mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$, using a weighted function if overlapping is involved
  - **if** ***the scale number is even*** **then**
    - according to the parabolic scaling :
    - $B = 2B$ (in the undecimated wavelet case)
    - $B = B/2$ (in the pyramidal wavelet case)
  - Apply a 3D inverse isotropic wavelet transform

end

three times the noise standard deviation, such that for an additive gaussian noise, the thresholding operator kills all noise coefficients except a small percentage, keeping the big coefficients containing information. The threshold we use is often a simple $\kappa\sigma$, with $\kappa \in [3, 4]$, which corresponds respectively to 0.27% and $6.3{\cdot}10^{-5}$ false detections. Sometimes we use a higher κ for the finest scale [144]. Other methods exist, that estimate automatically the threshold to use in each band like the False Discovery Rate (see [8, 114]). The correlation between neighbor coefficients intra-band and/or inter-band may also be taken into account (see [141, 140]). In this chapter, in order to evaluate the different transforms, we use a $\kappa\sigma$ Hard Thresholding in our experiments.

3.4.2 Algorithm

Due to the high redundancy of the proposed transforms, and because of the huge size of 3D data, the transforms were implemented in a filtering-oriented way, in order to spare the machine resources, and enable easy multi-threading. The two new Curvelet transforms operate blockwize, and when there is no intra-block correlation taken into account in the denoising process, each block can be treated independently and on a different processor. Therefore, we never have the full transform in memory, only the wavelet transform of the data, and the Curvelet transform of one block (times the number of CPUs if working on a cluster). Algorithm 8 summarizes the denoising process. Using this algorithm, the memory used to filter a cube of any size using the Curvelet transforms is about twice that of the isotropic Meyer wavelet transform, as the size of one block is usually negligible compared to the size of the data. As an example, a data cube of size $128 \times 128 \times 128$ (8MB with 32bit floating points) transformed with the BeamCurvelet and a block size of 17 with full 50% overlapping requires 3.3GB of memory, while it runs under 70MB with the low memory filtering algorithm. The transform and reconstruction with the RidCurvelet (BeamCurvelet) transform take respectively about 25(260) and 35(460) seconds without overlapping on a laptop (using a single core at 2.16GHz). With full overlapping, it takes about six times more time, but as the code is parallelized with OpenMP, it can use all the CPUs of the computer to lower the execution time dramatically.

Algorithm 8: Curvelet denoising Algorithm

Data: The observed data, the noise level σ and a threshold level $\kappa\sigma$.
Result: The denoised data. *// Memory usage*
begin
  Apply a 3D isotropic Meyer wavelet transform 1 *Wavelet transform*
  Duplicate it and set it to zero +1 *Wavelet transform*
  Separate the following loop on multiple CPUs
  for *each block* $(s, \mathcal{B}_x, \mathcal{B}_y, \mathcal{B}_z)$ ***of all scales except the coarsest*** **do**
    Extract the block +1 *block*
    Apply a 3D Beamlet or Ridgelet transform to the block +1 *transformed block*
    Hard Threshold the coefficients at $\kappa\sigma$
    Apply a 3D Beamlet or Ridgelet inverse transform to the block
    Put the block into the new wavelet scale, or add it with a weighing window if overlapping is involved
  Apply an inverse 3D wavelet transform
end

	RidCurvelets	BeamCurvelets	Undecimated Wavelets	Decimated Wavelets
Plane	**11.78**	8.11	5.06	3.13
Filament	1.02	**2.22**	1.30	1.16

TAB. 3.1 – Maximum value of the coefficients of a transformed cube containing either a plane or a filament, using a given representation.

3.4.3 Experiments

We have performed a few denoising experiments to emphasize the applicability domain and specificity of each transform.

Toy model : Structure detection

First of all, in order to have a quantitative indicator of the efficiency of the two curvelet transforms to approximate and detect specific structures, we transform two cubes, one containing a plane and the second a filament. Table 3.1 contains the maximum coefficients of the transformed cubes, showing the level of detection of each structure in a given space. As we can see, the RidCurvelet has the greatest coefficient for the plane and the BeamCurvelet for the filament. In many applications (denoising, inpainting, morphological component analysis, ...), the process is a thresholding of the coefficients at a level fixed by the parameters of the algorithm and the noise level. Having many coefficients above a given level (or much energy), is an indication of effectiveness.

In order to see the gain when using one transform rather than another, we threshold the RidCurvelet transform of the cube containing the plane at the maximum BeamCurvelet detection level (which is also far above the wavelet one). This means that, had we used another transform besides the RidCurvelet at the same detection level, we would not have detected anything. The reconstruction is shown in figure 3.9 (top row). We have also performed the same experiment with the BeamCurvelet transform of the cube containing the filament with a threshold equal to the level where the three other tested transforms don't detect anything. As expected, there is much information kept by the most *adapted* transform, even at a very high level of thresholding.

Structure denoising

Another way to see the power of each transform when associated to the right structures is to denoise a synthetic cube containing plane- and filament-like structures. Figure 3.10 shows a cut and a projection of the test cube containing parts of spherical shells and a spring-shaped filament. We denoise this cube using wavelets, RidCurvelets and BeamCurvelets. As shown in figure 3.11, the RidCurvelets denoise correctly the shells but poorly the filament, the BeamCurvelets restore the helix more properly while slightly underperforming for the shells, and wavelets are poor on the shell and give a dotted result and misses the faint parts of both structures. The PSNRs obtained with each transform are reported in Table 3.2. Here, the Curvelet transforms did very well for a given kind of features, and the wavelets were better on the signal power. In Starck *et al.* [150], in the framework of 3D image denoising, it was advocated to combine several transforms in order to benefit from the advantages of each of them.

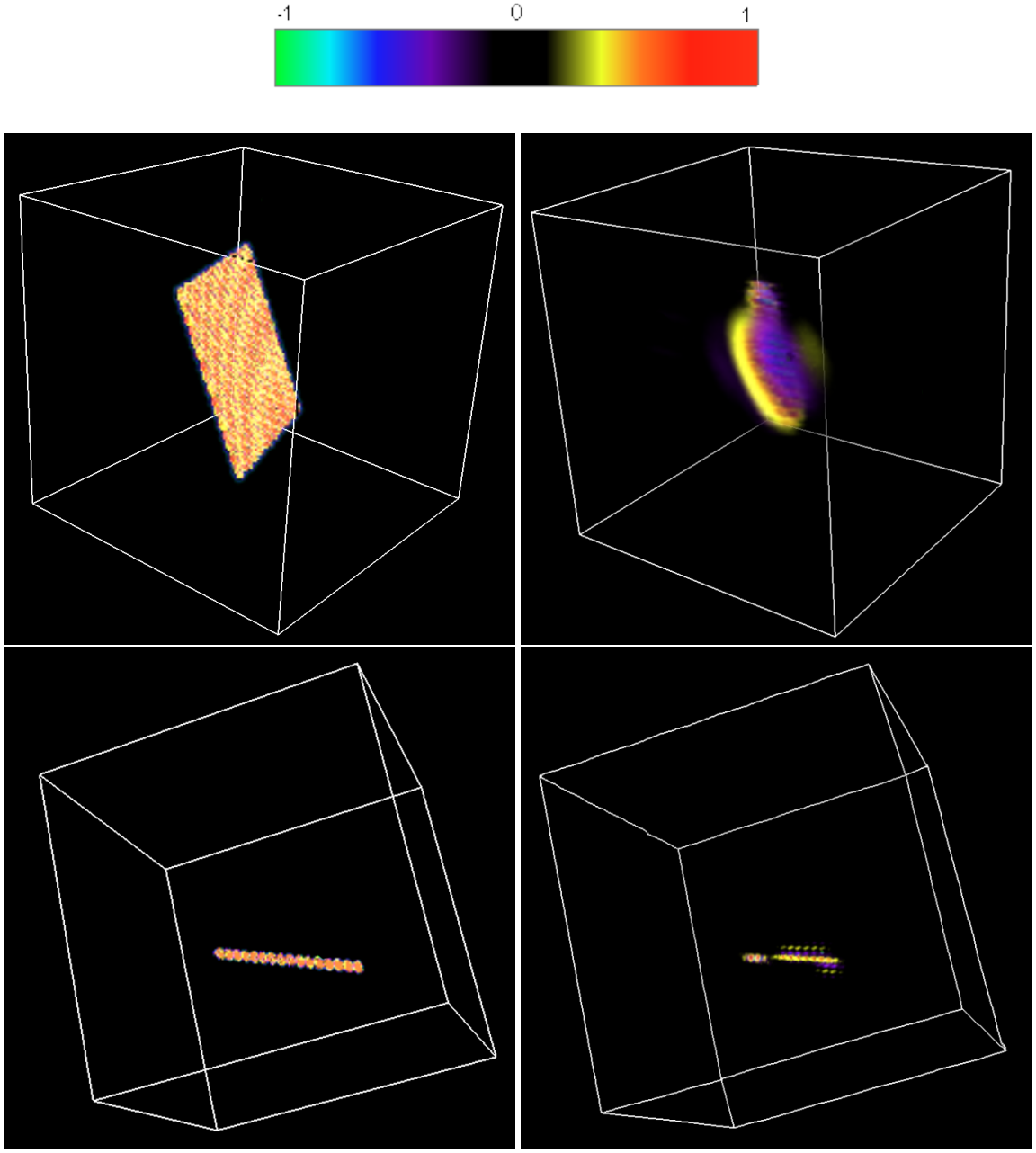


FIG. 3.9 – Top row : a plane, and the reconstruction after thresholding the RidCurvelet coefficients at 8.2 (see table 3.1). Bottom row : a filament, and its reconstruction after the thresholding of the BeamCurvelet coefficients at 1.31.

	Wavelets	RidCurvelets	BeamCurvelets
Shells & spring	40.4dB	40.3dB	43.7dB

TAB. 3.2 – PSNR of the denoised synthetic cube using wavelets, RidCurvelets or BeamCurvelets

FIG. 3.10 – From left to right : a 3D view of the cube containing pieces of shells and a spring-shaped filament, a slice of the previous cube, and finally a slice from the noisy cube.

FIG. 3.11 – From left to right : a slice from the filtered test-cube described in section 3.4.3 (orignial in figure 3.10) by the wavelet transform (isotropic undecimated), the RidCurvelets and the BeamCurvelets.

Combined Filtering

Let x be the unknown data, and z a white gaussian noise map of unitary variance. We observe $y = x + \sigma z$, where σ^2 is the noise variance. A combined filtering algorithm has been developed by Starck *et al.* [150], estimating x as $\tilde{x}$ as described below, using K dictionaries Φ_k, each having a fast transform and reconstruction. We denote by Φ_k^T the forward transform. Let $\alpha_k = (\alpha_{k,i})$ be the coefficient list indexed by i obtained by applying the k-th transform $\alpha_k = \Phi_k^T y$. Let Ω_k^y be the list of coefficients of y in Φ_k which have an absolute value greater than $\kappa\sigma : \Omega_k^y = \{\ i\ :\ |\alpha_{k,i}| > \kappa\sigma\}$, $\kappa\sigma$ being the final threshold level (usually 3σ). The Combined Filtering method consists in minimizing

$$\min_{\tilde{x}} \sum_{k=1}^{K} \left\|\Phi_k^T \tilde{x}\right\|_{\ell_1} \quad \text{s.t.} \quad \forall k \in [1,K] : \left\|\left(\Phi_k^T \tilde{x} - \Phi_k^T y\right)_{\Omega_k}\right\|_{\ell_\infty} \leq \frac{\sigma}{2}, \tag{3.24}$$

where $(\alpha_k)_{\Omega_k}$ denotes the restriction to atoms in Ω_k. The algorithm proposed in [150] solves this minimization problem by iteratively transforming the solution in each dictionary, comparing the coefficients with the noisy data's coefficients, and soft-thresholding with a decreasing level. The main drawback of this algorithm is that it requires to have in memory all coefficients of all transforms during the iterations in order to evaluate the right-hand part of the functional. With the size of 3D data and the redundancy of the proposed transforms, this requires a huge memory size. Hence, using this method, we lose the advantages of the low-memory implementation that make our new transforms useable on a laptop computer with very low memory consumption. Therefore, we propose an alternative approach for the Combined Filtering.

The idea is that the residual between the data and the filtered cube must not be sparse in any of the chosen transforms, and thus leads to small coefficients. It can be interpreted as "There is no structure (in the sense of detection with Φ_k) in the residual". We aim to solve the optimization problem

$$\min_{\tilde{x}} \|y - \tilde{x}\|_{\ell_2}^2 \quad \text{s.t.} \quad \forall k \in [1,K] : \left\|\Phi_k^T (y - \tilde{x})\right\|_{\ell_\infty} < \kappa\sigma \tag{3.25}$$

where σ is the noise estimated on the data (or the energy of the faintest detail we wish to recover). This problem can be solved using Algorithm 9. With this new algorithm, the memory required to treat data of size 128^3 with wavelets, BeamCurvelets and RidCurvelets with overlapping is about 110MB, while it would need about 4GB with the old algorithm if it were optimally coded (not using transforms as black boxes, but interacting with them). In our current implementation, it uses twice that.

We used this Combined Filtering method to denoise the synthetic data in Figure 3.10, and compared to the single-transform denoising PSNRs (see Table 3.2), we get a PSNR of $45.4dB$, which is almost 2dB higher than the best transform alone. Figure 3.12 shows the result of the new CFA.

λCDM denoising

In astrophysics, we use N-body simulations when making an initial homogeneous universe evolve with gravity to form structures, and during this process, there is a noise present in the low density areas. Simulations of Cold Dark Matter (λCDM) exhibit formation of clusters and filamentary, the density of the filaments being a thousand times lower than the clusters. As the noise is important at low density, because of material particles of a given mass, we have to apply a denoising between the time iterations of the simulation in

Algorithm 9: The Combined Filtering

Data: The observed data y, the noise level σ, an initial threshold level $k^{(0)}\sigma$, and a number of iterations N.
Result: The estimate $x^{(N)}$ of the unknown x.
Let $\mathcal{ST}_\lambda$ be the soft-thresholding operator with threshold λ.
Let $\Phi_k, k \in [0, K)$ be the K dictionaries to combine.
Let $\kappa = 3$ be the final threshold level (when multiplied by σ).
begin
  $x^{(0)} = 0$ *// Initial estimate*
  for $n = 0$ ***to*** $N - 1$ **do**
    $u^{(0)} = x^{(n)}$
    $\lambda^{(n)} = (k^{(0)}\sigma - \kappa\sigma)(N - 1 - n)/(N - 1) + \kappa\sigma$ *// Current threshold level*
    for $k = 0$ ***to*** $K - 1$ **do**
      $u^{(k+1)} = u^{(k)} + \Phi_k \, \mathcal{ST}_{\lambda^{(n)}} \left(\Phi_k^T (y - u^{(k)})\right)$ *// New estimate*
    $x^{(n+1)} = u^{(K)}$
end

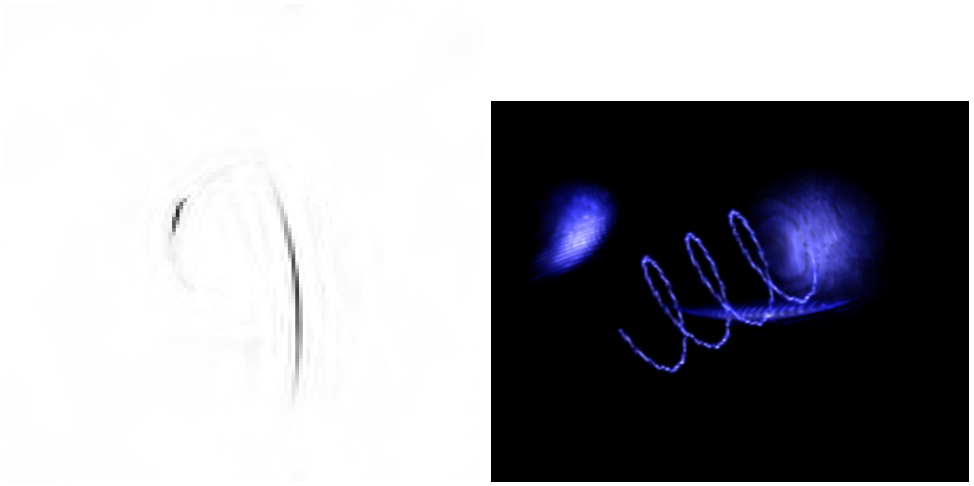


FIG. 3.12 – A slice and a 3D view of the noisy cube on Figure 3.10 after denoising with the presented Combined Filtering.

order to improve it.
In the following example, we show a denoising of a simulated λCDM cube obtained by a simulation using the RAMSES code [159] with cosmological parameters as follows : $\Omega_m = 0.3$ (i.e. 30% of standard matter), $\Omega_\lambda = 0.7$ (70% of Dark Matter, which is interacting gravitationally only), the Hubble parameter $h = 0.7$ (the rate of expansion of the universe) and $\sigma_8 = 0.9$ (the dark matter density fluctuations at present time at scale $8MPc$).
The noise added to the data has a standard deviation ($\sigma = 0.025$) comparable to the filaments amplitude (0.02), which is very small compared to the total amplitude of the data (≈ 100), as the most dense clusters are five orders of magnitude higher than the filaments. We first tried denoising the data using the undecimated wavelet transform, which correctly recovers the high-level clusters, but it fails to separate the filamentary structure from the noise. Facing this problem, we used the BeamCurvelet transform instead. The result, shown in figure 3.13, although recovering the filaments, reveals strong artifacts surrounding the high-density clusters, and therefore is useless. This phenomenon is not due to the noise, but to the form of the BeamCurvelet atoms. Fine-scale BeamCurvelet atoms have an oscillating behavior, and although these oscillations are very low (order $1e-4$ compared to the atom's amplitude), they become a serious problem when structures contained in the data have a huge dynamic range, as in our λCDM simulations where clusters are 10^5 larger than filaments. When the atom is used to approximate a high density object, and when we look round it at scales 10^5 smaller, the oscillations become dominant. In order to solve this problem, we must use conjointly the wavelets and the BeamCurvelets combined in the algorithm we presented in section 3.4.3.

The results of denoising the λCDM data using the new combined filtering algotithm (Alg. 9) are shown in figure 3.14. As expected, the result is far better than using one transform alone, and we restore the filaments very well. The oscillating structure has also naturally completely disappeared from the residual.

3.5 3D Inpainting

3.5.1 Introduction

When measuring quantities on sky observations, we often have to apply a mask to the data to cover inappropriate data, due to the stacking on the line of sight. For example, on the full-sky microwaves images, we have to mask out the whole galactic plane (roughly a cylinder on the sphere) because it is very polluted by the foreground, the Milky Way. In the case of 3D data, there can be a whole cone missing, a sphere of a given radius around the center, or any local area hidden by something. This problem has already been addressed with sparse representations in 2D, for example in weak lensing data [126]. We investigate, in this section, how the 2D sparse inpainting methods can be applied in 3D, using the new 3D Curvelet transforms.

3.5.2 The Inpainting problem

Inpainting is the process of recovering missing parts in altered data from the still available clues. Let x be our three-dimensional data cube with missing data indicated by the mask M. The available data is $y = Mx$. Given a dictionary Φ, we are trying to recover x from the observed y and the mask M. This is an ill posed inverse problem. To get a consistent solution, one must seek regularized solutions. One such regularization is to suppose that x is sparse in one dictionary of atoms Φ, which means that x can be

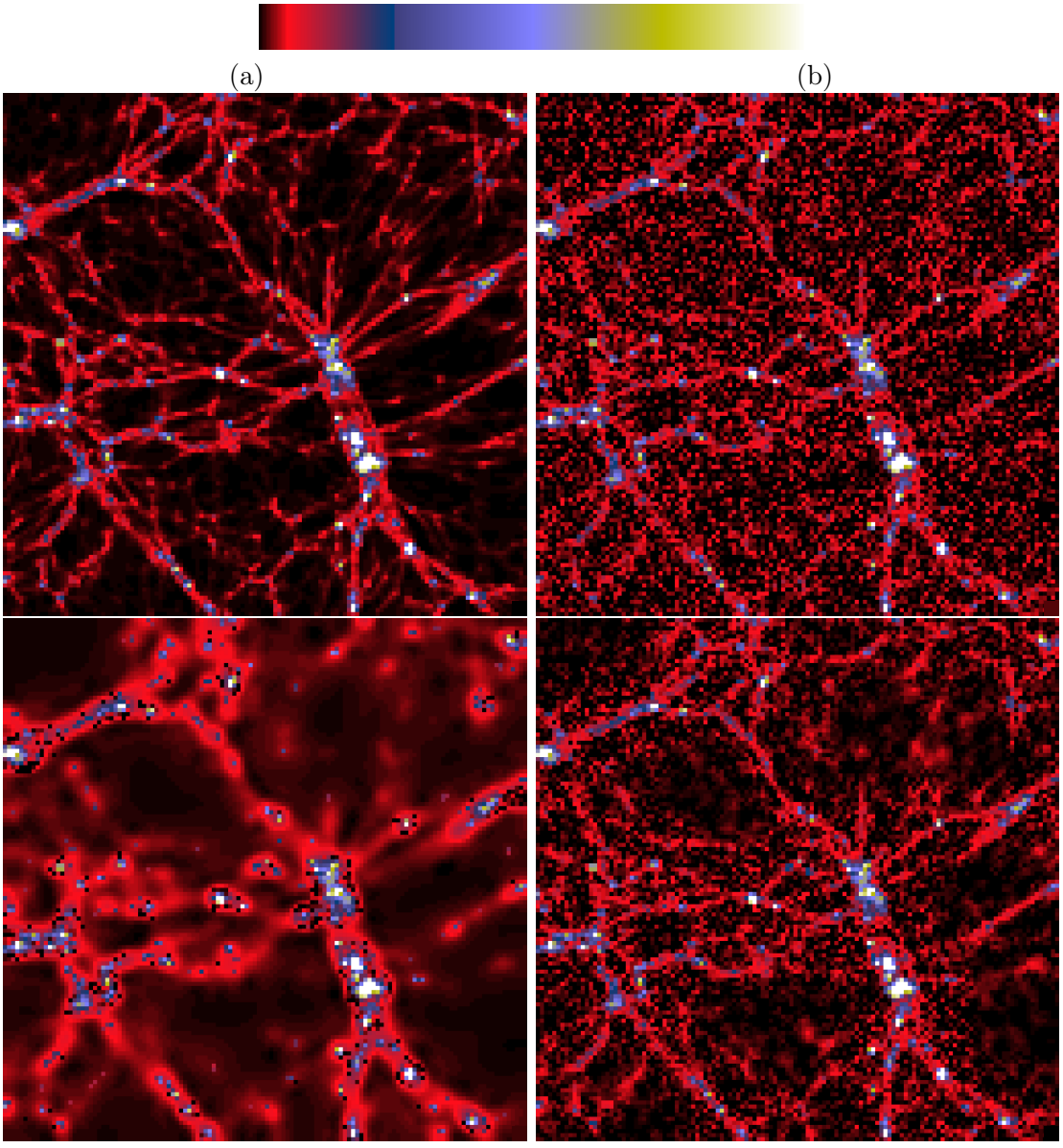


FIG. 3.13 – The central slice of (a) the original λCDM data cube, (b) the noisy data, (c) the data recovered by Wavelet Hard Thresholding, and (d) recovered by BeamCurvelet thresholding.

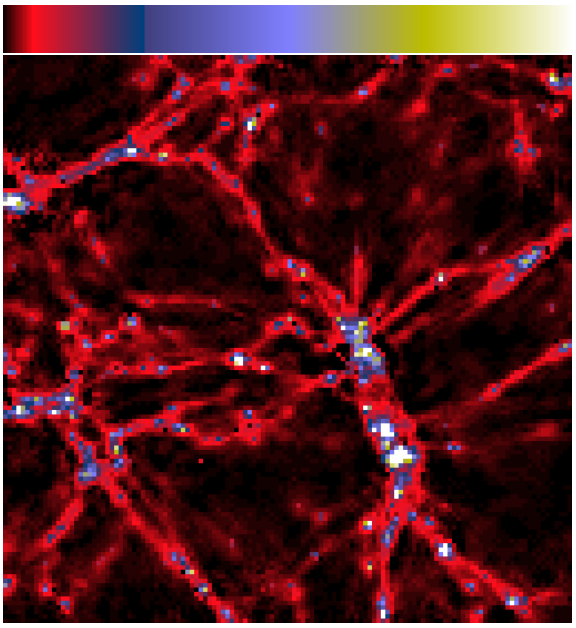


FIG. 3.14 – Denoised λCDM data using the Combined Filtering (see Algorithm 9).

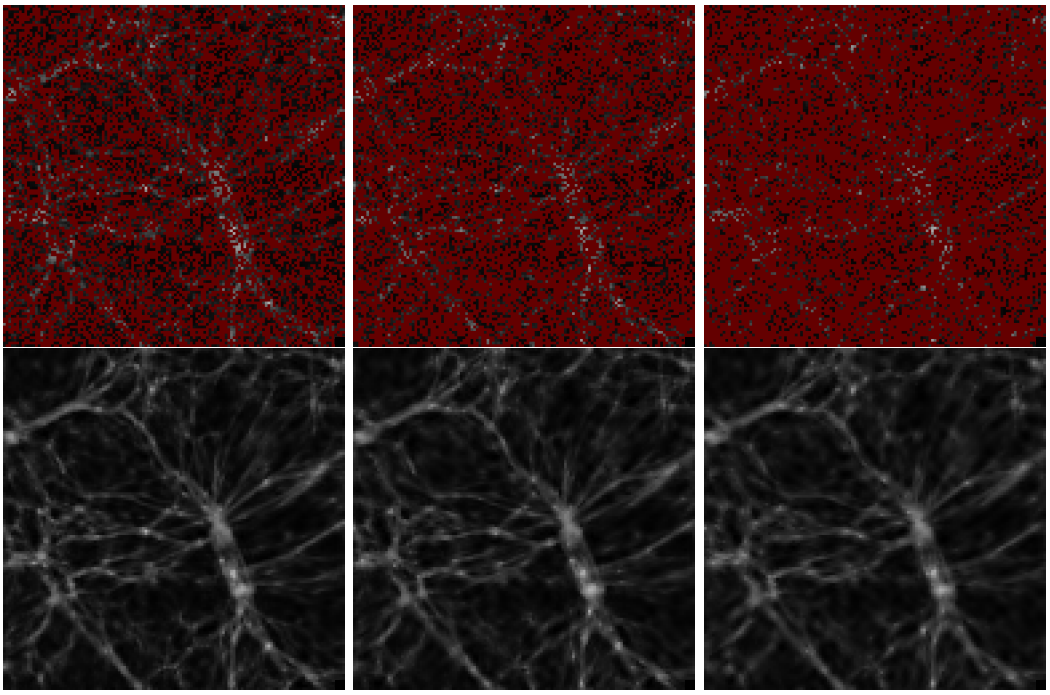
represented by a few atoms from Φ.
Therefore, we want to solve the following inpainting problem :

$$\arg\min_x \|\Phi^T x\|_0 \text{ s.t. } \|y - Mx\|_2 < \epsilon(\sigma), \tag{3.26}$$

where Φ^+ stands for the analysis operator (forward transform). To solve this problem, Algorithm 12 is applied (see [58]).

Algorithm 10: The Inpainting Process

Data: A mask M, the observed data $x = My$, y being the unknown cube , the number of iterations N, an initial threshold level $k^{(0)}$.
Result: The estimate $x^{(N)}$ of y.
Let $\mathcal{HT}_{\lambda^{(n)}}$ be the hard-thresholding operator with threshold $\lambda^{(n)}$.
MAD stands for *Median absolute deviation.*
begin
  $x^{(0)} = y$
  $\forall i \in [0, N) : k^{(i)} = k^{(0)}(N - 1 - i)/(N - 1)$
  for $n = 0$ ***to*** $N - 1$ **do**
    $\lambda^{(n)} = k^{(n)} \cdot MAD\left(\Phi^T x^{(n)}\right)$
    $x^{(n+1)} = \Phi\ \mathcal{HT}_{\lambda^{(n)}}\left(\Phi^T\left[x^{(n)} + M\left(y - Mx^{(n)}\right)\right]\right)$
    $\quad = \Phi\ \mathcal{HT}_{\lambda^{(n)}}\left(\Phi^T\left[(I - M)x^{(n)} + y\right]\right)$
end



(a) 20% missing (b) 50% missing (c) 80% missing

FIG. 3.15 – Central slice of the masked λCDM data with 20, 50, and 80% missing voxels, and the inpainted maps. The missing voxels are dark red.

3.5.3 Example 1 : Random missing voxels

We tried to push the degradation process to see how the BeamCurvelet performed. We applied different masks to the λCDM data, with an increasing percentage of randomly missing voxels. The data are strictly positive (matter density) ; to avoid high dynamic range issues, in the whole inpainting section we work on the logarithm of the data, but for more clarity, we display all figures with a linear scale, truncated at about 10 times the level of the filaments ; the strong clusters are thus saturated. Figure 3.15 presents slices of the original data with missing 20, 50 and 80% missing voxels, and the reconstructed data. The global structure is perfectly recovered in the three examples, and only the faintest filaments are lost in the 80% missing voxels data. The inpainting process and the transform used (the BeamCurvelet transform) are very well adapted to recover this kind of structure. Animations of the inpainting process and 3D representations of the results of this experiment and the following ones can be seen online at *http ://arnaud.woiselle.fr* in the research pannel.

3.5.4 Example 2 : Missing blocks

In order to see the gain when using one transform rather than another in inpainting when the missing data is spatially coherent, we applied a mask to our λCDM cube, with 50% randomly missing voxels, and an additional 3D *checkerboard* array of missing boxes of 343 contiguous voxels each ($7 \times 7 \times 7$). We applied the inpainting algorithm to restore the data using either the RidCurvelet transform, or the BeamCurvelet transform. The results are shown in Figure 3.16. The reconstruction is very good in both cases, but as the BeamCurvelets are more adapted to the structure of the data than the RidCurvelets, their reconstruction is better : the L^2 and L^1 norms of the errors are lower with the BeamCurvelts, and there is visually less residual noise in the reconstruction, which can

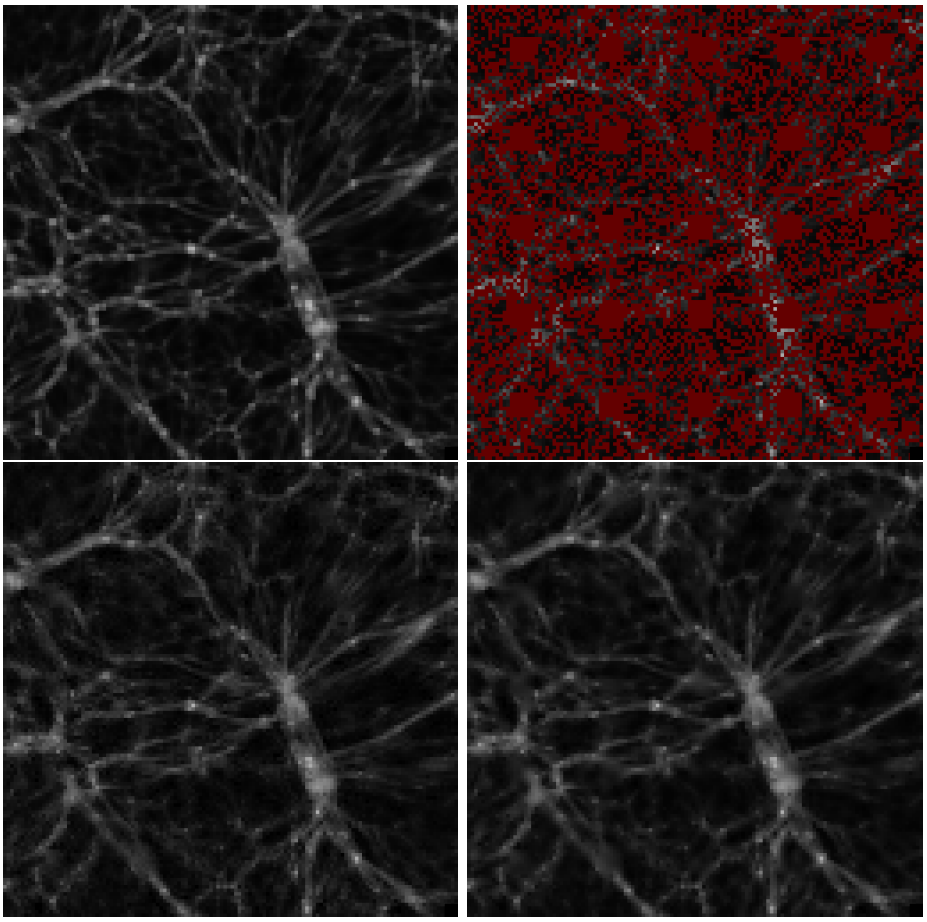


FIG. 3.16 – First row : original central frame of the λCDM data cube, and degraded version with missing voxels in red. Bottom row : the filtered results using the RidCurvelets (left) and the BeamCurvelets (right). Missing voxels are dark red.

be understood as the atoms (planes) of the RidCurvelet transform must spread around the data's filaments more than the atoms (filaments) of the BeamCurvelet transform to approximate the data.

3.6 Conclusion

We have presented two novel multiscale geometric decompositions of a three dimensional volume. The RidCurvelet represents well surfaces and the BeamCurvelet is well adapted for analyzing filamentary features. Preliminary results have shown that they produce interesting results for denoising and inpainting. An important aspect we did not treat is the noise property for denoising applications. Here we have considered only Gaussian noise. However, most 3D astronomical data sets require that we consider Poisson noise. A solution would be to consider non linear 3D decompositions, using both BeamCurvelet and the Multiscale Variance Stabilization, in a similar way to what has been done for the 2D curvelet transform Poisson denoising [173].

Chapitre 4

Curvelets 3D seconde génération faible redondance

Abstract

In this chapter, we first present a new implementation of the 3-D fast curvelet transform, which is nearly 2.5 less redundant than the Curvelab (wrapping-based) implementation as originally proposed in [170, 18], which makes it more suitable to applications including massive data sets. We report an extensive comparison in denoising with the Curvelab implementation as well as other 3-D multi-scale transforms with and without directional selectivity. The proposed implementation proves to be a very good compromise between redundancy, rapidity and performance. Secondly, we exemplify its usefulness on a variety of applications including denoising, inpainting, video de-interlacing and sparse component separation. The obtained results are good with very simple algorithms and virtually no parameter to tune.

Keywords : 3-D Curvelets, sparsity, denoising, inpainting, Morphological Component Analysis, video deinterlacing.

4.1 Introduction

Since the introduction of the multiscale point of view in image processing, an important gap has been crossed with the development of directional transforms such as the curvelet transform [20]. For instance, the latter provide a non-adaptive (near-)optimally sparse representation of smooth curvilinear structures in images, while their wavelet predecessors obtained as tensor-product of 1-D wavelets fail to efficiently represent these anisotropic structures. In fact, the curvelet transform provides atoms that are well localized in space and frequency and exhibit a strong directional selectivity. This transform has found a wide spectrum of applications including denoising [150, 144, 80], contrast enhancement [147], inpainting [58, 61] or deconvolution [155, 156].

With the increasing computing power and memory storage capabilities of computers, it becomes feasible to analyze 3-D data as a volume and not only slice-by-slice, which would mistakingly miss the 3-D geometrical nature of the data. In order to process such massive data, there is an urgent need to design new 3-D multiscale directional transforms. In the literature, some 3-D representations have already been developed, in addition to the natural extension of separable wavelets to 3-D. For example, 2-D wedgelets [48, 130] have

been extended to d-D Surflets for compression purposes [30, 31]. d-dimensional wedgelets correspond to an adaptive transform based on a 2^d-tree spatial partitioning of the data, where each leaf of the 2^d-tree has two constant regions separated by a smooth polynomial manifold of codimension 1. Surfacelets have been proposed in [98] as a non-adaptive transform to efficiently represent singularities along 2-D smooth surfaces. They correspond to a 3-D extension of contourlets, and similarly to contourlets, their implementation relies on a filterbank decomposition of the 3-D data in both scale and angle. They produce a tiling of the 3-D frequency domain quite similar to the one of curvelets although not strictly faithful to the sharp frequency localization of curvelets. The latter were first designed and characterized in a 2-D continuous setting [20, 26], then discretized and extended to higher dimensions [170, 18]. The 2-D curvelet representation was formally proved to yield a (near) optimal sparse representation of piecewise smooth (C^2) images away from singularities along C^2 edges. More generally, the curvelet transform would provide efficient representation of d-D smooth objects with singularities along smooth manifolds of codimension 1. The 3-D implementation of the fast curvelets was proposed in [170, 18] with a public code distributed (including the 2-D version) in Curvelab, a C++/Matlab toolbox available at `www.curvelet.org`. This 3-D fast curvelet transform has found applications mainly in seismic imaging, for instance for denoising [102] and inpainting [81]. However, a major drawback of this transform is its high redundancy factor, of approximately 25. As a straightforward and somewhat naive remedy to this problem, the authors in [170, 18] suggest to use wavelets at the finest scale instead of curvelets, which indeed reduces the redundancy dramatically to about 5.4 (see Section 4.2.3 for details). However, this comes at the price of the loss of directional selectivity of fine details. On the practical side, this entails poorer performance in restoration problems compared to the full curvelet version. Note that directional selectivity was one of the main reasons curvelets were built at the first place.

In this chapter, we present a new implementation of the fast curvelet transform (FCT) with a special emphasis on the 3-D case that has a reduced redundancy, of only 10, without sacrificing the directional selectivity property at the finest scale. Our implementation provides a very good tradeoff between the full FCT and its less-redundant counterpart using only wavelets at the finest scale to reduce the redundancy, as suggested in [170, 18]. In Section 4.2, we present our new implementation of the FCT, and compare it to the one implemented in Curvelab. In sections 4.3 through 4.6, we show how this new curvelet construction can be wisely used for many different applications including denoising, inpainting, video deinterlacing and sparse component separation.

4.2 Low redundancy 3-D fast curvelets

4.2.1 Background

The FCT of a 3-D object consists of a low-pass approximation subband, and a family of curvelet subbands carrying the curvelet coefficients indexed by their scale, position and orientation in 3-D. These 3-D FCT coefficients are formed by a proper tiling of the frequency domain following two steps (see Figure 4.1) :

- Cartesian coronization or multiscale separation : first decompose the object into (Cartesian) dyadic coronae based on concentric cubes ;
- Angular separation : each corona is separated into anisotropic wedges of trapezoidal shape obeying the so-called parabolic scaling law (to be defined shortly). The 3-D FCT coefficients are obtained by an inverse Fourier transform of applied to each

wedge appropriately wrapped to fit into a 3-D rectangular parallelepipeds.

In the rest of this section, we will rehearse the main details of these two steps of the 3-D FCT, and describe the modifications we propose in the implementation to reduce its redundancy.

Symmetry The FCT isolates, in the frequency domain, oriented and localized 3-D wedges. There is a symmetry on a 3-D Cartesian grid : the cube has six faces which can be processed in a similar way. Let $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3) \in [-1/2, 1/2]^3$ be a frequency in the 3-D Shannon band. Exploiting the above symmetry, we will only focus on the subspace $\{\omega_1 > 0, |\omega_2/\omega_1| < 1, |\omega_3/\omega_1| < 1\}$ which is a squared-based pyramid. The five other parts can be dealt with exactly in the same way by symmetry around the origin and exchange of axes.

Cartesian coronization The multiscale separation is achieved using a 3-D Meyer wavelet transform [149, 106], where the Meyer wavelet and scaling functions are defined in Fourier domain with compactly supported Fourier transforms.

Let's denote ψ_j as the Meyer wavelet at scale $j \in \{0, \cdots, J-1\}$, and ϕ_{J-1} the scaling function at the coarsest scale. Denote $M_j = \hat{\psi}_j = 2^{-3j/2}\hat{\psi}(2^{-j}\cdot)$ and $M_J = \hat{\phi}_{J-1} = 2^{-3(J-1)/2}\hat{\phi}(2^{-(J-1)}\cdot)$ their Fourier transforms (see Section 4.2.2 for a formal definition). M_J is a lowpass and the wavelet functions $\{M_j\}_{0\leq j<J}$ is a family of bandpass frequency localized windows that form a uniform partition of the unity. Applied to a 3-D object, the family $\{M_j\}_{0\leq j<J}$ separates it into Cartesian coronae (annuli), and M_J selects its low frequency content. This coarsest subband is kept unaltered, and after an inverse Fourier transform, provides us with the first curvelet coefficients, which corresponds to coarse scale isotropic atoms. Only the next detail scales, i.e. those corresponding to $\{M_j\}_{0\leq j<J}$, have to be processed further.

Angular separation Let N_a be the number of angles on one edge of one face of the finest scale, for a total of N_a^2 angles on each face, and thus six times more bands for the entire considered scale (see Figure 4.1). This number varies with the scale because the number of angles decreases with the scales becoming coarser to obey the parabolic scaling law of curvelets [18], with a parabolic scaling matrix $\mathrm{diag}(2^{-j}, 2^{-j/2}, 2^{-j/2})$ (one short direction and two long ones). This property is essential to ensure that the 3-D curvelets are a basis for sparsely representing smooth trivariate functions with 2-D smooth surface-like singularities.

The vector indexing the angular locations on a face at the j^{th} scale may be expressed with $\mathbf{l} = (l, l') \in \{0, \cdots, 2^{\lfloor -j/2 \rfloor} N_a - 1\}^2$, where $\lfloor \cdot \rfloor$ is the integer part of its argument[1]. Recall that a wedge is the trapezoidal region sharply localized along a given angle at a given scale, see the dark gray area in Figure 4.1. The center of the wedge is on the line going from the origin to the point $(1, \theta_l, \theta'_{l'})$, with

$$\theta_l = \left(-1 + \frac{2l+1}{2^{\lfloor -j/2 \rfloor} N_a}\right), \theta'_{l'} = \left(-1 + \frac{2l'+1}{2^{\lfloor -j/2 \rfloor} N_a}\right). \tag{4.1}$$

We can now define the angular separation by multiplying the dyadic annuli corresponding to the wavelet detail subbands by the smooth angular windows $V_{j,\mathbf{l}}$ in the Fourier domain. The angular windows are built from a smooth real-valued function V supported

[1] On a Cartesian grid, the slopes are equispaced not the angles.

on $[-1, 1]$ and satisfying the partition property

$$\sum_{l=-\infty}^{\infty} V^2(t-2l) = 1 \quad \forall t \in \mathbb{R}. \tag{4.2}$$

The angular window at scale j and orientation $\mathbf{l} = (l, l')$ is then constructed as

$$V_{j,\mathbf{l}}(\boldsymbol{\omega}) \;=\; V\left(2^{\lfloor -j/2 \rfloor} N_a \frac{\omega_2 - \theta_l \omega_1}{\omega_1}\right) \cdot V\left(2^{\lfloor -j/2 \rfloor} N_a \frac{\omega_3 - \theta'_{l'} \omega_1}{\omega_1}\right), \tag{4.3}$$

where θ_l and $\theta'_{l'}$ are defined in (4.1). Note the scaling factor $2^{-j/2}$ as dictated by the parabolic scaling. Owing to (4.2), the family of angular windows $\{V_{j,\mathbf{l}}\}_{\mathbf{l}}$ makes a uniform partition of the dyadic annulus at scale j, i.e.

$$\sum_{\mathbf{l}} V^2_{j,\mathbf{l}}(\boldsymbol{\omega}) = 1 \ . \tag{4.4}$$

However, because of the support constraint on V, this relation does not hold for all $\boldsymbol{\omega}$, and a special care should be taken at the corners where only three out of usually four windows overlap. We thus need to redefine them for (4.4) to hold for any $\boldsymbol{\omega}$. Here is a simple remedy to this problem. Let $\mathbf{l}_a, \mathbf{l}_b, \mathbf{l}_c$ be the indices of the three corner windows, we redefine them on their overlapping domain Ω as

$$\begin{aligned} &\forall \boldsymbol{\omega} \in \Omega, \forall \mathbf{l} \in \{\mathbf{l}_a, \mathbf{l}_b, \mathbf{l}_c\} \\ &V_{j,\mathbf{l}}(\boldsymbol{\omega}) \leftarrow \frac{V_{j,\mathbf{l}}(\boldsymbol{\omega})}{\sqrt{V^2_{j,\mathbf{l}_a}(\boldsymbol{\omega}) + V^2_{j,\mathbf{l}_b}(\boldsymbol{\omega}) + V^2_{j,\mathbf{l}_c}(\boldsymbol{\omega})}}. \end{aligned} \tag{4.5}$$

Piecing all ingredients together, the *scale-angular* wedge at scale-orientation $(j, \mathbf{l})$ is extracted by the frequency window

$$W_{j,\mathbf{l}}(\boldsymbol{\omega}) = M_j(\boldsymbol{\omega}) \cdot V_{j,\mathbf{l}}(\boldsymbol{\omega}) \ , \tag{4.6}$$

which is sharply localized near the trapezoid

$$\left\{(\omega_1, \omega_2, \omega_3) : 2^{j+1} < \omega_1 < 2^j, \left|\frac{\omega_2}{\omega_1} - \theta_l\right| < 2^{\lfloor j/2 \rfloor}/N_a, \left|\frac{\omega_3}{\omega_1} - \theta'_{l'}\right| < 2^{\lfloor j/2 \rfloor}/N_a\right\}. \tag{4.7}$$

Once a wedge is extracted, an inverse Fourier transform must be applied in order to get the curvelet coefficients at the corresponding scale and orientation. Prior to this, the trapezoidal wedge has to be transformed to a convenient form for which the 3-D FFT algorithm applies. As it can be seen from Figure 4.1, the wedge can be inscribed inside a 3-D parallelepiped which is $\sim 2^{j/2}$ long on the (ω_2, ω_3) coordinates (i.e. tangentially), and 2^j on ω_1, i.e. radially. Although this expands the area including the wedge, we can still wrap it inside a rectangular parallelepiped of dimensions $\sim (2^j, 2^{j/2}, 2^{j/2})$ centered at the origin aligned with the axes of the grid (see Section 4.2.2 for further details about wrapping). With appropriate choice of the size of the rectangular parallelepiped, the data does not overlap with itself after wrapping. With the wrapping trick, an inverse 3-D FFT can be readily applied to the rectangular parallelepiped to obtain the curvelet coefficients at the selected scale and orientation.

Algorithm 11 summarizes our implementation of the 3-D FCT and outlines its main

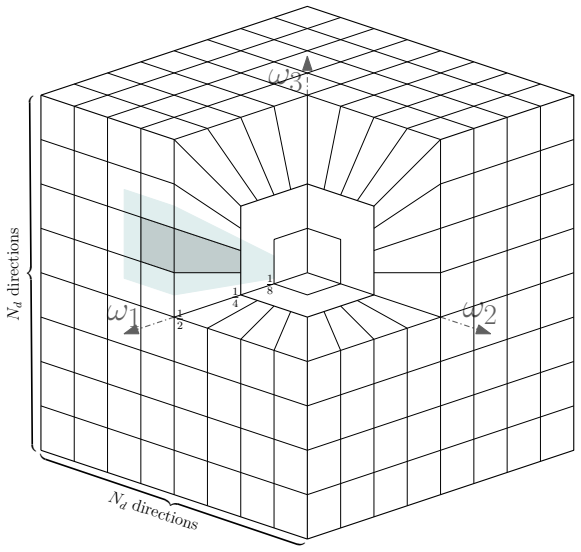


FIG. 4.1 – Example in 3-D of how the number of directions N_a is defined along an edge, (eight in this example), and showing the overlapping region around one wedge. The dark gray area is the main part of the wedge, and the light one represents the overlapping region.

Algorithm 11: The 3-D Fast Curvelet Transform

Data: A 3-D data object f of size $\mathbf{N} = (N_x, N_y, N_z)$.
Input: Number of scales J, number of angles N_a on each face at the finest scale.
begin

1. **Multiscale separation :** apply the 3-D Meyer wavelet transform in Fourier domain, get cubes of sizes $\mathbf{N}$, $\mathbf{N}/2$, ..., $\mathbf{N}/2^J$.
2. **Angular separation :**

foreach ***scale*** $j = 0$ ***to*** J **do**
foreach ***orientation*** $\mathbf{l} = (l, l')$ **do**
Multiply the wavelet cube at scale j with the angular window $V_{j,\mathbf{l}}$ in Fourier ;
Wrapping : wrap the result in a rectangular parallelepiped centered at the origin of minimal size $(2^j 3/8 \times 2^{j/2+1}/N_a \times 2^{j/2+1}/N_a)$;
Apply a 3-D inverse FFT to the rectangular parallelepiped to collect the curvelet coefficients ;

steps.

end
Result: 3-D FCT of f.

We are now ready to explain the differences we introduced in the 3-D FCT implementation to reduce its redundancy.

4.2.2 A new implementation

The overall implementation of this new FCT differs from the original one in several points. The main difference lays in the way to apply the Meyer wavelet transform to the data. Other changes have been introduced as described hereafter.

The multiscale separation

The extra redundancy of the curvelets as implemented in Curvelab originates mainly from the way the radial window is implemented, especially at the finest scale. For the sake of clarity, we will first explain the 1-D case for the Meyer wavelets and show the implications in higher dimensions later on.

The Meyer wavelets $\hat{\psi}(\xi)$ are defined in Fourier domain as follows :

$$\hat{\psi}(\xi) = \begin{cases} \exp^{-i2\pi\xi} \sin(\frac{\pi}{2}\nu(6|\xi|-1)), & \text{if } 1/6 < |\xi| \leq 1/3 \\ \exp^{-i2\pi\xi} \cos(\frac{\pi}{2}\nu(3|\xi|-1)), & \text{if } 1/3 < |\xi| \leq 2/3 \\ 0 & \text{elsewhere} \end{cases},$$

where ν is a smooth function, that goes from 0 to 1 on $[0,1]$ and satisfies $\nu(x)+\nu(1-x) = 1$. Associated to this wavelet is the Meyer scaling functions defined by

$$\hat{\phi}(\xi) = \begin{cases} 1, & \text{if } |\xi| \leq 1/6 \\ \cos(\frac{\pi}{2}\nu(6|\xi|-1)), & \text{if } 1/6 < |\xi| \leq 1/3 \\ 0 & \text{if } |\xi| > 1/3 \end{cases}.$$

Figure 4.2 displays in solid lines the graphs of the Fourier transforms of the Meyer scaling and wavelet functions at three scales.

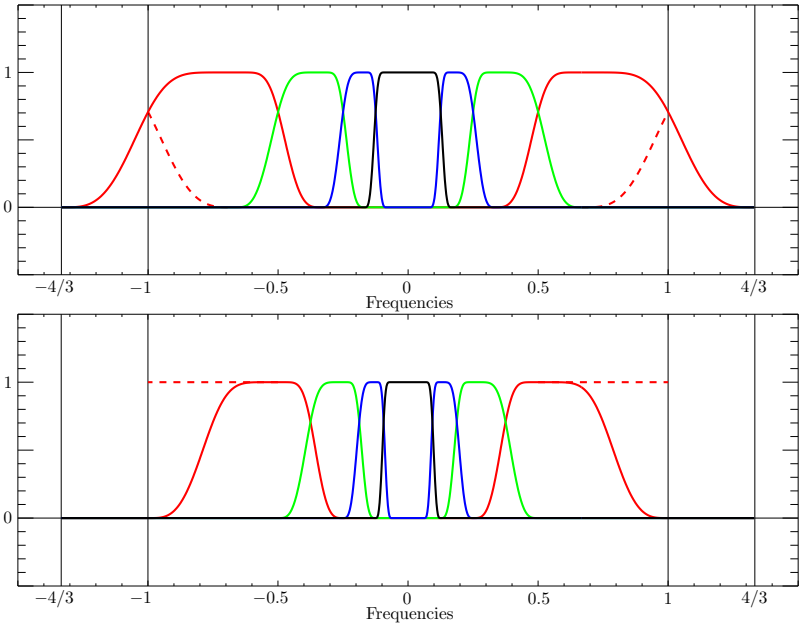


FIG. 4.2 – Meyer scaling and wavelets functions in Fourier domain. In the discrete case, we only have access to the Fourier samples inside the Shannon band $[-1/2, 1/2]$, while the wavelet corresponding to the finest scale (solid red line) exceeds the Shannon frequency band to 2/3. Top : In the Curvelab implementation, the Meyer wavelet basis is periodized in Fourier, so that the exceeding end of the finest scale wavelet is replaced with the mirrored dashed line on the plot. Bottom : In our implementation, the wavelets are shrunk so that they fit in the $[-1/2, 1/2]$ Shannon band, and the decreasing tail of the finest scale wavelet is replaced by a constant (dashed red line) to ensure a uniform partition of the unity.

There is a pair of conjugate mirror filters (h, g) associated to (ϕ, ψ) whose Fourier transforms $(\hat{h}, \hat{g})$ can be easily deduced from $(\hat{\phi}, \hat{\psi})$. $\hat{h}$ and $\hat{g}$ are thus compactly supported.

As a consequence, the Meyer wavelet transform is usually implemented in the Fourier domain by a classical cascade of multiplications by $\hat{h}$ and $\hat{g}$ [2]. However, the wavelet at the finest scale is supported on $[-2/3, -1/6[\cup]1/6, 2/3]$, hence exceeding the Shannon band. This necessitates to know signal frequencies that we do not have access to.

As the FCT makes central use of the FFT, it implicitly assumes periodic boundary conditions. Moreover, it is known that computing the wavelet transform of a periodized signal is equivalent to decomposing the signal in a periodic wavelet basis. With this in mind, the exceeding end of the finest scale wavelet is replaced with its mirrored version around the vertical axis at $|\xi| = 1/2$, as shown in dashed line in the top plot of Figure 4.2. Consequently, the support of the data to treat is $4/3$ larger than the original one, hence boosting the redundancy by a factor $(4/3)^d$ in d-D. The left part of Figure 4.3 shows the implication of periodization in two dimensions for the original curvelets in Fourier domain. The finest scale curvelets have the end of their decreasing window at high frequencies periodized[3].

In our implementation, we followed another path. First, toward our quest of reducing the redundancy, the supports of the scaling and wavelet functions (hence filters) were shrank by a factor of $4/3$. Furthermore, to maintain the uniform partition of unity, which plays an important role for isometry of the transform, following [149], the finest scale wavelet is modified by suppressing its decreasing tail so that the wavelet becomes a constant over $]-1/2, -1/4]\cup]1/4, 1/2]$ (see the dashed line at the bottom of Figure 4.2). The right part of Figure 4.3 shows the impact of the proposed modifications on the 2-D curvelets in the frequency domain. This strategy and the conclusions carry over to the 3-D case.

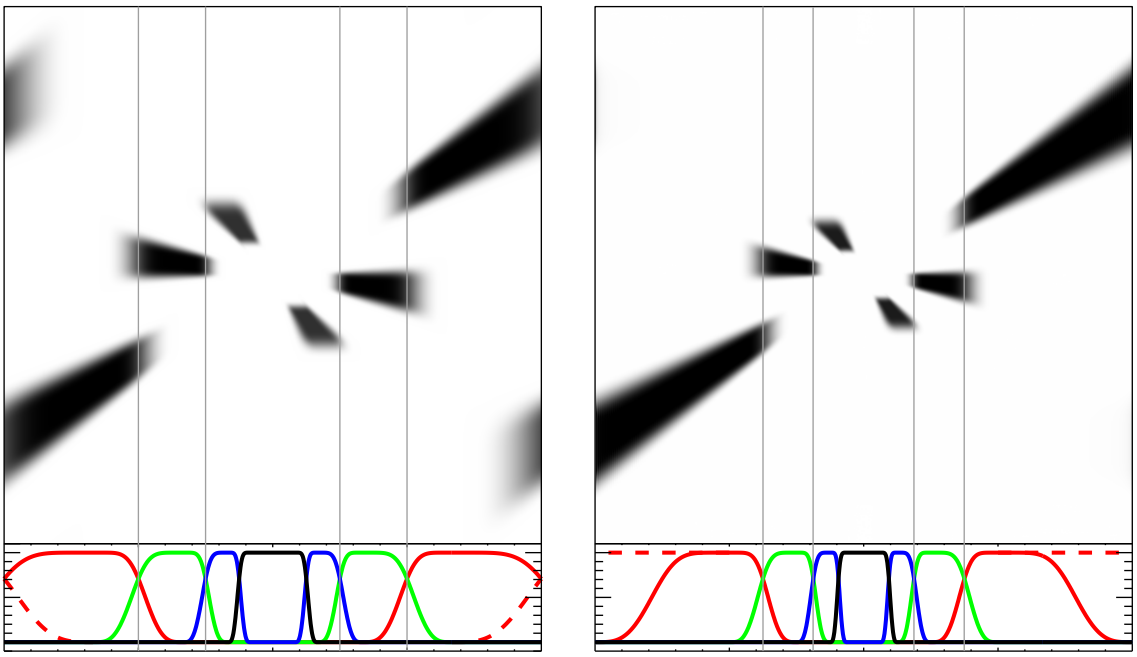
This proposed modification of the Meyer wavelets indeed reduces the redundancy of the transform (see Section 4.2.3). However, this comes at the cost of some changes undergone by the curvelet atoms at the finest scale. First of all, they are less sharply supported in the spatial domain than the original curvelets because of the discontinuity of their Fourier transform, while the decay of the other curvelets remain unchanged. This is one possible reason underlying the slight difference in denoising performance between the original and the modified FCT as shown in Section 4.2.5. Secondly, they obey a parabolic scaling but with a different constant compared to the curvelets at the other scales.

Corner extraction

The overlapping angular windows around the edges and corners are also implemented differently, leading to naturally normalized windows around the edges, but we still need to re-normalize them around the corners, which can not be avoided due to the local geometry. As the cube has a symmetry of rotation between its faces, we will study the wedge extraction only on the *positive* x face, corresponding to the sub-space $\omega_1 > 0$ et $|\omega_2/\omega_1| < 1, |\omega_3/\omega_1| < 1$. As we study angular windows, we can restrain us to the surface of the cube and forget the distance to the centre. On this face, the points are indexed by the coordinates $(1, y, z)$, or just $(\theta_Y, \theta_Z) = (\arctan(y), \arctan(z))$, to which we add $\theta_X = \arctan(y/z)$ for commodity. Figure 4.4 shows the coordinates system and the areas of the windows near the corner $(1, 1, 1)$. As we can see, only three windows, instead of usually four, overlap in the corner region. The light gray area is covered by the corner windows of each face (C_X, C_Y, C_Z),

[2] Here, we do not deal with an orthonormal version of the Meyer wavelet transform which can be viewed as a lapped orthogonal transform in the Fourier domain and efficiently implemented with folding.

[3] FFT and periodization have another side effect in the spatial domain this time, that curvelet atoms localized near the boundaries wrap around to the opposite edge. Indeed,In the 2-D case, mirror-extended (reflexive boundary conditions) curvelets have been proposed in [40] to mitigate the curvelet-looking artifacts due to periodization, but their redundancy is still as high as regular curvelets.



(a) (b)

FIG. 4.3 – Top : Examples of 2-D (real) curvelets in Fourier domain at three consecutive scales and different orientations, with the zero frequency at the center. From the outermost edge to the inside, a finest scale curvelet and two lower scale curvelets. (a) According to the Curvelab implementation, and (b) with our modified low-redundancy implementation. Bottom : the corresponding Fourier transforms of 1-D Meyer scaling and wavelet functions.

and the dark gray triangles around are covered by (C_X, C_Y, A_Y) or $C_X, C_Z, A_Z)$.
The solution to have a proper partition of the space, is to renormalize the three windows in these areas. Considering a point A_x in the light gray region, we have to sum the components of this point in the windows (C_X, C_Y, C_Z), that is we have to find its coordinates in each referential. Of course, the point a_x as seen from C_X has (θ_Y, θ_Z) for coordinates, see figure 4.5. Because of rotation symmetry, the point a_x as seen from C_Y has the coordinates of a_z from C_X, which is $(\pi/2 - \theta_X, \pi/2 - \theta_Y)$. In a same way, the point a_x as seen from C_Z has the coordinates of a_y from C_X, which is $(\pi/2 - \theta_Z, \theta_X)$. We can now divide the current window at the current point a_x by the sum of its values at a_x, a_y and a_z.
A similar method, however a bit more complicated, is used to renormalize the windows over the six triangles surrounding the center of the corner.

Ensuring zero-mean subbands

In the original wrapping-based FCT [18, 39], the wedges are wrapped around the origin using a simple modulo operator, which makes every point fit into a well-sized rectangular parallelepiped centered at the origin whose size is designed so as the data does not overlap with itself after wrapping. However, nothing prevents the center of the parallelepiped from receiving a significant non-zero wrapped Fourier coefficient. After an inverse FFT of the wrapped wedge, it is likely to obtain curvelet coefficient subbands with non-zero means. This is obviously unsuitable since curvelet coefficients are expected to represent high frequency content, and typical thresholding-based processing (e.g. denoising) will be hampered in such a situation. Hopefully, the size and position of the wedges are so that

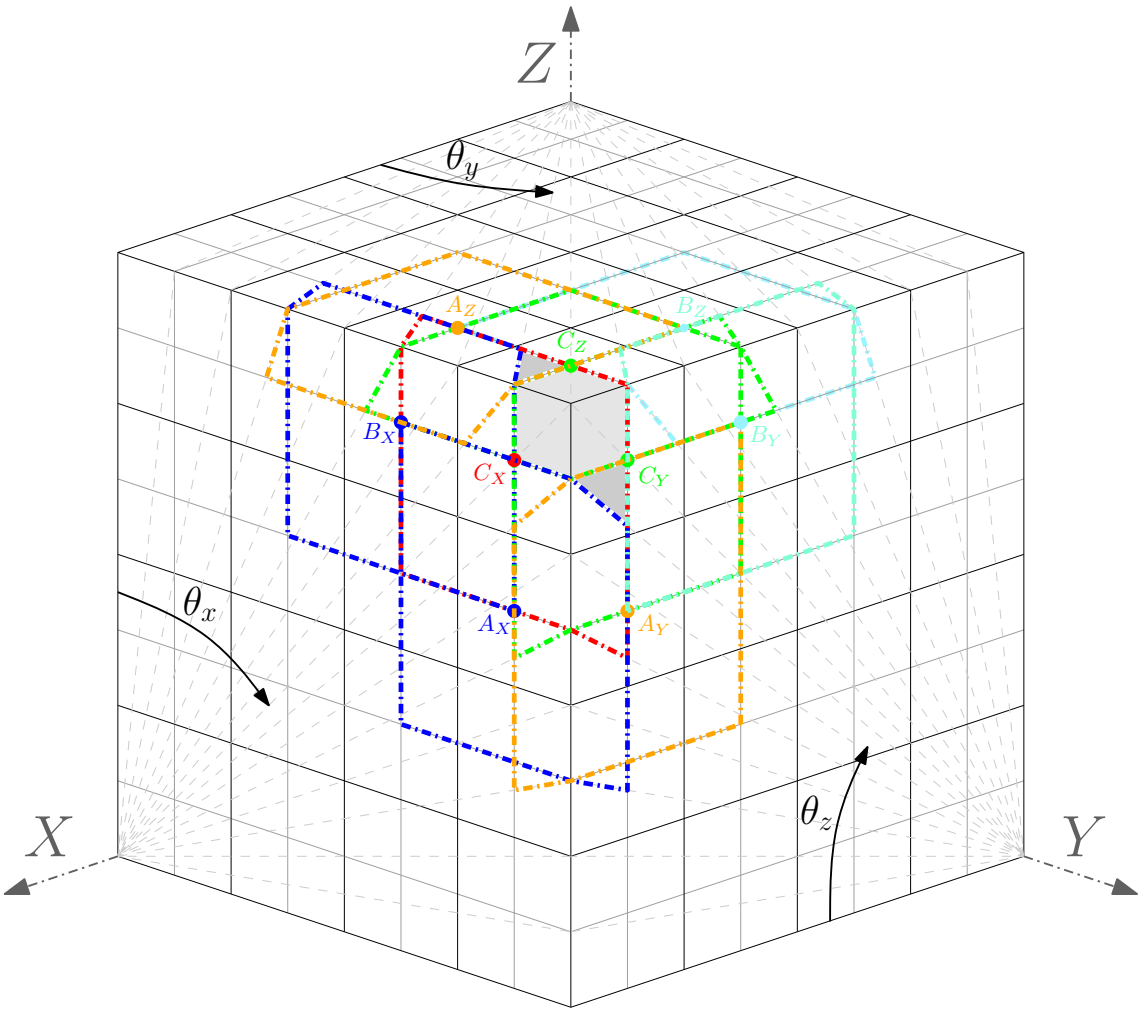


FIG. 4.4 – Overlaping windows (dotted coloured lines) around a corner. Usually four windows overlap, but near the corner, only three : the light gray area is covered by the red and the two green tiles (C_X, C_Y, C_Z) which are the corner ones. The dark gray areas are covered by C_X one green and one yellow, either (C_Y, A_Y), or (C_Z, A_Z).

this misleading phenomenon is generally prevented in practice. Nevertheless, this is not guaranteed in general.

Therefore, in order to ensure zero-mean curvelet subbands, a straightforward solution is to translate each rectangular parallelepiped where a wedge has to be wrapped in such a way that the center (zero frequency) gets a true zero coefficient, i.e. a point out of the wedge support, and then to wrap the data around the translated box. Doing so, the curvelet subbands are ensured to be zero-mean valued after wrapping. Figure 4.6 illustrates the difference between the two wrapping strategies in 2-D for the sake of legibility. The technique extends readily to the 3-D case.

Fine curvelet band normalization

Due to the change of the multi-resolution separation, and to the peculiarity of the finest scale filter compared to the others, we need to estimate the normalization coefficients of this new curvelet implementation. Since the proposed implementation of the curvelets is a

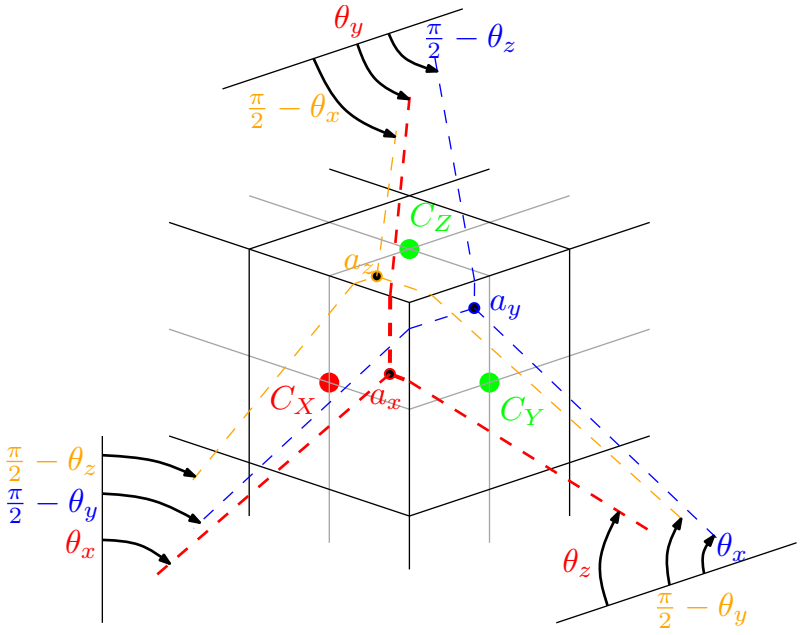


FIG. 4.5 – Using the symmetries to find the coordinates of the points of the overlapping windows near a corner.

tight frame (see section 4.2.4), the normalization coefficients correspond to the square root of the redundancy in each scale, which is by definition the number of coefficients (size of the band) divided by the number of data-points. In this case it corresponds to the energy of the band in Fourier domain, since we implement a paving of this domain. The Meyer filters shown in figure 4.2 can be idealized by filters which are stepwise constant with exactly the same energy, and are represented in figure 4.7. Now the problem reduces to to calculate the areas of such filters in any dimension. We denote by $V_{\psi_0}^d$, V_{ψ}^d, V_{ϕ}^d the volume of the $\hat{\psi}_0, \hat{\psi}_n, \hat{\phi}_n$ filters in d-dimensions, relative to the size of the Fourier domain. According to figure 4.7, $V_{\psi_0}^1 = 2 \cdot (1/2 - 3/16) = 5/8$, because of the symmetry with respect to the zero frequency, $V_{\psi}^1 = 2 \cdot 3/16 = 3/8$, and $V_{\phi}^1 = 2 \cdot 3/8 = 3/4$, since this scale is further decimated by a factor two in each dimension, we consider that it spans from 0 to 3/8 in terms of frequency. In two dimensions, the Meyer wavelets tiling is represented in figure 4.8, and the related volumes are $V_{\psi_0}^2 = 1 - (3/8)^2$, $V_{\psi}^2 = (3/4)^2 - (3/8)^2$, and $V_{\phi}^2 = (3/4)^2$. A simple analysis of the shape of the 2D domains and their extensions in more dimensions lead to these volumes in d-dimensions :

$$V_{\psi_0}^d = 1 - (3/8)^d, \tag{4.8}$$

$$V_{\psi}^d = (2^d - 1)(3/8)^d, \tag{4.9}$$

$$V_{\phi}^d = (3/4)^d. \tag{4.10}$$

From the Meyer wavelets to the curvelets is only the angular separation in k directions, all of which extract an equivalent volume of the available Fourier domain, $V_c = V_{\psi}^d/k$. Nevertheless, curvelets are not undecimated by keeping the total Fourier volume V_{F_j}, but wrapped inside a volume V_{w_j}, and the coefficients are separated in real and imaginary

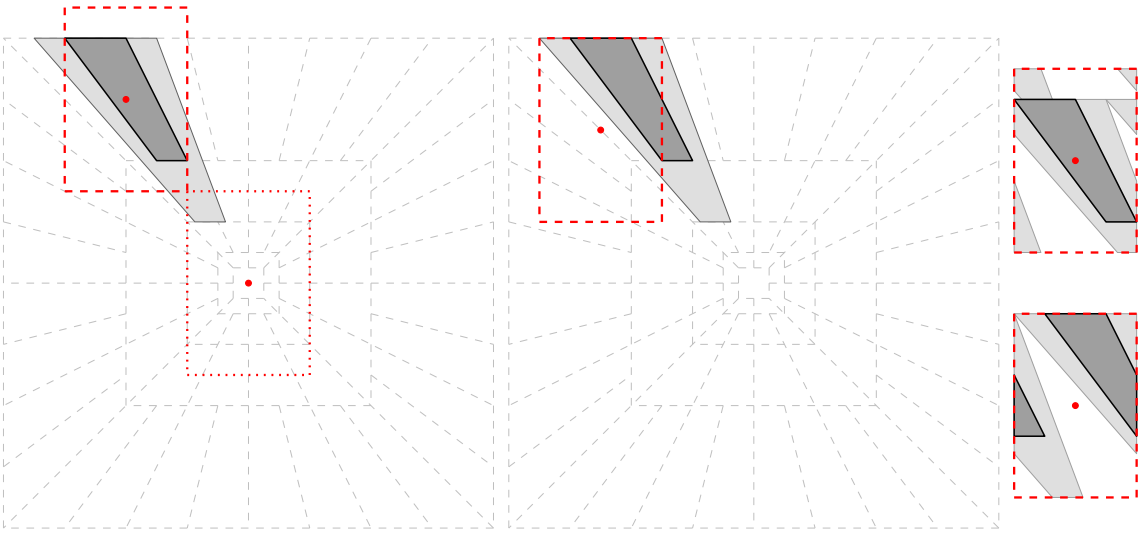


FIG. 4.6 – (a) Representation of the influence of a wedge and its overlapping region. The centered dotted rectangle corresponds to the minimal size in which the wedge will be wrapped. (a') The result of the wrapping. (b) The wrapping in a rectangle of the same size, but whose center is chosen such that the zero frequency (big dot) falls outside the support of the wedge. (b') The corresponding wedge after translation and wrapping.

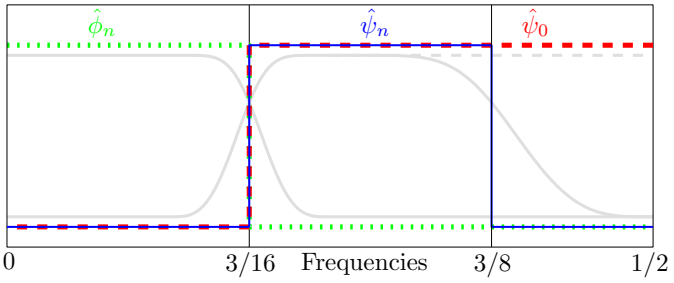


FIG. 4.7 – Idealized Meyer filters in Fourier domain. The dotted green filter $\hat{\phi}_n$ represents the general low-pass filter, it spans from 0 to 3/16. The plain blue line filter $\hat{\psi}_n$ represents the general high-pass filter which spans from 3/16 to 3/8, and the dashed red one $\hat{\psi}_0$ is the finest scale modified filter, from 3/16 to 1/2.

parts, which lead to normalization coefficients for each band which are

$$\sigma^d_{\psi_0} = \sqrt{\frac{1}{2}\frac{V^d_{\psi_0}}{k}\frac{V_{F_0}}{V_{w_0}}}, \tag{4.11}$$

$$\sigma^d_{\psi_j} = \sqrt{\frac{1}{2}\frac{V^d_{\psi}}{k}\frac{V_{F_j}}{V_{w_j}}}, \tag{4.12}$$

$$\sigma^d_{\phi} = \sqrt{(3/4)^d}. \tag{4.13}$$

These coefficients permit to apply the correct thresholds when using this transform in presence of additive white gaussian noise.

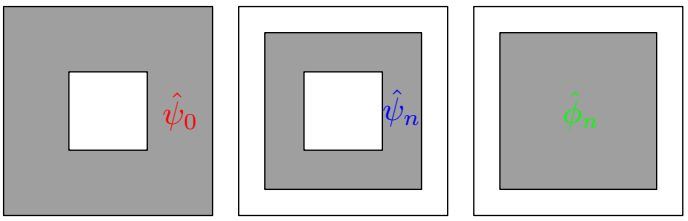


FIG. 4.8 – Idealized 2D Meyer filters in Fourier domain. From left to right : area of influence of the the finest scale filter, the general fine scale filter, and coarse scale filter.

4.2.3 Redundancy

In this section, we shall quantify analytically the redundancy reduction induced by our new FCT in any dimension d. Without loss of generality, we assume that the data object is a d-D hypercube of unit side. Let N_a be the number of angles along an edge on one face at the finest scale, for a total of N_a^{d-1} orientations on each face. Let $N_f = 2d$ be the number of faces of the d-D data hypercube.

The redundancy of the Cartesian coronization or multiscale separation assuming a dyadic frequency tiling is given by

$$\sum_{j=0}^{J} \left(\frac{1}{2^d}\right)^j, \tag{4.14}$$

which is upper-bounded by $R_w = \frac{2^d}{2^d-1}$. As explained in Section 4.2.2, an extra-redundancy R_{add} comes into play in the Meyer wavelet transform with the original FCT implementation, while this is avoided by ours :

$$R_{add} = \begin{cases} \left(\frac{4}{3}\right)^d & \text{Original FCT,} \\ 1 & \text{Proposed FCT.} \end{cases} \tag{4.15}$$

At the finest scale and on each face, there are N_a^{d-1} wedges, where the size of each of them is

$$\frac{3}{8} \times \underbrace{\frac{2}{N_a} \times \ldots \times \frac{2}{N_a}}_{d-1 \text{ times}}. \tag{4.16}$$

The factor 3/8 corresponds to the radial depth of the scale ; see Section 4.2.2 and Figure 4.2. In the other orthogonal directions, a wedge has a size of $\frac{1}{N_a}$ which we double because of overlapping. The redundancy of a face at the finest scale is then

$$R_f = N_a^{d-1} \cdot \frac{3}{8} \left(\frac{2}{N_a}\right)^{d-1} = 3 \cdot 2^{d-4}. \tag{4.17}$$

As it can be seen, the R_f redundancy is independent of N_a, and is therefore valid at all scales. For a large enough number of scales, it can be reasonably assumed that coarsest (wavelet) scale has the same redundancy as a curvelet subband at the same scale. Consequently, the overall redundancy of the FCT is upper-bounded by (see (4.14))

$$\begin{aligned} R &= N_f \cdot R_f \cdot R_w \cdot R_{add} \\ &= 3d \frac{2^{2d-3}}{2^d - 1} \cdot R_{add}. \end{aligned} \tag{4.18}$$

	Original FCT		Proposed FCT	
	C	W	C	W
2-D	7.11	3.56	4.00	2.00
3-D	24.38	5.42	10.29	2.29

TAB. 4.1 – Redundancy of the original FCT and the proposed one in 2-D and 3-D, when wavelets (W) or curvelets (C) are used at the finest scale.

In the case where wavelets are used instead of curvelets at the finest scale, the redundancy upper-bound is changed to

$$\begin{aligned} R' &= (N_f \cdot R_f \cdot (R_w - 1) + 1) \cdot R_{add} \\ &= \left(3d\frac{2^{d-3}}{2^d - 1} + 1\right) \cdot R_{add}. \end{aligned} \tag{4.19}$$

Table 4.1 compares numerically the redundancy of the original and the proposed FCT in 2-D and 3-D, when wavelets (W) or curvelets (C) are implemented at the finest scale. It may be worth mentioning that for the practitioner, the memory storage requirement of the original FCT (as implemented in Curvelab) is twice larger than the one predicted by the redundancy formula. Indeed, the original curvelets are complex and real curvelets are obtained by hermitian symmetry. This explains the redundancy 40[4] claimed by [99] emphasizing the need of lower redundancy curvelets.

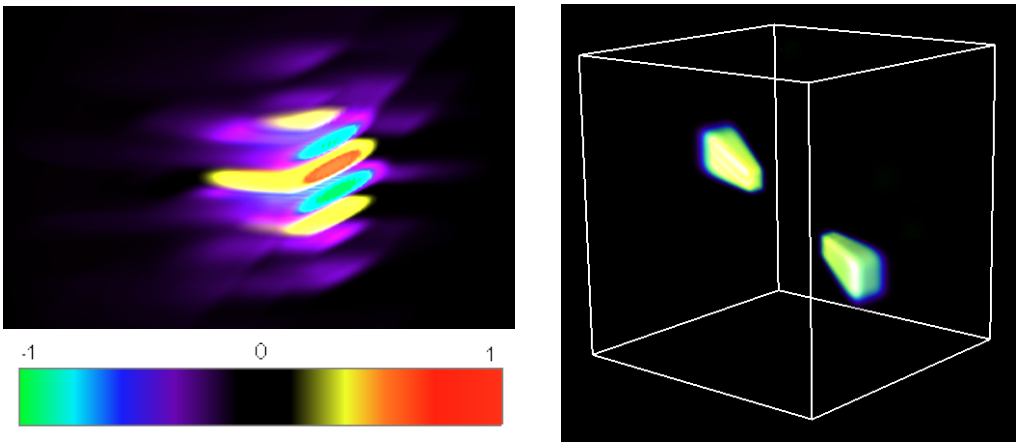


FIG. 4.9 – Left : volume rendering of a 3-D curvelet atom in the spatial domain corresponding to our implementation, cut by a vertical plane to see its inner structure. Right : the magnitude if its Fourier transform. The colorbar scale is valid only for the left image.

4.2.4 Properties

This section enumerates the main properties enjoyed by the FCT according to our new implementation.

– **Reduced redundancy** : with a reduction factor of $(4/3)^d$ compared to the original version. This is one of the distinctive properties of the proposed FCT, and was the main goal underlying our modifications in the first place. For example, our FCT with

[4] In fact, it should be ≈ 50 as can be read from Table 4.1.

full curvelets at all scales is (almost) as redundant as the original one with wavelets at the finest scale. In 3-D, redundancy implied by our implementation is 2.5 times lower that the original FCT with curvelets at all scales. In short, our implementation achieves a low redundancy while maintaining the directional selectivity property at the finest scale unlike the original FCT where wavelets are advocated at the finest scale to lower the redundancy [170, 18].

- **Isometry and fast exact reconstruction** : owing to the uniform partition property of the Meyer wavelets, and the coverage property (4.2) of the angular window, the collection of curvelets in Fourier obtained by multiplication of the scale and angular windows also ensures a uniform partition of the unity. Therefore, with a proper normalization of the FFT (wrapping is a simple reindexing), the proposed FCT corresponds to a Parseval tight frame (PTF), i.e. the frame operator $CC^* = I$, where C^* is the FCT analysis operator and C its adjoint. With the PTF property, C turns out to be also the inverse transform operator associated to a fast reconstruction algorithm (each step of the forward transform is easily invertible).
- **Parabolic scaling** : by construction, the curvelets obey the parabolic scaling law with one short and two long sides $\sim (2^{-j}, 2^{-j/2}, 2^{-j/2})$. Although at the finest scale, this property is less faithful to the continuous construction compared to the original FCT (see also the discussion at the end of Section 4.2.2).
- **Non-equal ℓ_2 norm atoms** : although our FCT implements a PTF, the modified curvelets at the finest scale do not have the same ℓ_2 norm as the curvelets in the other (coarsest) scales. These ℓ_2 norms can nonetheless be calculated analytically so as to normalize the associated curvelets coefficients, which is important for instance in every processing which involves thresholding.
- **Guaranteed zero-mean subbands** : this is a consequence of the wise translation trick prior to wrapping explained in Section 4.2.2. Of course, this operation preserves isometry and ℓ_2 norms.

4.2.5 Comparison with Curvelab

In the sequel, the new variant of FCT is dubbed LR-FCT, where LR stands low-redundancy. In this section, we compare the 3-D LR-FCT to the original FCT as implemented in the CurveLab toolbox in terms of denoising performance over several datasets, where the noise is additive white Gaussian (AWGN). The goal is to assess experimentally how the LR-FCT version positions itself with respect to the two variants of the CurveLab FCT depending whether wavelets or curvelets are used at the finest scale.

For fair comparison, the three transforms were applied in exactly the same conditions. Our goal was not to pick up a very sophisticated thresholding/shrinkage estimator operating in the transform domain (the literature is very rich on the subject), but rather to get a fair picture about differential performance of the transforms themselves with a simple and fast statistical estimator. This is why we opted for hard thresholding with the popular 3σ threshold, where σ is the AWGN standard deviation[5]. The experiments have been carried out on different sorts of data (3-D space simulation, hyperspectral images, still and fast-moving videos), and on ten noise realizations for each triplet transform-dataset-standard deviation. Figure 4.10 depicts the evolution of the output peak signal-to-noise ratio (PSNR) as a function of the noise standard deviation (translated in PSNR). One can see that the LR-FCT implementation is not as good as the full redundant CurveLab version, but pro-

[5]The curvelet coefficients are normalized properly by the ℓ_2 norms of the associated curvelets so that this threshold is valid at all subbands. See also previous section.

vides a very good tradeoff between redundancy and the denoising performance as measured by PSNR with less than 1dB lower than the CurveLab FCT (with curvelets at all scales) in the least favorable case, and 2-4 dB better than the CurveLab variant using wavelets at the finest scale. As stated in Section 4.2.2, we believe that the difference in performance between the all-curvelets original FCT and the LR-FCT can be explained by the more oscillating behavior of the LR-FCT atoms at the finest scale.

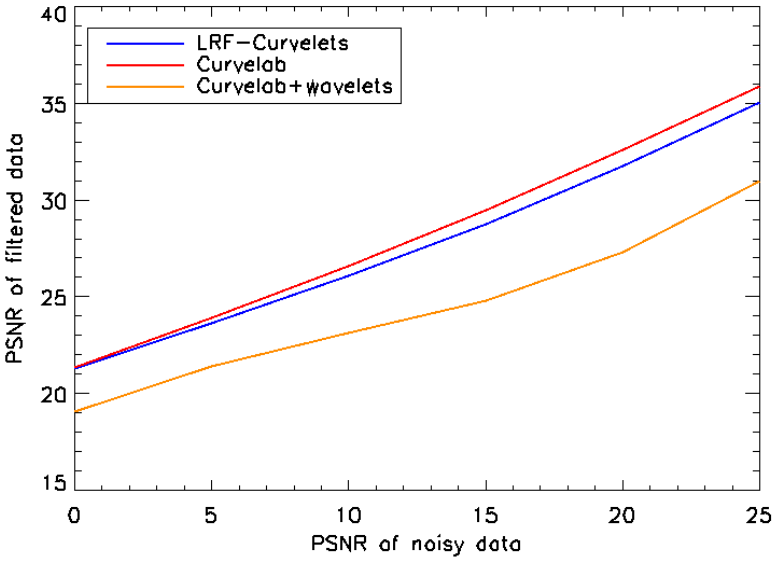


FIG. 4.10 – Mean denoising PSNR versus noise level (as measured by the input PSNR) using different FCT implementations. The denoising PSNR was averaged over ten noise realizations and several datasets. The LR-FCT is in blue. Original FCT implementation of Curvelab using curvelets (red) and wavelets (orange) at the finest scale.

The following sections show a wide range of applications, and do not aim to give state-of-the-art results, but instead concentrate on demonstrating how the new transform can be harnessed easily and directly to a variety of problems with reasonable – and beyond – results. Further work will be required to fine-tune these methods and optimize their performance in each specific application. All results reported in these sections, especially those involving videos, are available in full resolution online at `http ://www.morphologicaldiversity.org` awaiting inspection by the interested reader.

4.3 LR-FCT denoising : a good tradeoff between efficiency and memory storage

In this experiment, we compare the denoising performance using the LR-FCT to several other 3-D multiscale transforms on various types of datasets (3-D spatial data, hyperspectral images and videos). Again, the noise is AWGN and simple hard thresholding is used. The video datasets included in our experiments were the standard videos used in [] as well as *mobile*, *tempete*, and *coastguard* CIF sequences available at `www.cipr.rpi.edu`. For hyperspectral data, we used dataset from the OMEGA spectrometer on Mars Express (`www.esa.int/marsexpress`) with 128 wavelength from $0.93\mu m$ to $2.73\mu m$. Beside LR-FCT, the other transforms involved in this comparative study are : the dual-tree complex

wavelet transform [90, 137], the surfacelet transform [99] and the orthogonal (decimated) and translation-invariant (undecimated) wavelet transforms.

Figures 4.11 and 4.12 show the output PSNR after denoising as a function of the input PSNR for each transform. Each point on each curve is the average output PSNR on ten noise realizations. The reader may have noticed that the wavelet results are much better here than those tabulated in [99]. The reason behind this is that unlike those authors, we used the Cohen-Daubechies-Fauveau 7/9 filterbank, which is much better for denoising. Figure4.12 displays the results for the hyperspectral data from Mars Express (see caption for details). From these experiments, it ca be clearly seen that the proposed LR-FCT compares very favorably to the other multiscale geometrical 3-D transforms, and is particularly better at the low PSNR regime. In a nutshell, it can be safely concluded that LR-FCT provides a very good compromise between denoising performance and memory/CPU requirements.

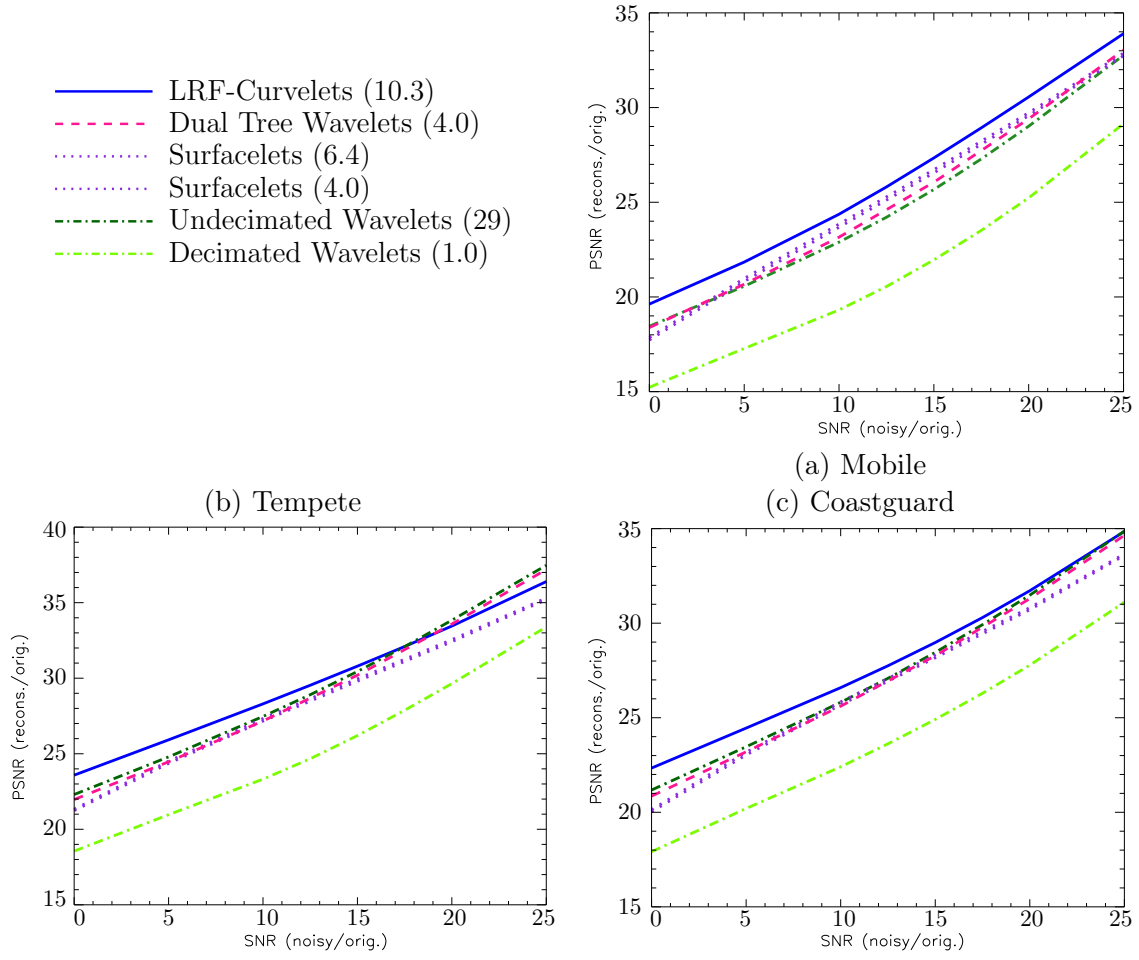


FIG. 4.11 – Output PSNR as a function of the input PSNR for three video sequences. (a) *mobile*, (b) *tempete*, and (c) *coastguard* CIF sequence. The redundancy of each transform is indicated in parentheses on the legend.

4.4 Inpainting

Inpainting is to restore missing data information based upon the still available (observed) cues from destroyed, occluded or deliberately masked subregions of the data. In-

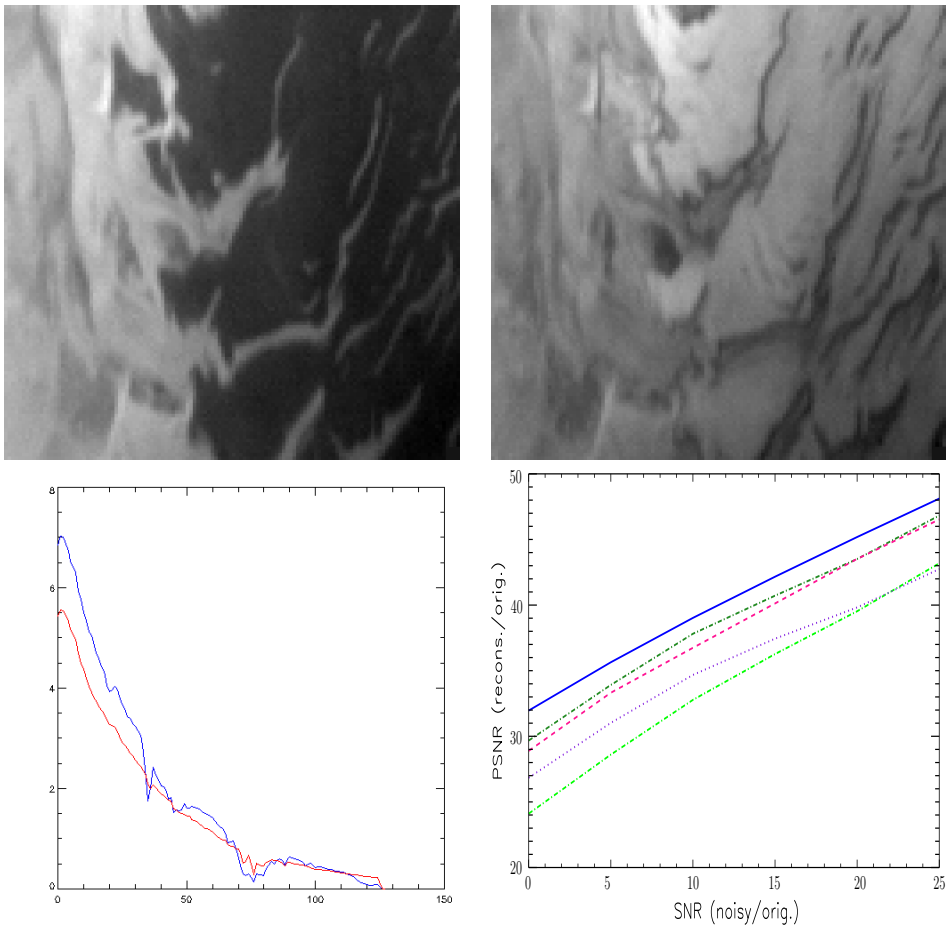


FIG. 4.12 – Top row : Mars Express observations at two different wavelengths. Bottom-left : two spectra at two distinct pixels. Bottom-right : output PSNR as a function of the input PSNR for different transforms ; see Figure 4.11 for legend of the curves.

painting has received considerable interest and excitement and has been attacked using diffusion and transport PDE/Variational principles, non-local exemplar region fill-in and sparsity-based regularization ; see e.g. [58, 61] and references therein.

Let $f \in \mathbb{R}^N$ be a vectorized form of the sought after 3-D data cube which is $\sqrt{N} \times \sqrt{N} \times \sqrt{N}$, and $\mathrm{M} \in \{0,1\}^{P \times N}$, $P < N$ be a binary rectangular matrix where each of its rows is zero except at the entry where a voxel is not missing. The observed (incomplete) data g is then the result of applying the lossy operator M to f :

$$g = \mathrm{M}f + \varepsilon \ .$$

where ε is some noise of finite variance σ^2 that may contaminate the observed values. Restoring f from g is an ill-posed which necessitates some form of regularization to reduce the space of candidate solutions. Here, we promote solutions that are sparse in some prescribed overcomplete dictionary of atoms $\Phi \in \mathrm{R}^{N \times L}$, $L \geq N$, meaning that $x := \Phi\alpha$ (a synthesis prior) can be sparsely represented to a high accuracy by a few number of atoms in Φ. Put formally, we are seeking to solve the following optimization problem :

$$\min_{\alpha \in \mathbb{R}^L} \|\alpha\|_0 \text{ s.t. } \|g - \mathrm{M}\Phi\alpha\|_2 \leq \epsilon(\sigma) \ , \tag{4.20}$$

where $\|\cdot\|_0$ is the ℓ_0 pseudo-norm that counts the number of nonzero entries of its argument, and $\epsilon(\sigma)$ is the constraint radius that depends on the noise variance. This is very challenging NP-hard optimization problem, and one has to resort to alternative formulations or greedy algorithms to attempt to solve it. For instance, convex ℓ_1 relaxation could be used instead of the ℓ_0 penalty.

Here, we use the algorithm devised in [58] which can be viewed as a stagewise hybridization of matching pursuit with block-coordinate relaxation. The adjective "stagewise" is because their algorithm exploits the fact that the dictionary is structured (union of transforms $\Phi = [\Phi_1, \cdots, \Phi_K]$) with associated fast analysis and synthesis operators Φ_k^{T} and Φ_k ; see [58, 148] for details. For the reader convenience, Algorithm 12 recalls the main steps of this inpainting algorithm.

Algorithm 12: Inpainting Algorithm.

Data: Observed data g and mask M.
Input: Dictionary $\Phi = [\Phi_1 \cdots \Phi_K]$, number of iterations T_{iter}, final threshold τ (e.g. 3).
begin
  Initial components $f_k^{(0)} = 0, k = 1, \cdots, K$.
  Initial residual $r^{(0)} = g$.
  Initial threshold : let $k^\star = \arg\max_k \|\Phi_k^{\mathrm{T}} g\|_\infty$, set $\lambda_0 = \max_{k \neq k^\star} \|\Phi_k^{\mathrm{T}} g\|_\infty$.
  for $t = 1$ **to** T_{iter} **do**
    for $k = 1$ **to** K **do**
      Compute marginal residuals $r_k^{(t)} = r^{(t-1)} + f_k^{(t-1)}$.
      Update kth component coefficients by thresholding
      $\alpha_k^{(t)} = \mathrm{Thresh}_{\lambda_{t-1}} \left(\Phi_k^{\mathrm{T}} r_k^{(t)} \right)$.
      Update kth component $f_k^{(t)} = \Phi_k \alpha_k^{(t)}$.
    Update the inpainted data $f^{(t)} = \sum_{k=1}^K f_k^{(t)}$.
    Update the residuals $r^{(t)} = g - \mathrm{M} f^{(t)}$.
    Update the threshold $\lambda_t = \lambda_0 - t \left(\lambda_0 - \tau\sigma \right) / T_{\mathrm{iter}}$.
end
Result: The estimate $f^{(T_{\mathrm{iter}})}$ of f.

In the following experiments, this algorithm was used with a dictionary containing two transforms : the LR-FCT and the undecimated discrete wavelet transform (UDWT), in order to better take into account the morphological diversity of the features contained in the data. Figure 4.13 shows the inpainting result on a synthetic cerebral MRI volume available on BrainWeb [128] at `http ://www.bic.mni.mcgill.ca/brainweb/` with two masks : 80% random missing voxels, and 10% missing z slices. We can see that even with 80% missing voxels, we can still see incredibly faint details in the restored anatomical structures such as in the sulci and the cerebellum.

For video inpainting, we artificially created a mask containing 5% random missing pixels on each frame, a missing line, three clusters of 12 dead pixels each, two consecutive missing frames, and a rotating swirl two-pixel wide. The top row of Figure 4.14 shows the central frame of the original video, as well as one of the two consecutive missing frames and one frame of the masked video. The inpainting of these missing areas was performed either with a dictionary containing the LR-FCT alone, or another dictionary containing

both the LR-FCT and the UDWT. The results are displayed respectively in the middle and bottom rows of Figure 4.14. While inpainting using the LR-FCT alone or combining it with wavelets give similar PSNR values, visually, inpainting with both transforms yields a smoother result.

4.5 Video De-interlacing

Video is often acquired in interlaced format, meaning that every even frame contains only the even lines and similarly for odd frames. The problem of reconstructing the missing lines in the sequence is called de-interlacing and has been addressed with many dedicated methods : using simple spatio-temporal neighborhood linear interpolation [158, 79], using a directional interpolation to better preserve the edges in the images [57, 171, 120], with more complicated methods relying on motion estimation and compensation [164, 169, 78], and recently with variational inpainting [4]. In this section, it will be demonstrated that adapting the idea of sparsity-based inpainting as described in Section 4.4, and introducing a proper regularization that takes into account the special structure of the mask, will yield very promising de-interlacing results.

De-interlacing could be naively viewed as a 3-D inpainting problem, where the mask is nonzero at the observed even/odd lines in the consecutive even/odd frames ; see Figure 4.15 for a graphical illustration. However, the major issue that rises with such a mask is that it exhibits a strong coherence with some curvelet atoms[6]. Indeed, curvelets are plate-shaped atoms, and those oriented at 45 degrees are very correlated with the inclined planes formed by the alternating even-odd lines along the time axis. One could think of excluding such atoms from the curvelet dictionary (e.g. by just zeroing their coefficients after the forward FCT in the above inpainting algorithm), or weight the FCT coefficients differently depending on those of the mask. These solutions were tried but the results were not conclusive.

Following this discussion, we chose to introduce a regularization term acting directly in the spatial domain. In fact, we implicitly assume that the video to be restored does not contain structures that correlate strongly with the mask, otherwise there is no chance that such structures will be identifiable. Thus, a wise regularization would penalize formation of singularities (edges) in the de-interlaced video that coincide with those of the mask, which is also consistent with the way the correlation between the mask and the curvelets manifests itself. Put formally, the de-interlacing optimization problem to solve writes

$$\min_{\alpha \mathbb{R}^T} \|\alpha\|_0 + \mu \|\nabla(\Phi\alpha) \odot \nabla f_{\text{mask}}\|_1 \text{ s.t. } \|g - \text{M}\Phi\alpha\|_2 \leq \epsilon(\sigma), \tag{4.21}$$

where f_{mask} is the video representing the mask (Figure 4.15) vectorized in R^N, ∇ is the spatio-temporal discrete gradient operator that produces a vector field in $\mathbb{R}^{N\times 3}$, $\odot$ is the Hadamard product, and for $u = (u_x, u_y, u_t) \in \text{R}^{N\times 3}$, $\|u\|_1 = \sum_i |u[i]|$ where $|u[i]| = \sqrt{u_x[i]^2 + u_y[i]^2 + u_t[i]^2}$. It is not difficult to see that the above penalty can be rewritten as a weighted TV semi-norm :

$$\|\nabla(\Phi\alpha) \odot \nabla f_{\text{mask}}\|_1 = \|\text{W}_{\text{mask}} (\nabla(\Phi\alpha))\|_1 = \|\Phi\alpha\|_{\text{TV,W}_{\text{mask}}} ,$$

where W_{mask} is a (diagonal) weighting operator whose entries are given by ∇f_{mask}, and which acts on vector fields in $\mathbb{R}^{N\times 3}$ by component-wise multiplication by its entries. This yields yet another nice interpretation of our penalty, i.e. promote candidate solutions that

[6] A similar complicating coherence phenomenon occurs also in super-resolution.

	DOI [171]	STELA [120]	STIM [4]	LR-FCT
Stockholm	31.48	30.79	**37.04**	34.17
Shields	35.30	32.00	34.62	**35.34**

TAB. 4.2 – PSNR the de-interlacing results on the video sequences *stockholm* and *shields* using our method and three other ones from the literature.

are smooth at the locations where the mask changes abruptly, which is exactly the goal we were pursuing.

The resulting optimization problem we end up with is even more challenging that the original inpainting problem. Inspired by the work in [152] and [58], we slightly modified Algorithm 12 by incorporating a (sub)gradient-descent correction step with respect to this penalty, namely by adding a step of the form :

$$f^{(t)} \leftarrow f^{(t)} + \mu\gamma \mathrm{div}\left(\mathrm{W}_{\mathrm{mask}} G\left(\mathrm{W}_{\mathrm{mask}}(\nabla f^{(t)})\right)\right)$$

just after the update of $f^{(t)}$, where $G(.)$ is a subgradient of the ℓ_1 norm in $\mathbb{R}^{N\times 3}$, and γ is the descent step-size. Although this update is quite heuristic, the obtained algorithm works fairly well in practice. Further work is needed to solve exactly the above problem (or rather its convex relaxation) with theoretical guarantees. We will be left to a future work.

Figures 4.16, 4.17, 4.18, 4.19 show close-ups of the de-interlacing results on the test sequences *stockholm* and *shields*, and compare them the methods proposed in [171, 4, 120]. We will refer to them respectively as DOI, STIM, and STELA. The quantitative PSNR results are grouped in Table 4.2. Although the proposed method is not intended to compete with methods specifically designed for de-interlacing in terms of computational cost, this experiment at least provides a proof of concept that sparsity-based regularization with the LR-FCT transform is efficient for de-interlacing. The results clearly show that the proposed method is at the level of the state-of-the-art both visually and in terms of PSNR. For some videos, it even outperforms the other methods with visually more pleasant quality and less artifacts than dedicated methods (see e.g. Figure 4.17).

4.6 Morphological Component Analysis

The concept of morphological diversity, introduced by Starck *et al.* [151, 152] is based on the fact that complex signals are linear superposition of more elementary signals (the so-called morphological components). There, such complex signals cannot be efficiently (sparsely) represented with a dictionary built upon a single transform, but rather but rather with an overcomplete dictionary built by amalgamating several sub-dictionaries. Once the sub-dictionaries are identified, solving a sparse decomposition problem in such an overcomplete dictionary yields as a by-product a separation of the structural/morphological components. The Morphological Component Analysis (MCA) algorithm proposed [151] was proposed to achieve this goal.

A key to the success of this separation process is a good choice of the sub-dictionaries. Each sub-dictionary has to be chosen such that its atoms match the shape of the (geometrical) structures they are intended to sparsify, while leading to a non-sparse (or at least not as sparse) representation on the other signal content. For instance, the isotropic wavelet transform [148] is a very good candidate for representing isotropic singularities. On the other hand, the curvelet transforms have plate-like atoms well adapted for representing

smooth 3-D objects with smooth singularities along smooth surfaces. Using a dictionary built as a union of the 3-D isotropic wavelets and LR-FCT sub-dictionaries, we exemplify the MCA algorithm on a toy 3-D dataset, containing parts of 3-D shells, and 3-D Gaussians. The results in the noiseless and noisy cases are shown in Figures 4.20-4.21.

4.7 Conclusion

We have proposed a new variant of the fast curvelet transform in d-D with a special emphasis on 3-D, lowering its redundancy from 25 to about 11 in 3-D, providing a versatile tool to use in a wide range of inverse problems. We have shown good results in video and spectral data denoising, video and MRI inpainting, sparse component separation, and extended the sparse inpainting approach to the de-interlacing problem. Following the philosophy of reproducible research, a C++/Matlab toolbox implementing the proposed LR-FCT are made available for download at `http ://www.morphologicaldiversity.org`.

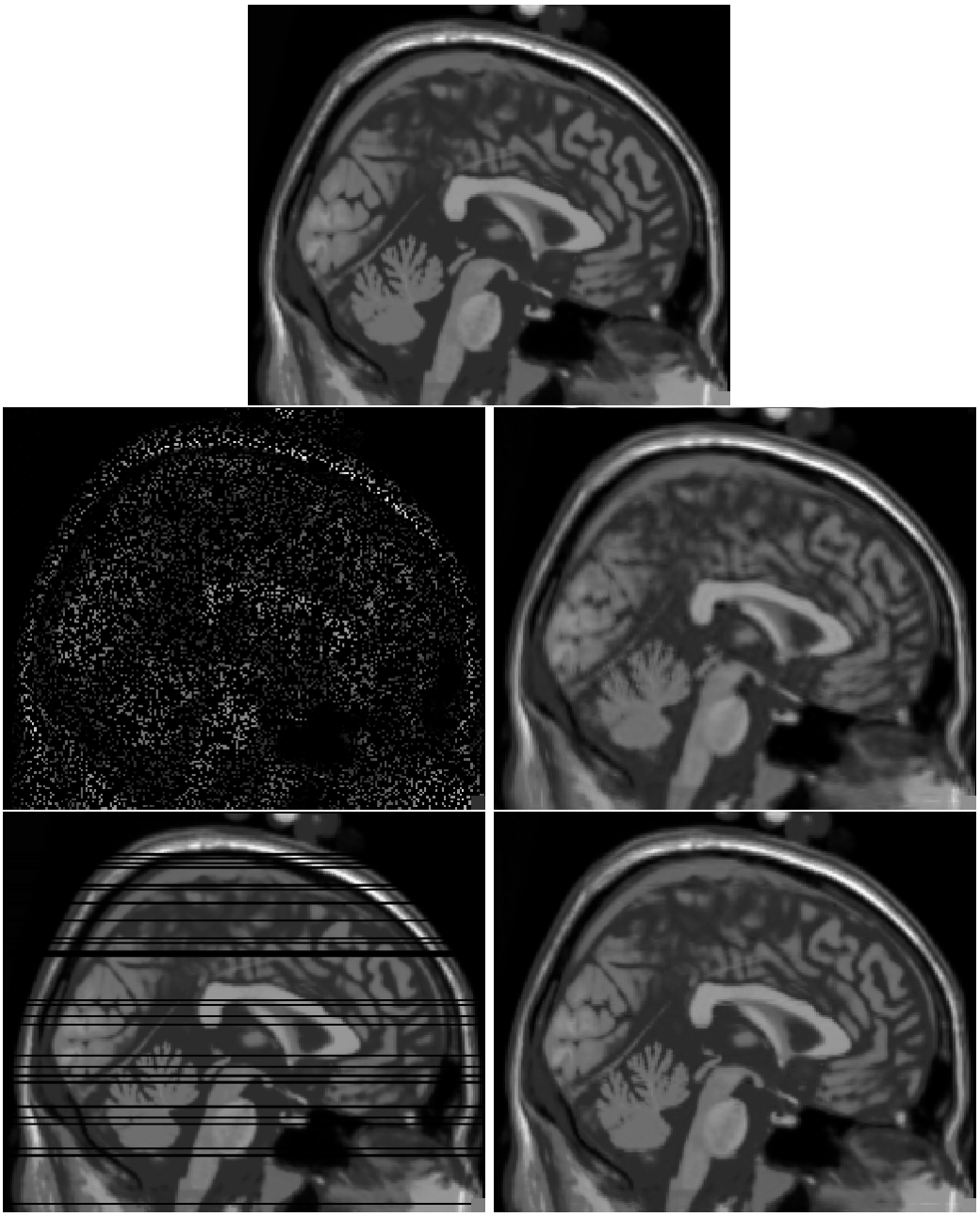


FIG. 4.13 – Top : a sagittal ((y, z)) slice of the original synthetic MRI volume from BrainWeb[128]. Left column : the data with missing areas, random 80% missing voxels, and 10% missing z slices. Right : inpainting results with a LF-FCT+UDWT dictionary.

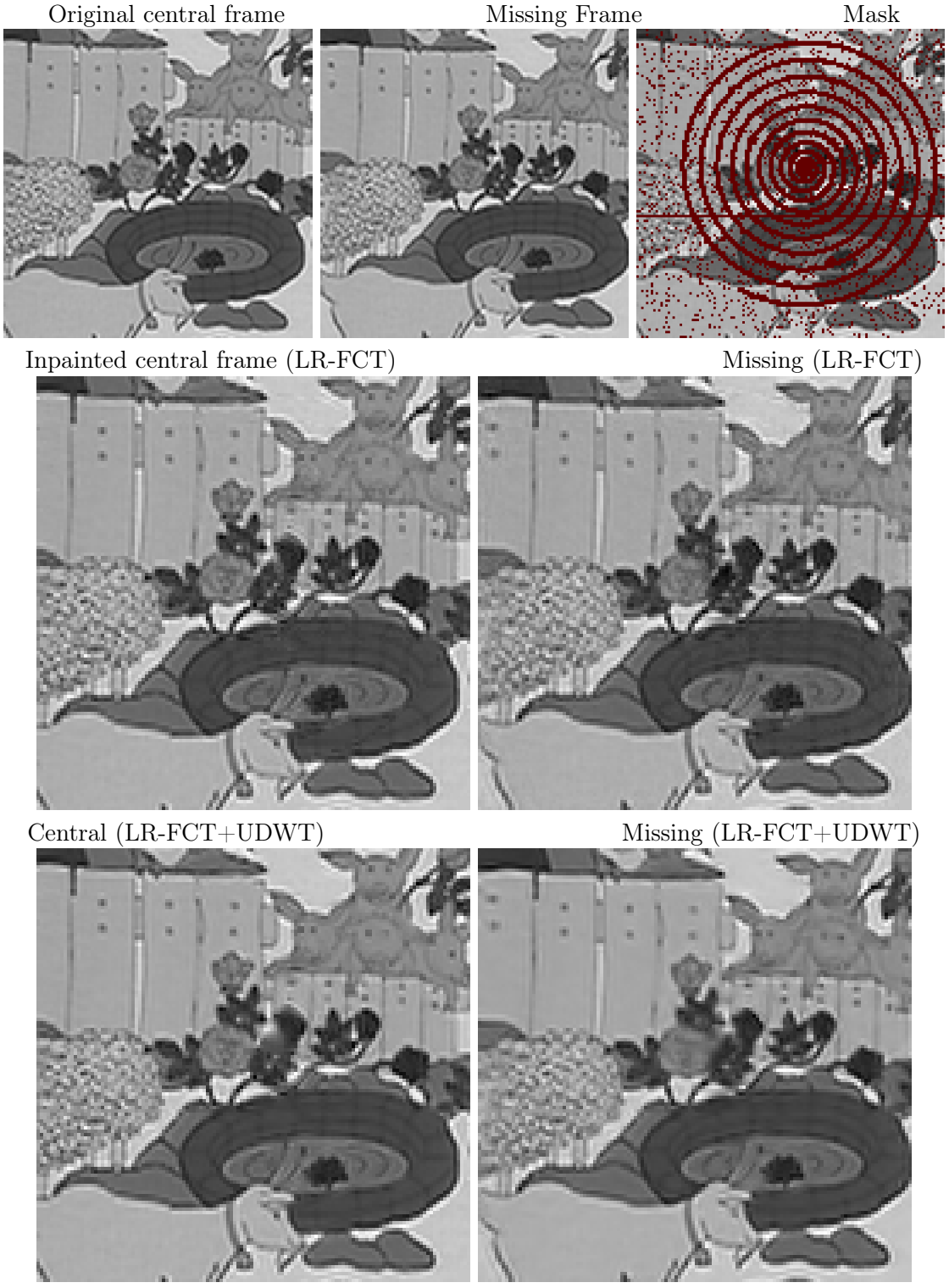


FIG. 4.14 – Inpainting the *mobile* video. Top row : the central frame of the original video, a missing frame, and a frame of the masked video. Middle row : central and missing frames inpainted using only LR-FCT. Bottom row : using LR-FCT and UDWT.

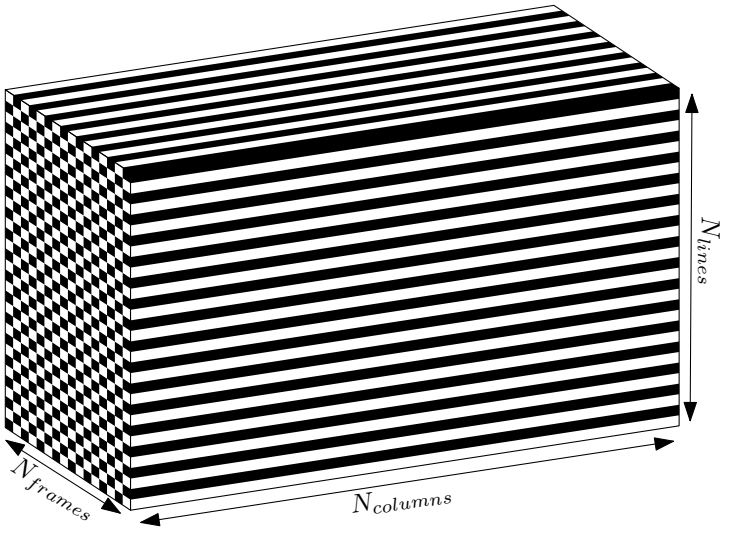


FIG. 4.15 – Example of interlacing mask on 16 frames of a video, black and white indicating known and lost pixels.

FIG. 4.16 – Close-ups of the centeral part of a frame from the *shields* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

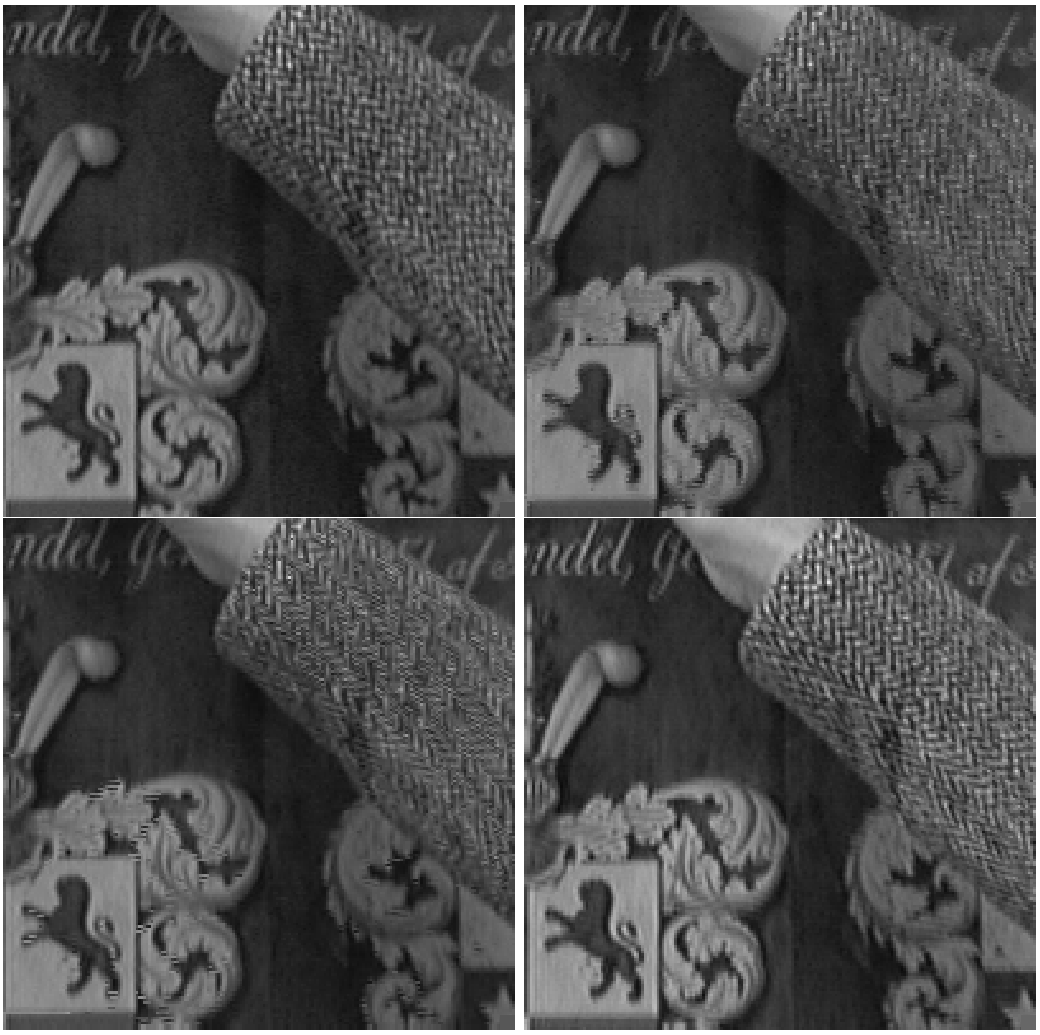


FIG. 4.17 – Close-ups of a fast-moving part of the *shields* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

FIG. 4.18 – Close-ups of the stairs in the *stockholm* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

FIG. 4.19 – Close-ups of a building in the *stockholm* video sequence de-interlaced using different methods. Top : Original, and de-interlaced result using STELA. Bottom : results using STIM and proposed method.

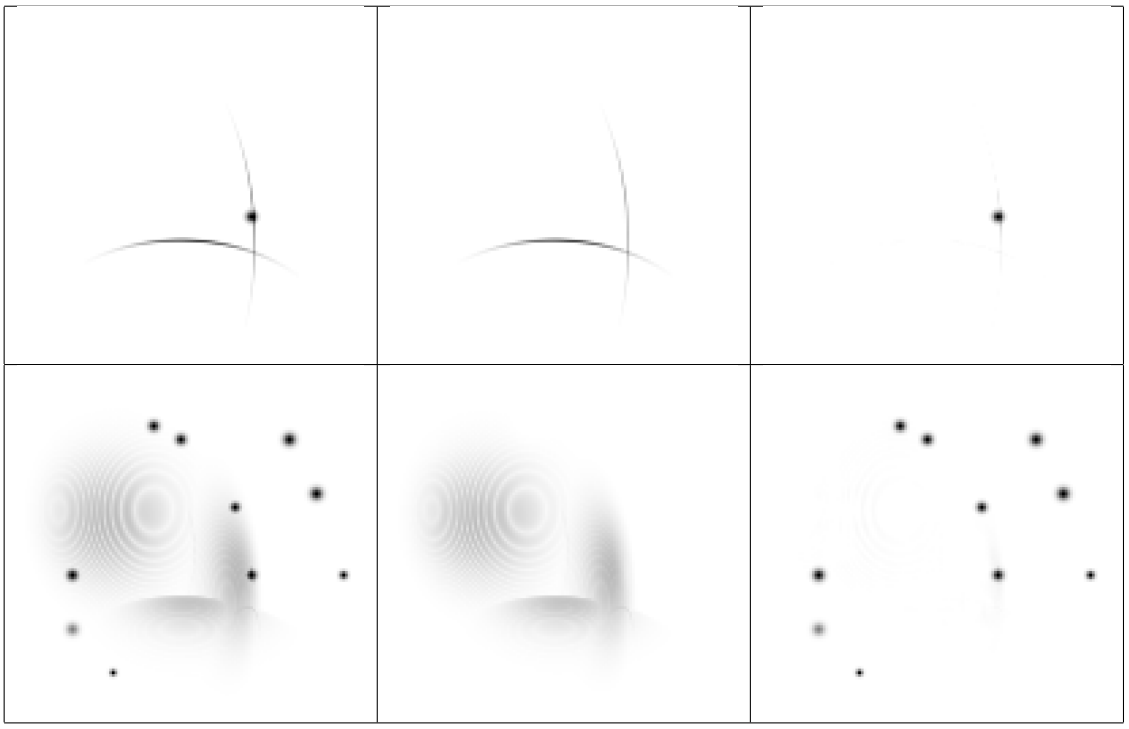


FIG. 4.20 – Morphological component separation results from a data cube containing pieces of 3-D shells and 3-D gaussians. From left to right : original dataset, and morphological components extracted with the LR-FCT and the isotropic wavelet transform. Top : a 2-D slice of each 3-D cube. Bottom : maximum intensity projection view.

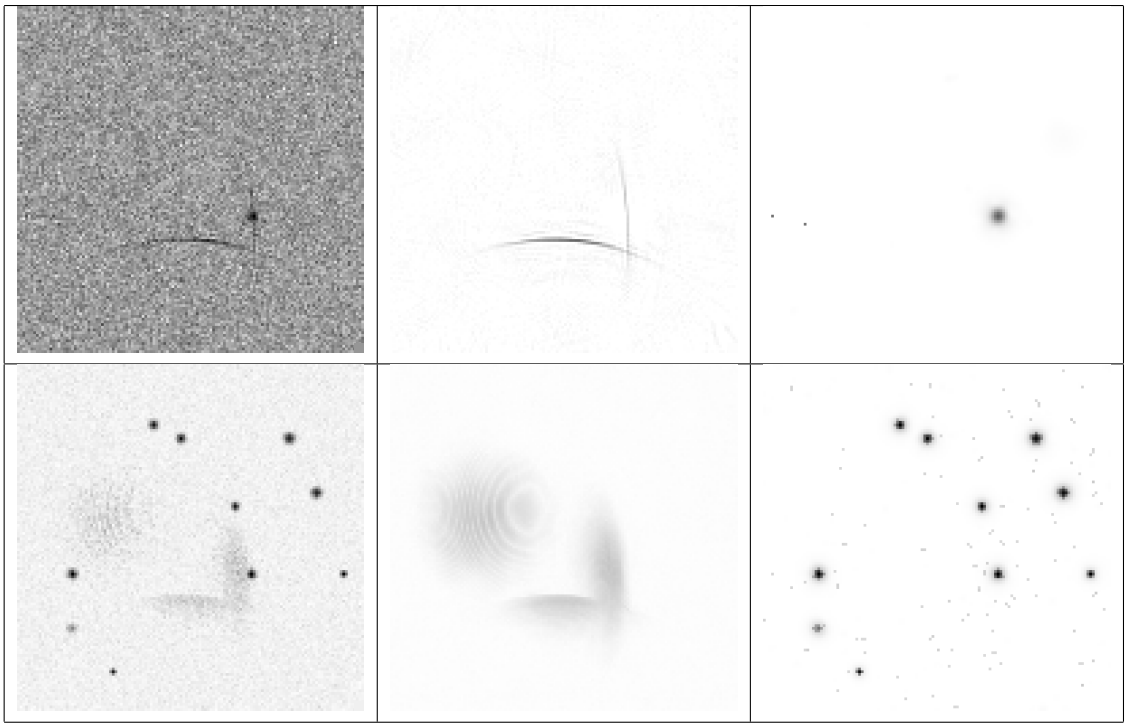


FIG. 4.21 – The same as Figure 4.20, with an additive white Gaussian noise. The two components are jointly separated and denoised.

Chapitre 5

Application en vidéo Infrarouge

Section confidentielle Sagem.

Conclusion et perspectives

Dans ce mémoire, nous avons étudié les représentations parcimonieuses de données bi- et tridimensionnelles, et leurs applications à la restauration de ces données, en insistant sur la restauration de vidéos infrarouges.
Nous avons tout d'abord présenté le contexte de cette thèse, à savoir les traitements et problèmes rencontrés par l'équipe de Sagem du groupe Safran au sein de laquelle s'inscrivait le projet, puis présenté le concept de parcimonie au travers des propriétés mathématiques liées à la théorie de l'approximation, puis montré des exemples concrets de représentations des signaux vivant dans des espaces de différentes dimensions.

Notre première contribution a été d'étendre la transformée en curvelets 2-D première génération à la 3-D, après avoir constaté un manque dans la diversité morphologique des atomes liés aux représentations existantes, à savoir l'absence de représentations adaptées aux filaments. Cette extension a mené à deux nouvelles représentations, les RidCurvelets 3-D et les BeamCurvelets 3-D, qui ont été appliquées avec succès en débruitage et inpainting.
La seconde contribution consiste à fournir aux chercheurs une alternative moins couteuse en mémoire à la représentation en curvelets rapides 3-D proposée par Ying *et al.* [170]. En effet, nous y avons proposé un compromis qualité/redondance sous la forme d'une version deux fois et demi moins redondante, avec seulement une faible perte de performances due à des atomes plus oscillants à l'échelle la plus fine. Nous avons par ailleurs montré que ce type d'outil peut être utilisé pour résoudre de nombreux problèmes avec des algorithmes simples ayant pour cœur le seuillage itératif des coefficients.
Finalement, nous avons commencé à utiliser ces transformées bi- et tridimensionnelles en restauration de vidéos infrarouges, et malgré une utilisation pour le moment faible de la spécificité de ces données, qu'il s'agisse de l'anisotropie due à la direction temporelle ou des données infrarouges, les résultats obtenus sont prometteurs.

Pour conclure, nous avons développé de nouvelles représentations tridimensionnelles, et avons montré leur utilité et complémentarité avec les outils existants.
Afin de tirer le meilleur parti de ces représentations 3-D pour des applications au traitement vidéo, il faudra prendre en compte les propriétés de ce type de données, et optimiser les interactions entre ces outils et les traitements classiques 2-D+temps. Cela peut passer par

une modification du pavage des curvelets rapides pour obtenir des structures plus anisotropes dans une direction privilégiée, ou une approche adaptative dans l'idée des grouplets de Mallat et Peyre [108, 123].
Finalement, pour ce qui est des BeamCurvelets, nous avons montré qu'elles étaient spécialement adaptées aux données cosmologiques à très grande échelle, il est donc envisagé de les adapter au bruit de Poisson pour pouvoir les appliquer sur des relevés de galaxies.

Bibliographie

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD : An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11) :4311, 2006.

[2] D. Alleysson. 30 ans de démosaïcage. *Traitement du Signal*, 21(6) :561–581, 2004.

[3] A. Averbuch, R.R. Coifman, D.L. Donoho, M. Israeli, and J. Wald'en. Fast slant stack : A notion of radon transformform data in a cartesian grid which is rapidely computible, algebrically exact, geometrically faithful and invertible. *SIAM Journal on Scientific Computing*, 2001.

[4] C. Ballester, V. Caselles M. Bertalmío, L. Garrido, A. Marques, and F. Ranchin. An inpainting-based deinterlacing method. *IEEE Transactions on Image Processing*, 16(10) :2476–2491, Oct. 2007.

[5] B.E. Bayer. Color imaging array, Jul. 1976. US Patent 3,971,065.

[6] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1) :183–202, 2009.

[7] Y. Benchekroun and J. Budin. Sk-0000139526-04, spécification technique de besoin du calcul des vitesses dans l'image. Technical report, Sagem, Oct. 2007.

[8] Y Benjamini and Y Hochberg. Controlling the false discovery rate : a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1) :289–300, 1995.

[9] S. Berlemont, A. Bensimon, and J.-C. Olivo-Marin. Detection of curvilinear objects in biological noisy image using feature-adapted fast slant stack. *SPIE conference Wavelets XII, Special Session on Wavelet in Bioimaging*, 2007.

[10] Ch. Bernard. *Wavelets and ill-posed problems : optic flow estimation and scattered data interpolation*. PhD thesis, Ecole Polytechnique, France, 24th Nov. 1997.

[11] J. Bobin, Y. Mudden, J. Fadili, and J.-L. Starck. Morphological diversity and sparsity for multichanel data restoration. *Journal of Mathematical Imaging and Vision*, 33(2) :149–168, 2009.

[12] J. Bobin, J.-L. Starck, and R. Ottensamer. Compressed sensing in astronomy. *IEEE Journal of Selected Topics in Signal Processing*, 2, 2008. 718–726.

[13] J. Bobin, J.L. Starck, J. Fadili, and Y. Moudden. Sparsity and morphological diversity in blind source separation. *IEEE Transactions on Image Processing*, 16(11) :2662–2674, November 2007.

[14] S. Borman and R. Stevenson. Spatial resolution enhancement of low-resolution image sequences - a comprehensive review with directions for future research. Technical report, University of Notre Dame, 1998.

[15] A.M. Bruckstein and M. Elad. A generalized uncertainty principle and sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 48(9) :2558–2567, September 2002.

[16] J. Budin. Sk-0000141435-01, spécifiaction technique de besoin du calibrage sur scène et de la détection des pixels défectueux sur scène. Technical report, Sagem, June 2006.

[17] E.J. Candès. *Ridgelets, theory and applications.* PhD thesis, Stanford University, 1998.

[18] E.J. Candès, L. Demanet, D.L. Donoho, and L. Ying. Fast discrete curvelet transforms. *SIAM Multiscale Model. Simul.*, 5(3) :861–899, 2006.

[19] E.J. Candès, L. Demanet, D.L. Donoho, and L. Ying. Fast discrete curvelet transforms. *Multiscale Modeling and Simulation*, 5(3), 2006. 861–899.

[20] E.J. Candès and D.L. Donoho. Curvelets – a surprisingly effective nonadaptive representation for objects with edges. In A. Cohen, C. Rabut, and L.L. Schumaker, editors, *Curve and Surface Fitting : Saint-Malo 1999*, Nashville, TN, 1999. Vanderbilt University Press.

[21] E.J. Candès and D.L. Donoho. Ridgelets : the key to high dimensional intermittency. *Philosophical Transactions of the Royal Society of London A*, 357 :2495–2509, Sep. 1999.

[22] E.J. Candès and D.L. Donoho. Ridgelets : the key to high dimensional intermittency ? *Philosophical Transactions of the Royal Society of London A*, 357, 1999. 2495–2509.

[23] E.J. Candès and D.L. Donoho. Curvelets, multiresolution representation, and scaling laws. In *SPIE Wavelet Applications in Signal and Image Processing*, 2000.

[24] E.J. Candès and D.L. Donoho. Curvelets and curvilinear integrals. *Journal of Approximation Theory*, 113(1) :59–90, Nov. 2001.

[25] E.J. Candès and D.L. Donoho. Recovering edges in ill-posed inverse problems : Optimality of curvelet frames. *Annals of Statistics*, 30 :784–842, 2002.

[26] E.J. Candès and D.L. Donoho. New tight frames of curvelets and optimal representations of objects with C^2 singularities. *Communications on Pure and Applied Mathematics*, 57(2) :219–266, 2003.

[27] E.J. Candès, J.K. Romberg, and T. Tao. Robust uncertainty principles : Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2), 2006. 489–509.

[28] E.J. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12) :4203—4215, 2005.

[29] P. Carre, D. Helbert, and E. Andres. 3D fast ridgelet transform. In *International Conference on Image Processing*, volume 1, pages 1021–1024, 2003.

[30] V. Chandrasekaran, M.B. Wakin, D. Baron, and R.G. Baraniuk. Surflets : A sparse representation for multidimensional functions containing smooth discontinuities. *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, July 2004.

[31] V. Chandrasekaran, M.B. Wakin, D. Baron, and R.G. Baraniuk. Representation and compression of multidimensional piecewise functions using surflets. *IEEE Transactions on Information Theory*, 55(1) :374–400, Jan. 2009.

[32] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1), 1999. 33–61.

[33] Shaobing Scott Chen. *Basis Pursuit*. PhD thesis, Stanford University, Nov. 1995.

[34] Z. Chen and R. Ning. Breast volume denoising and noise characterization by 3D wavelet transform. *Computerized Medical Imaging and Graphics*, 28(5) :235–246, 2004.

[35] O. Christensen. Frames, riesz bases, and discrete gabor/wavelet expansions. *BULLETIN-AMERICAN MATHEMATICAL SOCIETY*, 38(3) :273–292, 2001.

[36] O. Christensen. *An introduction to frames and Riesz bases*. Birkhauser, 2003.

[37] P.L. Combettes. Solving monotone inclusions via compositions of nonexpansive averaged operators. *Optimization*, 53(5–6) :475–504, Dec. 2004.

[38] M.E. Davies and R. Gribonval. Restricted isometry constants where ℓ_p sparse recovery can fail for $0 < \mathrm{p} \leq 1$. *IEEE Trans. on Information Theory*, 5(55) :2203—2214, 2009.

[39] L. Demanet. *Curvelets, Wave Atoms, and Wave Equations*. PhD thesis, California Institute of Technology, May 2006.

[40] L. Demanet and L. Ying. Curvelets and wave atoms for mirror-extended images. In *SPIE Wavelets XII conference*, August 2007.

[41] R.A. DeVore and V.N. Temlyakov. Some remarks on greedy algorithms. *Advances in Computational Mathematics*, 5(2-3) :173–187, 1996.

[42] A. Dima, M. Scholz, and K. Obermayer. Semiautomatic quality determination of 3D confocal microscope scans of neuronal cells denoised by 3D wavelet shrinkage. In H. H. Szu, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 3723, pages 446–457, March 1999.

[43] M.N. Do and M. Vetterli. The contourlet transform : an efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14(12) :2091–2106, Dec. 2005.

[44] D. L. Donoho and X. Huo. Beamlets and multiscale image analysis. *Multiscale and Multiresolution Methods, Lecture Notes in Computational Science and Engineering*, 20 :149–196, Springer, NY, USA, 2001.

[45] D.L. Donoho. Nonlinear solution of linear inverse problems by wavelet-vaguelette decomposition. *Applied and Computational Harmonic Analysis*, 2(2) :101–126, 1995.

[46] DL Donoho. Digital ridgelet transform via rectopolar coordinate transform. Technical report, Stanford Univ, Stanford, CA, 1998.

[47] D.L. Donoho. Tight frames of k-plane ridgelets and the problem of representing functions smooth away from d-dimensional singularities. *Applied Mathematics, Proc. Natl. Acad. Sci. USA*, 96 :1828–1833, March 1999.

[48] D.L. Donoho. Wedgelets : Nearly minimax estimation of edges. *The Annals of Statistics*, 27(3) :859–897, 1999.

[49] D.L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4), 2006. 1289–1306.

[50] D.L. Donoho. High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension. *Discrete and Computational Geometry*, 35(4) :617—652, 2006.

[51] D.L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l^1 minimisation. *Proceedings of the National Academy of Sciences*, 100 :2197—2202, 2003.

[52] D.L. Donoho, M. Elad, and V. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1) :6–18, 2006.

[53] D.L. Donoho and X. Huo. Uncertainty principle and ideal atomic decompositions. *IEEE Transactions on Information Theory*, 47(7) :2845–2862, November 2001.

[54] D.L. Donoho and O. Levi. Fast X-Ray and Beamlet Transforms for Three-Dimensional Data. In D. Rockmore and D. Healy, editors, *Modern Signal Processing*, pages 79–116, 2002.

[55] D.L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 2006. submitted.

[56] C. Dosssal. A necessary and sufficient condition for exact recovery by l1 minimization, 2007.

[57] T. Doyle. Interlaced to sequential conversion for EDTV applications. In *Proc. 2nd Int. Workshop Signal Processing of HDTV*, pages 412–430, 1990.

[58] M. Elad, J.-L. Starck, P. Querre, and D.L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis. *Applied and Computational Harmonic Analysis*, 19 :340–358, 2005.

[59] K. Engan, S.O. Aase, and J.H. Husoy. Method of optimal directions for frame design. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing- Proceedings*, volume 5, pages 2443–2446, 1999.

[60] E. Escalera, E. Slezak, and A. Mazure. New evidence for subclustering in the Coma cluster using the wavelet analysis. *Astronomy and Astrophysics*, 264 :379–384, October 1992.

[61] M. J. Fadili, J.-L. Starck, and F. Murtagh. Inpainting and zooming using sparse representations. *The Computer Journal*, 52(1) :64–79, 2007.

[62] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Robust shift and add approach to super-resolution. *Proc. IEEE Conf. on Applic. Digital Signal and Image Processing*, pages 121—130, 2003.

[63] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Advances and challenges in super-resolution. *International Journal of Imaging Systems and Technology*, 14(2) :47, 57 2004.

[64] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10) :1327–1344, 2004.

[65] A. Feuer and A. Nemirovsky. On sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 49(6) :1579—1581, 2003.

[66] D.J. Field, A. Hayes, and R.F. Hess. Contour integration by the human visual system : Evidence for a local. *Vision Research*, 33(2) :173–193, 1993.

[67] S. Foucart and M.-J. Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3) :395—407, 2009.

[68] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *J. Amer. Statist. Asso.*, 76 :817–823, 1981.

[69] F. Friedrich, L. Demaret, H. Fuhr, and K. Wicker. Efficient moment computation over polygonal domains with an application to rapid wedgelet approximation. *SIAM Journal on Scientific Computing*, 29(2) :842, 2008.

[70] J.-J. Fuchs. On sparse representations in arbitrary redundant bases. *IEEE Transactions on Information Theory*, 50(6) :1341–1344, 2004.

[71] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Trans. Pattern Analysis Machine Intelligence*, 14(3) :267–383, 1992.

[72] R. Gribonval and M. Nielsen. Sparce decomposition in incoherent dictionaries. In *Proceedings of the IEEE International Conference on Image Processing*, Barcelona, Spain, 2003.

[73] R. Gribonval and M. Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49(12) :3320–3325, 2003.

[74] A. Grossmann, R. Kronland-Martinet, and J. Morlet. In : Combes et al.(eds.) Wavelets : Time-Frequency Methods and Phase Space, 1989.

[75] B.K. Gunturk, J. Glotzbach, Y. Altunbasak, R.W. Schafer, and R.M. Mersereau. Demosaicking : color filter array interpolation. *IEEE Signal Processing Magazine*, 22(1) :44–54, 2005.

[76] K. Guo, D. Labate, W.Q. Lim, G. Weiss, and E. Wilson. Wavelets with composite dilations and their MRA properties. *Applied and Computational Harmonic Analysis*, 20(2) :202–236, 2006.

[77] K. Guo, WQ Lim, D. Labate, G. Weiss, and E. Wilson. Wavelets with composite dilations. *Electron. Res. Announc. Amer. Math. Soc*, 10 :78–87, 2004.

[78] G. De Haan and E. B. Bellers. De-interlacing of video data. *IEEE Trans. Consum. Electron.*, 43(8) :819–825, Aug. 1997.

[79] G. De Haan and E. B. Bellers. Deinterlacing : An overview. *Proc. IEEE*, 86(9) :1839–1857, Sep. 1998.

[80] G. Hennenfent and F.J. Herrmann. Seismic denoising with nonuniformly sampled curvelets. *IEEE Computing in Science and Engineering*, 8(3) :16–25, May 2006.

[81] F.J. Herrmann and G. Hennenfent. Non-parametric seismic data recovery with curvelet frames. *Geophysical Journal International*, 173(1) :233–248, 2008.

[82] J.A. Högbom. Aperture synthesis with a non-regular distribution of interferometer baselines. *Astronomy and Astrophysics Supplement Series*, 15, 1974. 417–426.

[83] M. Holdschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real time algorithm for signal analysis with the help of the wavelet transform.. Wavelet, 1989.

[84] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17 :185–204, 1981.

[85] F. Jacquelin. Sk-0000400704-01, note de synthèse sur l'estimation de mouvement dense. Technical report, Sagem, Oct. 2008.

[86] F. Jacquelin. Sk-0000402431-01, spécification technique de la fonction de postaccumulation pour les cameras ibis et matis. Technical report, Sagem, Nov. 2008.

[87] F. Jacquelin. Sk-0000412754-01, note d'implémentation de l'algorithme de cah local par harmonisation des moyennes et des variances locales. Technical report, Sagem, Apr. 2009.

[88] J. Kalifa, S. Mallat, and B. Rougé. Deconvolution by thresholding in mirror wavelet bases. *IEEE Transactions on Image Processing*, 12(4) :446–457, 2003.

[89] D. Keren, S. Peleg, and R. Brada. Image sequence enhancement using sub-pixel displacements. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 742–746, 1988.

[90] N. Kingsbury. Complex wavelets for shift invariant analysis and filtering of signals. *Applied and Computational Harmonic Analysis*, 10(3) :234–253, May 2001.

[91] E. Kolaczyk. *Wavelet Decomposition of certain Homogeneous Linear Inverse problems.* PhD thesis, Stanford University, 1994.

[92] A. Lannes, E. Anterrieu, and P. Marechal. Clean and Wipe. *AAS*, 123 :183–198, May 1997.

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

[94] J.S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2 :165–168, 1980.

[95] W.A. Lentz. *Characterisation of Noise in Uncooled IR Bolometer Arrays.* PhD thesis, Massachusetts Institute of Technology, May 1998.

[96] Feng Li, Tim. J. Cornwell, and Frank De Hoog. The applications of compressed sensing to radio astronomy. In Gopal Pandurangan, V. Anil Kumar, Gu Ming, Yunhao Liu, and Yingshu Li, editors, *Wireless Algorithms, Systems, and Applications*, volume 6221 of *Lectures notes in computer science*, pages 352–359. Springer Berlin / Heidelberg, 2010.

[97] P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16 :964–979, 1979.

[98] Y. Lu and M. N. Do. 3-D directional filter banks and surfacelets. In *Proc. of SPIE Conference on Wavelet Applications in Signal and Image Processing XI, San Diego, USA*, 2005.

[99] Y.M. Lu and M.N. Do. Multidimensional directional filter banks and surfacelets. *IEEE Transactions on Image Processing*, 16(4) :918–931, 2007.

[100] B.D. Lucas. *Generalized image matching by the method of differences.* PhD thesis, Carnegie Mellon University Pitsburg, PA, USA, 1985.

[101] L.B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79(6) :745–754, 1974.

[102] J. Ma and M.Y. Hussaini. Three-dimensional curvelets for coherent vortex analysis of turbulence. *Appl. Phys. Letters*, 91(184101), October 2007.

[103] A. Maalouf and M.C. Larabi. Grouplet-based color image super-resolution. In *17th European Signal Processing Conference (EUSIPCO 2009) Glasgow, Scotland, August 24-28, 2009*, 2009.

[104] S. Mallat. A theory for multiresolution signal decomposition : The wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7) :674–693, 1989.

[105] S. Mallat. Multiresolution approximations and wavelet orthonormal bases of l2(r). *Transactions of the American Mathematical Society*, 315(1) :69–87, 1989.

[106] S. Mallat. *A wavelet tour of signal processing.* Academic Press, 1998.

[107] S. Mallat. *A Wavelet Tour of Signal Processing, The Sparse Way.* Academic Press, 3rd edition, December 2008.

[108] S. Mallat. Geometrical grouplets. *Applied and Computational Harmonic Analysis*, 26(2) :161–180, 2009.

[109] S. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2) :617–643, 1992.

[110] S.G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12) :3397–3415, Dec. 1993.

[111] C. Malpertu. Sk-0000406826-01, etude du contrôle automatique d ?histogramme comparaison d'algorithmes. Technical report, Sagem, Feb. 2009.

[112] V. J. Martínez, J.-L. Starck, E. Saar, D. L. Donoho, S. C. Reynolds, P. de la Cruz, and S. Paredes. Morphology of the Galaxy Distribution from Wavelet Denoising. *Astrophysical Journal*, 634 :744–755, December 2005.

[113] V.J. Martinez, S. Paredes, and E. Saar. Wavelet analysis of the multifractal character of the galaxy distribution. *MNRAS*, 260 :365–375, January 1993.

[114] C. J. Miller, C. Genovese, R. C. Nichol, L. Wasserman, A. Connolly, D. Reichart, A. Hopkins, J. Schneider, and A. Moore. Controlling the false-discovery rate in astrophysical data analysis. *The Astronomical Journal*, 122(6) :3492–3505, Dec. 2001.

[115] J.-J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93 :273—299, 1965.

[116] J.P. Mur. Sk-0000140790, spécification technique de besoin de l'algorithme de stabilisation d ?image. Technical report, Sagem, June 2006.

[117] Y. Nesterov. *Introductory lectures on convex optimization : Basic course*. Kluwer, Boston, 2003.

[118] Y. Nesterov. Gradient methods for minimizing composite objective function. *CORE Discussion Paper*, 2007.

[119] M. Nicolle. Sk-0000157342-01, comparaison d'algorithmes de déconvolution. Technical report, Sagem, June 2008.

[120] H.S. Oh, Y. Kim, Y.Y. Jung, SJ Ko, and AW Morales. Spatio-temporal edge-based median filtering for deinterlacing. In *IEEE International Conference on Consumer Electronics*, pages 52–53. IEEE ; 1999, 2000.

[121] B.A. Olshausen and B.J. Field. Sparse coding with an overcomplete basis set : A strategy employed by v1 ? *Vision Research*, 37(23) :3311–3325, 1997.

[122] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Omp. *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, 1993.

[123] G. Peyré. Texture Synthesis with Grouplets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4) :733–746, 2010.

[124] G. Peyré, J. Fadili, and J.L. Starck. Learning the morphological diversity. *SIAM J. Imaging Sciences*, to appear, 2010.

[125] G. Peyré and S. Mallat. Discrete bandelets with geometric orthogonal filters. *Proceedings of ICIP'05*, 1 :65–68, Sep. 2005.

[126] S. Pires, J.-L. Starck, A. Amara, R. Teyssier, A. Refregier, and J. Fadili. FASTLens (FAst STatistics for weak Lensing) : Fast method for Weak Lensing Statistics and map making. *ArXiv e-prints*, April 2008.

[127] M. Protter, M. Elad, H. Takeda, and P. Milanfar. Generalizing the nonlocal-means to super-resolution reconstruction. *IEEE Transactions on Image Processing*, 18(1) :36–51, 2009.

[128] K.S.K. Remi, A.C. Evans, and G.B. Pike. MRI simulation-based evaluation of image-processing and classification methods. *IEEE Transactions on Medical Imaging*, 18(11) :1085, 1999.

[129] W.H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1) :55–59, 1972.

[130] J.K. Romberg, M. Wakin, and R. Baraniuk. Multiscale wedgelet image analysis : fast decompositions and modeling. In *in IEEE Int. Conf. on Image Proc. 2002*, volume 3, pages 585–588, Jun. 2002.

[131] A. B. Romeo, C. Horellou, and J. Bergh. N-body simulations with two-orders-of-magnitude higher performance using wavelets. *MNRAS*, 342 :337–344, June 2003.

[132] A. B. Romeo, C. Horellou, and J. Bergh. A wavelet add-on code for new-generation n-body simulations and data de-noising (jofiluren). *MNRAS*, 354 :1208–1222, November 2004.

[133] D. Ruppert. Empirical-Bias Bandwidths for Local Polynomial Nonparametric Regression and Density Estimation. *Journal of the American Statistical Association*, 92(439), 1997.

[134] D. Rusanovskyy and K. Egiazarian. Video Denoising Algorithm in Sliding 3D DCT Domain. *Lecture Notes in Computer Science*, 37(08) :618–625, 2005.

[135] Boyd R.W. *Radiometry and the detection of optical radiation.* Wiley Interscience, 1983.

[136] S. Sardy, A. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries. *Journal of Computational and Graphical Statistics*, 9(2) :361–379, 2000.

[137] I.W. Selesnick. The double-density dual-tree DWT. *IEEE Transactions on Image Processing*, 52(5) :1304–1314, May 2004.

[138] I.W. Selesnick, R.G. Baraniuk, and N.C. Kingsbury. The dual-tree complex wavelet transform. *IEEE Signal Processing Magazine*, 22(6) :123–151, 2005.

[139] I.W. Selesnick and K.Y. Li. Video denoising using 2D and 3D dual-tree complex wavelet transforms. In *Proc. of SPIE conference on Wavelet Applications in Signal and Image Processing X*, San Diego, USA, Aug. 2003.

[140] L. Sendur and I. W. Selesnick. Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency. *IEEE Trans. on Signal Processing*, 50(11) :2744–2756, Nov. 2002.

[141] L. Sendur and I. W. Selesnick. Bivariate shrinkage with local variance estimation. *IEEE Signal Processing Letters*, 9(12) :438–441, Dec. 2002.

[142] E. Slezak, V. de Lapparent, and A. Bijaoui. Objective detection of voids and high density structures in the first CfA redshift survey slice. *Astrophysical Journal*, 409 :517–529, 1993.

[143] J.-L. Starck, A. Bijaoui, B. Lopez, and C. Perrier. Image reconstruction by the wavelet transform applied to aperture synthesis. *Astronomy and Astrophysics*, 283, 1994. 349–360.

[144] J.-L. Starck, E.J. Candès, and D.L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6) :670–684, June 2002.

[145] J.-L. Starck, E.J. Candès, and D.L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6), 2002. 131–141.

[146] J.-L. Starck and F. Murtagh. *Astronomical Image and Data Analysis.* Astronomical image and data analysis, by J.-L. Starck and F. Murtagh. Astronomy and astrophysics library. Berlin : Springer, 2006, 2006.

[147] J.-L. Starck, F. Murtagh, E.J. Candès, and D.L. Donoho. Gray and color image contrast enhancement by the curvelet transform. *IEEE Transactions on Image Processing*, 12(6) :706–717, 2003.

[148] J.-L Starck, F. Murtagh, and M.J. Fadili. *Sparse Signal and Image Processing : Wavelets, Curvelets and Morphological Diversity*. Cambridge University Press, Cambridge, UK, 2010.

[149] J.L. Starck, A. Bijaoui, B. Lopez, and C. Perrier. Image reconstruction by the wavelet transform applied to aperture synthesis. *Astronomy and Astrophysics*, 283 :349–360, 1999.

[150] J.L. Starck, D.L. Donoho, and E.J. Candès. Very high quality image restoration by combining wavelets and curvelets. In A. Laine, M.A. Unser, and A. Aldroubi, editors, *SPIE conference on Signal and Image Processing : Wavelet Applications in Signal and Image Processing IX, San Diego, 1-4 August*. SPIE, 2001.

[151] J.L. Starck, M. Elad, and D.L. Donoho. Redundant multiscale transforms and their application for morphological component analysis. *Advances in Imaging and Electron Physics*, 132 :287–348, 2004.

[152] J.L. Starck, M. Elad, and D.L. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *IEEE Transactions on Image Processing*, 14(10) :1570–1582, October 2005.

[153] J.L. Starck, V.J. Martinez, D.L. Donoho, O. Levi, P. Querre, and E. Saar. Analysis of the spatial distribution of galaxies by multiscale methods. *Eurasip Journal on Applied Signal Processing*, 15 :2455–2469, November 2005.

[154] J.L. Starck and F. Murtagh. Image restoration with noise suppression using the wavelet transform. *Astronomy and Astrophysics*, 288 :342–348, 1994.

[155] J.L. Starck, M.K. Nguyen, and F. Murtagh. Deconvolution based on the curvelet transform. In *International Conference on Image Processing*, pages II : 993–996, 2003.

[156] J.L. Starck, M.K. Nguyen, and F. Murtagh. Wavelets and curvelets for image deconvolution : a combined approach. *Signal Processing*, 83 :2279–2283, 2003.

[157] T. Strohmer and R.W. Heath. Grassmannian frames with applications to coding and communication. *Applied and Computational Harmonic Analysis*, 14(3) :257–275, 2003.

[158] A. M. Tekalp. *Digital Video Processing*. Prentice Hall, 1995.

[159] R. Teyssier. Cosmological hydrodynamics with adaptive mesh refinement. a new high resolution code called ramses. *Astronomy and Astrophysics*, 385 :337–364, April 2002.

[160] J.A. Tropp. Greed is good : Algorithmic results for sparse approximation. *IEEE Trans. on Information Theory*, 50(10) :2231–2242, 2004.

[161] J.A. Tropp. Just relax : Convex programming methods for subset selection and sparse approximation. *IEEE Transactions on Information Theory*, 52(3) :1030–1051, 2006.

[162] J.A. Tropp and A.C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. on Information Theory*, 53(12) :4655–4666, 2007.

[163] B.P. Wakker and U.J. Schwarz. The multi-resolution Clean and its application to the short-spacing problem in interferometry. *Annual Reviews of Astronomy and Astrophysics*, 200, 1988. 312.

[164] F.M. Wang, D. Anastassiou, and A.N. Netravali. Time-recursive deinterlacing for IDTV and pyramid coding. *Signal Processing : Image Communication*, 2(3) :365–374, 1990.

[165] Y. Wiaux, L. Jacques, G. Puy, A. M. M. Scaife, and P. Vandergheynst. Compressed sensing imaging techniques for radio interferometry. *MNRAS*, 395 :1733–1742, May 2009.

[166] N. Wiener. Extrapolation, interpolation and smoothing of stationary time series. *MIT Press, Cambridge*, 1949.

[167] A. Woiselle, J.L. Starck, and J. Fadili. 3D curvelet transforms and astronomical data restoration. *Applied and Computational Harmonic Analysis*, 2010.

[168] A. Woiselle, J.L. Starck, and J. Fadili. 3D Data Denoising and Inpainting with the Fast Curvelet transform. *Journal of Mathematical Imaging and Vision*, submitted, 2010.

[169] S. Yang, Y.Y. Jung, H.O.L.E.E. Young, and R.H. Park. Motion compensation assisted motion adaptive interlaced-to-progressive conversion. *IEEE transactions on circuits and systems for video technology*, 14(9) :1138–1148, 2004.

[170] L. Ying, L. Demanet, and E.J. Candès. 3D Discrete Curvelet Transform. In *Proceedings of Wavelets XI conference, San Diego*, July 2005.

[171] H. Yoo and J. Jeong. Direction-oriented interpolation and its application to deinterlacing. *IEEE Trans. Consum. Electron.*, 48(4) :954–962, Apr. 2002.

[172] G. Yu and S. Mallat. Sparse super-resolution with space matching pursuits. In *SPARS'09 – Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[173] B. Zhang, J. Fadili, and J.L. Starck. Wavelets, ridgelets, and curvelets for poisson noise removal. *IEEE Transactions on Image Processing*, 17(7) :1093–1108, Jul. 2008.

Annexe A

Echantillonnage compressif en Fourier

A.1 Abstract

This technical report presents how to reconstruct an image from an incomplete coverage of the Fourier domain using a regularization prior based on sparsity. The sparse solution can be obtained using a minimization algorithm such as FISTA, MCA or Nesterov. Using a simulation relative to radio-interferometric image deconvolution, we compare these minimization methods. An example illustrates the results.

A.2 Introduction

The principle of imaging by aperture synthesis lies in the determination of fringe contrasts and phases from different configurations. The number of spatial frequencies after observations would be ideally the same than the number of independent pixels in the restored images. This is however difficult to reach with radio interferometers. The frequency holes are the consequence of this incomplete coverage. The point spread function (PSF), the inverse Fourier transform of the transfer function, is an irregular function with large rings and wings. A clean image restoration needs to fill the frequency holes. Hence, reconstructing the inverse from it incomplete Fourier coverage is an ill-posed inverse problem.

The best methods which have been proposed in the past for reconstructing such images are based on sparsity. Indeed, the famous CLEAN algorithm [82] and its multiscale version [163, 143, 146] can be seen as matching pursuit algorithms [92], and therefore enforce the sparsity of the solution, in the direct space for CLEAN and in the wavelet space for Multiresolution CLEAN. These results are also very well supported by the new sampling theory, i.e. the Compressed Sensing (CS) [27, 49]. If the solution is sparse, for instance in the wavelet domain, then CS can indeed be considered as a very elegant framework to consider the radio-interferometric image reconstruction problem [12, 165], and recent l_0 and l_1 CS reconstruction methods (see for instance [55, 27]) would certainly improve the computation time to derive these dirac-sparse or wavelet-sparse solutions.

A.3 What is Sparsity ?

A signal x, $x = [x_1, \cdots, x_N]^T$, is sparse if most of its entries are equal to zero. For instance, a k-sparse signal is a signal where only k samples have a non-zero value. A less

strict definition is to consider a signal as weakly sparse or compressible when only a few of its entries have a large magnitude, while most of them are close to zero.

If a signal is not sparse, it may be *sparsified* using a given data representation. For instance, if x is a sine, it is clearly not sparse but its Fourier transform is extremely sparse (i.e. 1-sparse). Hence we say that a signal x is sparse in the Fourier domain if its Fourier coefficients $\hat{x}[u]$, $\hat{x}[u] = \frac{1}{N}\sum_{k=-\infty}^{+\infty} x[k]e^{2i\pi\frac{uk}{N}}$, is sparse. More generally, we can model a vector signal $x \in \mathbb{R}^N$ as the linear combination of T elementary waveforms, also called *signal atoms* : $x = \mathbf{\Phi}\alpha = \sum_{i=1}^{T} \alpha[i]\phi_i$,, where $\alpha[i] = \langle x, \phi_i \rangle$ are called the decomposition coefficients of x in the dictionary $\mathbf{\Phi} = [\phi_1, \cdots, \phi_T]$ (the $N \times T$ matrix whose columns are the atoms normalized to a unit ℓ_2-norm, i.e. $\forall i \in [1, T], \|\phi_i\|_{\ell^2} = 1$).

Therefore to get a sparse representation of our data we need first to define the dictionary $\mathbf{\Phi}$ and then to compute the coefficients α. x is sparse in Φ if the sorted coefficients in decreasing magnitude have a fast decay ; i.e. most of coefficients α vanish but a few.

What is the Best Dictionary ?

Obviously, the best dictionary is the one which leads to the sparsest representation. Hence we could imagine having a huge overcomplete dictionary (i.e. $T \gg N$), but we would be faced with prohibitive computation time cost for calculating the α coefficients. Therefore there is a trade-off between the complexity of our analysis step (i.e. the size of the dictionary) and the computation time. Some specific dictionaries have the advantage of having fast operators and are very good candidates for analyzing the data. The Fourier dictionary is certainly the most famous, but many others have been proposed in the literature such as wavelets [107], ridgelets [22], curvelets [19, 145], bandlets [93], to name only a few.

A.4 Inverse Problems and Sparsity

Many image processing problems can be formalized as a linear inverse problem,

$$y = Ax + \varepsilon \ , \tag{A.1}$$

where y is a set of noisy measurements, ε is an additive noise, x is the solution of our problem, and A is a linear operator. Finding x knowing the data y and A is an inverse problem. When it has not a unique and stable solution, it is an *ill-posed problem*, and a regularization is necessary to reduce the space of candidate solutions. Once the dictionary Φ is chosen, inverse problems can be regularized using a sparsity penalty. Between all possible solutions, we want the one which has the sparsest representation in the dictionary Φ. Noting α the representation coefficients in Φ, the solution x can be reconstructed as $x = \Phi\alpha$, the sparsity can be measured through the $\|\alpha\|_{\ell^0}$ pseudo norm, which indicates the limit of ℓ^p when $p \to 0$. This counts in fact the number of non-zero elements in the sequence. This approach leads to the following minimization problem :

$$\min_{\alpha} \|\alpha\|_{\ell^0} \text{ s.t. } \|y - A\Phi\alpha\|_{\ell^2} \leq \sigma \ . \tag{A.2}$$

It was proposed to convexify the constraint by substituting the convex ℓ_1 norm for the ℓ_0 norm leading to [32] :

$$\min_{\alpha} \|\alpha\|_{\ell^1} \text{ s.t. } \|y - A\Phi\alpha\|_{\ell^2} \leq \sigma \ . \tag{A.3}$$

This equation can also be recast in its Lagrangian form :

$$\min_{\alpha} \lambda \|\alpha\|_{\ell^1} + \frac{1}{2}\|y - A\Phi\alpha\|^2_{\ell^2} \ . \tag{A.4}$$

Depending on the A operator, there are several ways to obtain the solution of this problem.

A.5 Radio-interferometric Image Reconstruction

In the case of radio-interferometric image reconstruction, we need to solve the following inverse problem :

$$\min_{\alpha} \lambda \|\alpha\|_{\ell^1} + \frac{1}{2}\|y - MF\Phi\alpha\|^2_{\ell^2} \ , \tag{A.5}$$

where y are visibilities (i.e. the measurements in the Fourier domain), F is the Fourier operator, M is the mask (i.e. $M[u, v] = 1$ if we have a measurement at the frequency (u, v) and zero otherwise), and α are the coefficients of the solution in the dictionary Φ (i.e. $x = \Phi\alpha$). If we define $A = MF$, then we see immediately that Eq. A.5 and Eq. A.4 are equivalent.

The simplest method to solve this problem is to apply an iterative soft thresholding (ISTA). But the convergence of such algorithm is very slow.
Starck *et al.* proposed [151] to use a decreasing threshold, which accelerates greatly the convergence. This is the Morphological Component Analysis MCA. Note that in [151], the algorithm can be applied with many representations, while we will consider here only one representation (wavelets).
Beck *et al.* [6] proposed an iterative thresholding algorithm in which the shrinkage operator is not applied to the previous solution but rather on a linear combination of the two previous ones. This accelerated ISTA is called FISTA.
Nesterov proposed [117, 118] a different approach, estimating two sequences, one for the estimate and one for the gradient, and applying the proximal computation (soft thresholding in our case) to the integrated gradient sequence and to a linear combination of the previous solution and the gradient. Each iteration of this algorithm is two times longer than any of the other algorithms.

A.6 Software

The attached program contains implementations of FISTA, MCA and Nesterov in the general case, with an example when $A = MF$, F being the Fourier transform and M a mask of missing samples. It is called using the command-line with two possible patterns : Either you enter an image (full synthetic data or reconstructed from fourier samples), and a mask to be applied in Fourier domain :

```
mr_uv_inpainting options image_name mask_name output_name
```

or you give only $N \times N \times 2$ floating data with the planes containing the real and imaginary parts of the Fourier image, and the mask is estimated from the zeros in the data :

```
mr_uv_inpainting options fourier_name output_name
```

The options in common for all algorithms are :

`-m method`	Minimisation method 1 : ISTA 2 : FISTA (default) 3 : MCA 4 : NESTEROV
`-t transform`	The transform used 1 : decimated wavelets 2 : undecimated wavelets 3 : isotropic wavelets (default)
`-v`	Verbose mode.
`-n nscale`	Specifies the number of scales used in the wavelet transform.
`-g sigma`	To set the level of the noise. Default $\sigma = 1$.
`-i Niter`	To set the maximal number of iterations.
`-P`	To add a positivity constraint on the sequence of estimations.
`-mu Mu`	To specify the step μ of the algorithm. Default μ=1.
`-tolvar TolVar`	To define the stopping criterion : stops when $\|x^{n+1} - x^n\|_\infty < TolVar$. Default 10^{-6}.
`-use-init`	To use the input as an initial point.

A specific option for Nesterov :

`-cont Ncont`	To set the number of continuations.

When the input parameter is in the Fourier domain, the input Fourier coefficients are not supposed to be normalized. In our experiments, we generated the input Fourier image using the centered IDL FFT program.

The degration model, i.e. $A = MF$, can easily be modified in the c++ code (see `_degrade` function), so this code can also be used for more complex inverse problems.

A.7 Experiment

Fig. A.1 shows the simulated data used in this experiment. Fig. A.1 top left shows the simulated image x which contains gaussians with different sizes and intensity at random positions, and top right its Fourier transform in log space. Fig. A.1 middle shows the same two images, but a gaussian noise of standard deviation $\sigma = 0.01$ was added to the previous image. Fig. A.1 bottom shows the mask M, i.e. the Fourier coverage, and the noisy data in Fourier space multiplied by the mask. This last image represent the input data of the algorithm.

Fig. A.2 top left shows the minimum energy reconstruction $\tilde{x} = F^{-1}y$ (also called dirty map), and top right shows the same image rescaled to the dynamic of the original image. Fig. A.2 bottom left and right shows respectively the reconstruction obtained using FISTA and its Fourier transform, using with the following parameters and command-line :

```
mr_uv_inpainting -a 2 -t 3 --mu 1 --tolvar 1e-6 -n 7 -P -i 200
-g 0.01 image_n mask out_fista
```

For the proposed example, the best result is obtained with FISTA with 200 iterations and the positivity constraint, which recovers all the sources with their correct size, even the small one on top of the extended source (referred to as *hidden peak*). On this same data, letting FISTA and Nesterov converge ($\|\tilde{x}^n - \tilde{x}^{n-1}\| < TolVar = 2 \cdot 10^{-5}$, without

the positivity constraint), takes respectively about 2300 and 1400 iterations. Both results are good, but the Nesterov one doesn't recover the hidden peak. The results obtained with MCA are good too (slightly better than Nesterov after convergence), but less accurate than with FISTA. The best result for this example is the FISTA reconstruction with 200 iterations and the positivity constraint.

A.8 Conclusion

The use of the positivity constraint `-P` accelerates the convergence and leads to better results, but in this case, the algorithms may not converge. The user may have to set a lower number of iterations, depending on the data and the method. In our example, Nesterov with positivity actually slowly diverges after 80 iterations (at which state the result is already good), while FISTA gives very good results with 200 iterations.

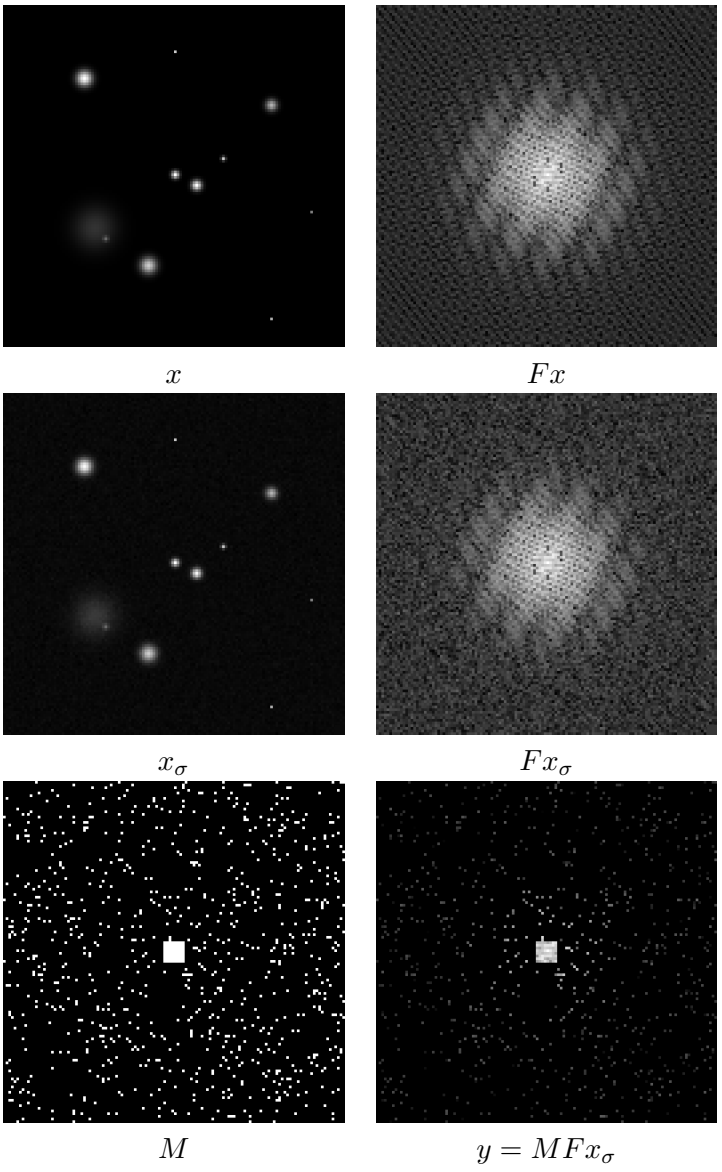


FIG. A.1 – **The simulated data**. Top : Simulated image and its Fourier transform (FFT) in log space. Middle : same image but contaminated contaminated with noise ($\sigma = 0.01$), and its FFT. Bottom : the binary mask containing the low frequencies and 5% of the higher frequencies, and the observed data y in Fourier space (i.e. the input data of the algorithm).

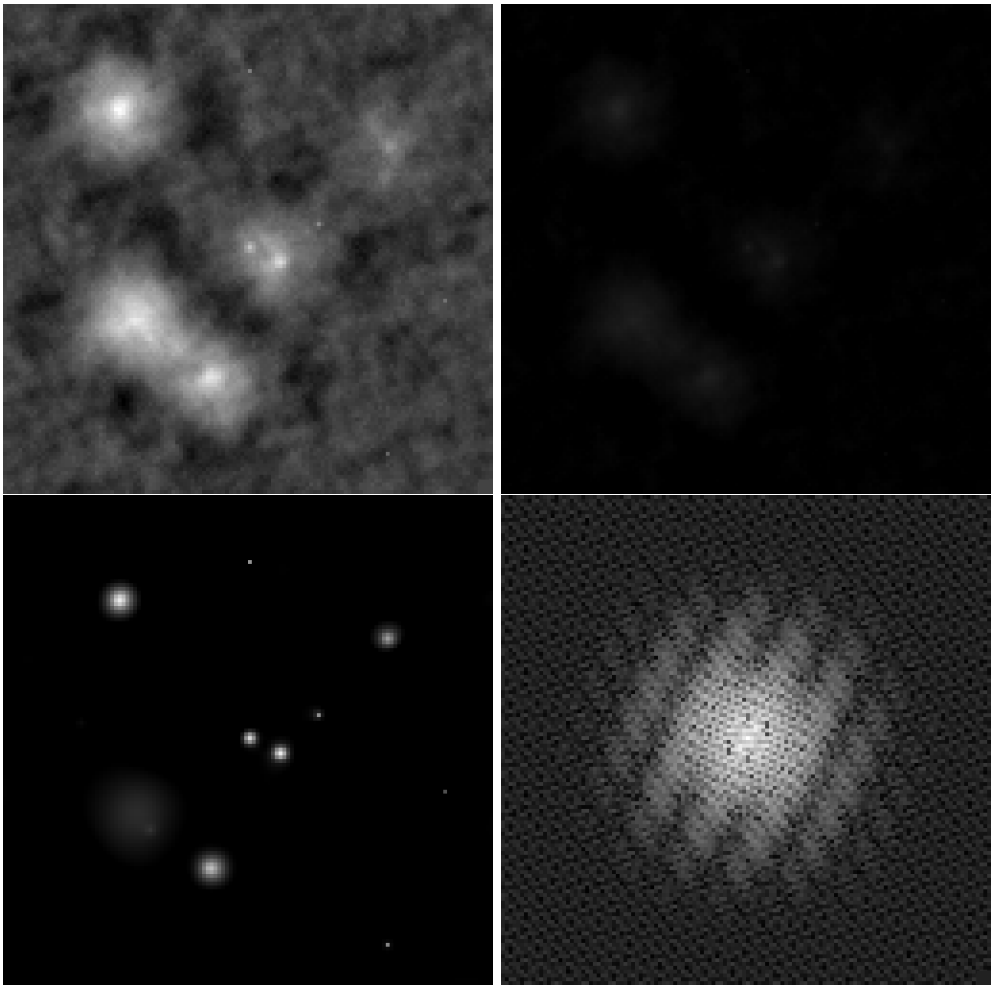


FIG. A.2 – **The recovered signals**. Top : the inverse Fourier transform of the data $\tilde{x}_0 = F^{-1}y$ with adapted dynamic, and on the dynamic of the input image. Bottom : the reconstruction in spatial and spectral domains, using FISTA with the proposed parameters.

Annexe B

SparseBox : A 2-D/3-D Sparse Transforms ToolBox

Introduction

This ToolBox contains implementations of multi-resolution representations with fast analysis and synthesis operators, that can be used to analyze or process data in two or three dimensions.
The first Chapter briefly explains the theory and the implementation of the transforms, and the second one presents the functions available to the user, with some examples. The third one lists all functions and gives their full prototype with possible options.

B.1 Data representations

This chapter will briefly introduce the theory and mechanics behind the decompositions proposed in this ToolBox, starting from 2-D representations as wavelets, discrete cosine and curvelets, and 3-D BeamCurvelets and FastCurvelets.

B.2 Wavelets

In one dimension, *wavelets* are a family of oscillating functions at different scales on which we project the data. Depending on the application, we can use wavelets that are more oscillating or more compact, and the family of functions can either be a basis (the same number of functions as the number of samples in the signal) of a frame (more wavelets than samples), inducing redundancy. In order to be able to represent every signal, wavelets are associated with *scaling functions*, on which we project the signal as well, which produces lower resolution approximations of the signal. The list of coefficients obtained as the projection of a signal onto all these wavelets and scaling functions forms the *wavelet transform* of the signal. The wavelet functions $\psi_{s,k}$ are defined as translations and dilations of a given *mother wavelet* ψ with chosen properties, and thus are indexed by a scale parameter s and a position parameter k :

$$\psi_{s,k} = \frac{1}{\sqrt{s}}\psi\left(\frac{x-k}{s}\right).$$

Figure B.1 shows an example of wavelet functions at different scales and a scaling function, and Figure B.2 the decomposition of a signal on such functions translated at every position.

The wavelet transform in higher dimensions is separable, and can be applied successively on each dimension of the data. This decomposition thus divides a 2-D image in three bands per scale : horizontal and vertical wavelets, horizontal wavelets and vertical scaling functions (approximation), and horizontal approximation and vertical wavelets. The fourth band, the approximation in both directions, can be further decomposed in more scales. In three dimensions, there are seven bands per scale ($2^3 - 1 = 7$).
This ToolBox provides 2-D and 3-D decimated (basis) and undecimated (redundant frame) transforms and reconstructions using wavelets. It provides also 2-D isotropic wavelets with one band per scale instead of three in the separable case.

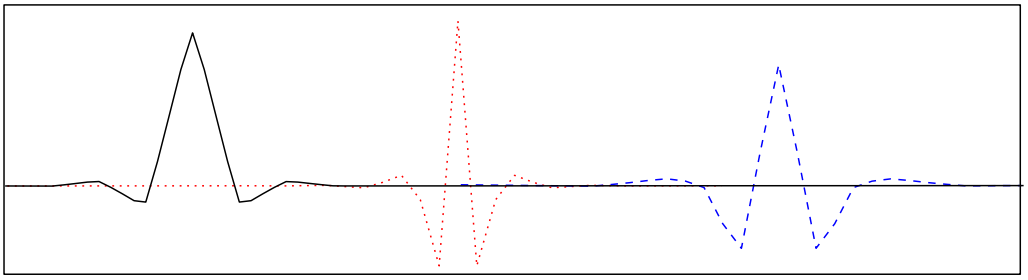


FIG. B.1 – From left to right : a scaling function (with non zero mean), and an associated wavelets at two different scales. The right dashed one is a dilation of the central dotted one.

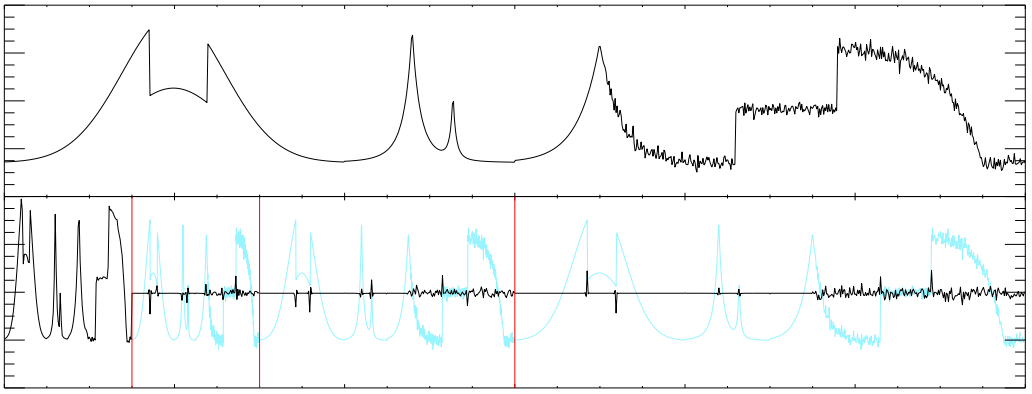


FIG. B.2 – A wavelet decomposition of the signal on top on four scales below from the finest to the largest. The coefficients of each scale are ordered from the coarsest scale (large scaling functions) to the finest scale (small oscillating wavelets).

B.3 2-D Local Discrete Cosine

The local discrete cosine transform (LDCT) in 2-D tiles the image into patches of a given size, and projects each of these patches on 2-D functions that are the product of a horizontal sinusoid and a vertical sinusoid at various frequencies. This transform is well adapted to represent oscillating patterns and textures. Figure B.3 shows a LDCT basis for patches of size 8×8.

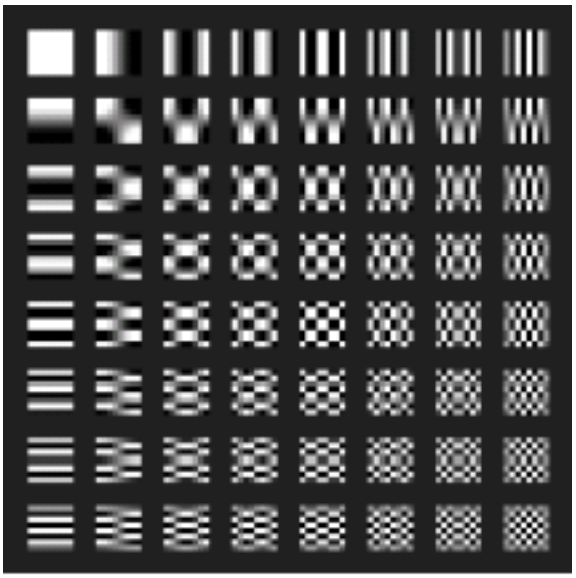


FIG. B.3 – Local discrete cosine basis functions on a 8×8 patch.

B.4 2-D Fast Curvelets

In order to gain directionality, Candès *et al.* [18] designed a representation which is localized ad multi-scale like the wavelets, but is also directional. Fast curvelets are implemented as a paving of Fourier space : a Fourier transform is applied to the image ; a window is used to extract a wedge at a given scale and direction, as shown in Figure B.4 ; an inverse Fourier transform is applied to obtain the coefficients associated to the scale and orientation selected. The resulting functions on which an image is projected in this process are shown in Figure B.5.

B.5 3-D BeamCurvelets

The BeamCurvelet transform projects a 3-D volume on atoms that look like filaments : they are constant in one direction, and oscillate like a 2-D wavelet in the orthogonal plane. This 3-D transform, proposed by Woiselle *et al.* [167] is a 3-D extension to the 2-D first generation curvelets [25, 144], and is implemented as follows :
First, the data-cube is decomposed using an isotropic decimated wavelet transform, and each scale s is partitioned into 3-D blocks $\mathbf{k} = (k_1, k_2, k_3)$. Then on each block is applied a 3-D beamlet transform. A beamlet transform is composed of two steps : a 3-D partial radon transform that projects the cube on planes at all orientations (θ_1, θ_2), and a 2-D wavelet transform on all obtained projections. Figure B.6 summarizes the process and B.7 shows the resulting 3-D functions.

This representation has also been adapted to deal with Poisson noise. The adaptation is done by changing the 3-D wavelet transform with a stabilized one in variance (a MS-VST, see [173]). Then the same blockwise beamlet transform is applied to the stabilized scales.

B.6 3-D Low redundancy Fast Curvelets

An implementation of the 3-D extension to the fast curvelets was proposed by Ying *et al.* [170], which had a redundancy of about fifty, which is huge for 3-D datasets. Woiselle

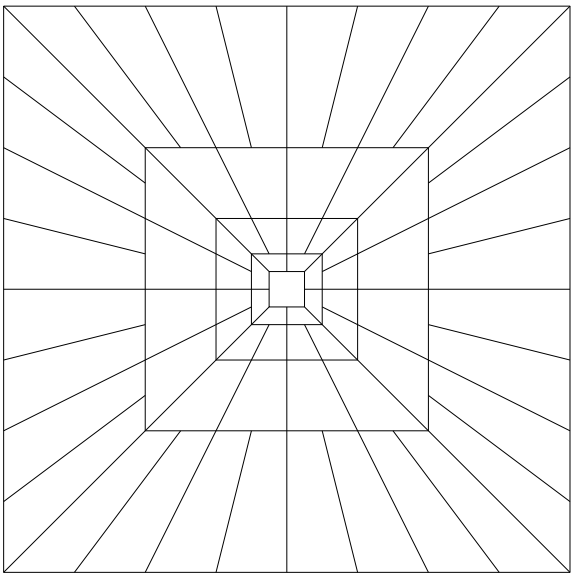


FIG. B.4 – Implementation of the Fast curvelet transform : angular wedges are extracted at every scale in Fourier space, and an inverse Fourier transform is applied to obtain the curvelet coefficients. The central square corresponds to low frequencies and will provide an approximation of the image, and all angular wedges will extract directional content at given scales.

et al. [168] proposed a lower redundancy implementation at the cost of more oscillating fine scale atoms. As in 2-D, the 3-D fast curvelet transform is implemented as a paving of the Fourier domain, as shown in Figure B.8, and it provides atoms which are plate-like, and oscillating in the normal direction. These atoms are localized in space, scale and orientation, and provide a good approximation of surfaces in a 3-D volume. Figure B.9 shows a 3-D fast curvelet atom.

B.7 Using the ToolBox

This chapter first explains how to install the ToolBox, and then how to use the different functions associated to the proposed data representations.

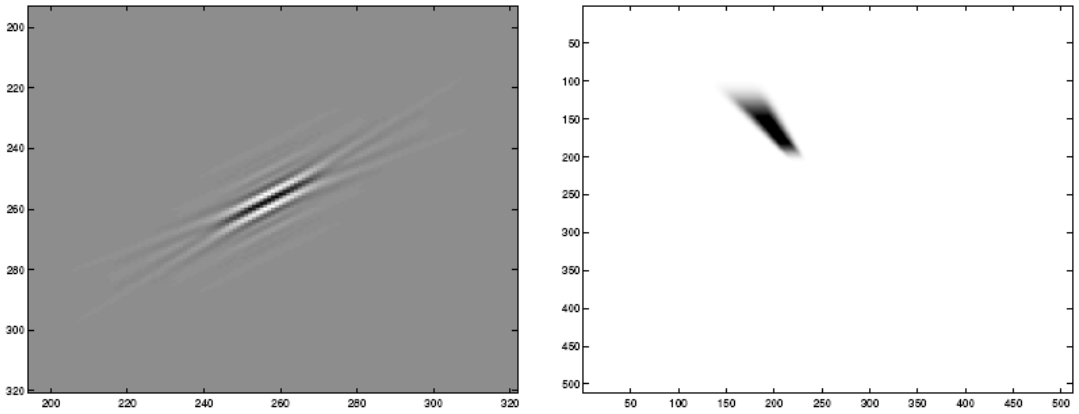


FIG. B.5 – A 2-D fast curvelet atom in direct space and Fourier space.

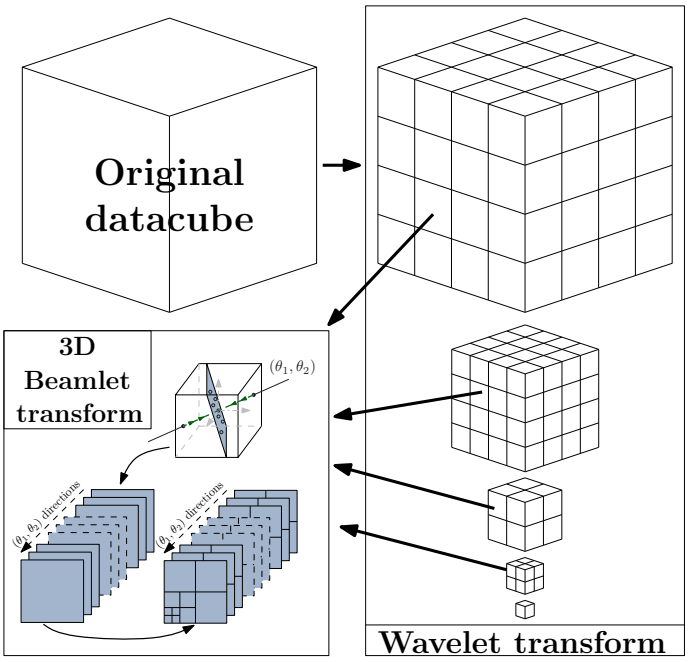


FIG. B.6 – Scheme illustrating the steps of a BeamCurvelet decomposition : the 3-D isotropic wavelet decomposition, the partitioning, and the beamlet transform.

B.8 ToolBox installation

The user should copy the ToolBox directory on his drive, and add it to the Matlab *path*, by using the `file > path` menu or typing `pathtool`.
In order to work properly with the 3-D fast curvelets, the user must create a directory where some files used by the ToolBox will be stored, and define an *environment variable* named `SAP_COM` which contains the path of this directory.

B.9 2D transforms

All transforms, two or three dimensional have the same core functions : the forward transform, the inverse transform (called reconstruction), the thresholding operator, and a filtering function that combines transform, thresholding and reconstruction in a single line. Since their use is very similar for all representations, a single example is shown for the 2-D decimated wavelets, and later on only specificities are shown.

B.9.1 Decimated Orthogonal Wavelets – 2D-DWT

The decimated orthogonal wavelet transform is obtained using the `dwt2d_transform` routine, and the inverse transform with `dwt2d_recons`. Two additional routines can perform the filtering : either `dwt2d_threshold` can modify given coefficients by applying a thresholding/shrinkage method, or `dwt2d_filter` executes the complete filtering, transforming the data, filtering the coefficients and applying an inverse transform.
Their prototype is as follows :

```
Coef = dwt2d_transform(Image, N_scale, [wavelet_type]);
Coef = dwt2d_threshold(Coef, threshold, [filtering_type]);
```

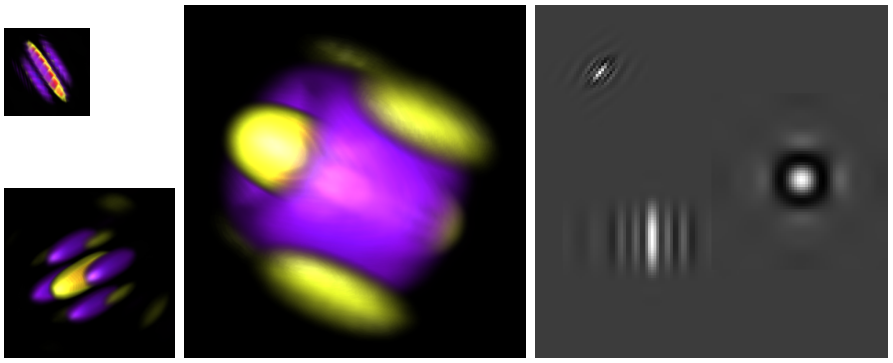


FIG. B.7 – Examples of a BeamCurvelet atoms at different scales and orientations. These are 3D density plots : the values near zero are transparent, and the opacity grows with the absolute value of the voxels. Positive values are red/yellow, and negative values are blue/purple. The right map is a slice of a cube containing these three atoms in the same position as on the left. The top left atom has an arbitrary direction, the bottom left is in the slice, and the right one is normal to the slice.

```
Image2 = dwt2d_recons(Coef, [wavelet_type]);
Image2 = dwt2d_filter(Image, N_scale, threshold, ...
                              [filtering_type], [wavelet_type]);
```

The coefficients are stored in a cell vector of length the number of scales, from the finest scale (first cell) to the low approximation coefficients. For each scale, the bands are stored as a cell vector of images. So it creates three images (horizontal, vertical and horizontal) for each fine scale, and only one for the approximation. As an example, for a four scale wavelet transform, one gets $C\{\text{scale}\}\{\text{band}\}$:

- $C\{1\}\{1 \text{ to } 3\}$ are the finest scale three bands,
- $C\{2\}\{1 \text{ to } 3\}$ are the second scale three bands,
- $C\{3\}\{1 \text{ to } 3\}$ are the third scale three bands,
- $C\{4\}\{1\}$ is the approximation.

Since the transform is decimated, the size of a band at a scale s is 2^{-s} times smaller than the original image, except for the approximation which is the same size as the previous scale's bands.

Here are examples of 2-D wavelet soft and hard thresholding with three scales :

Filtering example

```
% Load an image
  I = double(imread('lena.png'));
  HARD = 1; SOFT = 2;
  lvl = 10; % thresholding level
  Nscale = 3; % number of scales

% Soft thresholding using transform and reconstruction
  C = dwt2d_transform(I,Nscale);
  C = dwt2d_threshold(C,lvl,SOFT);
  % here you can modify the coefficients
  rs = dwt2d_recons(C);
  imagesc(rs);

% Hard thresholding using the filtering function
```

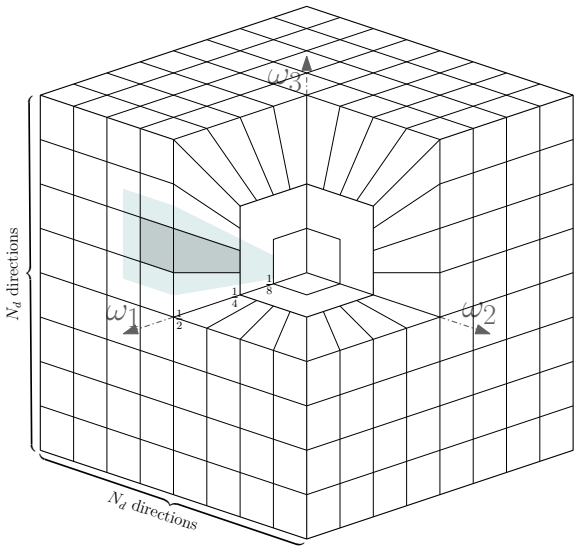


FIG. B.8 – Tiling of the Fourier domain done by the 3-D fast curvelet transform.

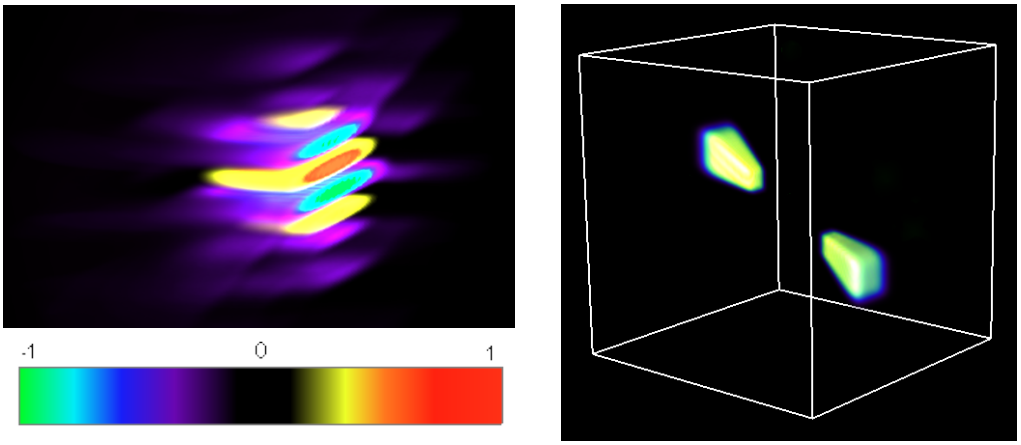


FIG. B.9 – Left : example of a FastCurvelet atom in direct space, cut by a plane to see its inner structure. Right : the same atom in Fourier domain. (The colorbar is valid only for the left image).

```
% in this case, you don't access the coefficients and modify them
rh = dwt2d_filter(I,Nscale,lvl,HARD);
imagesc(rh);
```

B.9.2 Undecimated Orthogonal Wavelets – 2D-UWT

There are four functions available for the undecimated, translation invariant 2-D wavelet transform :

```
Coef = uwt2d_transform(Image, N_scale, [wavelet_type]);
Coef = uwt2d_threshold(Coef, threshold, [filtering_type], [wavelet_type]);
Image2 = uwt2d_recons(Coef, [wavelet_type]);
Image2 = uwt2d_filter(Image, N_scale, threshold, ...
                              [filtering_type], [wavelet_type]);
```

The functions are used in a similar way as the decimated wavelets, and the coefficients have the same structure, $C\{\text{scale}\}\{\text{band}\}$, but all bands have the same size as the input image.

B.9.3 Isotropic Wavelets – 2D-IWT

There are four functions available for the isotropic 2-D wavelet transform :

```
Coef = iwt2d_transform(Image, N_scale);
Coef = iwt2d_threshold(Coef, threshold, [filtering_type]);
Image2 = iwt2d_recons(Coef);
Image2 = iwt2d_filter(Image, N_scale, threshold, [filtering_type]);
```

This transform generates one band per scale, and thus in stored in a cell vector. $C\{\text{scale}\}$, and all bands have the same size as the input image.

B.9.4 Fast Curvelets – 2D-FCT

There are five functions available for the 2-D fast curvelet transform : the curvelet transform must be initialized with the function `curvelet2d_params`, which returns a structure that should be passed to the other classical functions :

```
Param = curvelet2d_params(Image, N_scale, N_direction);
Coef = curvelet2d_transform(Image, Param);
Coef = curvelet2d_threshold(Coef, threshold, [filtering_type]);
Image2 = curvelet2d_recons(Coef, Param);
Image2 = curvelet2d_filter(Image, Param, threshold, [filtering_type]);
```

The coefficient cell-vector is structured in a similar way as with wavelets : $C\{\text{scale}\}\{\text{direction}\}$, where the number of directions depends on the scale, and the last scale (approximation) has only one sub-band.

B.10 1D transforms

As in 2-D, the following 1-D transforms have the same four core functions : the forward transform, the inverse transform, the thresholding operator, and a filtering function. For an example of their use, the user can refer to section B.9.1 on 2-D decimated wavelets. The difference with 1-D operators, is that they work on any kind of data, from one-dimensional to three dimensional, the chosen transformation being applied to a specified direction of the data.

B.10.1 Decimated Orthogonal Wavelets – 1D-DWT

There are four functions available for the 1-D decimated wavelet transform :

```
Coef = dwt1d_transform(Data, N_scale, [direction], [wavelet_type]);
Coef = dwt1d_threshold(Coef, threshold, [filtering_type]);
Cube2 = dwt1d_recons(Coef, [direction], [wavelet_type]);
Cube2 = dwt1d_filter(Data, N_scale, threshold, [direction], ...
                           [filtering_type], [wavelet_type]);
```

The `direction` parameter is set by default to the last axis of the data : the only dimension in 1-D, the horizontal in 2-D and the third axis (time axis) in 3-D. The coefficients are stored in a cell vector of length `N_scale`, containing all the coefficients of each scale in a single 1-D, 2-D or 3-D array.

B.10.2 Undecimated Orthogonal Wavelets – 1D-UWT

There are four functions available for the 1-D undecimated wavelet transform :

```
Coef = uwt1d_transform(Data, N_scale, [direction], [wavelet_type]);
Coef = uwt1d_threshold(Coef, threshold, [filtering_type]);
Cube2 = uwt1d_recons(Coef, [direction], [wavelet_type]);
Cube2 = uwt1d_filter(Data, N_scale, threshold, [direction], ...
                           [filtering_type], [wavelet_type]);
```

The `direction` parameter is set by default to the last axis of the data : the only dimension in 1-D, the horizontal in 2-D and the third axis (time axis) in 3-D. The coefficients are stored in a cell vector of length `N_scale`, containing all the coefficients of each scale in a single 1-D, 2-D or 3-D array the same size as the data.

B.11 3D transforms

As in 2-D, all transforms have the same four core functions : the forward transform, the inverse transform, the thresholding operator, and a filtering function. For an example of their use, the user can refer to section B.9.1 on 2-D decimated wavelets.

B.11.1 Decimated Orthogonal Wavelets – 3D-DWT

As in two dimensions, there are four functions available for the decimated 3-D wavelet transform :

```
Coef = dwt3d_transform(Cube, N_scale, [wavelet_type]);
Coef = dwt3d_threshold(Coef, threshold, [filtering_type]);
Cube2 = dwt3d_recons(Coef, [wavelet_type]);
Cube2 = dwt3d_filter(Cube, N_scale, threshold, ...
                           [filtering_type], [wavelet_type]);
```

The functions are used in a similar way as the 2-D case, and the coefficients have the same structure, $C\{\text{scale}\}\{\text{band}\}$, but in 3-D, there are seven bands per scale except for the approximation. The bands at scale s are 2^s times smaller than the input cube, except the approximation which has the size of the previous scale's bands.

B.11.2 Undecimated Orthogonal Wavelets – 3D-UWT

There are four functions available for the undecimated, translation invariant 3-D wavelet transform :

```
Coef = uwt3d_transform(Cube, N_scale, [wavelet_type]);
Coef = uwt3d_threshold(Coef, threshold, [filtering_type], [wavelet_type]);
Cube2 = uwt3d_recons(Coef, [wavelet_type]);
Cube2 = uwt3d_filter(Cube, N_scale, threshold, ...
                           [filtering_type], [wavelet_type]);
```

The coefficients are stored as $C\{\text{scale}\}\{\text{band}\}$, all bands having the size of the original datacube.

B.11.3 Fast Curvelets – 3D-FCT

The 3-D fast curvelet transform must be initialized with the `curvelet3d_params` function which returns a structure containing the default parameters of the transform, and which can be accessed and changed (see Section B.14 for details about this structure). It also contains the filtering properties, such as the type of thresholding (soft or hard), and the threshold to apply. A simple example is given after the prototypes of the functions :

```
Param = curvelet3d_params(Cube);
Coef = curvelet3d_transform(Image, Param);
Coef = curvelet3d_threshold(Coef, Param, [threshold]);
Cube2 = curvelet3d_recons(Coef, Param);
Cube2 = curvelet3d_filter(Cube, Param);
```

The coefficients are stored in a cell vector $C\{\text{scale}\}\{\text{direction}\}$, the last scale (the approximation) having only one sub-band.

Example of 3-D Fast Curvelet filtering

```
%% Transform - threshold - reconstruct approach
% Transform parameters
  Param = curvelet3d_params(Cube);
  Param.NScale = 4;
  Coef = curvelet3d_transform(Image, Param);

% First thresholding strategy : using Param
  Param.SigmaNoise = 5; % noise level of the data
  Param.NSigma = 3;     % thresholding will be at NSigma*SigmaNoise
  Coef = curvelet3d_threshold(Coef, Param);

% Second thresholding strategy :
%     giving the threshold directly, ingnoring Param
  Coef = curvelet3d_threshold(Coef, Param, 3*5);

% Reconstruction
  Cube2 = curvelet3d_recons(Coef, Param);

%% All in one Filtering : Param must be used
  Param = curvelet3d_params(Cube);
  Param.SigmaNoise = 5; % noise level of the data
  Param.NSigma = 3;     % thresholding will be at NSigma*SigmaNoise
  Cube2 = curvelet3d_filter(Image, Param, threshold, [filtering_type]);
```

B.11.4 BeamCurvelets – 3D-BCur

There are four functions available for the BeamCurvelet transform :

```
Coef = bcur3d_transform(Cube, N_scale, BlockSize, BlockOverlap);
Coef = bcur3d_threshold(Coef, threshold, [filtering_type]);
Cube2 = bcur3d_recons(Coef, NxCube, NyCube, NzCube, N_scale, ...
```

```
                        BlockSize, BlockOverlap)
Cube2 = bcur3d_filter(cube, NbrScale, BlockSize, BlockOverlap, ...
                        threshold, [filtering_type])
```

The coefficients are arranged in a cell vector of size `N_scale`. Each cell contains a cube of size `BlockSize` $\times$ `BlockSize` $\times P$, where `BlockSize` is the size of the block at the current scale, and the third dimension contains the P 2-D projections of all the blocks on all the possible angles : to a given block and angle correspond a slice of the cube, and this 2-D image is structured as a 2-D decimated wavelet as a mosaic.

B.12 Functions list

B.13 Notations and common parameters

The optional inputs and outputs are indicated in square brackets `[ ]`, and the default options are specified the same way.
Many of the routines have common multi-choice parameters, and for the sake of clarity we will describe them here.

Type of wavelet filter

```
[F]         Type of wavelet filter
            [1]: MALLAT 7/9
             2 : DAUBECHIES 4
             3 : BI2HAAR
             4 : BI4HAAR
             5 : ODEGARD 7/9
             6 : 5/3
             7 : LEMARIE 1
             8 : LEMARIE 3
             9 : LEMARIE 5
            11 : HAAR
```

B.14 Functions prototype and options

This part is deleted in the context of this Ph.D. thesis, as details of the options is not available to the public.