ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

EDITE - ED130

TELECOM ParisTech

2014-ENST-0006

Doctorat ParisTech

THÈSE

pour obtenir le grade de docteur délivré par

Télécom ParisTech

Spécialité : Signal et Images

Présentée et soutenue publiquement par

Thierry GUILLEMOT

le 3 Février 2014

Méthodes et structures non locales pour la restauration d'images et de surfaces 3D

Jury :

Mme. Julie DELON, Professeur, Université Paris Descartes	Présidente
M. Pierre ALLIEZ, Directeur de recherche, Inria Sophia-Antipolis	Rapporteur
M. Jean-Michel MOREL, Professeur, CMLA, Ecole Normale Supérieure de Cachan	Rapporteur
M. Cyril CRASSIN, Chercheur, Nvidia	Examinateur
Mme Pooran MEMARI, Chargé de Recherche, CNRS-LTCI, Télécom ParisTech	Examinatrice
M. Andrés ALMANSA, Chargé de Recherche, CNRS-LTCI, Télécom ParisTech	Directeur
M. Tamy BOUBEKEUR, Professeur, CNRS-LTCI, Télécom ParisTech	Directeur

THÈSE

TELECOM ParisTech
École de l'Institut Mines-Télécom - Membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

Remerciements

Tout d'abord, je tiens à remercier mes directeurs de thèse Andrès Almansa et Tamy Boubekeur pour leurs qualités pédagogique, scientifique mais surtout humaine. Grâce à eux, j'ai pu découvrir le monde de la recherche, apprendre beaucoup sur le traitement des images et des données 3D. Leur soutien, gentillesse et optimisme m'ont permis de travailler dans un cadre idéal pour réaliser ce doctorat.

Je remercie les membres du Jury, Pierre Alliez, Jean-Michel Morel, Cyril Crassin, Julie Delon et Pooran Memari d'avoir accepté d'évaluer mon travail ainsi que pour leurs remarques et conseils.

Mais que serait une thèse sans les soutiens de ces collègues devenus des amis au fils des années ? Je pense ici aux amis du bureau C07 (ou "Bureau Link") : Baptiste et Cécilia soutiens infaillibles depuis les premiers jours de stage, Alasdair pour sa touche anglaise et son optimisme, Yann pour la sagesse de l'ancien et son goût de la bonne bière et Guillaume pour sa gloutonnerie et ces blagues qui ne font rire que lui (et Alasdair). Je pense également aux autres collègues de TSI, les sages Noura et Jean-Marc pour leurs innombrables conseils et soirées improvisées, Charline, Emilie et Loredana pour leur inflexible bonne humeur, Guillaume pour ses remotivations ludiques, Edoardo pour son amour de la cuisine, David pour son amour de la bière, Joseph pour sa passion des débats, Julien pour son accent américain, Pooran pour sa gentillesse, ainsi que tous les autres Malik, Hélène, Stéphane, Flora, Sonia, Flora, Beibei, Leila, Mathias et Bert pour leurs discussions et conseils.

Je souhaite également remercier les anciens collègues pour m'avoir acceuilli dans les premiers jours de thèse : Charles, Vincent et plus particulièrement Benoit pour son humour, sa passion du monde ludique, son immense culture... et ses talents d'improvisation. Je voudrais aussi remercier Andrés, Tamy, Isabelle, Yann, Julie, Florence et Michel pour m'avoir donné l'envie de faire une thèse quand j'étais étudiant et pour leur accueil au laboratoire TSI.

Avant de terminer, je voudrais exprimer ma gratitude à tous mes amis. Les plus Vieux : Ludovic, Tom, Fabien, Sandrine et Mylène, les plus Récents : Cédric, Marine, Sophie et Valérie, les plus Toulousains : Delphine et Manu, les plus Anglais : Terry, Cécile et Laurie, les plus Théatreux : Marie, Cyril, Anne, Benoit, Florian, Bianka, Emmanuelle, Alban et Sophie et les plus Télécommiens : Alexis, Romane, Rémi, Chloé, Laure et Morgane qui ont toujours été là pour moi, même dans les moments les plus difficiles.

Je voudrais terminer ces remerciements en exprimant ma gratitude à mes proches : mon père, ma mère et mes deux soeurs Karine et Coralie qui m'ont toujours encouragé, aidé et soutenu pendant toutes ces années. A vous tous, sans qui rien de tout cela n'aurait été possible, merci !

Je dédie cette thèse à mes neuveux et nièces à qui je souhaite d'avoir autant de soutien, d'aide et d'encouragement que ce que j'ai pu avoir jusqu'à présent pour réussir ce qu'ils entreprendront.

Table des matières

Table des matières iii

1 Introduction 1
- 1.1 Contextes . . . 1
- 1.2 Du Local au Non Local . . . 3
  - 1.2.1 Méthodes de restauration locales . . . 3
  - 1.2.2 Méthodes de restauration non locales . . . 4
- 1.3 Problématiques . . . 4
- 1.4 Contributions . . . 5
- 1.5 Organisation de la thèse . . . 5
- 1.6 Prix et Publications . . . 6

2 Tour d'horizon des méthodes de restauration non locales 7
- 2.1 Contexte . . . 7
- 2.2 Filtres à moyennes non locales . . . 11
  - 2.2.1 Formulations . . . 11
  - 2.2.2 Limitations et améliorations des NL-Means . . . 12
- 2.3 Filtres 3D non locaux . . . 14
  - 2.3.1 Filtres surfaciques non locaux . . . 15
  - 2.3.2 Filtres non locaux de nuage de points . . . 15
  - 2.3.3 Limitations . . . 16
- 2.4 Filtres collaboratifs . . . 17
  - 2.4.1 Principe . . . 17
  - 2.4.2 Méthodes de restauration collaboratives . . . 18
  - 2.4.3 Généralisation des filtres collaboratifs . . . 19

3 Surface de points non locale 21
- 3.1 Contexte . . . 21
  - 3.1.1 Acquisition 3D . . . 21
  - 3.1.2 Travaux existants . . . 21
  - 3.1.3 Généralisation des PSS . . . 23
  - 3.1.4 Problématique . . . 24
- 3.2 Surfaces de points non locales . . . 25
  - 3.2.1 Principe général . . . 25
  - 3.2.2 Carte de déplacement . . . 26
  - 3.2.3 Fonction de pondération . . . 26
  - 3.2.4 Opérateur de projection non local . . . 29
- 3.3 Résultats . . . 30
  - 3.3.1 Détails d'implémentation et performance . . . 30
  - 3.3.2 Analyse des paramètres . . . 31
  - 3.3.3 Performances . . . 41

3.3.4 Analyse de la qualité de la surface . . . 42
3.3.5 Applications . . . 46
3.4 Limitation et possibles améliorations . . . 47

4 Arbre de covariances **51**
4.1 Contexte . . . 51
4.1.1 Filtrage à hautes dimensions . . . 51
4.1.2 Travaux existants . . . 52
4.1.3 Problématique . . . 53
4.2 Arbre de Covariance Simplifié . . . 54
4.2.1 Principe . . . 54
4.2.2 Construction . . . 58
4.2.3 Apprentissage . . . 60
4.2.4 Requête . . . 62
4.3 Généralisation . . . 64
4.3.1 Notations . . . 64
4.3.2 CovTree généralisé . . . 65
4.3.3 Algorithme généralisé . . . 67
4.4 Analyse et temps de calcul . . . 69
4.4.1 Complexité . . . 71
4.4.2 Discussion . . . 71
4.5 Limitations et possibles améliorations . . . 72

5 Filtrage collaboratif généralisé **75**
5.1 Débruitage d'images par filtre collaboratif . . . 75
5.1.1 Non Local Bayes . . . 76
5.1.2 Débruitage d'images par CovTree . . . 78
5.1.3 Analyse des paramètres . . . 79
5.1.4 Résultats . . . 86
5.2 Restauration par dictionnaires . . . 88
5.2.1 Principe général . . . 89
5.2.2 Débruitage d'images . . . 90
5.2.3 Reconstruction d'image échantillonnée aléatoirement . . . 92
5.2.4 Limitations et possibles améliorations . . . 93
5.3 Restauration de surface 3D . . . 94
5.3.1 Positionnement du problème . . . 94
5.3.2 Surface de points par NLB . . . 95
5.3.3 Restauration de surface par dictionnaire . . . 96

6 Conclusion **99**

7 Perspectives **101**

Bibliographie **103**

CHAPITRE 1

Introduction

1.1 Contextes

Les technologies à base de semi-conducteurs ont évolué de façon significative au début des années 90 et ont permis l'arrivée de nouveaux systèmes d'acquisition numérique. Cependant, leur qualité de capture était à l'origine trop limitée par rapport aux systèmes analogiques. L'augmentation de la résolution des capteurs est donc rapidement devenue un enjeu crucial de cette décennie. C'est à cette époque qu'apparaissent les premiers appareils photographiques entièrement numériques : les pellicules et leur développement sont respectivement remplacées par des cartes de stockages réutilisables et des impressions instantanées. Les appareils numériques réussissent ainsi à s'imposer progressivement face aux analogiques.

Les précédentes avancées ont permis le développement d'autres techniques dans des domaines différents de la photographie. De nouveaux systèmes de capture ont ainsi émergés tels que les scanners 3D. Les systèmes de capture 3D classiques, estimant la profondeur par contact, sont ainsi améliorés par l'utilisation de capteurs CCD et de Laser. Ce nouveau type de scanner est particulièrement employé dans des domaines tels que la conservation du patrimoine (par ex. *project Michelangelo* - Figure 1.1) où il est nécessaire de numériser des objets sans qu'ils ne subissent de détériorations. Cependant la nature des données 3D diffère de celle des images 2D. Des outils spécifiques à leur traitement ont donc été introduits et définissent ainsi les toutes premières chaines d'acquisition 3D.

À la fin des années 90, les capteurs numériques réussissent à atteindre des résolutions équivalentes ou supérieures à celles obtenues par des systèmes classiques. Malheureusement, l'utilisation des appareils numériques reste réservée à une élite. Ce phénomène peut être expliqué par plusieurs facteurs : un prix d'achat trop élevé, et des systèmes de capture qui ne sont généralement pas adaptés aux besoins du grand public (difficulté d'utilisation, encombrement, poids élevé, etc.). Ainsi, l'adaptation des systèmes d'acquisition numérique à ces besoins est devenu l'enjeu principal des années 2000.

Cette démocratisation est particulièrement visible avec l'apparition des appareils photos *compacts*. L'utilisation de nouveaux algorithmes de gestion automatique des paramètres intrinsèques (focus, ouverture, balance des blancs, etc.) les rendent facile d'utilisation pour les non-connaisseurs. La photographie numérique devient donc accessible à tous à partir

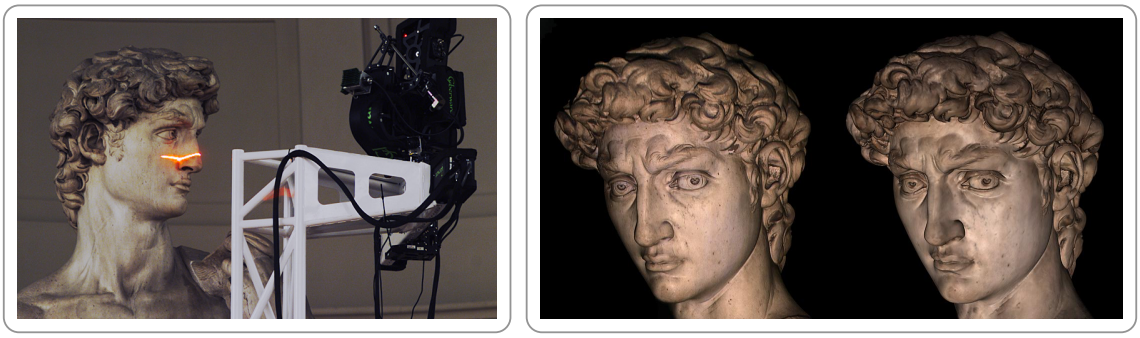


FIGURE 1.1: Résultat du projet *Michelangelo* : *Gauche* : numérisation 3D sans détérioration utilisant des capteurs CCD et Laser. *Droite* : comparaison entre la statue réelle et le résultat 3D obtenu. Les images proviennent de [LPC+00]

du début des années 2000. À partir de là, de nombreux constructeurs se sont mis à développer des appareils de plus en plus performants à des prix toujours plus bas afin de mieux satisfaire les demandes de ces nouveaux utilisateurs. Au milieu des années 2000, de nouvelles avancées technologiques permettent de produire des capteurs miniaturisés à l'extrême pour des coûts dérisoires. Désormais plus petits qu'une pièce de monnaie, ils se retrouvent rapidement intégrés aux téléphones portables et bouleversent par la même occasion les habitudes d'utilisation du grand public. La résolution et les performances de capture (zoom, iso etc.) deviennent progressivement des arguments de vente incontournables.

Pendant ces années 2000, les systèmes de captures 3D ne cessent d'évoluer. De nouveaux types de capteurs font leur apparition en proposant l'exploitation de plusieurs images par stéréovision, la projection de motifs connus par lumière structurée, l'analyse du focus d'un appareil photo ou, plus récemment, les capteurs plénoptiques. Pendant cette période, l'utilisation des capteurs 3D se généralise à de nombreux domaines tels que le cinéma, les jeux vidéo, la robotique, l'automobile, etc. . Durant plusieurs années, l'utilisation de ces scanners 3D reste néanmoins réservée à un public restreint. Ce n'est qu'à partir du début des années 2010 que les capteurs 3D grand public sont démocratisés par l'industrie des jeux vidéos (par ex. Microsoft Kinect). Grâce à leur faible prix, il devient facile d'acquérir des données 3D. Cela permet d'attirer de nouveaux utilisateurs et d'élargir les types d'usage de ces données.

À l'heure actuelle, la démocratisation des capteurs numériques est telle, qu'en France, des millions d'appareils photos numériques sont vendus chaque année, environ la moitié de la population possède un capteur photo sur son téléphone (*smartphone*) et environ un million de personnes possèdent une Kinect. Ainsi, chaque jour, une quantité astronomique de données 2D et 3D est ainsi générée à travers le monde. Avec l'essor des réseaux de communications et avec l'apparition des *smartphones*, une grande partie de cette information se retrouve partagée avec le reste du monde. En guise d'exemple, des dizaines de milliards de photos sont échangées chaque année sur l'un des principaux réseaux sociaux actuels (Facebook).

Désormais, la majorité des systèmes d'acquisition numérique modernes sont connectés aux réseaux de communications (3G, 4G en préparation, etc.). Ils peuvent donc trans-

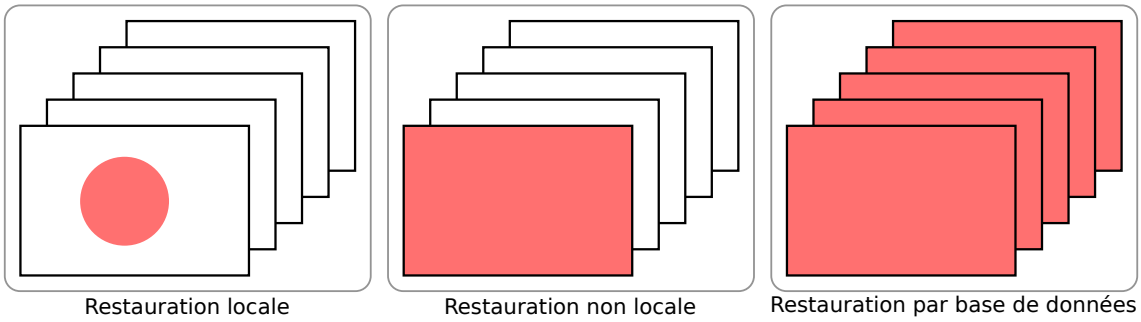


FIGURE 1.2: Illustration des différentes méthodes de restauration. Nous représentons en rouge le voisinage retenu.

mettre très facilement l'information acquise. Aujourd'hui, ces données partagées restent néanmoins difficilement exploitables par les machines de calculs modernes (ordinateur personnel, *smartphone*, etc.) dans la mesure où elles ne disposent pas d'une capacité de mémoire suffisante pour pouvoir les gérer. Ainsi, les différentes évolutions récentes laissent à penser que cette exploitation sera le nouveau challenge de ces années 2010.

Cette thèse s'inscrit au cœur de cette évolution en proposant quelques éléments de réponse à l'un des enjeux technologiques des années 2010, à savoir : la capture, transmission et traitements de masses de données 2D et 3D. Après cette brève introduction sur l'historique de la capture numérique, nous introduisons les enjeux de la restauration moderne.

1.2 Du Local au Non Local

Durant ces dernières années, les technologies d'acquisition numériques n'ont cessées de se perfectionner. Ces différentes évolutions ont donc permis l'apparition de capteurs miniaturisés, économes en énergie et surtout capables d'acquérir des données avec une qualité toujours plus fine. Malgré ces différentes améliorations, les capteurs restent sujet à des perturbations (tel que bruit électronique, résonance magnétique, etc.). Ces problèmes électroniques ajoutés aux contraintes d'acquisition (distorsions, effets de masquage, etc.) viennent perturber la capture entrainant l'apparition de défauts tels que du bruit, des points parasites, des trous dans les données acquises, etc.

Ces imperfections sont inhérentes à toute technologie de capture et ne peuvent pas être corrigées matériellement. Elles nécessitent donc un traitement logiciel particulier. Ainsi, en même temps que l'apparition des tout premiers capteurs numériques dès le début des années 90 apparaissent les premières méthodes de restauration telles que le débruitage, la reconstruction, le suréchantillonnage, etc.

1.2.1 Méthodes de restauration locales

Jusqu'au milieu des années 2000, ces approches s'appuient uniquement sur un traitement local des données (Figure 1.2). Dans le cadre du traitement des images, cette localité s'exprime de deux manières différentes :

Durant les années 90, la puissance de calcul et la capacité de stockage en mémoire restant limitées, seul des filtres simples pouvaient être évalués. Ainsi, le support du filtre a été restreint à un support local : un pixel est filtré par rapport aux pixels appartenant à un voisinage local. C'est à cette époque que sont introduits les filtres par minimisation de la variation totale, bilatéraux, etc..

Avant le milieu des années 2000, les réseaux de communications ne permettent pas d'échanger une quantité de donnée importante. Ainsi, les approches de restauration sont réduites à l'utilisation d'une seule image : un pixel est filtré à partir de l'information contenue dans l'image à filtrer.

1.2.2 Méthodes de restauration non locales

À partir du milieu des années 2000, les performances de calcul et la quantité de mémoire présentes sur les ordinateurs ont suffisamment évolué pour que les limitations locales précédentes soient supprimées.

Dans un premier temps, les méthodes non locales proposent d'étendre le support local du filtre de restauration à l'ensemble de l'image en faisant l'hypothèse que les données acquises sont autosimilaires (Figure 1.2). Ainsi, l'information détériorée est restaurée en utilisant des données similaires présentes en d'autres endroits de l'image. Face à leur succès en traitement des images, ces méthodes sont progressivement étendues au cas 3D où l'hypothèse d'autosimilarité peut être supposée de manière équivalente.

Dans un deuxième temps, de nouvelles méthodes plus générales ont permis de combiner les idées introduites par les méthodes non locales avec la multiplication de la quantité de données (Figure 1.2). La non localité est désormais définie par rapport à une information extraite à partir de plusieurs images.

1.3 Problématiques

S'il est vrai qu'il y a eu un développement récent des méthodes non locales, ces dernières ont principalement été utilisées afin de restaurer des données régulières et structurées telles que des images. À l'heure actuelle, ces méthodes restent encore peu étendues dans le cadre de données irrégulières. Dans le cas extrême des nuages de points 3D qui sont à la fois irréguliers et non structurés, il n'existait au début de cette thèse aucune méthode de restauration capable d'exploiter la redondance de l'information judicieusement afin de définir une surface 3D.

Avec la démocratisation des capteurs numériques, la quantité de données transmise sur les réseaux de communication ne cesse d'augmenter. Le challenge est donc de réussir à exploiter judicieusement ces grands ensembles d'information. Néanmoins, à l'heure actuelle seules quelques grosses entreprises possèdent les infrastructures de calcul et de stockage nécessaires pour y arriver. Bien évidémment, le grand public ne peut avoir à sa disposition de telles structures. Il faut donc réussir à trouver un moyen pour que cette exploitation puisse être réalisée sur des systèmes possédant des capacités limitées (par ex. *smartphones*).

Malheureusement, les données accessibles sont généralement brutes, semi-structurées ou non structurées. Il devient donc nécessaire de trouver un moyen pour les regrouper.

Les évolutions récentes des filtres non locaux tendent à incorporer une problématique orientée *big data* en proposant des nouvelles démarches qui analysent l'information de sorte à extraire un modèle pour améliorer la restauration des données. Malheureusement, ce type de méthodes reste peu utilisable en pratique à cause du temps qu'il nécessite pour effectuer l'analyse. Il devient donc nécessaire d'introduire des nouvelles méthodes non locales qui soient à la fois flexibles pour pouvoir être utilisées sur n'importe quel type de données et génériques pour être utilisables par une large variété d'applications.

1.4 Contributions

Le point de départ de cette thèse est le débruitage non local de nuages de points. Ce problème a mis en évidence le besoin de structures dédiées aux traitements non-locaux des données de dimension deux et supérieure, ainsi que la possibilité de généraliser les méthodes existantes de restauration non-locale 2D et 3D.

Les principales contributions de cette thèse sont :
- une définition de surface de points capable d'exploiter le caractère autosimilaire d'un nuage de points. Cette définition permet d'étendre n'importe quelle définition de surface de points locales précédente.
- une structure de données flexible et générique capable d'apprendre les distributions d'un grand ensemble d'échantillons avec une capacité de mémoire limitée. Cette structure permet de définir pour tout voisinage, la gaussienne anisotrope associée aux échantillons qui sont compris dans ce voisinage.
- une généralisation des méthodes de restauration par filtres collaboratifs appliquées aux données 2D et 3D qui utilisent la structure de données introduite précédemment pour apprendre un modèle statistique a priori à partir d'un grand ensemble de données.

1.5 Organisation de la thèse

Dans le Chapitre 2, nous introduisons de manière générale les méthodes de restauration non locales. Au travers ce chapitre, nous présenterons les méthodes de débruitage par moyenne non locale, leurs applications aux cas 3D ainsi qu'une analyse générale des méthodes collaboratives.

Le Chapitre 3 sera consacré à l'adaptation de la méthode de filtrage par moyennes non locales au cas des nuages de points.

Dans le Chapitre 4, nous introduisons une nouvelle structure capable d'apprendre rapidement une grande base de données en utilisant uniquement une quantité de mémoire limitée.

Cette structure sera utilisée au Chapitre 5 pour résoudre différents problèmes de restauration collaboratifs.

Finalement, le Chapitre 6 offrira une conclusion générale des contributions de cette thèse et le Chapitre 7 une ouverture générale des travaux de recherche à venir.

1.6 Prix et Publications

Covariance Tree for 2D and 3D Processing, *Thierry Guillemot, Andrés Almansa et Tamy Boubekeur*, Computer Vision and Pattern Recognition (CVPR), Columbus, 2014

Non Local Point Set Surface, *Thierry Guillemot, Andrés Almansa et Tamy Boubekeur*, 3D Imaging Modeling Processing Visualization Transmission (3DIMPVT), Zurich, 2012

Non Local Point Set Surface, *Thierry Guillemot, Andrés Almansa et Tamy Boubekeur*, Symposium on Geometry Processing (SGP), Tallinn, 2012 - **Best Poster Award**

CHAPITRE 2

Tour d'horizon des méthodes de restauration non locales

Dans ce chapitre, nous allons introduire un état de l'art des méthodes non locales. Nous commençons par présenter un contexte général de la restauration des images ainsi qu'un historique des méthodes de débruitage par moyenne de valeurs. Cet historique est suivi d'une analyse des méthodes non locales qui se concentrera sur l'influence des paramètres, les limitations et les évolutions des méthodes non locales. Nous présentons ensuite, une généralisation de ces méthodes au cas 3D en nous focalisant sur le débruitage surfacique et de nuage de points. Finalement, nous introduisons les filtres collaboratifs ainsi que leurs différentes applications en traitement des images. Durant ce chapitre, nous introduisons les notions indispensables qui seront utilisées durant l'ensemble de cette thèse. Nous précisons au lecteur que des analyses de l'état de l'art plus spécifiques aux travaux développés durant cette thèse seront présentées en début des Chapitres 3, 4 et 5.

2.1 Contexte

Quel que soit le système d'acquisition et la précision de ce dernier, les signaux modernes restent altérés lors de leur capture. Ces dégradations, généralement dues aux contraintes matérielles (optique, capteur CCD, etc.), algorithmiques (compression JPEG, etc.) ou transmissives (satellitaires, internet, etc.), doivent être traitées afin de récupérer au mieux l'information originalle contenue dans ces signaux. Dans cette partie, nous considérerons des images discrètes dont chaque élément (pixel) est associé à une couleur (Rouge, Vert, Bleu - RGB).

Plusieurs hypothèses sont généralement faites afin de modéliser les dégradations subies par l'image. En particulier, nous supposerons que nous avons accès à une observation dégradée de l'image qui associe à chaque pixel $\mathbf{x}_i$ une couleur $f(\mathbf{x}_i)$ obtenue à partir de la couleur réelle $u(\mathbf{x}_i)$ en ajoutant un bruit $n(\mathbf{x}_i)$ gaussien *i.i.d.* (indépendant et identiquement distribué) de variance σ connue :

$$\mathbf{f}_i = f(\mathbf{x}_i) = u(\mathbf{x}_i) + n(\mathbf{x}_i) \tag{2.1}$$

Ce modèle est utilisé de manière standard dans la communauté du traitement des images, car il est à la fois simple et efficace pour modéliser le bruit dû à l'acquisition de l'image.

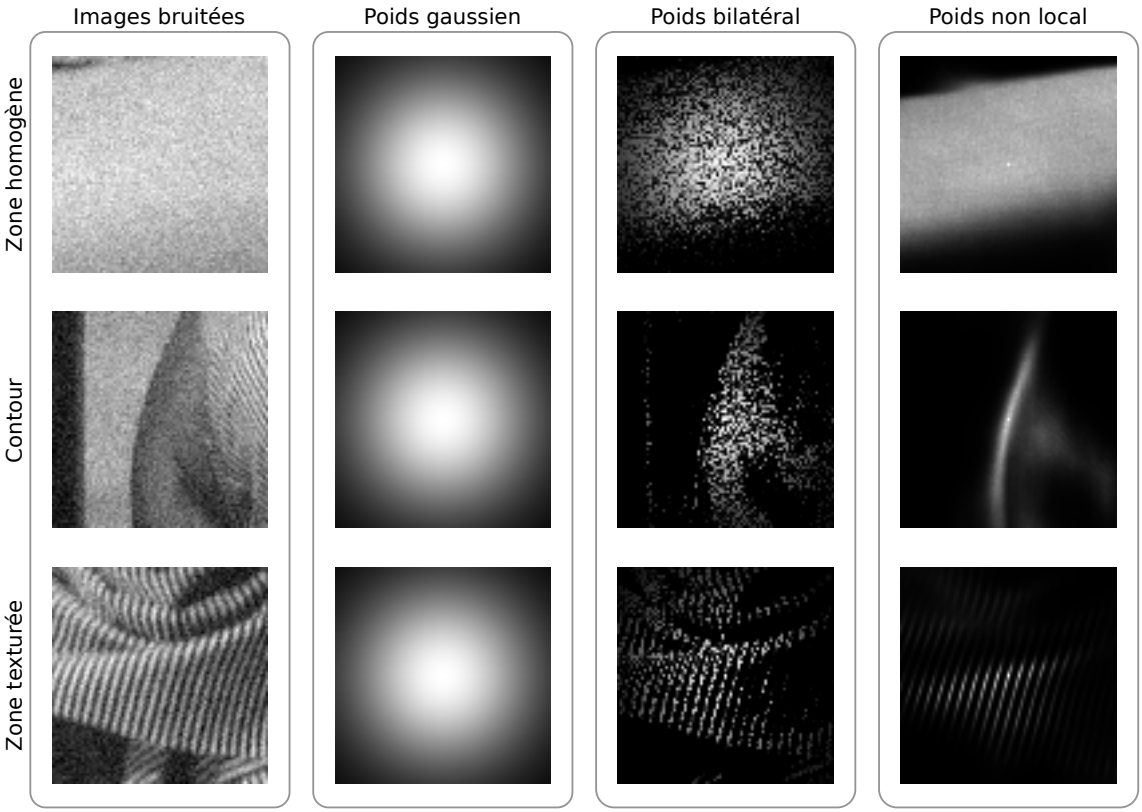


FIGURE 2.1: Nous illustrons la définition des poids gaussien, bilatéral et non local pour trois différentes zones de l'image : une zone homogène, un contour, une zone texturée. Pour tout ces exemples, les poids sont calculés en fonction du pixel central de chaque image. Les poids élevés sont représentés en blanc tandis que les poids faibles sont représentés en noir.

D'autres modèles, par exemple les modèles poissoniens, ne seront pas abordés dans cette thèse (voir [Bov05] pour plus de détails).

Le principe des méthodes de débruitages consiste à éliminer le bruit ajouté lors de l'acquisition afin d'estimer une couleur $\hat{u}(\mathbf{x}_i)$ proche de la couleur originale $u(\mathbf{x}_i)$. Pour résoudre ce problème, des nombreux travaux ont été introduits proposant une minimisation de la variation totale [ROF92], l'application de filtres de Wiener empiriques [YY85], une décomposition de l'image en ondelettes [Don95], l'utilisation de voisinages de taille variable [PS00, KEA02, KFEA04, FKEA04], ou plus récemment une analyse de l'a priori des textures de l'image par dictionnaire [DFKE09, ZDZS10, YSM10], etc..

Parmi toutes ces méthodes, un grand nombre de filtres peuvent s'exprimer comme une moyenne sur les pixels de l'image. En considérant un sous-ensemble de pixels $N(\mathbf{x}_i)$ proches du pixel d'intérêt $\mathbf{x}_i$, la forme générale de ces filtres peut être exprimée comme une combinaison des couleurs $\mathbf{f}_j$ pondérées par les poids w_{ij} :

$$\hat{u}(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} w_{ij} \mathbf{f}_j \tag{2.2}$$



FIGURE 2.2: Résultat des différents filtrages gaussien, bilatéral et non local appliqués sur l'image *barbara* bruitée avec un bruit gaussien de variance 0.08.

Les poids w_{ij} sont normalisés : $\sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} w_{ij} = 1$, et leur définition permet de définir les caractéristiques du filtre souhaité. Ce dernier peut dépendre de la position spatiale $\mathbf{x}_i$, de la couleur des pixels $\mathbf{f}_i$, ou de toute autre caractéristique de l'image. En particulier, l'Équation 2.2 peut être utilisée pour exprimer la convolution par un noyau gaussien ϕ d'écart type h_p. Dans ce cas, les poids w_{ij} sont fonction de l'éloignement spatial entre les pixels de l'image $\mathbf{x}_j$ et le pixel d'intérêt $\mathbf{x}_i$:

$$w_{ij} \propto \phi(\|\mathbf{x}_i - \mathbf{x}_j\|^2 / h_p^2) \tag{2.3}$$

Une analyse du filtre gaussien [BCM05] démontre que l'écart entre l'image originale et l'image filtrée est directement proportionnel au laplacien de l'image. Il en résulte que le filtre gaussien permet de récupérer correctement les valeurs des zones homogènes de l'image, mais floute les bords et les zones texturées (Figure 2.1 et 2.2). Par conséquent, à cause de ces faibles performances, ce filtre est assez peu utilisé par les systèmes de débruitage modernes.

La simplicité de la formulation du filtre gaussien restant très attrayante, de nombreux travaux ont proposé d'améliorer ses performances afin d'obtenir un débruitage des images plus efficace. En proposant une approche similaire aux travaux de Lee [Lee83] et de Yaroslavsky [YY85], Tomasi introduit le filtre bilatéral [TM98] en définissant un poids dépendant à la fois de la distance spatiale entre les pixels et de la distance entre leurs couleurs associées (distance colorimétrique). D'une manière totalement équivalente, le filtre bilatéral peut être vu comme un filtre gaussien défini dans un espace augmenté qui mélange les positions des pixels $\mathbf{x}_i$ et leurs valeurs associées $\mathbf{f}_i$. Les poids du filtre sont calculés en combinant deux noyaux ϕ, ψ définis respectivement dans les domaines spatial (différence des positions des pixels) et colorimétrique (différences des couleurs des pixels) :

$$w_{ij} \propto \phi(\|\mathbf{x}_i - \mathbf{x}_j\|^2/h_p^2)\psi(\|\mathbf{f}_i - \mathbf{f}_j\|^2/h_f^2) \tag{2.4}$$

Ces noyaux, généralement gaussiens, sont dépendants de deux paramètres h_p et h_c utilisés pour limiter la taille du voisinage spatial et colorimétrique. Désormais seuls les pixels qui possèdent à la fois des positions et des couleurs proches du point d'intérêt ont un poids élevé. Par exemple, les pixels proches d'un contour appartenant à une zone différente de celle du pixel d'intérêt possèdent dans ce cas un poids négligeable. Le filtre bilatéral permet de débruiter correctement les zones homogènes tout en préservant les bords de l'image. Néanmoins, l'utilisation d'une unique valeur pour estimer un voisinage colorimétrique reste trop sensible au bruit de l'image (Figure 2.1 et 2.2). Les zones texturées de l'image restent donc difficilement récupérables par une telle approche.

Face au succès des méthodes de synthèse de texture [EL99] et d'*inpainting* [CPT03,CPT04] utilisant la notion de patch (correspondant à une imagette de l'image considérée), Buades et coll. [BCM05] (et en même temps [AW05,KB06,AW06]) introduisent le filtre à moyennes non local (*Non Local Means* - NL-Means). Ce dernier étend l'idée du filtre bilatéral d'utiliser la couleur des pixels pour raffiner l'estimation du voisinage en la remplaçant par un *patch* (ou *imagette*) décrivant l'information contenue autour du pixel. Ce patch $\mathbf{P}^W_{f,\mathbf{x}_i} \in \mathbb{R}^{d_\mathbf{P}}$ de dimension $d_\mathbf{P} = (2W+1)^2$ est défini comme l'ensemble des valeurs de f comprises dans une fenêtre carrée de taille W centrée sur le pixel considéré $\mathbf{x}_i$. Les poids non locaux w_{ij} sont donc définis par l'intermédiaire d'un noyau de pondération ϕ dépendant de la distance entre patchs :

$$w_{ij} \propto \quad \phi(\|\mathbf{P}^W_{f,\mathbf{x}_i} - \mathbf{P}^W_{f,\mathbf{x}_j}\|^2/h^2) \propto \quad \phi(\sum_{k=1}^{d_\mathbf{P}} \|\mathbf{f}_{i+k} - \mathbf{f}_{j+k}\|^2/h^2) \tag{2.5}$$

Le paramètre h permet de définir le nombre de voisins utilisés pour le débruitage. Désormais le poids dépend uniquement de la similarité entre les patchs et non de l'éloignement spatial entre les pixels (d'où son caractère *non local*). Il est intéressant de noter qu'une version dégénérée des NL-Means, où la taille du patch est réduite à un unique pixel, nous permet d'obtenir une forme équivalente au filtre bilatéral. Tout comme ce dernier, le filtre NL-Means peut être exprimé de manière équivalente comme un filtre gaussien appliqué dans l'espace des patchs. Par l'utilisation d'un ensemble de couleurs $\{\mathbf{f}_{i+k}\}_{k\in[\![1,d_\mathbf{P}]\!]}$, les poids définis par les NL-Means sont moins dépendants du bruit et permettent d'obtenir un débruitage de l'image satisfaisant dans les zones homogènes, les contours et les zones texturées de l'image (Figure 2.1 et 2.2). Cette propriété est due au fait que les images naturelles présentent une auto similarité affine locale [Ale05]. De par leur formulation simple et leur capacité de débruitage, les NL-Means sont devenus durant ces dernières années l'un des filtres les plus utilisés pour le débruitage d'images.

2.2 Filtres à moyennes non locales

Dans cette partie, nous effectuons une étude plus approfondie des NL-Means. Nous nous intéresserons en particulier à l'analyse de ses différents paramètres, évolutions et limitations. Lebrun et. coll. [LCBM12] fait une analyse en profondeur des propriétés mathématiques de ces filtres et ses généralisations en traitement d'images.

2.2.1 Formulations

Nous rappelons ici la définition des NL-Means telle qu'elle a été introduite par Buades et coll. [BCM05] :

$$\hat{\mathbf{u}}_i = \sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} \phi(\|\mathbf{P}^W_{f,\mathbf{x}_i} - \mathbf{P}^W_{f,\mathbf{x}_j}\|^2/h^2)\mathbf{f}_j \tag{2.6}$$

où $N(\mathbf{x}_i)$ est un ensemble de pixels voisins de $\mathbf{x}_i$, ϕ un noyau de pondération, W la taille des patchs $\mathbf{P}^W_{f,\mathbf{x}_i}$ centrés sur $\mathbf{x}_i$ et h le facteur d'autosimilarité. Tous ces paramètres sont directement liés à la qualité de la restauration. De nombreux travaux ont proposé une analyse détaillée de leur influence respective.

Noyau de pondération

Le noyau de pondération ϕ permet de définir le poids de chaque échantillon en fonction de la similarité de leurs voisinages. La pondération est normalisée de sorte que :

$$\sum_{\mathbf{x}_j \in N(\mathbf{x}_i)} \phi(\|\mathbf{P}^W_{f,\mathbf{x}_i} - \mathbf{P}^W_{f,\mathbf{x}_j}\|^2/h^2) = 1 \tag{2.7}$$

Les travaux originels des NL-Means [BCM05, AW05, KB06, AW06] utilisent un noyau de pondération gaussien. Chaque pixel de la somme possède un poids non nul qui influence l'estimateur final. L'accumulation des poids des pixels les plus éloignés peut par conséquent biaiser le débruitage. Par conséquent, n'importe quel point, même le plus éloigné, influence l'estimateur non local. Pour résoudre ce problème, de nombreux travaux [GLPP08, DAG10, Sal10a] arrivent à la conclusion qu'il est préférable d'utiliser un noyau à support compact qui permet d'annuler l'influence des échantillons trop différents.

Dans l'Équation 2.6, le patch $\mathbf{P}^W_{f,\mathbf{x}_i}$ possède le double rôle de patch de référence (utilisé pour définir la similarité entre deux patchs) et de patch estimateur (combiné pour définir la moyenne). Ce pixel particulier possédera donc un poids plus important que les autres pixels de l'image biaisant par la même occasion l'estimateur non local. Pour résoudre ce problème, Buades et coll. [BCM05] suggèrent de le remplacer par le poids maximal des autres pixels utilisés dans la somme. Cette solution n'est pas vérifiée théoriquement, mais permet d'obtenir de bons résultats pratiques. Zimmer et coll. [ZDW08] proposent d'annuler l'influence de ce patch particulier. Néanmoins, une telle solution ne permet pas de concilier le double rôle du patch central. Salmon [Sal10b] résout ce problème en supprimant la variance du bruit dans la distance euclidienne utilisée pour comparer les patchs. Ainsi, tout patch qui possède une distance inférieure à la variance du bruit possède un poids égal à 1.

Facteur d'autosimilarité

Le facteur d'autosimilarité h définit le degré de similarité des échantillons. Il est choisi de manière globale et peut être assimilé à la quantité de flou des méthodes locales. Le choix d'une valeur globale pour ce facteur reste dfficile à choisir, en effet certaines textures de l'image sont plus redondantes que d'autres. Ce paramètre est donc généralement estimé par une analyse heuristique sur un ensemble représentatif d'images. Des estimations de h basées sur l'estimateur SURE (*Stein's Unbiased Risk Estimate*) sont néanmoins calculées globalement par Van et coll. [VDVK09] et localement par Duval et coll. [DAG10].

Taille des patchs

La taille des patchs W correspond au niveau de détail à utiliser pour définir la similarité entre deux patchs. Augmenter l'échelle W permet de définir une mesure de similarité plus robuste et moins dépendante du bruit. Malheureusement, cela limite le nombre de patchs similaires utilisés pour le débruitage et augmente le temps de calcul des NL-Means.

Par conséquent, la taille W doit être idéalement choisie en fonction de la quantité de bruit et de l'image à traiter. Généralement, elle est déterminée par une analyse statistique effectuée sur une base d'images représentatives.

Zones de recherche

Initialement, le voisinage $N(\mathbf{x}_i)$ a été introduit comme un moyen d'accélérer l'estimation des NL-Means [BCM05]. En effet, le temps de calcul est directement proportionnel au nombre d'échantillons utilisé par l'Équation 2.6. Nous pourrions penser qu'augmenter la taille du voisinage améliorerait la qualité de restauration. En effet, nous obtiendrions dans ce cas un nombre de patchs similaires plus élevé. Malheureusement, nous augmenterions par la même occasion le nombre de mauvais candidats susceptibles de parasiter le résultat final. Pour résoudre ce problème, Kervrann et coll. [KB06] proposent de définir un voisinage adaptatif permettant de récupérer un plus grand nombre de patchs similaires sans introduire un trop grand nombre de mauvais représentants.

2.2.2 Limitations et améliorations des NL-Means

Les NL-Means sous leur forme originale présentent des limitations. De nombreux travaux ont été proposés pour enrichir le modèle de départ afin de corriger ces problèmes tout en améliorant les performances qualitatives et temporelles de la restauration.

L'un des problèmes principaux des NL-Means vient de la redondance des patchs. En effet, nous avons supposé que le patch était suffisamment bien représenté pour pouvoir être débruité correctement. Malheureusement, il arrive que certains pixels de l'image soient moins bien représentés que d'autres. Une première catégorie de méthodes propose entre autres d'augmenter le nombre de patchs similaires en utilisant le caractère autosimilaire local des images naturelles [Ale05]. Plusieurs approches [ZDW08, JCSX09] utilisent une analyse des moments robuste aux bruits afin de rendre les patchs invariants par rotation.

D'autres travaux [BCM05] suggèrent l'utilisation d'espace échelle afin de simuler l'invariance par changement d'échelles. Dans ce cas, les patchs calculés sur les différentes images sous-échantillonnées sont ajoutés à l'ensemble des patchs utilisé classiquement.

La mesure de similarité est faite sur un ensemble de patchs bruités. Idéalement, cette dernière devrait être effectuée sur des patchs non bruités. Ces données étant inaccessibles, Buades et coll. [BCM05] ont proposé l'utilisation de la distance euclidienne, car elle permet d'être peu dépendant du bruit de l'image. D'autres méthodes [BC07,BKC08] proposent de calculer la similarité sur les patchs débruités de manière itérative. Ces NL-Means itératifs débruitent l'image en utilisant une mesure de similarité calculée sur des patchs extraits de l'image qui a été débruitée à l'itération précédente. En d'autres termes, ce type d'approches permet un regroupement des patchs en fonction de la ressemblance du voisinage débruité à l'itération précédente.

À l'origine, les patchs utilisés sont centrés sur le pixel d'intérêt $\mathbf{x}_i$. Dans le cas d'un patch contenant un contour de l'image, la mesure de similarité risque d'être biaisée et donc de favoriser un côté du contour par rapport à l'autre. Ce phénomène *d'adhérence* introduit des poids parasites dans la combinaison finale des pixels. Il en résulte une apparition de *halos* autour des bords de l'image. D'autres méthodes [DFKE07, DFKE08, DFKE09, LBM13] proposent d'utiliser des patchs carrés décentrés pour corriger ces *halos* en reformulant l'Équation 2.6 d'une manière équivalente par :

$$\hat{\mathbf{P}}^W_{f,\mathbf{x}_i} = \sum_{\mathbf{x}_j \in N(\mathbf{p}_i)} \phi(\|\mathbf{P}^W_{f,\mathbf{x}_i} - \mathbf{P}^W_{f,\mathbf{x}_j}\|^2/h^2)\mathbf{P}^W_{f,\mathbf{x}_j} \tag{2.8}$$

L'estimation finale est définie en effectuant des moyennes sur les estimateurs [SS12]. Ce type de solutions permet d'étendre l'agrégation d'estimateurs [Nem00] au cas des patchs. Parmi ces méthodes, Deledalle, Salmon et Duval [DDS12] proposent en particulier l'utilisation d'un estimateur *SURE* (Estimateur de risque sans biais de Stein - Stein's Unbiased Risk Estimate) couplée à l'utilisation de patchs de différentes formes (non carrés). Cette idée (utilisée de manière similaire par Baraniuk et coll. [MNB12]) permet de tirer parti de la géométrie locale de l'image tout en étant peu victime du phénomène d'*adhérence*. Salmon [Sal10a] propose une étude plus détaillée concernant l'agrégation d'estimateurs dans le cas des NL-Means.

En plus de ces facteurs limitant la qualité de restauration de l'image, l'un des problèmes majeurs des NL-Means reste son temps de calcul. De nombreuses approches ont été proposées afin d'accélérer l'estimation. Certaines proposent une estimation approchée des plus proches patchs [BSFG09, OA12, KA11, Tas09, HS12], une approximation de l'image filtrée par regroupement de patchs similaires [AGDL09, ABD10, HST10, GO12], etc. Une analyse plus détaillée de ce point de l'état de l'art sera proposée en Section 4.1.1.

La simplicité et les performances des NL-Means ont favorisé la généralisation des filtres non locaux à de nombreux domaines connexes tels que l'imagerie satellitaire [DDT09], la vidéo [BCM08, AGDL09], l'imagerie médicale [MCCL+08], etc. Nous focaliserons la suite de notre analyse sur le traitement 3D.



FIGURE 2.3: Nous présentons plusieurs exemples de surface numérisée à partir d'objet réels d'origine humaine. Tous ces objets peuvent être considérés comme autosimilaires du fait qu'ils présentent un ensemble de structures redondantes. Une telle hypothèse peut être faite pour la plupart des objets d'orgines humaine ou naturelle.

2.3 Filtres 3D non locaux

Une chaîne d'acquisition 3D permet d'obtenir une surface maillée à partir d'un objet réel. Cette dernière peut être résumée par deux principales étapes : l'étape d'acquisition et l'étape de reconstruction. Tout d'abord, l'*étape d'acquisition* définit à partir de l'objet réel un nuage de points 3D. Pour ce faire, cette étape utilise généralement des capteurs automatiques (par ex. systèmes d'acquisitions *Kinect*), des appareils photo et/ou caméras vidéo couplés à des méthodes de stéréovision [BBH08, BBB+10, BHB+11], etc. Lors de cette étape, le nuage de points obtenu peut être dégradé par du bruit, des trous, des points aberrants et/ou des taux d'échantillonnages variables. Ensuite, l'*étape de reconstruction* permet de définir une surface maillée à partir du nuage de points acquis. Plusieurs méthodes peuvent être utilisées telles que les Marching Cubes [LC87], la reconstruction de Poisson [KBH06], etc.

Si elles ne sont pas traitées correctement, les dégradations introduites lors de l'étape d'acquisition parasitent la reconstruction de la surface maillée. Durant ces dernières années, les méthodes de débruitage ont donc été particulièrement utilisées afin de corriger les erreurs apportées par la chaîne d'acquisition. Nous pouvons diviser ces méthodes en deux principales catégories. La première catégorie, que nous appellerons *filtres de nuages de points* est appliquée sur le nuage de points 3D obtenu en amont de l'étape de reconstruction. Ces filtres permettent d'obtenir une surface maillée résultante de meilleure qualité. La deuxième, appelée *filtre surfacique*, est appliquée sur la surface maillée obtenue en aval de l'étape de reconstruction. Ces filtres sont utilisés dans le cas où l'utilisateur ne peut pas avoir accès à la chaîne d'acquisition.

La plupart de ces méthodes ont été inspirées des avancées en traitement des images [FDCO03, JDD03, VB11]. Des hypothèses similaires à celles utilisées pour les images peuvent en effet être faites concernant les données 3D. En particulier, ces dernières sont considérées de nature autosimilaire, car elles sont généralement issues d'objets réels d'origines naturelles ou humaines et possèdent de nombreuses primitives et structures redondantes (Figure 2.3). Les méthodes non locales ont donc logiquement été adaptées afin d'améliorer la qualité de la restauration 3D. Pour ce faire, ces dernières nécessitent la définition d'un **signal** à débruiter et d'**une mesure de similarité** adaptée aux données 3D considérées. Contrai-

rement aux images qui possèdent un échantillonnage régulier et une définition de signal bruité évidente, les données 3D ne possèdent pas de support pouvant être utilisé pour les deux définitions précédentes. Plusieurs solutions ont donc été introduites dans le cas des surfaces et des nuages de points pour définir des notions équivalentes.

2.3.1 Filtres surfaciques non locaux

Yoshizawa et coll. [YBS06] étendent les NL-Means aux filtrages surfaciques comme un moyen de supprimer les dégradations tout en préservant les structures principales de la surface maillée. Ils proposent une approche itérative basée sur [FDCO03] qui déplace les points du maillage $\mathbf{x}_i$ dans la direction de la normale $\mathbf{n}(\mathbf{x}_i)$. Ce déplacement est calculé comme une combinaison des valeurs $< \mathbf{x}_i - \mathbf{x}_j, \mathbf{n}(\mathbf{x}_i) >$ des points $\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i)$ compris dans un voisinage de $\mathbf{x}_i$ pondérées par rapport à la similarité des surfaces locales autour de $\mathbf{x}_i$ et de $\mathbf{x}_j$. Dans ce cas, l'utilisation de *Radial Basis Functions* (RBF) permet de définir une interpolation de la surface ainsi qu'un repère de comparaison normalisé.

Wang et coll. [WCZ+08] proposent une démarche similaire à [YBS06] en remplaçant la mesure de similarité sur les RBF par une mesure plus proche des NL-Means classiques qui utilisent des patchs. Ainsi, ils introduisent un repère normalisé local défini par le plan tangent à la surface au point du maillage $\mathbf{x}_i$ orienté par rapport à l'analyse en composantes principales des points compris dans le voisinage de $\mathbf{x}_i$. Dans ce repère, ils définissent des patchs de la variation géométrique locale de la surface. Ils utilisent une grille régulière centrée sur $\mathbf{x}_i$ dont les valeurs sont calculées en interpolant la hauteur des points du maillage du voisinage. Pour accélérer le temps de calcul et obtenir de meilleurs résultats, ils proposent d'effectuer un regroupement des points du maillage par *Meanshift* [CM02].

Adams et coll. [AGDL09] introduisent leur propre filtre surfacique non local comme une application de sa structure d'accélération de filtres à hautes dimensions. Ils proposent d'utiliser une *surface de référence*. Cette dernière, calculée par application itérative du filtre laplacien sur le maillage, permet d'associer à chaque point du maillage une valeur bruitée (correspondant à l'écart entre la surface bruitée et la surface de référence). Dans leur approche la similarité est calculée en utilisant des histogrammes 2D représentant la répartition locale des points bruités du maillage appartenant aux différentes zones d'un descripteur cylindrique.

Morigi et coll. [MRS12] proposent un filtre surfacique différent capable de préserver les structures principales du maillage par un débruitage non local de la courbure moyenne. Dans ce cas la mesure de similarité de la géométrie locale est définie par rapport à un descripteur basé sur la courbure moyenne locale.

2.3.2 Filtres non locaux de nuage de points

Malgré les bonnes performances des filtres surfaciques non locaux, il reste difficile de corriger les dégradations de l'acquisition après reconstruction de la surface maillée. En effet, une partie de l'information contenue dans le nuage de point 3D est perdue à ce moment de la chaîne d'acquisition. Par conséquent, il semble plus judicieux de filtrer le nuage de points 3D avant de reconstruire une surface. Contrairement aux surfaces maillées,

les nuages de points 3D ne possèdent aucune information de structure. Cette dernière est malheureusement indispensable pour définir un signal à débruiter et une mesure de similarité appropriée indispensables à toute restauration non locale.

Pour résoudre ce problème, Deschaud et coll. [DG10] suggèrent l'utilisation de méthodes de projection par moindres carrés glissants (Moving Least Square - MLS) [Lev98, ABCO+01] qui définissent localement une surface à partir d'un nuage de points. Leur approche peut être résumée par trois principales étapes. Tout d'abord, ils définissent pour chaque point du nuage un système de coordonnées local défini à partir des vecteurs associés aux deux plus grands vecteurs propres de la covariance des points. Ensuite, les variations de la surface autour du point d'intérêt sont approchées localement par un polynôme bivarié exprimé dans ce repère. Finalement, la position débruitée est estimée comme une combinaison des points proches pondérée par la similarité de leur polynôme bivarié. Cette démarche appliquée à chaque point du nuage permet d'obtenir le nuage débruité. Malheureusement, comme le démontre [Dig12], la paramétrisation du repère local n'est pas suffisamment stable.

Digne [Dig12] propose de décomposer le nuage de points en une surface grossière et un champ scalaire bruité en utilisant un filtre isotropique [DMSL11]. Elle propose ensuite de filtrer le champ scalaire par NL-Means. Pour ce faire, elle utilise un patch qui décrit les variations locales de la surface définie dans un repère local. La paramétrisation du repère local est calculée à partir d'une analyse en composantes principales des normales du nuage plus robuste aux variations de la surface [DM11]. Les valeurs du patch sont calculées par une interpolation sur une grille des hauteurs des points voisins par RBF.

Plus récemment, une méthode de débruitage de nuage de point par analyse spectrale collaborative de patchs a été introduite [RDK13]. Ces patchs utilisent un opérateur de Laplace-Beltrami afin débruiter le nuage d'une manière robuste.

2.3.3 Limitations

Les méthodes non locales ont été introduites pour améliorer la correction des dégradations introduites lors de l'étape d'acquisition. Ces approches ont été définies pour être utilisées sur deux types de données : la surface maillée, corrigeant ainsi les erreurs introduites par l'utilisation d'un nuage de points dégradé lors de l'étape de reconstruction ; le nuage de points, avant que la surface ne soit définie.

En utilisant la méthode de débruitage surfacique, il devient difficile de récupérer l'information dégradée, car une partie a été perdue lors de la phase de reconstruction. Par conséquent, il est préférable d'utiliser les méthodes de débruitage de nuages de points. Néanmoins, dans ce cas, la reconstruction ne permet pas d'exploiter correctement l'ensemble de l'information présente dans le nuage de points. En particulier, la propriété autosimilaire du nuage n'est pas exploitée par la reconstruction de la surface.

Nous introduisons au Chapitre 3 un nouvel opérateur capable de définir une surface implicite qui exploite correctement le caractère autosimilaire du nuage de points. Nous utilisons ce dernier pour définir des surfaces maillées, le débruitage de nuage de points et l'intensification des structures de l'objet.

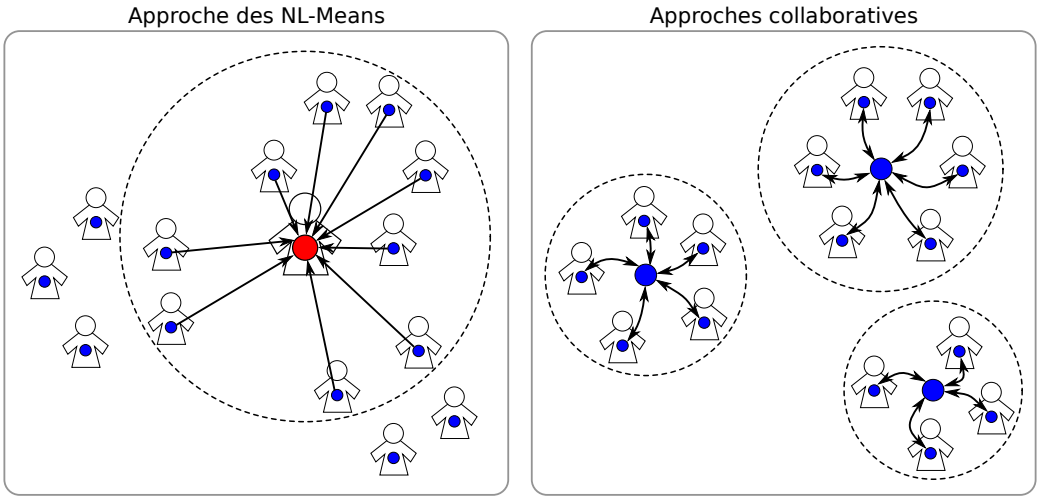


FIGURE 2.4: *Principe des filtres collaboratifs* : Nous considérons ici un ensemble d'individus possédant chacun une caractéristique bruitée (cercle interne à chaque individu). Pour les NL-Means, un individu (cercle rouge) utilise les caractéristiques que lui transmettent les individus voisins (cercle discontinu) pour restaurer la valeur de sa propre caractéristique. Une telle approche peut utiliser des individus qui ne font pas nécessairement partie de la même famille que l'individu considéré. Dans le cas des filtres collaboratifs, les individus sont regroupés en familles (cercles en pointillé). L'information sur les caractéristiques des individus d'une même famille est mise en commun (cercle bleu au centre de chaque famille), analysée puis retransmise à l'ensemble des individus de la famille. Dans ce cas, toutes les valeurs des individus d'une même famille sont restaurées en même temps en utilisant une information partagée par la famille uniquement.

2.4 Filtres collaboratifs

Les récents travaux sur les filtres non locaux ont permis une généralisation des NL-Means à des domaines d'applications différents de ceux de la restauration d'images et une évolution de ces principes originaux pour une meilleure exploitation de la redondance de l'information. De plus en plus, ces évolutions ont rapproché la restauration non locale d'un autre genre de filtrage : *le filtrage collaboratif*.

2.4.1 Principe

Le filtrage collaboratif est un procédé de filtrage de l'information utilisant des techniques qui impliquent une collaboration entre de multiples agents, points de vue, sources de données, etc. À l'origine, ces derniers ont été particulièrement utilisés dans le domaine de l'e-commerce comme un système de recommandation automatique à partir d'une connaissance de l'intérêt d'un grand nombre d'utilisateurs. De manière générale, ces méthodes consistent à prédire une information inconnue d'un individu (débruitage, complétion, etc.) par une analyse détaillée d'un groupe de personnes qui possède des caractéristiques communes à l'individu considéré. Ce genre d'idées diffèrent d'approches plus simples qui four-

nissent un score moyen non spécifique exprimant l'intérêt d'un ensemble d'utilisateurs (basé par exemple sur le nombre de votes).

Ce principe de filtrage a été généralisé au traitement des images par [DFKE07]. L'idée principale dans ce cas consiste à rassembler les patchs similaires par famille puis de les débruiter en utilisant l'information commune de la famille à laquelle ils appartiennent. Pour bien comprendre la différence entre NL-Means et les méthodes collaboratives, considérons un ensemble d'individus chacun associé à une caractéristique bruitée. Pour les deux types de méthodes, le but est de débruiter chaque caractéristique à l'aide d'autres individus (Figure 2.4). Dans le cas des NL-Means, elle est débruitée en combinant les caractéristiques des individus proches. L'individu en question ne retransmet aucune information qu'il a pu récupérer de ses voisins. Pour les filtres collaboratifs, les individus sont d'abord regroupés par familles d'individus proches. Ensuite, les individus d'une même famille se mettent d'accord mutuellement sur les valeurs débruitées de leurs caractéristiques respectives. En opposition aux filtres collaboratifs classiques, la collaboration en traitement d'image vient d'une analyse commune du groupe de patch qui est ensuite utilisée pour améliorer la qualité de la restauration. Ce type de filtre permet de révéler les détails partagés par l'ensemble du groupe tout en conservant les caractéristiques uniques de chaque patch. Le patch obtenu est donc plus proche de la position originale qu'en le débruitant par NL-Means.

2.4.2 Méthodes de restauration collaboratives

Dabov et coll. [DFKE07, DFKE08, DFKE09] sont les premiers à introduire le filtrage collaboratif pour la restauration des images. Leur méthode, appelée *BM3D* (Block Matching and 3D Filtering), se base sur trois principales étapes. Tout d'abord ils *empilent* les patchs en les regroupant en fonction de leurs similarités de sorte à former des piles de patchs. Ensuite, cette pile 3D est débruitée en gardant les fréquences principales d'une transformée (ondelettes, transformées en cosinus discret) appliquées sur la pile. Un filtrage appliqué sur cette troisième dimension permet de garder les caractéristiques communes de la famille tandis que le filtrage des deux autres dimensions permet de garder les caractéristiques spécifiques à chaque patch. Ainsi, la *collaboration* vient du fait que les patchs d'une même famille sont débruités conjointement en considérant l'analyse globale de la pile. Finalement, les patchs débruités sont *agrégés* de sorte à définir les valeurs débruitées de l'image. Ces trois étapes sont itérées deux fois afin d'améliorer la qualité finale du débruitage. BM3D constitue l'une des méthodes applicatives les plus performantes actuellement pour le débruitage d'images. Néanmoins, l'influence de ces nombreux paramètres reste difficile à analyser.

Yu et coll. [YSM12] proposent d'étendre le principe des filtrages collaboratifs à d'autres problèmes inverses tels que le débruitage, l'amélioration de la résolution ou d'*inpainting* en introduisant l'estimateur linéaire par morceaux (Piecewise Linear Estimator - PLE). Leur idée basée sur les travaux sur les structures parcimonieuses [MBP+09] approxime l'ensemble des patchs par un modèle de mélange de gaussiennes (Gaussian Mixture Model - GMM). Les gaussiennes sont ajustées à l'ensemble des patchs inconnus à restaurer via une Maximisation A Posteriori (MAP) qui utilise un algorithme d'espérance-maximisation (Expectation Maximisation - EM) itératif. Pour cette méthode il est difficile de trouver le bon nombre de classes des GMM ainsi qu'une bonne initialisation afin que la procédure

EM converge vers un bon minimum de la fonction objectif non convexe. En pratique, le nombre de gaussiennes est fixé à l'avance à environ une dizaine afin de garder la complexité du calcul sous contrôle.

Quand l'ensemble des patchs est corrompu par un bruit gaussien de variance σ^2, l'Équation 2.6 des NL-Means peut être reformulée comme une Espérance A Posteriori (EAP) bayésienne. Lebrun et coll. [LBM13] reformule les NL-Means pour définir les *Non Local Bayes* (NLB) en remplaçant la formulation EAP des NL-Means par une Maximisation A Posteriori (MAP). Désormais en plus de la moyenne non locale, chaque patch est associé à une matrice de covariance $\mathbf{\Sigma}$ décrivant la variabilité au sein du groupe de patchs. La covariance étant bruitée, le côté collaboratif de ce filtre vient du fait que tous les patchs sont débruités conjointement par l'opération de filtrage de la covariance $\hat{\mathbf{\Sigma}} = \mathbf{\Sigma} - \sigma^2 Id$. En utilisant une approche similaire aux BM3D, le patch est débruité en deux étapes : la première étape débruite à partir de variations estimées sur les patchs bruités et la deuxième à partir des patchs débruités à la première étape. Tout comme BM3D, une phase d'agrégation permet de définir un meilleur débruitage des pixels de l'image. Nous proposons en Section 5.1.1 plus de détails concernant les NLB. Les NLB permettent actuellement d'obtenir les meilleures performances de débruitage. Néanmoins, leur approche nécessite une grande quantité de calcul dans la mesure où un modèle gaussien local a besoin d'être appris deux fois autour de chaque patch. Pour être calculable, l'estimation du voisinage est approchée par une recherche des k plus proches voisins dans l'espace des patchs.

2.4.3 Généralisation des filtres collaboratifs

Durant ces dernières années, la formulation collaborative des méthodes de débruitage a permis d'améliorer significativement la qualité de la restauration d'images bruitées. Le principe introduit par BM3D [DFKE07, DFKE08, DFKE09] a ensuite été amélioré par NLB [LBM13] en remplaçant la formulation EAP des NL-Means par une formulation MAP qui permet de mieux récupérer les structures communes d'un groupe de patchs similaires tout en préservant ses caractéristiques spécifiques. Néanmoins, de telles approches nécessitent une grande quantité de calculs et sont généralement simplifiées pour pouvoir être calculables.

De plus, PLE [YSM12] a proposé une généralisation des filtres collaboratifs à d'autres applications du traitement des images. Cette solution offre d'excellentes performances, mais utilise une GMM simplifiée pour pouvoir obtenir de bons résultats dans un temps raisonnable. Par conséquent, une telle approche semble difficilement applicable sur de grands ensembles de données.

Nous introduisons au Chapitre 4 une nouvelle structure de données capable d'apprendre les variations d'un grand ensemble de points avec une quantité de mémoire limitée. Elle représente les variations des données apprises en n'importe quelles positions et échelles de l'espace des patchs par une gaussienne anisotropique. Une telle structure est définie comme une approche hybride entre PLE et NLB.

Nous utilisesons cette structure au Chapitre 5 pour reformuler des problèmes de restauration 2D et 3D d'un point de vue d'une MAP collaborative. En outre, nous apprendrons

les variations de grandes bases de données avec une quantité de mémoire limitée afin de mieux restaurer des données dégradées.

CHAPITRE 3

Surface de points non locale

Dans ce chapitre, nous proposons de définir la première surface qui exploite l'autosimilarité présente dans un nuage de points (Non Local Point Set Surfaces — NLPSS). Nous commencerons par décrire dans la Section 3.1 une généralisation des méthodes de surface de points. En Section 3.2, nous introduisons notre opérateur de projection non local ainsi que l'ensemble des outils nécessaires à sa définition. Nous montrerons en Section 3.3 que notre approche permet de retrouver les caractéristiques importantes de l'objet tout en comblant l'information manquante par une augmentation locale du ratio signal sur bruit. Finalement en Section 3.4, nous proposons différentes améliorations possibles pour notre opérateur non local.

3.1 Contexte

3.1.1 Acquisition 3D

Avec la démocratisation des appareils d'acquisition 3D, il est devenu aujourd'hui facile de numériser des objets réels avec une haute précision. Généralement sous la forme de nuages de points bruts corrompus par du bruit, des points aberrants, des trous et/ou des taux d'échantillonnages variables, ces acquisitions doivent être traitées pour définir des surfaces maillées exploitables.

Les méthodes basées points ont donc été introduites comme un moyen d'améliorer la qualité des données acquises sans pour autant faire de choix concernant la nature de la surface sous-jacente. Les surfaces de nuages de points (Point Set Surfaces — PSS) ont été définies en ce sens, et permettent de faire le lien entre les nuages de points et les surfaces maillées. Elles introduisent une représentation surfacique lisse non maillée essentiellement définie par l'ensemble des points fixes d'un opérateur de projection.

3.1.2 Travaux existants

Alexa [ABCO+01] introduit initialement les PSS en se basant sur les travaux de Levin [Lev98,Lev03] qui étend une classe de méthodes d'approximation fonctionnelles de don-

nées parcimonieuses — les moindres carrés glissants (Moving Least Squares - MLS) [She68] — au cas d'approximation de formes. Cette première version des PSS est définie par l'intermédiaire d'un opérateur de projection itératif. À chaque étape, un point d'évaluation est projeté sur une surface localement définie par un polynôme bivarié paramétré sur un plan qui approxime les données. Ces deux derniers sont estimés localement par une minimisation non linéaire du carré de la distance des données aux points d'évaluation.

Amanta et coll. [AK04] démontre qu'il est possible de définir une surface d'aussi bonne qualité en effectuant une moyenne pondérée des positions, sans utiliser de polynôme bivarié ou de minimisation non linéaire. Ils introduisent une fonction de distance signée permettant une représentation implicite de la surface. Cette représentation peut être utilisée par différents algorithmes de remaillage comme les *Marching Cubes* [LC87] ou une triangulation de Delaunay restreinte [BO05] pour définir une surface polygonale.

Adamson et coll. [AA04a] exploitent cette idée pour introduire un modèle simple de surface de nuages de points (Simple Point Set Surfaces — SPSS). Dans le cas de nuages de points munis de normales, le plan local de projection peut être estimé par une combinaison linéaire pondérée des positions et des normales. Si le nuage de point ne comporte pas de normales, une analyse en composantes principales [HDD+94] permet d'équiper chaque point du nuage avec une normale estimée. Avec les SPSS, Adamson [AA04b] introduit trois procédures différentes pour estimer itérativement la projection d'un point sur la surface : l'orthogonale qui projette le point d'évaluation précisément au terme de calculs sophistiqués, la basique d'une faible complexité, mais peu précise, et l'orthogonale approximative qui constitue un bon compromis entre simplicité et précision.

Fleishman et coll. [FCOS05] améliorent cette définition en introduisant une formule implicite des moindres carrés glissants (Implicit Moving Least Squares - IMLS) qui préserve les bords anguleux de la surface. Alexa et Adamson [AA09] proposent une surface de point hermitienne (Hermite Point Set Surfaces — HPSS) qui limite les effets de réduction des méthodes précédentes. Au lieu de projeter le point sur un seul plan, il est projeté sur l'ensemble des plans définis par chaque échantillon voisin. Ensuite la position est calculée par une combinaison pondérée de ces différentes projections permettant de définir une combinaison hermitienne.

La procédure d'estimation du plan peut devenir rapidement instable quand le nuage de points n'est pas assez dense par rapport à la taille du support de l'opérateur ou quand les taux d'échantillonnage du nuage de points sont trop variables. Guennebaud et coll. [GG07] proposent de remplacer le plan par une sphère algébrique définissant une nouvelle surface de points algébrique (Algébraic Point Set Surfaces — APSS). Ce modèle est ensuite amélioré en contraignant les gradients de la sphère algébrique à correspondre aux normales des échantillons [GGG08].

Les méthodes précédentes ne sont pas adaptées pour définir une surface à partir d'un nuage de points corrompu par des points aberrants. En utilisant une méthode de régression par noyau qui ajuste itérativement les poids de chaque échantillon, Öztireli et coll. [ÖGG09] introduisent une nouvelle formule implicite et robuste des moindres carrés glissants (Robust Implicit Moving Least Square - RIMLS). En plus d'une résistance aux points aberrants, cette définition permet une meilleure préservation des caractéristiques de la surface. Nous vous renvoyons vers [CWL+08] pour un état de l'art plus complet sur les surfaces MLS.

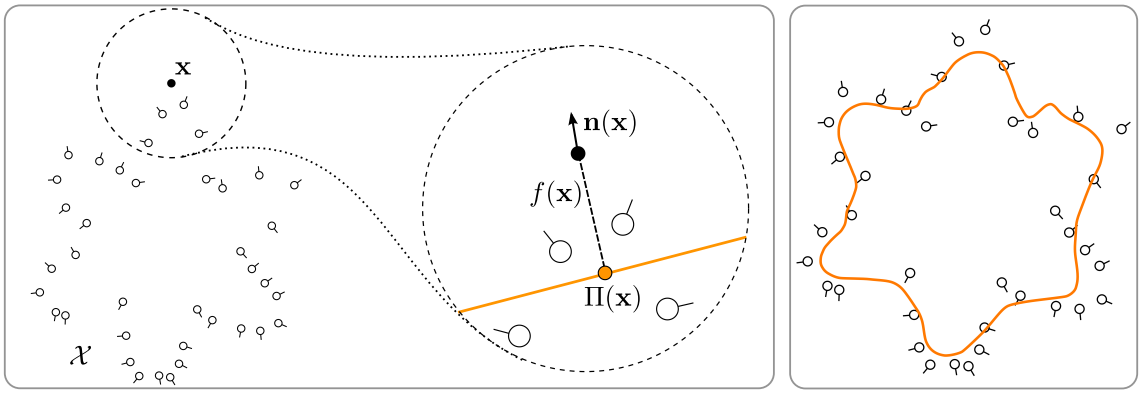


FIGURE 3.1: *Gauche* : Le SPSS [AA04a] définit une surface à partir d'un nuage de points $\mathcal{X}$ muni de normales par l'intermédiaire d'un opérateur Π qui projette pour tout $\mathbf{x} \in \mathbb{R}^3$ sur un plan estimé localement de normale $\mathbf{n}(\mathbf{x})$. *Droite* : L'opérateur de projection Π introduit une fonction implicite $f(\mathbf{x})$ dont l'ensemble des valeurs nulles définissent la surface de l'opérateur (en orange).

3.1.3 Généralisation des PSS

Jusqu'à la fin de ce chapitre, nous considérerons comme entrée des PSS un nuage de points $\mathcal{X} = \{\mathbf{x}_i, \mathbf{n}_i\}$ où $\mathbf{x}_i \in \mathbb{R}^3$ correspond à la position spatiale d'un point et $\mathbf{n}_i \in \mathbb{R}^3$ est la normale qui lui est associée. D'après la procédure de projection orthogonale approximative introduite par Adamson et Alexa [AA04b], l'ensemble des définitions PSS précédentes peuvent s'exprimer à l'aide d'un opérateur de projection Π :

$$MLS_{\mathcal{X}} : \mathbb{R}^3 \to \mathbb{R}^3, \mathbf{x} \to \Pi(\mathbf{x}) \tag{3.1}$$

Chaque point d'évaluation $\mathbf{x}$ est projeté sur une primitive Q localement estimée par moindres carrés (e. g. plan pour le SPSS, sphère algébrique pour l'APSS) :

$$\Pi(\mathbf{x}) \quad = \quad \mathbf{x} - f(\mathbf{x})\mathbf{n}(\mathbf{x}) \tag{3.2}$$

Cette formulation permet de mettre en évidence la distance implicite $f(\mathbf{x})$ entre $\mathbf{x}$ et sa projection sur Q et $\mathbf{n}(\mathbf{x})$ la normale PSS définie en cet emplacement. Q est paramétrée sur $\mathcal{X}$ par rapport à $\mathbf{x}$ à une échelle t liée à la taille du support choisie pour le noyau de pondération spatial sous-jacent.

Par exemple dans le cas des SPSS (Figure 3.1), cette définition générale peut être exprimée par :

$$\mathbf{n}(\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in \mathcal{X}} w_t(\mathbf{x}, \mathbf{x}_i)\mathbf{n}_i}{\| \sum_{\mathbf{x}_i \in \mathcal{X}} w_t(\mathbf{x}, \mathbf{x}_i)\mathbf{n}_i \|}$$

$$\mathbf{c}(\mathbf{x}) = \frac{\sum_{\mathbf{x}_i \in \mathcal{X}} w_t(\mathbf{x}, \mathbf{x}_i)\mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathcal{X}} w_t(\mathbf{x}, \mathbf{x}_i)}$$

$$f(\mathbf{x}) = < \mathbf{x} - \mathbf{c}(\mathbf{x}), \mathbf{n}(\mathbf{x}) >$$

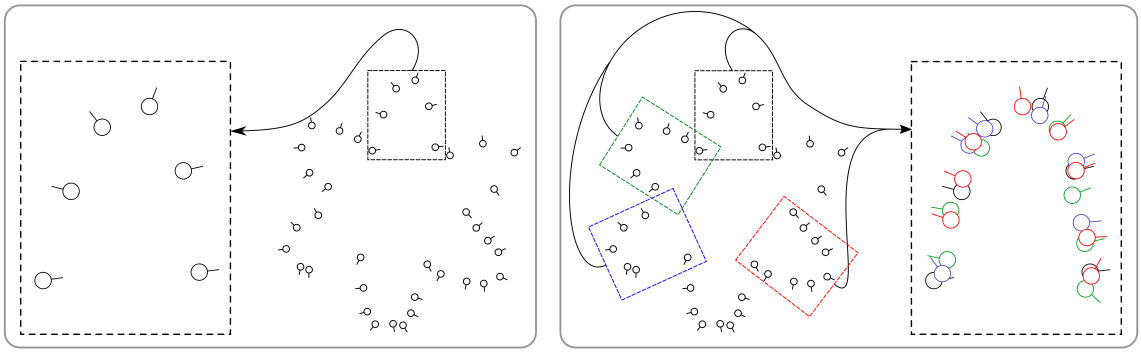


FIGURE 3.2: *Gauche* : les opérateurs PSS conventionnels définissent la surface en considérant un voisinage local. Face aux conditions réelles de capture, ce point de vue purement local devient insuffisant pour définir une surface correcte. *Droite* : le principe de notre algorithme est d'exploiter le caractère autosimilaire des nuages de points pour augmenter localement le ratio signal sur bruit et ainsi proposer une définition de surface plus adaptée aux conditions de capture.

Le noyau de pondération spatial $w_t(\mathbf{x}, \mathbf{x}_i)$ est généralement défini à partir d'une fonction à support compact polynomiale par morceaux g :

$$w_t(\mathbf{x}, \mathbf{x}_i) = \frac{1}{Z_t(\mathbf{x})} g(\frac{1}{t}\|\mathbf{x} - \mathbf{x}_i\|)$$

$Z_t(\mathbf{x})$ est choisie de manière à assurer la somme unitaire des poids $\sum_{\mathbf{x}_i \in \mathcal{X}} w_t(\mathbf{x}, \mathbf{x}_i) = 1$. Le paramètre t associé à w_t correspond au paramètre de filtrage de la surface : plus t est grand, plus la surface résultante est lisse. Faire varier t permet de décomposer le PSS à différentes échelles. Au cours de ce chapitre, nous appellerons respectivement $\Pi^t(\mathbf{x})$, $f^t(\mathbf{x})$ et $\mathbf{n}^t(\mathbf{x})$ l'opérateur de projection, la distance implicite et la normale définie par l'opérateur PSS à cette échelle t.

3.1.4 Problématique

Dans tous les modèles PSS, le choix de la taille du support du noyau spatial induit la qualité de la surface résultante : une taille élevée lissera la surface en éliminant les défauts introduits lors de l'acquisition, et à l'opposé une taille fine fera ressortir les détails importants de l'objet sans débruiter la surface. Malheureusement, il est difficile de trouver une taille de support idéale pour satisfaire le compromis lissage/préservation des caractéristiques de l'objet car le problème est restreint à un point de vue strictement local.

Pour résoudre ce problème de manière adéquate, nous proposons de ne plus nous limiter à un point de vue strictement local, mais d'utiliser l'information apportée par l'ensemble du nuage lors de la projection d'un point d'évaluation. Comme le montre la Figure 3.2, une telle approche permet d'augmenter le ratio information/bruit et ainsi de résoudre les ambiguïtés dues aux défauts du nuage de points.

Ainsi, nous proposons de définir une surface de points non locale (Non Local Points Set Surfaces — NLPSS) qui possède les avantages suivants :

1. **Définition d'une surface non locale** : Notre définition n'est pas restreinte au débruitage de nuages de points dans le sens où nous proposons un nouvel opérateur PSS exploitant l'autosimilarité du nuage et qui peut être utilisé pour de la reconstruction, du débruitage et d'autres traitements plus génériques sur un ensemble non organisé du nuage de points.
2. **Généralisation** : Nous proposons une extension de toutes les méthodes PSS précédentes en leur incorporant une propriété non locale.
3. **Dégénérescence** : Notre définition dégénère en la définition traditionnelle du PSS sur lequel il se base dans le cas de faible autosimilarité.

3.2 Surfaces de points non locales

Dans cette partie nous allons définir notre opérateur non local de surface de points. Pour pouvoir introduire cette définition, nous introduirons trois notions :

1. **Signal** : De par la nature non structurée des nuages de points, il n'est pas possible de définir un signal en considérant directement leur positions spatiales des points du nuage. Nous introduirons un signal adapté qui contient les défauts du nuage sur lequel nous appliquerons les méthodes non locales.
2. **Descripteur local** : La comparaison de deux sous-ensembles d'un nuage de points doit être peu sensible à la nature des nuages de points et aux défauts introduits lors de l'acquisition. Nous définirons un descripteur local adapté au nuage de points défini dans un repère local normalisé.
3. **Fonction de pondération** : Nous proposerons une fonction de pondération tenant compte de la quantité d'information non locale qui est présente.

3.2.1 Principe général

En considérant une échelle t_0 suffisamment large, le PSS peut être décomposé en une surface grossière S_{t_0} et un champ scalaire de déplacement résiduel $m(\mathbf{x}_i)\mathbf{n}^{t_0}(\mathbf{x}_i)$ qui contient les caractéristiques du nuage de points contaminées par le bruit. L'idée principale de notre approche est d'utiliser les méthodes non locales pour supprimer le bruit du champ scalaire résiduel, et, simultanément, calculer la projection (interpolation) à une échelle fine pour tous les points d'évaluation $\mathbf{x} \in \mathbb{R}^3$.

Nous appliquons tout d'abord un opérateur de projection PSS local Π^{t_0} à une échelle large t_0 pour définir une surface grossière S_{t_0}. Nous ajoutons ensuite à S_{t_0} un champ scalaire de déplacement fin $m_{NL}(\mathbf{x})\mathbf{n}^{t_0}(\mathbf{x})$ défini par une approximation non locale de la distance résiduelle $m(\mathbf{x})$ entre la surface grossière et la reconstruction de $\mathcal{X}$. Les différentes étapes de notre approche sont présentées figure 3.3.

Nous introduisons donc la première définition de surface de points non locale (Non Local Point Set Surfaces — NLPSS) par l'opérateur de projection :

$$\Pi_{NL}(\mathbf{x}) = \Pi^{t_0}(\mathbf{x}) - m_{NL}(\mathbf{x})\mathbf{n}^{t_0}(\mathbf{x}) \tag{3.3}$$

Nous présentons Figure 3.4 notre schéma de projection non local.

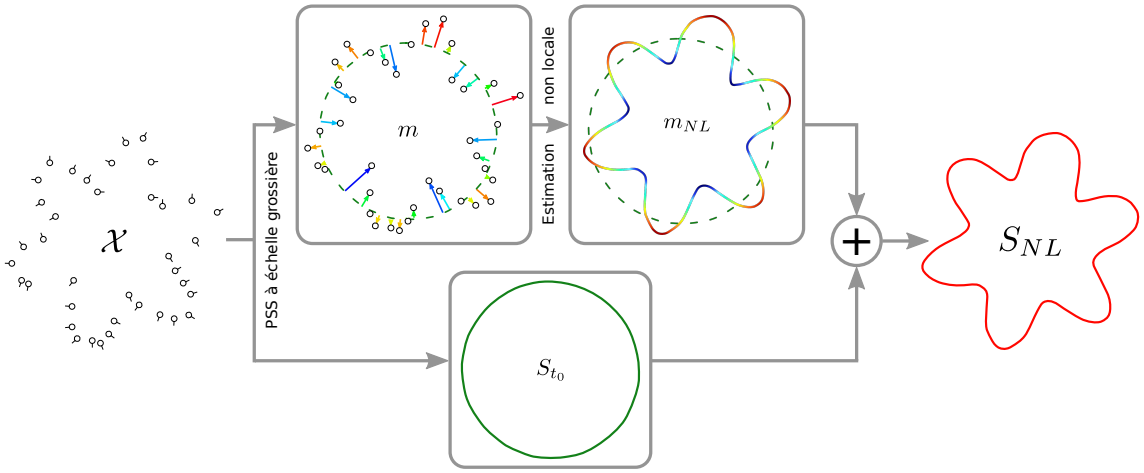


FIGURE 3.3: Nous utilisons un opérateur PSS grossier sur notre nuage de points d'entrée $\mathcal{X}$ pour décomposer le signal en une surface grossière S_{t_0} (en vert) et un champ scalaire de déplacement résiduel m. Ce dernier étant éventuellement bruité et défini seulement aux points du nuage de points original, nous approximons ces valeurs en tout point de la surface grossière m_{NL} par une méthode non locale qui améliore la qualité de l'approximation en exploitant le caractère autosimilaire du nuage. Notre surface non locale S_{NL} est par conséquent définie en ajoutant à la surface grossière le champ scalaire de déplacement approximé.

3.2.2 Carte de déplacement

Le champ scalaire de déplacement $m(\mathbf{x})$ est défini par la distance résiduelle entre la surface grossière S_{t_0} et $\mathcal{X}$. Pour tout $\mathbf{x} = \mathbf{x}_i \in \mathcal{X}$:

$$m(\mathbf{x}_i) = \left\langle \mathbf{x}_i - \Pi^{t_0}(\mathbf{x}_i),\, \mathbf{n}^{t_0}(\mathbf{x}_i) \right\rangle = f^{t_0}(\mathbf{x}_i) \tag{3.4}$$

Malheureusement, la nature parcimonieuse du nuage de points et les défauts introduits par l'acquisition 3D font que le champ de déplacement est défini seulement pour les points du nuage $\mathbf{x}_i \in \mathcal{X}$ et peut être bruité.

Dans le but de débruiter ces valeurs et d'étendre sa définition pour tous les points $\mathbf{x} \in \mathbb{R}^3$, nous considérons une moyenne pondérée non locale des $m(\mathbf{x}_i)$:

$$m_{NL}(\mathbf{x}) \quad = \quad \sum_{\mathbf{x}_i \in \mathcal{X}} w_{NL}(\mathbf{x}, \mathbf{x}_i) m(\mathbf{x}_i) \tag{3.5}$$

Ainsi nous allons devoir introduire une mesure de similarité w_{NL} entre les voisinages locaux des deux points $\mathbf{x}$ et $\mathbf{x}_i$. Pour cela, nous introduirons un descripteur 3D défini par l'intermédiaire de repères locaux pour faciliter leur comparaison.

3.2.3 Fonction de pondération

Dans cette partie nous allons introduire une mesure de similarité entre deux sous-ensembles du nuage de points. Par conséquent nous allons introduire un patch qui décrit localement

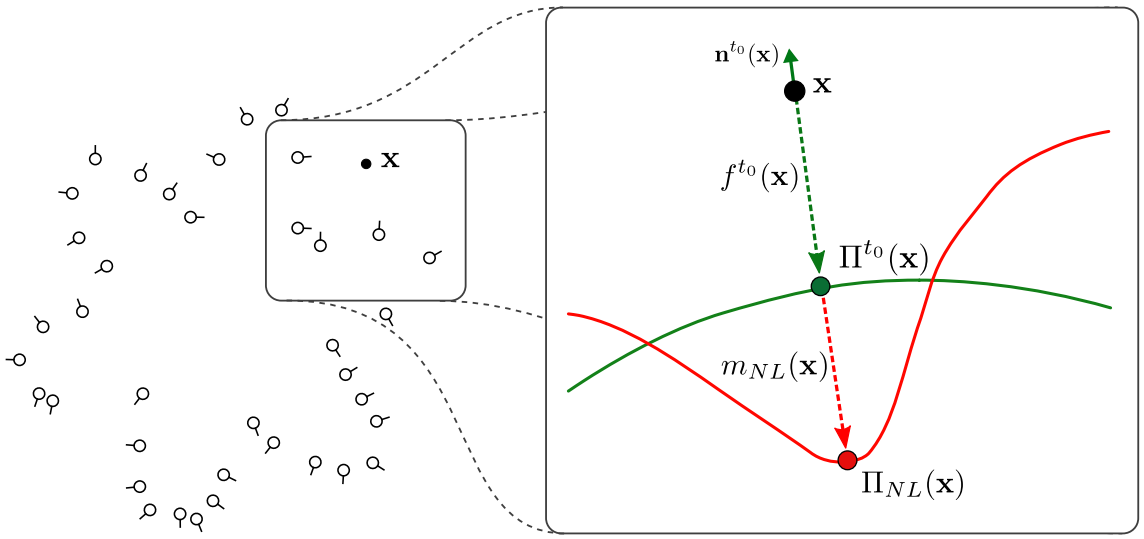


FIGURE 3.4: Notre définition permet de projeter tout point $\mathbf{x} \in \mathbb{R}^3$ par l'intermédiaire de l'opérateur Π_{NL}. $\mathbf{x}$ est projeté dans un premier temps sur la surface grossière (représentée en vert) en $\Pi^{t_0}(\mathbf{x})$. La projection $\Pi_{NL}(\mathbf{x})$ sur la surface NLPSS (représentée en rouge) est définie en ajoutant la valeur du champ scalaire de déplacement débruité m_{NL} dans la direction de la normale grossière $\mathbf{n}^{t_0}(\mathbf{x})$.

le champ scalaire de déplacement dans un repère normalisé tangent à la surface grossière S_{t_0}.

Descripteur 3D

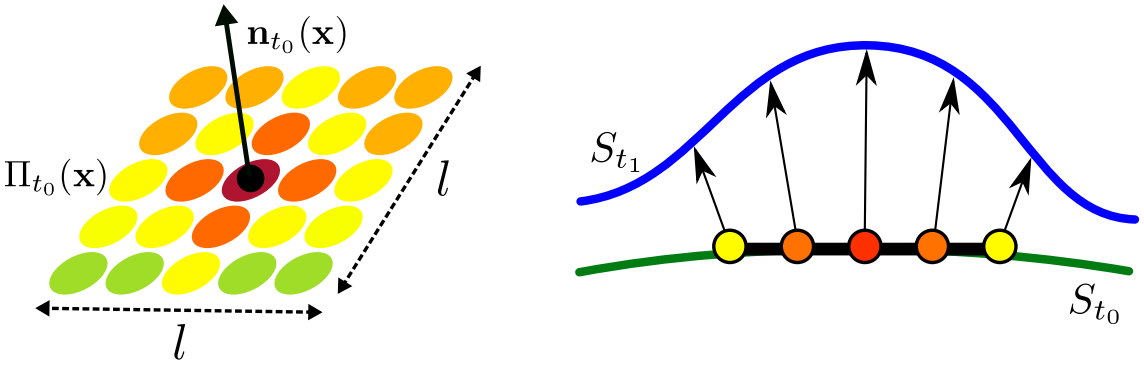


FIGURE 3.5: *Gauche* : Nous définissons la mesure de similarité entre deux sous-ensembles du nuage de points par l'intermédiaire d'un patch de taille $l = 5\delta$ (où δ correspond à l'écart moyen entre les points) avec $n = 2$ centré en $\Pi_{t_0}(\mathbf{x})$. L'ensemble des points qui composent le patch sont disposés sur le plan tangent à la surface S_{t_0} définie par la normale $\mathbf{n}_{t_0}(\mathbf{x})$. *Droite* : Les valeurs du patch sont calculées en utilisant la surface S_{t_1} (en bleu). Les valeurs colorées du patch représentent les différences locales entre S_{t_0} (en vert) et S_{t_1}.

La définition de notre patch est présentée figure 3.5. Pout tout point $\mathbf{x} \in \mathbb{R}^3$, notre patch local de déplacement est défini par un ensemble de dimension $(2n+1) \times (2n+1)$ points $D = [\![-n, n]\!]^2 \subset \mathbb{Z}^2$. Ces points sont localisés dans un carré de taille $l \times l$ défini par l'intermédiaire d'un systeme de coordonnées local centré en $\mathbf{x}_{t_0} = \Pi^{t_0}(\mathbf{x})$ dont deux axes définissent le plan tangent à la surface grossière en $\mathbf{x}_{t_0}$. Pour $(i,j) \in D$, chaque point $\mathbf{x}_{i,j}$ du patch a une valeur $\mathbf{P}_{\mathbf{x}_{t_0}}(i,j)$ correspondant à la valeur de déplacement $m(\mathbf{x}_{i,j})$.

Comme nous l'avons expliqué section 3.2.2, les valeurs de la carte de déplacement ne sont connues que pour les points du nuage $\mathbf{x} = \mathbf{x}_i \in \mathcal{X}$. Par conséquent, nous approximons les valeurs de $m(\mathbf{x}_{i,j})$ en utilisant un PSS local pour définir une surface S_{t_1} à une échelle fine $t_1 \ll t_0$. Ici, S_{t_1} représente une surface peu débruitée et interpolée de $\mathcal{X}$.

Les valeurs $\mathbf{P}_{\mathbf{x}_{t_0}}$ correspondent donc à la projection des points du patch sur la surface fine S_{t_1} :

$$\mathbf{P}_{\mathbf{x}_{t_0}}(i,j) = f^{t_1}(\mathbf{x}_{i,j}) \quad \forall (i,j) \in D. \tag{3.6}$$

Pour que la distance entre les points $\mathbf{x}_{i,j}$ et S_{t_0} soit négligeable par rapport à $\mathbf{P}_{\mathbf{x}_{t_0}}$, nous choisissons la taille du patch $l \approx \frac{t_0}{3}$.

Orientation du repère

Pour comparer les descripteurs locaux d'une manière similaire, nous introduisons un repère orthonormé local $(\mathbf{u}, \mathbf{v}, \mathbf{n}^{t_0})$. Pour bien définir le patch de distance, les deux axes $\mathbf{u}, \mathbf{v}$ appartiennent au plan tangent à la surface grossière S_{t_0} au point $\mathbf{x}_{t_0}$. Par conséquent notre patch est défini à une orientation près. Pour que cette dernière reste robuste au bruit et au taux d'échantillonnage, nous la choisissons en fonction des directions principales de courbures de la surface. Digne [Dig12, DM11] a montré qu'il était possible de les calculer par une simple analyse en composantes principales des normales. Le vecteur propre associé à la plus grande valeur propre est tangent à la direction principale.

Nous estimons la moyenne $\mathbf{n}_m$ et la matrice de covariance $\mathbf{\Sigma_n}$ des normales en considérant un sous-ensemble de points $\tilde{\mathcal{X}}$ autour de $\mathbf{x}_{t_0}$:

$$\mathbf{n}_m = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\mathbf{n}_i \in \tilde{\mathcal{X}}} \mathbf{n}_i \tag{3.7}$$

$$\mathbf{\Sigma_n} = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\mathbf{n}_i \in \tilde{\mathcal{X}}} (\mathbf{n}_i - \mathbf{n}_m).\ {}^t(\mathbf{n}_i - \mathbf{n}_m) \tag{3.8}$$

Le vecteur propre $\tilde{\mathbf{u}}$ associé à la plus grande valeur propre de la matrice de covariance $\mathbf{\Sigma_n}$ n'appartient pas nécessairement au plan tangent à la surface S_{t_0}. Par conséquent, nous redéfinissons les directions $\mathbf{u}$ et $\mathbf{v}$ par $\mathbf{v} = \frac{\mathbf{n}^{t_0} \wedge \tilde{\mathbf{u}}}{\|\mathbf{n}^{t_0} \wedge \tilde{\mathbf{u}}\|}$ et $\mathbf{u} = \frac{\mathbf{v} \wedge \mathbf{n}^{t_0}}{\|\mathbf{v} \wedge \mathbf{n}^{t_0}\|}$. Désormais, le repère $(\mathbf{u}, \mathbf{v}, \mathbf{n}^{t_0})$ est défini à une direction près.

Si nous définissons, les moments m_k d'ordre k du patch par :

$$\begin{aligned} m_k &= \sum_{i,j} < (\mathbf{x}_{i,j} - \mathbf{x}_{t_0}), \mathbf{u} >^k . < (\mathbf{x}_{i,j} - \mathbf{x}_{t_0}), \mathbf{n}_{t_0} > \\ m_k &= \sum_{i,j} < (\mathbf{x}_{i,j} - \mathbf{x}_{t_0}), \mathbf{u} >^k . \mathbf{P}_{\mathbf{x}_{t_0}}(i,j) \end{aligned} \tag{3.9}$$

Nous utilisons le moment m_1 d'ordre 1 pour lever l'ambiguïté sur la direction de $\mathbf{u}$ lors du calcul du patch. La direction pour le vecteur $\mathbf{u}$ est obtenue de telle sorte que $m_1 > 0$.

Mesure de similarité

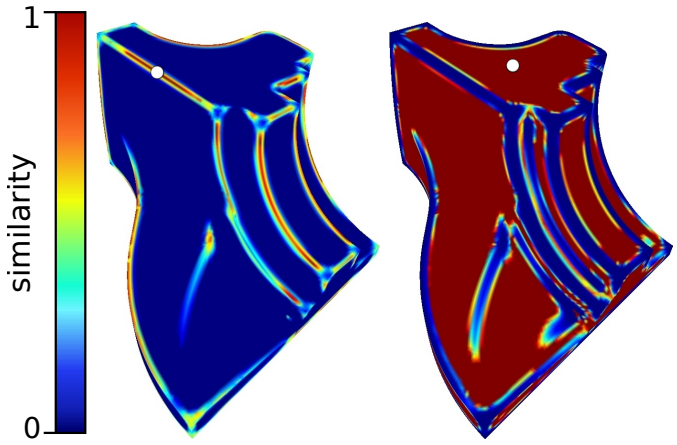


FIGURE 3.6: Résultats de notre mesure de similarité par rapport à un point appartenant à un bord de l'objet (*gauche*) et à un point appartenant à un plan (*droite*) représenté dans chaque cas en blanc. Les couleurs représentent la similarité entre le point de référence et les autres points de l'objet, allant du rouge (très similaire) au bleu (peu similaire).

Pour deux points $\mathbf{y}, \mathbf{z} \in \mathbb{R}^3$, nous définissons la distance entre leurs patchs associés par :

$$d(\mathbf{y}, \mathbf{z}) = \frac{1}{2n+1} \|\mathbf{P}_{\mathbf{y}_{t_0}} - \mathbf{P}_{\mathbf{z}_{t_0}}\|_2 \tag{3.10}$$

La mesure de similarité correspondante est définie par l'intermédiaire d'un paramètre h :

$$w_{NL}(\mathbf{y}, \mathbf{z}) = \frac{1}{Z_{NL}(\mathbf{y})} \exp\left(-\frac{d(\mathbf{y}, \mathbf{z})^2}{h^2}\right) \tag{3.11}$$

La constante de normalisation $Z_{NL}(\mathbf{y})$ assure que $\sum_{\mathbf{x}_i \in \mathcal{X}} w_{NL}(\mathbf{y}, \mathbf{x}_i) = 1$. Le paramètre h permet d'ajuster la mesure en fonction de l'autosimilarité du nuage de points.

Nous présentons figure 3.6 les résultats de notre mesure de similarité entre un point de référence et l'ensemble des points de l'objet.

3.2.4 Opérateur de projection non local

Notre opérateur de projection $\Pi_{NL}(\mathbf{x})$ est défini en utilisant la distance implicite $f_{NL}(\mathbf{x})$ d'un point $\mathbf{x} \in \mathbb{R}^3$ à la surface de points non locale :

$$\Pi_{NL}(\mathbf{x}) = \mathbf{x} - f_{NL}(\mathbf{x})\mathbf{n}^{t_0}(\mathbf{x}) \tag{3.12}$$

La distance implicite non locale est définie par :

$$f_{NL}(\mathbf{x}) = f^{t_0}(\mathbf{x}) - \sum_{\mathbf{x}_i \in \mathcal{X}} w_{NL}(\mathbf{x}, \mathbf{x}_i) m(\mathbf{x}_i) \tag{3.13}$$

Dégénérescence du noyau de pondération non local

Pour des points $\mathbf{x}$ dont le facteur d'autosimilarité est faible, notre définition aura tendance à ne pas débruiter suffisamment le point résultant introduisant ainsi des artefacts dans la définition de notre surface non locale. Dans ces cas-là, l'utilisation de PSS locaux semble plus adaptée. Notre définition précédente peut être facilement modifiée pour tenir compte de cette particularité. Pour cela, nous ajoutons au noyau non local w_{NL} un noyau spatial w_t (correspondant à la définition locale des PSS). Notre nouveau noyau de pondération $\tilde{w}_{NL}$ permet de combiner l'information locale et non locale par l'intermédiaire d'un facteur de contrôle α :

$$\tilde{w}_{NL}(\mathbf{x}, \mathbf{x}_i) = \frac{1}{Z(\mathbf{x})}(\alpha Z_t(\mathbf{x}) w_t(\mathbf{x}, \mathbf{x}_i) + Z_{NL}(\mathbf{x}) w_{NL}(\mathbf{x}, \mathbf{x}_i)) \tag{3.14}$$

La constante de normalisation $Z(\mathbf{x})$ est choisie pour assurer une somme unitaire. Dans cette formule, quand $\mathbf{x}$ fait partie d'un ensemble très représenté, le terme non local domine, permettant d'augmenter la qualité du débruitage et d'améliorer la résolution. Au contraire, si $\mathbf{x}$ appartient à une zone peu représentée, le terme spatial domine, assurant ainsi une meilleure completion des trous du nuage. Le paramètre α permet de choisir l'impact du noyau non local par rapport au noyau local dans la définition de la surface finale.

Noyau à support compact

Lors du calcul de la projection non locale, dû à l'utilisation d'un noyau gaussien, chaque point $\mathbf{x}_i \in \mathcal{X}$ possède un poids w_{NL} non nul. Par conséquent, la somme des points de poids faibles ne peut plus être considérée comme négligeable. Ce phénomène est d'autant plus important que le paramètre h est élevé (ce qui aura pour conséquence de rendre similaires l'ensemble des patchs de l'image).

En traitement des images, cette somme est généralement réduite à un sous ensemble de pixels locaux autour du point d'évaluation. Dans ce cas, l'impact des points similaires dans la somme finale est plus significatif. Malheureusement, une telle démarche peut difficilement être appliquée au cas des nuages de points. En effet, il est plus intéressant de considérer le nuage de points dans sa totalité permettant, par conséquent, de débruiter les caractéristiques du nuage les plus rares.

Ainsi, pour limiter l'impact des points parasites, nous remplaçons le voisinage local par l'ensemble des k patchs qui sont les plus similaires. Le poids des autres points du nuage est imposé à zéro. Cette démarche permet à la fois d'obtenir des résultats d'une meilleure qualité et une accélération du temps de calcul.

3.3 Résultats

3.3.1 Détails d'implémentation et performance

Nous avons implémenté notre algorithme en C++. La recherche des k plus proches patchs telle que nous l'avons décrite section 3.2.4 est accélérée par l'utilisation d'un algorithme

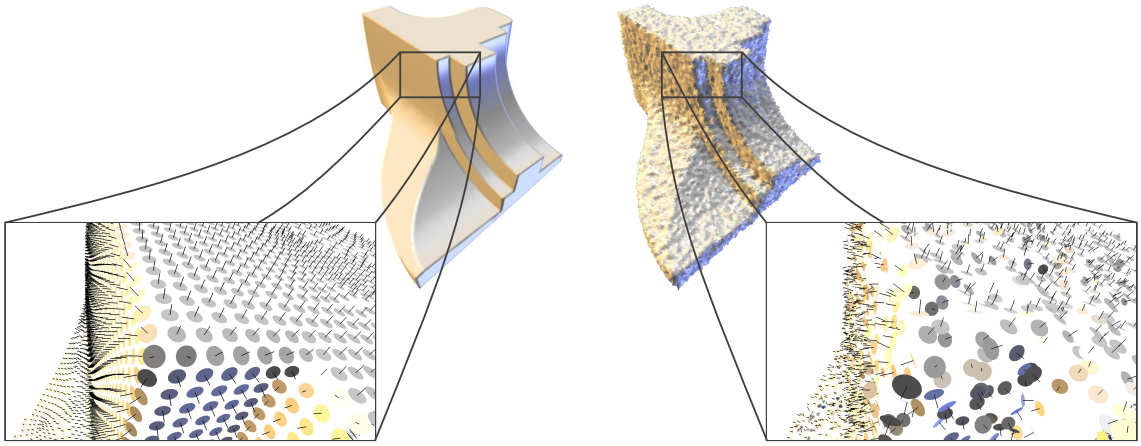


FIGURE 3.7: Nuage de points original *Fandisk* (*gauche*) qui a été bruité par un bruit uniforme (*droite*) permettant de simuler un système d'acquisition 3D de basse qualité.

de recherche approximatif des plus proches voisins : FLANN [ML12, ML09]. Pour mieux représenter les différences des résultats, nous présenterons les surfaces NLPSS maillées finement. L'entrée de notre algorithme restera malgré tout un nuage de points muni de normales et non une surface maillée.

3.3.2 Analyse des paramètres

Dans cette partie nous allons présenter en détail l'influence des paramètres de notre opérateur non local. Pour permettre une meilleure analyse des résultats, nous utiliserons, jusqu'à la fin de cette section, le même objet *Fandisk* comme entrée des différents algorithmes. Le nuage de points original a été bruité artificiellement avec un bruit uniforme de variance égale à l'espacement moyen entre les points du nuage (Figure 3.7). Toutes les mesures d'écart des surfaces résultantes seront effectuées par rapport au maillage original.

Échelle grossière

L'échelle grossière t_0 est utilisée pour définir la surface grossière S_{t_0} ainsi que le champ scalaire de déplacement résiduel bruité $m(\mathbf{x})$. Pour que l'utilisation des méthodes non locales ait du sens, cette dernière doit définir un signal autosimilaire bruité dont les variations principales sont liées aux traits principaux de l'objet considéré. Nous présentons Figure 3.8 différents choix pour l'échelle t_0 ainsi que le champ scalaire résiduel et les surfaces NLPSS associées.

Le choix idéal pour l'échelle t_0 (représenté en vert sur la Figure 3.8) est dépendant de la quantité de bruit appliqué à l'objet. Si nous choisisson une valeur de t_0 trop faible (cas $t_0 = 5$), la surface grossière résultante exhibera uniquement les structures de bruit locales introduisant un champ scalaire dont les variations ne sont pas liées à l'aspect de l'objet considéré. De la même manière, si t_0 est trop large (cas $t_0 = 30$) les lignes caractéristiques de l'objet seront filtrées excessivement entraînant l'apparition de variations parasites dans le champ de déplacement résiduel.

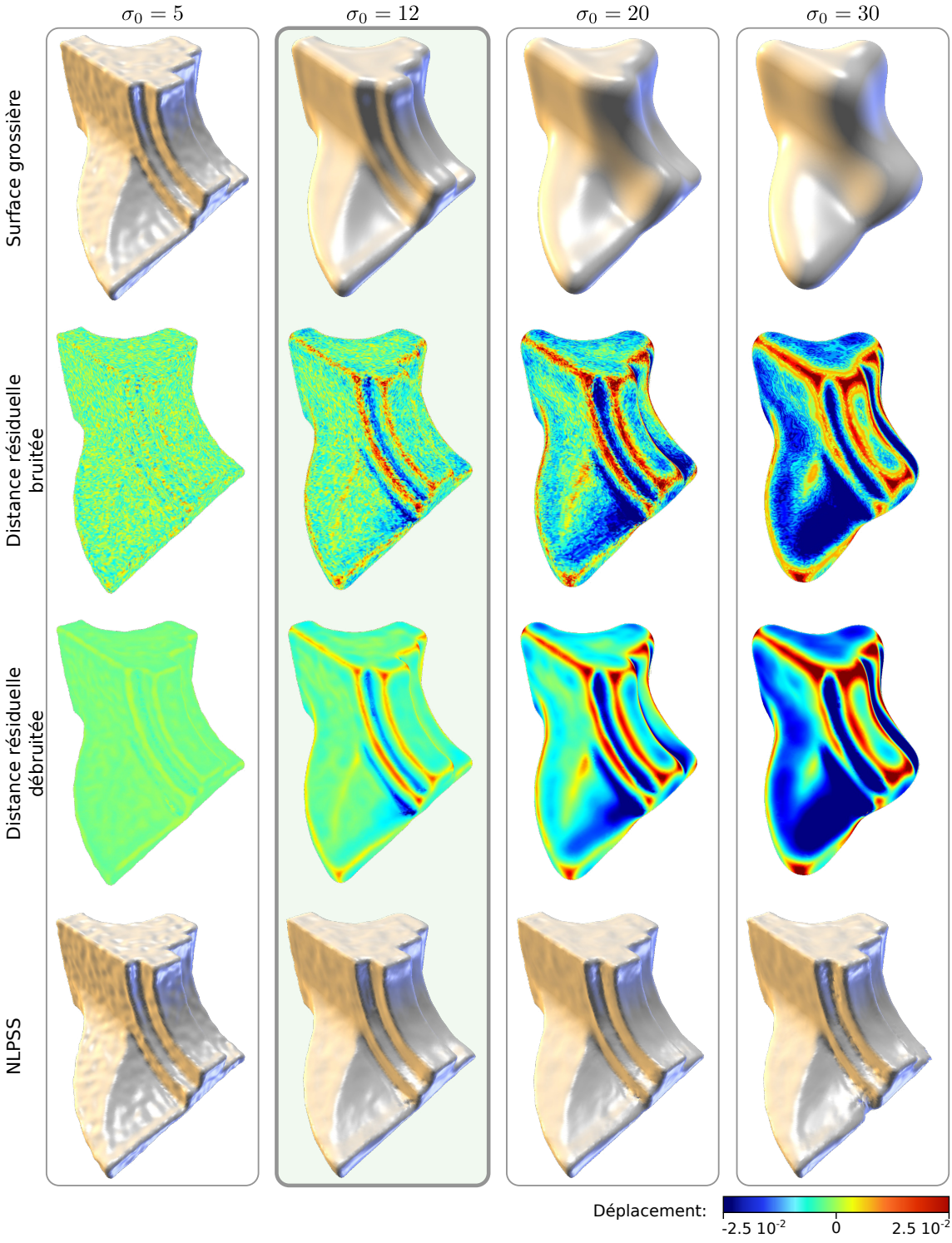


FIGURE 3.8: Évolution de la surface NLPSS en fonction de l'échelle grossière t_0. Cette dernière permet de définir une surface dont le bruit et les caractéristiques principales ont été lissés et *a fortiori* une carte de déplacement résiduel contenant les variations principales de l'objet. Un choix idéal pour le paramètre t_0 a été représenté en vert.

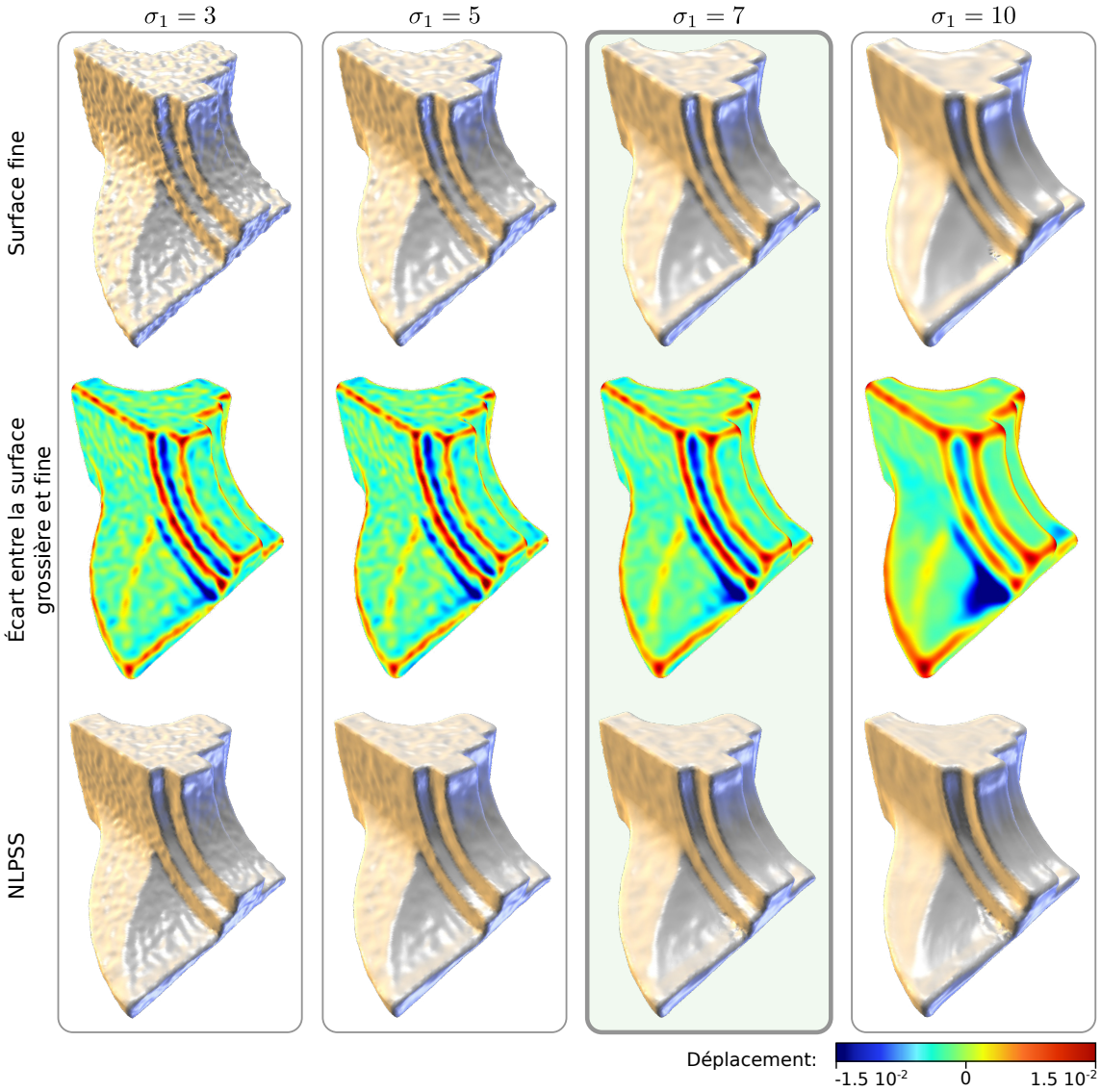


FIGURE 3.9: Évolution de la surface NLPSS en fonction de l'échelle fine t_1. Cette dernière permet de définir la carte d'écart entre la surface grossière S_{t_0} et S_{t_1} utilisée par nos patchs pour définir la mesure de similarité de deux sous-ensembles du nuage de points. Un choix idéal pour le paramètre t_1 a été représenté en vert.

Par conséquent, la valeur t_0 idéale doit être choisie de manière à définir un champ scalaire dont les variations correspondent à l'aspect de l'objet. En pratique, si la valeur de t_0 est choisie entre 10 et 20 fois l'écart moyen des points du nuage.

Échelle fine

L'échelle fine t_1 définit une surface fine S_{t_1} utilisée par notre mesure de similarité pour estimer les variations des patchs. Pour définir une surface d'une meilleure qualité, t_1 doit être choisi de manière à ce que la mesure de similarité soit peu dépendante des structures

locales de bruits. En Figure 3.9, nous présentons différents choix pour t_1, la carte d'écart entre S_{t_0} et S_{t_1} ainsi que les surfaces NLPSS associées.

Une fois encore, le choix idéal de l'échelle t_1 (représenté en vert sur la Figure 3.9) dépend du niveau de bruit appliqué à l'objet. Elle doit être choisie suffisamment petite par rapport à t_0 de manière à approcher les caractéristiques de l'objet tout en lissant les structures locales de bruit. Pour des valeurs de t_1 faibles (cas $t_1 = 3$), la mesure de similarité rassemblera des structures locales de bruits similaires exagérant ainsi les variations parasites de l'objet. Au contraire pour des valeurs de t_1 élevées (cas $t_1 = 10$), la mesure de similarité associera ensemble des traits non similaires de l'objet tendant à introduire des artefacts et à uniformiser les traits principaux de la surface NLPSS résultante.

Choisir une valeur de t_1 supérieure à t_0, n'aurait pas de sens et résulterait d'un mauvais choix de t_0. De plus, prendre $t_1 = t_0$ définirait un ensemble de patchs égaux rendant impossible toute mesure de similarité de deux sous-ensembles du nuage de points. En pratique, nous choisissons $t_1 \approx 0.5t_0$.

Opérateur PSS local sous-jacent

Notre opérateur NLPSS est défini pour étendre n'importe quel opérateur PSS local afin d'exploiter le caractère autosimilaire des nuages de points. Nous présentons en Figure 3.10 une comparaison des surfaces non locales générées en considérant différents opérateurs PSS locaux.

La qualité de notre NLPSS est très dépendante du choix du PSS local sous-jacent. Ce dernier permet de définir la surface grossière et la surface fine à partir du nuage de points (Figure 3.11). Il est important de noter que les propriétés intéressantes des PSS locaux sont transmises au NLPSS. Ainsi, un opérateur qui préserve les bords d'un objet produira un NLPSS qui les préserve également (*cf.* NLPSS basé sur le RIMLS). Notre définition permet également de corriger les imperfections introduites par les PSS locaux. Dans le cas du SPSS et HPSS qui exagèrent les caractéristiques principales d'un objet sans filtrer les structures de bruits locales, notre NLPSS réussit à définir une surface les supprimant tout en préservant les arêtes et les coins de l'objet.

Malheureusement, notre définition est très dépendante de la qualité de la surface grossière : toutes les discontinuités introduites par l'opérateur PSS local à échelle grossière seront difficilement corrigées par notre approche. De manière équivalente, un PSS local capable de définir une surface fine qui préserve les caractéristiques principales d'un objet définira une mesure de similarité plus précise. Mesure qui est utile pour définir une surface non locale de meilleure qualité. L'APSS et le RIMLS qui génèrent des surfaces grossières exemptes de défauts remarquables et des surfaces fines préservant les caractéristiques de l'objet définissent des NLPSS idéaux. En pratique, nous utilisons l'APSS comme opérateur de notre NLPSS car il donne de manière générale les meilleurs résultats.

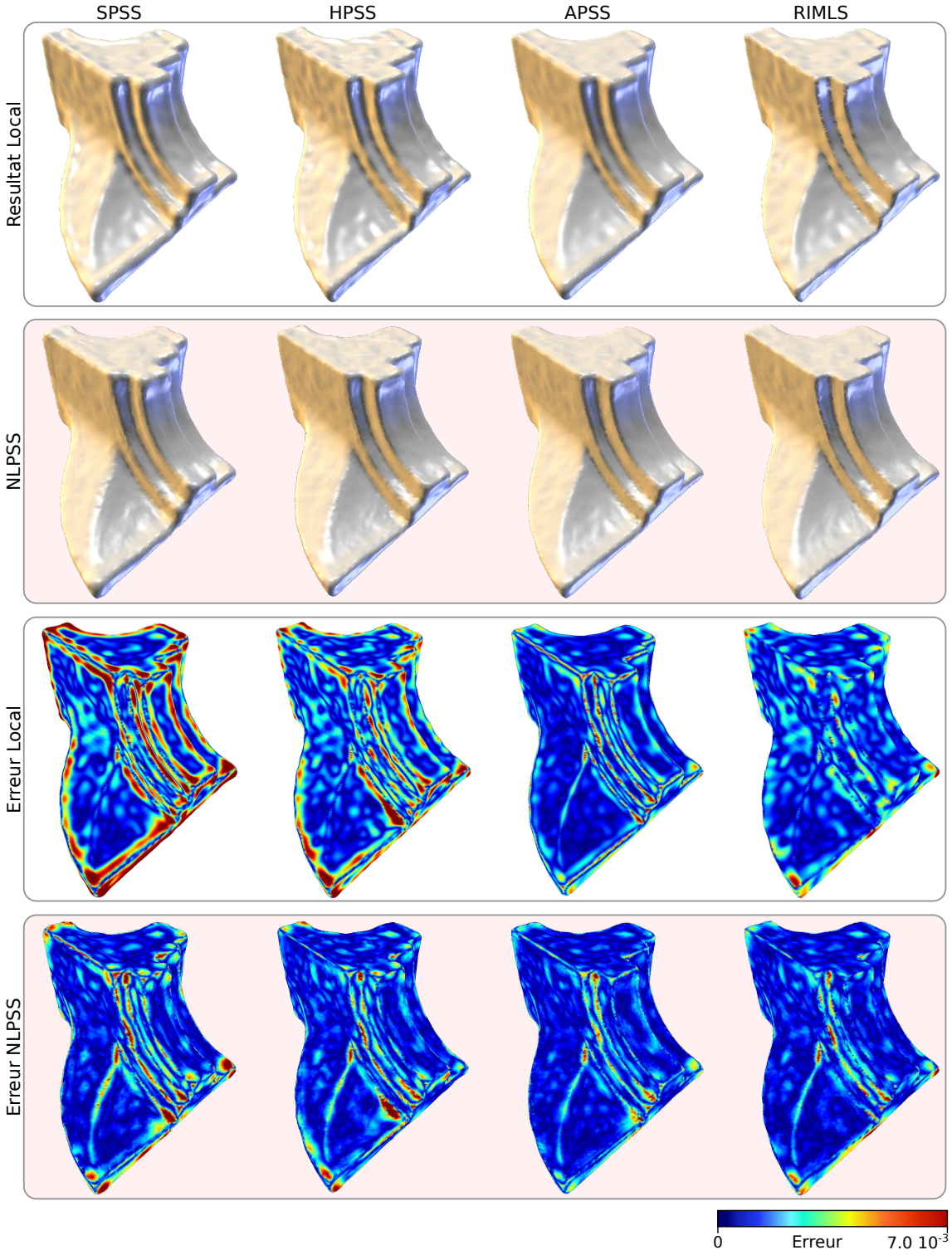


FIGURE 3.10: Comparaison des surfaces NLPSS utilisant différents opérateurs PSS sous-jacents et des surfaces PSS originales. Les erreurs ont été calculées par rapport au *Fandisk* original.

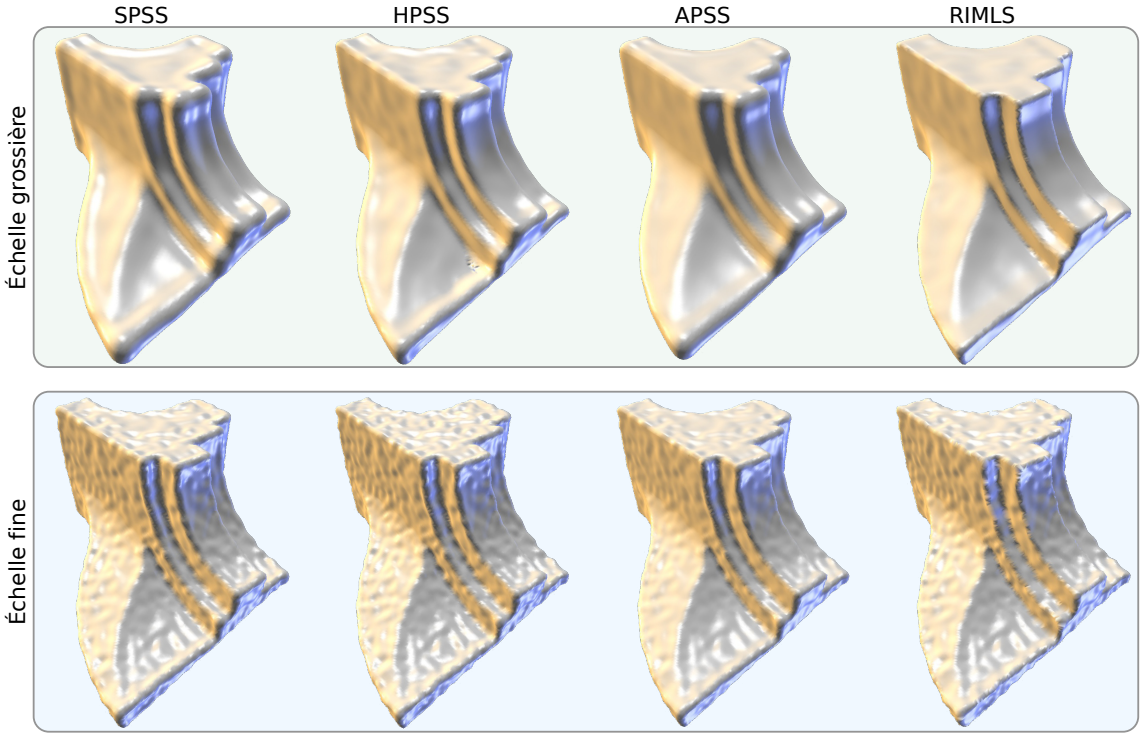


FIGURE 3.11: Surfaces fines S_{t_1} et grossières S_{t_0} utilisées par notre NLPSS pour étendre les PSS conventionnels.

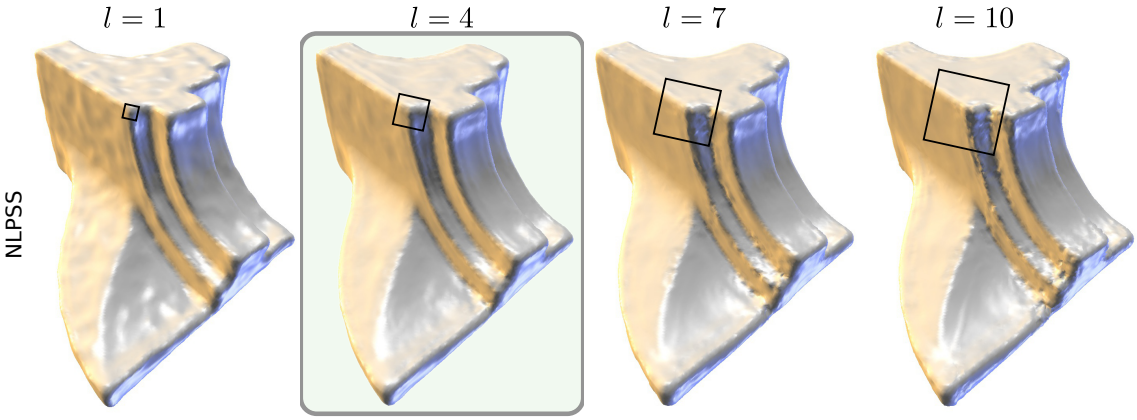


FIGURE 3.12: Évolution de la surface NLPSS en fonction de la taille du patch l. Cette taille est utilisée pour définir la taille des détails qui doivent être pris en compte lors du débruitage par les méthodes non locales. Si cette dernière est choisie trop fine, notre approche non locale dégénère en une approche bilatérale. La définition du patch sur un plan tangent à la surface grossière S_{t_0}, implique que le paramètre l ne doit pas être choisi supérieur à l'échelle fine t_1. Le choix idéal pour l a été représenté en vert.

Taille du descripteur

La largeur l des patchs permet de sélectionner la taille des traits de l'objet présents dans le nuage de points que notre NLPSS doit récupérer. En Figure 3.12, nous présentons différents choix pour la taille l du descripteur ainsi que les surfaces NLPSS résultantes.

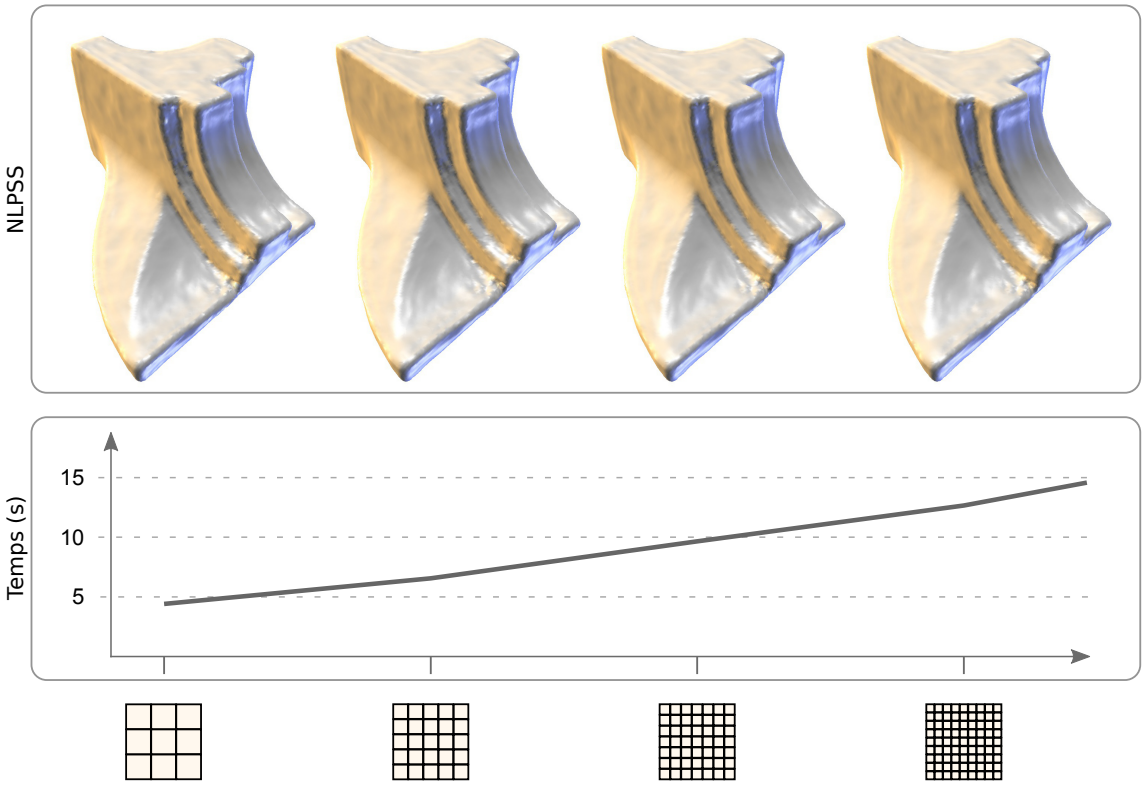


FIGURE 3.13: Évolution de la surface NLPSS en fonction du nombre de points n par patch. Les surfaces résultantes sont peu influencées par le nombre de points compris dans chaque patch. Néanmoins, le temps de calcul pour estimer ces dernières augmente linéairement avec le nombre de points. En définitive, des patchs de taille 5×5 semblent être un bon compromis qualité/temps de calcul.

Il est intéressant de noter que pour des valeurs faibles de l (cas $l = 1$), notre approche non locale dégénère en un simple opérateur de projection bilatéral [TM98, JDZ04]. La surface NLPSS résultante aura donc tendance à s'accrocher aux valeurs du champ scalaire résiduel les plus proches. Pour des valeurs de l plus élevées (cas $l = 4$), la mesure de similarité est peu influencée par le bruit présent dans le nuage de points résultant en une suppression des variations locales dans les zones plates.

Néanmoins, à cause de l'approximation de la surface grossière par un plan tangent à la surface (*cf.* Section 3.2.3), l doit être choisie inférieure à t_1. Pour des valeurs de l supérieures à t_1 (cas $l = 10$) les variations décrites par les patchs ne correspondent pas aux variations réelles entre la surface grossière et la surface fine. Ainsi, la mesure de similarité est faussée ce qui entraîne l'apparition d'artefacts dans la surface NLPSS. En pratique, nous choisissons l variant entre 3 et 5 fois l'écart moyen des points du nuage.

Nombre de points par patch

Le nombre de points du patch n est utilisé pour décrire plus précisément les variations locales de la surface. Figure 3.13, nous présentons une analyse du temps de calcul des

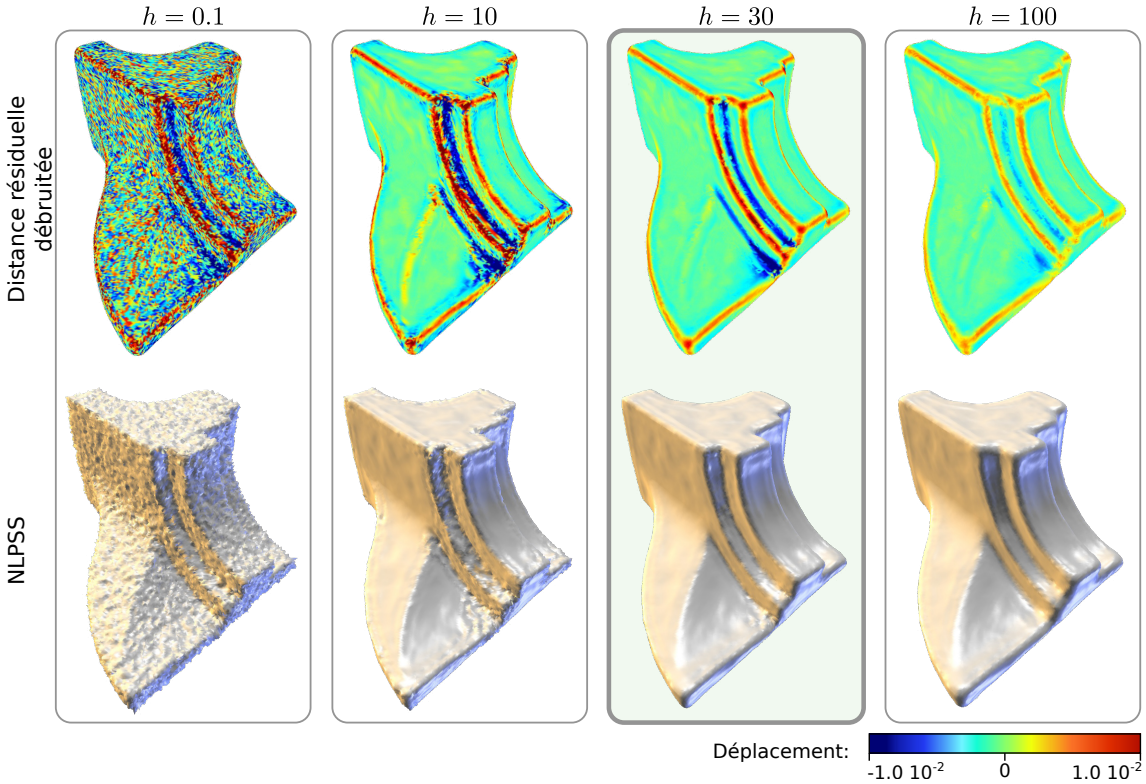


FIGURE 3.14: Évolution de la surface NLPSS en fonction du degré d'autosimilarité h du nuage de points. Ce dernier permet de sélectionner le degré de débruitage des caractéristiques principales de l'objet. Si ce dernier est choisi trop faible, les caractéristiques de l'objet resteront bruitées. À l'opposé, pour des valeurs trop élevées ces dernières seront complètement lissées. Le choix idéal (représenté en vert) correspond au degré de filtrage de notre méthode. L'utilisateur peut interagir avec ce paramètre sans contrainte de temps de calcul.

patchs en fonction de la qualité de la surface NLPSS associée à des nombres de points différents par patchs.

De toute évidence, augmenter le nombre de points par patch permet de définir une mesure de similarité plus fiable en contrepartie d'un temps de calcul élevé. En pratique, l'influence du nombre de points par patchs reste négligeable. Ainsi, le choix d'un nombre points par patch faible s'impose. Dans de rares cas, le choix de patch 3×3 peut être insuffisant pour décrire correctement la différence entre les surfaces grossière et fine. Par conséquent, nous avons opté pour des patchs 5×5 ce qui constitue un bon compromis qualité/temps de calcul.

Facteur d'autosimilarité du nuage

Le paramètre h définit le degré de similarité présent dans le nuage de points et correspond au facteur de filtrage du nuage de points. Nous présentons Figure 3.14 différents choix pour h ainsi que le champ scalaire résiduel débruité et les surfaces NLPSS associées.

Pour des valeurs $h \approx 0$ (cas $h = 0.1$), la mesure de similarité aura tendance à différencier l'ensemble des patchs ce qui s'exprimera en pratique par des poids qui tendent vers 0. Comme la somme des poids de l'équation 3.11 est assurée d'être unitaire, les points prendront la valeur $m(\mathbf{x}_i)$ du point p_i le plus similaire. Par conséquent, le champ scalaire résiduel ne sera pas débruité et la surface non locale résultante approximera le bruit contenu dans le nuage. De même, pour des valeurs trop faibles de h (cas $h = 10$), les lignes singulières de l'objet seront peu débruitées. À l'opposé des zones plates, ces zones sont plus rares dans l'objet entraînant un débruitage plus faible. À l'opposé, si h est choisi trop grand (cas $h = 100$), les poids de l'équation 3.11 seront tous égaux entraînant une uniformisation du champ scalaire de déplacement résiduel.

La valeur idéale de h (représentée en vert sur la Figure 3.14) doit être choisie par l'utilisateur et correspond au facteur de filtrage de notre méthode. Une fois l'ensemble des patchs calculés et les k plus proches patchs déterminés, l'estimation d'une surface pour différentes valeurs de h n'est pas coûteuse. Ainsi, l'utilisateur peut intéragir avec le paramètre h sans contrainte sur le temps de calcul.

Facteur de dégénérescence local

L'utilisation du facteur α de l'équation 3.14 nous permet de choisir des valeurs h moins élevées. Ainsi, les traits peu présents dans l'objet sont mieux préservés et les artefacts introduits par les éléments les plus rares de l'objet sont lissés par l'opérateur local. En Figure 3.15, nous présentons l'évolution de la surface NLPSS pour différentes valeurs de α.

Lorsque $\alpha = \infty$, l'équation 3.15 dégénère simplement en une définition locale des PSS. Au contraire quand $\alpha = 0$, le résultat obtenu correspond à la surface NLPSS. Les valeurs intermédiaires de α permettent d'obtenir une combinaison de ces deux surfaces. Pour des valeurs croissantes de α, les traits les moins présents dans le nuage sont les premiers à être remplacés par les valeurs des PSS locaux permettant d'améliorer la qualité de la surface NLPSS résultante. Ainsi, comme le montre la figure 3.15 pour des valeurs croissantes de α, les coins, les arêtes puis finalement les zones plates vont être remplacés par les PSS locaux.

La valeur idéale dépendra de la présence de points peu autosimilaires dans le nuage. Il est intéressant de noter que changer la valeur du facteur α ne nécessite que peu de calcul supplémentaire. En effet l'équation 3.2.4 peut s'exprimer comme une combinaison linéaire des points des nuages résultants de l'opérateur PSS local et non local.

Choix pratique

Parmi les sept différents paramètres que possède notre NLPSS, seulement deux d'entre eux ont besoin d'être réellement choisis par l'utilisateur : le facteur d'autosimilarité h et le facteur de dégénérescence α.

En pratique, pour des nuages de points issus de scanners 3D dont l'écart moyen entre les points est défini par $\bar{\delta}$, les meilleurs résultats sont obtenus pour des NLPSS basés sur l'APSS dont l'échelle grossière $t_0 = 12\bar{\delta}$, l'échelle fine $t_1 = 7\bar{\delta}$ et les patchs de taille $l = 4\bar{\delta}$

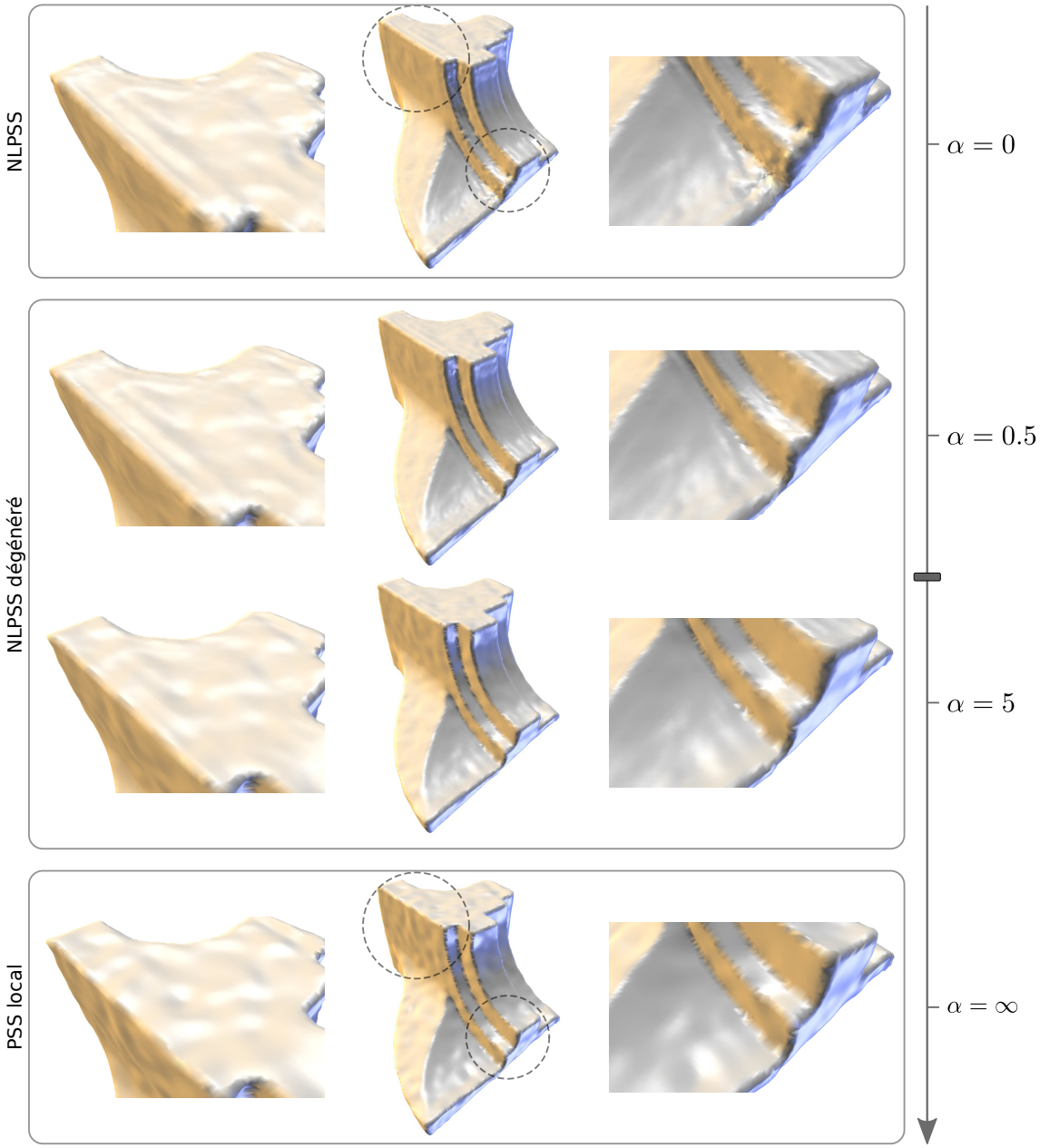


FIGURE 3.15: Évolution de la surface NLPSS en fonction du facteur α. Ce paramètre est complémentaire au paramètre h et permet de remplacer les structures peu débruitées (peu autosimilaires) par leurs versions débruitées en utilisant l'opérateur PSS sous-jacent. Pour des valeurs croissantes de α notre définition remplace en premier les structures peu autosimilaires pour finir par les caractéristiques les plus présentes dans le nuage de points. Nous pouvons par conséquent faire un compromis entre localité (pour les structures peu autosimilaires) et similarité (pour les structures très autosimilaires). Calculer les surfaces résultantes pour différentes valeurs de α est possible sans avoir besoin d'estimer de nouveau l'ensemble des patchs et les k plus proches voisins. α peut donc être choisi interactivement par l'utilisateur.

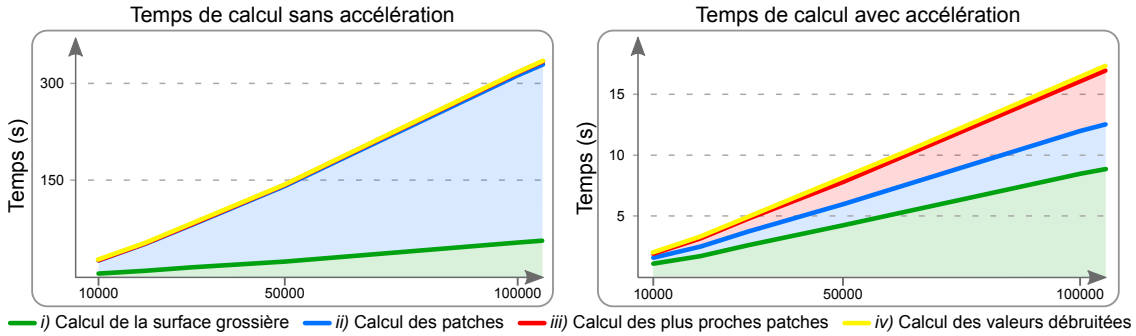


FIGURE 3.16: *Gauche* : Temps de calcul des différentes étapes de notre approche estimés en utilisant un seul cœur. Notre approche est plus longue que les opérateurs PSS précédents, car elle nécessite un plus grand nombre de projections (dû aux projections successives sur la surface grossière et sur la surface fine). *Droite* : Temps de calcul des différentes étapes de notre approche en utilisant différents outils d'accélération (parallélisation, GPU). En pratique, nous avons grandement accéléré le temps de calcul (20x) permettant de définir une utilisation plus interactive de notre NLPSS.

sont définis avec une précision de 5×5 points. De plus, dans la mesure où ces paramètres sont fixés, le calcul des patchs du nuage peut être effectué une fois pour toutes. Ainsi, les surfaces résultantes peuvent être générées rapidement pour différentes valeurs de h et α.

3.3.3 Performances

Notre approche non locale reste plus lente que des méthodes PSS conventionnelles. Ces performances inférieures peuvent s'expliquer par : *(i)* un nombre de projections sur les surfaces plus conséquent que les approches locales, *(ii)* la nécessité de comparer l'ensemble des patchs entre eux. Nous présentons en Figure 3.16, l'évolution des temps de calcul pris par chacune des étapes de notre algorithme en fonction du nombre de points à projeter.

Certains paramètres du NLPSS influencent directement le temps de calcul, comme les échelles t_0 et t_1 du PSS sous-jacent. Ces dernières sont directement liées à la taille du voisinage nécessaire pour estimer les projections sur les surfaces grossières et fines. Ainsi, plus ce voisinage est grand, plus le temps de calcul est élevé. De la même manière, le nombre de points influence le nombre de projections nécessaires (Figure 3.13). Néanmoins comme nous l'avons expliqué en Section 3.3.2, le nombre de points par patch est fixé car il influence peu la qualité de la surface résultante. Pour estimer rapidement le voisinage d'un point donné, chaque PSS est basé sur un *ball-tree* tels qu'il est introduit par Guennebaud et coll. [GGG08]. De même, pour accélérer la comparaison des patchs, la moyenne non locale définie par l'Équation 3.13 est limitée à une recherche approximée des 500 plus proches voisins du patch considéré par FLANN [ML12, ML09].

Pour une meilleure interactivité avec l'utilisateur, ces temps de calcul sont grandement diminués (Figure 3.16) grâce à des outils de parallélisation CPU et GPU. De plus, notre approche peut être divisée en quatre étapes successives : *(i)* calcul de la surface grossière, *(ii)* estimation des patchs, *(iii)* calcul approché des plus proches voisins et *(iv)* calcul des valeurs débruitées (réglage de h et α). Comme les paramètres utilisés par notre approche

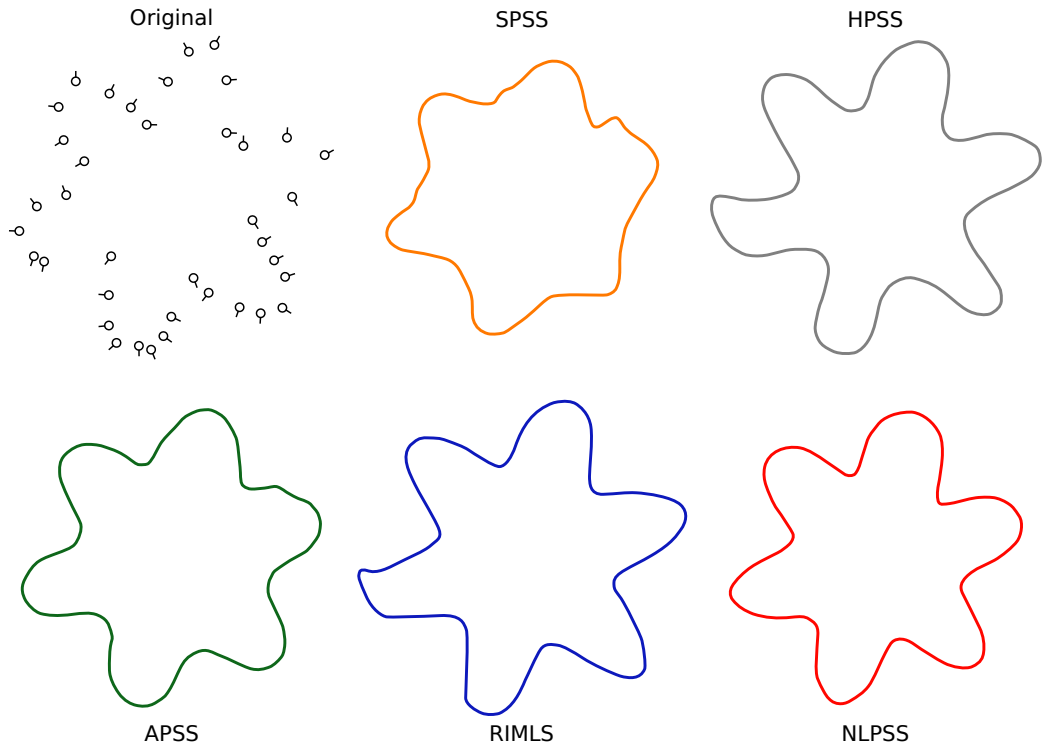


FIGURE 3.17: Comparaison entre les reconstructions PSS locales et notre NLPSS obtenues à partir d'un nuage de point 2D.

peuvent être généralisés, seule l'étape *(*iv) nécessite une interaction réelle avec l'utilisateur. De plus, une fois les étapes précédentes estimées, cette dernière peut être calculée rapidement (comme le montre le graphique de droite de la Figure 3.16) sans avoir besoin de réappliquer la chaîne complète. Les valeurs idéales de h et α peuvent donc être choisies interactivement par l'utilisateur.

3.3.4 Analyse de la qualité de la surface

Les opérateurs PSS conventionnels essayent d'extraire une surface à partir d'un voisinage purement local. Comme le montre la Figure 3.17 qui présente différentes reconstructions de surfaces PSS à partir d'un nuage de points 2D synthétique, ce point de vue ne permet pas de conserver les caractéristiques présentent dans le nuage de points d'entrée supprimant par la même occasion les symétries originales du nuage d'entrée. En utilisant un point de vue plus global, notre NLPSS permet de générer une surface capable de conserver les symétries présentes dans l'objet original.

Dans le cas d'un nuage de points réel issu d'un scanner 3D (Figure 3.18), les opérateurs PSS conventionnels doivent, en plus, faire un compromis entre la préservation des caractéristiques principales de l'objet et la suppression du bruit. Ce problème est particulièrement visible pour les SPSS qui tendent à trop filtrer les arêtes et pour le HPSS qui exagère les structures locales de bruits. D'autres opérateurs, comme l'APSS, possèdent un fort pouvoir débruitant, mais doivent être utilisés à des échelles de filtrage fines pour éviter une sup-

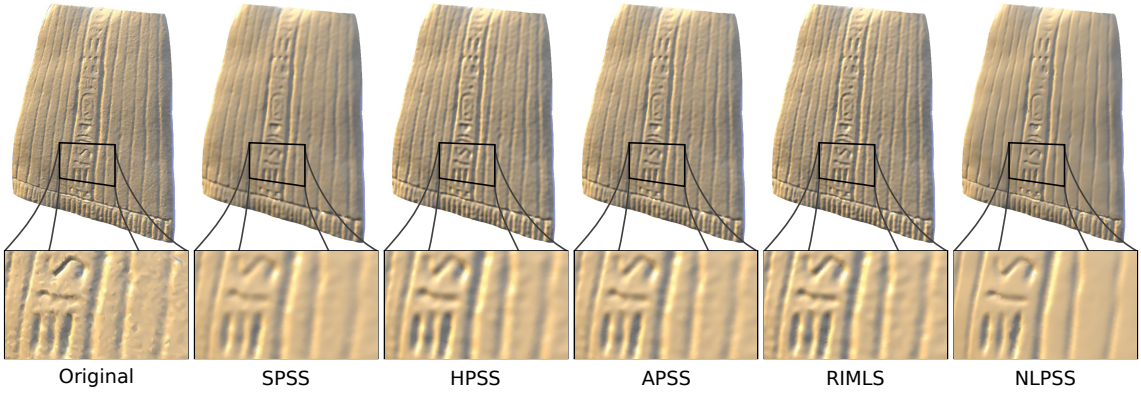


FIGURE 3.18: Comparaison entre différentes surfaces obtenues par des PSS locaux et notre NLPSS obtenues à partir du modèle 3D scanné *Ramesses*

pression complète des arêtes de l'objet. Malheureusement, à de telles échelles, ce dernier est incapable de supprimer complètement le bruit. Contrairement aux autres opérateurs, le RIMLS a été défini pour extraire les arêtes tout en supprimant le bruit de l'objet. Cette définition fournit de bons résultats dans le cas d'objets purement géométriques possédant des arêtes vives (par ex. objets manufacturés). Malheureusement, dans le cas d'objets réels, le RIMLS tend à exagérer les caractéristiques de l'objet.

En comparaison, notre NLPSS réussit à éliminer le bruit introduit par la numérisation sans altérer les structures présentes dans l'objet. Cet avantage est dû à l'utilisation de l'information non locale qui peut augmenter localement le rapport signal sur bruit. Contrairement au RIMLS qui exagère les structures, notre opérateur NLPSS réussit à générer une surface qui est proche du nuage d'entrée.

Il est important de comprendre que notre définition de surface non locale est plus générale qu'une simple combinaison d'un algorithme de filtrage de nuage de points (par exemple [DM11, Dig12]) et d'un modèle de reconstruction de surface existant. Pour mieux appréhender les différences fondamentales entre ces deux types d'approches, nous présentons en Figure 3.19 deux reconstructions de surfaces différentes. Dans le premier cas, la surface est définie en utilisant L'APSS à partir d'un nuage de points préalablement débruité par un algorithme de débruitage non local. Dans le deuxième cas, la surface est directement définie en utilisant notre NLPSS. La première approche réussit à débruiter les points en exploitant l'autosimilarité du nuage, mais reste incapable de reconstruire une surface en exploitant l'autosimilarité du nuage. Ainsi, la surface résultante ne permet de compléter les trous en respectant le caractère périodique du nuage original. Au contraire, notre modèle NLPSS est capable de reconstruire la surface *pour chaque point* en exploitant l'autosimilarité et la redondance des structures du nuage de points. De manière similaire, une troisième approche, qui consisterait à reconstruire une surface directement à partir des points du nuage puis à utiliser une méthode non locale de débruitage de maillage surfacique [FDCO03, YBS06, WCZ+08, MRS12], n'exploiterait pas totalement l'autosimilarité du nuage tout en introduisant des problèmes de définition.

Comme le montre la Figure 3.20, dans le cas de nuages très épars et bruités, les modèles PSS existants peuvent difficilement générer une surface correcte à des tailles de filtrage

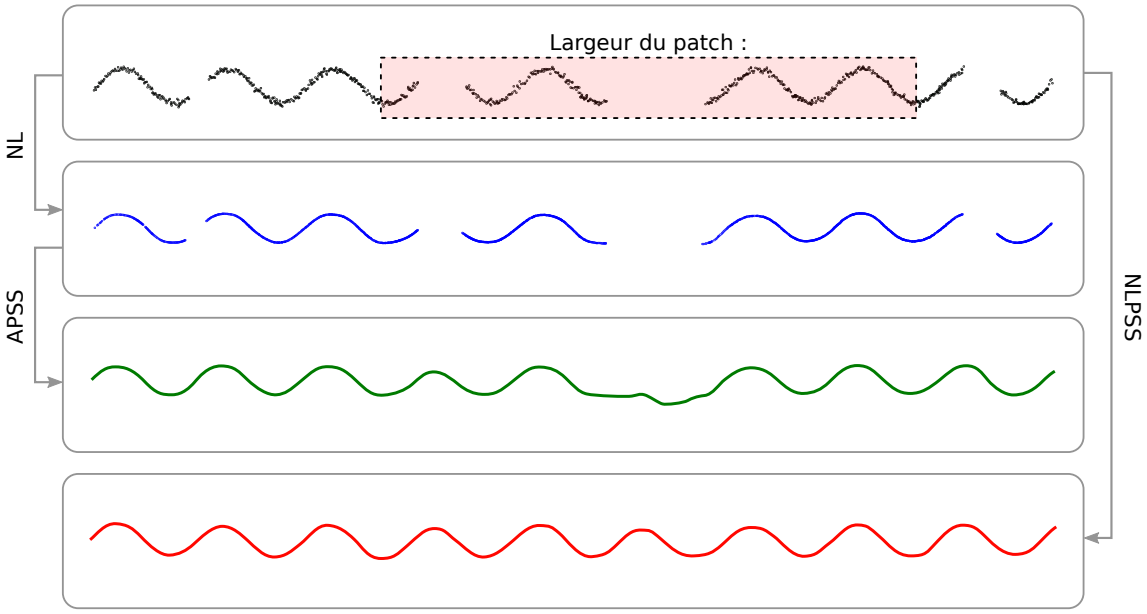


FIGURE 3.19: Comparaison des surfaces obtenues en utilisant l'APSS [GGG08] à partir d'un nuage de points débruité par un algorithme de filtrage non local et la surface obtenue directement avec notre définition de surface NLPSS.

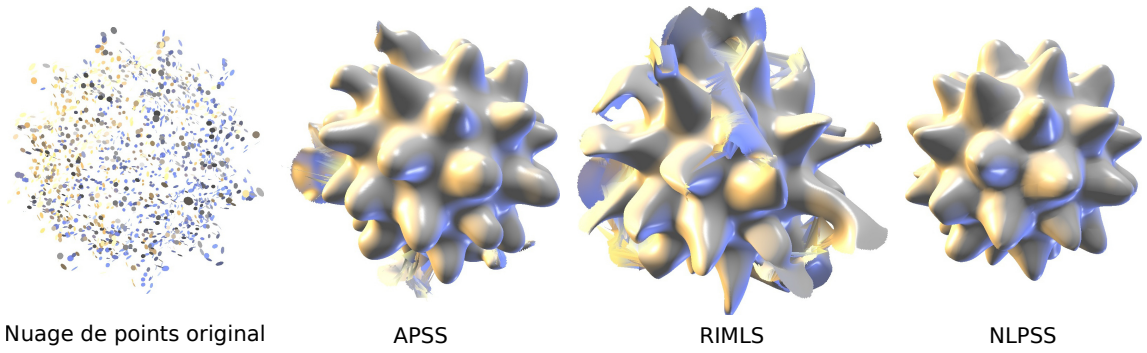


FIGURE 3.20: Comparaison entre différentes surfaces obtenues en utilisant des opérateurs PSS locaux et notre NLPSS sur un modèle d'étoile 3D synthétique.

petites. Pour résoudre ce problème, les modèles de PSS conventionnels doivent augmenter leur niveau de filtrage, supprimant par conséquent toute l'information de basse échelle présente dans le nuage. Contrairement à ces définitions, notre NLPSS est capable de générer une surface plus stable. Cette propriété est due à *(i)* l'utilisation d'une surface grossière comme base à notre surface qui définit une structure topologique à notre PSS et *(ii)* la moyenne non locale qui est capable de débruiter des structures fines et de les rajouter sur la surface grossière sans introduire de fausses structures comme le font les modèles de PSS conventionnels.

En Figure 3.21 et 3.22, nous présentons les surfaces obtenues par notre NLPSS à partir de nuages de points bruités issus de scanners 3D. Il est intéressant de noter que notre NLPSS est capable d'utiliser l'ensemble de l'information pour compléter les trous du nuage de

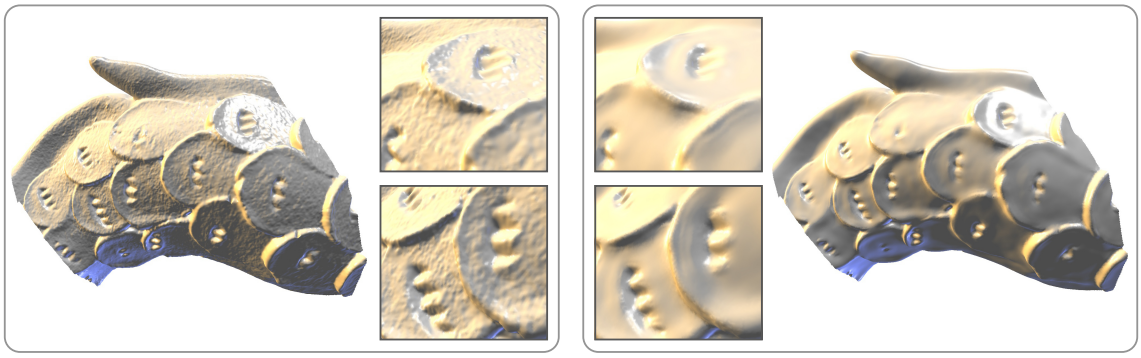


FIGURE 3.21: Illustration du pouvoir débruitant de notre NLPSS sur le nuage de points 3D *Dragon* (413k points) acquis par numérisation 3D.

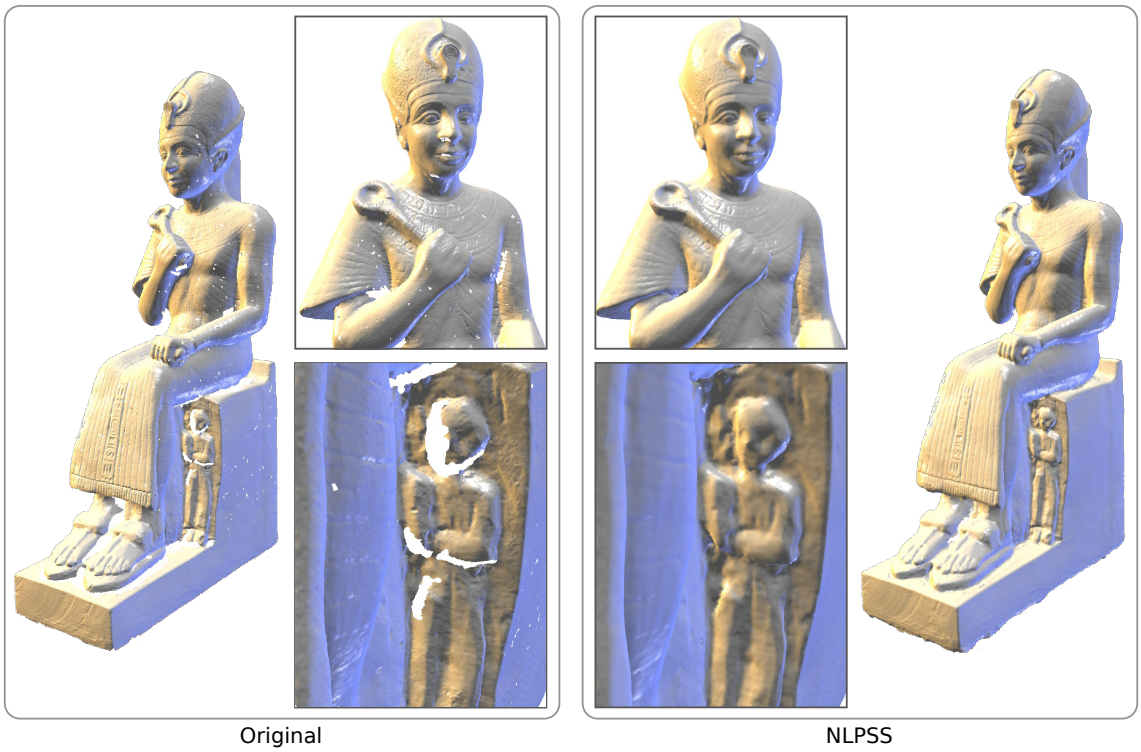


FIGURE 3.22: Application du NLPSS sur le nuage de points *Ramesses* (350K points) issu d'une numérisation 3D. Notre approche est capable de combler les trous de l'objet en utilisant l'autosimilarité présente dans le nuage de points.

points. Cette propriété est illustrée en Figure 3.22) où l'information inconnue du visage d'un relief est complétée par l'information connue d'un relief similaire présent sur la statue. Néanmoins, cette propriété reste très dépendante de la qualité de la surface grossière sous-jacente.

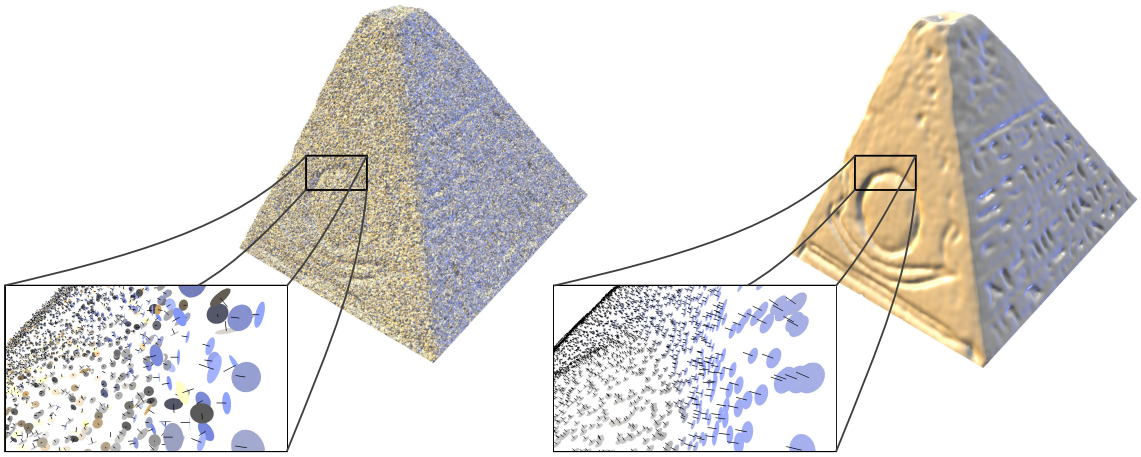


FIGURE 3.23: Application de notre opérateur NLPSS au cas du débruitage de nuages de points. Nous avons utilisé le jeu de données *Pyramid* (120k points) issue de [DAL+11]

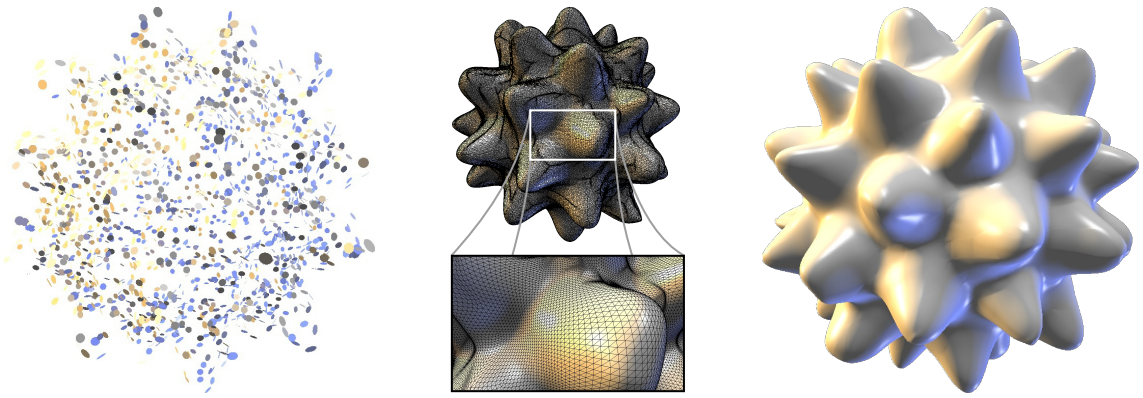


FIGURE 3.24: Application de notre NLPSS pour définir une surface maillée en utilisant un algorithme de type *Marching Cubes* [LC87].

3.3.5 Applications

Filtrage de nuages de points

Une des applications les plus évidentes des PSS est le filtrage de nuages de points. En considérant un nuage de points bruités, l'opération de débruitage du nuage consiste à remplacer chaque point $\mathbf{x}_i \in \mathcal{X}$ du nuage par leur projection $\tilde{\mathbf{x}}_i = \Pi_{NL}(\mathbf{x}_i)$ sur la surface définie par l'opérateur de projection Π_{NL} définie par l'Équation 3.12. Ainsi, le nuage débruité est défini par $\tilde{P} = \{\tilde{\mathbf{x}}_i\}$. Nous présentons en Figure 3.23, le résultat du débruitage obtenu par NLPSS.

Reconstruction de surface

L'une des utilisations majeures des opérateurs PSS est de définir une surface maillée à partir de la définition implicite de la surface. Nous définissons la surface comme la 0-

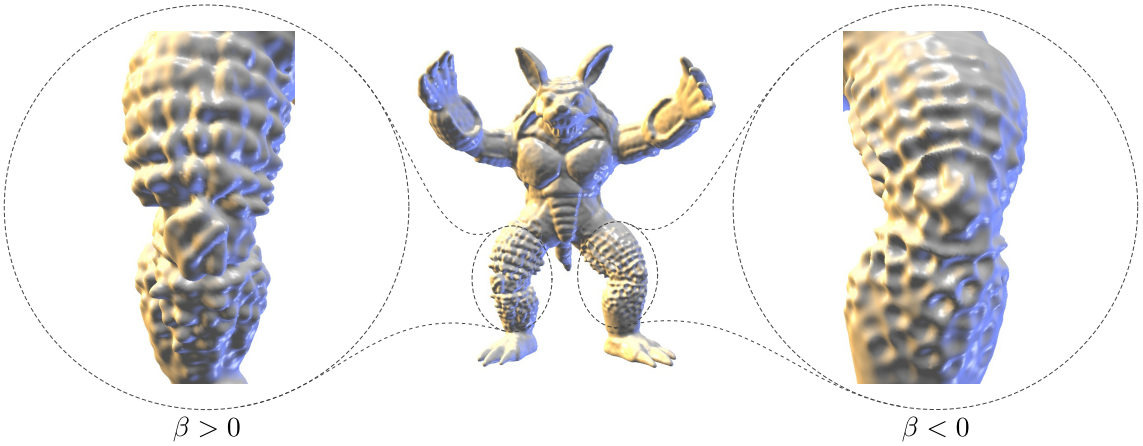


FIGURE 3.25: Application de notre NLPSS à l'édition de surfaces. Le paramètre β permet de moduler à quel point le champ scalaire de déplacement doit être ajouté à la surface grossière. *Gauche* : pour des valeurs $\beta > 1$, ces détails sont exagérés. *Droite* : pour des valeurs $\beta < 0$, ces derniers sont inversés.

surface de la distance implicite f_{NL} définie par l'Équation 3.13. La surface maillée est obtenue en utilisant un algorithme de *Marching Cubes*. Nous présentons en Figure 3.24 la surface NLPSS maillée obtenue à partir d'un nuage de points bruités.

Améliorations des détails

Pour illustrer le potentiel de notre NLPSS, nous proposons une utilisation de notre définition pour l'édition de surfaces. Comme nous l'avons expliqué en Section 3.2.1, notre opérateur NLPSS est défini par une surface grossière sur laquelle nous ajoutons un champ scalaire de déplacement. Nous pouvons modifier la fonction implicite f_{NL} définie en Équation 3.13 en multipliant le champ scalaire de déplacement par un champ scalaire continu $\beta(\mathbf{x})$:

$$f_{NL}(\mathbf{x}) = f^{t_0}(\mathbf{x}) - \beta(\mathbf{x}) \sum_{\mathbf{x}_i \in \mathcal{X}} w_{NL}(\mathbf{x}, \mathbf{x}_i) f^{t_0}(\mathbf{x}_i) \tag{3.15}$$

Par conséquent, $\beta(\mathbf{x})$ permet de décrire pour chaque $\mathbf{x}$ à quel point les détails fins $f_{NL}(\mathbf{x})$ sont ajoutés à la surface grossière S_{t_0}. Les détails sont ajoutés à la surface grossière pour des valeurs de $\beta(\mathbf{x}) > 0$, et exagérés si $\beta(\mathbf{x}) > 1$. Des valeurs de $\beta(\mathbf{x})$ négatives permettent de supprimer les détails de la surface grossière, inversant de la même manière les caractéristiques de l'objet. Ainsi, l'utilisateur peint directement sur la surface grossière S_{t_0} les valeurs de $\beta(\mathbf{x})$ choisissant, de la même manière, les éléments de l'objet qu'il souhaite exagérer. Nous présentons en Figure 3.25 une illustration de cette démarche.

3.4 Limitation et possibles améliorations

Dans cette partie nous avons défini un opérateur PSS capable d'exploiter le caractère autosimilaire des nuages de points pour définir une surface. Notre définition est capable

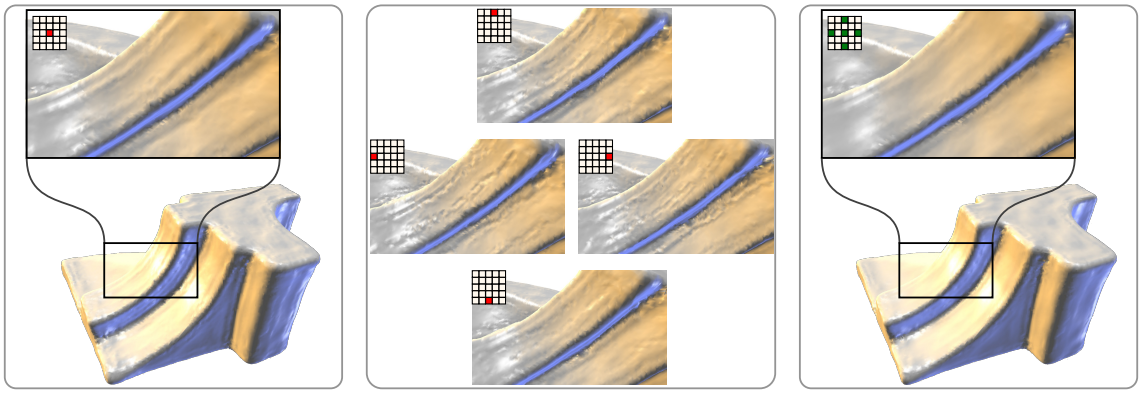


FIGURE 3.26: *Gauche* : Notre approche non locale peut résulter en l'apparition de *halos* sur la surface. *Centre* : Une solution simple pour résoudre ce problème consiste à définir différentes surfaces NLPSS en utilisant des patchs décentrés. *Droite* : Une simple moyenne de ces différentes valeurs débruitées permet de limiter les effets des *halos* introduits.

d'améliorer la qualité des surfaces définies par des opérateurs PSS locaux en augmentant le rapport signal/bruit. Néanmoins, les résultats de notre opérateur sont très dépendants de l'aspect autosimilaire du nuage à traiter. Plus le nuage de points présentera de parties similaires, plus la qualité de la surface générée par notre approche sera élevée. Dans le cas, peu probable, d'absence d'autosimilarité dans le nuage de points, notre NLPSS obtiendra des résultats identiques à l'opérateur PSS local sous-jacent (Section 3.2.4).

Il est courant que l'utilisation de patchs par les méthodes non locales introduise des oscillations ou *halos* autour des arêtes des images. Le même phénomène peut être observé sur les surfaces définies par notre approche autour des arêtes vives de l'objet (Figure 3.26). Ce problème peut s'expliquer par le phénomène d'*adhérence* des patchs. [LBM13] proposent une solution simple pour résoudre ce problème. Elle consiste à faire une moyenne des différentes cartes débruitées en considérant des patchs décentrés. Nous présentons en Figure 3.26 une extension de cette solution à notre NLPSS. Cette solution permet d'éliminer les halos et d'améliorer la qualité de la surface résultante au prix de temps de calcul plus coûteux.

Pour définir notre mesure de similarité (Section 3.2.3), nous avons utilisé une distance euclidienne entre les patchs. Cette distance nous permet de définir des surfaces NLPSS correctes, mais reste néanmoins trop sensible au bruit et aux points aberrants. Remplacer la norme L_2 par une norme L_1 permettrait de définir une surface moins sensible aux points aberrants. Il serait intéressant de redéfinir les patchs en ajoutant pour chaque point projeté la normale associée à cette projection. Ainsi, la distance entre les patchs correspondrait à une distance $L_{2,1}$ qui tient compte à la fois des variations des positions et des normales. Une telle approche revient à étendre les patchs de manières "géodésique" plus qu'euclidienne permettant ainsi de différencier les caractéristiques intrinsèques de l'objet tout en étant moins sensibles au bruit.

En comparaison avec les définitions de PSS conventionnelles, le NLPSS requiert un temps de calcul supérieur. Ce phénomène s'explique par la nécessité de projeter plus de points que les méthodes PSS précédentes et de rechercher les patchs les plus similaires. Durant

ces dernières années, beaucoup de travaux se sont fixés comme objectif d'accélérer le calcul des filtres à moyennes non locales (Section 4.1.1). Un point important de notre travail futur consistera à essayer d'adapter ces méthodes au cas des nuages de points et à utiliser les technologies de parallélisation offerte par les cartes graphiques modernes pour que notre approche soit utilisable en temps réel.

Pour conclure, notre NLPSS permet le remplissage de trous d'une taille similaire à celle utilisée pour définir les patchs de surface. Pour des trous d'une échelle supérieure, une telle approche n'est pas suffisante. Il est donc nécessaire d'utiliser une méthode itérative d'inpainting de surfaces. Malheureusement, de telles méthodes restent difficile à calculer et nécessitent la connaissance d'un *a priori* sur les surfaces 3D.

CHAPITRE 4

Arbre de covariances

Dans ce chapitre, nous présentons une structure de données capable d'apprendre les variations de grands ensembles d'échantillons en utilisant une quantité de mémoire limitée. En Section 4.1, nous présentons un historique des méthodes d'accélérations des filtres à hautes dimensions. Dans la Section 4.2, nous introduisons une version simplifiée de notre arbre définie dans un domaine spatial. Cette version est ensuite étendue en Section 4.3 en distinguant deux domaines : spatial et des attributs. Les performances de notre structure sont ensuite analysées en Section 4.4. Finalement, nous proposons en Section 4.5 diverses améliorations de notre structure de données.

4.1 Contexte

4.1.1 Filtrage à hautes dimensions

Avec l'apparition des nouvelles technologies et de nouvelles capacités de calculs, le filtrage à hautes dimensions est devenu durant ces dernières années une brique fondamentale pour une variété d'applications telles que le débruitage [BCM05], la récoloration [CPD07], le suréchantillonage [KCLU07], la manipulation de détails [BPD06,FAR07], le filtrage spatio-temporel [BM05]. Implémentés naïvement, de tels filtres restent difficiles et lents à calculer et nécessitent d'être accélérés. Ainsi, le rapport entre le temps de calcul et la qualité des résultats obtenue est devenu un enjeu important pour les communautés de la vision par ordinateur, du traitement des images et de la photographie calculatoire.

De nombreuses approches ont été introduites afin d'accélérer leur estimation. Parmi elles, une première catégorie de méthodes propose une accélération par estimation d'un champ approché des k-plus proches voisins comme par exemple *PatchMatch* [BSFG09] (récemment étendu à d'autres domaines [NFP+13, CFGS12, BRR11] et accéléré par *kd-tree* [HS12], l'utilisation d'images intégrales [OA12], l'utilisation d'une transformée de Fourier [WGY+06, DDS12], la réduction de la dimensionnalité des patches par les filtres de Haar [KA11] ou une analyse en composantes principales [APG07, Tas08, Tas09], la limitation de la recherche à un voisinage 2D local (utilisé en pratique par [BCM05, LBM13]).

Ces méthodes restent néanmoins trop spécifiques aux problèmes de traitement des images et peuvent difficilement être étendues au cas 3D. De plus, les approches de débruitage par

filtres collaboratifs, définissant une meilleure restauration de l'image que les approches par moyenne non locale, ne peuvent être accélérées par une estimation approchée des k-plus proches voisins. Par conséquent, nous focaliserons notre étude sur l'utilisation de structures d'accélération qui permettent une réduction de la complexité computationnelle des filtres au prix d'approximations visuelles résultantes.

4.1.2 Travaux existants

Parmi tous les filtres non linéaires, le filtre bilatéral [TM98] a été l'un des premiers à avoir été accéléré car il constitue une approximation intéressante des filtres anisotropes [Bar02]. Durand et Dorsey [DD02] sont parmi les premiers à proposer une accélération de ce filtre appliquée à l'affichage d'images à grande dynamique (High Dynamic Range - HDR). Leur idée principale est de réduire la complexité du filtre en utilisant une approximation linéaire par morceaux du filtre bilatéral, ce qui revient à appliquer un filtre gaussien, plus rapide à calculer, sur un sous-échantillonnage du domaine spatial. Ces valeurs sous-échantillonnées sont ensuite interpolées pour obtenir le signal filtré.

En remarquant que le filtre bilatéral peut être exprimé par un filtre gaussien appliqué dans un espace de dimensions supérieures, Paris et Durand [PD06] introduisent la grille bilatérale. Cette dernière sous-échantillonne le signal à filtrer par des voxels de taille uniforme en exprimant le signal à filtrer comme une variété linéaire par morceaux définie dans un espace qui associe le domaine spatial et colorimétrique (appelé par la suite espace augmenté). Pour rendre l'accélération plus performante, l'espace colorimétrique est réduit à une unique dimension. Néanmoins, les distances en chrominance ne sont pas respectées ce qui introduit des effets de flou parasites. Ce problème est résolu dans une version ultérieure de leurs travaux [PD09] par l'utilisation de toutes les dimensions colorimétriques. La grille bilatérale, définissant désormais un volume 5D, nécessite une quantité de mémoire et en temps supérieur. En particulier pour des échelles de filtrage faibles, cette méthode devient impraticable.

Adams et coll. [AGDL09] proposent de remplacer le pavage régulier par un pavage adaptatif de l'espace augmenté. Ils introduisent une structure d'accélération indépendante de l'échelle de filtrage, le *kd-tree* gaussien (Gaussian KD-Tree - GKD-Tree), en se basant sur les travaux de Aray et coll. [AMN+98]. Ils proposent de paver l'espace augmenté avec des voxels de tailles non uniformes en utilisant un *kd-tree* [Ben75] qui regroupe les échantillons proches. Les valeurs des voxels sont estimées par diffusion des valeurs des échantillons qui lui sont proches (*splatting*), filtrées entre elles (*blurring*) et finalement interpolées pour définir les valeurs filtrées de chaque échantillon (*slicing*). Cette approche peut être généralisée à l'accélération de l'ensemble des filtres non linéaires à poids gaussiens (filtre gaussien, bilatéral, à moyenne non locale, etc.). Ultérieurement, Adams et coll. [ABD10] étendent leurs travaux en pavant l'espace avec des simplexes et en stockant les valeurs de ces derniers dans une table de hachage. Ainsi, la structure d'accélération permet d'appliquer des filtres non linéaires de hautes dimensions avec une complexité linéaire par rapport au nombre de points et polynomiale par rapport à la dimensionnalité du filtre.

Dans le cas des filtres à très hautes dimensions, la complexité polynomiale par rapport aux nombres de dimensions des précédentes approches ne permet pas qu'elles soient appliquées en temps réel. He et coll. [HST10] proposent de réduire la dimensionnalité du problème

en comparant indirectement chaque pixel de l'image par rapport à leur relation à un guide. Même si cette approche permet une accélération significative, elle introduit des artefacts dus à l'utilisation d'une distance non euclidienne. De la même manière, Gastal et Oliveira [GO11] proposent, dans le cas des filtres bilatéraux, l'utilisation d'une transformée de l'image qui déforme la géométrie du domaine spatial 2D du signal à filtrer de sorte que la distance spatiale entre deux points de l'espace déformé corresponde à la distance géodésique dans l'espace augmenté 5D. Ainsi, le filtre bilatéral est approximé par un filtre gaussien appliqué dans le domaine spatial.

Ces travaux sont ensuite généralisés au cas de filtres de plus hautes dimensions [GO12]. Gastal et Oliveira démontrent que le signal à filtrer peut rarement être défini par une variété linéaire. Par conséquent, l'utilisation de voxels n'est pas adaptée pour obtenir un sous-échantillonnage correct de l'espace. À la place, ils proposent l'utilisation de variétés non linéaires adaptatives qui sont définies en appliquant récursivement un filtre passe-bas sur les échantillons du signal. Le signal filtré est estimé en utilisant une approche similaire au GKD-Tree [AGDL09] : les valeurs des variétés sont estimées par diffusion des valeurs des échantillons (*splatting*), puis filtrées entre elles indépendamment des valeurs des autres variétés (*blurring*) et finalement interpolées pour définir les valeurs filtrées de chaque échantillon (*slicing*). Cette approche permet d'obtenir une complexité qui est à la fois linéaire par rapport à la dimensionnalité du filtre et du nombre d'échantillons.

4.1.3 Problématique

Si l'on devait résumer les progrès de ces dernières années, le calcul de filtres non linéaires a été accéléré de deux manières : *(i)* en redéfinissant les filtres non linéaires par des filtres linéaires exprimés dans des espaces augmentés qui associent les domaines spatial et colorimétrique *(ii)* en interpolant une version sous-échantillonnée du signal filtré adapté à l'échelle de filtrage. Même si les structures d'accélérations associées sont aujourd'hui capables d'appliquer en temps réels des filtres non linéaires à hautes dimensions, elles ne peuvent pas être utilisées pour accélérer le calcul de filtres probabilistes collaboratifs tels que les Non-Local Bayes. De plus, elles doivent être entièrement reconstruites pour chaque modification des paramètres de filtrage.

D'un autre point de vue, avec l'augmentation de la quantité d'information, peu de méthodes à l'heure actuelle exploitent réellement l'information contenue dans les grandes bases de données ou au prix d'une quantité de mémoire et de calculs beaucoup trop importante. Les structures d'accélération de filtrage non linéaire actuelles ne sont pas conçues pour être directement exploitées sur ces bases.

Nous proposons d'associer les avancés récentes sur les structures d'accélération des filtres non linéaire avec les dernières avancées du domaine non local. Nous proposons une nouvelle structure de données, plus générale qu'une simple structure d'accélération, capable d'apprendre les distributions locales d'un grand ensemble d'échantillons en les représentant par des gaussiennes anisotropes. Contrairement aux méthodes qui représentent les distributions d'échantillons par un modèle de mixture de gaussiennes de nombre fini [YSM12], ces gaussiennes sont estimées en fonction de la région de l'espace considéré.

Ainsi, nous introduisons l'arbre de covariance (Covariance Tree - CovTree) qui possède les caractéristiques suivantes :

1. **Apprentissage de grande base de données** : En utilisant une approche similaire aux structures d'accélérations des filtres non linéaires, l'ensemble des échantillons est réparti en sous-ensembles adaptés représentés par des gaussiennes indépendantes du nombre d'échantillons.
2. **Optimisation de la mise à jour des données** : L'ajout d'un échantillon dans notre structure n'entraîne pas la reconstruction complète de la structure. Seules les gaussiennes qui décrivent les distributions de chaque sous-ensemble sont modifiées. L'ajout d'un point dans notre structure permet d'améliorer l'estimation des statistiques des distributions sans changer la quantité de mémoire utilisée.
3. **Estimation des distributions locales à différentes échelles** : Contrairement aux structures d'accélération des filtres non linéaires, notre CovTree a été conçu pour être indépendant des paramètres d'échelles. Ainsi, l'estimation des distributions locales peut être effectuée en un temps raisonnable sans avoir besoin de reconstruire entièrement la structure.
4. **Généralisation de la structure d'apprentissage** : Notre approche est générique et peut être appliquée dans un espace de dimensions arbitraires. Pour être utilisable par un plus grand nombre d'applications, elle est définie en utilisant deux domaines distincts : un domaine spatial (utilisé pour calculer une distance entre les échantillons) et un domaine des valeurs (utilisé pour exprimer les statistiques sur les distributions).

Notre structure d'apprentissage est beaucoup plus générique qu'une simple structure d'accélération de filtres collaboratifs. Comme nous le montrerons dans le chapitre 5, cette dernière peut être utilisée pour des applications 2D et 3D variées.

4.2 Arbre de Covariance Simplifié

Pour mieux appréhender les idées fondamentales de notre approche, nous introduisons dans cette section une version simplifiée de notre CovTree qui sera utilisé pour apprendre les distributions d'échantillons dans un espace fini. Elle nous permettra d'introduire les principes sous-jacents à notre structure de données.

Néanmoins, cette définition reste trop spécifique pour être utilisée en pratique. Par conséquent, nous introduisons en Section 4.3 une définition plus générique de notre CovTree utilisable pour résoudre des problèmes de restauration variés.

4.2.1 Principe

Idée générale

Considérons un ensemble d'échantillons $\mathcal{P} = \{\mathbf{p}_i\}$ où $\mathbf{p}_i \in \mathcal{S} \subset \mathbb{R}^{d_\mathcal{S}}$. Notre CovTree est défini comme une structure de données qui apprend localement la distribution des points de $\mathcal{P}$ à partir d'un sous-échantillonnage adaptatif (cellules) de $\mathcal{S}$. Pour n'importe quelle

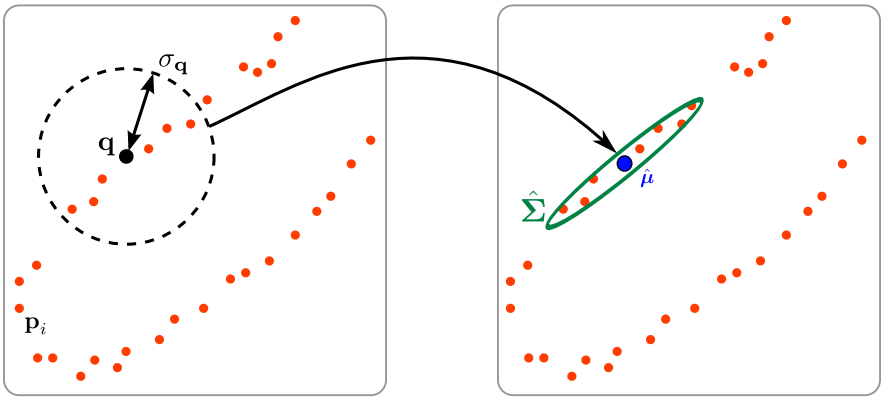


FIGURE 4.1: *Gauche* : Nous illustrons le principe de notre CovTree sur un nuage de points $\mathbf{p}_i$ extrait d'une courbe 2D pour un voisinage centré en $\mathbf{q}$ et de rayon $\sigma_{\mathbf{q}}$. *Droite* : Le but de notre CovTree est d'apprendre la distribution des points $\mathcal{P}$ et de modéliser la répartition des points qui sont compris dans ce voisinage par une gaussienne anisotrope représentée par une moyenne $\hat{\boldsymbol{\mu}}$ et une matrice de covariance $\hat{\boldsymbol{\Sigma}}$.

requête de position $\mathbf{q} \in \mathcal{S}$ et échelle $\sigma_{\mathbf{q}} \in \mathbb{R}$, la structure retourne la gaussienne anisotrope correspondante à la distribution locale des échantillons de $\mathcal{P}$ appris.

En Figure 4.1, nous illustrons les entrées et sorties de notre Cov-Tree dans le cas où $\mathcal{P}$ correspond à un nuage de points 2D issu d'une courbe. Dans ce cas, les $\mathbf{p}_i$ définis dans un espace $\mathcal{S} = \mathbb{R}^2$ représentent les positions des points du nuage. Notre CovTree permet donc d'apprendre la répartition des points du nuage et fournit pour n'importe quel voisinage euclidien, la gaussienne anisotrope correspondante à la distribution des points du nuage compris dans ce voisinage.

Dans le cas où $\mathcal{P}$ correspond à l'ensemble des patchs d'une image bruitée, cette structure de donnée simplifiée peut être utiliser pour estimer le résultat de filtres collaboratifs tels que les Non Local Bayes [LBM13].

Représentation des distributions locales

Pour apprendre les distributions des échantillons de $\mathcal{P}$, nous effectuons une analyse statistique de sous-ensembles $C \subset \mathcal{P}$ d'échantillons locaux. L'idée fondamentale de notre CovTree est de remplacer ces sous-ensembles par leurs statistiques respectives. Dans cette partie, nous fournissons une analyse de pouvoir synthétique de ces dernières et de leur impact sur les performances de notre CovTree.

En considérant que l'ensemble des points de C sont compris dans une boule de rayon négligeable, la distribution peut être représentée correctement par une simple moyenne des points de C. Cette solution est utilisée pour accélérer le calcul des filtres à hautes dimensions [AGDL09] car pour des échelles inférieures à l'échelle de filtrage, il est inutile de conserver de plus fines variations des échantillons. Comme nous le présentons en Figure 4.2, cette solution est équivalente à conserver l'ensemble des échantillons de $\mathcal{P}$ en mémoire pour des échelles de filtrage fines. De plus, comme la moyenne ne fournit aucune information

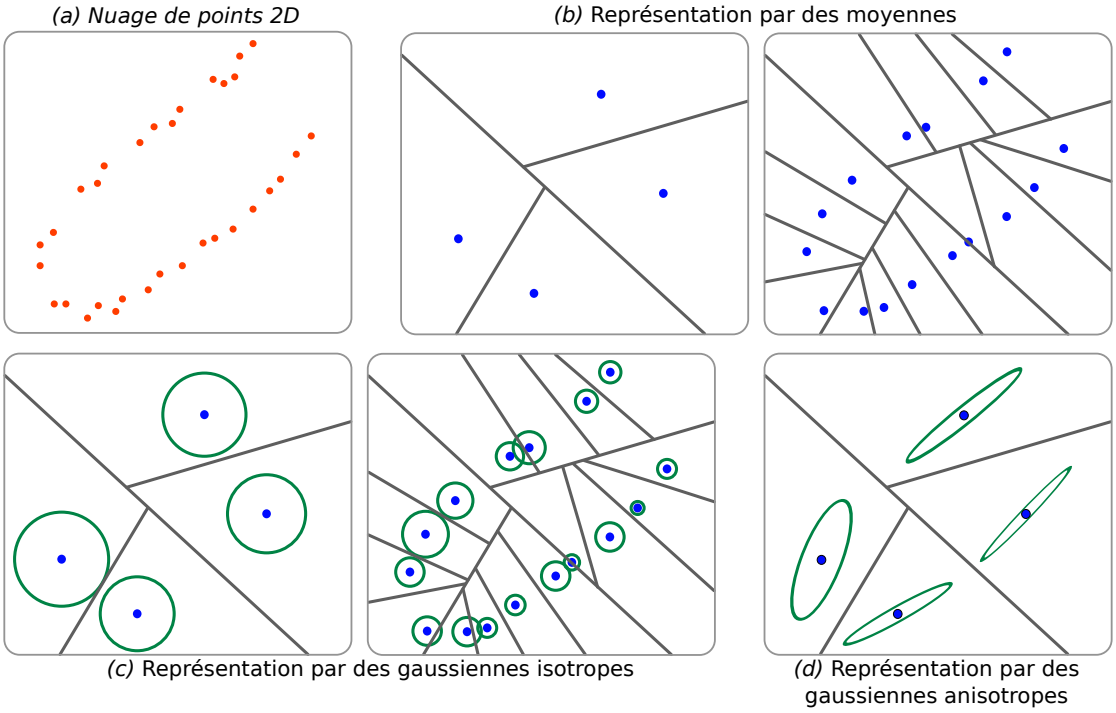


FIGURE 4.2: Différentes représentations statistiques du nuage de points 2D. (a) Nuage de points 2D original. (b) Utiliser une moyenne par cellules pour représenter un nuage de points revient à garder l'ensemble des points du nuage pour des tailles de cellules faibles et ne permet de représenter efficacement les variations du nuage à de grandes échelles. (c) En utilisant des gaussiennes isotropes, la modélisation devient plus fiable à des échelles fines, mais reste malgré tout trop grossière pour représenter la distribution à des échelles élevées. (d) Nous proposons l'utilisation de gaussiennes anisotropes qui permettent de représenter correctement les distributions à de grandes échelles avec un nombre de gaussiennes limitées.

sur les variations des échantillons qu'elle représente, elle ne peut être utilisée correctement à de larges échelles pour représenter la distribution de grands ensembles de données.

Pour de grandes échelles comme la moyenne ne fournit aucune information variationnelle, il devient difficile de représenter correctement la distribution des échantillons d'un grand ensemble de données.

Pour des ensembles d'échantillons de C distribués de manière isotrope dans une boule, l'approche précédente peut être améliorée en associant à la moyenne des points de C une variance. La distribution peut par conséquent être modélisée par une gaussienne isotrope. Désormais, la représentation dépend directement de la répartition des données et non de l'échelle de filtre à appliquer. Comme nous l'illustrons en Figure 4.2, cette contrainte d'isotropie reste difficile à satisfaire pour des échelles élevées. De plus, elle revient à garder en mémoire l'ensemble des échantillons pour des échelles fines.

Les deux approches précédentes ne sont pas suffisantes pour apprendre les distributions d'échantillons. Par conséquent, nous préférons les représenter par l'intermédiaire de gaussiennes anisotropes. Nous associons donc chaque sous-ensemble C à sa moyenne $\boldsymbol{\mu}$ et sa matrice de covariance $\boldsymbol{\Sigma}$. Comme nous le démontrons en Figure 4.2, ce choix permet de

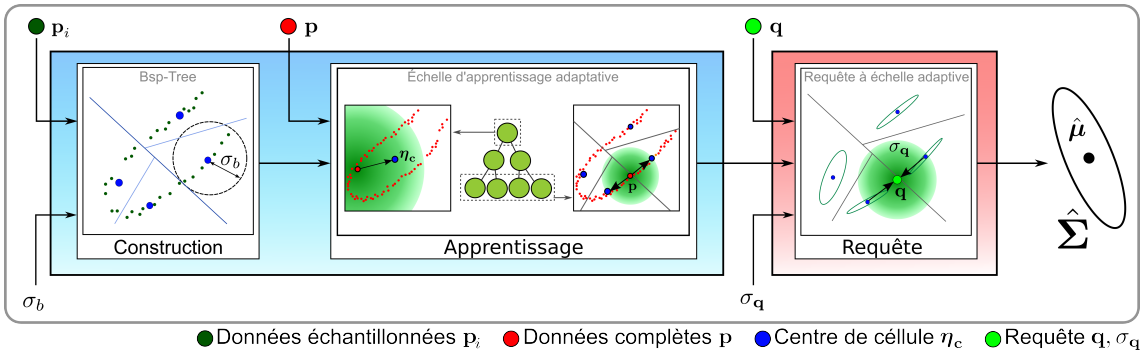


FIGURE 4.3: Notre CovTree est basé sur trois étapes principales. *Construction* : à partir d'un ensemble d'échantillons, nous construisons un arbre binaire qui partitionne l'espace en fonction des positions des échantillons $\{\mathbf{p}_i\}$ pour créer des cellules de taille σ_b. *Apprentissage* : chaque nœud apprend les distributions statistiques locales modélisées par des noyaux anisotropes en propageant l'ensemble des échantillons à travers l'arbre en fonction de leur position $\mathbf{p}_i$ et ajoutant une contribution pondérée des $\mathbf{p}_i$ au noyau de chaque nœud de l'arbre traversé. *Requête* : pour toute requête constituée d'une position $\mathbf{q} \in \mathcal{S}$ et échelle $\sigma_\mathbf{q} \in \mathbb{R}$, notre CovTree modélise la distribution locale des données apprises en $\mathbf{q}$ et à l'échelle $\sigma_\mathbf{q}$ par une gaussienne multivariée définie par sa moyenne $\hat{\boldsymbol{\mu}}$ et sa matrice de covariance $\hat{\boldsymbol{\Sigma}}$.

représenter correctement les distributions des échantillons à des échelles fines et élevées. De plus, l'utilisation d'une gaussienne anisotrope nous permet de diminuer la quantité de mémoire nécessaire, d'augmenter le pouvoir descriptif de notre CovTree et d'être plus indépendant des contraintes d'échantillonnages (complétion des trous/données manquantes).

Chaîne de traitement

Notre approche, résumée en Figure 4.3, peut être divisée essentiellement en trois étapes. Ces dernières peuvent être comparées avec les trois étapes de *splatting, blurring, slicing* utilisé par les travaux sur l'accélération de filtres à hautes dimensions [PD09, ABD10, GO12].

1. **Construction** : Nous effectuons un sous-échantillonnage hiérarchique pyramidal de $\mathcal{S}$ basé sur les échantillons de $\mathcal{P}$ jusqu'à l'obtention de cellules de taille σ_b. Il en résulte un arbre binaire dont chaque nœud (correspondant à un sous-espace de $\mathcal{S}$) est associé à un noyau anisotrope modélisant la distribution statistique des points de $\mathcal{P}$ appartenant à la cellule spatiale correspondante (Section 4.2.2).
2. **Apprentissage** : Nous apprenons les distributions de points par propagation (entraînement) des données à travers l'arbre binaire. Chaque point est classifié en considérant sa position et ajoute une contribution pondérée au noyau anisotrope de chaque nœud de l'arbre qu'il traverse (Section 4.2.3).
3. **Requête** : Pour toute requête définie par une boule de centre $\mathbf{q} \in \mathcal{S}$ et échelle $\sigma_\mathbf{q} \in \mathbb{R}$, notre CovTree fournit la distribution des données apprises correspondant à ce voisinage. Elle est modélisée par une gaussienne multivariée représentée par

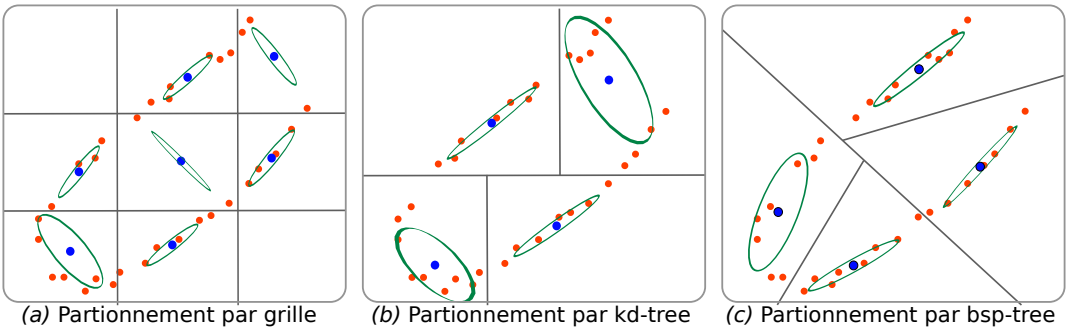


FIGURE 4.4: Nous proposons ici différents partitionnements de l'espace. (a) : Une grille régulière permet de définir des cellules régulières. Ces dernières sont définies indépendamment des données qu'elles contiennent résultant des cellules vides ou mal définies. (b) : Le *kd-tree* définit un partitionnement hiérarchique de l'espace dont les partitions sont faites dans des directions indépendantes des données à apprendre. Il ne crée pas de cellules vides, mais ne tient pas compte de l'anisotropie des données. (c) : Notre *bsp-tree* définit un partitionnement hiérarchique de l'espace qui permet de tenir compte de l'anisotropie des données à apprendre.

une une moyenne $\hat{\boldsymbol{\mu}}$ et d'une matrice de covariance $\hat{\boldsymbol{\Sigma}}$ interpolée en $\mathbf{q}$ à l'échelle $\sigma_{\mathbf{q}}$ (Section 4.2.4).

Il est important de comprendre que σ_b et $\sigma_{\mathbf{q}}$ sont liés à la quantité d'informations que nous voulons récupérer. L'application successive des trois étapes de construction, d'apprentissage et de requête est approximativement équivalente à estimer une matrice de covariance avec un noyau gaussien de taille $\sqrt{2}\sigma_{\mathbf{q}}$.

Pour prévenir tout problème de calcul $\sigma_{\mathbf{q}}$ doit être choisi plus grand que σ_b, suffisamment large pour outrepasser le bruit, mais suffisamment petit pour capturer les structures locales. Généralement, $\sigma_b \approx \sigma_n$ et $\sigma_{\mathbf{q}} \geq \sigma_n$.

4.2.2 Construction

Pavage adapté de l'espace

Durant l'étape de construction, nous définissons un partitionnement de l'espace $\mathcal{S}$ en cellules spatiales. Ces cellules doivent être adaptées à la distribution des échantillons de $\mathcal{P}$ afin de respecter les variations topologiques de $\mathcal{P}$ (et d'obtenir un meilleur apprentissage des données). Nous fournissons dans cette partie une analyse de plusieurs solutions pour partitionner l'espace et de leur impact sur les performances du CovTree.

Une des solutions les plus faciles est de diviser l'espace en un ensemble de cellules régulières par l'intermédiaire d'une grille [PD06, PD09]. Malgré sa simplicité de mise en œuvre, utiliser ce schéma de subdivision pour notre CovTree présente trois problèmes (Figure 4.4). Tout d'abord, la quantité de mémoire utilisée est d'autant plus grande que l'espace $\mathcal{S}$ est de dimension élevée et que la taille des cellules est faible. Ensuite, les cellules sont définies indépendamment du signal à considérer, ainsi, les gaussiennes anisotropes ne sont pas nécessairement représentatives des variations des échantillons. Le problème majeur vient

du fait que ces cellules ne contiennent pas obligatoirement des échantillons et accaparent inutilement de l'espace mémoire.

Pour résoudre ce problème, il est plus judicieux d'utiliser des structures de subdivision hiérarchique telle que le *kd-tree* [AGDL09]. Ce dernier est défini comme un arbre binaire dans lequel chaque nœud est associé à un sous-espace de $\mathcal{S}$. Chaque nœud interne de l'arbre divise l'espace en deux demi-espaces distincts selon un hyperplan normal à la direction de plus grande variation des points. Cette direction est choisie parmi les directions d'un repère orthonormé fixe de l'espace $\mathcal{S}$ définie indépendamment de $\mathcal{P}$. Une fois encore, les cellules spatiales échouent à capturer l'anisotropie des données correctement (Figure 4.4). Les échantillons, regroupés de manière inadéquate, définissent des distributions locales de $\mathcal{P}$ parasites. De plus, le partitionnement ainsi défini est différent pour tout changement d'orientation.

Par conséquent, nous définissons notre CovTree par l'intermédiaire d'une structure de partitionnement binaire de l'espace, le *bsp-tree* [FKN80], qui divise l'espace en deux sous-espaces distincts par l'intermédiaire d'un hyperplan orienté selon la direction de plus grande variation des données d'entrées. Les cellules ainsi obtenues ne sont pas nécessairement parallélépipédiques et permettent de tenir compte de l'anisotropie des données à apprendre (Figure 4.4). Les gaussiennes anisotropes résultantes permettent de mieux représenter le signal.

Algorithme

Nous divisons l'espace par l'intermédiaire d'un *bsp-tree* dont chaque nœud est associé à un sous-espace $C \subset \mathcal{S}$, un sous-ensemble d'échantillons $\mathbf{p}_j \in \mathcal{P} \cap C$ et à un rayon de cellule η_r (utilisé pour apprendre les distributions à différentes échelles). Chaque nœud interne de l'arbre divise l'espace en deux demi-espaces distincts par l'intermédiaire d'un plan de coupe $\{\boldsymbol{\eta_c}, \boldsymbol{\eta_d}\}$ dont la normale $\boldsymbol{\eta_d}$ est définie comme le vecteur propre normalisé associé à la plus grande valeur propre de la covariance des $\{\mathbf{p}_j\}$ et $\boldsymbol{\eta_c}$ par la moyenne des $\{\mathbf{p}_j\}$. En pratique, $\boldsymbol{\eta_d}$ est estimé avec une complexité linéaire par rapport à la dimension de $\mathcal{S}$ et au nombre d'échantillons en utilisant la méthode de la puissance itérée. Le rayon de cellule est défini par :

$$\eta_r \quad = \quad \max_{\mathbf{p}_j} \|\mathbf{p}_j - \boldsymbol{\eta_c}\| \tag{4.1}$$

L'ensemble des points $\{\mathbf{p}_j\}$ est partagé en deux sous-ensembles par rapport à leur distance signée au plan $(\mathbf{p}_j - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d}$. Les deux sous-branches du nœud η sont ensuite construites en se basant sur ces deux sous-ensembles.

En initialisant le nœud racine avec l'ensemble des données d'entrée $\{\mathbf{p}_i\}$, nous appliquons récursivement ce schéma de subdivision tant que le rayon de cellule $\eta_r > \sigma_b$. Par conséquent, l'utilisateur peut choisir la précision d'apprentissage de notre CovTree (ainsi que la quantité de mémoire qu'il va utiliser) en modifiant la valeur du paramètre σ_b. L'Algorithme 1 présente le pseudo-code de cette étape de construction.

Algorithm 1: Construction de l'arbre

Fonction *ConstruireNœud($C \subset \mathcal{P}, \sigma_b \in \mathbb{R}$)*

Entrées: C un sous-ensemble de $\mathcal{P}$, σ_b l'échelle d'arrêt
Sorties: η le nœud créé

Allouer un nouveau nœud η

// Estimation des paramètres de la cellule
$\boldsymbol{\eta_d} \leftarrow$ le vecteur associé à la plus grande valeur propre de $cov(C)$
$\boldsymbol{\eta_c} \leftarrow \underset{\mathbf{p}_i \in C_k}{\text{mean}} (\mathbf{p}_i)$
$\eta_r \leftarrow \underset{\mathbf{p}_i \in C_k}{\max} \|\mathbf{p}_i - \boldsymbol{\eta_c}\|$

si $\eta_r \leq \sigma_b$ **alors**
Marquer η comme feuille de l'arbre
sinon
// Répartition des points dans les deux sous-espaces
$C_{gauche} \leftarrow \{\mathbf{p}_i \in C_k, (\mathbf{p}_i - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \leq 0\}$
$C_{droit} \leftarrow \{\mathbf{p}_i \in C_k, (\mathbf{p}_i - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \geq 0\}$
// Création des deux sous-arbres
$\eta_{gauche} \leftarrow$ *ConstruireNœud*(C_{gauche}, σ_b)
$\eta_{droit} \leftarrow$ *ConstruireNœud*(C_{droit}, σ_b)
fin
retourner η
fin

4.2.3 Apprentissage

Algorithme

Une fois la structure de l'arbre initialisée, nous pouvons calculer les statistiques de chaque cellule en propageant l'ensemble des données d'apprentissage $\{\mathbf{p}_i\}$ à travers l'arbre. En partant de la racine, les points traversent l'arbre jusqu'à atteindre une feuille. Le parcours est effectué en considérant les positions des points $\mathbf{p}_i$. Les gaussiennes anisotropes associées à chacun des nœuds η traversés sont enrichies des valeurs $\mathbf{p}_i$ pondérées par un poids $w_i = \phi\left(\frac{\|\mathbf{p}_i - \boldsymbol{\eta_c}\|}{\eta_r}\right)$. Ce poids est défini grâce à une approximation compacte, linéaire par morceaux du noyau gaussien ϕ centré en $\boldsymbol{\eta_c}$ et de variance η_r :

$$\phi(x) = \begin{cases} 4 - 0.75x^2(2-x) & \text{si} \quad 0 < |x| \leq 1 \\ 0.25(2-x)^3 & \text{si} \quad 1 < |x| \leq 2 \\ 0 & \text{sinon} \end{cases} \tag{4.2}$$

Pour simplifier les deux étapes d'apprentissage et de requête, la moyenne et la covariance des échantillons ne sont pas directement conservées en chaque nœud. A la place, nous

Algorithm 2: Apprentissage des variations locales

Fonction *ApprendreVariations*$(\mathbf{p}, \eta)$

Entrées: $\mathbf{p}$ un point à ajouter dans la structure, η un nœud de l'arbre

// Calcul des poids w par approximation d'une gaussienne de variance η_r

$w \leftarrow \phi\left(\frac{\|\mathbf{p}_i - \boldsymbol{\eta_c}\|}{\eta_r}\right)$

// Mise à jour des statistiques de chaque nœud

$w_\eta := w_\eta + w$

$w_{2\eta} := w_{2\eta} + w.w$

$\boldsymbol{\mu}_\eta := \boldsymbol{\mu}_\eta + w.\mathbf{p}$

$\boldsymbol{\Sigma}_\eta := \boldsymbol{\Sigma}_\eta + w.\mathbf{p}^t\mathbf{p}$

// Apprentissage des statistiques à une échelle inférieure

si *η n'est pas une feuille de l'arbre* **alors**

  si $(\mathbf{p} - \boldsymbol{\eta_c})^t\boldsymbol{\eta_d} \leq 0$ **alors**

    ApprendreVariations$(\mathbf{p}, \eta_{gauche})$

  sinon

    ApprendreVariations$(\mathbf{p}, \eta_{droit})$

  fin

fin

fin

stockons une somme partielle $\boldsymbol{\mu}_\eta$, une somme croisée partielle des positions $\boldsymbol{\Sigma}_\eta$ et deux constantes de normalisation w_η et $w_{2\eta}$:

$$\begin{aligned} w_\eta &:= w_\eta + w_i \\ w_{2\eta} &:= w_{2\eta} + w_i.w_i \\ \boldsymbol{\mu}_\eta &:= \boldsymbol{\mu}_\eta + w_i.\mathbf{p}_i \\ \boldsymbol{\Sigma}_\eta &:= \boldsymbol{\Sigma}_\eta + w_i\mathbf{p}_i^t\mathbf{p}_i \end{aligned} \tag{4.3}$$

La moyenne et la covariance représentatives des variations du nœud peuvent être retrouvées à partir de ces quatres variable. L'Algorithme 2 présente le pseudo-code de l'étape d'apprentissage.

Complétion des données d'apprentissage

De manière générale, le même ensemble de points $\mathcal{P}$ est utilisé lors de l'étape de construction et d'apprentissage. Néanmoins, il est possible d'utiliser un sous-ensemble représentatif $\mathcal{P}' \subset \mathcal{P}$ différent pour définir un partitionnement de l'espace $\mathcal{S}$ lors de l'étape de construction. Cette approche possède deux avantages : *(i)* le partitionnement de l'espace peut être défini plus rapidement (en particulier dans le cas d'une large base de données), *(ii)* il n'est pas nécessaire de connaître l'ensemble des données pour commencer l'apprentissage des données (par ex. traitement vidéo).

De plus, comme nous utilisons une analyse statistique partielle (nous stockons en mémoire uniquement des sommes partielles et non directement les statistiques), il devient possible

d'ajouter au fur et à mesure des échantillons par l'intermédiaire de l'Équation 4.3. Chaque point ajouté durant l'étape d'apprentissage augmente la précision des distributions apprises par chaque cellule. L'avantage d'une telle approche est que l'espace mémoire utilisé reste constant quel que soit le nombre d'échantillons rajoutés. En effet, nous ne gardons pas en mémoire l'ensemble des échantillons, mais uniquement les quatre variables w_η, $w_{2\eta}$, $\boldsymbol{\mu}_\eta$ et $\boldsymbol{\Sigma}_\eta$.

Cette propriété peut être utilisée pour apprendre la distribution d'échantillons issus d'une grande base de données. Notre structure représente localement la répartition des échantillons par l'intermédiaire de gaussiennes multivariées utilisant peu d'espace mémoire. Cette propriété sera utilisée en Section 5.2 pour approcher l'*a priori* sous-jacent aux patchs des images naturelles.

4.2.4 Requête

Accélération de la requête

Pour tout voisinage de centre $\mathbf{q} \in \mathcal{S}$ et de rayon $\sigma_\mathbf{q} \in \mathbb{R}$, la gaussienne anisotrope est estimée par une combinaison des données apprises sur chaque nœud de l'arbre avec des poids décroissants par rapport à l'éloignement au centre de la requête $\mathbf{q}$. Pour accélérer le calcul de cette estimation, il suffit de sélectionner uniquement les nœuds qui possèdent les poids les plus élevés.

Certaines méthodes [AGDL09] proposent par exemple un schéma de requête à importance des échantillons. L'idée principale est de distribuer un nombre d'échantillons à travers l'arbre jusqu'à atteindre les feuilles en favorisant les nœuds de poids les plus élevés. Cette démarche peut être perçue comme une méthode de Monte-Carlo qui propage les points dans l'arbre de manière probabiliste de sorte à atteindre les nœuds les plus proches. Même si une telle démarche fonctionne relativement bien pour des espaces de basses dimensions (car peu d'échantillons sont nécessaires pour estimer correctement les distributions), pour des espaces de dimensions élevées il devient nécessaire de faire un choix entre précision de l'estimation et temps de calculs. En effet le nombre d'échantillons nécessaires pour avoir une précision de l'estimation suffisante est plus élevé.

Par conséquent, ce genre d'approche ne peut être utilisé dans notre CovTree. Pour résoudre ce problème, nous proposons d'utiliser l'apprentissage des distributions à différents niveaux de l'arbre et le rayon η_r associé à chaque nœud. Cette démarche nous permet de limiter le parcours en profondeur de l'arbre, le nombre de nœuds à explorer et d'être plus indépendant de l'échelle de la requête $\sigma_\mathbf{q}$.

Algorithme

Une fois notre CovTree complètement construit et appris, nous utilisons un schéma de requête pour estimer la gaussienne anisotrope décrivant la distribution des données apprises pour n'importe quel voisinage centré en $\mathbf{q} \in \mathcal{S}$ d'échelle $\sigma_\mathbf{q} \in \mathbb{R}$.

Tout d'abord, nous collectons l'ensemble des nœuds η qui intersecte la boule $[\mathbf{q}, \sigma_\mathbf{q}]$. Pour ce faire, chaque requête traverse l'arbre jusqu'à atteindre une feuille ou un nœud vérifiant

Algorithm 3: Interroger l'arbre

Fonction *InterrogerArbre*($\mathbf{q}, \sigma_\mathbf{q}$)

Entrées: $\mathbf{q} \in \mathcal{S}$ un point requête et $\sigma_\mathbf{q}$ une échelle de requête
Sorties: la gaussienne anisotrope $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$

// Requête de l'arbre en commençant par la racine η_0
$\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} \leftarrow$ *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta_0$)

// Estimation des paramètres de la gaussienne multivariée
$\hat{\boldsymbol{\mu}} \leftarrow \frac{1}{\hat{w}} \boldsymbol{\mu}$
$\hat{\boldsymbol{\Sigma}} \leftarrow \frac{\hat{w}^2 - \hat{w}_2}{\hat{w}} (\boldsymbol{\Sigma} - \hat{w} \hat{\boldsymbol{\mu}}^t \hat{\boldsymbol{\mu}})$

retourner $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$

fin

Fonction *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta$)

Entrées: $\mathbf{q} \in \mathcal{S}$ un point requête, $\sigma_\mathbf{q}$ une échelle de requête et η un nœud de l'arbre
Sorties: $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\hat{w}$ et $\hat{w}_2$

si $\eta_r \leq \sigma_\mathbf{q}$ ***où*** η *est une feuille de l'arbre* **alors**
// Calcul des poids w *par approximation d'une gaussienne de variance* $\sigma_\mathbf{q}$
$w \leftarrow \phi\left(\frac{\|\mathbf{q} - \boldsymbol{\eta_c}\|}{\sigma_\mathbf{q}}\right)$
// Ajout des variations du nœud η
$\hat{w} := \hat{w} + w$
$\hat{w}_2 := \hat{w}_2 + w^2$
$\boldsymbol{\mu} := \boldsymbol{\mu} + w \boldsymbol{\mu}_\eta$
$\boldsymbol{\Sigma} := \boldsymbol{\Sigma} + w \boldsymbol{\Sigma}_\eta$
sinon
// Ajout des variations du nœud η *en résultat*
si $(\mathbf{q} - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \leq \sigma_\mathbf{q}$ **alors**
$\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} := \{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} +$ *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta_{gauche}$)
sinon
$\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} := \{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} +$ *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta_{droit}$)
fin
fin
retourner $\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$
fin

$\eta_r > \sigma_\mathbf{q}$. Lors de la propagation de la requête, nous considérons que chaque cellule est élargie de $\sigma_\mathbf{q}$. Ainsi pour des distances au plan de coupe $|(\mathbf{q} - \eta_\mathbf{c})^t \eta_\mathbf{d}| < \sigma_\mathbf{q}$, les deux sous-arbres gauche et droit sont explorés.

Au final, la distribution en $\mathbf{q}$ est estimée par une combinaison des distributions des nœuds collectés $\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}$ pondérées par des poids w_i définis à partir de l'approximation du noyau gaussien $\phi_{\sigma_\mathbf{q}}$ centré en $\mathbf{q}$ et de variance $\sigma_\mathbf{q}$:

$$\begin{aligned} \hat{w} &= \sum_i w_i w_{\eta_i} \\ \hat{w_2} &= \hat{w}^2 - \sum_i w_i w_{2\eta_i} \\ \hat{\boldsymbol{\mu}} &= \frac{1}{\hat{w}} \sum_i w_i \boldsymbol{\mu}_{\eta_i} \\ \hat{\boldsymbol{\Sigma}} &= \frac{\hat{w}}{\hat{w_2}} \sum_i w_i \boldsymbol{\Sigma}_{\eta_i} - \hat{w} \hat{\boldsymbol{\mu}}^t \hat{\boldsymbol{\mu}} \end{aligned} \tag{4.4}$$

Si $\sigma_\mathbf{q}$ est choisi suffisamment large, la gaussienne anisotrope obtenue décrit la distribution de l'ensemble des données apprises. L'Algorithme 3 présente le pseudo-code de l'étape de requête.

4.3 Généralisation

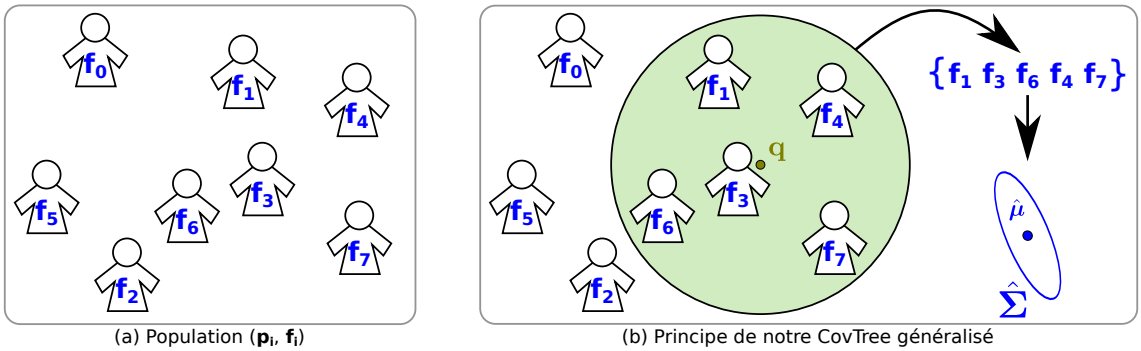
Dans la partie précédente, nous avons présenté une version simplifiée de notre CovTree qui permet d'apprendre la distribution d'un ensemble de points d'un espace de dimension arbitraire. Pour une majorité d'applications de notre CovTree, cette définition reste trop limitée. Nous proposons dans cette section de la généraliser en utilisant deux domaines distincts : le domaine spatial $\mathcal{S}$ qui définit les distances entre les différents échantillons de l'espace et le domaine des attributs $\mathcal{R}$ qui associe chaque échantillon à une valeur.

4.3.1 Notations

Soit un domaine spatial $\mathcal{S} \in \mathbb{R}^{d_\mathcal{S}}$ et un nuage de points $\mathcal{P} = \{\mathbf{p}_i\}_{i \in [\![1,N]\!]}$ de N échantillons. Nous allons considérer une correspondance $f : \mathcal{S} \to \mathcal{R}$ associant à chaque échantillon de position spatiale $\mathbf{p}_i \in \mathcal{S}$, une valeur $\mathbf{f}_i$ du domaine des attributs $\mathcal{R} \in \mathbb{R}^{d_\mathcal{R}}$.

Pour une meilleure compréhension, il peut être utile de considérer f comme une fonction. Néanmoins, notre structure ne requiert pas que la correspondance f soit connue explicitement, bien définie ou unique pour n'importe quel $\mathbf{p} \in \mathcal{S}$. L'association entre les $\mathbf{p}$ et les $\mathbf{f}$ est apprise à l'aide de notre approche à partir d'une base de données de paires d'échantillons $(\mathbf{p}_i, \mathbf{f}_i)$.

Une telle représentation peut par exemple être utilisée pour représenter des images RGB associant à chaque pixel de l'image $\mathbf{p}_i = (x_i, y_i)^T$ une valeur couleur $\mathbf{f}_i = (r_i, g_i, b_i)^T$, des nuages de points 3D définis par leurs positions spatiales et normales $\mathbf{p}_i = \mathbf{f}_i = (x_i, y_i, z_i, n_i^x, n_i^y, n_i^z)^T$, etc.



(a) Population ($\mathbf{p}_i$, $\mathbf{f}_i$) (b) Principe de notre CovTree généralisé

FIGURE 4.5: *Gauche* : Considérons une foule dont chaque individu possède une position spatiale $\mathbf{p}_i$ définie dans le domaine spatial $\mathcal{S}$ et un attribut $\mathbf{f}_i$ défini dans le domaine des attributs $\mathcal{R}$. À l'aide de notre CovTree généralisé, nous apprenons la répartition des $\{\mathbf{f}_i\}$ par rapport à leur position spatiale $\{\mathbf{p}_i\}$ et à différentes échelles spatiales. *Droite* : Ainsi en considérant un sous-ensemble spatial de centre $\mathbf{q} \in \mathcal{S}$ de taille $\sigma_\mathbf{q} \in \mathbb{R}$ (représenté en Vert), notre structure de données permet d'estimer la distribution des attributs des individus qui sont compris dans ce voisinage en la modélisant par une gaussienne multivariée définie dans l'espace des attributs $\mathcal{R}$. Cette dernière est définie par une moyenne $\hat{\boldsymbol{\mu}} \in \mathcal{R}$ et une matrice de covariance $\hat{\boldsymbol{\Sigma}} \in \mathcal{R} \times \mathcal{R}$.

4.3.2 CovTree généralisé

Principe général

Tout comme sa version simplifiée, la version généralisée de notre CovTree permet d'apprendre des distributions pour différentes échelles et de fournir pour tout voisinage requête une gaussienne anisotrope modélisant la distribution des données locales apprises comprise dans ce voisinage. Cette structure est définie par l'intermédiaire d'un domaine spatial $\mathcal{S}$ (utilisé pour définir le voisinage des échantillons) et d'un domaine des attributs $\mathcal{R}$ (utilisé pour définir les distributions). Ainsi, tout regroupement des échantillons et requête de voisinage sont effectués dans $\mathcal{S}$ alors que les distributions apprises et résultantes sont définies dans $\mathcal{R}$.

Cette différence étant difficile à appréhender, nous proposons une illustration du principe du CovTree généralisé ainsi que la distinction entre les deux domaines $\mathcal{S}$ et $\mathcal{R}$ en Figure 4.5 considérons une population d'individus possédant une position spatiale $\mathbf{p}_i \in \mathcal{S}$ (correspondant à leur position dans l'espace) et un attribut $\mathbf{f}_i \in \mathcal{R}$ associé (correspondant aux valeurs inscrites dans chaque individu). Ici, nous ne posons aucune condition sur le lien qui existe entre $\mathbf{p}_i$ et $\mathbf{f}_i$. Par l'intermédiaire de notre CovTree généralisé, nous souhaitons analyser les variations des attributs associés à un sous-ensemble d'individus de cette foule. Cette distribution d'attributs sera modélisée par l'intermédiaire d'une gaussienne anisotrope.

Applications	$\mathcal{S}$	$\mathcal{R}$	Méthode de reconstruction
Gaussien 1D	Position des pixels (x_i, y_i)	Niveau de gris (g_i)	Moyenne
Gaussian 3D	Position des pixels (x_i, y_i)	Valeur couleur (r_i, g_i, b_i)	Moyenne
Bilateral	Position des pixels + Valeur couleur $(x_i, y_i, r_i, g_i, b_i)$	Valeur couleur (r_i, g_i, b_i)	Moyenne
NLM	patches autour du pixel P_{x_i,y_i}	Valeur couleur (r_i, g_i, b_i)	EAP
NLB	Patches P_{x_i,y_i}	Patches P_{x_i,y_i}	MAP
Completion de trous pyramidal	Patches basse résolutions	Patches hautes résolutions	MAP
PSS	Point 3D $\begin{pmatrix} x_i, y_i, z_i \\ n_i^x, n_i^y, n_i^z \end{pmatrix}$	Point 3D $\begin{pmatrix} x_i, y_i, z_i \\ n_i^x, n_i^y, n_i^z \end{pmatrix}$	Moyenne
NLPSS	Patches 3D P_{x_i,y_i,z_i}	Champs scalaire v_i	EAP
NLB-PSS	Patches 3D P_{x_i,y_i,z_i}	Patches 3D + champs scalaire (P_{x_i,y_i,z_i}, v_i)	MAP

TABLE 4.1: Notre CovTree est très flexible : Cette table montre comment définir les domaines spatiaux $\mathcal{S}$ et des attributs $\mathcal{R}$ ainsi que la méthode de reconstruction à utiliser.

Les regroupements des individus (sous-ensembles) sont définis dans le domaine spatial $\mathcal{S}$, c'est-à-dire en considérant uniquement les positions spatiales $\mathbf{p}_i$ de chacun de ces individus. Ensuite, les distributions de ces différents regroupements sont estimées en considérant uniquement les attributs $\mathbf{f}_i$ associés à chaque individu de ce sous-ensemble. Par conséquent, la gaussienne anisotrope résultante correspond uniquement aux variations dans l'espace des attributs $\mathcal{R}$ de ce sous-ensemble d'individus (et non de leur position spatiale $\mathbf{p}_i$).

Notre CovTree généralisé permet donc d'apprendre de grandes bases de données de couples d'échantillons exemples $(\mathbf{p}_i, \mathbf{f}_i)$ et fournit pour tout voisinage spatial $\mathbf{q} \in \mathcal{S}$ de rayon $\sigma_{\mathbf{q}} \in \mathbb{R}$ la distribution des attributs associée à ces échantillons représentés par une gaussienne anisotrope définie dans $\mathcal{R}$. Il est possible de retrouver la version simplifiée de notre CovTree en confondant les deux domaines spatiaux et des attributs : $\mathcal{S} = \mathcal{R}$.

Problèmes de restauration simple

Avant de présenter les modifications apportées par la version généralisée de notre CovTree, nous illustrons ici quelques problèmes de restauration simple qui peuvent être traités par notre structure :

1. **Convolution gaussienne** : Considérons une image à niveaux de gris bruitée $\tilde{u}(x, y) = u(x, y) + n(x, y)$. Dans ce cas $d_{\mathcal{S}} = 2$ (correspondant aux deux coordonnées spatiales) et $d_{\mathcal{R}} = 1$ (correspondant au niveau de gris). En supposant l'image homogène, un *a priori* de la valeur $\tilde{u}(x, y)$ peut être estimé à partir des valeurs $\tilde{u}(x', y')$ d'un ensemble de voisins $\mathbf{p} = (x', y')$ proche du point requête $\mathbf{q} = (x, y)$. Dans ce cas, la convolution gaussienne peut être définie comme Estimation A Posteriori (EAP) de cet *a priori*.
2. **Filtre Bilateral** : Considérons désormais une image bruitée couleur $\tilde{u}(x, y) = u(x, y) + n(x, y)$. En définissant $d_{\mathcal{S}} = 5$ (correspondant aux deux coordonnées spatiales (x, y) plus trois canaux couleurs (r, g, b)) et $d_{\mathcal{R}} = 3$ (les trois canaux couleurs (r, g, b)). Cette fois l'estimation de la valeur débruitée obtenue est plus fine, en effet, les points (x, y) proches d'un contour pourront être rejetés grâce à l'information de couleur qui a été ajoutée au domaine spatial $\mathcal{S}$.
3. **Filtre Non-Local** : Pour des images hautement texturées, la supposition d'image lisse des deux précédentes illustrations n'est plus valable. Dans ce cas, les filtres non locaux supposent que l'ensemble des patchs d'une image texturée appartiennent à une variété lisse de basse dimension. En prenant $\mathbf{p}_i = \mathbf{f}_i$ comme l'ensemble de tout les patchs de l'image, un patch requête $\mathbf{q}$ peut être débruité par les NLB [LBM13] en calculant une gaussienne multivariée à partir d'un ensemble de patchs voisins $\{\mathbf{p}_j, \mathbf{p}_j \in N(\mathbf{q})\}$ proches de la requête $\mathbf{q}$. Cette distribution gaussienne est utilisée pour estimer le patch débruité via une Maximisation A Posteriori (MAP) Bayesienne.

En complément des exemples précédents, nous présentons en Table 4.1 d'autres problèmes de restauration qui peuvent être modélisés par cette structure.

4.3.3 Algorithme généralisé

Les principes sous-jacents et les différents explications proposées pour définir la version simplifiée de notre CovTree 4.2 restent toujours valables dans sa version généralisée. Ainsi,

Algorithm 4: Construction de l'arbre généralisé

Fonction *ConstruireNœud*($C \subset \mathbf{p}_i, \sigma_b \in \mathbb{R}$)

Entrées: C un sous-ensemble de $\mathbf{p}_i$, σ_b l'échelle d'arrêt
Sorties: η le nœud créé

allouer un nouveau nœud η

// Estimation des paramètres de la cellule
$\boldsymbol{\eta_d} \leftarrow$ le vecteur associé à la plus grande valeur propre de $cov(C)$
$\boldsymbol{\eta_c} \leftarrow \underset{\mathbf{p}_i \in C_k}{\text{mean}} (\mathbf{p}_i)$
$\eta_r \leftarrow \underset{\mathbf{p}_i \in C_k}{\max} \|\mathbf{p}_i - \boldsymbol{\eta_c}\|$

si $\eta_r \leq \sigma_b$ **alors**
Marquer η comme feuille de l'arbre
sinon
// Répartition des points dans les deux sous-espaces
$C_{gauche} \leftarrow \{\mathbf{p}_i \in C_k, (\mathbf{p}_i - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \leq 0\}$
$C_{droit} \leftarrow \{\mathbf{p}_i \in C_k, (\mathbf{p}_i - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \geq 0\}$
// Création des deux sous-arbres
$\eta_{gauche} \leftarrow$ *ConstruireNœud*(C_{gauche}, σ_b)
$\eta_{droit} \leftarrow$ *ConstruireNœud*(C_{droit}, σ_b)
fin
retourner η
fin

nous présenterons uniquement dans cette section les différences entre les opérateurs simplifiés et les opérateurs généralisés. Pour que le lecteur appréhende bien ces différences, nous fournissons le pseudo-code généralisé de chaque étape dans leur intégralité.

Construction

Durant l'étape de construction, nous définissons un partitionnement de l'espace spatial $\mathcal{S}$. Seules les positions spatiales $\mathbf{p}_i$ des couples d'échantillons interviennent ici pour définir des cellules représentatives. En effet, comme nous l'avons illustré en Figure 4.5, notre structure de données fournit une analyse statistique des variations des attributs en fonction de regroupements spatiaux (et non en fonction des attributs). Par conséquent, le critère d'arrêt $\sigma_b \in \mathbb{R}$ correspond désormais à une échelle spatiale. L'Algorithme 4 fournit le pseudo-code de l'étape de construction généralisée.

Apprentissage

Désormais, l'apprentissage des distributions des échantillons est effectué sur le domaine des attributs. Comme l'arbre est défini en considérant le domaine spatial $\mathcal{S}$, chaque couple d'échantillons $(\mathbf{p}_i, \mathbf{f}_i)$ explore l'arbre en fonction de sa position spatiale $\mathbf{p}_i$. Chaque échan-

Algorithm 5: Apprentissage des variations locales généralisé

Fonction *ApprendreVariations*$(\mathbf{p}, \mathbf{f}, \eta)$

Entrées: $\mathbf{p}$ un point à ajouter dans la structure, η un nœud de l'arbre

// Calcul des poids w par approximation d'une gaussienne de variance η_r

$w \leftarrow \phi\left(\frac{\|\mathbf{p}_i - \boldsymbol{\eta_c}\|}{\eta_r}\right)$

// Mise à jour des statistiques de chaque nœud

$w_\eta := w_\eta + w$

$w_{2\eta} := w_{2\eta} + w.w$

$\boldsymbol{\mu}_\eta := \boldsymbol{\mu}_\eta + w.\mathbf{f}$

$\boldsymbol{\Sigma}_\eta := \boldsymbol{\Sigma}_\eta + w.\mathbf{f}^t\mathbf{f}$

// Apprentissage des statistiques à une échelle inférieure

si η *n'est pas une feuille de l'arbre* **alors**

  si $(\mathbf{p} - \boldsymbol{\eta_c})^t\boldsymbol{\eta_d} \leq 0$ **alors**

    ApprendreVariations$(\mathbf{p}, \mathbf{f}, \eta_{gauche})$

  sinon

    ApprendreVariations$(\mathbf{p}, \mathbf{f}, \eta_{droit})$

  fin

fin

fin

tillon ajoute une contribution partielle pondérée de ces attributs $\mathbf{f}_i$. Le poids est défini par l'approximation de la gaussienne ϕ et dépend de la distance spatiale entre $\mathbf{p}_i$ et le centre spatial de la cellule $\boldsymbol{\eta_c}$.

Lors de cette étape, chaque nœud de l'arbre est associé à une gaussienne anisotrope définie dans le domaine des attributs $\mathcal{R}$, exprimant la distribution des attributs $\mathbf{f}_i$ des échantillons qu'il contient. L'Algorithme 5 présente le pseudo-code de l'opérateur d'apprentissage.

Requête

Pour toute requête définie par une position spatiale $\mathbf{q} \in \mathcal{S}$ et échelle spatiale $\sigma_\mathbf{q} \in \mathbb{R}$, notre CovTree fournit la distribution des attributs de l'ensemble des échantillons appris appartenant à ce voisinage spatial par l'intermédiaire d'une gaussienne multivariée définie dans $\mathcal{R}$. Par conséquent la moyenne $\hat{\boldsymbol{\mu}}$ et la matrice de covariance $\hat{\boldsymbol{\Sigma}}$ estimée par notre CovTree seront respectivement de dimensions $d_\mathcal{R}$ et $d_\mathcal{R} \times d_\mathcal{R}$. L'Algorithme 6 présente le pseudo-code de l'étape de requête généralisée.

4.4 Analyse et temps de calcul

Dans cette section, nous analysons la complexité et les temps de calcul de chaque étape de notre CovTree. Cette analyse est effectuée indépendamment des applications possibles

Algorithm 6: Interroger l'arbre

Fonction *InterrogerArbre*($\mathbf{q}, \sigma_\mathbf{q}$)

Entrées: $\mathbf{q} \in \mathcal{R}$ un point requête et $\sigma_\mathbf{q}$ une échelle de requête
Sorties: la gaussienne anisotrope $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$

// Requête de l'arbre en commençant par la racine η_0
$\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} \leftarrow$ *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta_0$)

// Estimation des paramètres de la gaussienne multivariée
$\hat{\boldsymbol{\mu}} \leftarrow \frac{1}{\hat{w}}\boldsymbol{\mu}$
$\hat{\boldsymbol{\Sigma}} \leftarrow \frac{\hat{w}^2 - \hat{w}_2}{\hat{w}}(\boldsymbol{\Sigma} - \hat{w}\hat{\boldsymbol{\mu}}^t\hat{\boldsymbol{\mu}})$

retourner $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$

fin

Fonction *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta$)

Entrées: $\mathbf{q} \in \mathcal{S}$ un point requête, $\sigma_\mathbf{q}$ une échelle de requête et η un nœud de l'arbre
Sorties: $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\hat{w}$ et $\hat{w}_2$

si $\eta_r \leq \sigma_\mathbf{q}$ ***où*** η *est une feuille de l'arbre* **alors**
// Calcul des poids w *par approximation d'une gaussienne de variance* $\sigma_\mathbf{q}$
$w \leftarrow \phi\left(\frac{\|\mathbf{q} - \boldsymbol{\eta_c}\|}{\sigma_\mathbf{q}}\right)$
// Ajout des variations du nœud η
$\hat{w} := \hat{w} + w$
$\hat{w}_2 := \hat{w}_2 + w^2$
$\boldsymbol{\mu} := \boldsymbol{\mu} + w\boldsymbol{\mu}_\eta$
$\boldsymbol{\Sigma} := \boldsymbol{\Sigma} + w\boldsymbol{\Sigma}_\eta$
sinon
// Ajout des variations du nœud η *en résultat*
si $(\mathbf{q} - \boldsymbol{\eta_c})^t\boldsymbol{\eta_d} \leq \sigma_\mathbf{q}$ **alors**
$\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} := \{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} +$ *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta_{gauche}$)
sinon
$\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} := \{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\} +$ *CalculerVariations*($\mathbf{q}, \sigma_\mathbf{q}, \eta_{droit}$)
fin
fin
retourner $\{\hat{w}, \hat{w}_2, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$

fin

de notre structure de données. Par conséquent, seules les performances temps et mémoire pour estimer des gaussiennes anisotropes seront prises en compte ici.

Pour une analyse plus détaillée concernant les performances qualitatives de notre CovTree, nous renvoyons le lecteur en Section 5.1.1 où une implémentation des filtrages collaboratifs réalisés par l'intermédiaire de notre structure est comparée aux méthodes de l'état de l'art.

4.4.1 Complexité

Notre CovTree est basé sur les trois principaux opérateurs de *construction*, *apprentissage* et *requête*. Nous rappelons que $\mathcal{S}$ est un domaine spatial de dimension $d_{\mathcal{S}}$ et $\mathcal{R}$ un domaine des valeurs de dimension $d_{\mathcal{R}}$.

Nous proposons d'analyser la complexité temporelle de ces trois principales étapes :

1. **Construction** : Supposons que nous utilisons N_b points pour définir le partitionnement de l'espace. Durant cette étape de construction, chacun des N_b points apparaît une seule fois dans chaque nœud de l'arbre. À chaque division est effectuée une analyse en composantes principales pour déterminer la direction de plus grande variation. Cette dernière est accélérée par la méthode des puissances qui estime le vecteur propre principal avec une complexité $O(2md_{\mathcal{S}}N_b)$ ou la constante $m = 3$. Ainsi, en considérant que notre arbre possède K_b nœuds l'étape de construction complète prend $O(d_{\mathcal{S}}N_b log(K_b))$.
2. **Apprentissage** : Supposons que nous utilisons N_l échantillons pour apprendre les distributions des données (e. g. $N_l >> N_b$). Comme les N_l points parcourent l'arbre en partant de la racine jusqu'à atteindre une feuille en n'explorant qu'une seule des deux sous-branches de chaque nœud, la classification des points prend donc $O(N_l d_{\mathcal{S}} log(K_b))$. Pour chaque nœud rencontré, les matrices de covariance et moyenne sont mises à jour ce qui requiert un temps de $O(N_l d_{\mathcal{R}}^2 log(K_b))$. L'étape d'apprentissage totale requiert donc $O(N_l(d_{\mathcal{S}} + d_{\mathcal{R}}^2)log(K_b))$.
3. **Requête** : Supposons que nous utilisons N_q points en requête de notre arbre. Tout d'abord nous effectuons la recherche des K_q plus proches nœuds demandant une complexité de $O(N_q d_{\mathcal{S}} K_q)$. Ensuite, l'estimation des gaussiennes anisotropes nécessite $O(N_q d_{\mathcal{R}}^2 K_q)$. Au total, notre étape de requête possède une complexité de $O(N_q(d_{\mathcal{S}} + d_{\mathcal{R}}^2)K_q)$.

Le coût mémoire de notre structure dépend uniquement du nombre de nœuds créés lors de l'étape de construction. En effet chaque point appris lors de l'étape d'apprentissage ne fait que mettre à jour les données apprises par chaque nœud de l'arbre sans prendre plus de mémoire. Par conséquent, le coût total en mémoire de notre structure de données est $O(K_b d_{\mathcal{R}}^2)$.

4.4.2 Discussion

Il est intéressant de noter que l'arbre peut être construit sur un ensemble d'échantillons différents de ceux de la base d'apprentissage. De toute évidence pour que cette propriété soit intéressante, il est nécessaire que l'ensemble utilisé pour la construction soit significatif par rapport à l'ensemble des échantillons de l'étape d'apprentissage. Outre l'accélération évidente qu'apporte la diminution du nombre d'échantillons, notre structure permet un apprentissage progressif des données. Il devient possible de raffiner la précision des distributions pour un budget mémoire constant en apprenant des données supplémentaires.

Cette propriété est l'un des aspects principaux de notre CovTree. Nous pouvons donc apprendre la distribution d'échantillons issue de grandes bases de données tout en contrôlant la quantité mémoire nécessaire à son apprentissage. Cette propriété permet également de

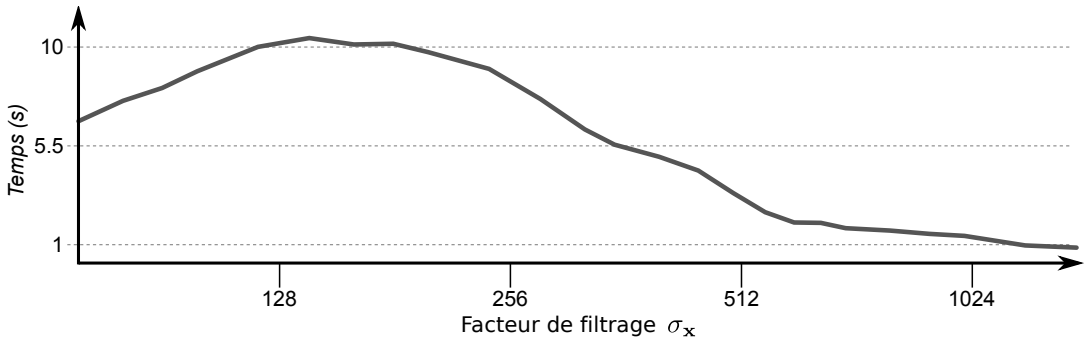


FIGURE 4.6: Temps de calcul (en seconde) pour estimer 5.10^5 requêtes (excluant les étapes de construction et d'apprentissage qui sont exécutées une seule fois) par notre CovTree pour différentes échelles $\sigma_{\mathbf{q}}$. Le temps est mesuré en se contraignant à l'utilisation d'un seul cœur de calcul sur un PC avec un processeur 2.4 GHz Intel Xeon avec 12 GB de mémoire.

pouvoir réapprendre et/ou mettre jour les distributions sans pour autant avoir besoin de reconstruire complètement l'arbre.

Une fois les données apprises, l'arbre peut être réutilisé pour différentes applications sans avoir besoin de recommencer son apprentissage complet. Cette propriété, particulièrement utile dans les technologies mobiles qui possèdent des capacités limitées, est possible grâce à notre algorithme de requête. L'estimation d'une gaussienne anisotrope pour tout voisinage de centre $\mathbf{q} \in \mathcal{S}$ et toute échelle $\sigma_{\mathbf{q}} \in \mathbb{R}$ ne requiert pas de recalculer l'arbre.

Nous présentons en Figure 4.6 la courbe des temps de calcul pour interroger notre CovTree pour différentes échelles $\sigma_{\mathbf{q}}$. Il est intéressant de noter que le temps requis pour résoudre une requête reste peu affecté par la taille $\sigma_{\mathbf{q}}$ de la requête. Cet avantage est dû à l'apprentissage des distributions à différentes échelles qui permet de limiter le nombre de nœuds (et du coup la complexité de calcul) nécessaires pour évaluer une requête. Comme nous le montrons en Figure 4.7, pour des échelles $\sigma_{\mathbf{q}}$ faibles, le parcours de l'arbre est uniquement effectué en profondeur. Pour des échelles $\sigma_{\mathbf{q}}$ élevées, au lieu de parcourir un grand nombre de nœuds (et ainsi de renvoyer les variations d'un grand nombre de feuilles), notre démarche permet d'effectuer un parcours en largeur de l'arbre tout en limitant son exploration en profondeur. Le pire des cas correspond aux échelles $\sigma_{\mathbf{q}}$ de taille moyenne. En effet, la requête doit effectuer un parcours en largeur et en profondeur. Il en résulte un grand nombre de nœuds à explorer pour pouvoir estimer cette requête.

4.5 Limitations et possibles améliorations

Dans ce chapitre, nous avons introduit une nouvelle structure de données capable de traiter une famille continue de gaussiennes multivariés locales. Notre structure est efficace pour apprendre de grandes quantités d'échantillons avec peu d'espace mémoire et permet une estimation rapide des requêtes. Le modèle extrait varie continûment sur un domaine de valeurs $\mathcal{R}$ quand le point requête varie sur un domaine spatial $\mathcal{S}$ (potentiellement différent). En plus de la position, différentes échelles peuvent être spécifiées à la requête permet-

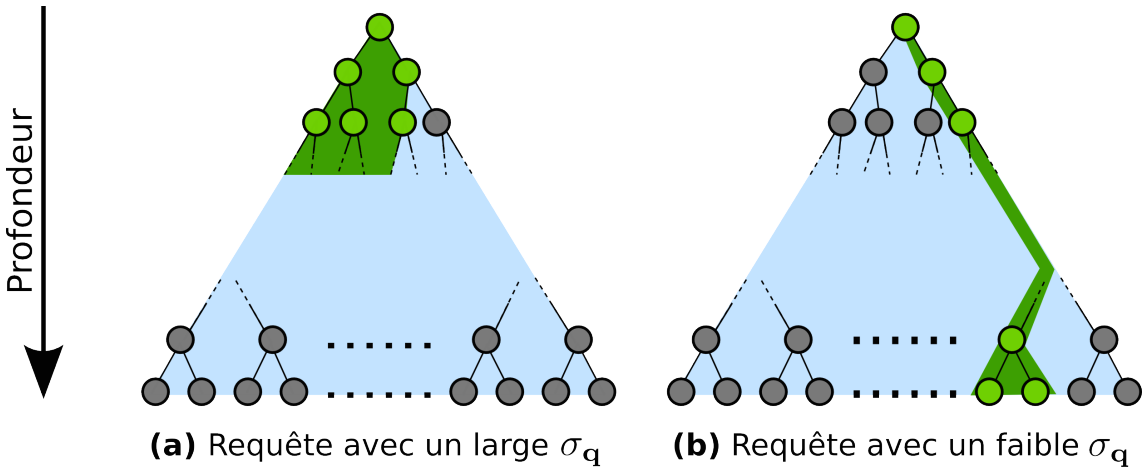


FIGURE 4.7: Notre requête accélérée permet de limiter le nombre de nœuds du CovTree à explorer (en vert) pour estimer une distribution anisotrope locale. (*a*) Quand la requête possède une échelle $\sigma_{\mathbf{q}}$ large, seuls les nœuds supérieurs de l'arbre sont explorés. (*b*) Pour des valeurs plus petites de $\sigma_{\mathbf{q}}$, l'arbre est parcouru plus en profondeur tout en limitant son parcours en largeur.

tant de définir différents degrés de localité spatiale pour le modèle statistique extrait. Cette structure de données sera utilisée dans le chapitre suivant pour résoudre différents problèmes de restauration.

Néanmoins, notre structure de données peut être améliorée. En effet, lors de la construction de l'arbre, nous définissons un partitionnement de l'espace par l'intermédiaire d'un *bsp-tree*. Par rapport à d'autres structures de partitionnement, cet type d'arbre permet de mieux gérer l'anisotropie des données. Néanmoins, il n'offre aucune garantie sur la conservation de la topologie des échantillons que nous souhaitons apprendre. Par conséquent, les sous-espaces définis de cette manière ne sont pas obligatoirement les plus significatifs pour représenter la variété sous-jacente aux données. Il en résulte une augmentation significative du nombre de nœuds pour représenter les échantillons ainsi qu'une mauvaise estimation des matrices de covariances correspondantes. Ce genre de problème peut être résolu par l'utilisation de structures de partitionnement qui respectent la topologie des données.

Dans tout ce chapitre, nous avons fait la supposition que l'ensemble des échantillons étaient connus à l'avance ou qu'une partie suffisamment significative était connue. Pour de nombreuses applications où les échantillons arrivent au fur et à mesure (par ex. traitement vidéo, internet), il est impossible de définir un sous-ensemble significatif sans connaître l'intégralité des données. Ce type de problèmes peut être géré par notre structure de données en modifiant l'étape de construction. Une solution possible serait de partitionner l'arbre progressivement en fonction des données qui sont fournies en entrée de l'arbre. Il faut donc distinguer trois cas : *(a)* quand le nombre d'échantillons d'un nœud est trop faible, il est nécessaire de garder en mémoire l'ensemble des échantillons *(b)* quand le nombre d'échantillons d'un nœud est trop élevé le sous-espace correspondant doit être divisé en deux *(c)* sinon il faut garder en mémoire la distribution des échantillons. Malheureusement, une telle solution ne permet pas de résoudre les problèmes concernant la propagation des données apprises d'un nœud à ses enfants.

Une autre amélioration possible de notre structure de données serait de faire évoluer les données apprises avec le temps. Considérons le cas où un ensemble d'échantillons a besoin d'être appris à des temps $\{t_0, t_1, \ldots, t_n\}$ (par ex. flux vidéo). Nous supposons que pour deux temps consécutifs t_k, t_{k+1}, les variations entre échantillons à apprendre sont faibles. Au temps t_{k+1}, il est intéressant de conserver partiellement les données apprises au temps t_k. Nous proposons d'introduire un facteur d'oubli $0 \leq \alpha \leq 1$. Ce dernier serait multiplié aux données apprises à chaque changement de temps pour permettre de simuler une persistance mémorielle. Pour $\alpha = 0$, l'ensemble d'échantillons appris est oublié d'un temps à l'autre. Pour $\alpha = 1$, l'ensemble des échantillons est gardé en mémoire indéfiniment. Pour toutes valeurs de α, les données sont oubliées au fur et à mesure et remplacées par les nouveaux échantillons. Une telle solution suppose que le partitionnement de l'espace spatial $\mathcal{S}$ n'a pas besoin d'être mis à jour, ce qui est rarement vérifiable en pratique.

CHAPITRE 5

Filtrage collaboratif généralisé

Dans cette partie, nous présentons plusieurs applications de restauration d'images et de surfaces 3D basées sur notre structure de données, le CovTree, introduite au Chapitre 4. En Section 5.1, nous proposons d'appliquer notre CovTree au débruitage d'images pour calculer l'algorithme des Non Local Bayes. Cette section nous permet de valider les performances qualitatives de notre structure de données, en comparant les performances des Non Local Bayes obtenues par notre CovTree et leur implémentation originale [LBM13]. En Section 5.2, nous utilisons notre CovTree pour apprendre un grand ensemble de patchs non bruités de sorte à définir un *a priori* des images naturelles. Nous l'utilisons pour améliorer le débruitages des images et pour définir la reconstruction d'images échantillonnées aléatoirement. En Section 5.3, nous introduisons les problèmes et les idées principaux pour pouvoir étendre les filtres collaboratifs au cas des nuages de points 3D. Dans cette section, nous utilisons les outils développés au Chapitre 3, pour étendre les NLPSS au cas des Non Local Bayes. Nous présentons également comment définir une base de données 3D appliquée à l'augmentation de la résolution du nuage de points.

5.1 Débruitage d'images par filtre collaboratif

Durant ces dernières années, les différentes évolutions des filtres non locaux ont permis de développer un nouveau type de filtre : *les filtres collaboratifs*. Contrairement aux approches par NL-Means qui débruitent les patchs de l'image sans se soucier d'une information globale, ce type de filtre effectue une analyse commune de l'information basée sur des regroupements de patchs similaires. Ces méthodes permettent d'extraire les détails communs d'un groupe tout en préservant les caractéristiques spécifiques du patch considéré. Nous allons particulièrement nous intéresser aux NLB (Non Local Bayes) [LBM13] car ils exploitent l'information commune par l'intermédiaire d'une analyse de la moyenne et de la covariance du groupe de patchs. Malheureusement, comme il est coûteux et difficile d'effectuer des regroupements directement dans l'espace des patchs, ces statistiques sont généralement approchées de sorte que le filtre reste calculable.

La formulation des NLB étant particulièrement adaptée, nous proposons, dans cette section, de les accélérer par l'intermédiaire de notre CovTree généralisé (Section 4.3.3). Cette application nous permet d'étendre au cas des NLB certaines accélérations classiques des

NL-Means (telle que la réduction de la dimensionnalité globale des patchs par une analyse en composantes principales [APG07, Tas08, Tas09]) et de fournir une analyse qualitative des performances de notre CovTree.

5.1.1 Non Local Bayes

Principe

Considérons un ensemble $\mathcal{P} = \{\mathbf{f}_i\}$ de patchs de dimension $d_{\mathbf{P}}$ obtenu à partir d'un ensemble de patchs non bruités $\tilde{\mathcal{P}} = \{\mathbf{r}_i\}$ corrompu par un bruit additif $\mathbf{n}_i$ gaussien d'écart type σ_n :

$$\mathbf{f}_i = \mathbf{r}_i + \mathbf{n}_i$$

La probabilité du bruit qui est rajoutée à chaque patch correspond donc à une gaussienne isotrope d'écart type σ_n.

Introduit par Lebrun et coll. [LBM13], les NLB débruitent un patch d'intérêt $\mathbf{q} \in \mathbb{R}^{d_{\mathbf{P}}}$ en utilisant une Maximisation A Posteriori (MAP) bayésienne. Cette maximisation, équivalente à projeter le patch $\mathbf{q}$ localement sur la variété de patchs *non bruités*, suppose que les variations probabilistes des patchs s'expriment par une gaussienne anisotrope $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{q}}, \boldsymbol{\Sigma}_{\mathbf{q}})$ de moyenne $\boldsymbol{\mu}_{\mathbf{q}}$ et de matrice de covariance $\boldsymbol{\Sigma}_{\mathbf{q}}$. Ne pouvant être directement calculée sur un ensemble de patchs non bruités, cette gaussienne est estimée à partir des patchs bruités $\{\mathbf{f}_i\}$.

Les NLB sont basés sur trois principaux opérateurs inspirés des travaux concernant les *BM3D* (Block Matching 3D) [DFKE07, DFKE08, DFKE09] :

1. **Regroupement** : un groupe de patchs $N(\mathbf{q})$ similaires à $\mathbf{q}$ est formé en considérant une distance quadratique normalisée. Le voisinage est défini à la fois dans l'espace des patchs et dans l'espace spatial (distance pixelique entre les centres des patchs). Cette restriction permet de rendre la recherche des patchs similaires calculables pour de grandes images, car elle devient indépendante de la taille de l'image. De plus, pour ne pas être trop sensible aux patchs parasites, le groupe est généralement limité aux k plus proches patchs.
2. **Débruitage collaboratif** : les variations des patchs associés au groupe sont modélisées de manière probabiliste par une gaussienne anisotrope $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{q}}, \boldsymbol{\Sigma}_{\mathbf{q}})$ qui est ensuite utilisée pour définir un estimateur par MAP du patch débruité. Lors de cette étape, chaque patch est débruité indépendamment les uns des autres.
3. **Agrégation** : chaque pixel de l'image, commun à plusieurs patchs, est associé à plusieurs valeurs débruitées. Une valeur débruitée finale est donc obtenue par une moyenne sur l'ensemble des valeurs débruitées.

Débruitage itératif

Afin d'améliorer les performances du débruitage et d'estimer une gaussienne anisotropique $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{q}}, \boldsymbol{\Sigma}_{\mathbf{q}})$ non biaisée, ces trois étapes sont itérées deux fois.

Lors de la première itération, le patch d'intérêt $\mathbf{q}$ est débruité en considérant l'ensemble des patchs bruités $\{\mathbf{f}_i\}$. La gaussienne anisotrope correspond donc aux variances de l'ensemble

des patchs bruités. Afin de débruiter conjointement les patchs du groupe tout en obtenant une modélisation des patchs non bruités, la matrice de covariance $\mathbf{\Sigma_q}$ est filtrée par un facteur σ_d^2 correspondant à la variance du bruit estimée que contient l'image ($\sigma_d \approx \sigma_n$) : $\mathbf{\Sigma'_q} = \mathbf{\Sigma_q} - \sigma_d^2 Id$. L'estimateur de débruitage $NLB^{(1)}$ est défini pour tout patch bruité $\mathbf{q}$ associé à une moyenne $\boldsymbol{\mu_q}$ et une matrice de covariance $\mathbf{\Sigma_q}$ calculée sur l'ensemble $\{\mathbf{f}_i\}$ par :

$$NLB^{(1)}(\mathbf{q}, \boldsymbol{\mu_q}, \mathbf{\Sigma_q}) = \boldsymbol{\mu_q} + [\mathbf{\Sigma_q} - \sigma_d^2 Id]\mathbf{\Sigma_q}^{-1}(\mathbf{q} - \boldsymbol{\mu_q}) \tag{5.1}$$

Les patchs bruités $\{\mathbf{f}_i\}$ sont donc débruités une première fois par :

$$\hat{\mathbf{r}}_i^{(1)} = NLB^{(1)}(\mathbf{f}_i, \boldsymbol{\mu}_{\mathbf{f}_i}, \mathbf{\Sigma}_{\mathbf{f}_i}) \tag{5.2}$$

Durant la deuxième itération, le patch d'intérêt $\mathbf{q}$ est débruité en considérant l'ensemble des patchs $\{\hat{\mathbf{r}}_i^{(1)}\}$. La gaussienne anisotrope décrit donc, désormais, les variations de l'ensemble des patchs débruités autour du patch d'intérêt débruité une première fois $\mathbf{q}^{(1)} = NLB^{(1)}(\mathbf{q}, \boldsymbol{\mu_q}, \mathbf{\Sigma_q})$. L'estimateur de débruitage $NLB^{(2)}$ est défini pour tout patch bruité $\mathbf{q}$ associé à une moyenne $\boldsymbol{\mu}_{\mathbf{q}^{(1)}}$ et une matrice de covariance $\mathbf{\Sigma}_{\mathbf{q}^{(1)}}$ calculée sur l'ensemble $\{\hat{\mathbf{r}}_i^{(1)}\}$ par :

$$NLB^{(2)}(\mathbf{q}, \boldsymbol{\mu}_{\mathbf{q}^{(1)}}, \mathbf{\Sigma}_{\mathbf{q}^{(1)}}) = \boldsymbol{\mu}_{\mathbf{q}^{(1)}} + \mathbf{\Sigma}_{\mathbf{q}^{(1)}}[\mathbf{\Sigma}_{\mathbf{q}^{(1)}} + \sigma_d^2 Id]^{-1}(\mathbf{q} - \boldsymbol{\mu}_{\mathbf{q}^{(1)}}) \tag{5.3}$$

Les patchs bruités $\{\mathbf{f}_i\}$ sont donc débruités une deuxième fois par :

$$\hat{\mathbf{r}}_i^{(2)} = NLB^{(2)}(\mathbf{f}_i, \boldsymbol{\mu}_{\hat{\mathbf{r}}_i^{(1)}}, \mathbf{\Sigma}_{\hat{\mathbf{r}}_i^{(1)}}) \tag{5.4}$$

Critère d'homogénéité

Lors du calcul de l'Équation 5.1, certaines instabilités peuvent apparaître lorsque $\sigma_d \leq \sigma_n$. Ces instabilités sont dues à l'estimation de la matrice de covariance débruitée $\mathbf{\Sigma'_q}$. Il faut donc s'assurer par une Analyse en Composantes Principales (ACP) que l'ensemble des valeurs propres de $\mathbf{\Sigma'_q}$ restent positives ou nulles. Malheureusement, effectuer une ACP est d'autant plus coûteux que la dimension de l'espace des patchs $d_{\mathbf{P}}$ est élevée. Les NLB proposent une résolution partielle de ce problème en introduisant un *critère d'homogénéité*.

Dans le cas où l'ensemble des valeurs propres de $\mathbf{\Sigma'_q}$ sont négatives, les variations du groupe dans n'importe quelle direction correspondent uniquement à celles du bruit (cas d'un groupe de patchs homogène). Par conséquent, le meilleur estimateur pour le patch débruité est un patch moyen $\bar{\mathbf{r}}$ dont les valeurs sont toutes définies par :

$$\forall k \in [\![1, d_{\mathbf{P}}]\!], \bar{\mathbf{r}}(k) = \sum_{\mathbf{f}_i \in N(\mathbf{q})} \sum_{l=1}^{d_{\mathbf{P}}} \mathbf{f}_i(l) \tag{5.5}$$

Le critère d'homogénéité permet de vérifier que le patch d'intérêt $\mathbf{q}$ n'est pas un patch homogène avant d'appliquer l'une des deux Équations 5.1 ou 5.3 accélérant ainsi le temps de calcul. En considérant un ensemble de m valeurs (correspondant à l'ensemble des valeurs contenues dans chaque patch du groupe), le critère d'homogénéité définit l'écart type des valeurs du groupe σ_h :

$$\sigma_h^2 = \frac{1}{m-1}\left(\sum_{\mathbf{f}_i \in N(\mathbf{q})} \sum_{k=1}^{d_{\mathbf{P}}} \mathbf{f}_i(k)^2 - \left(\sum_{\mathbf{f}_i \in N(\mathbf{q})} \sum_{k=1}^{d_{\mathbf{P}}} \mathbf{f}_i(k)\right)^2\right) \tag{5.6}$$

Lorsque $\sigma_h \leq \sigma_n$, le patch débruité est défini par l'Équation 5.5 sinon il reste débruité par les Équations 5.1 ou 5.3.

Il est important de noter que ce critère permet de résoudre les problèmes d'instabilités quand l'ensemble des valeurs propres de $\mathbf{\Sigma}'_{\mathbf{q}}$ sont négatives, mais ne permet pas résoudre les instabilités de calculs lorsqu'une partie des valeurs propres sont négatives. En pratique, ce problème reste peu visible, car il est atténué par l'étape d'*Agrégation* et l'utilisation de la double itération de l'algorithme.

5.1.2 Débruitage d'images par CovTree

Adaptation des NLB

Nous rappelons que le CovTree, tel qu'il a été introduit en Section 4.3.3, est une structure de données capable d'apprendre les variations d'un ensemble d'échantillons $(\mathbf{p}_i, \mathbf{f}_i)$ où $\mathbf{p}_i$ est une position dans le domaine spatial $\mathcal{S}$ et $\mathbf{f}_i$ est un attribut du domaine des attributs $\mathcal{R}$. Le domaine spatial $\mathcal{S}$ est utilisé pour effectuer des regroupements entre les échantillons et le domaine des attributs $\mathcal{R}$ est utilisé pour analyser les variations des échantillons. Pour toute requête $(\mathbf{q} \in \mathcal{S}, \sigma_{\mathbf{q}} \in \mathbb{R})$, la structure retourne la gaussienne anisotrope $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{q},\sigma_{\mathbf{q}}}, \mathbf{\Sigma}_{\mathbf{q},\sigma_{\mathbf{q}}})$ de moyenne $\boldsymbol{\mu}_{\mathbf{q},\sigma_{\mathbf{q}}} \in \mathcal{R}$ et de covariance $\mathbf{\Sigma}_{\mathbf{q},\sigma_{\mathbf{q}}} \in \mathcal{R} \times \mathcal{R}$ modélisant les variations des échantillons compris dans un voisinage spatial $N(\mathbf{q}, \sigma_{\mathbf{q}})$.

La formulation des NLB étant particulièrement adaptée pour être calculée par notre CovTree, nous pouvons faire un parallèle entre les opérateurs du CovTree et les étapes des NLB. Par exemple, les opérateurs d'*Apprentissage* et de *Requête* peuvent être utilisés pour estimer une gaussienne anisotropique équivalente à celle estimée lors de l'étape de *Débruitage collaboratif*. Dans ce cas-là, le domaine des attributs $\mathcal{R}$ est défini par l'espace des patchs. Les attributs appris correspondent donc aux patchs de l'image.

Pour pouvoir appliquer les deux itérations des NLB de manière équivalente, nous devons utiliser deux CovTree différents. La première est construite et apprise en utilisant l'ensemble des patchs de l'image bruités $\{\mathbf{f}_i\}$ puis utilisée afin de débruiter l'ensemble des patchs par l'Équation 5.1 pour obtenir un premier débruitage des patchs $\{\hat{\mathbf{r}}_i^{(1)}\}$. La deuxième est définie par rapport à l'ensemble des patchs débruités $\{\hat{\mathbf{r}}_i^{(1)}\}$ de sorte à pouvoir être utilisée par l'Équation 5.3 pour définir le patch débruité final.

La gaussienne anisotropique $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{q},\sigma_{\mathbf{q}}}, \mathbf{\Sigma}_{\mathbf{q},\sigma_{\mathbf{q}}})$ estimée par notre CovTree dépend explicitement de la taille du voisinage $\sigma_{\mathbf{q}}$ considéré. Contrairement aux NLB originaux qui fixent définitivement cette taille pour accélérer le calcul du filtre, nous pouvons estimer différentes images débruitées en changeant la taille $\sigma_{\mathbf{q}}$ avec une influence négligeable sur le temps de calcul total.

Critère de regroupement

La distinction entre le domaine spatial $\mathcal{S}$ et celui des attributs $\mathcal{R}$ nous permet de définir différents critères de regroupement. En effet l'étape de *Construction* de notre CovTree permet d'effectuer des divisions dans $\mathcal{S}$ qui servent à rassembler les échantillons proches.

La *Construction* du CovTree est donc équivalente à l'étape de *Regroupement*. La définition des données spatiales utilisées par notre CovTree permet d'introduire différents types de regroupement $N(\mathbf{q})$.

Nous introduisons donc trois différents types de voisinage pour les NLB :

1. **Voisinage global** : nous définissons un domaine spatial $\mathcal{S}$ qui est confondu avec le domaine des attributs $\mathcal{R}$ par $\mathbf{p} = \mathbf{f}_i$. Dans ce cas, le voisinage $N(\mathbf{q})$ dépend uniquement de la similarité entre les patchs bruités $\mathbf{f}_i$ et le patch requête $\mathbf{q}$. Le domaine spatial $\mathcal{S}$ est donc de dimension $d_{\mathcal{S}} = d_{\mathbf{P}}$.
2. **Voisinage local** : afin de définir un voisinage équivalent à celui utilisé dans la version originale des NLB, nous proposons d'augmenter $\mathcal{S}$ avec les coordonnées pixéliques du centre du patch $\mathbf{x} = (x_i, y_i)$. Par conséquent, les $\mathbf{p}_i$ doivent désormais être proches spatialement de $\mathbf{q}$. Ce voisinage est défini dans un domaine spatial $\mathcal{S}$ de dimension $d_{\mathcal{S}} = d_{\mathbf{P}} + 2$.
3. **Voisinage local compressé** : l'idée ici est d'adapter la réduction de dimensionnalité des patchs proposée dans le cas des NL-Means [APG07, Tas08, Tas09] au NLB. Nous proposons donc de réduire les $d_{\mathbf{P}}$ dimensions de l'espace des patchs aux $l < d_{\mathbf{P}}$ dimensions principales par une ACP globale de l'ensemble des patchs bruités de l'image. Dans ce cas, $\mathcal{S}$ est de dimension $d_{\mathcal{S}} = l + 2$.

Il est important de noter que seul le domaine spatial $\mathcal{S}$ est changé (et *a fortiori* les regroupements des différents patchs). En particulier, le domaine des attributs $\mathcal{R}$ ne change pas, ce qui permet de calculer les NLB avec différent types de voisinages sans avoir à modifier les équations originales.

Critère d'homogénéité

Tout comme dans la version originale des NLB, nous pouvons définir un critère homogène calculable à partir de la moyenne $\boldsymbol{\mu}_{\mathbf{q}}$, la matrice de covariance $\boldsymbol{\Sigma}_{\mathbf{q}}$ et des deux constantes de normalisation $w_{\mathbf{q}}$ et $w_{2\mathbf{q}}$ estimées par notre CovTree. Dans notre cas, l'écart type des valeurs du groupe σ_h est défini par :

$$\sigma_{\mathbf{q}}^2 = \frac{1}{w_{\mathbf{q}}^2 d_{\mathbf{P}} - w_{2\mathbf{q}}} \left((w_{\mathbf{q}}^2 - w_{2\mathbf{q}}) \left(Tr(\boldsymbol{\Sigma}_{\mathbf{q}}) + w_{\mathbf{q}} \left\| \boldsymbol{\mu}_{\mathbf{q}} \right\|_2^2 \right) - \frac{w_{\mathbf{q}}^2}{d_{\mathbf{P}}} \left(\sum_{l=1} \boldsymbol{\mu}_{\mathbf{q}}(l) \right)^2 \right) \tag{5.7}$$

De plus, l'ensemble des valeurs de patch moyen $\bar{\mathbf{r}}$ sont définies par

$$\forall k \in [\![1, d_{\mathbf{P}}]\!], \bar{\mathbf{r}}(k) = \frac{1}{d_{\mathbf{P}}} \sum_{l=1}^{d_{\mathbf{P}}} \boldsymbol{\mu}_{\mathbf{q}}(l) \tag{5.8}$$

Le critère d'homogénéité tel qu'il a été introduit dans les cas de NLB originaux reste inchangé dans le cas de l'utilisation de notre CovTree.

5.1.3 Analyse des paramètres

Dans cette partie, nous proposons une analyse des différents paramètres de l'application du CovTree au débruitage d'images. Nous n'analysons pas ici l'influence de l'échelle d'arrêt de notre CovTree car il est lié uniquement à la taille mémoire finale utilisée par notre

structure. Nous n'analysons pas non plus l'influence de la taille des patchs sur la reconstruction, nous renvoyons donc le lecteur vers une analyse plus approfondie de ce paramètre effectué par Lebrun et coll. [LBM13].

Facteur de filtrage

Le facteur de filtrage σ_d correspond à la quantité de bruit que contiennent les patchs bruités. Nous présentons en Figure 5.1 les restaurations d'une image corrompue par un bruit gaussien d'écart type $\sigma_n = 0.08$ obtenu pour différentes valeurs de σ_d. Pour mieux analyser l'influence de ce paramètre sur la qualité de la restauration finale, une seule itération des NLB sans utiliser le critère d'homogénéité a été appliquée (Équation 5.1). Une analyse de la qualité de la restauration de l'image en fonction de l'évolution de ce paramètre n'est généralement pas effectuée car il est évident que la meilleure restauration sera obtenue pour des valeurs , $\sigma_d \approx \sigma_n$. Dans notre cas, l'évolution de ce facteur nous permet d'analyser les performances qualitatives de notre CovTree.

Pour des valeurs de σ_d trop faibles (cas $\sigma_d = 0.04$), la matrice de covariance $\mathbf{\Sigma'_q} = \mathbf{\Sigma_q} - \sigma_d^2 Id$ n'est pas assez débruitée. Ainsi, le patch débruité obtenu est proche du patch bruité passé en requête. Pour des valeurs de σ_d trop élevées (cas $\sigma_d = 1.00$), l'ensemble des valeurs propres associées à la matrice de covariance $\mathbf{\Sigma'_q}$ sont nulles. Le patch débruité obtenu est donc confondu avec la moyenne $\boldsymbol{\mu}_\mathbf{q}$ estimée par notre CovTree. Pour des valeurs de σ_d moyennes (cas $\sigma_d = 0.08$), la matrice de covariance $\mathbf{\Sigma'_q}$ est exploitée efficacement de sorte à extraire l'information commune à l'ensemble des patchs bruités du groupe. L'image résultante réussit à supprimer le bruit tout en préservant les détails fins.

Il est important de noter que l'image moyenne (cas $\sigma_d = 1.00$) conserve les caractéristiques principales de l'image d'entrée. Ainsi, les différentes divisions effectuées par notre structure de données dans le domaine spatial $\mathcal{S}$ respectent la topologie des patchs. L'image moyenne réussit en effet à flouter les zones homogènes tout en préservant les contours principaux de l'image. Une grande majorité des détails de l'image ont néanmoins été supprimés, car le nombre de nœuds de notre structure reste limité. Contrairement à d'autres structures d'accélération telle que les *Gaussian KD-Tree* [AGDL09], notre structure utilise des matrices de covariance pour représenter les variations du groupe. Le pouvoir de représentation des covariances étant plus important que celui de simples moyennes, le nombre de noeuds nécessaire pour apprendre l'information contenue dans l'image est inférieur. Cette propriété est illustrée dans le cas $\sigma_d = 0.08$ où des détails fins de l'image sont restaurés, confirmant que les gaussiennes anisotropes estimées lors des étapes d'*Apprentissage* et de *Requête* sont représentatives des variations de chaque groupe de patchs.

Taille du voisinage spatial

Par l'intermédiaire de notre CovTree, on peut estimer une gaussienne anisotrope pour différentes tailles de voisinage $\sigma_\mathbf{q}$. Contrairement au NLB classique, il est possible d'estimer les gaussiennes anisotropes pour différentes échelles pour un coût équivalent (Section 4.4.2). Nous présentons en Figure 5.2, les restaurations de trois différentes images bruitées avec un bruit additif gaussien d'écart type $\sigma_n = 0.5$ obtenues pour différentes tailles $\sigma_\mathbf{q}$.



FIGURE 5.1: Restauration d'une image corrompue par un bruit gaussien d'écart type 0.08 obtenu pour différents σ_d. Le choix idéal de σ_d est représenté en gras.

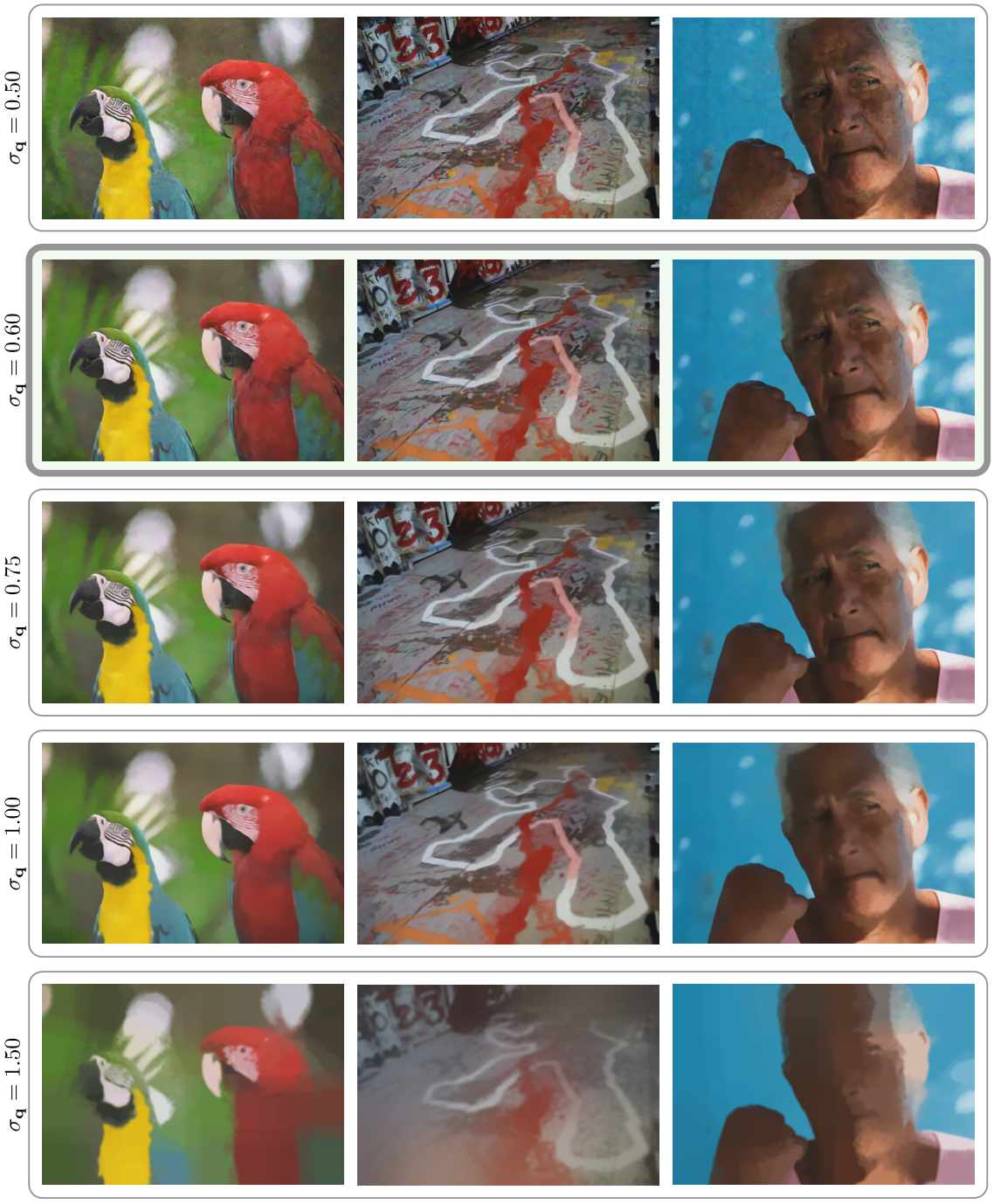


FIGURE 5.2: Restaurations d'une image corrompue par un bruit additif gaussien d'écart type 0.2 obtenues pour différentes échelles d'estimation de la gaussienne anisotrope $\sigma_{\mathbf{q}}$. Plus la taille $\sigma_{\mathbf{q}}$ est élevée plus la gaussienne anisotrope résultante est grossière. La taille idéale pour estimer la matrice de covariance est représentée en gras. Contrairement au NLB, la taille du voisinage peut être changée sans nécessiter de calculs supplémentaires.

Pour des valeurs de $\sigma_{\mathbf{q}}$ trop larges (cas $\sigma_{\mathbf{q}} = 1.5$), la gaussienne anisotrope estimée est trop grossière pour correctement modéliser la variété sous-jacente des patchs débruités.

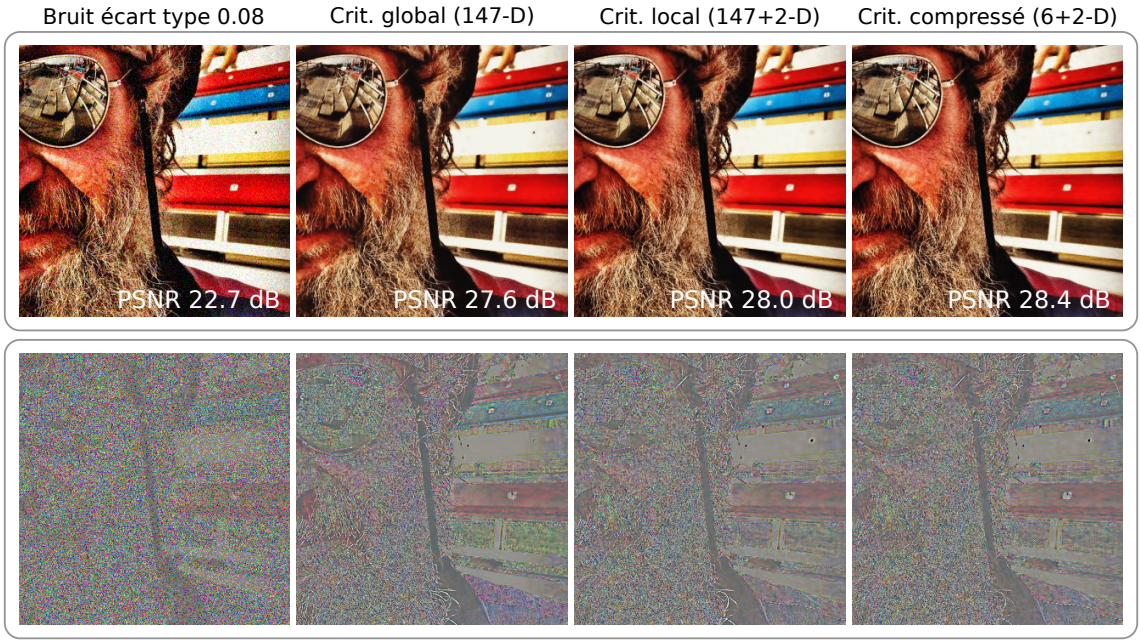


FIGURE 5.3: Comparaison des restaurations obtenues pour différents critères de voisinage. La première ligne correspond aux débruitages obtenus et la deuxième à l'écart entre le débruitage et l'image originale.

Par conséquent, l'image restaurée perd toute l'information haute fréquence qu'elle contient entraînant la création de zones floues.

Pour des tailles $\sigma_{\mathbf{q}}$ faibles (cas $\sigma_{\mathbf{q}} = 0.5$), il n'est pas possible d'extraire les caractéristiques communes à chaque groupe. Ainsi, la gaussienne anisotrope résultante permet de représenter uniquement les structures de bruit locales ce qui se traduit par l'apparition d'un bruit de haute fréquence sur les images restaurées. La taille de voisinage idéale (cas $\sigma_{\mathbf{q}} = 0.6$) doit être choisie par rapport à la quantité de bruit contenue dans les patchs bruités. Une telle taille permet de limiter l'apparition des hautes fréquences tout en préservant les caractéristiques de l'image.

Choix du voisinage

Nous présentons en Figure 5.3 différentes restaurations d'une image corrompue par un bruit additif gaussien d'écart type 0.08 pour différents types de voisinages.

En utilisant un critère *global*, les regroupements de patchs se font uniquement en considérant une distance calculée dans l'espace des patchs. L'ensemble des patchs appris étant bruité, cette distance est très dépendante de la quantité de bruit dans les patchs. Ainsi, les groupes de patchs contiennent plus facilement des patchs parasites influençant négativement la qualité de la restauration finale. Un critère global peut donc être difficilement utilisé dans le cas de patchs bruités.

Pour résoudre ce problème, il suffit d'augmenter de deux dimensions le domaine spatial correspondant à la position pixelique du centre du patch. Ce critère *local* permet de limiter

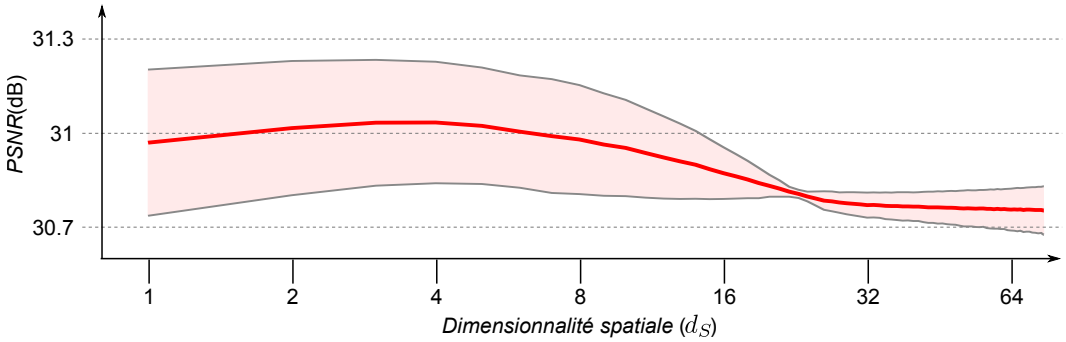


FIGURE 5.4: Évolution du PSNR du débruitage de 20 images par notre CovTree-NLB en fonction de la réduction par ACP de la dimensionnalité du domaine spatial. La courbe moyenne est représentée en rouge et l'écart type des données centrées en gris. Les courbes de PSNR de chaque images présentent des formes similaires et montrent un pic autour des valeurs $d_{\mathcal{S}} = 4$.

le nombre de points parasites présents dans un groupe. L'image restaurée est donc d'une meilleure qualité. Dans la version originale des NLB, ce critère est également utilisé afin d'accélérer le calcul du filtre en rendant indépendante de la taille de l'image l'étape de *Regroupement*.

Comme nous le présentons en Figure 5.4, le critère *compressé* qui effectue une réduction de dimensionnalité par ACP appliqué sur le domaine $\mathcal{S}$ permet non seulement d'accélérer la recherche, mais de produire également de meilleurs résultats. Ce phénomène peut être expliqué par le fait que les valeurs propres les plus petites contiennent principalement du bruit. En gardant uniquement les valeurs propres les plus élevées, la distance entre les patchs devient donc moins sensible au bruit permettant de définir des regroupements de patchs plus pertinents.

Schéma itératif

Nous présentons en Figure 5.5, trois images restaurées obtenues après la première itération (Équation 5.1 puis la deuxième itération (Équation 5.3) des NLB. Les images sont corrompues avec un bruit additif gaussien d'écart type $\sigma_n = 0.2$.

La première itération des NLB permet de récupérer les caractéristiques principales de l'image. Étant calculée uniquement sur l'ensemble des patchs bruités, l'estimation des gaussiennes anisotropes reste néanmoins biaisée entraînant l'apparition d'artefacts de hautes fréquences et de changements de teinte de certaines zones de l'image.

Appliquer une deuxième itération calculée sur l'ensemble des patchs débruités permet de corriger ces problèmes. Désormais les variations et les regroupements sont calculés en considérant une distance entre les patchs qui n'est plus biaisée par le bruit. Les regroupements formés sont donc plus pertinents et permettent de mieux faire ressortir les détails communs à chaque groupe de patchs. Les images résultantes obtenues préservent mieux les caractéristiques.



FIGURE 5.5: Images débruitées obtenues après la première et la deuxième itération de notre CovTree-NLB à partir de trois images corrompues par un bruit additif gaussien d'écart type 0.2. Afin de mieux comparer les performances, nous fournissons des agrandissements des trois images sur les deux lignes centrales.

Néanmoins, lorsque l'information d'un groupe est trop filtrée lors de la première itération, il devient difficile de récupérer cette information lors de la deuxième itération. Inversement, si le bruit présent dans le groupe n'est pas totalement éliminé alors il pourra être considéré comme une caractéristique principale du groupe diminuant ainsi la qualité de la restauration.

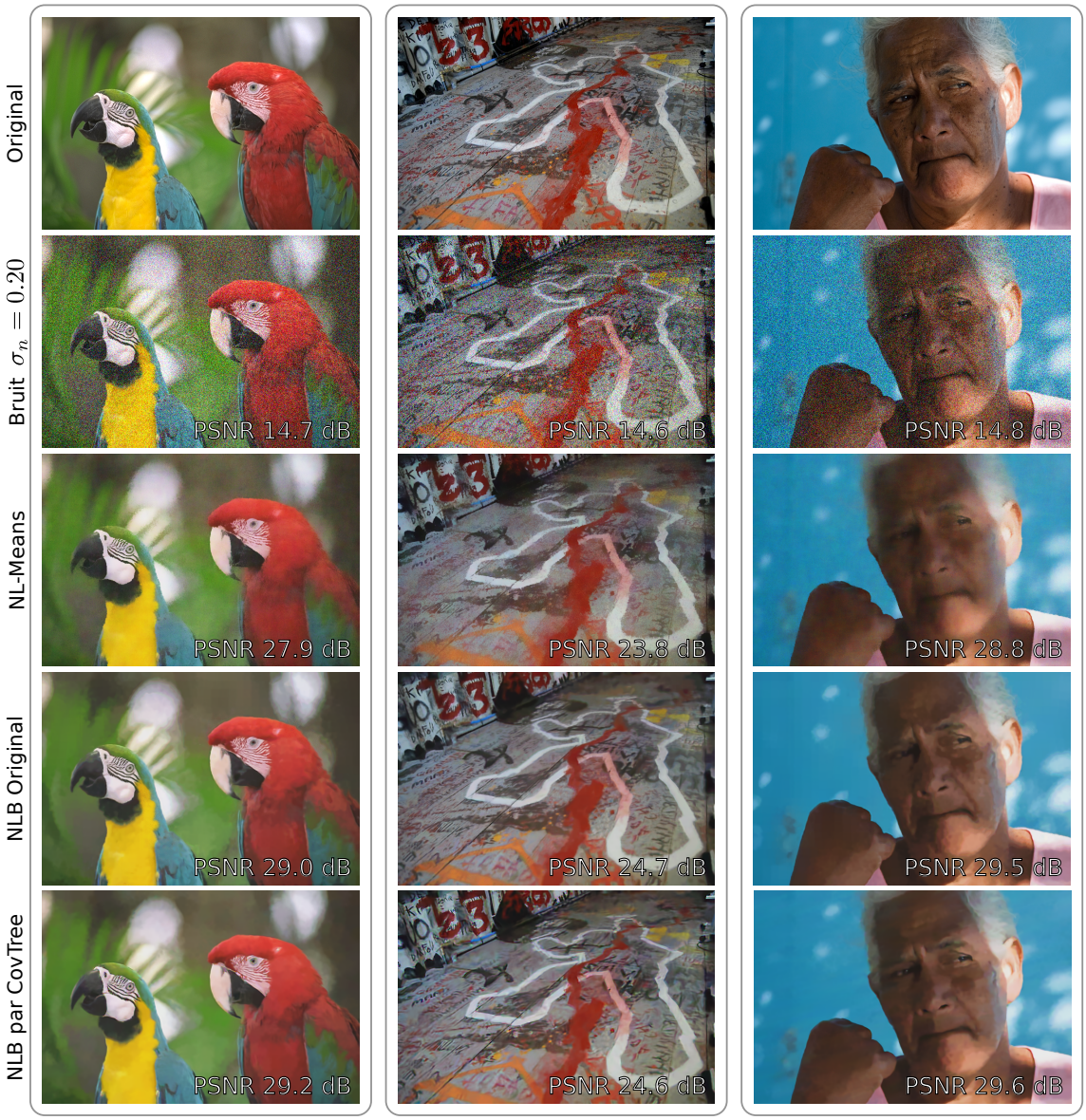


FIGURE 5.6: Résultats de restaurations obtenus pour différents algorithmes de débruitages non locaux : les NL-Means [GO12], les NLB originaux [LBM13] et notre CovTree-NLB. Un agrandissement de ces images est proposé en Figure 5.7

5.1.4 Résultats

Nous présentons en Figure 5.6 les résultats obtenus pour trois méthodes : les NL-Means calculées par l'approche de Gastal et coll. [GO12], les NLB originaux [LBM13] et notre CovTree-NLB. Ces méthodes sont appliquées à trois différentes images corrompues par un bruit additif gaussien d'écart type 0.2. Pour mieux percevoir les différences entre les différentes méthodes, nous présentons en Figure 5.7 des vues détaillées de ces différents résultats. Les résultats du CovTree sont obtenus sans utiliser le critère d'homogénéité et en réduisant la dimensionnalité des patchs aux 6 principales dimensions.

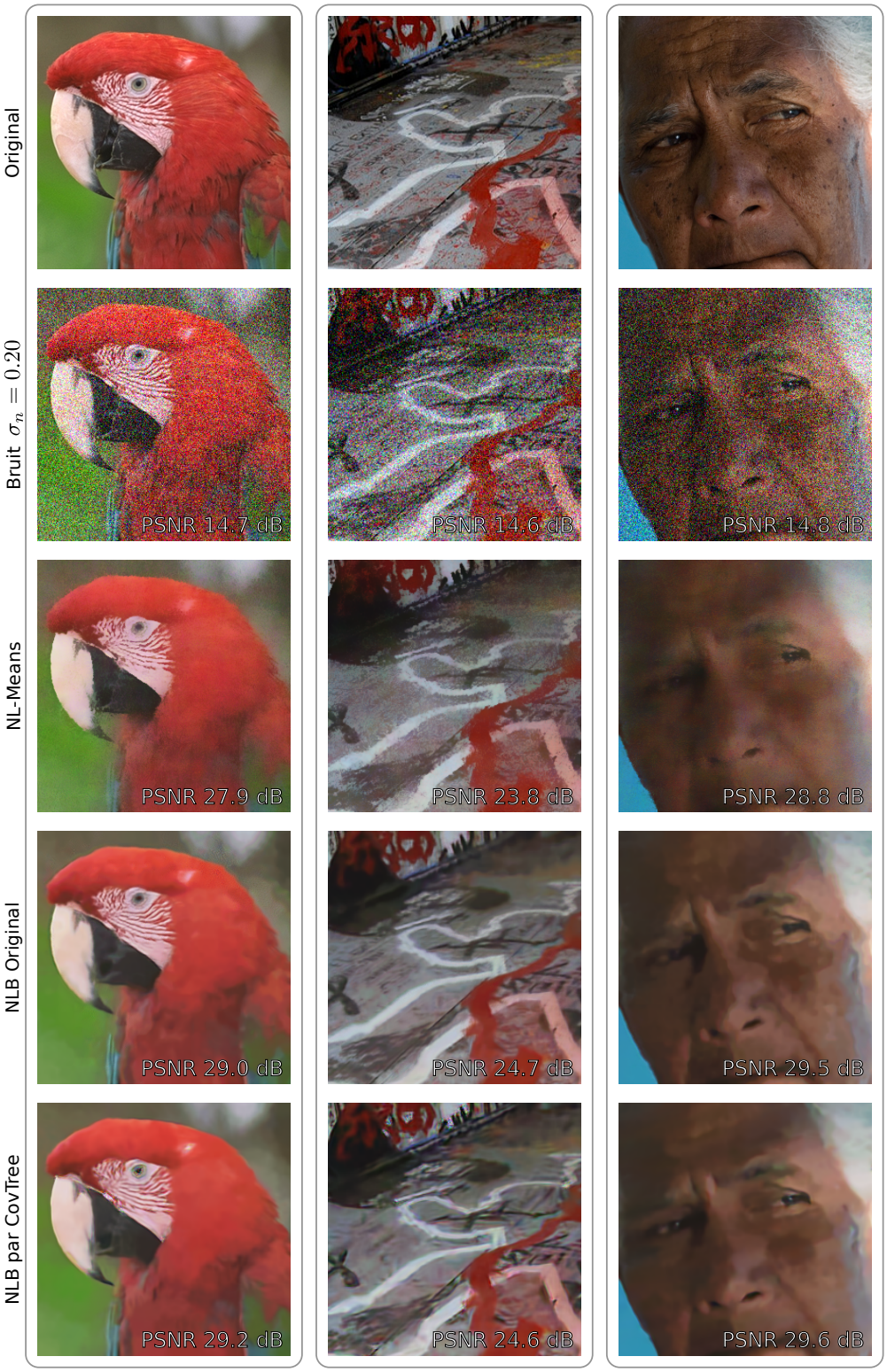


FIGURE 5.7: Comparaison de différents algorithmes non locaux : les NL-Means [GO12], les NLB originaux [LBM13] et notre CovTree-NLB.

Dans le cas des NL-Means, les pixels de l'image sont débruités à partir d'une combinaison de valeurs bruitées. Le poids de chaque élément est calculé en fonction d'une mesure de similarité évaluée à partir d'un voisinage autour du pixel. Malheureusement, ce voisinage est défini à partir de l'image bruitée. Par conséquent, la mesure de similarité aura tendance à rechercher des pixels dans l'image qui possèdent des voisinages qui sont bruités de la même manière que le patch d'intérêt. Ce phénomène reste peu visible pour des écarts types σ_n faibles, mais se traduit par l'apparition d'un bruit résiduel haute fréquence pour des valeurs de σ_n élevées. Ce problème est particulièrement visible en Figure 5.7, où il entraine l'apparition d'imprécisions et la modification des contours de l'image.

Ce problème est corrigé dans les NLB originaux en analysant l'information commune à un groupe de patchs similaires. À la différence des NL-Means, le filtrage est effectué collaborativement. Les caractéristiques principales communes à l'ensemble des patchs du groupe sont donc extraites indépendamment du bruit puis utilisées pour filtrer le patch. Les NLB permettent donc de mieux préserver les détails principaux contenus dans l'image sans introduire de bruit haute fréquence.

Notre CovTree-NLB permet d'obtenir des résultats légèrement supérieurs aux NLB originaux. Nous pouvons expliquer cette différence par l'utilisation du critère de voisinage local compressé. Il permet de former des groupes de patchs similaires moins dépendants du bruit. De plus, dans la formulation originale des NLB, la moyenne et la covariance sont estimées avec un poids constant à partir d'une combinaison des k plus proches voisins. Dans notre cas, la combinaison est pondérée en utilisant un noyau gaussien qui limite l'influence des patchs parasites.

En effectuant une comparaison plus détaillée, nous pouvons remarquer que les NLB originaux ont tendance à créer des zones régulières. Ce problème est dû à l'utilisation du critère homogénéité qui a tendance à uniformiser l'ensemble des valeurs du patch. Comme notre CovTree-NLB n'utilise pas le critère d'homogénéité, le patch est remplacé dans notre cas par la moyenne estimée durant la phase de requête de notre CovTree. Comme nous l'avons vu précédemment, la moyenne estimée par notre structure permet de bien préserver les caractéristiques principales. Elle remplace donc les zones régulières qui provoquent un phénomène de quantification par des variations lisses entre les différentes zones.

Pour conclure, les deux implémentations des NLB produisent des résultats globalement équivalents. Ce constat nous permet donc d'affirmer que notre CovTree est correctement défini pour estimer la moyenne et la covariance d'un ensemble d'échantillons pour différentes requêtes qui peuvent être utilisées dans des approches telles que les NLB.

5.2 Restauration par dictionnaires

Une propriété intéressante de notre CovTree est qu'il peut être utilisé pour apprendre les variations à différentes échelles de grandes bases d'échantillons en utilisant une quantité de mémoire limitée. Nous proposons d'exploiter ce caractère compressif afin de résoudre deux applications : le débruitage par dictionnaire et la reconstruction d'images échantillonnées aléatoirement.



FIGURE 5.8: Échantillons de la base de données de façades parisiennes comprenant plus de 120 images (soit plus de 10^8 patchs).

5.2.1 Principe général

Avec la démocratisation des capteurs numériques et la facilité d'accès aux réseaux de communication, il est devenu facile aujourd'hui de récupérer et/ou partager de l'information. Malgré la multiplication de ces données, il existe encore peu de moyens réussissant à exploiter cette grande quantité d'informations et d'en extraire des caractéristiques utiles. Pourtant, l'utilisation d'une telle connaissance pourrait être particulièrement utile dans des domaines tels que la restauration.

Durant ces dernières années, les méthodes de restauration des images ont proposé des approches de plus en plus performantes pour réussir à récupérer l'information contenue dans des images corrompues. Néanmoins, les performances de ces méthodes resteront toujours limitées du fait que les données sont trop détériorées ou trop rares pour être récupérées. L'utilisation d'une analyse de l'information commune à de grands ensembles d'images pourrait donc permettre de résoudre ce problème tout en améliorant la qualité de la restauration.

Cette idée est confirmée dans [LN11] où l'utilisation de grandes bases d'images est proposée comme un moyen d'apprendre l'*a priori* sous-jacent aux patchs d'images naturelles. Ce genre d'approche (appelée *shotgun NLM* [LCBM12]) a été utilisé pour estimer les limites fondamentales des méthodes de débruitage non locales. Néanmoins, aucune proposition n'a été faite afin de rendre ce genre de méthodes calculables dans le cas d'applications réelles. Une idée similaire a ensuite été proposée par [ZW11]. Elle définit un apprentissage sur une large base de données par l'intermédiaire d'une approche similaire à celle des PLE [YSM12]. Malheureusement, une telle approche nécessite plusieurs jours pour apprendre l'*a priori* sous-jacent.

Nous proposons donc d'utiliser notre CovTree généralisé afin d'apprendre l'*a priori* sous-jacent aux patchs des images naturelles rapidement et en utilisant une quantité de mémoire limitée. Une fois encore, cet avantage est possible, car la quantité de mémoire utilisée par

notre structure de données dépend uniquement de la précision de l'information que l'on souhaite apprendre et non du nombre d'échantillons à apprendre.

Dans toute cette section, nous considérons un ensemble de 10^8 patchs *non bruités* de dimension 7×7 extraits d'une base de plus de 120 images de façades parisiennes. Ces patchs seront utilisés lors de l'étape d'*Apprentissage* de sorte à pouvoir définir une approximation de l'*a priori* sous-jacent de l'ensemble des patchs appartenant aux images naturelles. Nous présentons en Figure 5.8 un sous-ensemble d'images appartenant à cette base de données.

En terme de performance, un tel apprentissage nécessite environ 5 heures de calcul sur un PC possédant un processeur 2.4 GHz Intel Xeon et utilise 8GB de RAM pour retenir l'ensemble de la structure de données. En comparaison, ce temps de calcul est bien plus rapide que les temps de calcul reportés dans [LN11, ZW11] alors que la quantité d'échantillons apprise par notre structure est largement supérieure.

5.2.2 Débruitage d'images

Principe

Les démarches par NLB restaurent l'image en utilisant uniquement un ensemble de patchs bruités. Par conséquent, la gaussienne anisotrope utilisée pour modéliser l'*a priori* sous-jacent représente les variations des patchs bruités. Les NLB résolvent ce problème en débruitant la matrice de covariance d'un facteur σ_d et en itérant deux fois leurs différentes étapes. Néanmoins, l'estimation de la gaussienne reste limitée.

Dans cette section nous proposons de définir une approche similaire à [LCBM12]. Nous définissons un algorithme de débruitage, le *Shotgun NLB*, qui utilise un *a priori* appris sur une grande base de données à partir notre CovTree. Cette idée a deux principaux avantages : nous pouvons augmenter le nombre de patchs appris (par rapport aux NLB classiques) et les données utilisées ne sont pas dégradées par du bruit, permettant ainsi d'augmenter la précision de la gaussienne anisotropique estimée.

En pratique, nous construisons notre arbre en considérant un critère de voisinage *global* (sans coordonnées pixeliques) afin de définir les cellules dans le domaine spatial $\mathcal{S}$. Ensuite, nous apprenons le modèle gaussien de chaque cellule à partir de l'ensemble des patchs non bruités provenant de la base de données. Finalement, la matrice de covariance $\mathbf{\Sigma_q}$ et le vecteur moyen $\boldsymbol{\mu}_\mathbf{q}$ sont estimés à partir d'un patch bruité $\mathbf{q}$ avec un rayon $\sigma_\mathbf{q}$ proche de l'écart type du bruit du patch. Contrairement aux NLB classiques, les gaussiens anisotropes $\mathcal{N}(\boldsymbol{\mu}_\mathbf{q}, \mathbf{\Sigma_q})$ estimées par la structure sont non bruitées. Par conséquent, nous appliquons directement l'Équation 5.3 sans avoir besoin de débruiter la matrice de covariance.

Résultats

En Figure 5.9, nous présentons une comparaison d'images filtrées par les NLB originaux, les CovTree-NLB et notre Shotgun NLB. Les résultats sont obtenus à partir d'images corrompues par un bruit gaussien additif d'écart type 0.08. Dans cette partie, les CovTree-NLB sont calculés à partir du critère de voisinage local compressé aux 6 principales dimen-

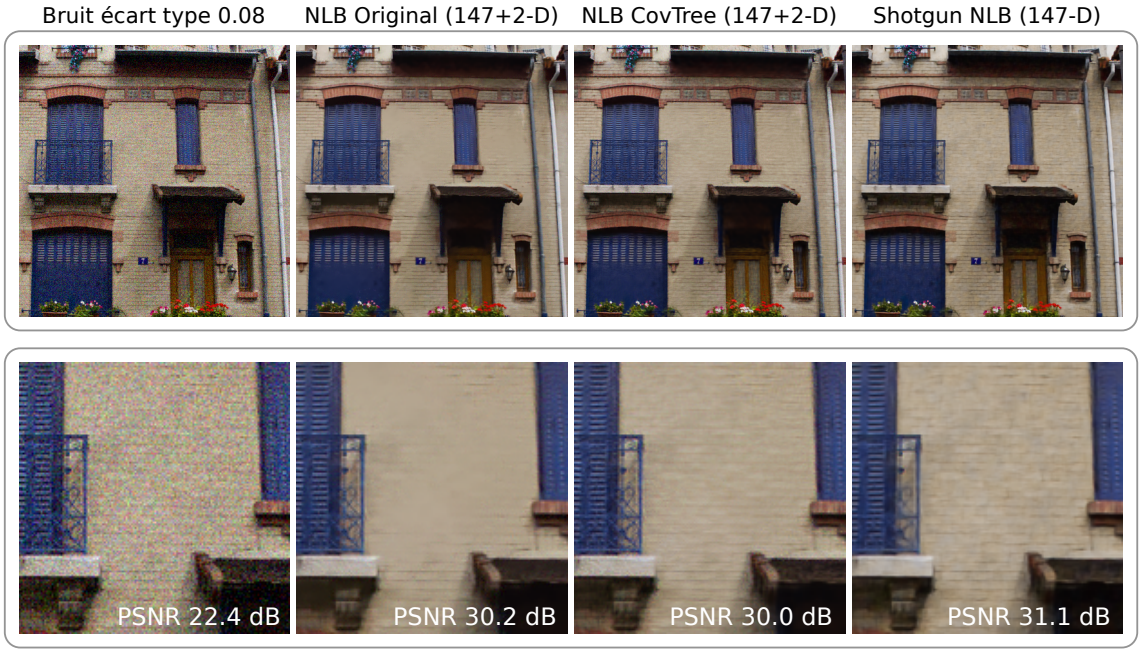


FIGURE 5.9: Comparaison des images débruitées obtenues pour différentes versions des NLB : le NLB original, notre CovTree NLB et notre Shotgun NLB. La deuxième ligne présente des agrandissements des résultats obtenus sur la première ligne.

sions sans utiliser de critère d'homogénéité. Notre Shotgun-NLB est évalué avec une seule itération.

Pour mieux illustrer nos propos, nous allons baser nos explications sur les agrandissements proposés en Figure 5.9 représentant un mur de briques. Ce cas de restauration est particulièrement difficile, car les détails fins et le bruit de l'image sont confondus. La gaussienne anisotrope résultante de ce groupe de patchs sera donc principalement composée du bruit du groupe. En effet, la distance utilisée pour former les groupes de patch aura tendance à rassembler les patchs par rapport à la similarité de leur bruit respectif plutôt que de l'information réelle qu'ils contiennent. Les groupes ainsi formés contiendront donc un grand nombre de patchs parasites qui perturberont l'extraction des caractéristiques communes. La légère perte de performance de notre CovTree-NLB par rapport aux NLB originaux est due à ce problème. Les NLB originaux utilisent le critère d'homogénéité pour éviter d'exacerber les structures de bruits locales. Il permet d'harmoniser l'ensemble des valeurs quand il est difficile de faire la distinction entre l'information contenue dans le groupe et le bruit.

Le Shotgun-NLB est particulièrement efficace pour résoudre ce genre de cas. En effet, il permet d'estimer, en toutes circonstances, des gaussiennes anisotropes représentatives des variations des patchs non bruités. Pour ce faire, il utilise l'information commune de la base de patchs non bruités pour définir un *a priori* qui approche celui des patchs des images naturelles. Cet *a priori* peut donc contenir une information sur les caractéristiques fines des patchs qui se retrouvaient précédemment confondues dans le bruit de l'image. Le Shotgun-NLB permet de restaurer des détails fins de l'image que les filtres classiques ne sont pas capables de retrouver. De plus amples expérimentations restent néanmoins nécessaires

pour déterminer si la restauration obtenue est proche de la limite fondamentale présentée par [LN11].

5.2.3 Reconstruction d'image échantillonnée aléatoirement

Principe

Dans cette section, nous proposons de reconstruire une image à partir d'un sous-ensemble aléatoire de ces pixels. Comme le proposent entre autres les PLE [YSM12], nous proposons d'utiliser l'*a priori* sous-jacent aux patchs des images naturelles qui a été appris par l'intermédiaire de notre CovTree sur une large base de données pour compléter l'information manquante.

Considérons un patch $\tilde{\mathbf{q}}$ échantillonné aléatoirement à partir d'un patch original $\mathbf{q}$ par un opérateur $\tilde{\mathbf{q}} = S\mathbf{q}$. Ici, l'opérateur S dépend de la position spatiale de $\mathbf{q}$. De manière équivalente aux filtres NLB, nous allons considérer ici que les patchs de l'image peuvent être modélisés localement par une gaussienne anisotrope dont la moyenne $\boldsymbol{\mu}_{\mathbf{q}}$ et la covariance $\boldsymbol{\Sigma}_{\mathbf{q}}$ ont été estimées à partir d'un dictionnaire local. Par conséquent, nous pouvons définir le patch reconstruit $\hat{\mathbf{q}}$ comme une MAP bayésienne :

$$\hat{\mathbf{q}} = (\boldsymbol{\Sigma}_{\mathbf{q}} S^H S + \frac{\sigma^2}{2} Id)^{-1} (\boldsymbol{\Sigma}_{\mathbf{q}} S^H \tilde{\mathbf{q}} + \frac{\sigma^2}{2} \boldsymbol{\mu}_{\mathbf{q}}) \tag{5.9}$$

Dans la minimisation précédente, la gaussienne anisotrope $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{q}}, \boldsymbol{\Sigma}_{\mathbf{q}})$ est estimée autour du patch complet original $\mathbf{q}$. Ce patch étant bien évidemment inconnu, nous utilisons une approche itérative basée sur trois étapes pour approcher cette gaussienne :

1. **Estimation** : le patch $\hat{\mathbf{q}}_{t-1}$ reconstruit à l'itération précédente $t-1$ est utilisé en requête de notre CovTree afin de définir une approximation de l'*a priori* statistique local autour de $\mathbf{q}$ représentée par une gaussienne anisotrope $\mathcal{N}(\boldsymbol{\mu}_{\hat{\mathbf{q}}_{t-1}}, \boldsymbol{\Sigma}_{\hat{\mathbf{q}}_{t-1}})$.
2. **Reconstruction** : le patch $\hat{\mathbf{q}}_t$ est reconstruit à l'itération t à partir d'une MAP bayésienne et de la gaussienne anisotrope $\mathcal{N}(\boldsymbol{\mu}_{\hat{\mathbf{q}}_{t-1}}, \boldsymbol{\Sigma}_{\hat{\mathbf{q}}_{t-1}})$ par :

$$\hat{\mathbf{q}}_t = (\boldsymbol{\Sigma}_{\hat{\mathbf{q}}_{t-1}} S^H S + \frac{\sigma^2}{2} Id)^{-1} (\boldsymbol{\Sigma}_{\hat{\mathbf{q}}_{t-1}} S^H \tilde{\mathbf{q}} + \frac{\sigma^2}{2} \boldsymbol{\mu}_{\hat{\mathbf{q}}_{t-1}}) \tag{5.10}$$

3. **Agrégation** : tout comme pour les NLB, un pixel de l'image appartient à plusieurs patchs de l'image. Par conséquent, le patch $\hat{\mathbf{q}}_t$ et l'image sont estimés à l'itération t par une moyenne des valeurs des pixels reconstruits.

En pratique, nous initialisons la reconstruction à partir d'une estimation initiale $\hat{\mathbf{q}}_0$ du patch complet effectué en utilisant une interpolation cubique basée sur une triangulation de Delaunay des pixels connus. De plus, pour obtenir une meilleure reconstruction de l'image, lors de l'étape d'*Estimation* nous utilisons des échelles de filtrage $\sigma_{\hat{\mathbf{q}},t} = \alpha^t \sigma_{\hat{\mathbf{q}},0}$ de plus en plus fines à partir d'une échelle grossière $\sigma_{\hat{\mathbf{q}},0}$ et d'un facteur d'échelle $\alpha \in [0, 1]$.

Résultats

Nous présentons en Figure 5.9 la restauration obtenue à partir d'une image où seulement 20% des pixels échantillonnés aléatoirement ont été conservés. Notre approche est initia-



FIGURE 5.10: Résultat de la reconstruction d'une image dont 20% des pixels échantillonnés aléatoirement ont été retenus. Notre approche est initialisée à partir d'une reconstruction cubique définie sur la triangulation de Delaunay des pixels connus.

lisée par une première interpolation cubique définie sur la triangulation de Delaunay des pixels connus pour des valeurs $\alpha = 0.8$ et $\sigma_{\hat{\mathbf{q}},0} = 0.65$.

Comme nous l'avons expliqué précédemment, nous avons théoriquement besoin de connaître le patch restauré pour pouvoir définir un modèle local des patchs. Pour résoudre ce problème, nous avons introduit une approche itérative qui évalue à chaque étape une reconstruction de plus en plus fine. Ce schéma itératif est initialisé à partir d'une restauration grossière de l'image. Il est important de comprendre que tout le cœur de notre problème de restauration réside dans l'estimation d'une gaussienne anisotrope correcte. C'est grâce à elle que l'information manquante est comblée. Par conséquent, plus la requête sera proche du patch original meilleure sera la restauration.

Pour éviter de bloquer les itérations dans des minimums locaux, nous utilisons des échelles de requêtes $\sigma_{\hat{\mathbf{q}},t}$ décroissantes. Cette idée nous permet de récupérer des détails de l'image qui n'était pas nécessairement présents dans l'approximation originale. Ce phénomène est parfaitement illustré en Figure 5.10 par la restauration des briques ou des carreaux de la porte de la maison. Malheureusement, une telle solution ne permet pas de résoudre toutes les ambiguïtés. Les patchs restaurés peuvent parfois être assez éloignés de leur version originale. Néanmoins, ce problème n'est pas forcément choquant visuellement.

5.2.4 Limitations et possibles améliorations

Dans cette section, nous avons utilisé notre CovTree pour apprendre rapidement la distribution d'une grande base de données de patchs issues d'images naturelles et avec une

quantité de mémoire limitée. Nous avons démontré que ce modèle peut être utilisé efficacement pour résoudre des problèmes de restauration par MAP bayésienne tels que le débruitage ou la reconstruction d'images.

Nous avons fait la supposition que l'*a priori* sous-jacent est appris sur une base de patchs non bruités représentatifs des patchs à restaurer. Cette hypothèse n'est pas toujours vérifiée en pratique. En effet, l'apprentissage est effectué ici sur un ensemble de patchs extraits d'images différentes. Ces images ne sont pas nécessairement acquises avec des appareils ou dans des conditions équivalentes. Par conséquent, des changements de teintes ou de contraste peuvent limiter la qualité de l'apprentissage.

Ce problème peut bien évidemment être résolu en augmentant la quantité d'images apprises. Mis à part l'augmentation de la quantité de temps nécessaire à l'apprentissage, les statistiques apprises pourraient être corrompues par l'introduction de patchs aberrants. Pour résoudre ce problème, une première solution consisterait à apprendre les statistiques indépendamment des changements de contraste et/ou de teintes. Cette solution reviendrait en quelque sorte à normaliser les patchs avant de les apprendre. Notre CovTree devrait donc être modifié pour gérer indépendamment la moyenne et les covariances. Plus simplement, nous pourrions utiliser l'information contenue dans les patchs corrompus quand celles de l'*a priori* ne sont pas suffisante. Cette idée est équivalente à celle de la dégénérescence du noyau non local introduit en Section 3.2.4 pour les NLPSS.

5.3 Restauration de surface 3D

Notre CovTree peut être utilisé pour résoudre des problèmes de restauration d'images tels que le débruitage ou la reconstruction. Nous proposons d'étendre ces différentes applications au cas des nuages de points 3D. Pour cela, nous utilisons les principes et outils introduits pour les NLPSS ainsi que le CovTree dans sa version étendue. Nous présentons uniquement les idées et les problèmes sous-jacents à la mise en œuvre de ces applications.

5.3.1 Positionnement du problème

Nous rappelons ici les notations introduites au Chapitre 3. Soit $\mathcal{X} = (\mathbf{x}_i, \mathbf{n}_i)$ un nuage de point 3D extrait à partir d'une surface S inconnue, où $\mathbf{x}_i \in \mathbb{R}^3$ est la position d'un point et $\mathbf{n}_i \in \mathbb{R}^3$ est la normale qui lui est associée. Pour tout point $\mathbf{x} \in \mathbb{R}^3$, l'opérateur PSS Π^t projette $\mathbf{x}$ sur la surface S^t définie à l'échelle t. Nous appelons respectivement $\Pi^t(\mathbf{x})$, $f(\mathbf{x})$ et $\mathbf{n}(\mathbf{x})$, la projection de $\mathbf{x}$ sur S^t, la distance implicite et la normale associée. En appliquant Π^{t_0} à une échelle t_0 large, nous obtenons une surface grossière S^{t_0} et un champ scalaire résiduel bruité $m(\mathbf{x}_i)$ défini uniquement pour les points du nuage $\mathbf{x}_i \in \mathcal{X}$ (Section 3.2.2).

De manière générale, adapter les méthodes restaurations MAP non locales pose principalement deux contraintes :

1. Tout d'abord, elles nécessitent la **définition d'un signal corrompu et d'un descripteur** pour définir la mesure de similarité entre deux échantillons. En traitement des images (et en particulier dans le cas des filtres collaboratifs), c'est le patch qui



FIGURE 5.11: Reconstruction de surface obtenue à partir de notre opérateur NLB-PSS.

possède le double rôle de descripteur et de signal à débruiter. Dans le cas des NLPSS, le signal est défini par le champ scalaire résiduel bruité $m(\mathbf{x}_i)$ (uniquement pour tout $\mathbf{x}_i \in \mathcal{X}$) et le descripteur par un patch de surface 3D (Section 3.2.3). [1]

2. Ensuite, les formulations MAP (par ex. l'Équation 5.1) nécessitent de **connaître le signal corrompu en tout point du domaine**. En effet, elles peuvent être vues généralement comme une projection d'un échantillon corrompu sur une variété locale (Section 5.1.1). Leur formulation fait donc explicitement intervenir le signal corrompu. Un tel problème ne se pose pas nécessairement dans le cas de formulations EAP (par ex. l'Équation des NLPSS).

En conclusion, tout l'enjeu de l'adaptation des restaurations MAP au cas de surface 3D sera dans la définition d'une valeur du signal corrompue pour tout point de l'espace $\mathbf{x} \in \mathbb{R}^3$.

5.3.2 Surface de points par NLB

Nous proposons maintenant de définir un opérateur NLB-PSS qui étend les NLPSS au cas des NLB. Dans le cas des nuages de points 3D, il est difficile de définir un signal bruité pour tout $\mathbf{x} \in \mathbb{R}^3$. Dans les NLPSS, ce problème a été résolu en utilisant le champ scalaire résiduel bruité $m(\mathbf{p}_i)$ qui est uniquement défini pour les points du nuage $\mathbf{x}_i \in \mathcal{X}$. Pour pouvoir définir un patch de surface 3D, ce signal a été approché par un opérateur PSS Π^{t_1} à une échelle t_1. Π^{t_1} permet de définir une surface S^{t_1} qui approxime finement $\mathcal{X}$. Le patch de surface 3D peut donc être considéré comme une approximation peu filtrée du signal bruité. Il peut donc être débruité par l'Équation 5.1.

Nous définissons l'opérateur NLB-PSS Π_{NLB} pour tout $\mathbf{x} \in \mathbb{R}^3$ par :

$$\Pi_{NLB}(\mathbf{x}) = \mathbf{x} - f_{NLB}(\mathbf{x})\mathbf{n}^{t_0}(\mathbf{x}) \tag{5.11}$$

1. Ces deux notions sont intimement liées aux deux domaines des attributs $\mathcal{R}$ et spatial $\mathcal{S}$ introduit dans la version généralisée de notre CovTree. $\mathcal{R}$ correspond au signal à débruiter et $\mathcal{S}$ aux descripteurs.

où $f_{NLB}(\mathbf{x})$ correspond à la valeur centrale du patch de surface débruitée définie à partir du patch de surface bruité $\mathbf{P}_\mathbf{x}$ par l'Équation 5.3. Pour les NLB-PSS, nous n'effectuons aucune étape d'agrégation En effet, contrairement aux patchs définis en traitement des images, les patchs de surface ne possèdent aucunes valeurs en commun. En pratique, le NLB-PSS est accéléré par notre CovTree. Nous le construisons et l'apprenons en considérant l'ensemble des patchs [2] $\mathbf{P}_{\mathbf{x}_i}$ associé à chaque point $\mathbf{x}_i \in \mathcal{X}$. Une requête de notre arbre est donc définie par un patch de surface $\mathbf{P}_\mathbf{x}$ et un rayon de requête $\sigma_\mathbf{x} \in \mathbb{R}$. Nous présentons en Figure 5.11 les premiers résultats des NLB-PSS.

5.3.3 Restauration de surface par dictionnaire

Comme nous l'avons proposé dans le cas des images, nous pouvons apprendre les variations d'un ensemble de patchs de surface non bruités afin de définir un *a priori* des surfaces 3D. Il peut être utilisé pour des applications telles que le débruitage et la reconstruction de surface.

Problèmes

Dans les NLPSS, les patchs de surface (Section 3.2.3) sont définis par rapport à une surface grossière S^{t_0} (utilisée pour définir les valeurs du patch) et à une largeur de patchs (employée pour estimer l'orientation du patch). Malheureusement ces deux paramètres sont dépendants de la résolution du nuage de points considérée. Par conséquent, un nuage acquis à différentes résolutions ne produira pas des ensembles de patchs qui contiennent des informations équivalentes.

Dans le cas de grands ensembles de nuages de points, ce problème est d'autant plus présent que les nuages ne sont pas acquis par le même capteur. Pour résoudre ce problème, nous proposons d'extraire les patchs du nuage à différentes échelles. La base de données ainsi constituée sera donc moins dépendante des problèmes d'acquisition. Plus d'expérimentations sont néanmoins nécessaires pour valider cette hypothèse.

Densification de nuages de points 3D

L'apprentissage de grand ensemble de patchs est particulièrement utile pour augmenter la résolution d'un nuage de points. Pour pouvoir être défini de manière équivalente aux problèmes de reconstruction d'image échantillonnée aléatoirement, nous proposons d'introduire un patch de surface incomplet $\tilde{\mathbf{P}}_\mathbf{x}$, pour tout $\mathbf{x} \in \mathbb{R}^3$, en projetant les points $\mathbf{x}_j \in N(\mathbf{x}) \subset \mathcal{X}$ sur le plan du patch (Figure 5.12). Une discrétisation du plan permet de définir les valeurs du patch ainsi qu'un opérateur d'échantillonnage S. Les valeurs sont obtenues par une combinaison pondérée des $m(\mathbf{x}_j)$ et l'opérateur d'échantillonnage comme la somme des poids reçus dans les cases correspondantes.

2. Pour tout point $\mathbf{x}_i \in \mathcal{X}$, la valeur centrale du patch correspond à la valeur du signal bruité $m(\mathbf{x}_i)$. Par conséquent, chaque $\mathbf{x}_i \in \mathcal{X}$ est associé à un patch amélioré $\mathbf{P}'_{\mathbf{x}_i}$. En pratique, notre arbre est donc construit par rapport à l'ensemble des patchs approchés $\{\mathbf{P}_{\mathbf{x}_i}\}$ et appris sur l'ensemble des patchs améliorés $\{\mathbf{P}'_{\mathbf{x}_i}\}$.

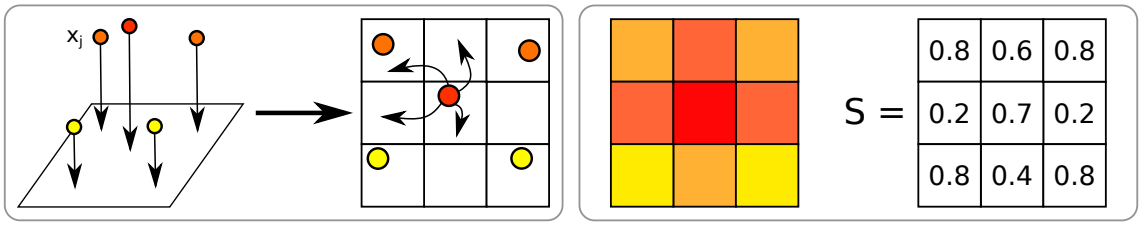


FIGURE 5.12: Définition d'un patch de surface incomplet. Les points $\mathbf{x}_j$ sont projetés sur le plan du patch. À partir d'une discrétisation du plan, nous obtenons les valeurs du patch et l'opérateur d'échantillonnage S.

Par conséquent, les patchs incomplets peuvent être reconstruits en considérant une approche similaire à celle proposée en Section 5.2.3. Une fois l'ensemble des $\{\tilde{\mathbf{P}}_{\mathbf{x}_i}\}$ reconstruits par l'Équation 5.10 un nuage est densifié en ajoutant des points en fonction des valeurs du patch reconstruit (Figure 5.12).

CHAPITRE 6

Conclusion

À partir du débruitage non local de nuages de points, nous avons mis en évidence le besoin de structures dédiées aux traitements non-locaux des données de dimension deux et supérieure, ainsi que la possibilité de généraliser les méthodes existantes de restauration non-locale 2D et 3D. Pour pouvoir y répondre nous avons introduit les contributions suivantes :

Surface de point non locale Les opérateurs de surface de points (PSS) conventionnels permettent de définir une surface à partir d'un voisinage local. Malheureusement face aux conditions réelles de capture, ce point de vue devient rapidement insuffisant pour définir une surface correcte. Pour résoudre ce problème, nous avons proposé une nouvelle définition qui étend toutes les définitions PSS locales précédentes en exploitant le caractère autosimilaire du nuage de points. Pour ce faire, nous utilisons un opérateur PSS défini à une échelle grossière pour définir une surface grossière (servant de support à notre méthode) et un champ scalaire résiduel contenant les caractéristiques bruitées de la surface. Notre surface non locale est définie en ajoutant à cette surface grossière le champ scalaire de déplacement résiduel débruité par la méthode des moyennes non locales. La mesure de similarité entre deux parties du nuage de points est définie par l'intermédiaire de patchs de surfaces dont les valeurs représentent les variations entre la surface grossière et une surface calculée par un PSS à échelle fine. Nous avons démontré que notre définition est capable d'améliorer la qualité des surfaces PSS locales en augmentant le rapport signal/bruit. Néanmoins ces résultats sont très dépendants du caractère autosimilaire du nuage à traiter. Dans le cas peu probable, d'absence d'autosimilarité, notre opérateur non local obtiendra des résultats identiques à ceux de l'opérateur PSS sous-jacent. En pratique, l'estimation de notre opérateur non local est accélérée par GPU permettant une utilisation plus interactive.

Arbre de covariances Avec l'augmentation de la quantité d'information, peu de méthodes à l'heure actuelle exploitent réellement l'information contenue dans les grandes bases de données ou au prix d'une quantité de mémoire et de calculs beaucoup trop importante. Comme solution, nous introduisons le CovTree, une nouvelle structure de données capable d'apprendre les distributions locales d'un grand ensemble d'échantillons en les représentant par des gaussiennes anisotropes. Cette structure est définie par l'in-

termédiaire de trois principales étapes. Tout d'abord, nous construisons un arbre binaire à partir d'un ensemble d'échantillons qui partitionne l'espace en fonction de la position des différents échantillons. Ensuite, nous apprenons les distributions statistiques locales de chaque nœud en propageant l'ensemble des échantillons à travers l'arbre en fonction de leur position et ajoutant une contribution pondérée des points au noyau de chaque nœud de l'arbre traversé. Finalement, pour toute requête, notre arbre modélise la distribution locales des données apprises. Pour permettre une utilisation générique, nous définissons une version généralisée de notre structure par l'intermédiaire de deux domaines : spatial correspondant au descripteur du point et des attributs correspondant à signal associé. Par l'intermédiaire d'un schéma de requête adaptatif, il devient possible d'interroger à n'importe quelle échelle pour des temps sensiblement équivalents.

Restauration collaboratives Les filtres collaboratifs utilisent l'information commune à des ensembles de patchs similaires afin d'améliorer la qualité de la restauration. Malheureusement, il est coûteux et difficile d'analyser l'information contenue dans une grande base de données. Ces approches sont donc généralement simplifiées et limitées à un ensemble de points réduit. Pour résoudre ce problème, différents ensemble de données sont appris par l'intermédiaire du CovTree. Nous introduisons différentes applications de débruitages et de reconstructions de données 2D et 3D. Par rapport aux approches précédentes, il devient possible d'apprendre un *a priori* à partir d'une grande base d'échantillons de données non détériorées. Nous utilisons cet *a priori* pour améliorer les performances des méthodes de restauration. Dans cette partie, nous effectuons une analyse qualitative des performances de notre CovTree qui permet de valider son utilisation.

En conclusion, nos différentes contributions permettent de s'attaquer aux problématiques soulevées par les évolutions récentes des méthodes de restaurations non locales et de l'augmentation de la quantité de données. Cette thèse s'inscrit donc au cœur de cette évolution en proposant quelques éléments de réponse à l'un des enjeux technologiques des années 2010, à savoir : la capture, transmission et traitements de masses de séquences temporelles de données 2D et 3D.

CHAPITRE 7

Perspectives

Dans cette thèse, nous avons travaillé sur différents thèmes découlant de la restauration par méthodes non locales d'un ensemble de données 2D et 3D. Au-delà des contributions techniques proposées, un certain nombre de nouvelles pistes plus générales que celle abordées en Section 3.4 et 4.5 peut découler de ce travail de recherche.

La définition des NLPSS suppose que le nuage de points 3D est corrompu par un bruit additif gaussien 3D. Cette supposition est indispensable pour pouvoir utiliser les moyennes non locales. Néanmoins, à l'heure actuelle, il n'existe (à notre connaissance) aucune analyse du modèle statistique du bruit introduit par les scanners 3D car il est trop dépendant du type de capteurs utilisé. Il serait donc intéressant de définir les NLPSS pour des bruits différents comme c'est proposé en traitement des images dans [DDT09]. Une telle idée nécessiterait de définir des mesures de similarité spécifiques. Les travaux de Deledalle et coll. [DDT12] et Desolneux et coll. [DD13] donnent quelques pistes le deuxième étant particulièrement pertinent car il pourrait permettre de rendre le NLPSS plus robuste aux points aberrants. Néanmoins, le changement du type de bruit entrainerait nécessairement un problème dans l'estimation du repère de normalisation des patchs.

L'amélioration des systèmes de captures par agrégation des données 3D du type Kinect Fusion [IKH+11] pourrait être une autre piste de recherche intéressante. Une telle application nécessiterait de coupler notre CovTree afin de cumuler les données dans le temps, avec un système d'acquisition de données 3D en temps réel (tels que *Microsoft Kinect* par exemple). Une telle idée nécessite néanmoins de faire évoluer notre CovTree temporellement (Section 4.5).

Bibliographie

[AA04a] A. Adamson and M. Alexa. Approximating bounded, nonorientable surfaces from points. In *Proceedings of Shape Modeling Applications*, pages 243–252. IEEE, 2004.

[AA04b] M. Alexa and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *Symposium on Point-Based Graphics*, pages 149–155. Citeseer, 2004.

[AA09] M. Alexa and A. Adamson. Interpolatory point set surfaces - convexity and hermite data. *ACM Transactions on Graphics (TOG)*, 28(2) :20, 2009.

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proceedings of the Conference on Visualization*, pages 21–28. IEEE, October 2001.

[ABD10] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum (CGF)*, 29(2) :753–762, 2010.

[AGDL09] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian kd-trees for fast high-dimensional filtering. *ACM Transactions on Graphics (TOG)*, 28(3) :21, July 2009.

[AK04] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Transactions on Graphics (TOG)*, 23(3) :264–270, August 2004.

[Ale05] S. K. Alexander. Multiscale methods in image modelling and image processing. Ph.d. thesis, University of Waterloo, Ontario, Canada, 2005.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM),*, 45(6) :891–923, 1998.

[APG07] N. Azzabou, N. Paragios, and F. Guichard. Image denoising based on adapted dictionary computation. In *International Conference on Image Processing (ICIP)*, volume 3, pages 109–112. IEEE, 2007.

[AW05] S. P. Awate and R. T. Whitaker. Higher-order image statistics for unsupervised, information-theoretic, adaptive, image filtering. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 44–51. IEEE, 2005.

[AW06] S. P. Awate and R. T. Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 28(3) :364–376, 2006.

[Bar02] D. Barash. Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 24(6) :844–847, 2002.

[BBB+10] T. Beeler, B. Bickel, P. Beardsley, B. Sumner, and M. Gross. High-quality single-shot capture of facial geometry. *ACM Transactions on Graphics (TOG)*, 29(3) :40, 2010.

[BBH08] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[BC07] T. Brox and D. Cremers. Iterated nonlocal means for texture restoration. In *Scale Space and Variational Methods in Computer Vision*, pages 13–24. Springer, 2007.

[BCM05] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65. IEEE, 2005.

[BCM08] A. Buades, B. Coll, and J.-M. Morel. Nonlocal image and movie denoising. *Springer International Journal of Computer Vision (IJCV)*, 76(2) :123–139, 2008.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :509–517, 1975.

[BHB+11] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, and M. Gross. High-quality passive facial performance capture using anchor frames. *ACM Transactions on Graphics (TOG)*, 30 :75, August 2011.

[BKC08] T. Brox, O. Kleinschmidt, and D. Cremers. Efficient nonlocal means for denoising of textural patterns. *IEEE Transactions on Image Processing*, 17(7) :1083–1092, 2008.

[BM05] E. P. Bennett and L. McMillan. Video enhancement using per-pixel virtual exposures. *ACM Transactions on Graphics (TOG)*, 24(3) :845–852, 2005.

[BO05] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5) :405–451, 2005.

[Bov05] A. C. Bovik. *Handbook of image and video processing.* Academic Press, 2005.

[BPD06] S. Bae, S. Paris, and F. Durand. Two-scale tone management for photographic look. *ACM Transactions on Graphics (TOG)*, 25(3) :637–645, 2006.

[BRR11] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo - stereo matching with slanted support windows. In *British Machine Vision Conference (BMVC)*, volume 11, pages 1–11, 2011.

[BSFG09] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch : a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)*, 28(3) :24, 2009.

[CFGS12] X. Chen, T. Funkhouser, D. B. Goldman, and E. Shechtman. Non-parametric texture transfer using meshmatch. *Adobe Technical Report*, November 2012.

[CM02] D. Comaniciu and P. Meer. Mean shift : A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 24(5) :603–619, 2002.

[CPD07] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics (TOG)*, 26(3) :103, 2007.

[CPT03] A. Criminisi, P. Pérez, and K. Toyama. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–721. IEEE, 2003.

[CPT04] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9) :1200–1212, 2004.

[CWL+08] Z-Q. Cheng, Y-Z. Wang, B. Li, K. Xu, G. Dang, and S-Y. Jin. A survey of methods for moving least squares surfaces. In *Proceedings of conference on Point-Based Graphics*, pages 9–23. IEEE, 2008.

[DAG10] V. Duval, J.-F. Aujol, and Y. Gousseau. On the parameter choice for the non-local means. *Technical Report hal-00468856*, 2010.

[DAL+11] J. Digne, N. Audfray, C. Lartigue, C. Mehdi-Souzani, and J-M. Morel. Farman Institute 3D Point Sets - High Precision 3D Data Sets. *Image Processing On Line (IPOL)*, 2011.

[DD02] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics (TOG)*, 21(3) :257–266, 2002.

[DD13] J. Delon and A. Desolneux. A patch-based approach for removing impulse or mixed gaussian-impulse noise. *Journal on Imaging Sciences (SIAM)*, 6(2) :1140–1174, 2013.

[DDS12] C.-A. Deledalle, V. Duval, and J. Salmon. Non-local methods with shape-adaptive patches (NLM-SAP). *Journal of Mathematical Imaging and Vision*, 43(2) :103–120, 2012.

[DDT09] C.-A. Deledalle, L. Denis, and F. Tupin. Iterative weighted maximum likelihood denoising with probabilistic patch-based weights. *IEEE Transactions on Image Processing*, 18(12) :2661–2672, 2009.

[DDT12] C.-A. Deledalle, L. Denis, and F. Tupin. How to compare noisy patches ? patch similarity beyond gaussian noise. *Springer International Journal of Computer Vision (IJCV)*, 99(1) :86–102, 2012.

[DFKE07] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8) :2080–2095, 2007.

[DFKE08] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image restoration by sparse 3D transform-domain collaborative filtering. In *Proceedings of Electronic Imaging 2008*, page 681207, 2008.

[DFKE09] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. BM3D Image Denoising with Shape-Adaptive Principal Component Analysis. In *Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, 2009.

[DG10] J.-E. Deschaud and F. Goulette. Point cloud non local denoising using local surface descriptor similarity. *IAPRS*, 2010.

[Dig12] J. Digne. Similarity based filtering of point clouds. *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[DM11] J. Digne and J.-M. Morel. Numerical analysis of differential operators on raw point clouds. *Numerische Mathematik*, pages 1–35, 2011.

[DMSL11] J. Digne, J.-M. Morel, C.-M. Souzani, and C. Lartigue. Scale space meshing of raw data point sets. *Computer Graphics Forum (CGF)*, 30(6) :1630–1642, 2011.

[Don95] D. L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3) :613–627, 1995.

[EL99] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of International Conference on Computer Vision (ICCV)*, volume 2, pages 1033–1038. IEEE, 1999.

[FAR07] R. Fattal, M. Agrawala, and S. Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics (TOG)*, 26(3) :51, 2007.

[FCOS05] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (TOG)*, 24(3) :544–552, 2005.

[FDCO03] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics (TOG)*, 22(3) :950–953, 2003.

[FKEA04] A. Foi, V. Katkovnik, K. Egiazarian, and J. Astola. A novel anisotropic local polynomial estimator based on directional multiscale optimizations ". In *International conference of IMA*, pages 79–82, 2004.

[FKN80] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. *ACM SIGGRAPH Computer Graphics*, 14(3) :124–133, 1980.

[GG07] G. Guennebaud and M. H. Gross. Algebraic point set surfaces. *ACM Transactions on Graphics (TOG)*, 26(3) :23, 2007.

[GGG08] G. Guennebaud, Marcel Germann, and M. H. Gross. Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum (CGF)*, 27(2) :653–662, 2008.

[GLPP08] B. Goossens, Q. Luong, A. Pizurica, and W Philips. An improved non-local denoising algorithm. In *International workshop on Local and Non-Local Approximation in Image Processing (LNLA)*, pages 143–156, 2008.

[GO11] E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Transactions on Graphics (TOG)*, 30(4) :69, 2011.

[GO12] E. S. L. Gastal and M. M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics (TOG)*, 31(4) :33, 2012.

[HDD+94] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. *Surface reconstruction from unorganized points.* PhD thesis, University of Washington, 1994.

[HS12] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *Computer Vision and Pattern Recognition (CVPR)*, pages 111–118. IEEE, 2012.

[HST10] K. He, J. Sun, and X. Tang. Guided image filtering. In *European Conference on Computer Vision (ECCV)*, pages 1–14. Springer, 2010.

[IKH+11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, and A. Davison. KinectFusion : real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[JCSX09] Z. Ji, Q. Chen, Q.-S. Sun, and D.-S. Xia. A moment-based nonlocal-means algorithm for image denoising. *Information Processing Letters*, 109(23) :1238–1244, 2009.

[JDD03] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics (TOG)*, 22 :943–949, July 2003.

[JDZ04] T. R. Jones, F. Durand, and M. Zwicker. Normal improvement for point rendering. *IEEE Computer Graphics and Applications*, 24(4) :53–56, 2004.

[KA11] S. Korman and S. Avidan. Coherency sensitive hashing. In *International Conference on Computer Vision (ICCV)*, pages 1607–1614. IEEE, 2011.

[KB06] C. Kervrann and J. Boulanger. Optimal spatial adaptation for patch-based image denoising. *IEEE Transactions on Image Processing*, 15(10) :2866–2878, 2006.

[KBH06] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of Symposium on Geometry processing (SGP)*, 2006.

[KCLU07] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (TOG)*, 26(3) :96, 2007.

[KEA02] V. Katkovnik, K. Egiazarian, and J. Astola. Adaptive window size image de-noising based on intersection of confidence intervals (ICI) rule. *Journal of Mathematical Imaging and Vision*, 16(3) :223–235, 2002.

[KFEA04] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola. Directional varying scale approximations for anisotropic signal processing. In *Proceedings in European Signal Processing Conference (EUSIPCO)*, pages 101–104. Citeseer, 2004.

[LBM13] M. Lebrun, A. Buades, and J-M. Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. *Image Processing On Line (IPOL)*, 2013 :1–42, 2013.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH*, pages 163–169, 1987.

[LCBM12] M. Lebrun, M. Colom, A. Buades, and J-M. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21(1) :475–576, 2012.

[Lee83] J.-S. Lee. Digital image smoothing and the sigma filter. *Computer Vision, Graphics, and Image Processing*, 24(2) :255–269, 1983.

[Lev98] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67 :1517–1531, 1998.

[Lev03] D. Levin. Mesh-independent surface interpolation. *Geometric Modeling for Scientifc Visualization(2003)*, pages 1517–1523, 2003.

[LN11] A. Levin and B. Nadler. Natural image denoising : Optimality and inherent bounds. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2833–2840. IEEE, 2011.

[LPC+00] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, and J. Ginsberg. The digital michelangelo project : 3d scanning of large statues. In *Proceedings of Computer Graphics and Interactive Techniques*, pages 131–144. ACM, 2000.

[MBP+09] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *International Conference on Computer Vision*, pages 2272–2279. IEEE, 2009.

[MCCL+08] J. V. Manjón, J. Carbonell-Caballero, J. J. Lull, G. García-Martí, L. Martí-Bonmatí, and M. Robles. MRI denoising using non-local means. *Medical image analysis*, 12(4) :514–523, 2008.

[ML09] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP)*, pages 331–340. INSTICC Press, 2009.

[ML12] M. Muja and D. G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV)*, pages 404–410. IEEE, 2012.

[MNB12] A. Maleki, M. Narayan, and R. G. Baraniuk. Anisotropic nonlocal means denoising. *Applied and Computational Harmonic Analysis*, 2012.

[MRS12] S. Morigi, M. Rucci, and F. Sgallari. Nonlocal surface fairing. *Scale Space and Variational Methods in Computer Vision*, pages 38–49, 2012.

[Nem00] A. Nemirovski. Topics in non-parametric. *Ecole d'Ete de Probabilites de Saint-Flour*, 28 :85, 2000.

[NFP+13] A. Newson, M. Fradet, P. Pérez, A. Almansa, and Y. Gousseau. Towards fast, generic video inpainting. HAL Technical Report 00838927, June 2013.

[OA12] I. Olonetsky and S. Avidan. Treecann-kd tree coherence approximate nearest neighbor algorithm. In *European Conference on Computer Vision (ECCV)*, pages 602–615. Springer, 2012.

[ÖGG09] A. C. Öztireli, G. Guennebaud, and M. H. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum (CGF)*, 28(2) :493–501, 2009.

[PD06] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *European Conference on Computer Vision (ECCV)*, pages 568–580. Springer, 2006.

[PD09] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision (IJCV)*, 81(1) :24–52, 2009.

[PS00] J. Polzehl and V. G. Spokoiny. Adaptive weights smoothing with applications to image restoration. *Journal of the Royal Statistical Society (Statistical Methodology)*, 62(2) :335–354, 2000.

[RDK13] G. Rosman, A. Dubrovina, and R. Kimmel. Patch-collaborative spectral point-cloud denoising. In *Computer Graphics Forum (CGF)*. Wiley Online Library, 2013.

[ROF92] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D : Nonlinear Phenomena*, 60(1) :259–268, 1992.

[Sal10a] J. Salmon. *Agrégation d'estimateurs et méthodes à patch pour le débruitage d'images numériques*. PhD thesis, Université Paris-Diderot-Paris VII, 2010.

[Sal10b] J. Salmon. On two parameters for denoising with non-local means. *IEEE Signal Processing Letters*, 17(3) :269–272, 2010.

[She68] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the ACM national conference*, pages 517–524. ACM, 1968.

[SS12] J. Salmon and Y. Strozecki. Patch reprojections for non-local methods. *Signal Processing*, 92(2) :477–489, 2012.

[Tas08] T. Tasdizen. Principal components for non-local means image denoising. In *IEEE International Conference on Image Processing (ICIP)*, pages 1728–1731, 2008.

[Tas09] T. Tasdizen. Principal neighborhood dictionaries for nonlocal means image denoising. *IEEE Transactions on Image Processing*, 18(12) :2649–2660, 2009.

[TM98] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of International Conference on Computer Vision (ICCV)*, page 839. IEEE Computer Society, 1998.

[VB11] G. Vialaneix and T. Boubekeur. Sbl mesh filter : fast separable approximation of bilateral mesh filtering. In *SIGGRAPH Talks*, page 24. ACM, 2011.

[VDVK09] D. Van De Ville and M. Kocher. SURE-based non-local means. *IEEE Signal Processing Letters*, 16(11) :973–976, 2009.

[WCZ+08] R. F. Wang, W. Z. Chen, S. Y. Zhang, Y. Zhang, and X. Z. Ye. Similarity-based denoising of point-sampled surfaces. *Journal of Zhejiang University-Science*, 9(6) :807–815, 2008.

[WGY+06] J. Wang, Y. Guo, Y. Ying, Y. Liu, and Q. Peng. Fast non-local algorithm for image denoising. In *IEEE International Conference on Image Processing (ICIP)*, pages 1429–1432, 2006.

[YBS06] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Smoothing by example : Mesh denoising by averaging with similarity based weights. In *In Proceedings of IEEE International Conference on Shape Modeling and Applications*, pages 38–44, 2006.

[YSM10] G. Yu, G. Sapiro, and S. Mallat. Image modeling and enhancement via structured sparse model selection. In *IEEE International Conference on Image Processing (ICIP)*, pages 1641–1644, 2010.

[YSM12] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators : from gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5) :2481–2499, 2012.

[YY85] L. P. Yaroslavsky and L. P. Yaroslavskij. Digital picture processing. an introduction. *Springer Seriesin Information Sciences*, 1, 1985.

[ZDW08] S. Zimmer, S. Didas, and J. Weickert. A rotationally invariant block matching strategy improving image denoising with non-local means. In *Proceedings of International Workshop on Local and Non-Local Approximation in Image Processing*, pages 135–142, 2008.

[ZDZS10] L. Zhang, W. Dong, D. Zhang, and G. Shi. Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition*, 43(4) :1531–1549, 2010.

[ZW11] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 479–486. IEEE, 2011.

Méthodes et structures non locales pour la restauration d'images et de surfaces 3D

Thierry GUILLEMOT

Résumé : Durant ces dernières années, les technologies d'acquisition numériques n'ont cessé de se perfectionner, permettant d'obtenir des données d'une qualité toujours plus fine. Néanmoins, le signal acquis reste corrompu par des défauts qui ne peuvent être corrigés matériellement et nécessitent l'utilisation de méthodes de restauration adaptées.

Jusqu'au milieu des années 2000, ces approches s'appuyaient uniquement sur un traitement ocal du signal détérioré. Avec l'amélioration des performances de calcul, le support du filtre a pu être étendu à l'ensemble des données acquises en exploitant leur caractère autosimilaire. Ces approches non locales ont principalement été utilisées pour restaurer des données régulières et structurées telles que des images. Mais dans le cas extrême de données irrégulières et non structurées comme les nuages de points 3D, leur adaptation est peu étudiée à l'heure actuelle. Avec l'augmentation de la quantité de données échangées sur les réseaux de communication, de nouvelles méthodes non locales ont récemment été proposées. Elles utilisent un modèle a priori extrait à partir de grands ensembles d'échantillons pour améliorer la qualité de la restauration. Néanmoins, ce type de méthode reste actuellement trop coûteux en temps et en mémoire.

Dans cette thèse, nous proposons, tout d'abord, d'étendre les méthodes non locales aux nuages de points 3D, en définissant une surface de points capable d'exploiter leur caractère autosimilaire. Nous introduisons ensuite une nouvelle structure de données, le CovTree, flexible et générique, capable d'apprendre les distributions d'un grand ensemble d'échantillons avec une capacité de mémoire limitée. Finalement, nous généralisons les méthodes de restauration collaboratives appliquées aux données 2D et 3D, en utilisant notre CovTree pour apprendre un modèle statistique a priori à partir d'un grand ensemble de données.

Mots-clefs : non local, restauration, structure de données, débruitages, surface, nuage de points, filtre collaboratif

Abstract : In recent years, digital technologies allowing to acquire real world objects or scenes have been significantly improved in order to obtain high quality datasets. However, the acquired signal is corrupted by defects which can not be rectified materially and require the use of adapted restoration methods.

Until the middle 2000s, these approaches were only based on a local process applyed on the damaged signal. With the improvement of computing performance, the neighborhood used by the filter has been extended to the entire acquired dataset by exploiting their self-similar nature. These non-local approaches have mainly been used to restore regular and structured data such as images. But in the extreme case of irregular and unstructured data as 3D point sets, their adaptation is few investigated at this time. With the increase amount of exchanged data over the communication networks, new non-local methods have recently been proposed. These can improve the quality of the restoration by using an a priori model extracted from large data sets. However, this kind of method is time and memory consuming.

In this thesis, we first propose to extend the non-local methods for 3D point sets by defining a surface of points which exploits their self-similar of the point cloud. We then introduce a new flexible and generic data structure, called the CovTree, allowing to learn the distribution of a large set of samples with a limited memory capacity. Finally, we generalize collaborative restoration methods applied to 2D and 3D data by using our CovTree to learn a statistical a priori model from a large dataset.

Keywords : non local, restauration, data structure, denoising, surface, points set, collaborative filter

TELECOM ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY