Document de synthèse

Présenté par

Bruno Galerne

en vue de l'obtention de

l'**Habilitation à Diriger des Recherches**
de l'Université Paris Descartes
Spécialité : Mathématiques Appliquées

Modèles stochastiques pour la synthèse de textures

Soutenue le 7 novembre 2016 devant le jury composé de :

Rapporteurs :	Jean-François Aujol	-	Université de Bordeaux
	Hermine Biermé	-	Université de Poitiers
	Victor Ostromoukhov	-	Université de Lyon
Président du jury :	Anne Estrade	-	Université Paris Descartes
Examinateurs :	Pierre Chainais	-	Ecole centrale de Lille
	Yann Gousseau	-	Télécom ParisTech
	Jean-Michel Morel	-	Ecole Normale Supérieure de Cachan

Remerciements

Je remercie tout d'abord Jean-François Aujol, Hermine Biermé et Victor Ostromoukhov pour le temps qu'ils ont consacré à la lecture de ce manuscrit et à la rédaction de leur rapport. Je suis également honoré qu'ils se soient tous les trois déplacés pour assister à la soutenance. Je remercie également Pierre Chainais et Anne Estrade qui me font l'honneur de participer au jury de soutenance. Enfin, je suis extrêmement ravi de la présence de Yann Gousseau et Jean-Michel Morel, mes deux directeurs de thèse, au sein de ce jury de soutenance. C'est une nouvelle occasion pour moi de leur exprimer toute ma gratitude pour m'avoir transmis leur goût pour la recherche. Depuis la fin de ma thèse, nous avons continué à avoir des échanges réguliers, toujours éclairants pour moi.

J'ai eu la chance de pouvoir travailler avec de nombreux chercheurs de diverses cultures scientifiques et cela a toujours été une expérience autant enrichissante qu'agréable. Je profite de ces quelques lignes pour remercier tous mes collaborateurs : Thibaut Briand, Julie Delon, George Drettakis, Noura Faraj, Thomas Hurtut, Raphaël Lachièze-Rey, Ares Lagae, Pierre-Edouard Landes, Arthur Leclaire, Sylvain Lefebvre, Lionel Moisan, Alasdair Newson, Lara Raad, Julien Rabin, Jonathan Vacher. J'ai une pensée particulière pour Arthur Leclaire qui restera le premier doctorant pour lequel j'ai participé à l'encadrement de thèse. Vu ses qualités scientifiques et humaines, cette première expérience d'encadrement ne pouvait pas être plus positive. Enfin, je remercie également les tout jeunes doctorants Jorge Gutierrez et Claire Launay ainsi que leurs co-encadrants de thèse, Agnès Desolneux pour Claire et Thomas Hurtut et Julien Rabin pour Jorge. Je suis honoré de travailler avec eux et je suis très enthousiaste pour les trois années à venir.

Tout ce travail a été réalisé dans des conditions plus qu'agréable au laboratoire MAP5 dont je remercie chaleureusement tous les membres pour la précieuse bonne entente qui règne au quotidien. Je remercie particulièrement les deux directrices que j'ai connu depuis mon arrivée, Annie Raoult et Fabienne Comte, pour leur disponibilité et leur soutien. Je n'oublie pas Marie-Hélène Gbaguidi dont l'efficacité et la bonne humeur rendent presque agréable les tâches administratives. Je remercie plus largement tous les membres de l'UFR math-info, et notamment sa directrice Christine Graffigne dont l'appui m'a permis d'obtenir un CRCT d'un an l'an dernier, essentiel pour la préparation de ce manuscrit.

Enfin, je remercie Amélie pour son soutien et ses encouragements au fil des ans, ainsi que notre petite Manon pour son enthousiasme quotidien.

Table des matières

1 Introduction **1**
1.1 Synthèse de textures 1
1.2 Organisation du manuscrit 3
1.2.1 Shot noise discret pour la synthèse et l'inpainting de microtextures 3
1.2.2 Synthèse par l'exemple de textures procédurales de type shot noise 3
1.2.3 Modèles de type germe-grain : Modélisation d'images et variations 3
1.3 Liste de publications 4

2 Shot noise discret pour la synthèse et l'inpainting de microtextures **7**
2.1 Shot noise discret et bruit à phase aléatoire : Définitions 7
2.1.1 Définition du spot noise discret asymptotique 7
2.1.2 Définition du bruit à phase aléatoire 11
2.1.3 Différences théoriques entre le SNDA et le BPA 11
2.2 Shot noise discret et bruit à phase aléatoire pour la synthèse de textures 12
2.2.1 SNDA et BPA des images couleurs 13
2.2.2 Suppression des artefact dus à la non-périodicité 16
2.2.3 Synthèse sur un domaine plus grand 18
2.2.4 Résultats 18
2.2.5 Conclusion et discussion 20
2.3 Un texton orienté synthèse pour la synthèse de microtextures 22
2.3.1 SND et SNDA sur $\mathbb{Z}^2$ 23
2.3.2 Algorithme pour le calcul d'un texton orienté synthèse 25
2.3.3 Résultats 25
2.4 Inpainting de microtextures par simulation gaussienne conditionnelle 29
2.4.1 Simulation gaussienne conditionnelle et estimateur de krigeage 30
2.4.2 Algorithme d'inpainting gaussien 31

3 Synthèse par l'exemple de textures procédurales de type shot noise **35**
3.1 Introduction 35
3.2 Simulation parallèle et à la demande d'un processus de Poisson . . . 37
3.3 Synthèse par l'exemple de textures procédurales de type *Gabor noise* 40
3.3.1 Le modèle Gabor noise 40
3.3.2 Gabor noise by example 42
3.4 *Texton noise*, un bruit procédural minimaliste pour la synthèse de textures gaussiennes 49
3.4.1 Un shot noise à noyau fixe 49

3.4.2 Algorithme de calcul du texton 51
3.4.3 Filtrage antialiasing pour le texton noise 55
3.4.4 Mélange progressif de textures 55
3.4.5 Conclusion et discussion sur la synthèse de textures gaussiennes 59

4 Modèles de type germe-grain : Modélisation d'images et variations 63
4.1 Modèles germe-grain pour la modélisation et la génération d'images 66
4.1.1 Modèle feuilles mortes transparentes 66
4.1.2 Un modèle pour la synthèse de textures vectorielles prenant en compte les formes des objets 71
4.1.3 Modélisation stochastique du grain des pellicules argentiques et algorithme de rendu à une résolution arbitraire 75
4.2 Variation des champs et des ensembles aléatoires 83
4.2.1 Champs aléatoires à variation bornée et calcul de leur variation totale moyenne 84
4.2.2 Le cas des ensembles aléatoires 87
4.2.3 Variation totale moyenne des modèles germe-grain 91

5 Quelques perspectives de recherches 95
5.1 Transport optimal pour la synthèse de textures 95
5.1.1 Imposition de multi-histogrammes par transport optimal numérique dans les méthodes par patchs pour la synthèse de textures solides 96
5.1.2 Transport optimal numérique semi-discret par méthode Monte-Carlo 96
5.2 Processus ponctuels déterminantaux pour la synthèse d'images . . . 97
5.2.1 Contexte 97
5.2.2 Processus déterminantaux dans le cadre des images 98
5.2.3 Shot noise basé sur des processus déterminantaux 98
5.2.4 Processus ponctuels déterminantaux pour l'échantillonnage stochastique 99
5.3 Synthèse de textures de bois 99

Chapitre 1

Introduction

Ce rapport résume l'intégralité de mes travaux de recherche. Mes travaux portent essentiellement sur l'étude et l'utilisation de modèles stochastiques pour la synthèse de textures.

1.1 Synthèse de textures

La synthèse de textures est un problème important en traitement d'images et en *computer graphics* (on préférera ce terme anglais à "infographie" pour référer à la discipline scientifique relevant du graphisme). Le problème de la synthèse de texture par l'exemple est le suivant : A partir d'une image de texture (une image de bois, pierre, tissu, etc.), comment générer une nouvelle image de texture qui soit à la fois perçues comme étant similaire à l'image d'entrée tout en étant différente pixels à pixels (et si possible de taille plus grande) ? Autrement dit, il s'agit de simuler une image qui puisse être considérée comme étant une autre région du même matériau que l'image d'entrée, comme illustré par la Figure 1.1. Les méthodes de synthèse de textures sont présentes dans la plupart des applications de l'industrie du *computer graphics* (films d'animations, jeux vidéos, réalité virtuelle).

D'un point de vue méthodologique, on peut séparer les méthodes de synthèse de textures en deux familles. Il y a d'une part des méthodes reposant sur des modélisations stochastiques ou des contraintes statistiques et d'autre part des méthodes par patchs cherchant à constituer des textures par agencements de voisinages de la texture d'entrée. Les méthodes de synthèse par patchs ont été beaucoup développées dans les années 2000 et il a été montré qu'elles permettaient d'obtenir des résultats souvent impressionnants. Cependant, ces méthodes restent mal comprises, notamment en ce qui concerne le réglage de leurs différents paramètres et également le manque de garantie sur la qualité du résultat.

A l'opposée, les méthodes que nous avons développées et que nous résumons dans ce manuscrit reposent sur des modèles stochastiques bien définis et se situent donc dans la première famille de méthodes. Cette modélisation stochastique permet d'apporter des garanties théoriques et pratiques sur la qualité des résultats de synthèse, au prix de ne pouvoir reproduire qu'un sous-ensemble assez limité de textures. J'ai plus spécifiquement travaillé sur des modèles shot noise et leur limite gaussienne pour la synthèse de textures par l'exemple. Nous avons utilisé ces modèles pour la synthèse dans deux contextes différents : la synthèse d'images de textures (faites de pixels) et la synthèse de textures procédurales, c'est-à-dire de programmes exécutables sur cartes graphiques permettant d'évaluer une texture en

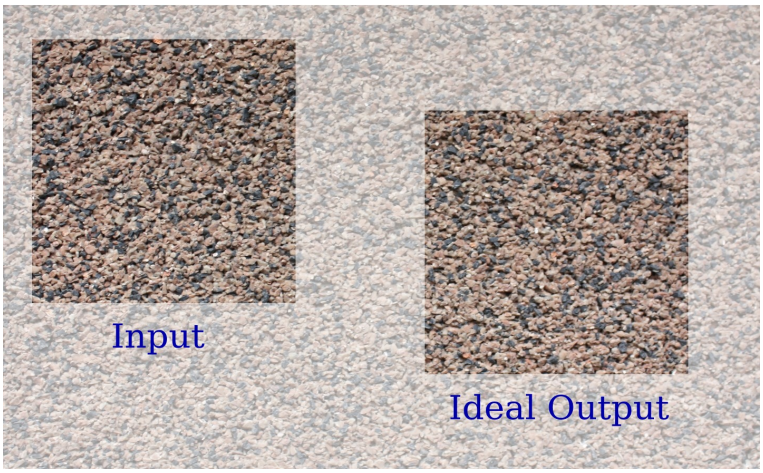


FIGURE 1.1 – But de la synthèse de textures par l'exemple : Simuler une image de texture qui soit perçue comme un autre morceau du matériau auquel appartient l'image d'entrée.

tout point du plan. Bien que les modèles mathématiques utilisés pour ces travaux soient très proches voir parfois identiques, il me semble important de différencier ces deux approches de la synthèse de textures. En effet, la synthèse de textures procédurales doit répondre à de nombreuses contraintes très spécifiques au *computer graphics* et à la programmation sur carte graphique (qu'il m'a parfois été difficile d'appréhender...). En conséquence, j'ai choisi de consacrer un chapitre spécifique de ce manuscrit à mes travaux sur la synthèse procédurale.

Avant de décrire plus en détail le contenu de ce manuscrit, je tiens à aborder le problème de la reproductibilité des résultats de mes travaux. J'ai effectué mon doctorat au CMLA au moment de la création du journal *Image Processing On Line* ce qui m'a grandement sensibilisé à cette question essentielle (je suis aujourd'hui co-auteur de trois articles dans ce journal [J2, J7, J10], et deux autres articles sont en préparation avancée), d'autant plus pour un problème comme la synthèse de textures pour lequel il est primordial de pouvoir comparer ses résultats aux méthodes concurrentes puisque seule la qualité visuelle du résultat compte. Or il est très souvent difficile d'implémenter un algorithme à partir d'un article de recherche où tous les détails algorithmiques ne peuvent généralement pas être abordés, sans parler du réglage des paramètres. Un des principaux avantages des méthodes que j'ai pu développé avec mes différents collaborateurs est qu'elles ne reposent pas sur des jeux de paramètres dont le réglage est critique pour les résultats. Afin d'en attester, les codes sources de tous mes travaux sur la synthèse de textures sont disponibles en ligne, avec des liens depuis ma page web (mis à part ceux de l'article [J6] car le format de fichiers vectoriels svg étant très complexe, chaque expérience demandait un traitement manuel des entrées afin de supprimer les informations non géométriques).

1.2 Organisation du manuscrit

1.2.1 Shot noise discret pour la synthèse et l'inpainting de micro-textures

Ce premier chapitre présente en détail mes travaux sur la synthèse de textures discrètes basés sur le modèle shot noise discret et sa limite gaussienne. Dans un cadre fini périodique, on discute d'abord des liens entre ce modèle shot noise asymptotique et la randomisation de la phase de Fourier d'une image. On apporte ensuite des compléments techniques afin d'utiliser ces deux procédés aléatoires pour la synthèse de textures par l'exemple. Au final on montre expérimentalement que toutes les textures stationnaires sont convenablement reproduites par ces algorithmes.

On présente ensuite un algorithme pour calculer un "texton orienté synthèse", c'est-à-dire une imagette permettant de résumer le contenu fréquentielle d'une texture gaussienne et avec laquelle la synthèse par shot noise discret à faible intensité produit des résultats visuellement proche de la limite gaussienne. Enfin, on montre que le modèle de texture gaussienne peut être également exploité pour obtenir un algorithme d'inpainting spécifique basé sur une simulation conditionnelle du champ gaussien.

1.2.2 Synthèse par l'exemple de textures procédurales de type shot noise

Ce deuxième chapitre porte sur le problème de bruit procédural par l'exemple. Comme nos deux contributions sur ce sujet reposent sur des champs aléatoires de type shot noise définis sur le plan, on commence par rappeler en détail l'algorithme de simulation parallèle et à la demande d'un processus de Poisson.

On décrit ensuite un algorithme permettant d'approcher une texture gaussienne quelconque par une texture procédurale de type *Gabor noise* [Lagae et al., 2009], c'est-à-dire un shot noise où l'on somme des noyaux de Gabor ayant différents paramètres.

Après avoir discuté des limitations de cette approche, on présente une nouvelle méthode beaucoup plus performante baptisée *texton noise* qui propose de calculer une fonction noyau unique qui détermine à elle seule le contenu fréquentiel de la texture procédurale. Afin de pouvoir exploiter pleinement les modules des cartes graphiques, on impose à cette fonction d'être une interpolation bilinéaire, ce qui permet entre autre de proposer une méthode de filtrage anti-aliasing efficace à un moindre coût.

1.2.3 Modèles de type germe-grain : Modélisation d'images et variations

Les modèles shot noise noise reposent sur un processus ponctuel de Poisson pour définir un champ aléatoire. Plus généralement, en géométrie stochastique on parle de modèles de type germe-grain pour tout modèle de champ aléatoire construit à partir

de formes géométriques colorées (les grains) placées sur des points aléatoirement répartis (les grains). Ces modèles germe-grain sont présents dans tous mes travaux.

Nous présentons d'abord trois articles reposant sur l'étude de modèles germes-grains particuliers. Nous évoquons tout d'abord notre étude du modèle feuilles mortes transparentes, un modèle germe-grain construit comme superposition d'une infinité de formes transparentes. On décrit ensuite un algorithme pour la synthèse de textures vectorielles reposant sur un processus de Gibbs constitué de formes géométriques disjointes. Enfin, on évoque des travaux récents où l'on utilise un modèle booléen non homogène afin de reproduire un effet de grain argentique sur des images numériques dépourvues de grain.

Nous passons ensuite à une étude théorique sur la variation totale moyenne des champs aléatoires et en particulier des modèles germe-grain. Pour cela on définit et caractérise les champs aléatoires à variation bornée. On obtient alors une expression simple pour la variation totale moyenne d'un champ aléatoire à accroissements stationnaires. Ces résultats généraux permettent d'étendre grandement des résultats connus en morphologie mathématique concernant le périmètre moyen des ensembles aléatoires. En outre, en s'appuyant sur les bonnes propriétés topologiques de l'ensemble des ensembles de périmètre fini, on établit une nouvelle caractérisation du problème de réalisabilité d'une fonction covariogramme. Enfin, les résultats sur les champs à variation bornée et les ensembles aléatoires de périmètre fini permettent d'obtenir des expressions de la variation totale moyenne de tous les modèles germe-grain classiques. En particulier, pour ces modèles seuls la surface moyenne et le périmètre moyen des grains interviennent dans l'expression de la variation totale moyenne.

1.3 Liste de publications

Je produis ci-dessous la liste de mes publications classées en quatre catégories :

— Articles publiés ou acceptés dans des journaux internationaux,

— Actes de conférences avec comité de relecture,

— Thèse,

— Articles soumis.

A noter que deux de mes articles correspondent à des proceedings de conférences de graphisme (SIGGRAPH 2012 et EGSR 2013). Dans cette communauté où les conférences sont très sélectives, les articles acceptés pour la conférence sont des articles à part entière dans les journaux associés (Transactions on Graphics et Computer Graphics Forum).

Toutes mes publications sont disponibles sur ma page web :
`http://www.math-info.univ-paris5.fr/~bgalerne/publications.html`

Articles de journaux internationaux

[J1] B. Galerne, Y. Gousseau, and J.-M. Morel. Random phase textures : Theory and synthesis. *IEEE Trans. Image Process.*, 20(1) :257 – 267, 2011.

[J2] B. Galerne, Y. Gousseau, and J.-M. Morel. Micro-texture synthesis by phase randomization. *Image Processing On Line*, 2011.

[J3] B. Galerne. Computation of the perimeter of measurable sets via their covariogram. Applications to random sets. *Image Anal. Stereol.*, 30 :39–51, 2011.

[J4] B. Galerne and Y. Gousseau. The transparent dead leaves process. *Adv. Appl. Probab.*, 44(1) :1–20, 2012.

[J5] B. Galerne, A. Lagae, S. Lefebvre, and G. Drettakis. Gabor noise by example. *ACM Trans. Graph.*, 31(4) :73 :1–73 :9, jul 2012.

[J6] P.-E. Landes, B. Galerne, and T. Hurtut. A shape-aware model for discrete texture synthesis. *Computer Graphics Forum (Proc. EGSR)*, 32, 2013.

[J7] T. Briand, J. Vacher, B. Galerne, and J. Rabin. The heeger & bergen pyramid based texture synthesis algorithm. *Image Processing On Line*, 4 :276–299, 2014.

[J8] B. Galerne and R. Lachièze-Rey. Random measurable sets and covariogram realisability problems. *Adv. Appl. Probab.*, 47 :611–639, 2015.

[J9] B. Galerne. Random fields of bounded variation and computation of their variation intensity. *Adv. Appl. Probab.*, in press, 2016.

[J10] L. Raad and B. Galerne. Efros and Freeman image quilting algorithm for texture synthesis. *Image Processing On Line*, (accepted).

[J11] B. Galerne, A. Leclaire, and L. Moisan. Texton noise. *Computer Graphics Forum*, (accepted).

Actes de Conférences avec comité de lecture

[C12] B. Galerne. Variation totale moyenne de modèles stochastiques de texture. In *GRETSI*, Bordeaux, 2011.

[C13] B. Galerne, A. Leclaire, and L. Moisan. A texton for fast and flexible Gaussian texture synthesis. In *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, pages 1686–1690, 2014.

[C14] B. Galerne, A. Leclaire, and L. Moisan. Microtexture inpainting through Gaussian conditional simulation. In *Proceedings of IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP 2016)*, 2016.

Thèse de doctorat

[T15] B. Galerne. *Modèles d'image aléatoires et synthèse de texture.* PhD thesis, Ecole Normale Supérieur de Cachan, Déc. 2010.

Articles soumis

[S16] A. Newson, B. Galerne, and J. Delon. Stochastic modeling and resolution-free rendering of film grain. submitted to Computer Graphics Forum, 2016.

Chapitre 2

Shot noise discret pour la synthèse et l'inpainting de microtextures

Les deux premières sections de ce chapitre résument mon article de thèse [J1] co-écrit avec Y. Gousseau et J.-M. Morel et portant sur la synthèse de microtextures à l'aide des modèles de shot noise discret asymptotique et de bruit à phase aléatoire.

On décrit ensuite deux travaux plus récents effectués en collaboration avec A. Leclaire et L. Moisan. On résume tout d'abord l'acte de conférence [C13] qui traite de la définition d'un texton pour la synthèse rapide de ces modèles de microtextures. L'extension de cette notion de texton au cadre continu des textures procédurales sera présentée au chapitre 3. Enfin on décrit les résultats récents présentés à ICASSP 2016 [C14] sur l'inpainting de microtextures par simulation conditionnelle de champs gaussiens. Ce travail fera l'objet d'un article long actuellement en préparation.

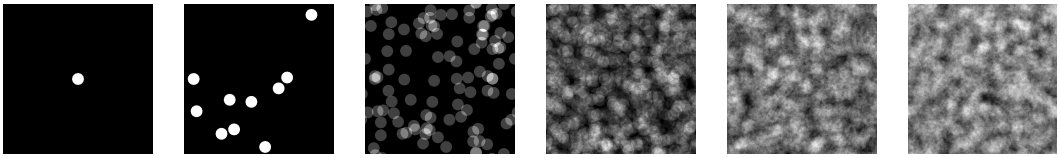
2.1 Shot noise discret et bruit à phase aléatoire : Définitions

Les deux modèles que nous allons définir, à savoir le *spot noise* discret et le bruit à phase aléatoire, ont été introduits en graphisme par van Wijk [van Wijk, 1991]. Le terme *spot noise* a été introduit comme un équivalent spatial du *shot noise*, considéré par l'auteur comme étant temporel. En réalité le terme *shot noise* est couramment utilisé pour des champs aléatoires définis en dimension supérieure à deux. On préférera donc le terme shot noise pour toute étude générique de ces champs. On conserve en revanche le terme spot noise lorsque l'on réfère au modèle de synthèse de textures étudié ci-après.

2.1.1 Définition du spot noise discret asymptotique

Cadre et notations On considère des images $f \in \mathbb{R}^{M\times N}$ de taille $M \times N$ et dont les indices sont $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$. Une image est étendue à tout $\mathbb{Z}^2$ par périodicité.

Un spot noise discret d'ordre n et de spot $h \in \mathbb{R}^{M\times N}$ est une image aléatoire obtenue en sommant n translations i.i.d. du spot h. Plus formellement on a la définition suivante.



Spot h | $n = 10$ | $n = 100$ | $n = 1000$ | $n = 10000$ | $n = 100000$

FIGURE 2.1 – Réalisations du spot noise discret pour différentes valeurs de n dans le cas où un spot h étant l'indicatrice d'un disque. Pour chaque image, les niveaux de gris extrémaux sont ramenés à 0 et 255.

Définition 2.1 (Spot noise discret (SND))**.** *Le spot noise discret d'ordre $n \in \mathbb{N}$ et de spot $h \in \mathbb{R}^{M\times N}$ est l'image aléatoire*

$$f_n(x) = \sum_{k=1}^{n} h(x - X_k), \quad x \in \Omega,$$

où les (X_k) sont des vecteurs aléatoires indépendants et uniformément distribués dans Ω.

Le spot noise discret (SND) est un modèle stationnaire, au sens où sa distribution est invariante par translation périodique. La Figure 2.1 illustre ce modèle pour différentes valeurs de n dans le cas où h est l'indicatrice d'un disque. On peut observer que la séquence de textures (f_n) ainsi obtenue semble converger vers une texture limite. Cet objet limite est formellement obtenu en appliquant le théorème central limite à la somme d'images i.i.d. définissant le spot noise discret. Ainsi, une normalisation de la suite (f_n) converge en loi vers une image aléatoire gaussienne ayant pour moyenne $\mathbb{E}(f_1)$ et pour covariance $\mathrm{Cov}(f_1)$. Comme f_1 est une translation aléatoire du spot h, on obtient immédiatement que $\mathbb{E}(f_1)$ est une image constante égale à la moyenne arithmétique m de h,

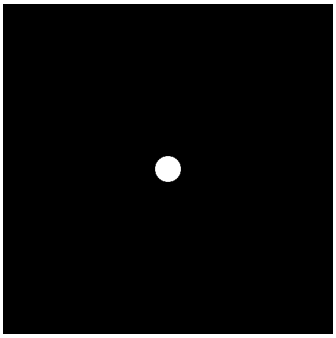
$$\mathbb{E}(f_1(x)) = \frac{1}{MN} \sum_{y\in\Omega} h(y) =: m.$$

De même, la covariance $\mathrm{Cov}(f_1)$ correspond à l'autocorrélation $C_h = \frac{1}{MN}(h - m) * (\tilde{h} - m)$ de l'image h ($*$ désigne bien sûr le produit de convolution), à savoir

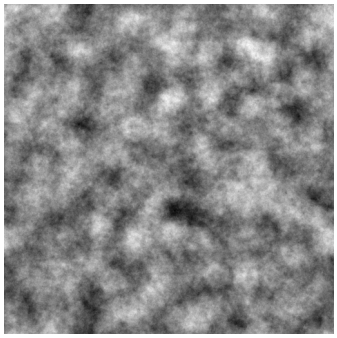
$$\mathrm{Cov}(f_1)(x, y) = \frac{1}{MN} \sum_{z\in\Omega} (h(x+z)-m)(h(y+z)-m) = \frac{1}{MN}(h-m)*(\tilde{h}-m)(x-y) =: C_h(x-y),$$

où $\tilde{h}(x) = h(-x)$ est l'image symétrique de h. On peut alors définir la limite de la suite des SND comme étant l'image gaussienne (au sens vecteur gaussien) de moyenne m et de covariance C_h.

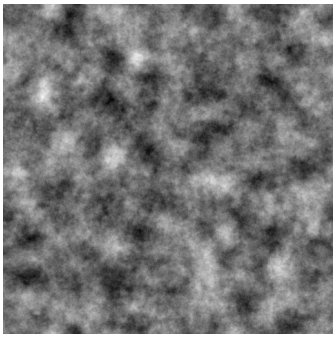
Définition 2.2 (Spot noise discret asymptotique (SNDA))**.** *Le spot noise discret asymptotique associé au spot $h \in \mathbb{R}^{M\times N}$ est l'image aléatoire de distribution gaussienne stationnaire $\mathcal{N}(m, C_h)$ ayant pour moyenne l'image constante à la moyenne*



Spot h



SND avec $n = 100000$



SNDA

FIGURE 2.2 – Comparaison visuelle d'une réalisation d'un spot noise discret d'ordre élevé et d'une spot noise discret asymptotique pour le spot de la Figure 2.1.

arithmétique m de h et pour covariance l'autocorrélation $C_h = \frac{1}{MN}(h-m)*(\tilde{h}-m)$ de h. On note $\mathrm{SNDA}_\Omega(h) = \mathcal{N}(m, C_h)$ *cette distribution gaussienne.*

La définition du SNDA est naturelle d'un point de vue théorique. On peut par ailleurs la justifier par une toute autre approche : le champ gaussien stationnaire $\mathcal{N}(m, C_h)$ associé à h correspond au champ gaussien stationnaire réalisant le maximum de vraisemblance pour l'observation h (voir [Xia et al., 2014] et [Leclaire, 2015, Section 3.1.1]). Toutefois cette définition n'apporte pas de construction explicite du champ gaussien limite $Y \sim \mathrm{SNDA}_\Omega(h)$, notamment pour la simulation. Comme on va le voir, il se trouve que ce champ gaussien Y se simule très facilement par convolution avec un bruit blanc.

Théorème 2.1 (Simulation d'un SNDA)**.** *Soit $h \in \mathbb{R}^{M\times N}$ un spot de moyenne m et X un vecteur de bruit blanc gaussien (les valeurs $X(x)$ des pixels sont indépendantes de distribution $\mathcal{N}(0,1)$). On note*

$$t_h(x) = \frac{1}{\sqrt{MN}}(h-m), \quad x \in \Omega,$$

la version centrée-normalisée de h. Alors,

$$Y = m + t_h * X \sim \mathrm{SNDA}_\Omega(h).$$

Ainsi, le vecteur gaussien $\mathrm{SNDA}_\Omega(h)$ *se simule en convolant un bruit blanc gaussien par une version centrée-normalisée du spot h.*

La simulation d'un SNDA se réduit donc au coup de générer un bruit blanc gaussien et de le convoler avec l'image t_h, soit une complexité en $\mathcal{O}(MN\log(MN))$. en utilisant une convolution discrète calculée par transformée de Fourier discrète rapide (FFT, pour *Fast Fourier Transform*). La simulation d'un SNDA est comparée à un SND avec un ordre n grand à la Figure 2.2.

Remarque (Sur la simulation de champs gaussiens discrets)**.** La simulation exacte ou approchée de champs gaussiens est une problématique centrale de ce chapitre et du chapitre suivant. Le résultat du Théorème 2.1 propose une solution simple et très efficace pour simuler les champs SNDA. On peut en fait interpréter ce résultat comme étant un cas trivial de la simulation de champs gaussiens par méthode de *circulant embedding* (que l'on pourrait traduire par imbrication circulante) car l'on exploite implicitement la structure de matrice circulante par blocs de la matrice de covariance du SNDA.

En général, la simulation de vecteurs gaussiens s'effectue en utilisant une factorisation de la matrice de covariance. Si A est une matrice et X est un vecteur gaussien de coordonnées i.i.d. de loi $\mathcal{N}(0,1)$, alors le vecteur de combinaisons linéaires AX est un vecteur gaussien de moyenne nulle et de matrice de covariance AA^T. Ainsi, étant donnée une matrice de covariance C, il suffit de déterminer une matrice A telle que $C = AA^T$ (par factorisation de Cholesky par exemple) et de calculer le produit AX pour générer un vecteur gaussien de loi $\mathcal{N}(0, C)$. Toutefois cet algorithme standard de simulation de vecteurs gaussiens n'est pas utilisable en pratique pour des images, puisqu'il fait intervenir la matrice de covariance de taille $MN \times MN$, que le calcul d'une factorisation de Cholesky est de complexité $\mathcal{O}((MN)^3)$, et que rien que la multiplication matrice-vecteur AX est de complexité $\mathcal{O}((MN)^2)$.

Cette méthode générale ne suppose toutefois aucune structure sur les matrices de covariance. Dans le cas de champs stationnaires, les matrices de covariances ne dépendent que de la différence $x-y$ des coordonnées ce qui leur confère une structure de matrice de Toeplitz (*i.e.* une matrice constante sur ces diagonales) en 1D et Toeplitz par blocs de Toeplitz en 2D. Une approche efficace et élégante consiste alors à étendre ces matrices de covariances pour les rendre circulantes, c'est la méthode du *circulant embedding* introduite indépendamment par [Wood and Chan, 1997] et [Dietrich and Newsam, 1997]. En effet les matrices circulantes sont les matrices qui sont diagonalisées par la matrice de la FFT, soit encore les matrices de convolution discrète périodiques. Le calcul d'une factorisation de type $C = AA^T$ et le produit matrice vecteur AX peut alors être calculé implicitement par FFT.

Dans le cas du SNDA, la matrice de covariance

$$C(x,y) = C_h(x-y) = \frac{1}{MN}(h-m) * (\tilde{h}-m)(x-y) = t_h * \tilde{t}_h(x-y)$$

est une matrice de covariance circulante dont une factorisation $C = AA^T$ est directement donnée par l'opérateur de convolution par le spot normalisé t_h. Ainsi l'imbrication circulante est trivial. Pour finir, remarquons que si l'on applique l'algorithme naturel pour le calcul d'une factorisation $C = AA^T$ alors on obtient la convolution par l'image dont la FFT est la racine carré de la FFT de $C_h(x)$, à savoir $\hat{C}_h(\xi) = \left|\hat{h}(\xi)\right|^2$, $\xi \in \Omega$. Cette image a le même module que h et une phase nulle, c'est le texton canonique associé à h introduit par [Desolneux et al., 2012, Desolneux et al., 2016].

2.1.2 Définition du bruit à phase aléatoire

Le bruit à phase aléatoire associé à un spot h est l'image obtenue en remplaçant la phase de la transformée de Fourier discrète (TFD) de h par une phase aléatoire. La notion de phase aléatoire est assez intuitive, c'est une image d'angles aléatoires qui sont indépendants entre eux modulo la contrainte de symétrie impaire que l'on doit imposer à la phase pour que l'image aléatoire obtenue reste à valeurs réelles. Plus précisément on adopte la définition suivante :

Définition 2.3 (Phase aléatoire)**.** *Un champ aléatoire $\theta : \Omega \to \mathbb{R}$ est une* phase aléatoire *si*

- *θ est impaire : $\theta(-x) = -\theta(x) \mod 2\pi$, $x \in \Omega$.*
- *La loi de marginale de $\theta(x)$ est soit uniforme sur $]-\pi, \pi]$ si $x \neq -x$ soit uniforme sur $\{0, \pi\}$ si $x = -x$ (ce cas ne concerne que les un, deux, ou quatre points où la TFD est nécessairement réelle selon la parité de M et N).*
- *Pour tout sous-ensemble $S \subset \Omega$ ne contenant pas de paire de points symétriques et distincts, les variables aléatoires $\{\theta(x),\ x \in S\}$ sont indépendantes.*

Définition 2.4 (Bruit à phase aléatoire (BPA))**.** *Un* bruit à phase aléatoire *associé à une image de spot $h \in \mathbb{R}^{M \times N}$ est une image aléatoire Z telle que la DFT $\hat{Z}$ de Z ait pour expression*

$$\hat{Z}(\xi) = \left|\hat{h}(\xi)\right| e^{i\theta(\xi)},$$

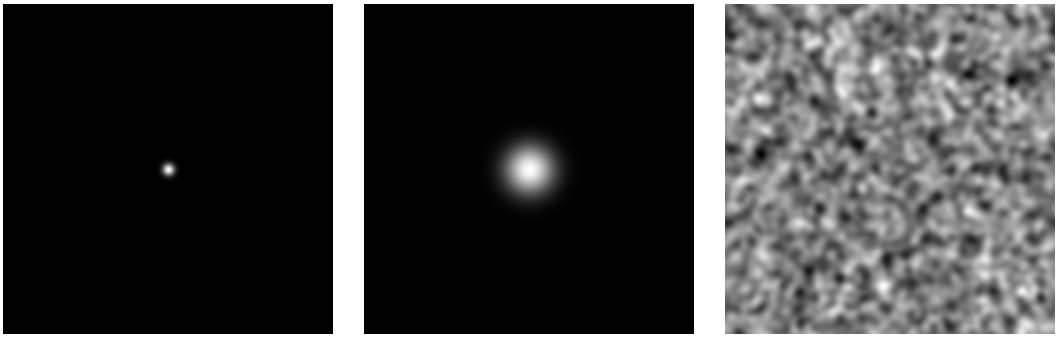
où θ est une phase aléatoire au sens de la Définition 2.3.

Remarque (Mise à zéro des phases ou décalage des phases)**.** On peut également définir le BPA comme $\hat{Z}(\xi) = \hat{h}(\xi)e^{i\theta(\xi)} = \left|\hat{h}(\xi)\right| e^{i(\varphi(\xi)+\theta(\xi))}$ où φ est la phase de la TFD de h. En effet, si θ est une phase aléatoire alors la somme $\varphi + \theta$ est également une phase aléatoire. On verra par la suite que c'est cette deuxième approche consistant à décaler les phases plutôt que les mettre à zéro qui permet de définir convenablement le BPA d'une image couleur.

La Figure 2.3 illustre le modèle BPA dans le cas où h est un spot gaussien isotrope. Par construction le BPA a strictement le même module de Fourier que le spot h. Les deux images h et Z ont donc la même fonction d'autocorrélation. En conséquence, la taille caractéristique ainsi que la régularité du spot correspondent à celle de la texture comme on peut le voir à la Figure 2.3.

2.1.3 Différences théoriques entre le SNDA et le BPA

Contrairement à ce qui est écrit dans [van Wijk, 1991], les deux champs SNDA et BPA associés à un même spot h ne sont pas identiques. Pour comparer la différence entre ces deux distributions, le plus simple est de considérer la TFD des deux champs Y et Z. $Y = m + t_h * X$ est la convolution entre t_h et le bruit blanc gaussien X. La loi de la TFD de $\hat{X}$ est la suivante : le module $\left|\hat{X}\right|$ est un bruit blanc de Rayleigh, la phase de $\hat{X}$ est une phase aléatoire, et ce module et cette phase sont indépendants.



Spot h | Module $\left|\hat{h}\right|$ | BPA

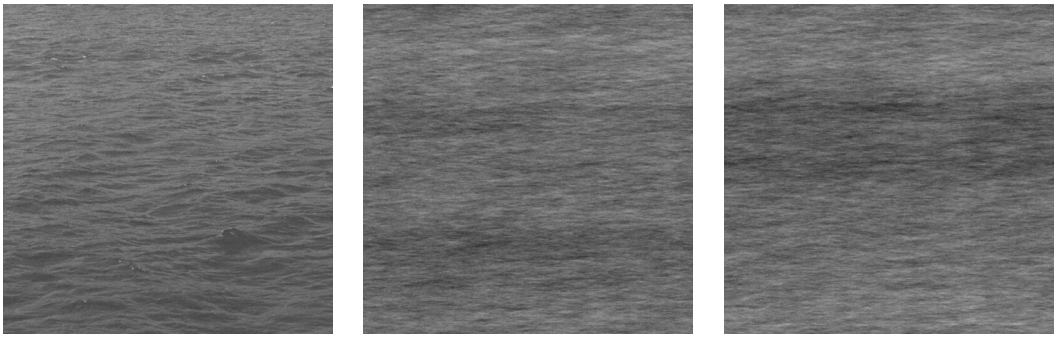
FIGURE 2.3 – Bruit à phase aléatoire d'un spot gaussien. De gauche à droite : Spot gaussien h, module de le TFD $\hat{h}$ du spot h, et réalisation du BPA associé à h.

Ainsi, comme le BPA, $\hat{Y}$ a une phase aléatoire. En revanche les deux processus différent au niveau de leur module. Le module du BPA est par construction celui du spot h alors que le module du SNDA est le module de h multiplié par un bruit blanc de Rayleigh.

Cette différence théorique étant établie, une des contributions de l'article [J1] est de démontrer expérimentalement que la différence entre les deux modèles SNDA et BPA n'est pas perceptible pour des images dont le spectre n'est pas dégénéré (ce qui est toujours le cas pour les images naturelles). Cette observation a d'ailleurs contribué à l'introduction de nouveaux indices de qualité image par G. Blanchet et L. Moisan [Blanchet and Moisan, 2012] et poursuivi par A. Leclaire et L. Moisan [Leclaire and Moisan, 2015] reposant sur la cohérence de phase. En effet, dans un premier travail novateur [Blanchet et al., 2008], les auteurs avaient introduit un indice de qualité image reposant sur l'écart entre la variation totale d'une image et la variation totale de son modèle BPA associé. Toutefois, ce modèle ne permettant pas de faire de calcul explicite de moyenne et variance de la variation totale, le calcul de l'indice reposait nécessairement sur de longues simulations par Monte Carlo. En remplaçant le modèle BPA par son homologue SNDA, le calcul de la moyenne et de la variance de la variation totale devient explicite [Blanchet and Moisan, 2012] et l'indice peut être calculé rapidement avec quelques FFT. En plus de la justification de non différentiation perceptuelle, on renvoie à l'appendice A de [Leclaire and Moisan, 2015] pour une discussion de l'approximation de la variation totale moyenne du BPA par celle du SNDA.

2.2 Shot noise discret et bruit à phase aléatoire pour la synthèse de textures

La principale contribution de l'article [J1] est d'avoir montré que ces deux modèles très simples SNDA et BPA pouvaient être utilisés pour effectuer de la synthèse



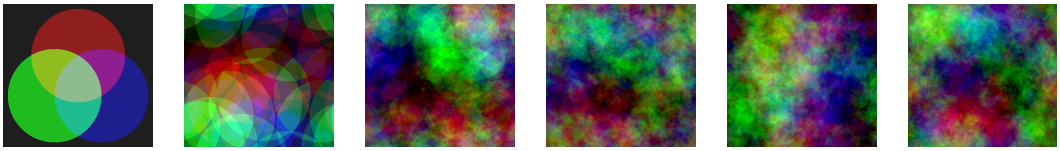
Spot h SNDA BPA

FIGURE 2.4 – SNDA et BPA associés à une image de texture en niveau de gris.

de textures par l'exemple. En effet, leur utilisation dans la littérature se limitait essentiellement à l'aspect génératif [van Wijk, 1991] : Pour tout spot géométrique à support localisé (*e.g.* disque, triangle, gaussienne,...), ces deux modèles génèrent une texture dont l'échelle et la régularité dépendent du spot. Toutefois, il semble difficile de déterminer le spot pour générer du sable, du bois, du tissu, etc. (ce problème est justement l'objet de [C13] et [J11] dont on détaillera la contenu dans la suite). Mais plutôt que chercher à résoudre un problème inverse une démarche simple consiste à utiliser comme spot h une image texturée. Une telle expérience est reproduite à la Figure 2.4. Comme on peut le voir avec cette expérience, les deux modèles SNDA et BPA produisent des résultats semblables. On peut aussi remarquer que les deux textures synthétisées contiennent des bandes horizontales contrastées qui ne correspondent pas au contenu de la texture. Ces artefacts sont dus à des problèmes de bord, l'image de texture naturelle h n'étant pas périodique au sens où elle présente des discontinuités au bord. Afin d'obtenir des algorithmes de synthèse complet, nous allons montrer comment étendre les deux modèles pour les images couleurs et comment éliminer les artefacts dus à la non périodicité des images naturelles.

2.2.1 SNDA et BPA des images couleurs

On considère des images couleurs RGB $\boldsymbol{h} = (h_R, h_G, h_B)^T$, cet espace de couleur linéaire permettant l'addition d'images couleurs. Le modèle SNDA s'étend simplement aux images couleurs. Le SND couleur d'ordre n est obtenu en sommant n translations indépendantes de l'image de spot couleur h. Le théorème centrale limite s'applique à cette séquence d'images aléatoires et le SNDA couleur est le champ gaussien de moyenne la couleur moyenne de $\boldsymbol{h}$ et de matrice de covariance la matrice d'autocorrélation de $\boldsymbol{h}$. La différence ici est que la moyenne $\boldsymbol{m} = (m_R, m_G, m_B) \in \mathbb{R}^3$ est une couleur et que les valeurs de la matrice d'autocorrélation sont des matrices



Spot h | $n = 10$ | $n = 100$ | $n = 1000$ | $n = 10000$ | SNDA

FIGURE 2.5 – Réalisations de SND couleur pour différents ordres et SNDA couleur pour un spot couleur géométrique. On remarque que la palette couleur du SNDA correspond bien à celle de l'image originale.

3×3 représentant les corrélations croisées entre les canaux couleurs, à savoir,

$$C_{\boldsymbol{h}}(x-y) = \sum_{z\in\Omega} (\boldsymbol{h}(x-y+z) - \boldsymbol{m})(\boldsymbol{h}(z) - \boldsymbol{m})^T \in \mathbb{R}^{3\times 3}.$$

En ce qui concerne la simulation, le SNDA couleur s'obtient simplement en convolant chaque canal couleur de l'image $\boldsymbol{t_h} = \frac{1}{\sqrt{MN}}(\boldsymbol{h} - \boldsymbol{m})$ par le même bruit blanc gaussien $X \in \mathbb{R}^{M\times N}$, à savoir

$$\boldsymbol{Y} = \boldsymbol{m} + \frac{1}{\sqrt{MN}} \begin{pmatrix} (h_R - m_R) * X \\ (h_G - m_G) * X \\ (h_B - m_B) * X \end{pmatrix}.$$

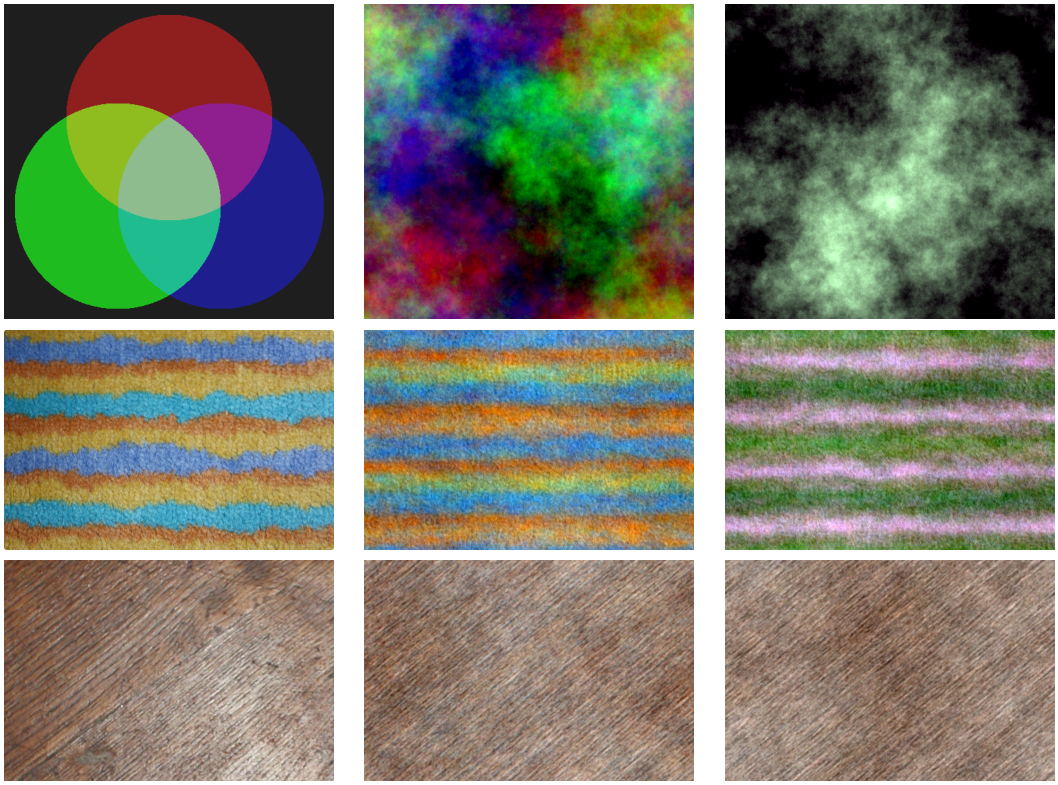
Le fait de convoler par le même bruit blanc est essentiel afin de conserver la corrélation croisée entre les canaux couleurs. Au delà des équations, convoler par le même bruit blanc est tout à fait naturel puisque pour les translations aléatoires les trois canaux sont également translatés conjointement, et donc un SND d'ordre n est obtenu par convolution de chaque canal couleur par la même somme de Dirac aléatoire.

La TFD d'une image couleur se comprend comme la concaténation de la TFD de chaque canal couleur, c'est-à-dire $\hat{\boldsymbol{h}} = \left(\hat{h}_R, \hat{h}_G, \hat{h}_B\right)^T$. La TFD du SNDA couleur $\boldsymbol{Y}$ est donc obtenue en multipliant termes à termes la TFD de l'image couleur $\boldsymbol{h}$ par la TFD du bruit blanc scalaire X. Ainsi, on multiplie le module de chacun des canaux par le même bruit blanc de Rayleigh et on ajoute à la phase de chacun des canaux couleur la même phase aléatoire. Cette dernière observation est essentielle pour définir convenablement le BPA d'une image couleur.

Définition 2.5 (BPA couleur)**.** *Le BPA d'une image couleur $\boldsymbol{h}$ est obtenu en ajoutant à la TFD de chacun des canaux de $\boldsymbol{h}$ la même phase aléatoire θ, c'est-à-dire, en notant $\hat{\boldsymbol{h}} = \left(\hat{h}_R, \hat{h}_G, \hat{h}_B\right)^T = \left(|\hat{h}_R|e^{i\varphi_R}, |\hat{h}_G|e^{i\varphi_G}, |\hat{h}_B|e^{i\varphi_B}\right)^T$ les modules et phases des canaux de $\boldsymbol{h}$, la TFD de $\hat{\boldsymbol{Z}}$ est donnée par*

$$\hat{\boldsymbol{Z}} = \begin{pmatrix} \hat{h}_R e^{i\theta} \\ \hat{h}_G e^{i\theta} \\ \hat{h}_B e^{i\theta} \end{pmatrix} = \begin{pmatrix} |\hat{h}_R| e^{i(\varphi_R+\theta)} \\ |\hat{h}_G| e^{i(\varphi_G+\theta)} \\ |\hat{h}_B| e^{i(\varphi_B+\theta)} \end{pmatrix}$$

où θ est une phase aléatoire.



Spot h | BPA | "Mauvais" BPA

$$\hat{\boldsymbol{h}} = \begin{pmatrix} |\hat{h}_R| e^{i\varphi_R} \\ |\hat{h}_G| e^{i\varphi_G} \\ |\hat{h}_B| e^{i\varphi_B} \end{pmatrix} \qquad \hat{\boldsymbol{Z}} = \begin{pmatrix} |\hat{h}_R| e^{i(\varphi_R+\theta)} \\ |\hat{h}_G| e^{i(\varphi_G+\theta)} \\ |\hat{h}_B| e^{i(\varphi_B+\theta)} \end{pmatrix} \qquad \hat{\boldsymbol{Z}}_M = \begin{pmatrix} |\hat{h}_R| e^{i\theta} \\ |\hat{h}_G| e^{i\theta} \\ |\hat{h}_B| e^{i\theta} \end{pmatrix}$$

FIGURE 2.6 – Réalisations d'un BPA couleur, qui conserve le décalage de phase entre les trois canaux couleurs, et d'un "mauvais" BPA où les décalage de phase sont mis à zéro pour trois images différentes. La dernière ligne de la figure représente explicitement le lien et les différences entre les TFD de chaque image.

Cette définition du BPA couleur semble moins élémentaire que celle du SNDA couleur. Toutefois on vérifie aisément que $\mathrm{Cov}(Z) = C_{\boldsymbol{h}}$. A priori, une solution plus simple, mais incorrecte, consiste à conserver le module de chaque canaux et d'imposer à chacun des canaux une phase aléatoire différente. Toutefois cette définition simpliste produit des images aléatoires dont les canaux couleurs sont indépendants, avec des palettes couleurs denses dans le cube RGB, et donc des couleurs aberrantes sans relation avec la texture de départ. Une autre définition fausse consiste à conserver les modules de chacun des canaux et de leur imposer une même phase aléatoire. Alors, les décalages de phase entre les canaux couleurs sont perdus et la covariance de l'image aléatoire ainsi définie n'est pas l'autocorrélation de $\boldsymbol{h}$. La différence entre le BPA couleur et ce "mauvais" BPA sans décalage de phase est illustrée sur trois exemples dans la Figure 2.6. Sur l'image géométrique, la mise à zéro des décalages de phase revient à aligner les trois disques des canaux RGB, ce qui explique que la palette couleur ne corresponde qu'à la couleur centrale de superposition des trois disques. Le deuxième exemple illustre une situation similaire sur une texture naturelle. En revanche cette situation est rarement rencontrée car les microtextures naturelles sont généralement monochromes, et dans ce cas le "mauvais" BPA produit des résultats semblables ou proches du BPA couleur.

Remarque (Textures gaussiennes et espace couleur)**.** L'expérience montre que les microtextures bien reproduites par les modèles SNDA et BPA sont toutes dominées par les variations de couleur de leur teinte principale, comprise comme la direction principale obtenue par analyse en composante principale (ACP) du nuage de couleur associée à la texture d'entrée. Lors de l'élaboration de l'article [J5], nous avons observé que toutes les textures gaussiennes bien reproduites par le SNDA ou le BPA pouvaient être également obtenues en effectuant trois synthèses indépendantes dans l'espace couleur ACP de l'image, comme proposé dans l'article d'Heeger et Bergen [Heeger and Bergen, 1995] (auquel nous avons consacré un article en une démo en ligne [J7] avec T. Briand, J. Vacher et J. Rabin). En résumé la corrélation entre les canaux couleurs est essentielle dans l'espace couleur RGB mais elle peut être ignorée dans un espace couleur ACP adapté à l'image si on se restreint aux images de textures bien reproduites par le modèle.

Toutefois, dans le cadre discret périodique, il est toujours plus rentable de ne pas faire de synthèse indépendante sur chaque canaux car cela nécessite de générer trois fois plus de variables aléatoires pour les phases aléatoires θ ou les bruits blancs X. Aussi, comme on le verra au chapitre suivant, nous avons récemment développé un nouvel algorithme pour la synthèse de textures procédurales gaussiennes baptisé *texton noise* [J11] qui respecte les corrélations dans l'espace RGB et n'a pas recours à l'ACP.

2.2.2 Suppression des artefact dus à la non-périodicité

Les algorithmes de synthèse SNDA et BPA reposent tous les deux sur la TFD et supposent donc implicitement que les images sont périodiques. Toutefois, les images naturelles ne sont jamais périodiques. Ces problèmes de bord équivalent à

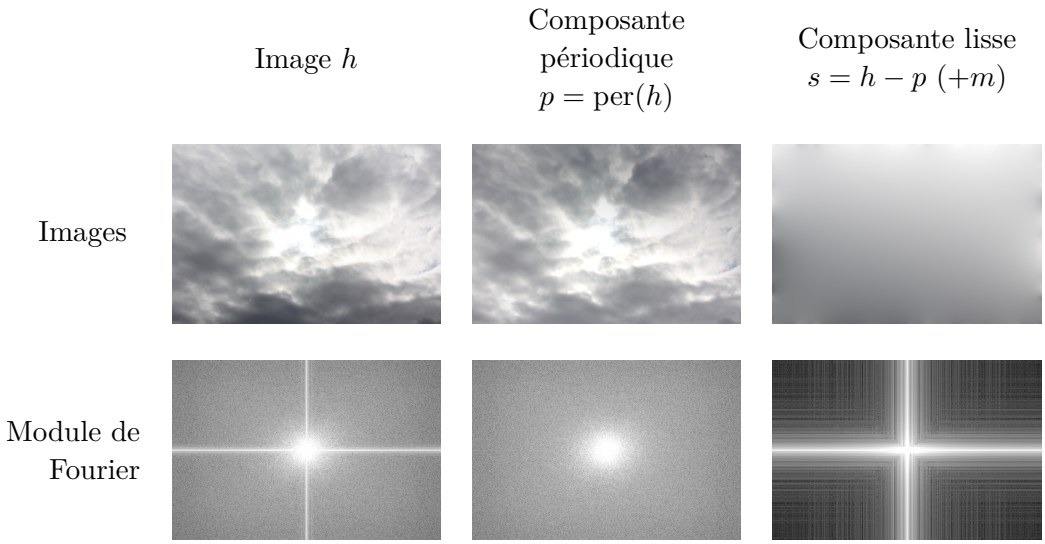


FIGURE 2.7 – Décomposition périodique + lisse [Moisan, 2011] d'une image de texture de nuage présentant des variations d'illumination entre les bords gauche et droit et haut et bas, et modules de Fourier associés. On remarque que le module de Fourier de la texture h comporte une structure en croix qui ne correspond pas au contenu fréquentiel de l'image alors que cette structure en croix n'est pas présente dans le module de Fourier de la composante périodique.

la présence de fortes discontinuités horizontales et verticales dans les images, ce qui résultent en une forte activation des fréquences horizontales et verticales dans la TFD. Au final, après un déphasage indépendant de chacune de leurs fréquences, les textures SNDA et BPA présentent de fortes oscillations verticales et horizontales qui ne correspondent pas au contenu fréquentiel des textures d'entrée.

Par chance lors de l'élaboration de l'article [J1] L. Moisan venait de développer un nouvel algorithme de décomposition d'une image en une composante périodique et une composante lisse (décomposition *periodic + smooth*) [Moisan, 2011]. La composante périodique est définie comme une solution d'un problème de Poisson pour le laplacien périodique, à savoir la composante périodique p du spot h est l'unique image ayant la même moyenne que h et telle que le laplacien périodique de p soit égale au laplacien de h calculé avec des conditions de bord usuelles (les pixels du bord n'ont que trois voisins et les coins seulement deux).

Nous avons alors établi que remplacer l'image d'entrée par sa composante périodique permet d'éliminer les artefacts fréquentiels horizontaux et verticaux. La Figure 2.7 permet d'illustrer la légitimité de cette approche. On remarque dans cet exemple que la non-périodicité de l'image résulte en une forte énergie dans les fréquences horizontales et verticales. En revanche le module de la composante périodique ne contient pas de trace de cette croix fréquentielle et ses fréquences actives correspondent au spectre isotrope que l'on attend pour une texture de nuages. Ainsi

le SNDA et le BPA associés à la composante périodique p ne présenteront pas d'oscillations horizontales ou verticales prédominantes, alors que ce sera nécessairement le cas pour les modèles associés à la texture originale h. La Figure 2.8 montre que les artefacts de non périodicité sont bien éliminés en remplaçant le spot par sa composante périodique.

Sur le plan pratique on a également montré que l'on pouvait calculer la TFD de la composante périodique d'une image en effectuant un seul appel à la FFT grâce à la résolution classique du problème de Poisson périodique (pour les détails on renvoie à l'appendix A de [T15]). Ainsi, le *preprocessing* qui consiste à remplacer h par sa composante périodique p n'augmente pas la complexité des deux algorithmes de synthèse.

2.2.3 Synthèse sur un domaine plus grand

L'article [J1] propose une méthode assez pragmatique pour simuler une texture SNDA ou BPA de taille plus grande que la texture originale. La solution proposée consiste à calculer un spot équivalent qui soit de grande taille et qui contienne en son centre le spot original. Cette approche donne des résultats tout à fait acceptables. Cependant, au vue des développements qui ont suivi avec notamment l'introduction de la notion de texton pour les textures gaussiennes [Desolneux et al., 2012, Desolneux et al., 2016] et surtout le travail de thèse d'Arthur Leclaire [C13], il est bien plus élégant et avantageux de définir des textures SND et SNDA sur le domaine $\mathbb{Z}^2$ tout entier. Nous reviendrons sur ces définitions à la Section 2.3.

2.2.4 Résultats

On présente à la Figure 2.9 des résultats de synthèse des deux modèles SNDA et BPA. Comme discuté à la Section 2.1.3, les deux modèles SNDA et BPA ont tous deux une phase aléatoire. Afin de pouvoir comparer les réalisations des textures générées par ces deux modèles, les résultats de synthèse de la Figure 2.9 utilisent la même phase aléatoire.

Comme déjà évoqué, la première observation de ces expériences est que les deux modèles SNDA et BPA produisent des résultats indiscernables. On a donc deux modèles mathématiques différents, mais d'un point de vue pratique ils correspondent au final à un même algorithme de synthèse de textures. Par la suite on parlera plutôt de textures gaussiennes pour désigner les textures synthétisées ou bien reproduites par les modèles SNDA et BPA.

La deuxième observation concerne la qualité des textures synthétisées. Comme les deux modèles SNDA et BPA sont des modèles de champs aléatoires stationnaires, les textures synthétisées ont un aspect très homogène. Les exemples de la Figure 2.9 sont classés par qualité de ressemblance à la texture d'entrée. On peut observer que les deux premières textures sont bien reproduites par les modèles SNDA et BPA. Dès le troisième exemple on observe la perte de détails isolés. Les exemples suivants

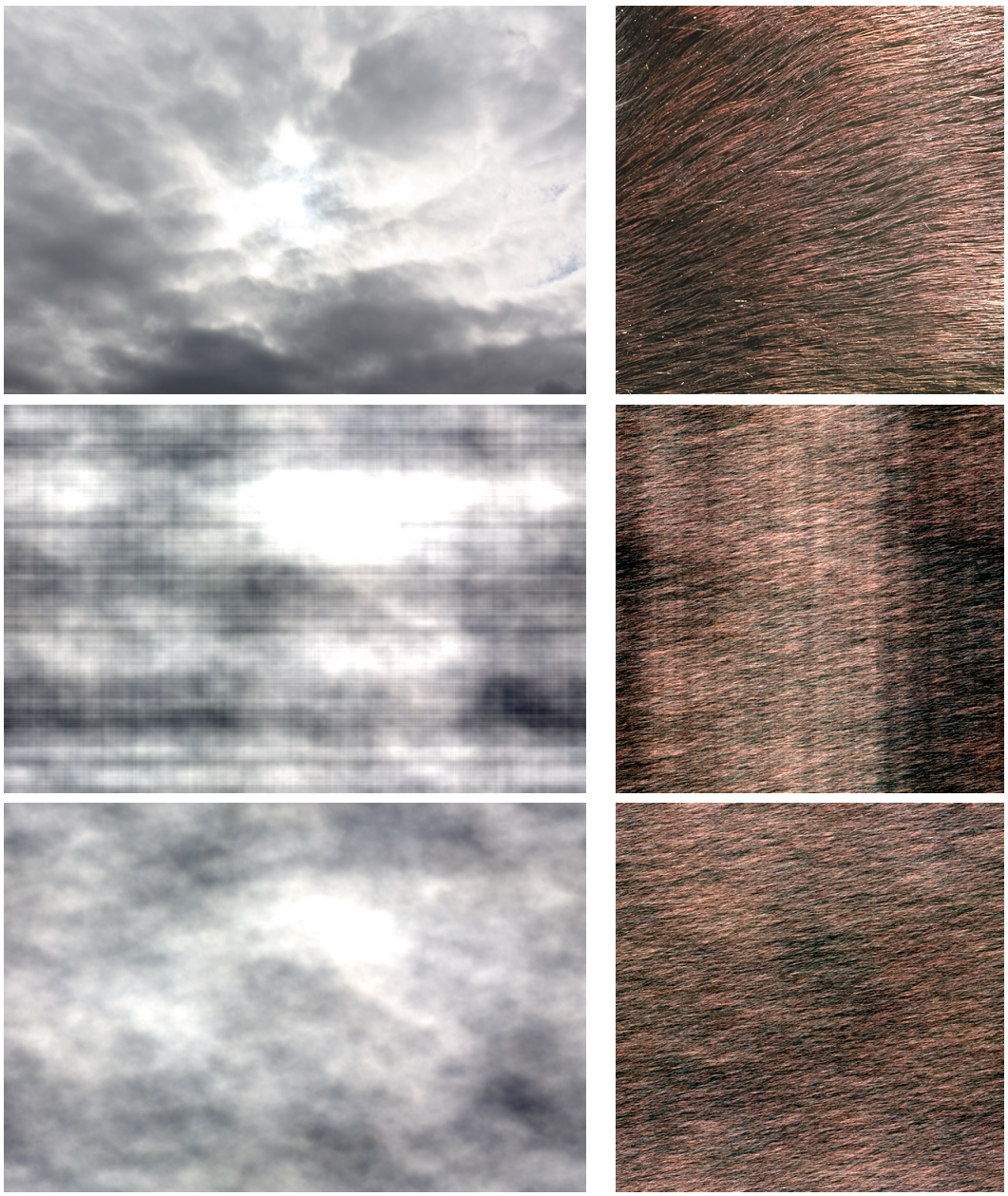


FIGURE 2.8 – De haut en bas : Images originale h, SNDA associé à h, SNDA associé à la composante périodique p de h. On remarque que les textures associées à la composante périodique ne comportent pas d'artefacts horizontaux et verticaux.

sont de plus en plus éloignés de la texture de départ du fait de la perte de structure géométrique dans les textures synthétisées. En effet, les textures gaussiennes ne présentent aucun contours nets. Cette observation empirique peut-être rendue parfaitement rigoureuse : Comme on le verra au Chapitre 4, dans un cadre continu, un champs aléatoire gaussien stationnaire a une variation totale moyenne finie si et seulement si ses réalisations sont dans un espace de Sobolev, et donc avec une partie de sauts nulle. La conséquence pratique de cette observation est que l'algorithme de synthèse proposé n'est pas adapté aux macro-textures pour lesquels on distingue des objets singuliers comme le mur de briques de la Figure 2.9.

2.2.5 Conclusion et discussion

On a tenu à présenter en détails les définitions théoriques et les résultats pratiques de l'article [J1]. L'intérêt principal de cet article est de proposer des modèles et des algorithmes simples et stables pour la synthèse de textures gaussiennes stationnaires. Avec cette approche non paramétrique assez élémentaire, à savoir simuler la texture gaussienne dont la moyenne est la moyenne de l'image et dont la covariance est l'autocorrélation de l'image, on obtient un large panel de textures gaussiennes car le modèle ne limite pas la fonction de covariance contrairement à des approches antérieures comme les modèles autorégressifs [Chellappa and Kashyap, 1985] (Pour plus de détails, on renvoie à la discussion de la Section 2.2.4 de la thèse d'A. Leclaire [Leclaire, 2015]).

Ces résultats ont permis de montrer qu'un certain ensemble de textures naturelles étaient bien reproduites par un modèle gaussien. Ceci a suscité le développement et la poursuite de plusieurs travaux. Comme développé dans la suite de ce chapitre et le chapitre suivant, nous avons principalement dédié nos efforts à l'élaboration d'algorithmes très efficaces pour la synthèse de textures gaussiennes sur des domaines arbitrairement grands, discrets [C13] et continus [J5, J11]. En guise d'épilogue sur ce sujet, on argumentera l'intérêt de la synthèse de textures gaussiennes par rapport à la synthèse de textures par patchs à la fin du prochain chapitre (voir Section 3.4.5). Nous avons également récemment proposé un algorithme spécifique pour l'inpainting de textures gaussiennes (voir Section 2.4).

Nous avons eu la chance de développer les travaux de l'article [J1] au moment de la création du groupe de travail associé au projet ANR MATAIM au MAP5 dédié à l'étude de modèles de champs aléatoires pour l'analyse de textures issues de l'imagerie médicale (mammographies et radiographies d'os). Les champs gaussiens étaient un des modèles étudiés, notamment des modèles de champs brownien fractionnaire anisotrope (voir par exemple [Biermé et al., 2015] et les références citées dans cet article). Nos résultats ont renouvelé l'intérêt pour les modèles SNDA et BPA ce qui a entraîné de nombreux développements. A. Desolneux, L. Moisan et S. Ronsin ont développé la notion de texton canonique comme représentant de la classe des images produisant les même modèles BPA et SNDA [Desolneux et al., 2012, Desolneux et al., 2016]. En outre, comme déjà évoqué, G. Blanchet et L. Moisan [Blanchet and Moisan, 2012] puis A. Leclaire et L.

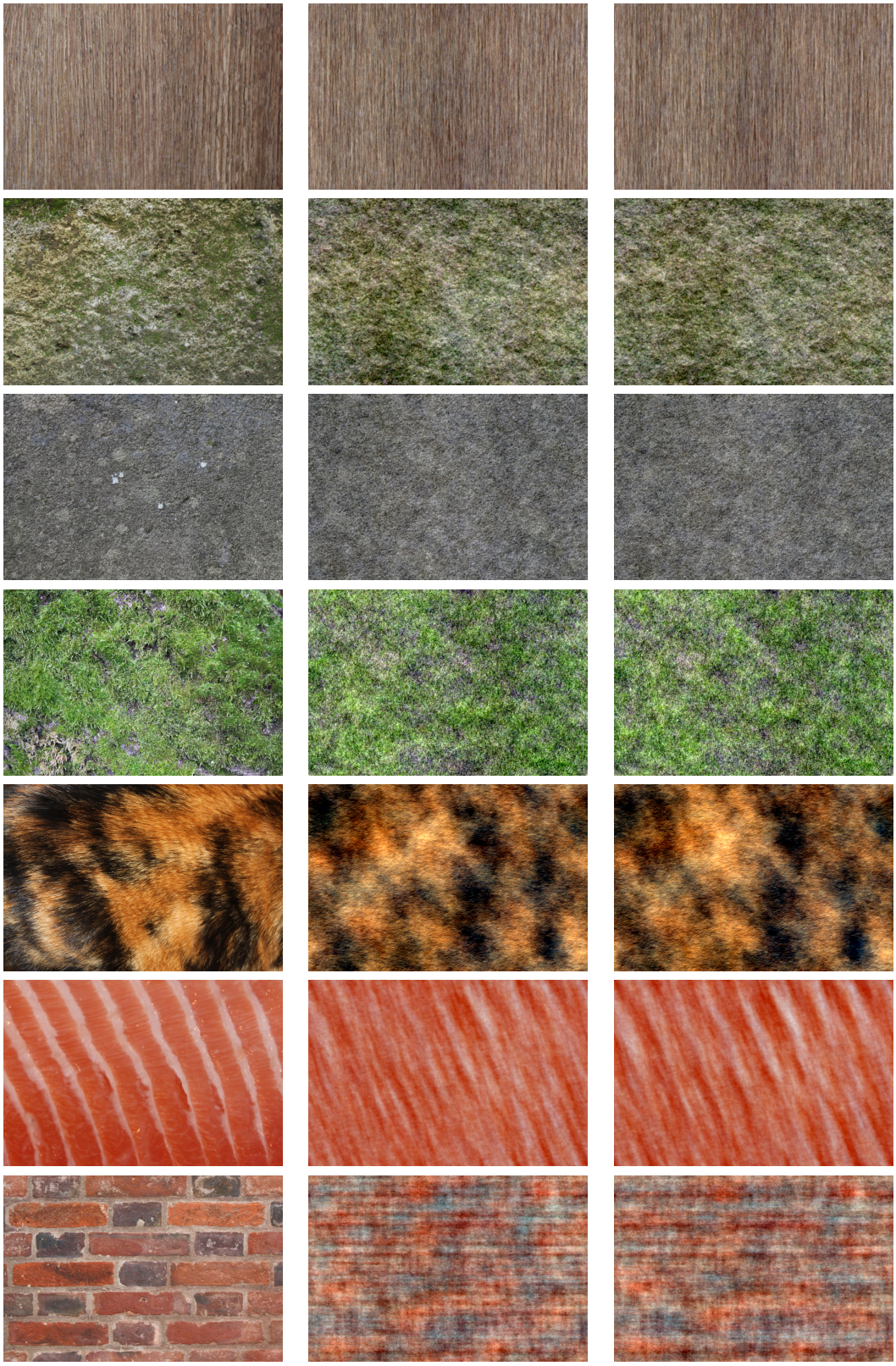


FIGURE 2.9 – Réalisations de SNDA (milieu) et BPA (droite) associés à différentes textures (gauche). Afin de comparer les deux modèles, la même phase aléatoire est utilisée pour le SNDA et le BPA associés à une même texture.

Moisan [Leclaire and Moisan, 2015] ont développé de nouveaux indices de qualité image reposant sur la similarité entre modèle BPA et SNDA. Enfin, S. Ronsin, H. Biermé et L. Moisan ont étudié l'extension du modèle de BPA sur le tore continu $\mathbb{T}^d = \mathbb{R}^d/(2\pi\mathbb{Z}^d)$ [Ronsin et al., 2016].

En dehors du projet MATAIM, G.-S. Xia, S. Ferradans, G. Peyré et J.-F. Aujol ont proposé de nouveaux algorithmes innovants relatifs à la synthèse de textures gaussiennes [Xia et al., 2014] permettant d'une part de synthétiser des textures gaussiennes dynamiques (textures variants dans le temps) et d'autres part de résoudre le problème du mélange de textures (*texture mixing*) pour les textures gaussiennes à l'aide de barycentres de Wasserstein (barycentres pour la distance de transport optimal entre deux vecteurs gaussiens [Agueh and Carlier, 2011]). Comme on le verra, nos travaux récents [J11] présentés au Chapitre 3 permettent d'obtenir un mélange de textures variant spatialement (voir Figure 3.10).

Plus récemment, de nouvelles contributions importantes en synthèse de textures se sont appuyées en partie sur les modèles SNDA et BPA. Dans leur approche de synthèse variationnelle, G. Tartavel, Y. Gousseau et G. Peyré ont montré que l'utilisation d'une contrainte spectrale, correspondant à la prescription du module de Fourier du modèle BPA, avait un rôle complémentaire par rapport à la décomposition parcimonieuse dans un dictionnaire de patchs et permettait d'améliorer significativement leurs résultats [Tartavel et al., 2014]. Par ailleurs, L. Raad, A. Desolneux et J.-M. Morel ont proposé un nouvel algorithme itératif qui génère une texture à l'aide de patchs gaussiens générés selon des modèles gaussiens locaux contrairement à l'approche classique reposant sur une simple recopie de patchs de la texture d'entrée [Raad et al., 2014, Raad et al., 2015].

2.3 Un texton orienté synthèse pour la synthèse de microtextures

Cette section résume l'acte de conférence [C13], travail précurseur de nos récents résultats en synthèse de textures procédurales [J11].

La motivation de ce travail est la suivante. La simulation de textures gaussiennes par les SNDA et BPA présentés à la section précédentes ont plusieurs limitations. Les principales limitations sont inhérentes au modèle gaussien stationnaire et ont déjà été discutées. Cependant d'autres limitations viennent des algorithmes de simulation basés sur la FFT :

— La méthode est globale : L'image de texture toute entière doit être calculée même si l'on ne souhaite synthétiser qu'une partie de l'image.

— Les images produites sont périodiques avec une taille fixe et ne peuvent donc pas être agrandie a posteriori.

Nous proposons ici une nouvelle méthode de simulation de textures gaussiennes qui ne souffre pas de ces limitations. Plus précisément, la méthode proposée a les propriétés suivantes :

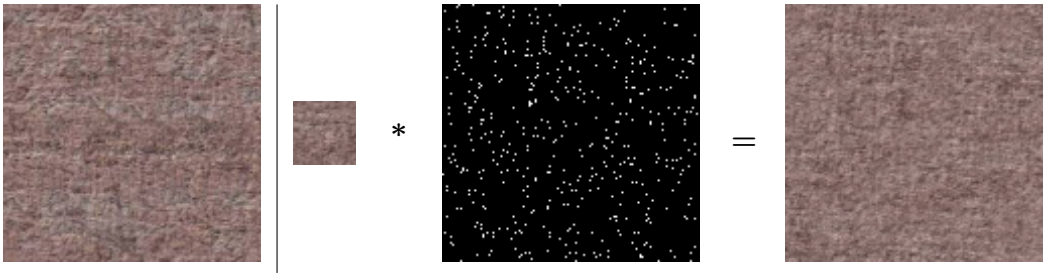


FIGURE 2.10 – Synthèse de spot noise discret à faible intensité : La texture synthétisée à droite est obtenue en convolant le texton orienté synthèse associé à l'image d'entrée de gauche par un bruit de Poisson de faible intensité (et donc parcimonieux).

— Evaluation locale et parallèle : Les textures peuvent être calculées localement et en parallèle sans avoir besoin de calculer une image entière.
— Extension de la synthèse : Les textures synthétisées peuvent être arbitrairement étendues car l'algorithme de simulation repose sur un modèle sur le réseau infini $\mathbb{Z}^2$.
— Faible coût mémoire : Les textures gaussiennes sont résumées par une petite image appelée "texton orienté synthèse" et il n'est pas nécessaire de conserver en mémoire les textures originales.

La Figure 2.10 illustre le principe de l'utilisation du texton orienté synthèse. L'algorithme de synthèse repose sur un spot noise discret défini sur le domaine $\mathbb{Z}^2$ que nous définissons maintenant.

2.3.1 SND et SNDA sur $\mathbb{Z}^2$

2.3.1.1 Définitions et simulation

Soit $h : \mathbb{Z}^2 \to \mathbb{R}^d$ (avec $d = 1$ pour les images en niveau de gris et $d = 3$ pour les images couleur) ayant un support fini $S_h = \{x \in \mathbb{Z}^2,\ h(x) \neq 0\}$. On définit $\tilde{h}(x) = h(-x)$ la version symétrique de h.

Définition 2.6 (Spot noise discret et spot noise discret asymptotique sur $\mathbb{Z}^2$ ($\mathrm{SND}_{\mathbb{Z}^2}$ et $\mathrm{SNDA}_{\mathbb{Z}^2}$))**.** *Le spot noise discret sur $\mathbb{Z}^2$ ($\mathrm{SND}_{\mathbb{Z}^2}$) associé au spot h est le champ aléatoire*

$$F_{\lambda,h}(x) = \sum_{y\in\mathbb{Z}^2} P_\lambda(y)h(x-y) = P_\lambda * h,$$

où P_λ est un bruit blanc de Poisson d'intensité $\lambda > 0$, c'est-à-dire les v.a. $P_\lambda(x)$, $x \in \mathbb{Z}^2$ sont i.i.d. et suivent une loi de Poisson de paramètre $\lambda > 0$.

Le SND renormalisé

$$G_{\lambda,h} = \frac{F_{\lambda,h} - \mathbb{E}(F_{\lambda,h})}{\sqrt{\lambda}} = \frac{1}{\sqrt{\lambda}}\Big(h * P_\lambda - \lambda \sum_{y\in\mathbb{Z}^2} h(y)\Big)$$

est de moyenne nulle et sa fonction de covariance est $h * h^T$. Lorsque l'intensité λ du bruit de Poisson tend vers l'infini, $G_{\lambda,h}$ converge en distribution (*i.e.* au sens des lois finies dimensionnelles) vers le champ gaussien de moyenne nulle et de covariance $h * h^T$. Ce champ gaussien, que l'on appelle *spot noise discret asymptotique* sur $\mathbb{Z}^2$ associé au spot h correspond à la convolution du spot h par un bruit blanc gaussien standard W sur $\mathbb{Z}^2$.

La simulation de la restriction d'un SND sur un domaine rectangulaire $\Omega \subset \mathbb{Z}^2$ nécessite de simuler le bruit blanc de Poisson sur le domaine augmenté $\Omega - S_h = \{x - y,\ x \in \Omega,\ y \in S_h\}$. On peut alors simuler la restriction du SND à Ω par convolution périodique suivi d'une restriction à l'intérieur du domaine. Cependant, pour des valeurs faibles de l'intensité λ du bruit de Poisson, il est plus pertinent de simuler le SND par sommation directe. Quelque soit la méthode choisie, il est alors possible d'étendre a posteriori la synthèse obtenue sur le domaine Ω en simulant le bruit de Poisson à l'extérieur de Ω tout en gardant la même réalisation de bruit de Poisson aux endroits où il a déjà été tiré, à savoir sur $\Omega - S_h = \{x - y,\ x \in \Omega,\ y \in S_h\}$. Pour cela, le plus simple est d'avoir recourt à une simulation à la demande de bruit de Poisson basé sur une partition de $\mathbb{Z}^2$ en cellules. Ce procédé est l'équivalent discret de l'algorithme de simulation à la demande d'un processus de Poisson sur $\mathbb{R}^2$ qui est détaillé à la section 3.2 du chapitre 3. Le coup moyen de l'algorithme de sommation direct est de $\lambda|S_h|$ opérations par pixels. Cette grandeur $\lambda|S_h|$ est le nombre moyen de recouvrements d'un point par les translations de h. On l'appelle nombre moyen d'impacts (NMI).

2.3.1.2 SND et SNDA sur $\mathbb{Z}^2$ associés à une image de texture

Etant donnée une image de texture $u : \Omega \to \mathbb{R}^d$ définie sur le domaine discret fini Ω de taille $M \times N$, on doit définir un spot correspondant défini sur $\mathbb{Z}^2$. Comme on souhaite générer des images ayant même moyenne et variance que u, on considère la version centrée normalisée de u, à savoir

$$t_u = \frac{1}{\sqrt{MN}}(u - \mathrm{moy}(u))$$

où $\mathrm{moy}(u)$ est la moyenne de u. On génère alors le spot noise discret

$$\mathrm{moy}(u) + G_{\lambda,t_u}$$

ou son équivalent asymptotique. Alors ce champ a pour moyenne $\mathrm{moy}(u)$ et pour fonction de covariance

$$t_u * t_u^T(x) = \frac{1}{MN} \sum_{y \in \mathbb{Z}^2} (u(x+y) - \mathrm{moy}(u))(u(y) - \mathrm{moy}(u))^T$$

où par convention $u(x+y) - \mathrm{moy}(u) = 0$ si $x + y \notin \Omega$. Ainsi on a la même expression de covariance que dans le cas circulaire si ce n'est qu'ici tous les couples de faux voisins $(x + y, y)$ obtenus par condition de périodicité ne sont pas comptabilisés.

2.3.1.3 Liens entre SNDA sur $\mathbb{Z}^2$ et SNDA circulaire

Soit h un spot à support S_h fini. Pour tout domaine rectangulaire Ω de taille $M \times N$ qui soit plus grand que S_h, on peut définir une version (M, N)-périodique de h que l'on note encore h. Il est alors pertinent de comparer la restriction au domaine Ω du champ aléatoire $\mathrm{SNDA}_{\mathbb{Z}^2}(h)$ et le champ gaussien stationnaire (M, N)-périodique $\mathrm{SNDA}_{\Omega}(h)$. Ces deux champs sont obtenus par convolution de h par un bruit blanc gaussien standard. La seule différence est que pour $\mathrm{SNDA}_{\mathbb{Z}^2}(h)$ le bruit blanc et la convolution sont définies sur $\mathbb{Z}^2$ et que pour $\mathrm{SNDA}_{\Omega}(h)$ le bruit blanc et la convolution sont définies sur Ω avec des conditions périodiques. Il en résulte que les deux champs ont exactement la même distribution à l'intérieur du domaine Ω. Plus précisément, la restriction de $\mathrm{SNDA}_{\mathbb{Z}^2}(h)$ et la restriction de $\mathrm{SNDA}_{\Omega}(h)$ au sous-ensemble $\Omega \ominus (-S_h) = \{x \in \Omega, x - S_h \subset \Omega\}$ ont exactement la même distribution.

2.3.2 Algorithme pour le calcul d'un texton orienté synthèse

Le problème que l'on cherche à résoudre est le suivant : Etant donné un support $S \subset \mathbb{Z}^2$ et une texture d'exemple u, déterminer un spot t ayant pour support S tel que la $\mathrm{SND}_{\mathbb{Z}^2}$ associé à t soit une bonne approximation de la texture gaussienne associée à u (et ce même pour une faible intensité λ).

Autrement dit, on cherche à imposer une contrainte de support et une contrainte de valeurs de modules de Fourier à t. Inspiré par la littérature du problème de restitution de phase (*phase retrieval*) [Hayes, 1982], on propose l'Algorithme 1 pour calculer le texton orienté synthèse t associé à u. Pour une image couleur RGB, l'étape d'imposition du spectre puissance devient

$$\widehat{p_{t_u}(t)} = \widehat{t}_u \left(\frac{\widehat{t}_u^* \widehat{t}}{|\widehat{t}_u^* \widehat{t}|} \mathbf{1}_{\widehat{t}_u^* \widehat{t} \neq 0} + \mathbf{1}_{\widehat{t}_u^* \widehat{t} = 0} \right).$$

La (non) convergence de l'Algorithme 1 est discuté dans l'acte de conférence [C13]. Les expériences montrent que l'algorithme converge pour chaque initialisation mais que les points de convergence sont aléatoires. En effet, l'algorithme n'a pas de garantie de convergence car la projection spectrale n'est pas convexe. En pratique la variance se stabilise au bout d'une vingtaine d'itérations et l'on a fixé le nombre d'itérations à 50. Notons que cette convergence vers un point aléatoire est un phénomène classique pour les méthodes variationnelles itératives de synthèse de texture qui cherche à minimiser une énergie en partant d'un bruit blanc [Portilla and Simoncelli, 2000, Tartavel et al., 2014].

2.3.3 Résultats

La Figure 2.11 présentent plusieurs résultats de synthèse de textures utilisant le texton orienté synthèse calculé avec l'Algorithme 1 et en utilisant seulement un nombre moyen d'impacts (NMI) de 50. On remarque que des microtextures de qualité raisonnable peuvent-être synthétisée à partir de textons de support très réduit (les tailles des textons sont 51×51 et 31×31 pixels). Par ailleurs, avec seulement

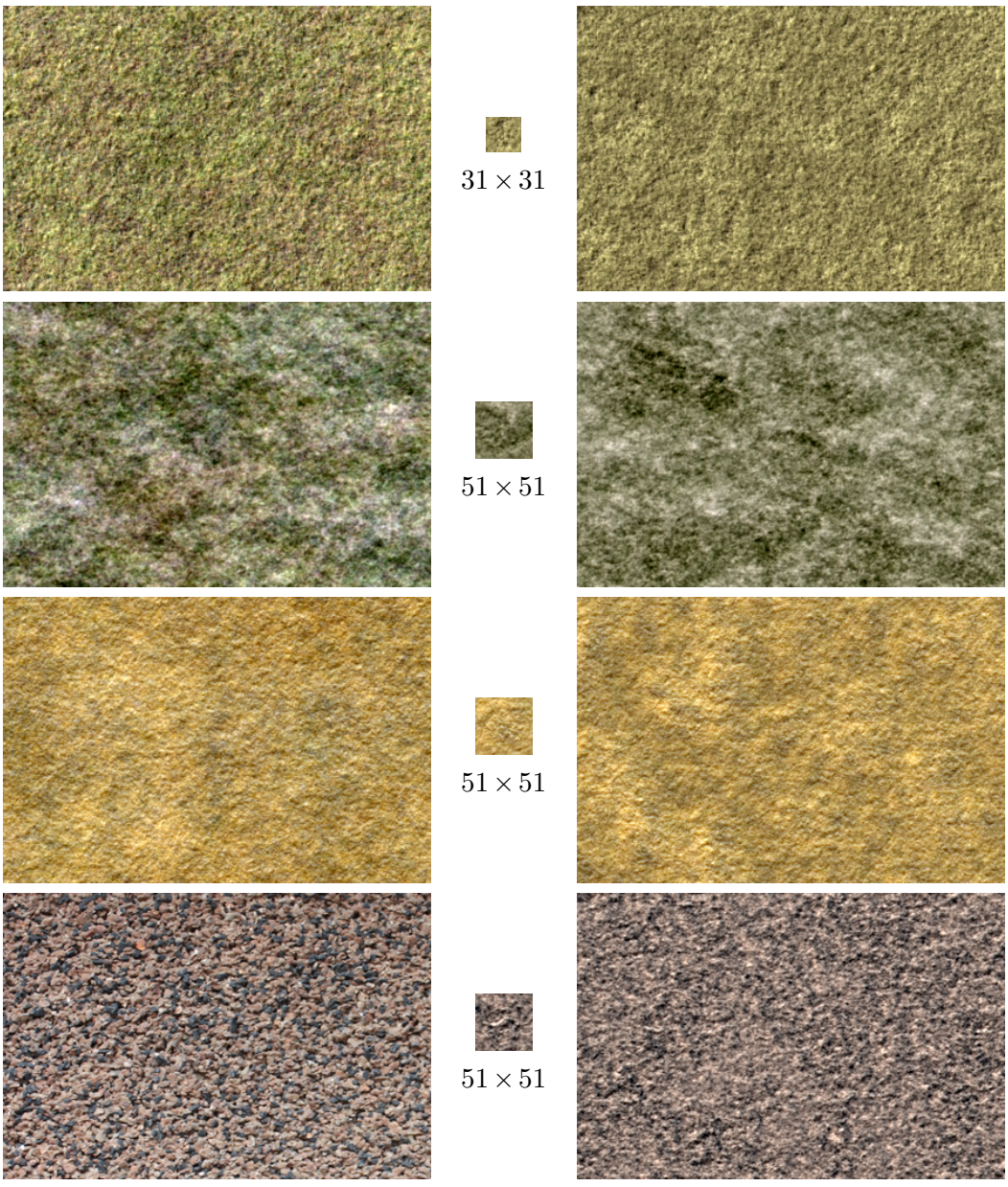


FIGURE 2.11 – Texton orienté synthèse (milieu) et spot noise discret associé (droite) pour un nombre moyen d'impacts de 50 pour différentes textures (gauche). Les images originales sont toutes de taille 384×256, les tailles des textons sont précisées sous les images.

Dans cet algorithme, toutes les images ont la même taille que l'entrée u définie sur le domaine Ω et le symbole $\hat{\cdot}$ correspond à la TFD sur Ω calculée par FFT.

- **Entrée :** Image u définie sur Ω, support $S \subset \Omega$
- **Calcul du spectre cible :** Calculer $t_u = \frac{1}{\sqrt{|\Omega|}}(u - \text{moy}(u))$ et sa TFD $\hat{t}_u$
- **Initialisation aléatoire :** Initialiser $t \in \mathbb{R}^\Omega$ par un bruit blanc gaussien standard
- **Projection itérative sur les contraintes :** Faire $n = 50$ fois
  1. **Imposition du spectre de puissance** de $\hat{t}_u$:
     (a) Calculer la TFD $\hat{t}$ de t
     (b) $\hat{t} \leftarrow |\hat{t}_u| \frac{\hat{t}}{|\hat{t}|}$
     (c) Calculer t par TFD inverse
  2. **Imposition du support** S : $t \leftarrow \mathbb{1}_S \cdot t$
- **Sortie :** Retourner t

Algorithme 1 : Calcul du texton orienté synthèse associé à une image u

un NMI de 50 le SND ressemble visuellement à sa texture gaussienne limite. Cette observation empirique a des conséquences pratiques très importantes : Plutôt que de convoler le texton par un bruit blanc gaussien pour lequel tous les pixels sont non nuls et qui impose une convolution par FFT, on peut se contenter de convoler le texton par un bruit de Poisson à faible intensité qui est très parcimonieux. Pour ce type de bruit, l'algorithme de convolution est encore valable, mais il peut avantageusement être remplacé par une simple sommation directe des translations du texton.

La Figure 2.12 montre que cette importante propriétés n'est pas satisfaite par d'autre approches visant à résumer la texture gaussienne par une image de petit support. Pour cette comparaison nous avons considéré la restriction du texton de luminance [Desolneux et al., 2012] qui lui n'est pas "orienté synthèse" car par construction il concentre l'énergie au centre de son support. Nous avons également comparer avec un crop d'une synthèse BPA, ce qui correspond à une seule itération de l'Algorithme 1. Cette comparaison montre qu'à la fois l'initialisation aléatoire et les itérations visant à bien résumer le spectre sur le support S sont nécessaires pour obtenir une synthèse de qualité avec NMI faible.

A la suite de la publication de ces travaux [C13], nous avons développer une version CUDA de l'algorithme de synthèse par SND permettant une évaluation parallèle de la texture grâce à la simulation à la demande d'un processus de Poisson discret (voir la section 3.2 du chapitre 3 pour les détails). Nous avons alors obtenu un temps de calcul de 12,5 ms (soit 80 images par secondes) pour une image $1024 \times$

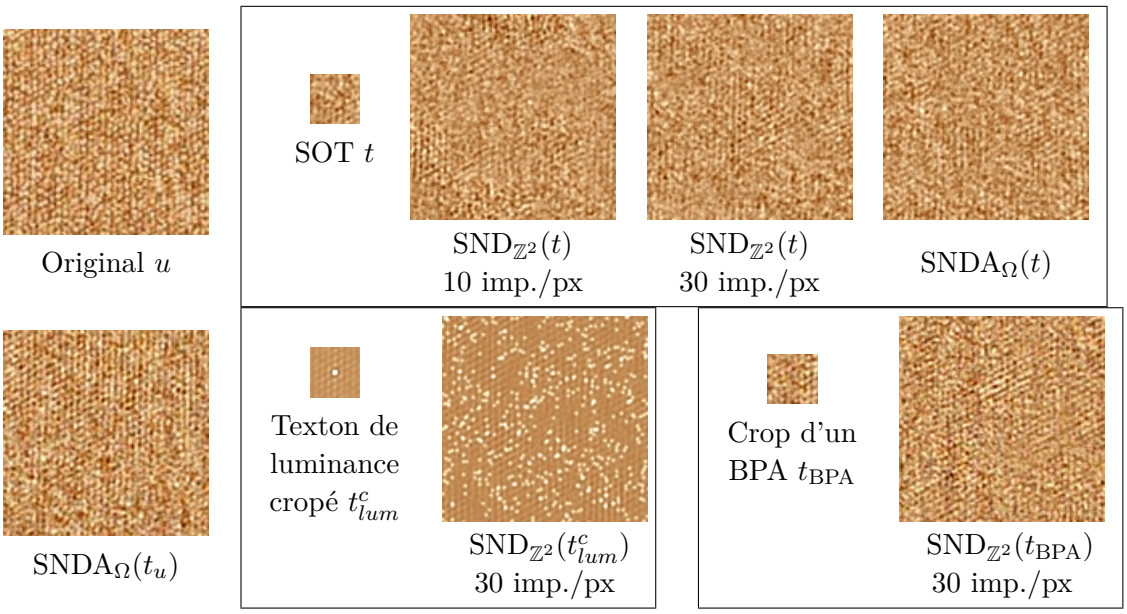


FIGURE 2.12 – Comparaison des SND à faible intensité associés à un texton orienté synthèse, à la restriction du texton de luminance [Desolneux et al., 2012], et à la restriction d'une réalisation du BPA associé à la texture d'entrée u. Les textures sont de tailles 128×128 et les textons sont de taille 31×31. Seul le texton orienté synthèse permet d'obtenir une texture proche de sa limite gaussienne avec un nombre moyen d'impacts de 30.

1024 avec 30 impacts par pixels. Ces performances étant bien supérieures à celles obtenues pour la synthèse de textures procédurales par *Gabor noise* [J5], nous nous sommes naturellement intéressés à développer un modèle de textures procédurales reposant sur le calcul d'un texton. Ce nouveau modèle, baptisé *texton noise* [J11], sera présenté à la Section 3.4 du Chapitre 3.

2.4 Inpainting de microtextures par simulation gaussienne conditionnelle

Etant donnée une image contenant des zones manquantes, le problème d'inpainting consiste à remplir ces zones manquantes en accord avec le voisinage connu de ces zones. L'inpainting est un problème de restauration relativement récent en traitement d'images, le premier article datant de la fin des années 1990 [Masnou and Morel, 1998] et le nom inpainting ne fut introduit qu'en 2000 [Bertalmio et al., 2000]. Toutefois de nombreuses solutions ont été proposées pour résoudre ce problème, et ce manuscrit n'est pas le lieu pour discuter d'un état de l'art complet. On notera seulement que les premières méthodes variationnelles permettaient d'obtenir de bons prolongements de contours [Masnou and Morel, 1998, Bertalmio et al., 2000, Chan and Shen, 2002] mais pouvaient difficilement restaurer des parties texturées à cause d'un a piori sur la régularité des zones inpaintées. Dans un deuxième temps, des approches inspirées par les nouvelles méthodes de synthèse de textures par patchs [Efros and Leung, 1999, Wei and Levoy, 2000, Efros and Freeman, 2001] ont permis d'obtenir de bien meilleurs résultats sur les zones texturées [Criminisi et al., 2004]. Enfin, plus récemment, des approches variationnelles reposant sur l'utilisation des patchs ont été proposées et étudiées [Aujol et al., 2010, Arias et al., 2011].

Nous présentons ici un algorithme qui permet uniquement d'inpainter des textures gaussiennes stationnaires. Par définition, l'inpainting est un problème mal posé. En revanche, si l'on dispose d'un modèle stochastique $p(u)$ réaliste pour l'image entière u, alors une solution plausible peut être obtenue par simulation conditionnelle, c'est-à-dire simuler une image aléatoire u avec probabilité $p(u)$ étant données les valeurs connues de u. Ce point de vue stochastique n'a pas d'intérêt pratique pour l'inpainting d'une image naturelle quelconque u car l'on ne connaît pas de loi $p(u)$. Cependant, si u est une image de texture pour laquelle on est en mesure d'estimer un modèle aléatoire à partir des valeurs connues, alors la simulation conditionnelle permet de résoudre l'inpainting. Nous montrons ci-dessous que pour des textures gaussiennes la simulation conditionnelle peut être assez facilement mise en œuvre pour "inpainter" des zones de taille et forme quelconque.

2.4.1 Simulation gaussienne conditionnelle et estimateur de krigeage

Décrivons rapidement la méthode de simulation gaussienne conditionnelle présentée dans [Lantuéjoul, 2002]. On considère un ensemble fini Ω (qui sera l'ensemble des indices des pixels pour notre application) et un vecteur gaussien $(F(x))_{x\in\Omega}$ de moyenne nulle et de covariance $\Gamma(x,y) = \mathrm{Cov}(F(x),F(y)) = \mathbb{E}(F(x)F(y))$. On note également $\mathcal{C} \subset \Omega$ un ensemble de points de conditionnement. Alors, en tout point $x \in \Omega$, l'estimateur de krigeage $F^*(x)$ de $F(x)$ sachant les valeurs $(F(c))_{c\in\mathcal{C}}$ est par définition l'espérance conditionnelle

$$F^*(x) = \mathbb{E}(F(x)|F(c),\ c \in \mathcal{C}).$$

Comme F est gaussien, un résultat standard de probabilités [Doob, 1953] assure que $F^*(x)$ est la projection orthogonale de $F(x)$ sur le sous-espace vectoriel des combinaisons linéaires des variables aléatoires (v.a.) $(F(c))_{c\in\mathcal{C}}$. Ainsi, il existe des coefficients $(\lambda_c(x))_{c\in\mathcal{C}}$ (déterministes), appelés coefficients de krigeage, tels que

$$F^*(x) = \sum_{c\in\mathcal{C}} \lambda_c(x)F(c).$$

En particulier F^* est également un vecteur gaussien. On montre facilement que les vecteurs gaussiens F^* et $F - F^*$ sont indépendants [Lantuéjoul, 2002]. En adoptant la notation

$$\varphi^*(x) = \sum_{c\in\mathcal{C}} \lambda_c(x)\varphi(x)$$

pour toute fonction $\varphi : \Omega \to \mathbb{R}$, il résulte de l'observation précédente qu'un échantillon de la loi gaussienne conditionnelle de F sachant $F_{|\mathcal{C}} = \varphi_{|\mathcal{C}}$ est donnée par la somme $\varphi^* + F - F^*$. Dans cette décomposition, φ^* est appelée la *composante de krigeage* (qui est fixe étant donnée les valeurs $(\varphi(c))_{c\in\mathcal{C}}$) et $F - F^*$ est appelée la *composante d'innovation* (qui est indépendante des valeurs de conditionnement).

Afin de pouvoir effectuer une simulation conditionnelle de F sachant $F_{|\mathcal{C}} = \varphi_{|\mathcal{C}}$, nous devons donc seulement déterminer les coefficients de krigeage $(\lambda_c(x))_{c\in\mathcal{C}}$ pour chaque point $x \in \Omega$. En regroupant tous ces coefficients dans la matrice $\Lambda = (\lambda_c(x))_{x\in\Omega,c\in\mathcal{C}}$ de taille $|\Omega|\times|\mathcal{C}|$, on montre aisément que Λ est solution de l'équation

$$\Lambda\Gamma_{|\mathcal{C}\times\mathcal{C}} = \Gamma_{|\Omega\times\mathcal{C}}.$$

Les coefficients de krigeage Λ sont donc donnés par $\Lambda = \Gamma_{|\Omega\times\mathcal{C}}\Gamma^{-1}_{|\mathcal{C}\times\mathcal{C}}$ si la matrice $\Gamma_{|\mathcal{C}\times\mathcal{C}}$ est inversible (si elle ne l'est pas on remplace $\Gamma^{-1}_{|\mathcal{C}\times\mathcal{C}}$ par la pseudo-inverse de $\Gamma^{\dagger}_{|\mathcal{C}\times\mathcal{C}}$ pour obtenir une des solutions possibles). Pour finir, remarquons qu'avec ces notations matricielles, alors la composante de krigeage s'écrit $\varphi^* = \Lambda\varphi_{|\mathcal{C}}$ avec la convention que $\varphi_{|\mathcal{C}}$ et φ^* s'écrivent sous forme de vecteurs colonnes.

2.4.2 Algorithme d'inpainting gaussien

On propose dans [C14] d'utiliser la simulation gaussienne conditionnelle pour la synthèse de microtextures modélisées par de SNDA stationnaire. Dans le cadre général ci-dessus, la composante de krigeage est donnée par

$$\varphi^* = \Lambda\varphi_{|\mathcal{C}} = \Gamma_{|\Omega\times\mathcal{C}}\Gamma^{-1}_{|\mathcal{C}\times\mathcal{C}}\varphi_{|\mathcal{C}}.$$

Or pour notre application multiplier par la matrice Γ (non restreinte) correspond à une convolution calculable rapidement par FFT. Grâce à cette observation nous proposons dans [C14] une implémentation de la simulation conditionnelle où seule l'opération $\Gamma^{-1}_{|\mathcal{C}\times\mathcal{C}}\varphi_{|\mathcal{C}}$ (ou plutôt la résolution du système linéaire $\Gamma_{|\mathcal{C}\times\mathcal{C}}X = \varphi_{|\mathcal{C}}$) est effectuée à l'aide de matrices. Cela réduit le coup d'une approche naïve, mais le coup en $\mathcal{O}(|\mathcal{C}|^3)$ de la résolution explicite du système linéaire contraint tout de même à utiliser un ensemble de points de conditionnement $\mathcal{C}$ le plus restreint possible. Nous travaillons actuellement sur une nouvelle implémentation faisant appel à une méthode itérative pour résoudre le système $\Gamma_{|\mathcal{C}\times\mathcal{C}}X = \varphi_{|\mathcal{C}}$ ne faisant appel qu'à des opérations de convolution rapide afin de pouvoir utiliser l'ensemble des données comme points de conditionnement. C'est pourquoi nous ne rentrerons pas dans les détails de l'algorithme ici, et nous renvoyons le lecteur intéressé à l'acte de conférences [C14] et à la page web d'Arthur Leclaire depuis laquelle les codes sources matlab sont téléchargeables.

Cependant, même en limitant l'ensemble $\mathcal{C}$ des points de conditionnement à une bande de 3 pixels le long de la frontière du masque, les résultats de simulation conditionnelle sont tout à fait satisfaisants dès lors que le modèle gaussien a pu être convenablement estimé sur la partie non corrompue de l'image. On présente pour commencer à la Figure 2.13 une validation de la simulation conditionnelle gaussienne pour l'inpainting de texture dans le cas où le modèle gaussien est estimé sur toute l'image d'entrée (y compris les valeurs masquées). On observe sur cet exemple le rôle complémentaire de la composante de krigeage qui étend les structures venant des données connues, et celui de la composante d'innovation qui ajoute des détails de grain dans les zones éloignées des points de conditionnement. On observe également avec cet exemple que la simulation conditionnelle permet de remplir des zones de taille et de forme quelconque.

La Figure 2.14 montre différents exemples d'inpainting où le modèle gaussien est appris sur une zone non corrompue de l'image masquée. On observe que le résultat est satisfaisant si la texture d'entrée est bien synthétisable par un modèle gaussien (le troisième exemple se situe à la limite et on observe une petite perte de structure). Enfin, pour finir, la Figure 2.15 compare notre méthode à l'algorithme d'Efros-Leung [Efros and Leung, 1999]. Cet algorithme est loin d'être au niveau état de l'art pour l'inpainting mais c'est un algorithme performant pour la synthèse de textures, et notamment des macro-textures. On observe que pour cette expérience, l'inpainting gaussien reproduit plus fidèlement le contenu fréquentiel de la texture que l'algorithme d'Efros-Leung, ce qui valide l'intérêt de notre approche même si elle se limite aux textures gaussiennes (voir aussi la discussion sur l'intérêt de la

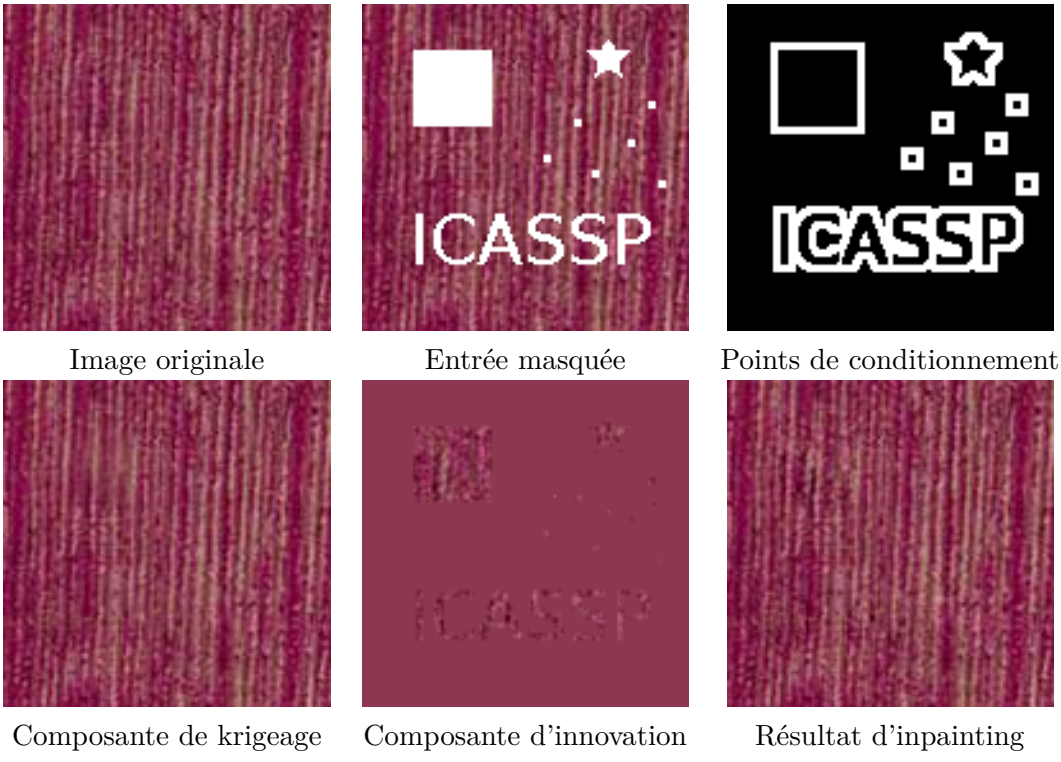


FIGURE 2.13 – Inpainting à l'aide du modèle gaussien oracle : La texture masquée est inpaintée en utilisant le modèle gaussien estimé sur l'image originale non masquée. L'ensemble des points de conditionnement est une bande de largeur 3 pixels le long de la frontière du masque. La deuxième ligne montre la composante de krigeage et la composante d'innovation, ainsi que le résultat final obtenu comme la somme de ces deux textures (modulo la moyenne). La méthode d'inpainting proposée permet de remplir des zones de taille et de forme quelconque.

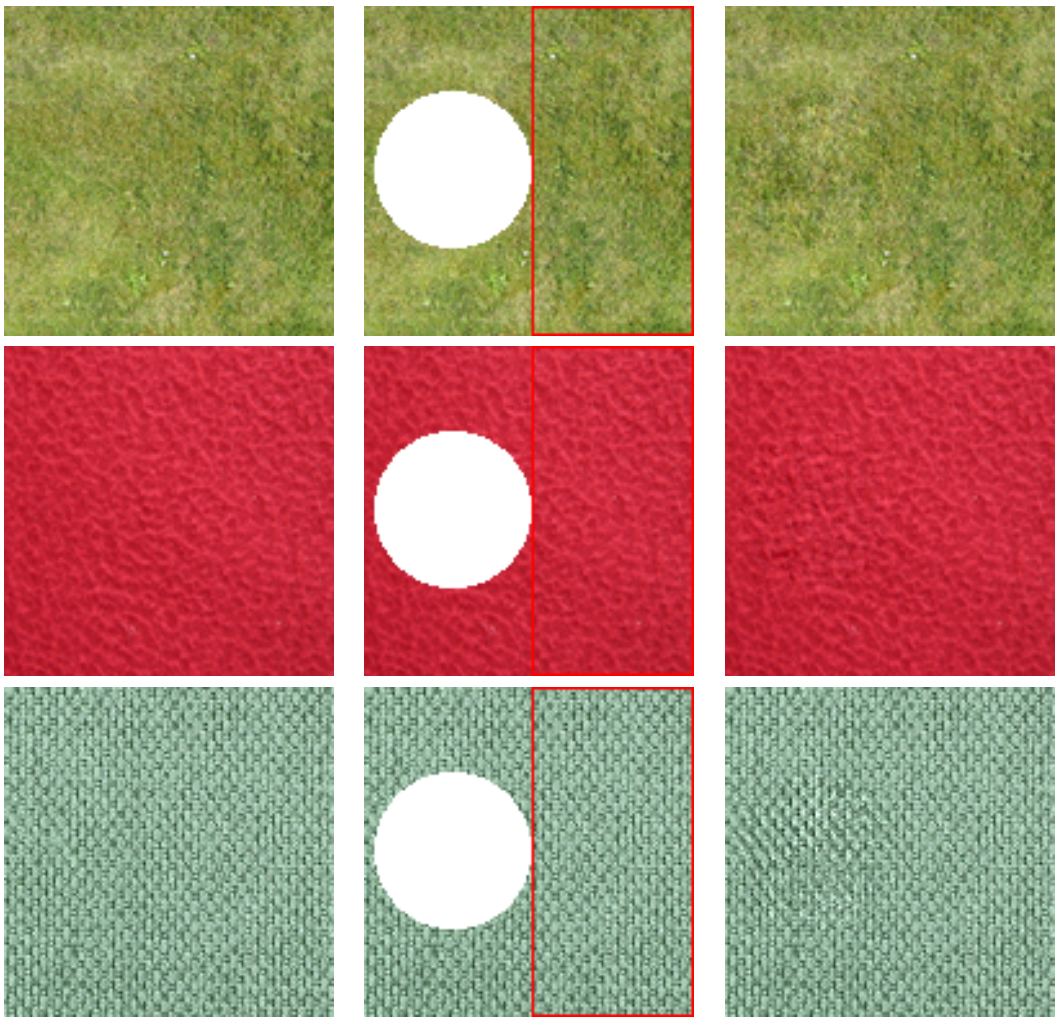


FIGURE 2.14 – Exemples d'inpainting de texture à partir d'un modèle gaussien appris sur une partie non corrompue de l'image d'entrée délimitée par un rectangle rouge.

Input | Comp. de krigeage | Inpainting gaussien | Efros-Leung

FIGURE 2.15 – Comparaison avec Efros-Leung [Efros and Leung, 1999]. L'image d'entrée est inpaintée à l'aide du modèle gaussien appris sur le rectangle rouge. Comme on peut l'observer, notre algorithme préserve mieux le contenu fréquentiel de la texture grâce notamment au prolongement des structures apporté par la composante de krigeage.

synthèse de textures gaussiennes à la Section 3.4.5).

Chapitre 3

Synthèse par l'exemple de textures procédurales de type shot noise

3.1 Introduction

Les textures procédurales sont un outil de base en *computer graphics* pour synthétiser des textures sur la surface d'objets 3D. Lorsque l'on vient de la communauté traitement d'images, il est assez difficile d'appréhender la notion de texture procédurale. En effet, une texture procédurale n'est pas une image mais une procédure (*i.e.* un algorithme) qui étant donnée une coordonnée spatiale dans le plan ou dans l'espace renvoie une couleur. Il y a donc une approche différente de celle du traitement d'images sur la création de contenu visuelle : On privilégie le calcul à chaque image plutôt que le stockage de fichiers. Illustrons ces différences avec un exemple élémentaire. Supposons que l'on souhaite créer une texture d'échiquier. Si l'on fait du traitement d'images on fixera une taille d'image en on remplira un tableau de 0 et de 1 pour créer une image d'échiquier que l'on pourra par la suite paver périodiquement si besoin. En revanche, une texture procédurale produisant un échiquier sera la ligne de code suivante évaluant la fonction :

$$\text{checkerboard}(x_1, x_2) = \left(\text{mod}(\lfloor x_1/r \rfloor, 2) \neq \text{mod}(\lfloor x_2/r \rfloor, 2)\right),$$

avec r le paramètre de la taille d'une case.

L'intérêt des textures procédurales par rapport aux images de textures est multiple.

- Les textures procédurales ont une représentation compact en mémoire (quelques lignes de codes contre la taille d'un fichier image).
- Les textures procédurales peuvent être évaluées en parallèle sur différentes coordonnées ce qui est idéal pour l'évaluation sur carte graphique (GPU, pour *Graphics Processing Unit*).
- Les textures procédurales permettent plus facilement d'appliquer les textures 2D sur des surfaces 3D (*texture mapping*) car elles n'entraînent pas de problème d'interpolation (il reste cependant à gérer les problèmes d'aliasing que l'on abordera par la suite).

Afin de pouvoir représenter des textures irrégulières et non déterministes, des modèles de bruits procéduraux (*procedural noise*) ont été développés comme le

Perlin noise [Perlin, 1985], le *wavelet noise* [Cook and DeRose, 2005], l'*anisotropic noise* [Goldberg et al., 2008] et le *Gabor noise* [Lagae et al., 2009]. Ces bruits procéduraux ont en premier lieu été conçus comme des générateurs de nombres aléatoires qui respectent une certaine corrélation spatiale. Ces bruits procéduraux étaient alors combiner grâce à des fonctions non linéaires astucieuses telles que l'application de cartes de couleurs (*colormap*) ou la perturbation de la géométrie de structures régulières [Perlin, 1985, Lagae et al., 2010a]. On peut souligner l'importance de ces techniques pour l'industrie des films d'animation et des effets spéciaux par le fait que K. Perlin a reçu un oscar technique pour l'élaboration du *Perlin noise*. Toutefois il est clair que de créer de bonnes procédures de design pour générer des textures réalistes demande d'avoir une bonne intuition, une réelle expertise technique et des qualités artistiques. Cette difficile tâche est d'ailleurs réservée dans l'industrie à des experts dénommés *texture artists*.

Proposer une méthode automatique permettant de calculer les paramètres d'un bruit procédural permettant de reproduire une texture réelle donnée en entrée sous forme d'image numérique est donc un problème ayant potentiellement un enjeu industriel important. On appelle ce problème "bruit procédural par l'exemple" (*noise by example*). Avec les modèles tels que l'*anisotropic noise* [Goldberg et al., 2008] et le *Gabor noise* [Lagae et al., 2009] il est devenu possible d'avoir un contrôle fin sur la corrélation des bruits procéduraux, ou plutôt de leur spectre de puissance (que l'on définit mathématiquement comme la transformée de Fourier de la fonction de covariance du champ aléatoire représentant le bruit). Le contenu visuel d'un bruit procédural sans application de fonctions non linéaires devenait intéressant. Il s'est alors naturellement posé la question de savoir quels types de textures pouvaient être reproduites par un bruit procédural.

L'article *Gabor noise by example* [J5], co-écrit avec les chercheurs en computer graphics A. Lagae, S. Lefebvre et G. Drettakis, a été le premier à proposer une solution précise et automatique permettant de reproduire toutes les textures gaussiennes introduites au Chapitre 2. En comparaison, les premiers travaux sur ce problème de *noise by example* étaient soit limités à des textures isotropes [Lagae et al., 2010b] soit non automatiques (*i.e.* ils nécessitaient une intervention manuelle pour déterminer les paramètres du bruit) [Gilet et al., 2010]. La solution proposée dans [J5] est elle automatique et optimale en terme de classe de textures traitées. En effet, comme chaque modèle de bruit procédural somme localement des grandeurs indépendantes, il a été vérifié que tous les bruits procéduraux correspondent approximativement à des champs gaussiens [Lagae et al., 2010a], ce qui est une justification formelle de la pratique usuelle qui consiste à caractériser un bruit procédural par son spectre de puissance. Autrement dit, se limiter aux textures gaussiennes est d'une part justifié par la nature des bruits procéduraux et d'autre part permet de rendre le problème du *noise by example* bien posé car alors deux textures sont identiques ou proches si leurs spectres de puissance respectifs sont identiques ou proches. On mentionne toutefois qu'en 2014 *Gilet et al* [Gilet et al., 2014] ont introduit un nouveau modèle de bruit procédural baptisé *local random phase noise* et ont démontré que l'on pouvait reproduire certaines textures structurées (et donc non gaussienne) à l'aide de

ce modèle qui impose une certaine préservation des phases de l'image d'entrée.

Malheureusement, les performances de la méthode [J5] limitaient son potentiel applicatif. En effet, pour une petite texture d'entrée de taille 128×128, le temps de calcul nécessaire pour la partie analyse déterminant les paramètres du bruit de Gabor associé est de l'ordre de deux minutes. Le temps de calcul nécessaire pour la synthèse d'une image full HD (1920×1080) est de l'ordre d'une seconde, ce qui rend la méthode prohibitive pour toute application temps réel.

Ces limitations pratiques nous ont amenés à chercher le modèle le plus simple permettant de synthétiser toutes les textures gaussiennes, plutôt que de s'appuyer sur un modèle de bruit existant comme le *Gabor noise*. Pour cela nous avons développé avec A. Leclaire et L. Moisan le *texton noise* [J11] qui est défini comme un simple shot noise poissonnien utilisant une seule et même fonction de spot appelée *texton*. Comme on le verra à la Section 3.4, ce nouveau modèle de bruit conserve les avantages du *Gabor noise by example* et permet de gagner deux ordres de grandeur de temps de calcul, à la fois pour l'analyse et pour la synthèse sur GPU.

Avant de présenter brièvement les idées principales du *Gabor noise by example* [J5] à la Section 3.3 et du *texton noise* [J11] à la Section 3.4, nous commençons par présenter l'algorithme de simulation parallèle et à la demande d'un processus ponctuel de Poisson sur GPU car cet algorithme est central pour les deux méthodes.

3.2 Simulation parallèle et à la demande d'un processus de Poisson

Nous présentons en détail dans cette section l'algorithme de simulation parallèle et à la demande d'un processus ponctuel de Poisson sur $\mathbb{R}^2$. Cet algorithme a été introduit par Worley [Worley, 1996] et il est au cœur de l'implémentation de l'évaluation des textures procédurales de type Gabor noise [Lagae et al., 2009] [J5] présentée à la Section 3.3 ci-dessous. Il est encore plus central dans l'implémentation OpenGL du texton noise [J11] présentée à la Section 3.4 car c'est la seule procédure aléatoire pour ce bruit procédural. Ceci nous a conduits à optimiser certains aspects de cette procédure. Le même principe peut être utilisé pour simuler à la demande un bruit de Poisson sur $\mathbb{Z}^2$ (c'est en fait la première étape de l'algorithme pour simuler un processus de Poisson), ce que nous avons mis en œuvre pour notre implémentation CUDA de l'évaluation parallèle du SND sur $\mathbb{Z}^2$ associé au texton orienté synthèse présenté au chapitre précédent (voir Section 2.3). Enfin, nous avons très récemment eu l'opportunité de réutiliser notre implémentation OpenGL pour un tout autre projet, à savoir la simulation du grain des pellicules argentiques sur des images numériques [S16]. En effet, comme on le verra à la Section 4.1.3 du Chapitre 4, à l'échelle microscopique les images provenant de photographies argentiques sont des ensembles binaires aléatoires que l'on peut modéliser par des modèles booléens poissonniens (*i.e.* des union de disques placés selon un processus ponctuel de Poisson). Le rendu final de telles images nécessite alors une évaluation intensive de

ces ensembles binaires, et donc du processus de Poisson sous-jacent. Grâce à notre implémentation OpenGL de simulation à la demande du processus de Poisson, nous avons gagné plusieurs ordres de grandeur en temps de calcul en comparaison à notre rendu sur CPU.

Rappelons pour commencer la définition formelle d'un processus de Poisson stationnaire sur $\mathbb{R}^2$.

Définition 3.1 (Processus ponctuel de Poisson (*e.g.* [Baddeley, 2007]))**.** *Un processus ponctuel $\Phi \subset \mathbb{R}^2$ est un processus ponctuel de Poisson d'intensité λ si*

(i) Pour tout domaine borné B (ensemble mesurable), le nombre $N(B) := \#\Phi \cap B$ de points de Φ inclus dans B suit une loi de Poisson de paramètre $\lambda \mathcal{L}^2(B)$, où $\mathcal{L}^2(B)$ désigne la mesure de Lebesgue du domaine B.

(ii) Si B_1, B_2, ..., B_n sont des domaines disjoints, alors les variables aléatoires $N(B_1)$, $N(B_2)$, ..., $N(B_n)$ sont indépendantes.

A première vue cette définition ne permet pas de voir qu'un processus de Poisson est un ensemble de points "uniformément répartis sur $\mathbb{R}^2$". Toutefois on montre facilement que conditionnellement à $N(B) := \#\Phi \cap B = n$, les n points $(x_1, \ldots, x_n)$ de $\Phi \cap B$ sont indépendants et uniformément répartis dans B (voir *e.g.* [Schneider and Weil, 2008, Theorem 3.2.2]). Ceci permet de simuler facilement un processus de Poisson d'intensité λ sur un domaine fini Ω. La simulation s'effectue en deux étapes : on tire d'abord un nombre de points $N \in \mathbb{N}$ suivant une loi de Poisson de paramètre $\lambda \mathcal{L}^2(\Omega)$, où $\mathcal{L}^2(\Omega)$ désigne la mesure de Lebesgue du domaine Ω, puis on tire ensuite indépendamment N points x_i selon la loi uniforme sur le domaine Ω. De plus, grâce à la propriétés d'indépendance sur des domaines disjoints, la réunion de deux processus de Poisson indépendants d'intensité λ sur deux domaines disjoints Ω_1 et Ω_2 est un processus de Poisson sur l'union $\Omega_1 \cup \Omega_2$. On peut donc simuler un processus de Poisson sur un domaine fini et étendre *a posteriori* cette simulation sur un domaine voisin si nécessaire.

Cependant en computer graphics il est nécessaire de disposer de procédure qui soit le plus parallèle possible pour une exécution rapide sur GPU. De plus le domaine de simulation du processus de Poisson n'est pas connu à l'avance (parcours inconnu de la caméra, changement d'échelle, etc.). Ainsi, il est primordial de disposer d'une procédure qui puisse évaluer à différents endroits du plan la réalisation d'un même processus de Poisson sur $\mathbb{R}^2$. La (très élégante) solution est de s'appuyer sur une partition du plan $\mathbb{R}^2$ en cellules carrées disjointes $C_k = a(k + [0,1[^2)$, $k \in \mathbb{Z}^2$, de côté $a > 0$, et d'associer à chaque cellule C_k son propre générateur de nombres pseudo-aléatoires (PRNG en anglais pour *pseudo-random number generator*). Ceci permet de s'assurer que la séquence de nombres pseudo-aléatoires utilisée pour chaque simulation du processus de Poisson dans une cellule C_k soit reproduisible. Comme un PRNG est déterminé par une *seed* (graine) entière, en utilisant une fonction injective seed : $\mathbb{Z}^2 \to \mathbb{N}$, il est alors possible de définir un PRNG pour chaque cellule C_k par $\text{PRNG}_k = \text{PRNG}(\text{seed}(k))$. A partir de cette partition $\mathbb{R}^2 = \bigcup_{k \in \mathbb{Z}^2} a(k+[0,1[^2)$ en cellules $C_k = a(k+[0,1[^2)$ et de ces PRNG associés on peut alors

simuler les points du processus de Poisson sur n'importe quel domaine carré avec un coût constant proportionnel au nombre de cellules qui intersectent le domaine. La procédure complète est décrite en détail par l'Algorithme 2. Remarquons que comme les PRNG sont déterministes pour une graine seed donnée, on peut lancer l'Algorithme 2 en parallèle sur deux voisinages proches et les points de l'intersection seront bien les mêmes. Ce type d'approche est typique des textures procédurales et des contraintes liées au GPU. En effet, il est plus efficace et plus simple de recalculer les points à chaque fois que nécessaire plutôt que de les calculer une fois et de les stocker en mémoire.

- Entrées : Point d'intérêt $y = (y_1, y_2) \in \mathbb{R}^2$, rayon $r > 0$ du voisinage carré, $\lambda > 0$ intensité du processus de Poisson, taille de cellules $a > 0$
- Sortie : La restriction A du processus de Poison Φ au carré $S = [y_1 - r, y_1 + r] \times [y_2 - r, y_2 + r]$
- Initialiser A à l'ensemble vide : $S \leftarrow \emptyset$
- Simuler Φ sur chaque cellule intersectée par par le domaine carré S :
  Pour $k_1 \in \left\{ \left\lfloor \frac{y_1 - r}{a} \right\rfloor, \ldots, \left\lfloor \frac{y_1 + r}{a} \right\rfloor \right\}$ et $k_2 \in \left\{ \left\lfloor \frac{y_2 - r}{a} \right\rfloor, \ldots, \left\lfloor \frac{y_2 + r}{a} \right\rfloor \right\}$
  1. Initialiser le PRNG local PRNG_k avec la graine $\text{seed}(k_1, k_2)$
  2. Tirer à l'aide de PRNG_k le nombre de points N dans la cellule C_k suivant une loi de Poisson de paramètre λa^2
  3. Pour $n = 1$ à N, tirer à l'aide de PRNG_k un point x uniformément dans C_k et ajouter x à A si $x \in S$.
- Retourner A.

Algorithme 2 : Simulation à la demande d'un processus de Poisson sur un voisinage carré d'un point

Remarque (Injectivité de la fonction seed)**.** En pratique la fonction seed est injective modulo la restriction de la représentation des entiers sur GPU (compris entre 0 et $2^{32} - 1$). Une solution élémentaire 2^{16}-périodique est

$$\text{seed}(k_1, k_2) = 2^{16}(k_1 \mod 2^{16}) + (k_2 \mod 2^{16}).$$

Toutefois cette fonction seed est trop régulière spatialement et il est nécessaire de lui appliquer une fonction de hachage en sortie pour décorréler le PRNG des cellules voisines. On renvoie au très instructif article de blog [Reed, 2013] pour une illustration visuelle.

Remarque (Choix de la taille des cellules)**.** Pour le Gabor noise [Lagae et al., 2009] le paramètre a de la taille des cellules avait été fixée à $a = 2r$ où r est le rayon du support d'une gaussienne tronquée. En implémentant l'Algorithme 2 pour l'article [J11] nous avons remarqué que ce paramètre a était assez sensible. En effet,

pour $a = 2r$, alors le carré $S = [y_1 - r, y_1 + r] \times [y_2 - r, y_2 + r]$ intersecte quatre cellules C_k de même taille que S et on simule donc en moyenne quatre fois trop de points. En diminuant la taille a des cellules, on limite le nombre de points simulés en dehors de S, mais on augmente par ailleurs le nombre de cellules intersectant S et donc le nombre de variables de Poisson à simuler. Pour une intensité $\lambda = 30/r^2$, nous avons déterminé expérimentalement qu'une taille $a = (1.2)r < 2r$ permettait d'obtenir un gain de performance de 30% par rapport à la taille $a = 2r$.

3.3 Synthèse par l'exemple de textures procédurales de type *Gabor noise*

Cette section résume l'article *Gabor noise by example* [J5]. Les motivations de ce travail ont déjà été discutées dans l'introduction de ce chapitre (voir Section 3.1). On renvoie également à l'article [J5] pour un état de l'art complet sur le sujet.

3.3.1 Le modèle Gabor noise

Nous commençons par présenter le modèle *Gabor noise* élémentaire comme introduit dans [Lagae et al., 2009, Lagae and Drettakis, 2011]. On rappelle qu'un noyau de Gabor est une sinusoïde modulée par une enveloppe gaussienne, de sorte que la transformée de Fourier d'un noyau de Gabor est la somme d'un couple de gaussiennes symétriques par rapport à l'origine.

Le *Gabor noise* est une simple somme de noyaux de Gabor placés sur les points d'un processus de Poisson. C'est donc un processus shot noise avec un noyau de Gabor, et en *computer graphics* ce modèle est appelé *sparse convolution noise* [Lewis, 1984] ou encore *spot noise* [van Wijk, 1991]. Par rapport au modèle initial [Lagae et al., 2009], il est nécessaire d'ajouter une phase aléatoire aux sinusoïde des noyaux de Gabor (ce qui revient à décorréler le maximum de la sinusoïde et le maximum de l'enveloppe gaussienne) afin d'obtenir un spectre de puissance qui soit une somme de gaussiennes symétriques. J'avais observé l'intérêt de l'ajout d'une phase aléatoire pendant mes travaux de thèse [T15] et cela a été utilisé en parallèle par A. Lagae et G. Drettakis [Lagae and Drettakis, 2011] qui ont montré qu'avec l'ajout d'une phase aléatoire les sections 2D de Gabor noise 3D restaient encore des Gabor noise 2D.

Nous présentons maintenant les équations relatives à ce modèle afin d'introduire les notations et le vocabulaire nécessaire pour la section suivante. Le noyau de Gabor (avec phase) est noté

$$g\left(x; K, a, \omega, \phi\right) = K e^{-\pi a^2 |x|^2} \cos\left(2\pi x \cdot \omega + \phi\right), \quad x \in \mathbb{R}^2,$$

où

— $K > 0$ est l'amplitude du noyau,

— $a > 0$ est la bande passante du noyau,

— $\omega \in \mathbb{R}^2$ est la fréquence du noyau,

— $\phi \in \mathbb{R}$ est la phase du noyau.

Le *Gabor noise* élémentaire est alors défini de la manière suivante.

Définition 3.2. *Soit* $\Phi = \{(x_i, \phi_i),\ x_i \in \mathbb{R}^2,\ \phi_i \in [0, 2\pi]\}$ *un processus de Poisson sur* $\mathbb{R}^2$ *d'intensité* λ *indépendamment marqué par des phases aléatoires uniformément distribuées sur* $[0, 2\pi]$. *Le* Gabor noise *(normalisé) d'amplitude* $K > 0$, *bande passante* $a > 0$ *et fréquence* $\omega \in \mathbb{R}^2$ *est le shot noise*

$$n(x; K, a, \omega) = \frac{1}{\sqrt{\lambda}} \sum_{(x_i,\phi_i)\in\Phi} g(x - x_i; K, a, \omega, \phi_i), \quad x \in \mathbb{R}^2. \tag{3.1}$$

Le *Gabor noise* élémentaire est donc un shot noise sur un processus de Poisson marqué. Les expressions de son espérance, sa covariance et son spectre de puissance sont facilement calculées à partir de la moyenne, l'autocorrélation et le module au carré de la transformée de Fourier de la fonction noyau (voir par exemple [T15, Proposition 3.1]). On trouve alors que l'espérance de n est nulle, sa variance est $K^2/4a^2$ et que son spectre de puissance a pour expression

$$S_n(\xi; K, a, \omega) = \frac{K^2}{8a^2} \mathcal{G}\left(\xi; \pm\omega, \frac{a}{2\sqrt{pi}}\right) \tag{3.2}$$

où $\mathcal{G}(\xi; \mu, \sigma)$ est la fonction de densité d'une distribution gaussienne 2D de moyenne μ et de variance σ et $\mathcal{G}(\xi; \pm\mu, \sigma) = \mathcal{G}(\xi; \mu, \sigma) + \mathcal{G}(\xi; -\mu, \sigma)$ est une abréviation pour la somme du couple de gaussiennes symétriques de moyenne $\pm\mu$.

L'évaluation procédurale du *Gabor noise* n nécessite d'effectuer une approximation de la somme infinie (3.1). Pour cela, les noyaux de Gabor sont tronqués et mis à zéro en dehors du disque de rayon a^{-1} [Lagae et al., 2009]. Ainsi l'évaluation en un point x se fait alors en sommant simplement les contributions des points x_i contenus dans le disque de centre x et de rayon a^{-1} qui sont simulés à la demande grâce à l'Algorithme 2 avec une taille de grille de l'ordre du rayon a^{-1}.

Pour finir, rappelons que dans ce cadre de shot noise poissonniens définis sur $\mathbb{R}^2$, on a là encore convergence du champ aléatoire vers un champ gaussien lorsque λ tends vers $+\infty$ [Papoulis, 1971]. Pour une intensité λ assez grande, les réalisations de n sont donc approximativement des champs gaussiens ayant un spectre de puissance égal à S_n. Ces spectres sont localisés autour des fréquences $\pm\omega$ avec une largeur de bandes déterminée par a. C'est cette double localisation spatiale et fréquentielle qui fait l'intérêt du *Gabor noise* : la localisation spatiale permet de tronquer les noyaux pour avoir une simulation procédurale rapide et la localisation fréquentielle permet de contrôler l'orientation et la fréquence des pattern de bruits. Aussi, la forme explicite du spectre permet d'implémenter des technique de "filtrage analytique" (*analytic filtering*) [Lagae et al., 2009] qui consiste à effectuer des atténuations fréquentielles adaptées à la distance et à l'orientation par rapport à la caméra afin d'éliminer en amont les effets d'aliasing.

3.3.2 Gabor noise by example

Comme on vient de le voir, un Gabor noise élémentaire permet finalement de simuler approximativement un champ gaussien dont le spectre de puissance est une paire de gaussiennes symétriques. Et réciproquement toute paire de gaussiennes symétriques correspond à un certain Gabor noise élémentaire. Enfin lorsque l'on somme des champs aléatoires indépendants le spectre de puissance final est égal à la somme des spectres de puissance. Finalement l'idée directrice de [J5] est de décomposer le spectre d'une image de texture en une somme de couples de gaussiennes symétriques par rapport à l'origine.

Cette idée invite à résoudre un problème d'optimisation permettant d'approcher le spectre de puissance d'une image discrète comme une somme de paires de gaussiennes, puis de sommer les différents noyaux de Gabor correspondants à chacune de ces paires de gaussiennes. Cependant on se confronte à plusieurs difficultés. Premièrement, le spectre de puissance de l'image d'entrée est discret alors que le spectre de puissance du bruit est lui continu. Deuxièmement, les paramètres de fréquence ω et de bande passante a apparaissant dans l'expression du spectre de puissance S_n (3.2) sont complètement non-linéaires. Enfin, même si l'on était en mesure de calculer tous ces paramètres optimaux, l'évaluation procédurale de la somme des noyaux de Gabor ne serait pas efficace car il existe un certain coût de calcul pour simuler un processus de Poisson différent pour chaque gaussienne. On comprend alors la réelle difficulté rencontrée : il faut élaborer un modèle assez générique pour représenter tous les spectres de puissance, pour lequel on soit capable de déterminer les paramètres, et qui soit évaluable de manière procédurale en des temps raisonnables.

On observe premièrement que l'on peut se contenter de comparer le spectre de puissance discret de l'image à l'échantillonnage selon la même grille du spectre du bruit et que l'on peut limiter les fréquences ω à vivre sur cette grille. Enfin, la solution proposée a été de quantifier selon des valeurs dyadiques la gamme de bandes passantes a. Le gain a alors été double. Par rapport à l'évaluation des paramètres, une fois les fréquences et les bandes passantes quantifiées, la donnée d'un spectre n'est plus que linéaire par rapport au paramètre d'amplitude au carré K^2 ce qui permet d'utiliser des méthodes standard d'optimisation. Par ailleurs, en ce qui concerne l'évaluation procédurale, l'utilisation d'un nombre fini de bandes passantes permet de regrouper les différents *Gabor noise* utilisant les mêmes bandes passantes (mais avec des fréquences différentes) afin de ne simuler qu'un nombre restreint de processus de Poisson ayant des tailles de grilles différentes. C'est pour cela que nous avons appelé le modèle de bruit *Gabor noise* à bandes passantes quantifiées.

Il reste néanmoins une difficulté : la représentation d'un bruit procédural doit être compact en mémoire. Cela signifie que le nombre de paramètres doit resté limité. Ce nombre de paramètres correspond ici au nombre de gaussiennes utilisées pour décomposer le spectre. Afin d'obtenir un nombre de gaussiennes limité, nous avons modélisé le problème comme un problème de moindre carré avec une

contrainte de positivité et une régularisation pour une norme ℓ_1 afin d'imposer un certain degré de parcimonie pour le vecteur des amplitudes au carré $\alpha = K^2$ (qui est de taille nombre de fréquences discrètes MN par nombre de bandes passantes B, avec typiquement $B = 6$ bandes passantes). En notant $\alpha \mapsto S(\alpha)$ l'opérateur linéaire discret calculant la somme de gaussiennes symétriques ayant différentes bandes passantes dyadiques avec les poids $\alpha(k, l, b) \geq 0$, on cherche à résoudre

$$\min_{\alpha \geq 0} \|S(\alpha) - S_{\text{ex}}\|_2^2 + \nu \|\alpha\|_1. \tag{3.3}$$

C'est un problème convexe appelé *Non-Negative Basis Pursuit Denoising* (NNBPDN) que nous avons résolu grâce à l'algorithme itératif FISTA [Beck and Teboulle, 2009] avec un critère d'arrêt contrôlant l'écart au problème dual (*duality gap*) comme proposé par [Kim et al., 2007, Mairal et al., 2011].

Comme toujours, l'introduction d'un tel problème variationnel régularisé pose la question du réglage du poids $\nu > 0$ du terme de régularisation qui ici détermine le degré de parcimonie de la solution. Cette question est centrale en traitement d'images, et les paramètres optimaux sont même devenus des objets d'étude, *e.g.* [Kunisch and Pock, 2013, Reyes et al., 2016]. Il est important de remarquer qu'en graphisme la philosophie vis à vis des paramètres est assez différente. Il y a bien sûr des paramètres qu'il est bon de régler automatiquement si ceux-ci assurent la robustesse des algorithmes. En revanche, il y a également des paramètres qu'il est préférable de laisser à l'utilisateur dès lors que ceux-ci permettent un compromis entre le temps de calcul et la qualité visuelle. Nous allons illustrer que nous nous trouvons ici dans cette deuxième situation. Afin d'être plus adaptatif, on introduit un paramètre de régularisation relatif $\kappa \in [0, 1]$ défini par $\kappa\nu_{\max}$, où $\nu_{\max}$ est la plus petite valeur de ν pour laquelle $\alpha = 0$ est une solution de (3.3), à savoir, $\nu_{\max} = 2 \left\| S_n^T \left(S_{ex} \right) \right\|_\infty$ [Kim et al., 2007]. On résout ensuite le problème (3.3) en suivant la stratégie par homotopie, à savoir, on utilise une séquence décroissante de paramètre $\kappa_i = 1/2^i$ en initialisant chaque problème par le résultat du problème précédent. Cela rend la résolution plus efficace, et a l'avantage de fournir une gamme de solutions de moins en moins parcimonieuses, comme illustré par la Figure 3.1, où l'on voit que les textures correspondantes sont visuellement enrichies à chaque étape. L'utilisateur peut alors choisir à quel étage la qualité de la texture est suffisamment bonne pour lui, sachant qu'il paie un coût mémoire et temps de calcul pour la synthèse. Néanmoins, pour des images 128×128, nous avons remarqué que $\kappa = 1/256$ était un bon compromis et tous les résultats qui suivent sont obtenus avec ce paramètre.

L'évaluation procédurale du *Gabor noise* à bandes passantes quantifiées a nécessité de relever plusieurs défis techniques. Nous ne rentrerons pas dans les détails ici car ce travail a été essentiellement effectué par A. Lagae qui était un charge de la partie GPU sur ce projet. On retiendra tout de même l'utilisation d'un processus de Poisson différent pour chaque bande passante et également la nécessité de faire un tirage aléatoire des fréquences des noyaux de Gabor parmi les coefficients actifs $\alpha \neq 0$ déterminés par (3.3). Pour limiter le coût de calcul nous avons eu recourt

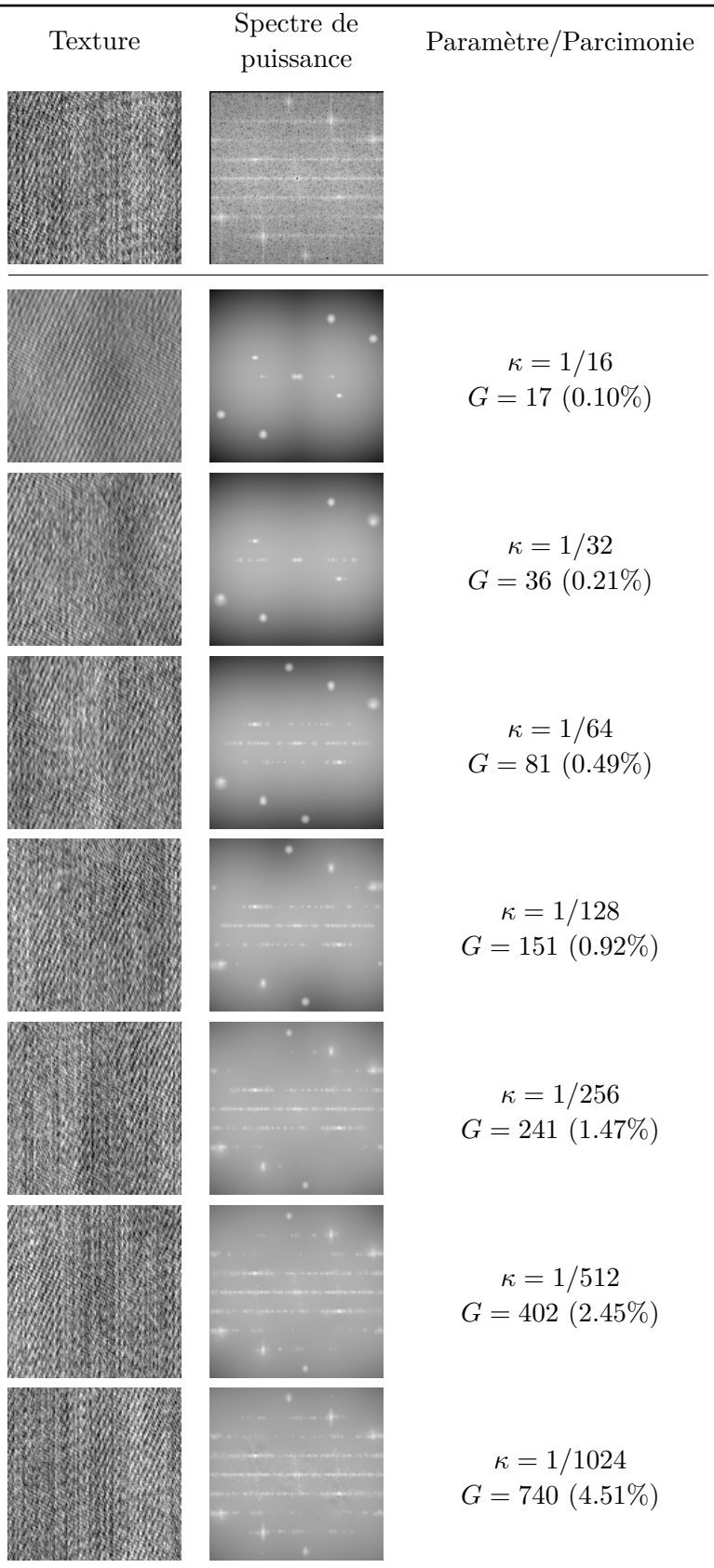


FIGURE 3.1 – Estimation des paramètres pour le *Gabor noise* par l'exemple. La première ligne montre l'image d'exemple et son spectre de puissance estimé. Les lignes suivantes montrent les décompositions en somme parcimonieuse de G paires de gaussiennes avec une diminution du paramètre de régularisation κ et les textures gaussiennes associées à ces spectres.

au tirage aléatoire par la méthode d'*alias* de [Walker, 1977]. Le principe de cette méthode de simulation de variable discrètes (qui mériterait d'être plus enseignée, plutôt grâce à l'article de Kronmal et Peterson [Kronmal and A. V. Peterson, 1979] que l'article original) est de limiter les nombres de rejets à un rejet maximum en calculant un tableau pour associer à chaque valeur un "suppléant" (*alias*) en cas de rejet. Le calcul du tableau est linéaire, et, une fois calculé, la simulation des variables est à coût constant (limité au tirage de une ou deux variables uniformes sur $[0, 1]$ et à la lecture de une ou deux cases du tableau d'*alias*). Ceci est très important pour la synthèse sur GPU. En ce qui concerne la complexité de l'évaluation procédurale, nous avons remarqué expérimentalement qu'il fallait sommer en moyenne environ 1000 noyaux de Gabor sur chaque points afin d'obtenir une texture proche de la texture gaussienne limite.

Enfin on mentionne rapidement que nous avons cherché dans l'article [J5] à calculer l'espace couleur adaptatif permettant de décorréler au mieux toutes les corrélations croisées des canaux couleurs de l'image d'entrée, à savoir les matrices

$$C_{\boldsymbol{h}}(x) = \sum_{y \in \Omega} (\boldsymbol{h}(x+y) - \boldsymbol{m})(\boldsymbol{h}(y) - \boldsymbol{m})^T \in \mathbb{R}^{3\times 3}$$

introduites à la Section 2.2.1. Le but de cette approche est de faire une synthèse gaussienne scalaire indépendante sur chaque canal couleur une fois les canaux décorrélés plutôt que d'effectuer une décomposition spectrale d'un spectre de puissance RGB. Cette technique est utilisée sur le canaux ACP du nuage de couleur dans [Heeger and Bergen, 1995], c'est-à-dire en ne diagonalisant que $C_{\boldsymbol{h}}(0)$. Nous avons résolu un problème de diagonalisation simultanée approchée avec l'algorithme de [Cardoso and Souloumiac, 1996]. Cette approche fonctionne mais force est de constater que la base de diagonalisation simultanée est toujours très proche de la base ACP pour les textures gaussiennes, et donc que l'intérêt pratique de cette technique est assez limité. En revanche, nos résultats expérimentaux montrent que l'on ne peut pas faire beaucoup mieux que la diagonalisation ACP, ce qui justifie l'utilisation de cette base pour décorréler des canaux couleurs de textures proches de textures gaussiennes. Enfin, comme on le verra dans la prochaine section, notre nouvel algorithme *texton noise* gère les couleurs de manière beaucoup plus simple et intuitive avec un simple shot noise dans l'espace RGB.

Résultats Le Figure 3.2 présente de manière détaillée deux résultats de synthèse pour le *Gabor noise* par l'exemple avec deux images intermédiaires. La première image est la texture d'entrée, la deuxième est sa version gaussienne (obtenue par simulation d'un BPA, voir Chapitre 2). La version gaussienne est la qualité cible pour le résultat final. La troisième image est, comme pour la Figure 3.1, une texture obtenue par BPA à partir du spectre déterminé par l'algorithme de décomposition en somme de gaussiennes symétriques. La différence entre la deuxième image et la troisième souligne les approximations dues à cette décomposition. La quatrième image est le *Gabor noise* généré sur GPU avec notre implémentation procédurale codée en CUDA. La différence entre la troisième image et la quatrième souligne les

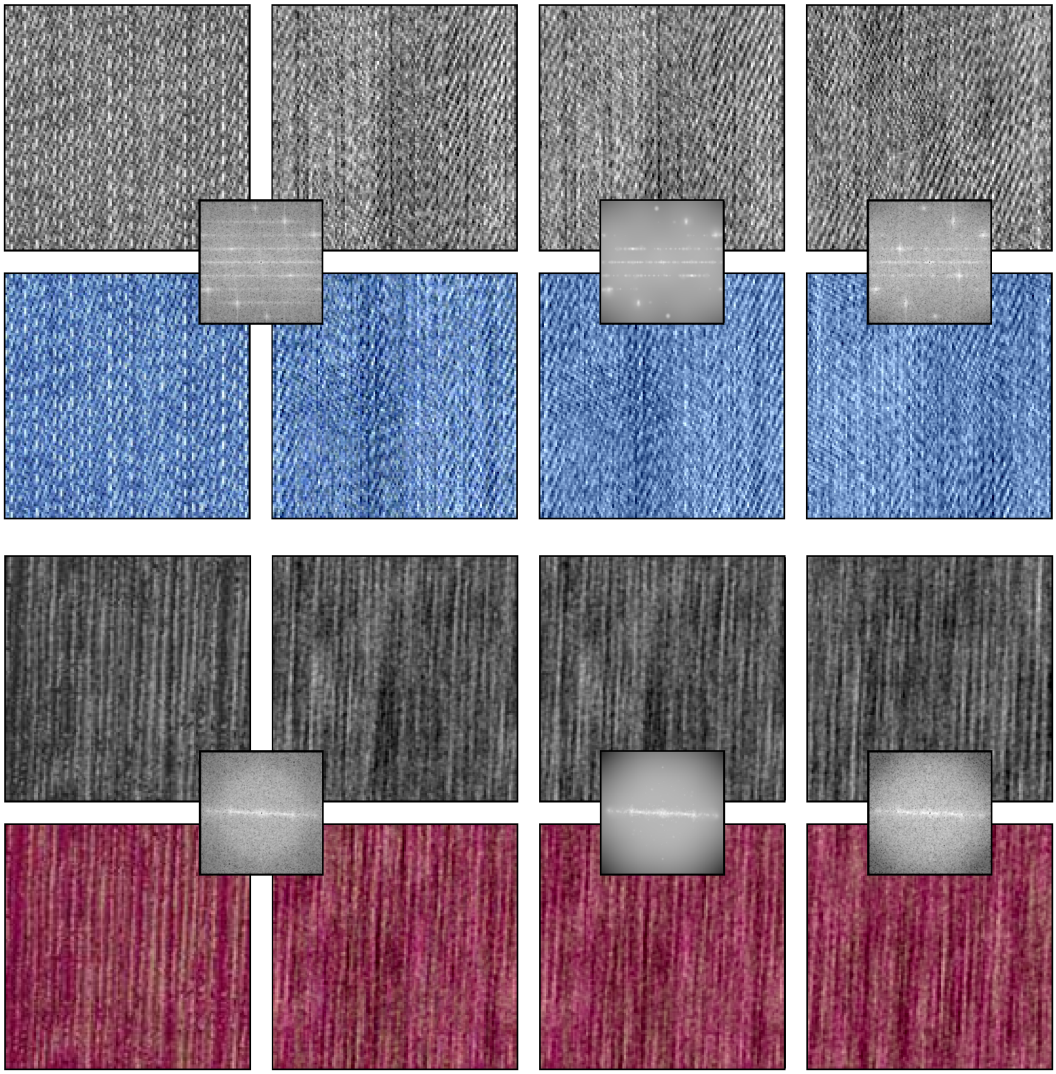


FIGURE 3.2 – Résultats détaillés pour le *Gabor noise* par l'exemple pour deux textures en version niveau de gris et en version couleur. La première et deuxième colonne montrent respectivement l'image d'entrée et sa version gaussienne qui partage le même spectre (en gris sur la ligne intermédiaire). La troisième colonne contient le spectre calculé par le problème NNBPDN et une réalisation d'une texture gaussienne ayant ce spectre. La quatrième colonne contient une évaluation procédurale du *Gabor noise* avec une estimation de son spectre.

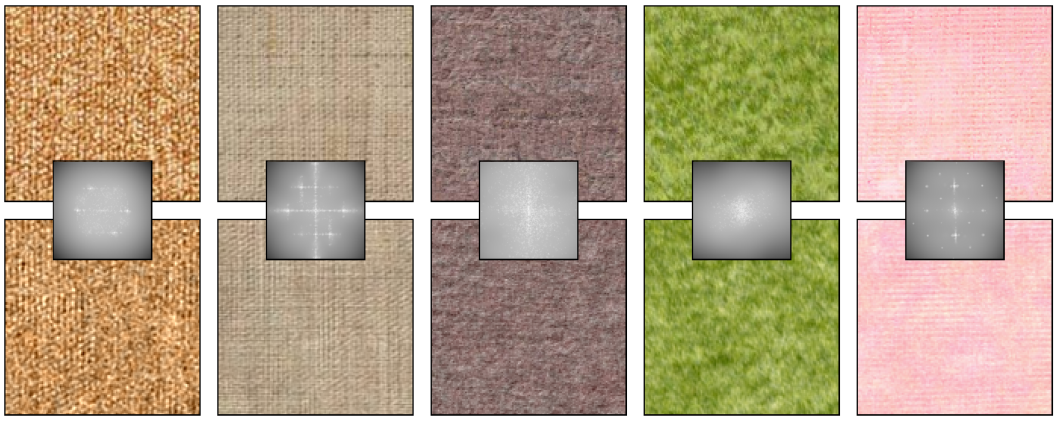


FIGURE 3.3 – Résultat du *Gabor noise* par l'exemple pour différentes textures gaussiennes, avec au centre le spectre de puissance déterminé pour la composante couleur principale.

approximations dues à l'utilisation d'un shot noise fini (1024 noyaux de Gabor sur chaque point en moyenne) pour approcher la limite gaussienne. On montre aussi les spectres calculés ou estimés pour comparer le contenu fréquentiel. On remarque que la méthode est robuste et que chaque approximation est visuellement négligeable. Nous montrons d'autres résultats satisfaisants sur des textures gaussiennes à la Figure 3.3 ainsi que des résultats sur des textures non gaussiennes à la Figure 3.4 qui pointent les limites dues au modèle gaussien déjà discutée à la Section 2.2. Comme le BPA et le SNDA du Chapitre 2, le *Gabor noise* par l'exemple reproduit la version gaussienne des textures d'entrée.

Le lecteur septique pourra objecter que rien ne prouve que ces textures sont constituées de noyaux de Gabor, ce qui est une réussite de la méthode. Nous présentons à la Figure 3.5 la convergence du Gabor noise vers sa texture gaussienne limite lorsque le nombre de noyaux de Gabor augmente. Ceci répare un tort de l'article [J5] où l'on parle beaucoup de noyau de Gabor sans jamais en montrer un seul...

Performances et limitations Comme rappelé en introduction, l'exécution de notre implémentation Matlab de la résoluation de (3.3) par l'algorithme FISTA prend deux minutes pour une image 128×128 (chaque appel à l'opérateur linéaire S ou son adjoint a le coût de B convolutions par FFT). L'évaluation procédurale du *Gabor noise* prend elle une seconde pour une image full HD 1920×1080. Par ailleurs, cette évaluation est faite de manière procédurale mais sur une implémentation CUDA, et donc non intégrée dans un pipeline de visualisation de scène 3D virtuelle comme OpenGL. En effet, la complexité des listes de paramètres et l'utilisation de plusieurs méthodes d'alias nécessitaient des outils de contrôle précis sur la mémoire de texture sur GPU, ce qui était plus facile à gérer avec les fonctionnalités de CUDA. Mais au final, les textures procédurales du *Gabor noise* par l'exemple

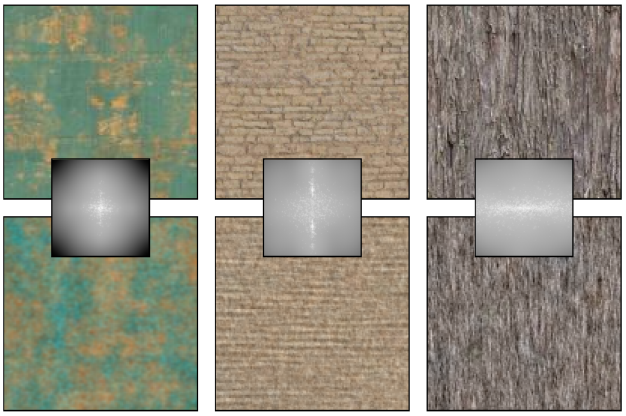


FIGURE 3.4 – Même configuration qu'à la Figure 3.3 avec en entrée trois textures non gaussiennes.

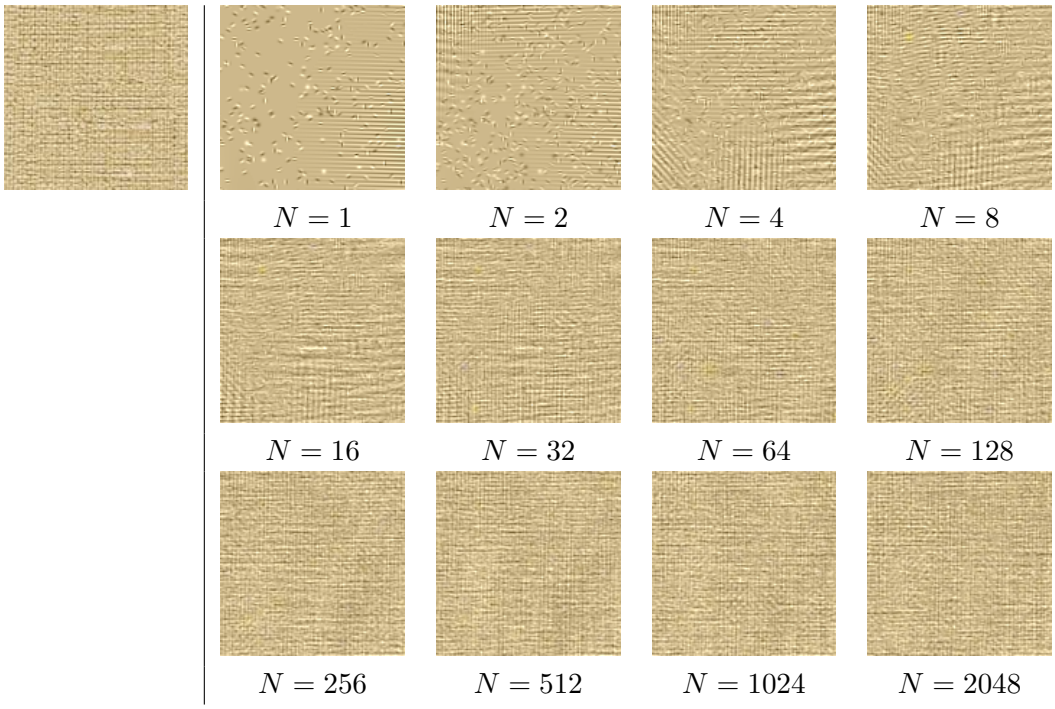


FIGURE 3.5 – Convergence du *Gabor noise* vers sa limite gaussienne. Pour passer d'une image à l'autre on ajoute (en moyenne) deux fois plus de noyaux de Gabor (avec la même séquence). On remarque que la qualité de la texture devient acceptable pour $N = 256$ pour cet exemple (N est la nombre moyen de noyaux de Gabor recouvrant un point). Dans toutes nos expériences on utilise $N = 1024$.

n'ont jamais été utilisées dans un environnement 3D virtuel. Aussi, même si théoriquement le filtrage analytique [Lagae et al., 2009] peut s'appliquer au *Gabor noise*, il n'a pas été implémenté sur les *Gabor noise* complexes utilisés pour la synthèse par l'exemple, et on peut avancer que sa mise en œuvre sera assurément technique et que les coûts de calculs seront prohibitifs. Comme on va le voir à la section qui suit, ces problèmes sont largement résolus par le *texton noise*.

3.4 *Texton noise*, un bruit procédural minimaliste pour la synthèse de textures gaussiennes

Comme on vient de voir, le Gabor noise par l'exemple permet de reproduire une texture gaussienne quelconque par un bruit procédural. Cependant, la complexité du modèle déterminé à l'aide d'une décomposition du spectre pose des problèmes pratiques de performance pour l'évaluation de la texture procédurale obtenue. Le but principal du *texton noise* est de proposer le modèle de texture procédurale le plus simple qui permette de reproduire toutes les textures gaussiennes. Sur le plan personnel, ce travail en collaboration avec A. Leclaire et L. Moisan a représenté un important investissement puisque j'ai dû apprendre à coder des bruits procéduraux en OpenGL. Cela n'aurait pas été possible si A. Lagae et G. Drettakis n'avaient pas mis en ligne leur code OpenGL pour le filtrage des Gabor noise tridimensionnels [Lagae and Drettakis, 2011]. Comme on le verra dans la discussion, la vitesse d'exécution de l'algorithme étant cent fois plus rapide que celle du Gabor noise par l'exemple, on peut dire que cet investissement a été fructueux, d'autant plus qu'une partie des codes a pu être adaptée pour la génération de modèle booléen (voir Section 4.1.3). Cette expérience m'a définitivement convaincu qu'il est intéressant de développer des implémentations sur cartes graphiques à l'aide de CUDA ou OpenGL dès lors que les algorithmes étudiés s'y prêtent.

Nous présentons maintenant en détail les différentes contributions de notre article *Texton Noise* soumis au journal Computer Graphics Forum [J11].

3.4.1 Un shot noise à noyau fixe

Le texton noise est un simple shot noise poissonnien qui utilise une seule fonction noyau, appelée *texton* (pour rappel, en *computer graphics* on parle plutôt de *sparse convolution noise* [Lewis, 1984] ou *spot noise* [van Wijk, 1991]). Le texton détermine à lui seul le spectre de puissance du bruit procédural. Afin d'assurer une évaluation rapide du bruit, cette fonction noyau est modélisée par une interpolation bilinéaire d'une certaine image ayant un support carré de petite taille. L'intérêt de ce choix est que ce type de fonctions est très facilement évaluable sur GPU à l'aide des opérations de *fetching* de textures GPU. En effet, par défaut, sur GPU l'évaluation des images stockées dans la mémoire de texture (*texture memory*) renvoie les valeurs de l'interpolation bilinéaire de l'image.

Notre contribution principale est de montrer que ce simple modèle shot noise

permet de reproduire toutes les texture gaussiennes, avec un algorithme d'analyse et une évaluation du bruit procédural qui sont un ou deux ordres de grandeur plus rapides que les méthode concurrentes [J5] [Gilet et al., 2014]. Par ailleurs, la méthode est sans paramètre et nous avons mis à disposition en ligne nos implémentations (codes matlab pour l'analyse et codes OpenGL pour la synthèse).

Du point de vue du modèle on considère donc un shot noise sur un processus de Poisson Φ_λ d'intensité $\lambda > 0$ sur $\mathbb{R}^2$ noté

$$f_\lambda(x) = \sum_{x_j \in \Pi_\lambda} h(x - x_j), \quad x \in \mathbb{R}^2, \tag{3.4}$$

où h est une interpolation bilinéaire d'une certaine image $\alpha : \mathbb{Z}^2 \to \mathbb{R}^d$ ($d = 1$ pour les images en niveaux de gris et 3 pour les images couleurs) à support fini, c'est-à-dire

$$h(y) = \sum_{k \in \mathbb{Z}^2} \alpha(k)\psi(y - k), \quad y \in \mathbb{R}^2, \tag{3.5}$$

où ψ est le noyau d'interpolation bilinéaire (*i.e.* $\psi(y_1, y_2) = (1 - |y_1|)^+(1 - |y_2|)^+$ où $x^+ = x$ si $x \geq 0$ et 0 sinon). On sait alors que le shot noise normalisé

$$g_\lambda(x) = \frac{f_\lambda(x) - \mathbb{E}(f_\lambda(x))}{\sqrt{\lambda}} \tag{3.6}$$

converge vers un champ gaussien de moyenne nulle et de fonction de covariance

$$C(y) = \tilde{h} * h(y) = \sum_{k \in \mathbb{Z}^2} (\tilde{\alpha} \star \alpha)(k) \left(\tilde{\psi} * \psi\right)(y - k)$$

(dans cette section, on différencie l'opérateur de convolution $*$ entre fonctions sur $\mathbb{R}^2$ et l'opérateur $\star$ de convolution entre images discrètes définies sur $\mathbb{Z}^2$ ou sur un domaine rectangulaire $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$ avec condition de bords périodique).

Etant donnée une image $u : \Omega \to \mathbb{R}^d$, le spot noise discret asymptotique sur $\mathbb{Z}^2$ $\mathrm{SNDA}_{\mathbb{Z}^2}(u)$ introduit à la Section 2.3 fournit un modèle de texture gaussienne associée à u sur le plan discret. En revanche, on cherche ici un modèle de texture sur le plan continu $\mathbb{R}^2$. Afin de garantir que le bruit procédural g_λ corresponde à un champ gaussien visuellement similaire à u (ou du moins sa version gaussienne), on impose que l'échantillonnage de g_λ selon n'importe quelle grille $y + \mathbb{Z}^2$, $y \in [0, 1[^2$, ait la même covariance que $\mathrm{SNDA}_{\mathbb{Z}^2}(u)$. Cela se traduit par

$$\tilde{h} * h(k) = \sum_{l \in \mathbb{Z}^2} (\tilde{\alpha} \star \alpha)(l) \left(\tilde{\psi} * \psi\right)(k - l) = \tilde{t}_u \star t_u(k), \quad k \in \mathbb{Z}^2,$$

Comme $\tilde{\psi} * \psi = \psi * \psi$ est le noyau $\boldsymbol{B}$-spline d'ordre 3, en notant b l'échantillonnage sur $\mathbb{Z}^2$ de ce noyau (b est nul en dehors de $\{-1, 0, 1\}^2$ et vaut $\begin{pmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{pmatrix}^T \begin{pmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{pmatrix}$ sur ce domaine) et en prenant la transformée de Fourier sur $\mathbb{Z}^2$ on obtient

$$|\hat{\alpha}(\xi)|^2 \hat{b}(\xi) = \left|\hat{t}_u(\xi)\right|^2, \quad \xi \in [-\tfrac{1}{2}, \tfrac{1}{2}]^2. \tag{3.7}$$

Les termes $\left|\hat{t}_u(\xi)\right|^2$ et $\hat{b}(\xi)$ étant des polynômes trigonométriques, on voit qu'en général $|\hat{\alpha}(\xi)|^2$ n'est pas un polynôme trigonométrique et donc que si une solution α existe alors le support de α est infini. On est donc amené à considérer des approximations afin de pouvoir prescrire le support de α.

3.4.2 Algorithme de calcul du texton

Comme il n'y a pas de solution exacte pour l'Equation (3.7), on cherche une solution approchée à ce problème. Etant donné un petit domaine carré $S \subset \mathbb{Z}^2$, on cherche des coefficients d'interpolation $(\alpha(k))_{k\in\mathbb{Z}^2}$, $k \in \mathbb{Z}^2$, tels que

1. α est à support dans S : pour tout $k \notin S$, $\alpha(k) = 0$.
2. $|\hat{\alpha}(\xi)|^2 \hat{b}(\xi) \approx \left|\hat{t}_u(\xi)\right|^2$ pour tout $\xi \in [-\frac{1}{2}, \frac{1}{2}]^2$.

Le deuxième point laisse un certain degré de liberté. Comme les fonctions qui interviennent sont régulières (ce sont de polynômes trigonométriques), on peut se contenter de vérifier que les valeurs des deux fonctions sont proches sur une grille du domaine fréquentiel $[-\frac{1}{2}, \frac{1}{2}]^2$. Or la FFT permet justement d'évaluer des polynômes trigonométriques selon une grille. Finalement, tous les calculs peuvent être effectués par FFT et on propose l'Algorithme 3 pour calculer le texton par projections alternées, comme pour le calcul du "texton orienté synthèse" dans le cadre discret (voir l'Algorithme 1).

Comme pour le cas discret, l'Algorithme 3 s'étend au cas des images couleurs en changeant l'étape de projection spectrale par

$$\hat{\boldsymbol{\alpha}} \leftarrow \frac{1}{\sqrt{\hat{b}}} \frac{\hat{\boldsymbol{t}}_{\boldsymbol{u}}^* \hat{\boldsymbol{\alpha}}}{\left|\hat{\boldsymbol{t}}_{\boldsymbol{u}}^* \hat{\boldsymbol{\alpha}}\right|} \hat{\boldsymbol{t}}_{\boldsymbol{u}}.$$

On observe alors comme pour le cas discret qu'un nombre moyen d'impacts de 30 donne des résultats visuellement similaires à la limite gaussienne. Ceci permet donc à l'algorithme d'évaluation d'être très rapide puisqu'il suffit en moyenne de sommer 30 évaluations de la même texture pour calculer la couleur en un point. Aussi, la synthèse a lieu directement dans l'espace RGB contrairement aux méthodes concurrentes qui utilisent des espaces couleurs adaptatifs. La Figure 3.6 illustrent l'influence de la taille du support sur la qualité de la synthèse ainsi que sur l'approximation du spectre. Cependant pour certaines textures, il n'y a pas d'intérêt visuel à augmenter la taille du support. La Figure 3.7 montre que la méthode proposée permet bien de reproduire les textures gaussiennes. Elle montre également que pour certaines textures un texton de petit support est suffisant pour une synthèse de qualité. Enfin, comme annoncé, les temps de calcul de l'évaluation GPU du texton noise sont très faibles. On obtient une cadence de 100 images par seconde pour une résolution full HD 1920×1080 sur une carte graphique Quadro K5000 (comportant 1536 Cuda cores).

Dans cet algorithme, toutes les images ont la même taille que l'entrée u définie sur le domaine Ω et le symbole $\hat{\cdot}$ correspond à la TFD sur Ω calculée par FFT.

- **Entrée :** Image u définie sur Ω, support $S \subset \Omega$
- **Calcul du spectre cible :**
  - Calculer $t_u = \frac{1}{\sqrt{|\Omega|}}(u - \text{moy}(u))$ et sa TFD $\hat{t}_u$
  - Calculer l'échantillonnage du noyau $\boldsymbol{B}$-spline d'ordre 3 b (avec condition de bords périodique) et sa TFD $\hat{b}$.
- **Initialisation aléatoire :** Initialiser $\alpha \in \mathbb{R}^\Omega$ par un bruit blanc gaussien standard
- **Projection itérative sur les contraintes :** Faire $n = 50$ fois
  1. **Imposition du spectre de puissance** :
     (a) Calculer la TFD $\hat{\alpha}$ de α
     (b) $\hat{\alpha} \leftarrow \frac{|\hat{t}_u|}{\hat{b}} \frac{\hat{\alpha}}{|\hat{\alpha}|}$
     (c) Calculer α par TFD inverse
  2. **Imposition du support** S : $\alpha \leftarrow \mathbb{1}_S \cdot \alpha$
- **Sortie :** Retourner α

Algorithme 3 : Calcul du texton α à support S associé à une image u

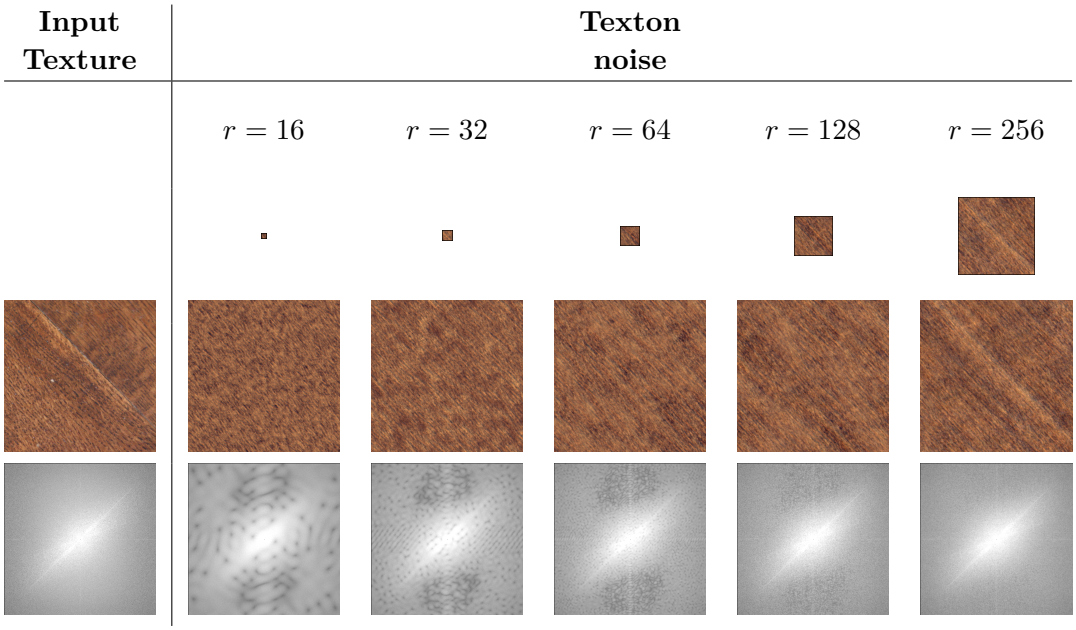


FIGURE 3.6 – Influence du support du texton : Le support du texton est un compromis entre les détails de la texture et la taille mémoire. Colonne de gauche : Image originale (512×512) et son spectre de puissance. Colonnes suivantes : Texton noise avec un support de taille $r \times r$, texton correspondant, et spectre de puissance estimé en moyennant 10 réalisations indépendantes de texton noise. On remarque que le spectre de puissance est de plus en plus proche du spectre de puissance original lorsque le support augmente.

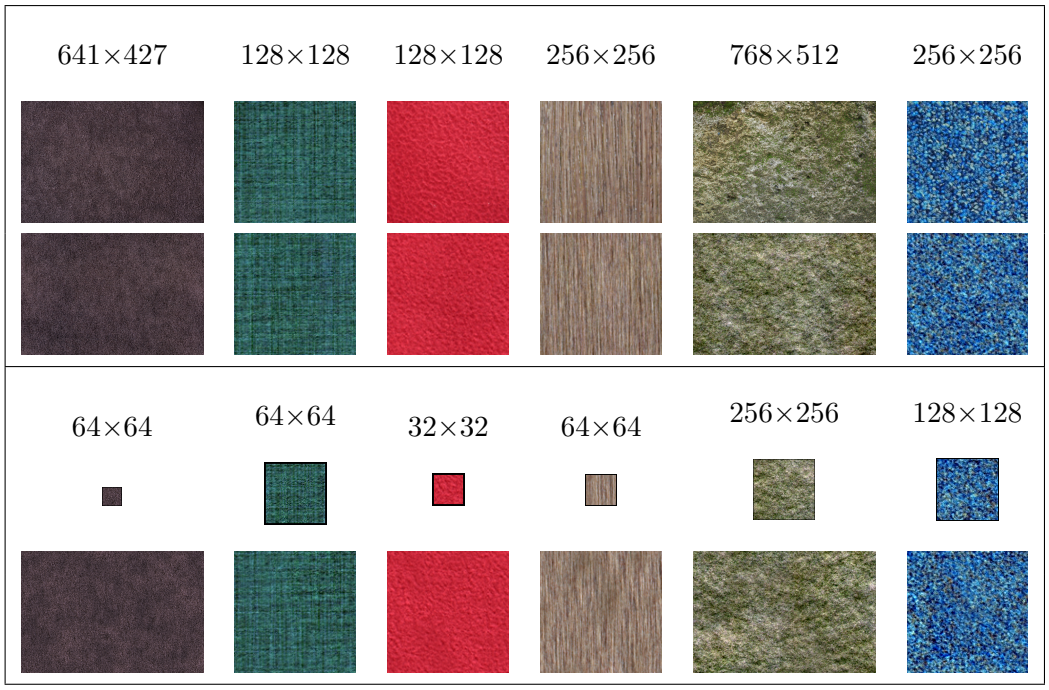


FIGURE 3.7 – Résultat de synthèse : Partie supérieure : Image originale et version gaussienne (obtenue par SNDA, voir Section 2.2), Partie inférieure : texton et texton noise. Les tailles des images et des textons sont données sur chaque colonne. La taille des textons a été choisie visuellement comme étant la plus petite qui permette un résultat de même qualité que la texture gaussienne (parmi des tailles de la forme $2^n \times 2^n$). On vérifie bien que le texton noise permet de reproduire toutes les textures gaussiennes.

3.4.3 Filtrage antialiasing pour le texton noise

L'aliasing est un problème inhérent à toute application graphique où l'on visualise des images dans des conditions de vue qui changent en permanence avec les mouvements de caméra virtuelle. Il est donc primordial de disposer de méthodes prenant en compte la distance à la caméra pour adapter le contenu à afficher grâce à des atténuations fréquentielles. Les GPU disposent de pilotes OpenGL qui implémentent le filtrage anisotrope d'images pour évaluer les textures GPU convenablement dans toute condition de visualisation. Cela repose sur un MIP-mapping (qui n'est autre qu'un scale space de l'image) et une procédure de filtrage anisotrope pour combiner les différents zooms.

En revanche, pour les bruits procéduraux on ne dispose pas d'algorithme générique, et le problème du filtrage est bien plus complexe car il nécessite de filtrer une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ que l'on ne connaît qu'à travers un algorithme d'évaluation ponctuelle. Cependant, des méthodes d'atténuations fréquentielles peuvent être mises en œuvre pour certains modèles de bruit (*e.g.* [Lagae et al., 2009]) mais elles ont un coût non négligeable.

Un des atouts majeurs du texton noise est qu'il suffit d'utiliser les méthodes standard de MIP-mapping sur le texton pour obtenir un filtrage anisotrope du bruit. Ceci est une simple conséquence de la linéarité de la convolution et du fait que le texton est une texture GPU. On obtient alors un algorithme de filtrage anisotrope pour notre bruit procédural à un moindre coût, autant en ce qui concerne le temps de calcul (quasi-négligeable pour des conditions de vue normales) que le temps d'implémentation (quelques lignes de code pour activer le filtrage anisotrope du texton lors de son évaluation). Cette simple procédure de filtrage est illustrée par la Figure 3.8.

Par ailleurs, on montre dans [J11] que comme tout sparse convolution noise, le texton noise peut-être appliquer sur des surfaces sans paramétrisation grâce à la technique du *surface noise*. Cette technique inspirée de [Chainais, 2007] consiste à projeter sur le repère tangent de la surface les points d'un processus de Poisson 3D se trouvant proche du plan tangent (voir [Lagae et al., 2009, Lagae et al., 2010a] pour les détails). Comme toutes ces opérations sont linéaires, notre algorithme de filtrage peut également être utilisé dans ce contexte, comme illustré par la Figure 3.9.

3.4.4 Mélange progressif de textures

L'article [Xia et al., 2014] propose une solution élégante et efficace pour résoudre le problème du mélange de texture (*texture mixing*) pour les textures gaussiennes qui consiste à définir et simuler une texture qui soit intermédiaire entre deux textures données $\boldsymbol{u}$ et $\boldsymbol{v}$ (voir plusieurs textures données). Pour les textures gaussiennes, une solution consiste à considérer le barycentre de Wasserstein [Agueh and Carlier, 2011] entre les deux distributions gaussiennes $\mathrm{SNDA}_\Omega(\boldsymbol{u})$ et $\mathrm{SNDA}_\Omega(\boldsymbol{v})$. Ces deux distributions sont associées aux noyaux $\boldsymbol{t_u} = \frac{1}{\sqrt{|\Omega|}}(\boldsymbol{u} - \mathrm{moy}(\boldsymbol{u}))$ et $\boldsymbol{t_v} = \frac{1}{\sqrt{|\Omega|}}(\boldsymbol{v} - \mathrm{moy}(\boldsymbol{v}))$. Comme démontré dans [Xia et al., 2014], pour

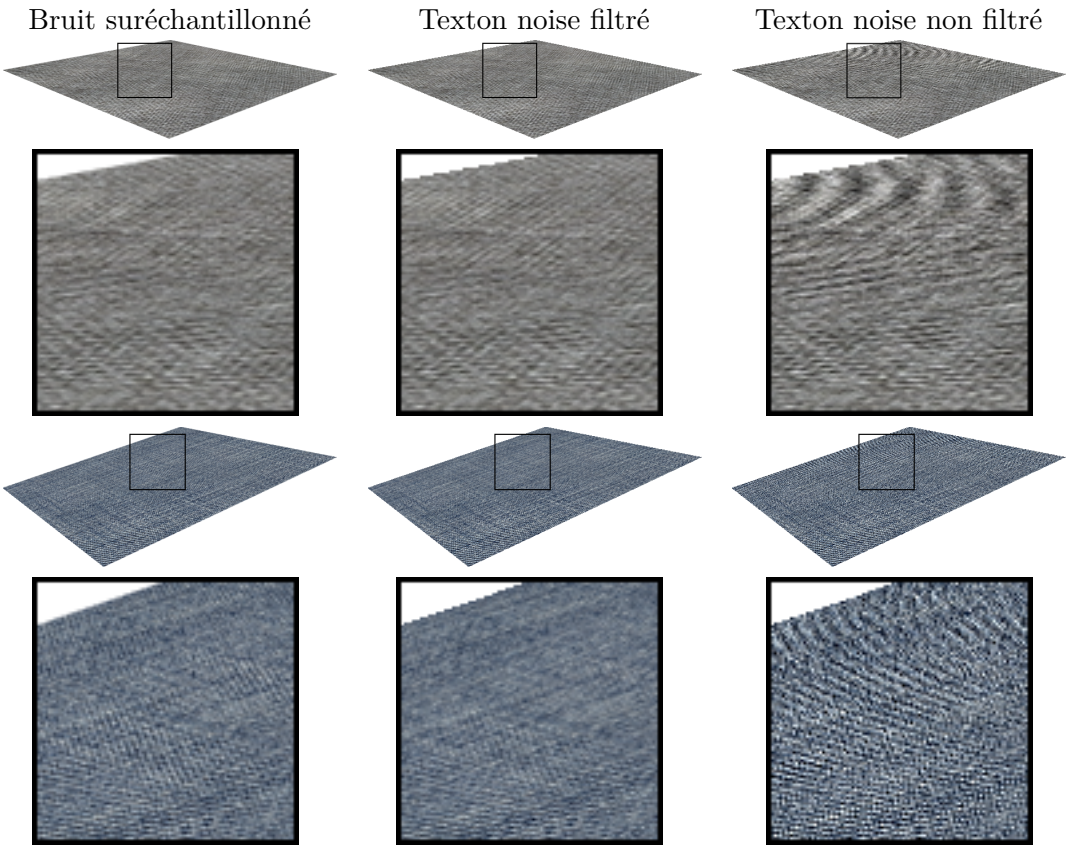


FIGURE 3.8 – Filtrage anisotrope : Gauche : Image idéale obtenu par suréchantillonnage d'un facteur 400 (chaque pixel est la moyenne de 20×20 évaluations), Centre : Texton noise filtré à la volée en utilisant le filtrage anisotrope par MIP-mapping du GPU, Droite : Texton noise non filtré (chaque pixel est une unique évaluation). Les lignes deux et quatre montrent des grossissements des zones carrées des images des lignes 1 et 3. On remarque que l'approche proposée pour le filtrage à la volée donne des résultat visuellement très proche du suréchantillonnage idéal.

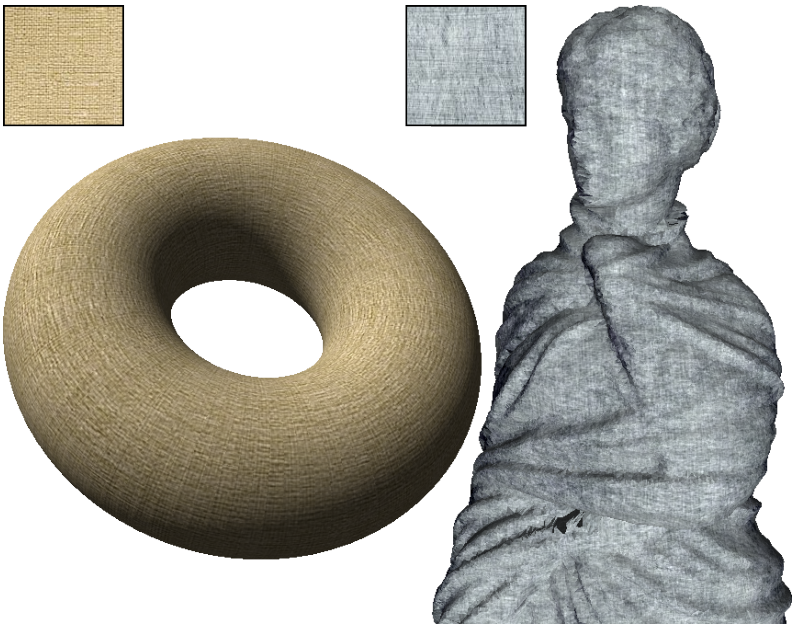


FIGURE 3.9 – Surface noise : Deux exemples de *surface noise* obtenus avec notre modèle *texton noise* (textures originales en haut à gauche de chaque exemple). Le *surface noise* permet d'appliquer une texture procédurale sur une surface sans coordonnées de texture. Ceci est particulièrement intéressant pour des surfaces acquises par scanner 3D et qui sont difficiles à paramétrer à cause de leur irrégularité (*outlier*, trous,...), comme l'exemple de droite (maillage obtenu à partir d'une fraction du nuage de points Tanagra de [Digne et al., 2011]).

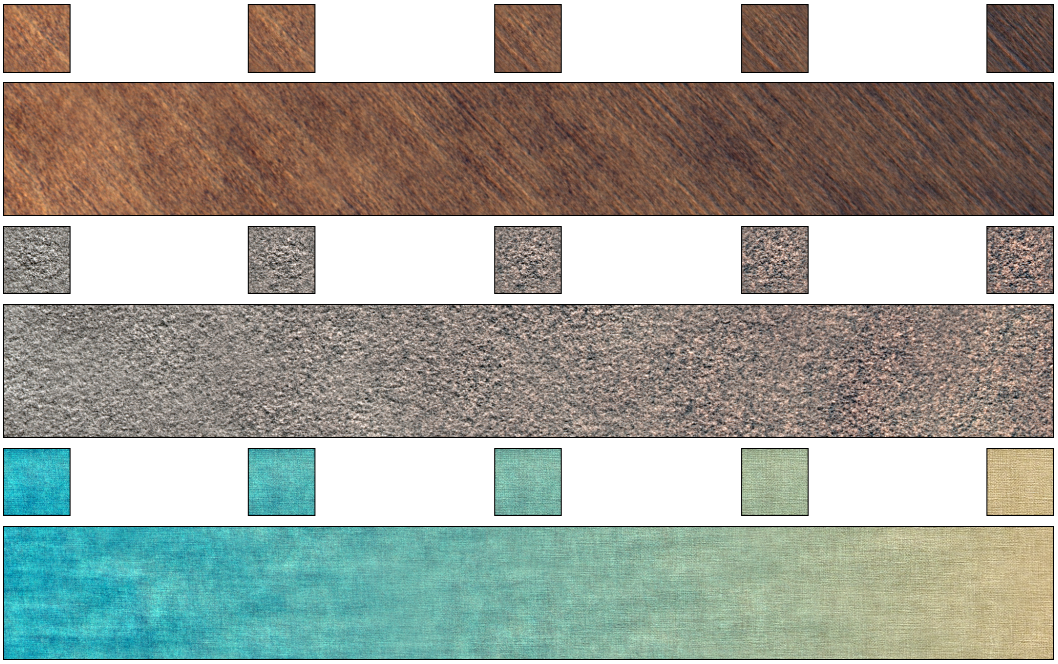


FIGURE 3.10 – Trois exemples de mélange progressif de texture avec une variation du texton selon l'axe horizontal. Pour chaque exemple, la première ligne montre cinq textons pour les paramètres de mélange $\rho = 0, 0.25, 0.5, 0.75, 1$ parmi les 21 textons utilisés pour obtenir la texture procédurale de la deuxième ligne. Remarquons que les variations des textures obtenues semblent continues bien que l'ensemble des textons soit discrétisé.

tout poids de pondération $\rho \in [0,1]$, le barycentre de Wasserstein pour le couple pondéré $((\mathrm{SNDA}_\Omega(\boldsymbol{u}), 1-\rho), (\mathrm{SNDA}_\Omega(\boldsymbol{v}), \rho))$ est le SNDA associé au noyau

$$\boldsymbol{\tau}_\rho = (1-\rho)\boldsymbol{\tau}_0 + \rho\boldsymbol{\tau}_1 \quad \text{où} \quad \hat{\boldsymbol{\tau}}_0 = \frac{\hat{\boldsymbol{t}}_{\boldsymbol{u}}^* \hat{\boldsymbol{t}}_{\boldsymbol{v}}}{|\hat{\boldsymbol{t}}_{\boldsymbol{u}}^* \hat{\boldsymbol{t}}_{\boldsymbol{v}}|} \hat{\boldsymbol{t}}_{\boldsymbol{u}} \text{ et } \boldsymbol{\tau}_1 = \boldsymbol{t}_{\boldsymbol{v}}$$

(ces noyaux ne définissent que la covariance, la moyenne doit elle être la moyenne pondérée $(1-\rho)\,\mathrm{moy}(\boldsymbol{u}) + \rho\,\mathrm{moy}(\boldsymbol{v})$).

Grâce à ces noyaux on dispose d'un modèle gaussien pour chaque poids ρ. On propose dans [J11] d'exploiter le caractère local du texton noise pour faire varier spatialement le poids ρ afin de passer continûment d'un modèle gaussien à l'autre. Ces expériences sont reproduites à la Figure 3.10. Remarquons toutefois que ce type de mélange n'a d'intérêt que pour des couples de textures dont les motifs sont assez similaires.

3.4.5 Conclusion et discussion sur la synthèse de textures gaussiennes

Le *texton noise* propose un modèle de texture procédurale qui permet de synthétiser toutes les textures gaussiennes. De plus, l'évaluation du texton noise est dix à cent fois plus rapide que celle des méthodes concurrentes de bruit procédural par l'exemple [J5] [Gilet et al., 2014]. Enfin, le texton noise dispose de tous les atouts que l'on puisse souhaiter d'un bruit procédural : filtrage anisotrope à la volée, *surface noise*, et même mélange de textures. Tous les voyants sont au vert donc, mais vu les critiques reçues pour ce travail, il reste semble-t-il à justifier de l'intérêt même de la synthèse de textures gaussiennes. En effet les textures gaussiennes forment un ensemble limité de textures, et elles sont de plus considérées comme des textures "faciles" pour les méthodes par patchs.

Contrairement à ce que laissent prétendre de nombreux articles, les méthodes de synthèse par agencement de patchs ne sont pas des méthodes génériques donnant des résultats sans défaut pour tout type de texture. En collaboration avec L. Raad (doctorante au CMLA sous la direction d'Agnès Desolneux et Jean-Michel Morel), nous avons étudié et implémenté l'algorithme image quilting [Efros and Freeman, 2001] qui construit une texture en juxtaposant séquentiellement des patchs en cherchant une frontière la plus invisible possible. Ce travail a donné lieu à une soumission au journal Image Processing On Line [J10] (article soumis, la démo en ligne est accessible sous forme de preprint). Notre implémentation permet notamment la visualisation des cartes de coordonnées des patchs utilisés pour obtenir la texture synthétisée. Comme déjà démontré dans [Aguerrebere et al., 2013] pour l'algorithme d'Efros-Leung [Efros and Leung, 1999] ceci permet de mettre en avant les artefacts de *verbatim copy* et de *growing garbage* des méthodes de synthèse par patchs.

La Figure 3.11 présente une comparaison du texton noise avec la méthode image quilting lorsque l'on synthétise des textures sur un domaine (seulement) deux fois plus grand dans chaque dimension. Comme on peut l'observer, les résultats obtenus par image quilting sont localement plus semblables à l'image originale, puisque les patchs sont des copies exactes de patchs de l'entrée. Cependant, les images synthétisées présentent plusieurs défauts : l'algorithme peut parfois faire des copies périodiques (growing garbage, voir la texture de bois), on distingue souvent des répétitions dans l'image, et surtout le réseau des frontières de patchs apparaît dans la plupart des textures (on renvoie aussi à la Figure 7 de l'article [J5] pour une comparaison avec l'algorithme de Lefebvre et Hoppe [Lefebvre and Hoppe, 2005] où l'on observe aussi les problèmes de répétitions). On peut interpréter cela par le fait que les textures gaussiennes ne comportent pas de bords contrastés dans lesquels cacher les coutures entre patchs non voisins. Ainsi, contrairement à une intuition qui semble être répandue, les textures gaussiennes ne sont pas des textures "faciles" pour les méthodes par patchs. Bien au contraire les défauts des méthodes par patchs sont visibles dans ces textures du fait de leur homogénéité et leur manque de contours contrastés.

Ces expériences confirment selon nous l'intérêt de disposer de méthodes propres

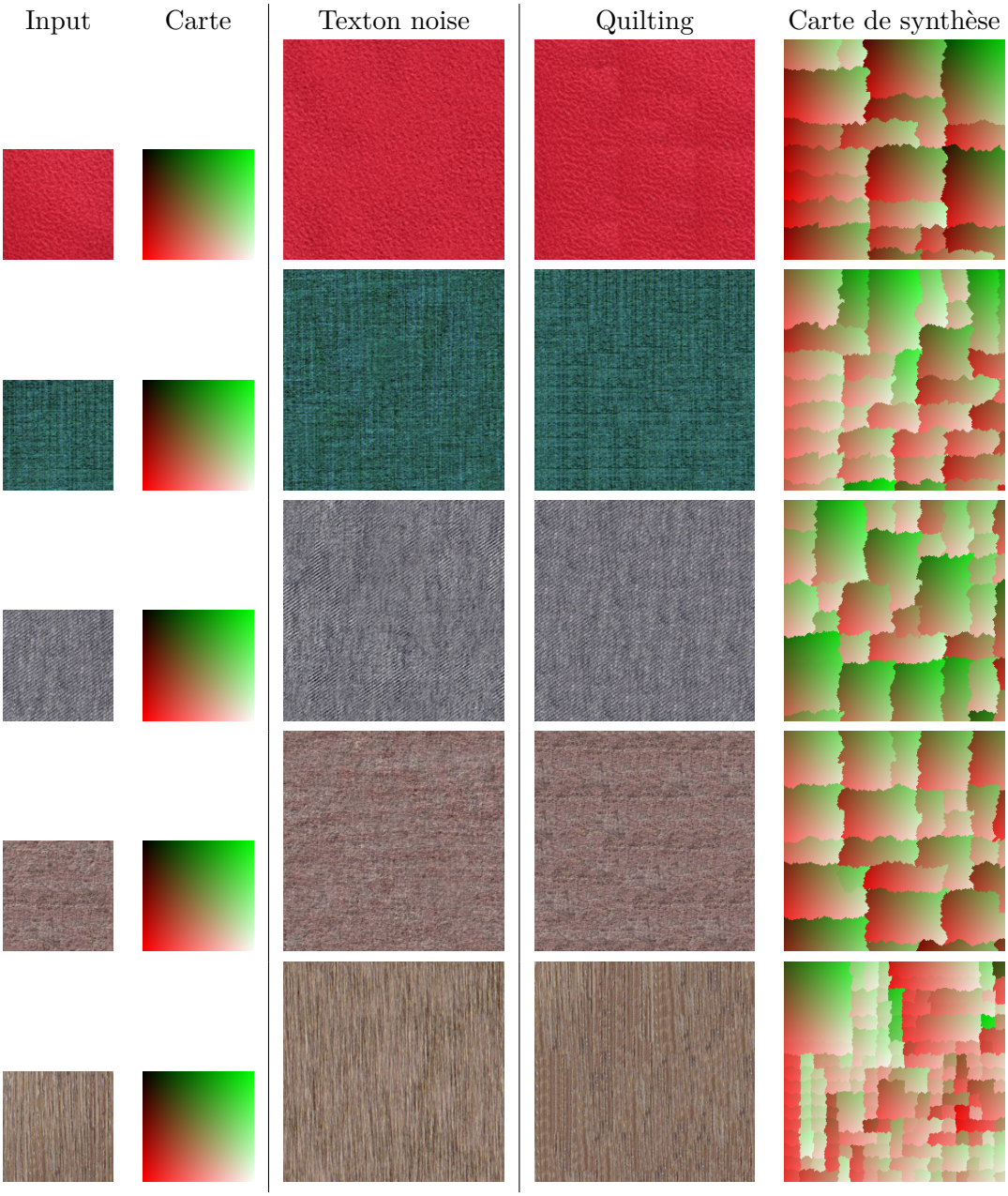


FIGURE 3.11 – Comparaison avec image quilting [Efros and Freeman, 2001] : De gauche à droite : Image d'entrée (input), carte de coordonnées de l'entrée, texton noise sur un domaine deux fois plus grand, synthèse par image quilting sur une grille de pixels deux fois plus grande, et carte de synthèse correspondant au quilting. L'*image quilting* permet de générer de nombreuses textures, et notamment des macro-textures que le texton noise ne peut reproduire. En revanche, lorsque cette méthode est utilisée pour des textures gaussiennes, les résultats comportent souvent des répétitions (*verbatim copy*) et parfois du *growing garbage* dus au *raster scan order*, alors que le texton noise produit des images stationnaires qui ne comportent pas ces artefacts.

aux textures gaussiennes, d'autant plus que ces méthodes basées sur la simulation de champs aléatoires garantissent une stabilité statistique et des propriétés de stationnarité.

Enfin, après notre série de travaux sur la synthèse de textures gaussiennes [J1, J2, J5] [C13], il nous semble difficile d'envisager de faire plus efficace que le *texton noise* [J11] qui propose un modèle de texture procédurale très simple et très rapide, avec toutes les propriétés souhaitables énumérées plus haut. Et il nous semble salutaire de concentrer nos efforts de recherche sur de nouveaux sujets.

Chapitre 4

Modèles de type germe-grain : Modélisation d'images et variations

Les objets d'étude des deux chapitres précédents étaient les modèles shot noise poissonniens et leur limite gaussienne pour la synthèse de textures. Le shot noise est un modèle particulier parmi la famille des modèles germe-grain. On appelle ici modèle germe-grain tout champ aléatoire (ou modèle d'image) constitué à partir d'une famille de formes ou de fonctions aléatoires réparties dans le plan selon un certain processus ponctuel, et combinées selon une certaine règle d'interaction. Pour le modèle shot noise, la règle d'interaction est l'addition. Le modèle germe-grain le plus simple et le plus étudié est le modèle booléen pour lequel la règle est l'union [Schneider and Weil, 2008]. On peut également citer le modèle feuilles mortes [Matheron, 1968, Bordenave et al., 2006] pour lequel la règle d'interaction est l'occlusion. On renvoie au chapitre 5 de mon manuscrit de thèse pour plusieurs exemples illustrés [T15] (voir aussi la Section 4.2.3 à la fin de ce chapitre).

Les formes permettant de constituer l'image sont réparties selon un processus ponctuel. Cela peut être un processus de Poisson, mais cela peut également être un processus ponctuel plus complexe pour lequel les points ne sont pas indépendants. Par exemple, l'article [J6] présenté à la Section 4.1.2 repose sur un processus de Gibbs particulier, généralisant le modèle de Strauss, qui contrôle la distance entre les différentes formes afin de reproduire des textures vectorielles constituées de petits motifs disjoints. Notons également qu'une de nos perspectives de recherche est l'étude de processus shot noise sur des processus ponctuels déterminantaux (voir Section 5.2) qui sont des processus ponctuels non poissonniens ayant un caractère répulsif.

La rédaction de ce manuscrit me permet de constater que les modèles germe-grain sont centraux dans tous mes travaux (bien que cela n'ait jamais été un parti pris !). Les modèles germe-grain ce sont imposés pour des travaux en *computer graphics* très orientés vers les applications. La raison principale est que les modèles germe-grain sont à la fois riches et flexibles, mais aussi robustes au sens où chaque réalisation est assurée de respecter une certaine homogénéité/stationnarité. Pendant ma thèse nous avons défini et étudié avec Y. Gousseau un nouveau modèle germe-grain baptisé modèle feuilles mortes transparentes et pour lequel la règle d'interaction est la transparence entre formes planes 2D [J4]. La motivation initiale était d'étudier en détail la règle de transparence qui ressortait des expériences de syn-

thèse automatique d'art abstrait de L. Alvarez, Y. Gousseau, et J.-M. Morel comme étant très plaisante visuellement (mon stage de master portait sur ces travaux qui ont récemment était publiés dans [Alvarez et al., 2015]). En effet, l'accumulation d'objets par transparence générait des images ayant un aspect plus texturé que les images obtenues par simple occlusion du fait de la multiplicité des contours. Ceci a d'ailleurs été confirmé par notre résultat principal : en faisant varier le paramètre de transparence, le modèle feuilles mortes transparentes fourni une famille de modèles entre le modèle feuilles mortes (avec occlusion) et le modèle gaussien (même limite gaussienne que le shot noise sur la famille de formes). Par la suite, j'ai collaboré avec T. Hurtut, à l'époque membre du laboratoire d'informatique LIPADE voisin du MAP5 et aujourd'hui professeur à l'école polytechnique de Montréal, et P.-E. Landes sur la synthèse par l'exemple de textures vectorielles. Inspirés par les travaux de Ma et al. [Ma et al., 2011], nous souhaitions travailler sur des modèles généralisant la copie par patchs à ce cadre particulier. Mais nous avons vite réalisé que cette approche entraînait de fortes instabilités et nous avons développer un modèle multi-Strauss anisotrope qui s'est avéré bien plus performant [J6]. Enfin, dernièrement, les modèles germe-grain se sont une fois de plus imposés de manière non préméditée dans notre travail avec A. Newson et J. Delon [S16]. Notre problématique était de reproduire l'aspect du grain des pellicules argentiques sur des images numériques. Alors que nous pensions travailler sur un modèle de synthèse de textures modulées par l'intensité de l'image, nous nous sommes rendu compte que la méthode la plus naturelle était de simuler un modèle booléen à l'échelle microscopique afin de représenter les amas de cristaux d'argent sur une pellicule virtuelle. La première partie de chapitre reviendra en détail sur ces trois travaux.

La deuxième partie de ce chapitre aborde des questions plus théoriques concernant la variation totale des champs aléatoires, et notamment celle des champs aléatoires de type germe-grain. Bien que ce travail soit motivé par l'étude de modèles aléatoires d'images il relève plus des probabilités appliquées que du traitement d'images comme l'atteste les journaux dans lesquels les travaux ont été publiés [J3] [J9] [J8]. En fait, le rôle des disciplines est ici inversé par rapport au reste du manuscrit : C'est le cadre théorique des fonctions à variation bornée bien connu en traitement d'images qui permet de généraliser des résultats de géométrie stochastique.

Ce travail a été motivé en premier lieu par le constat d'un certain vide concernant les champs aléatoires à variation bornée. Dans un cadre déterministe, le modèle des fonctions à variation bornée (BV pour *bounded variation*) est fondamentale en traitement d'images depuis l'article de Rudin, Osher et Fatemi [Rudin et al., 1992] sur la régularisation par variation totale (TV pour *total variation*). Les fonctions BV ont été utilisées pour résoudre de nombreux (tous ?) problèmes de traitement d'images : le débruitage avec les modèles $TV\text{-}L^2$ et $TV\text{-}L^1$ [Rudin et al., 1992, Chambolle, 2004, Nikolova, 2004, Andreu-Vaillo et al., 2004, Chan and Esedoglu, 2005, Duval et al., 2009], le zoom d'images [Guichard and Malgouyres, 1998] la déconvolution [Chan and Wong, 1998]. Plus proche de nos préoccupations, les fonctions

BV sont généralement présentées comme un modèle d'images non texturées dans les approches de décomposition d'images en partie *cartoon* plus partie texturée introduites par Y. Meyer [Meyer, 2001] et développées dans de nombreux travaux [Vese and Osher, 2003, Aujol et al., 2005, Aujol and Chambolle, 2005, Buades et al., 2010]. Par ailleurs, les champs aléatoires sur $\mathbb{R}^d$ ($d = 2$ pour nos applications) sont bien évidemment le cadre théorique pour l'étude de modèles de textures stationnaires, comme par exemple les champs shot noise présentés au Chapitre 3 ou le modèle feuilles mortes transparentes que l'on définit ci-dessous (voir Section 4.1.1). Cependant, lorsque je me suis intéressé à la variation totale du modèle feuilles mortes transparentes, j'ai constaté au cours de ma thèse qu'il n'existait aucun travaux sur la définition et l'étude des champs aléatoires à variation bornée, à l'exception du très court article [Ibragimov, 1995]. Ceci a motivé l'initiation d'une étude générale des champs aléatoires à variation bornée et leur variation totale moyenne [T15, Partie III], qui a finalement donnée lieu à l'article [J9]. Il est alors rapidement apparu que la variation totale des modèles germe-grain était à la fois liée au périmètre de leur grain X_i mais aussi au caractère Lipschitz en 0 de leur covariogramme $\gamma_X(y) = \mathbb{E}(\mathcal{L}^d(X \cap (y + X)))$. Ceci nous a permis de montrer [J3] que la célèbre formule reliant le périmètre d'un ensemble A aux dérivées directionnelles en 0 de son covariogramme sous certaines hypothèses de régularité sur la frontière de A est en fait valide pour tout ensemble mesurable A dès lors que l'on définit le périmètre d'un ensemble au sens de la variation totale de sa fonction indicatrice [Ambrosio et al., 2000].

L'étude générale des champs BV initiée à la fin de ma thèse [T15, Partie III] [C12] avaient plusieurs limitations. Notamment je ne considérais que des champs définis sur $\mathbb{R}^d$ et non un ouvert $U \subset \mathbb{R}^d$ quelconque et de nombreuses preuves établies pour des champs stationnaires étaient en fait généralisables à des champs à accroissements stationnaires. Ces travaux sont donc restés longtemps inachevé car j'ai mis cette problématique entre parenthèses pour me consacrer essentiellement à la synthèse de textures et à une nouvelle collaboration avec R. Lachièze-Rey sur le problème de réalisabilité des fonctions de covariogramme. Ce travail nous a amenés à développer la notion d'ensembles aléatoires mesurables, dont notamment la notion d'ensembles aléatoires de périmètre fini (voir aussi le récent article de I. Rataj [Rataj, 2015]). Comme l'illustre le résultat principal de notre article [J8], grâce à ses bonnes propriétés topologiques, le cadre des ensembles mesurables de périmètre fini peut permettre d'établir des résultats théoriques nouveaux là où le cadre des ensembles fermés aléatoires n'est pas adapté.

Ces nouveaux résultats ainsi que la série d'articles de H. Biermé et A. Desolneux portant sur l'étude des périmètres des ensembles de niveaux et de la variation totale de processus shot noise en 1D [Biermé and Desolneux, 2012a, Biermé and Desolneux, 2012b] puis en dimensions supérieures [Biermé and Desolneux, 2016] ont motivé la poursuite de l'étude générale des champs BV et donné lieu à la rédaction de l'article [J9]. Ces travaux apportent des théorèmes de caractérisation généraux pour qu'un champ à accroissements stationnaires ait une variation totale moyenne localement finie,

ainsi qu'une formule simple pour le calcul des cette variation totale moyenne. Une conséquence importante est que le modèle BV n'a aucun intérêt pour les champs gaussiens. En effet, on montre que si un champ gaussien est BV alors il est dans l'espace de Sobolev $W^{1,1}_{\text{loc}}(\mathbb{R}^d)$ et donc sa mesure dérivée n'a ni composante de saut, ni composante de Cantor. On illustre également l'intérêt et la facilité d'utilisation des résultats établis en calculant la variation totale moyenne de nombreux exemples de champs aléatoires de type germe-grain (et on confirme au passage que ces champs sont bien à variation totale moyenne finie si et seulement si les grains sont de périmètre moyen fini). Pour tous les modèles germe-grain considérés, on a pu vérifier que les seuls grandeurs géométriques d'intérêt pour la variation totale moyenne de ces champs sont le volume moyen et le périmètre moyen des grains.

4.1 Modèles germe-grain pour la modélisation et la génération d'images

4.1.1 Modèle feuilles mortes transparentes

Le modèle feuilles mortes transparentes (FMT) est un modèle germe-grain proche du modèle feuille morte [Matheron, 1968, Bordenave et al., 2006] mais pour lequel le principe d'occlusion entre les objets est remplacé par une composition par transparence. Cette transparence est la même pour chaque objet et est contrôlé par un indice de transparence $\alpha \in]0,1]$. Lorsqu'un objet transparent $x + X$ (*i.e.* une forme $X \subset \mathbb{R}^2$ translatée par le vecteur $x \in \mathbb{R}^2$) ayant un niveau de gris $a \in \mathbb{R}$ est placée au dessus d'une image $f : \mathbb{R}^2 \to \mathbb{R}$ on obtient alors l'image $\tilde{f}$ définie par

$$\tilde{f}(y) = \begin{cases} \alpha a + (1-\alpha) f(y) & \text{si } y \in x + X, \\ f(y) & \text{sinon.} \end{cases}$$

Cette règle d'interaction est illustrée par la Figure 4.1.

Intuitivement le modèle FMT est obtenu en superposant séquentiellement une infinité d'objets transparents. Comme cela ne complique en rien la présentation, nous présentons maintenant la définition formelle du FMT en dimension d quelconque, même si nous n'illustrerons le processus qu'en dimension $d = 2$. La famille de formes colorées est donnée par un processus de Poisson marqué indépendamment

$$\Phi = \{(t_i, x_i, X_i, a_i),\ i \in \mathbb{N}\}$$

où $\{(t_i, x_i), i \in \mathbb{N}\} \subset]-\infty, 0[\times\mathbb{R}^d$ est un processus de Poisson d'intensité 1 représentant le temps de chute et la position d'un objet, (X_i) est une suite de variables aléatoire (v.a.) i.i.d. sur l'ensemble $\mathcal{F} = \mathcal{F}(\mathbb{R}^d)$ des fermés de $\mathbb{R}^d$ (on parle d'ensemble aléatoire fermé ou *random closed sets* (RACS)), et (a_i) est une suite de v.a. i.i.d. sur $\mathbb{R}$. On note P_X la loi des X_i et P_a la loi des a_i. On suppose que $0 < \mathbb{E}(\mathcal{L}^d(X)) < +\infty$.

$x + X$ avec $a = 200$ | Scène originale f | Scène finale $\tilde{f}$

FIGURE 4.1 – Principe de transparence : Le disque de l'image de gauche est ajouté au-dessus de la scène originale f par transparence selon le coefficient α pour obtenir la scène $\tilde{f}$.

Alors le FMT d'indice de transparence α est le champ aléatoire stationnaire défini par

$$f(y) = \sum_{i \in \mathbb{N}} \mathbb{1}\left(y \in x_i + X_i\right) \alpha a_i \left(1-\alpha\right)^{\left(\sum_{j \in \mathbb{N}} \mathbb{1}\left(t_j \in (t_i,0) \text{ and } y \in x_j + X_j\right)\right)}. \tag{4.1}$$

Cette équation correspond bien à la construction décrite plus haut car l'objet $x_i + X_i$ recouvrant le point y contribue à la couleur finale $f(y)$ avec le niveau de gris αa_i, puis est atténué par un facteur $(1-\alpha)$ à la puissance le nombre d'objets N tombés après l'objet i sur le point y, soit

$$N = \sum_{j \in \mathbb{N}} \mathbb{1}\left(t_j \in (t_i, 0) \text{ and } y \in x_j + X_j\right).$$

Les niveaux de gris (a_i) étant i.i.d., on montre que

$$\mathbb{E}(f(y)) = \mathbb{E}(a) \quad \text{et} \quad \operatorname{Var}(f(y)) = \frac{\alpha}{2-\alpha} \operatorname{Var}(a).$$

La première contribution concernant ce modèle a été de proposer un algorithme de simulation à une précision $\varepsilon > 0$ arbitrairement petite sous l'hypothèse que les niveaux de gris a_i étaient bornés. Cette algorithme repose sur un argument de *coupling from the past* propre au modèle feuilles mortes [Kendall and Thönnes, 1999]. Nous montrons à la Figure 4.2 plusieurs simulations de modèles FMT obtenues avec la même famille de formes mais avec différents indices de transparence $\alpha \in]0, 1]$. On peut remarquer sur cette figure que la variance de l'image diminue bien avec α.

La deuxième contribution concernant l'étude de ce modèle concerne le calcul de la covariance du FMT. Comme pour tous les modèles germe-grain, cette covariance fait intervenir le covariogramme des RACS X_i, à savoir la fonction

$$\gamma_X(y) = \mathbb{E}(\mathcal{L}^d(X \cap (y + X))), \quad y \in \mathbb{R}^d.$$

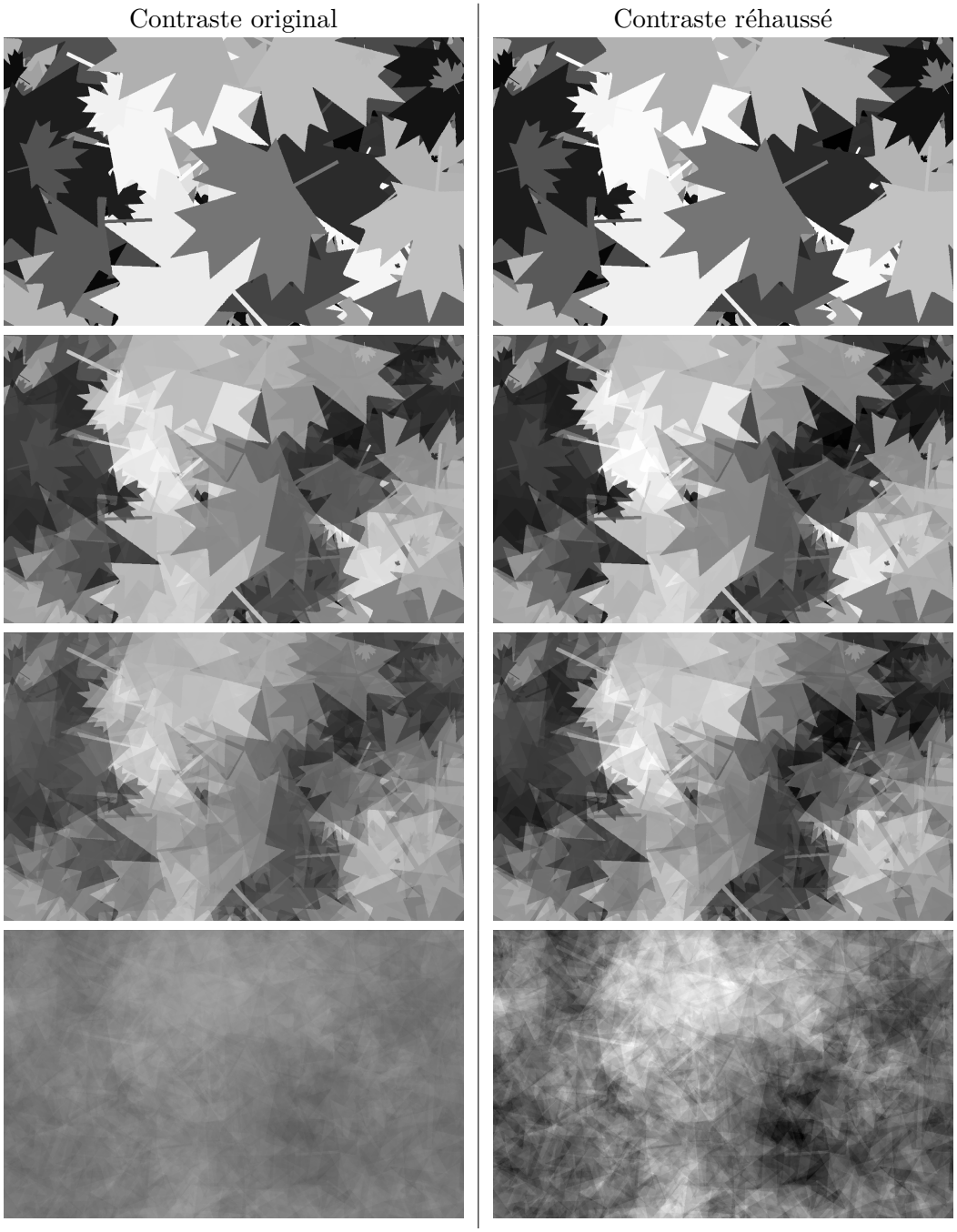


FIGURE 4.2 – Réalisations de modèles FMT à partir de la même réalisation de famille de formes transparentes mais avec différentes valeurs d'indice de transparence α (de haut en bas $\alpha = 1$ (occlusion), $0,7$, $0,4$, $0,05$). On remarque que la variance diminue avec l'indice de transparence. La colonne de droite présente les mêmes images avec un contraste réhaussé.

On reviendra sur les propriétés du covariogramme à la Section 4.2.2 où l'on discute de notre étude théorique sur cette fonction [J3]. On a alors montré que

$$\mathrm{Cov}(f)(y) = \frac{\alpha\gamma_X(y)}{2\mathbb{E}(\nu_d(X)) - \alpha\gamma_X(y)}\,\mathrm{Var}(a), \quad y \in \mathbb{R}^d.$$

Le FMT $f(y)$ étant la somme sur les points du processus de Poisson Φ d'une fonctionnelle de la forme $g((t_i, x_i, X_i, a_i), \Phi)$, l'approche standard pour calculer la covariance de f est d'utiliser la formule de Slyvniak-Mecke [Schneider and Weil, 2008]. Toutefois cela conduit à des calculs assez lourds (voir [T15, Section D.2]). Nous avons montré qu'il était plus facile de calculer la covariance en utilisant une généralisation du caractère sans mémoire d'un processus de Poisson temporel [J4].

La dernière et plus importante contribution concernant l'étude du FMT est d'avoir établi le théorème suivant, où l'on note f_α le modèle FMT avec indice de transparence $\alpha \in]0, 1]$.

Théorème 4.1. *Supposons que* $\mathrm{Var}(a) > 0$ *(sinon f est contant à* $\mathbb{E}(a)$*). Alors, lorsque l'indice de transparence* α *tend vers* 0*, la famille de champs aléatoires normalisés*

$$\left(\frac{f_\alpha - \mathbb{E}(f_\alpha)}{\sqrt{\mathrm{Var}(f_\alpha)}}\right)_{\alpha\in]0,1]}$$

converge au sens des lois finies dimensionnelles vers un champ gaussien stationnaire ayant pour fonction de covariance

$$C(y) = \frac{\gamma_X(y)}{\mathbb{E}(\mathcal{L}^d(X))} = \frac{\gamma_X(y)}{\gamma_X(0)}.$$

On remarque que ce champ gaussien limite est le même que pour la limite gaussienne lorsque l'intensité $\lambda \to +\infty$ de la version centrée normalisée du shot noise

$$f_{\mathrm{SN}}(y) = \sum_i a_i \mathbb{1}(y \in x_i + X_i)$$

où les $\{x_i,\ i \in \mathbb{N}\}$ forment un processus de Poisson d'intensité $\lambda > 0$. La Figure 4.3 illustre la convergence gaussienne du modèle FMT. On remarque ainsi qu'en faisant varier l'indice de transparence α, le FMT effectue une transition entre le modèle feuilles mortes ($\alpha = 1$) et le champ gaussien ($\alpha \to 0$). Comme discuté en introduction de chapitre, ce résultat théorique est une justification formelle de l'observation que les textures obtenues par transparence ont un aspect intermédiaire entre les textures gaussiennes et les textures feuilles mortes obtenues par occlusion.

On remarque facilement que les réalisations des modèles FMT présentent des discontinuités de type saut le long d'un bord en tout point du plan (comme le suggère les images des Figures 4.2 et 4.3 dans les cas d'indices de transparence faibles). Toutefois la hauteur de ces sauts est atténuée par des facteurs de la forme $(1-\alpha)^n$. Comme on le discutera à la Section 4.2.3, nous avons démontré que malgré cette omniprésence des contours, la variation totale des modèles FMT est finie dès lors que le périmètre moyen des ensembles X_i est fini.

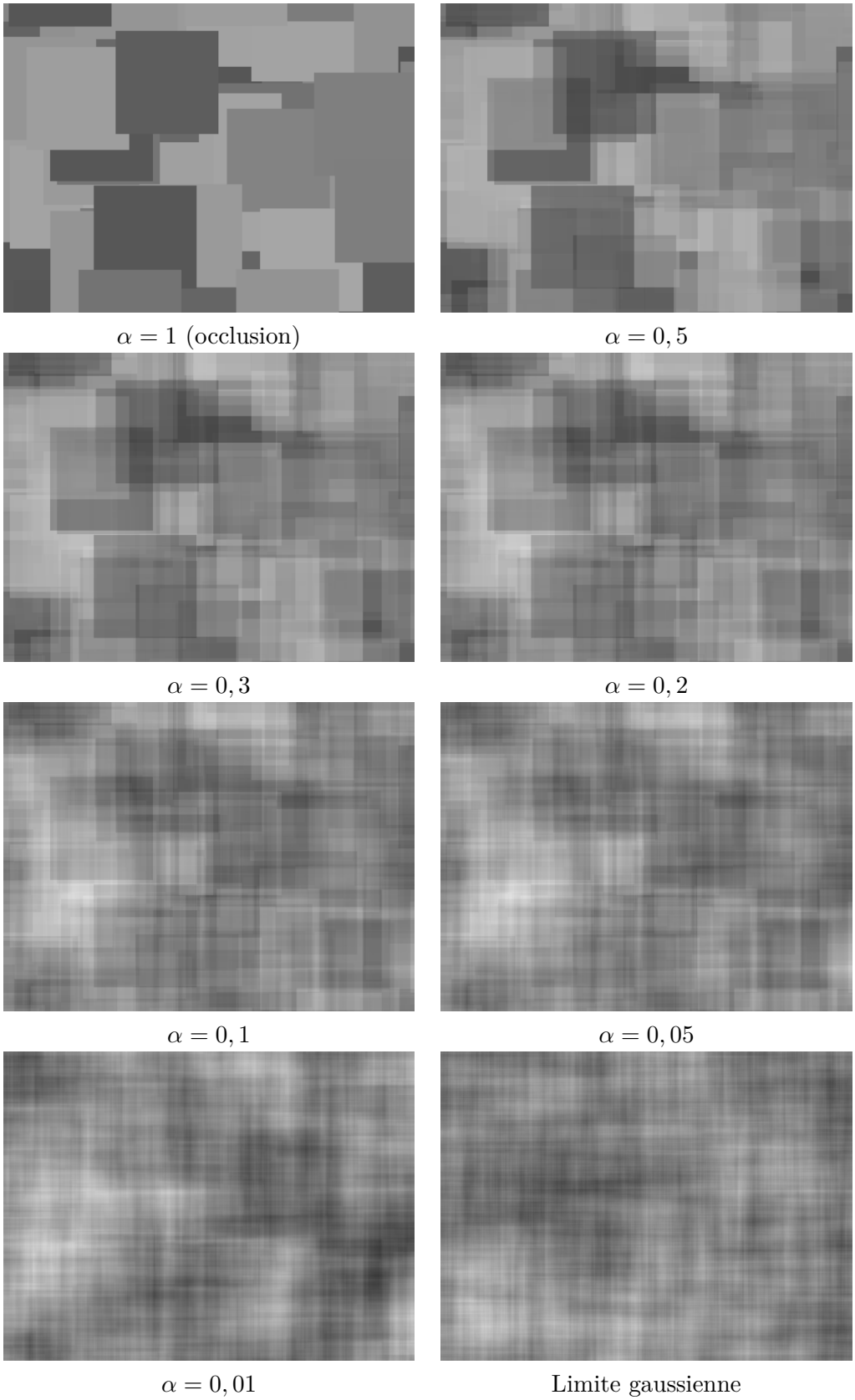


FIGURE 4.3 – Du modèle feuille morte au champ gaussien : Illustration visuelle de la convergence gaussienne de la séquence de FMT normalisé lorsque l'indice de transparence tend vers 0.

4.1.2 Un modèle pour la synthèse de textures vectorielles prenant en compte les formes des objets

Nous présentons maintenant notre article [J6] présenté à la conférence EGSR 2013 et qui portent sur la synthèse par l'exemple de textures vectorielles. Les finalités de la synthèse par l'exemple de textures vectorielles sont les mêmes que pour le problème de la synthèse par l'exemple d'images de textures discuté en introduction de ce manuscrit : déterminer un algorithme qui soit capable à partir d'un exemple de générer une nouvelle texture vectorielle de taille plus grande et visuellement similaire à l'entrée. Cependant la nature des données rend le problème complètement différent. Alors qu'avec les images de textures on cherche des méthodes pouvant reproduire les motifs locaux des textures (patchs, coefficients de Fourier, etc.) avec les textures vectorielles les motifs sont donnés sous forme de dessin vectoriel que l'on peut parfaitement reproduire par translation. En effet, une texture vectorielle est une union de motifs disjoints répartis dans une fenêtre (voir la première colonne de la Figure 4.5 qui contient plusieurs exemples de textures vectorielles). En revanche, ce qu'il est nécessaire de modéliser est l'aléa qui gouverne le placement des objets les uns par rapport aux autres, ce qui fait donc naturellement appel à la théorie des processus ponctuels marqués.

Avant de poursuivre, il est (malheureusement) nécessaire de faire un point de vocabulaire concernant les textures vectorielles qui sont dénommées dans la littérature comme *element arrangment* ou plus récemment comme *discrete texture*. Le terme *discrete texture* venant de [Ma et al., 2011] est réellement discutable (tout comme les résultats de l'article étant donné qu'il nous fut impossible de les reproduire (voir la Figure 4.5) malgré plusieurs échanges avec les auteurs...). Par ailleurs, dans ce manuscrit les "textures discrètes" réfèrent aux textures définies sur de sous-domaine de $\mathbb{Z}^2$ par opposition aux textures procédurales définies sur le domaine continu $\mathbb{R}^2$. Pour toutes ces raisons on retiendra ici les termes d'*arrangement d'éléments* et de *texture vectorielle*. On renvoie à l'article [J6] pour un état de l'art sur les méthodes de synthèse d'arrangements, et notamment pour une description rapide des méthodes auxquelles nous comparons nos résultats à la Figure 4.5. Une des limitations communes des ces méthodes existantes étaient de ne pas prendre en compte soit l'anisotropie des éléments constituant l'arrangement, soit l'anisotropie de la répartition des positions des éléments.

Afin de combler ses limitations nous proposons un modèle de Gibbs dont la fonction d'interaction dépend de la distance et de l'orientation de chaque couple d'éléments de l'arrangement. Cependant, une première observation est que ces deux notions de distance et d'orientation doivent prendre en compte les formes des éléments. Nous définissons la distance entre deux formes par la distance euclidienne entre deux ensembles compacts. Pour l'orientation, nous utilisons une orientation "de force gravitationnelle" qui a l'avantage de diminuer l'angle d'orientation lorsque deux objets longitudinaux sont proches et simplement décalés [J6, Figure 3]. Mentionnons toutefois que pour des raisons de performance, les distances et les orientations entre couples d'éléments sont calculées sur des versions grossières des éléments

appelés *proxy* (voir la partie gauche de la Figure 4.4).

Avant de pouvoir introduire les équations définissant notre modèle, nous devons fixer quelques notations propres à ce paragraphe. Un arrangement est un ensemble $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ de N éléments qui sont sous la forme (u_n, s_n) où $u_n \in \mathbb{R}^d$ est une position ($d = 2$ ou $d = 3$) et $s_n \subset \mathbb{R}^d$ est une forme centrée sur l'origine, de sorte que $x_n = u_n + s_n$. Remarquons que cette décomposition des éléments en couple $x_n = (u_n, s_n) = u_n + s_n$ est standard en géométrie stochastique, et même si le couple (u_n, s_n) n'est pas donné tel quel en entrée il se calcule aisément à l'aide d'une fonction dite "centroïde" [Schneider and Weil, 2008], comme par exemple le centre du rectangle englobant la forme s. On note également $d_S(x_n, x_m)$ la distance entre deux formes et $o_S(x_n, x_m)$ l'orientation gravitationnelle entre deux formes. On remarque que ces deux fonctions sont invariantes par translation du couple (x_n, x_m).

Un processus de Gibbs (de cardinal fini) est décrit par sa fonction de densité f par rapport à la loi du processus de Poisson d'intensité 1 sur le même domaine. Cette densité f est définie sur l'ensemble des configurations finies de points et elle donne un poids à chaque configuration en prenant en compte la répartition des points (alors que le processus de Poisson donne le même poids à chaque configuration, les points étant uniformément répartis et indépendants). Beaucoup de modèle de Gibbs existent et leur étude est un domaine très actif de recherche en probabilité (voir par exemple [Dereudre et al., 2012] et ces nombreuses références). Nous considérons ici un modèle où l'interaction entre les points se limite à l'interaction de chaque couple (on parle de *pairwise interaction model*). Plus précisément, on a

$$f(\boldsymbol{x}) = f(\{x_1, \ldots, x_N\}) = \beta^N \prod_{n \neq m} \Gamma(x_n, x_m) \tag{4.2}$$

où β est appelé intensité du processus et où Γ est appelée fonction d'interaction. On se limite au cas où $\Gamma(x_n, x_m)$ ne dépend que de la distance $d_S(x_n, x_m)$ et de l'orientation $o_S(x_n, x_m)$ entre x_n et x_m.

Etant donné une intensité β et une fonction Γ, il est très simple de simuler le processus de Gibbs associé par un algorithme de type Metropolis-Hastings. Pour cela on doit introduire l'intensité conditionnelle de Papangelou définie par

$$\lambda((u,s); \mathbf{x}) = \begin{cases} \frac{f(\mathbf{x})}{f(\mathbf{x} \setminus (u,s))} & \text{si } (u,s) \text{ est un des éléments de } \mathbf{x}, \\ \frac{f(\mathbf{x} \cup (u,s))}{f(\mathbf{x})} & \text{sinon.} \end{cases} \tag{4.3}$$

Pour ces travaux, nous avons utilisé un simple algorithme de vie ou de mort [Geyer and Møller, 1994], même si l'on pourrait envisager des propositions plus complexes dans la procédure de Metropolis-Hastings. Cet algorithme est décrit par l'Algorithme 4. En pratique, pour les résultats présentés plus bas, le nombre d'itérations utilisé est $T = 50000$ pour les arrangements 2D et $T = 10000$ pour les arrangements 3D.

Remarque (Calcul de l'intensité conditionnelle de Papangelou)**.** On remarque que cet algorithme fait intervenir l'intensité de Papangelou $\lambda((u,s); \mathbf{x})$ à chaque étape.

Simulation d'un arrangement $\boldsymbol{x}$ dans une fenêtre W (avec condition de bords périodique). On note $\#\boldsymbol{x}$ le cardinal de $\boldsymbol{x}$ et $|W|$ la mesure de la fenêtre W.

— Initialiser $\boldsymbol{x}$ par l'ensemble vide.

— Pour $t = 1$ à T, appliquer une des deux perturbations suivantes avec probabilité $\frac{1}{2}$:

— **Naissance :**

1. Tirer uniformément dans W une position candidate u.
2. Tirer uniformément une forme candidate s parmi les formes de l'exemple d'entrée.
3. Calculer le taux d'acceptation $R_b = \lambda((u,s);\mathbf{x})\frac{|W|}{\#\mathbf{x}+1}$.
4. Ajouter (u,s) à $\boldsymbol{x}$ avec probabilité $\min(R_b, 1)$.

— **Mort :**

1. Sélectionner uniformément un élément $(u,s) \in \boldsymbol{x}$ (si $\boldsymbol{x}$ est vide on ne fait rien).
2. Calculer le taux d'acceptation $R_d = \frac{1}{\lambda((u,s);\mathbf{x})}\frac{\#\mathbf{x}}{|W|}$.
3. Retirer (u,s) de $\boldsymbol{x}$ avec probabilité $\min(R_d, 1)$.

— Renvoyer l'arrangement $\boldsymbol{x}$ obtenu après T itérations.

Algorithme 4 : Algorithme de vie ou de mort pour la simulation du processus de formes de densité f donnée par (4.2).

Le calcul rapide de cette intensité est donc critique pour les performances de l'algorithme. Pour cela il est important de remarquer que de nombreux termes sont en commun entre le numérateur et le dénominateur et que seuls les termes dépendant de (u, s) subsistent. Par ailleurs, en pratique $\Gamma((u, s), x_n)$ vaut 1 si la forme x_n est éloignée de (u, s) (interaction à portée limitée), et par conséquent seules les formes x_n dans le voisinage de (u, s) interviennent dans le calcul. Ainsi le coup d'une étape de l'algorithme ne dépend pas de la taille de W (mais le nombre d'itérations T nécessaires pour être "assez proche" de la loi invariante de la chaîne dépend lui de W).

De manière assez similaire aux problèmes rencontrés pour le design du modèle pour le *Gabor noise* par l'exemple (voir Section 3.3), un point critique de ce travail résidait dans le fait de déterminer une fonction d'interaction Γ qui soit assez générale pour satisfaire nos attentes vis à vis de la prise en compte et de la reproduction de l'anisotropie des arrangements, mais qui soit également assez simple pour pouvoir être apprise à partir d'échantillons disposant de peu d'éléments. Pour cela, on a simplement généralisé un modèle multi-Strauss en lui ajoutant une dépendance directionnelle. Plus précisément, on suppose que la fonction $\Gamma(x_n, x_m)$ est constante par morceaux vis à vis des mesures d'orientation $o_S(x_n, x_m)$ et de distance $d_S(x_n, x_m)$ entre les formes (x_n, x_m). Pour cela on se donne D directions principales e_i qui partitionnent le domaine angulaire, puis dans chaque cône de direction e_i on se donne K intervalles de distances $0 = \delta_{i,0} \leq \cdots \leq \delta_{i,k} \leq \cdots \leq \delta_{i,K}$. Alors, pour un couple de forme dont l'orientation $o_S(x_n, x_m)$ appartient au cône d'orientation e_i la fonction Γ est définie par

$$\Gamma(x_n, x_m) = \begin{cases} \gamma_{i,k} & \text{si } \delta_{i,k} \leq d_S(x_n, x_m) < \delta_{i,k+1}, \\ 1 & \text{si } d_S(x_n, x_m) \geq \delta_{i,K}. \end{cases}$$

En pratique on prend $K = 3$ et on choisit les seuils de distance grâce aux quantiles $\{0, 0.1, 0.2\}$ des distances empiriques entre les couples. Enfin, les valeurs $\gamma_{i,k}$ qui régissent les interactions à faible distance des couples de formes sont déterminés par maximisation de la pseudo-vraisemblance [Baddeley and Turner, 2000]. La procédure complète d'estimation de modèle à partir d'un exemple est décrite par le Figure 4.4. Mentionnons aussi que l'on peut complexifier le modèle en introduisant une marque supplémentaire de catégorie entre les formes comme déjà utilisé par [Hurtut et al., 2009]. Cela permet par exemple de différencier l'interaction entre les formes qui n'ont pas la même couleur, comme à l'exemple (f) de la Figure 4.5.

On reproduit des résultats pour des arrangements 2D à la Figure 4.5 avec des comparaisons avec les méthodes concurrentes qui tournent toutes en la faveur du modèles de Gibbs proposé. On montre également des exemples d'arrangements 3D à la Figure 4.6. On voit que l'estimation des paramètres $\gamma_{i,k}$ du modèle par maximum de pseudo-vraisemblance est stable et permet de reproduire fidèlement les arrangements. Je dois saluer ici les qualités de Pierre-Edouard Landes qui s'est chargé d'implémenter les méthodes en 2D et 3D et a réussi à utiliser la méthode de maximum de vraisemblance [Baddeley and Turner, 2000] dans notre contexte

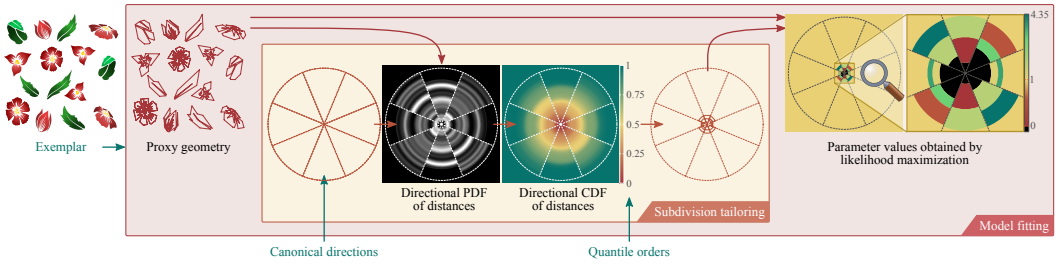


FIGURE 4.4 – Design du support de la fonction d'interaction de paires de formes et apprentissage des coefficient d'interaction. Le nombre de directions canoniques de la fonction d'interaction et le nombre de palier dans chaque direction sont fixés. On détermine les distances seuil grâce à la fonction de répartition empirique dans chaque direction selon les quantiles $\{0, 0.1, 0.2\}$. On détermine ensuite les valeurs de la fonction d'interaction par maximum de vraisemblance.

particulier. Cette méthode, qui fait intervenir des méthodes de quadratures sur des grilles irrégulières contenant chacune des données ponctuelles, est en effet loin d'être triviale. Les temps de calcul pour générer ces différents résultats varient entre 20 secondes et 6 minutes.

Enfin, un des intérêts de disposer d'un modèle paramétrique pour un arrangement est que l'on peut générer des variantes des arrangements en changeant les paramètres de manière pertinente. Ceci est illustré par la Figure 4.7 ou par exemple des effets de clustering sont créés dans un arrangement homogène. On doit tout de même préciser que la compréhension de l'influence des paramètres est loin d'être simple et que déterminer la bonne façon d'explorer ces paramètres reste un problème ouvert.

4.1.3 Modélisation stochastique du grain des pellicules argentiques et algorithme de rendu à une résolution arbitraire

Nous présentons maintenant une dernière application des modèles germe-grain pour le *computer graphics*. Il s'agit de la modélisation du grain des pellicules argentiques par un modèle booléen afin de pouvoir reproduire ce grain sur des images numériques qui en sont dépourvues. Ce travail [S16] est une collaboration en cours avec Alasdair Newson (Postdoc au MAP5) et Julie Delon (Professeur au MAP5). Nous nous contentons ici d'illustrer nos premiers résultats après avoir discuté la pertinence de l'utilisation d'un modèle génératif pour la simulation de grain argentique. L'article soumis décrivant ce travail [S16] comporte une importante discussion algorithmique avec deux implémentations CPU et une version GPU rapide qui nous ne discutons pas en détail ici. Un article co-écrit avec Alasdair Newson, Julien Delon et Nourra Faraj (ingénieure de recherche au MAP5) et décrivant précisément ces deux algorithmes est en préparation et sera soumis prochainement au journal *Image Processing On Line (IPOL)*.

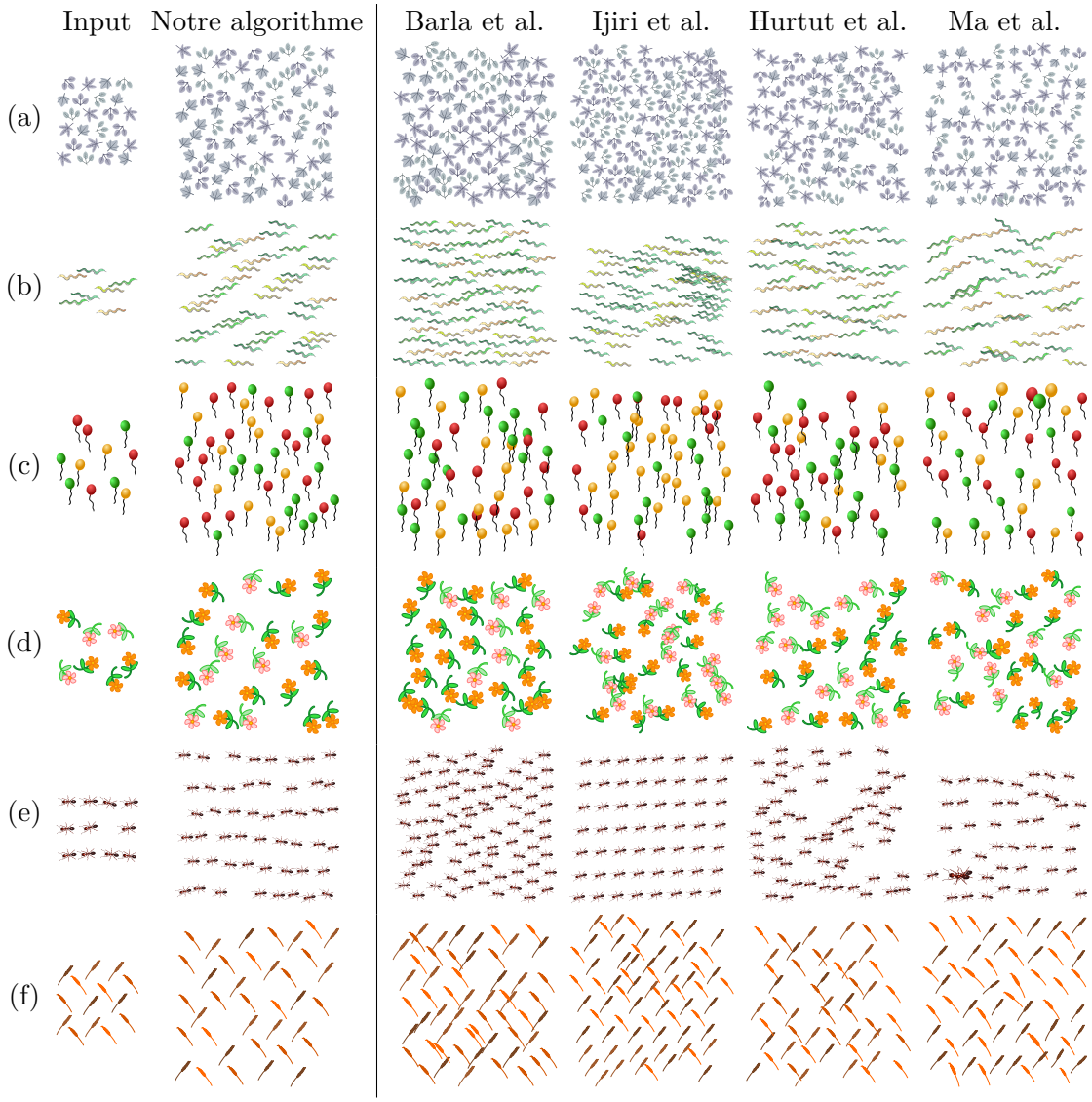


FIGURE 4.5 – Résultats de synthèse 2D et comparaison avec différentes méthodes de l'état de l'art [Barla et al., 2006], [Ijiri et al., 2008], [Hurtut et al., 2009], et [Ma et al., 2011]. Tous les résultats sont obtenus avec les mêmes paramètres par défaut, à l'exception de *(f)* où l'on utilise deux catégories. On observe que la distribution des espacements entre les objets est globalement bien reproduites par notre algorithme. Grâce à notre fonction d'interaction anisotrope, notre modèle capture les interactions des paires d'éléments à chaque échelle et selon chaque orientation. On remarque également que des formes anisotropes *(b,c,e)* sont bien gérées par la méthode et que toutes les méthodes concurrentes sont sujet à des artefacts de collisions entre formes.

FIGURE 4.6 – Deux résultats de synthèse sur des volumes 3D. Même si les entrées contiennent un nombre limité d'éléments notre modèle permet de synthétiser des résultats cohérents sur des volumes arbitraires.

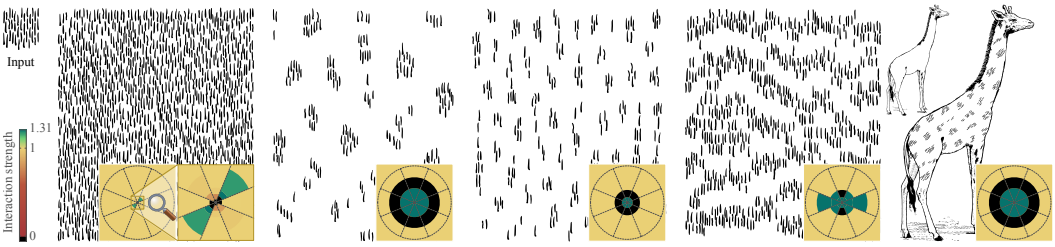


FIGURE 4.7 – *Editing* d'arrangements : En faisant varier les différents paramètres de la fonction d'interaction apprise sur l'exemple, de nouvelles textures vectorielles sont générées par l'algorithme de synthèse. Les opérations sur la fonction d'interaction sont les suivantes : changement des seuils de distance, mise à zéro de coefficients d'interaction, rotation de la fonction d'interaction.

Les pellicules utilisées en photographie analogique sont (ou plutôt étaient car la plupart ne sont plus produites...) constituées d'une suspension de cristaux d'argent qui réagissent à la lumière et deviennent opaques après développement de la photographie. Ainsi, à l'échelle microscopique des cristaux d'argent (dont la taille caractéristique est un μm sur une surface de pellicule de 3×2 cm), le négatif est constitué de la réunion de cristaux d'argent dont la densité varie selon la luminosité de la scène. En particulier c'est une image binaire. Pour passer l'image en positif, on effectue alors une projection optique du négatif sur un papier photosensible. On peut alors modéliser l'image finale comme une convolution entre un noyau de flou (dû à la projection optique et à la perception de l'œil humain) et une image binaire correspondant à la réunion de cristaux blancs sur un fond noir. On obtient alors une image en niveau de gris. Ces niveaux de gris traduisent la densité locale de cristaux, et donc l'intensité lumineuse de la scène photographiée. Cependant, les fluctuations dues à la répartition aléatoire des cristaux dans la pellicule se traduisent par une certaine irrégularité dans l'image finale que l'on appelle grain de la pellicule.

Cet effet de grain a des aspects esthétiques appréciés des amateurs comme des professionnels de la photographie ou du cinéma. C'est notamment une des raisons pour lesquelles certains films sont encore tournés sur pellicule, bien qu'au final les films sont convertis et projetés en numérique. Ainsi il y a une forte demande pour une technologie permettant d'ajouter a posteriori un grain argentique sur des images numériques. Plusieurs solutions commerciales existent, notamment le logiciel FilmPack de DxO [DxO, 2016], le logiciel TrueGrain [Grubba Software, 2015], ou encore le module dédié de [G'MIC, 2016]. Toutes ces solutions reposent sur des images scannées de photographies argentiques, la motivation étant de reproduire fidèlement l'aspect de différents type de pellicules. Cependant cette approche a plusieurs limitations. Premièrement, la résolution des images scannées limite l'utilisation des grains à certaines résolutions. De plus, la multiplication des scans pose des problèmes de stockage mémoire. Par ailleurs, l'utilisation d'un seul scan ne permet pas de générer facilement différentes réalisations indépendantes du grain, ce qui est pourtant nécessaire pour la vidéo. Enfin, une limitation liée à la modélisation du phénomène physique est que le grain n'a pas les mêmes propriétés de corrélation dans les zones sombres et les zones foncées de l'image. Ainsi, lorsque l'on transfert un grain par simple modulation de variance d'une image de grain scannée ayant un niveau de gris moyen constant, on ne respecte pas les propriétés du grain dans les zones très claires ou très sombres.

Pour toutes ces raisons, nous avons souhaité développer un algorithme de simulation de grain argentique basé sur un modèle mathématique approchant la réalité physique. Il s'est avéré qu'une façon simple et pertinente pour reproduire les différentes corrélations du grain qui dépendent du niveau de gris est de réussir à simuler une version flou de l'image positive binaire. La solution que nous proposons est simplement de définir un modèle booléen à intensité λ variable, choisie telle que la surface moyenne du modèle booléen soit égale au niveau de gris de l'image numérique.

Commençons par quelques rappels sur le modèle booléen. Etant donnée la dif-

férence d'échelle entre les pixels (de l'ordre de 10^3 par centimètre) par rapport aux cristaux (de l'ordre de 10^6 par centimètre) on se limite à modéliser les cristaux par des disques $B(x_i, r_i)$ où les rayons r_i sont donnés par une loi de variance finie. On considère donc des ensembles booléens

$$Z = \bigcup_i B(x_i, r_i)$$

où $\{(x_i, r_i)\}$ est un processus de Poisson marqué indépendamment d'intensité $\lambda > 0$. La fraction volumique de Z (plutôt la surface moyenne par unité de surface ici) est [Chiu et al., 2013]

$$p = \mathbb{P}(0 \in Z) = 1 - \mathbb{P}(0 \notin Z) = 1 - e^{-\lambda \pi \mathbb{E}(r_1^2)} \tag{4.4}$$

où r_1 suit la loi commune des rayons.

Définissons maintenant un modèle booléen dont l'intensité λ varie dans le domaine de l'image de sorte que la fraction volumique correspondent aux niveaux de gris de l'image. Etant donnée une image numérique $u : \{0, \dots, m-1\} \times \{0, \dots, n-1\} \to \{0, \dots, 255\}$, on définit une image normalisée à valeurs dans $[0, 1[$ par $\tilde{u} = \frac{u}{256}$ (on évite la valeur 1 qui correspond à un modèle booléen ayant une intensité λ dégénérée égale à $+\infty$). On définit ensuite une fonction d'intensité constante par morceaux $\lambda : [0, m[\times[0, n[\to \mathbb{R}+$ constante sur le domaine $[k, k+1[\times[l, l+1[$ de chaque pixel (k, l) et telle que la fraction volumique du modèle booléen d'intensité $\lambda(y)$ soit égal à $\tilde{u}(\lfloor y \rfloor)$ (où $\lfloor y \rfloor$ correspond aux coordonnées du pixel contenant le point y). En inversant (4.4), on obtient pour tout $y \in [0, m[\times[0, n[$,

$$\lambda(y) = \frac{1}{\pi \mathbb{E}(r_1^2)} \log\left(\frac{1}{1 - \tilde{u}(\lfloor y \rfloor)}\right).$$

Notre modèle binaire d'image positive à résolution infinie (c'est-à-dire la version positive du négatif, qui n'existe jamais physiquement) est donc le modèle booléen non homogène Z de fonction d'intensité λ définie par l'équation ci-dessus. Comme λ est constante par morceaux sur la partition donnée par la grille de pixels, on peut en fait définir Z comme union de modèles booléens indépendants $Z_{k,l}$ dont les points x_i résident dans le domaine $[k, k+1[\times[l, l+1[$ du pixel (k, l) et sont donnés par un processus de Poisson d'intensité $\lambda((k, l))$.

On a alors un modèle aléatoire $\mathbb{1}_Z$ pour la fonction indicatrice de l'union des cristaux défini sur le domaine continu $\mathbb{R}^2$. Une fois que la liste des centre x_i et des rayons r_i est tirée, on peut alors évaluer la fonction indicatrice $\mathbb{1}_Z(y)$ en tout point y du plan. Cependant, l'image finale doit correspondre à une interpolation de $\mathbb{1}_Z$ par un noyau de flou, typiquement un noyau gaussien g_σ avec un écart-type σ proche de l'unité. Une évaluation exacte du produit de convolution

$$\mathbb{1}_Z * g_\sigma(y) = \int_{\mathbb{R}^2} \mathbb{1}_Z(t) g_\sigma(y - t) dt$$

n'est pas possible car les deux fonctions sont définies sur le domaine continu $\mathbb{R}^2$. Afin d'évaluer ce produit de convolution selon une grille de taille quelconque, nous

utilisons une procédure d'intégration de Monte-Carlo : Soit $(Y_i)_{i\in\mathbb{N}^*}$ une suite de vecteurs gaussiens de $\mathbb{R}^2$ de loi normale de moyenne y et de covariance $\sigma^2 I_2$, alors

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_Z(Y_i) \underset{N\to+\infty}{\longrightarrow} \mathbb{E}(\mathbb{1}_Z(Y_1)) = \int_{\mathbb{R}^2} \mathbb{1}_Z(t) g_\sigma(y-t)dt = \mathbb{1}_Z * g_\sigma(y).$$

En pratique on choisit une grille de taille quelconque que l'on translate N fois par un vecteur gaussien de moyenne nulle et de covariance $\sigma^2 I_2$. On moyenne ensuite les N évaluations binaires pour obtenir une image en niveau de gris. Les expériences montrent que $N = 800$ est suffisant pour ne plus distinguer le bruit d'échantillonnage Monte Carlo dans le bruit du grain argentique. Ce point reste cependant à justifier théoriquement.

Un paramètre important pour l'aspect visuel du bruit est le choix de la distribution des rayons des disques du modèle booléen. Nous utilisons des rayons constants et des rayons suivant une loi log-normale, comme suggéré par [Mees, 1942]. (Pour rappel, une v.a. aléatoire $X > 0$ suit une loi log-normale si $\log(X)$ suit une loi normale). Pour un certain niveau de gris médian et un même noyau de convolution g_σ, la Figure 4.8 montre différent aspects visuels du grain que l'on peut obtenir en faisant varier les paramètres de taille moyenne μ_r et de variance σ_r des rayons dans la loi log-normale. On observe en particulier que le grain devient plus grossier lorsque la taille des rayons augmentent (voir dernière colonne). On observe également que pour des variances élevées (voir dernière ligne), le grain contient des tâches blanches qui correspondent aux rares cristaux de taille anormalement grande. On illustre l'impact du paramètre σ du noyau de flou gaussien sur le grain à la Figure 4.9. Les conclusions de cette expérience sont sans surprise : augmenter le niveau de flou régularise le grain.

Nous présentons à la Figure 4.10 des résultats d'application du grain à des images numériques. Une des contributions importantes de notre approche est que notre modèle booléen nous permet d'effectuer un rendu de l'image correspondante à une résolution arbitraire, ce qui n'est pas possible avec des techniques à base d'images de grain scannées. Par ailleurs, une propriété importe du grain simulé selon notre modèle booléen est qu'il s'adapte à la luminosité de l'image. En effet, comme on peut l'observer à la Figure 4.10, on a bien un bruit parcimonieux dans les zones sombres, un bruit corrélé à forte variance dans les gris médian, et un bruit très fin et uniforme dans les zones très claires.

Enfin une des difficultés de notre approche est de générer rapidement le rendu du modèle booléen considéré. Même en ayant recours à de la parallélisation massive, générer une image de taille 1024×1024 nécessite 188 secondes avec notre implémentation CPU actuelle qui calcule la contribution de chaque disque. Ce temps de calcul est dû au fait que pour une telle image le modèle booléen est constitué de 52 millions de disques. Toutefois, dans le cas de rayons constants, nous avons mis en œuvre une première implémentation GPU à l'aide de l'algorithme de simulation à la demande du processus de Poisson décrit à la Section 3.2 et le temps de calcul est tombé à 3.67 secondes, soit un gain d'un facteur 50. Cette exemple montre encore la

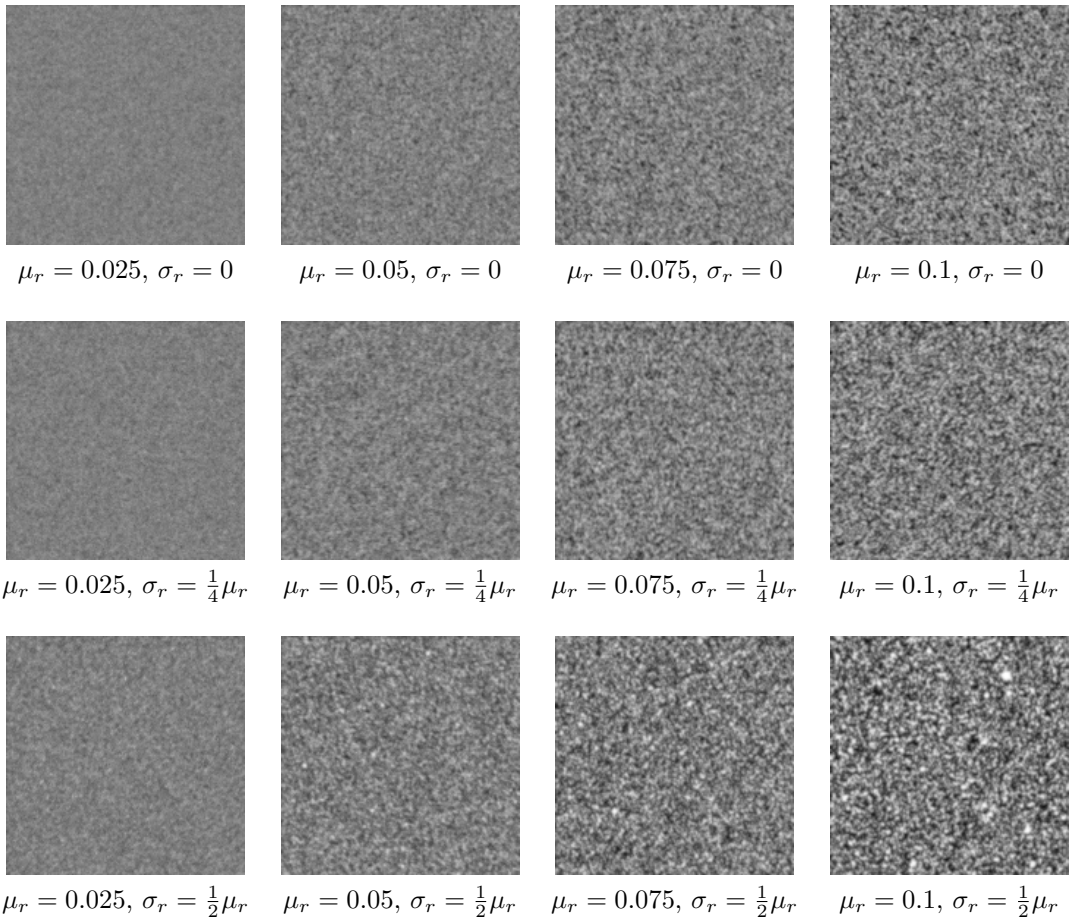


FIGURE 4.8 – Impact de la moyenne μ_r et de la variance σ_r des rayons sur l'aspect visuel du grain (l'unité est le pixel et les images représentées sont de taille 164×164). Les rayons des disques suivent une loi log-normale (ou constante). La taille moyenne des rayons augmente de gauche à droite. La variance des rayons augmentent de haut en bas.

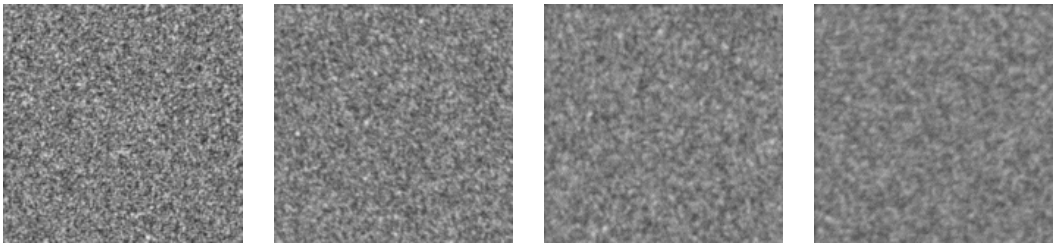
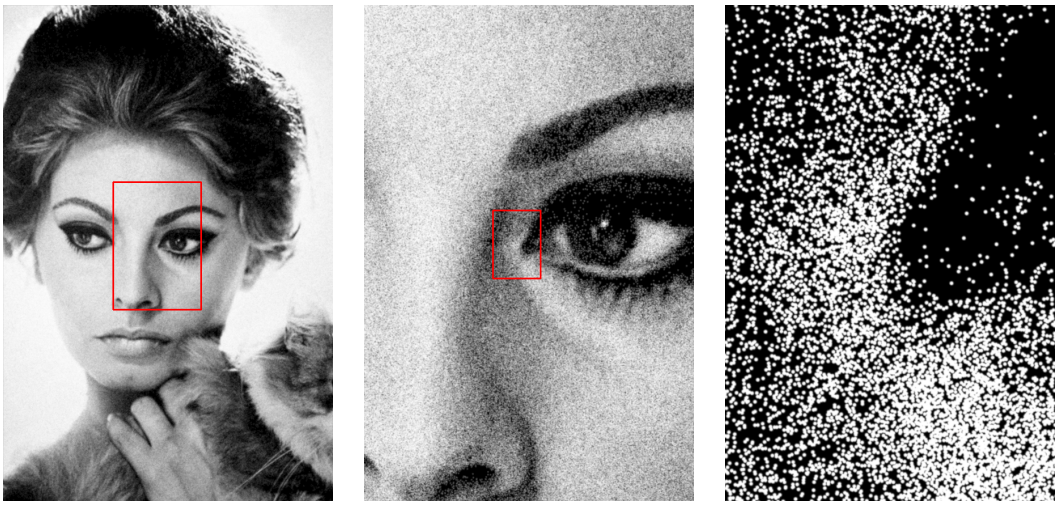
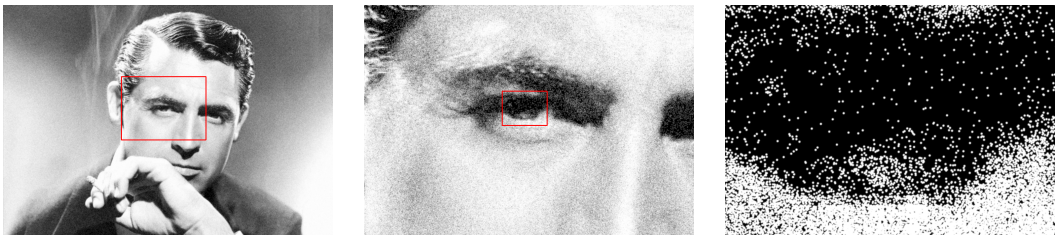


FIGURE 4.9 – Impact du paramètre σ du noyau de flou gaussien sur l'aspect visuel du grain. Pour cette expérience les autres paramètres sont fixés à $\mu_r = 0.05$ et $\sigma_r = \frac{1}{2}\mu_r$.



Full resolution | Zoom 4× | Zoom 30×



Full resolution | Zoom 4× | Zoom 30×

FIGURE 4.10 – Rendu de grain argentique sur des images "vintage" sur trois résolutions différentes.

force de cet algorithme qui permet de ne générer que les données utiles pour chaque point d'intérêt.

4.2 Variation des champs et des ensembles aléatoires

Dans cette section on évoque les résultats des trois articles [J3, J8, J9] qui ont tous les trois pour sujet l'étude de la variation totale moyenne des champs aléatoires et/ou du périmètre moyen des ensembles aléatoires.

Précisons quelques notations. La sphère euclidienne de $\mathbb{R}^d$ est notée S^{d-1} et $\mathcal{H}^{d-1}$ désigne la mesure de Hausdorff de dimension $d-1$. Pour U un ouvert de $\mathbb{R}^d$, on note $L^1(U)$ (resp. $L^1_{\text{loc}}(U)$) l'espace des fonctions intégrables (resp. localement intégrables) sur U, et on note $V \Subset U$ si V est un ouvert relativement compact dans U.

Soit U un ouvert de $\mathbb{R}^d$. On rappelle qu'une fonction $f \in L^1(U)$ est à variation bornée sur U si la dérivée de f au sens des distributions est une mesure de Radon vectorielle Df. Par ailleurs, $f \in L^1(U)$ est à variation bornée sur U dans la direction $u \in S^{d-1}$ si la dérivée directionnelle de f au sens des distributions est une mesure de Radon signée $D_u f$:

$$\forall \varphi \in \mathcal{C}^\infty_c(U, \mathbb{R}), \quad \int_U f(x) \langle \nabla \varphi(x), u \rangle dx = - \int_U \varphi(x) D_u f(dx),$$

où $\mathcal{C}^\infty_c(U, \mathbb{R})$ est l'espace des fonctions $\mathcal{C}^\infty$ à support compact de U dans $\mathbb{R}$. On note alors $|Df|(U)$ la variation totale de f et $|D_u f|(U)$ la variation totale directionnelle de f dans la direction u. On note $BV(U)$ et $BV_u(U)$ les espaces fonctionnels correspondant. On note également $BV_{\text{loc}}(U)$ (resp. $BV_{u,\text{loc}}(U)$) l' espace des fonctions localement intégrables qui sont localement à variation bornée (resp. à variation directionnelle bornée). On renvoie au livre de référence [Ambrosio et al., 2000] pour les propriétés de ces espaces fonctionnelles.

Les variations directionnelles $D_u f$ ne sont pas couramment utilisées bien qu'elles apportent autant d'information que la variation Df. On a notamment la formule de moyennage directionnelle suivante :

$$|Df|(A) = \frac{1}{2\omega_{d-1}} \int_{S^{d-1}} |D_u f|(A) \mathcal{H}^{d-1}(du), \tag{4.5}$$

où ω_{d-1} désigne la mesure de Lebesgue de la boule unité de $\mathbb{R}^{d-1}$. Notre stratégie générale pour l'étude des champs aléatoires à variation bornée est d'obtenir des résultats sur les variations directionnelles pour ensuite les étendre au cadre non directionnel par intégration sur S^{d-1}. L'intérêt de cette approche est principalement technique, mais on peut également remarquer que les résultats directionnels peuvent donner des informations sur le caractère anisotrope des champs aléatoires. L'intérêt technique réside dans le fait que la variation directionnelle s'exprime simplement comme la limite de l'intégrale de la valeur absolue des taux d'accroissements directionnels. Plus précisément, pour une fonction déterministe $f \in L^1(U)$, en notant où

$U \ominus [0, ru] = \{x \in U,\ [x, x + ru] \subset U\}$, on a les deux propriétés suivantes :

$$\int_{U \ominus [0,ru]} \frac{|f(x + ru) - f(x)|}{|r|} dx \leq |D_u f|(U), \quad r \neq 0, \tag{4.6}$$

et

$$\begin{aligned} |D_u f|(U) &= \liminf_{r \to 0} \int_{U \ominus [0,ru]} \frac{|f(x + ru) - f(x)|}{|r|} dx \\ &= \lim_{r \to 0} \int_{U \ominus [0,ru]} \frac{|f(x + ru) - f(x)|}{|r|} dx. \end{aligned} \tag{4.7}$$

A l'aide de ces expressions (que l'on ne trouve pas dans les ouvrages de références sur le sujet [Ambrosio et al., 2000, Evans and Gariepy, 1992]), on peut alors "passer à l'espérance" grâce aux théorèmes usuels de théorie de l'intégration (théorème de Fubini, lemme de Fatou, théorème de convergence dominée de Lebesgue) modulo quelques subtilités concernant la définition des champs aléatoires. Dans le cadre de ce manuscrit on ne développera aucune preuve et on renvoie à l'article [J9] pour un exposé complet.

4.2.1 Champs aléatoires à variation bornée et calcul de leur variation totale moyenne

La première difficulté rencontrée pour définir un champ aléatoire à variation bornée est de déterminer quelle est la bonne notion de champ aléatoire pour ce cadre. En effet, si l'on opte pour la définition classique de champ aléatoire défini par ses distributions finies dimensionnelles, à savoir les lois sur $\mathbb{R}^n$ de chaque vecteur $(f(x_1), f(x_2), \ldots, f(x_n))$ pour $n \in \mathbb{N}^*$ et $x_1, \ldots, x_n$ quelconque, alors l'intégrale de f contre une fonction test φ n'est pas nécessairement mesurable.

On peut alors opter pour se limiter aux champs $f : \Omega \times \mathbb{R}^d \to \mathbb{R}$ qui sont mesurables pour la tribu produit et dont les trajectoires $f(\omega, \cdot)$ sont presque sûrement (p.s.) intégrables. C'est ce choix qui avait été préféré dans la version initiale de ces travaux [T15]. Cependant, aucun de nos résultats ne fait intervenir ce caractère mesurable, mais seulement le caractère mesurable des intégrales de f. Ainsi, nous avons compris que le bon cadre pour l'étude des champs aléatoires BV étaient celui des champs aléatoires intégrables au sens où les champs f sont des variables aléatoires à valeur dans l'espace mesuré $(L^1(U), \mathcal{B}(L^1(U)))$ où $\mathcal{B}(L^1(U))$ est la tribu borélienne induite par la convergence dans $L^1(U)$. Cette approche convient aussi pour l'espace $L^1_{\text{loc}}(U)$ muni de la convergence locale pour laquelle on ne dispose pas d'une norme mais qui reste métrisable. On est alors en mesure de définir les champs aléatoires à variation bornée.

Définition 4.1 (Champs aléatoires à variation (directionnelle) bornée)**.** *Un champ aléatoire intégrable $f \in L^1(U)$ est un* champ aléatoire à variation bornée *dans U si il existe une mesure de Radon aléatoire à valeurs dans $\mathbb{R}^d$ $Df = (D_1 f, \ldots, D_d f)$ telle que $|Df|(U)$ est p.s. finie et telle que pour tout $\varphi = (\varphi_1, \ldots, \varphi_d) \in \mathcal{C}^\infty_c(U, \mathbb{R}^d)$,*

$$\int_U f(x) \operatorname{div} \varphi(x) dx = -\sum_{i=1}^{d} \int_U \varphi_i(x) D_i f(dx) \ \ a.s.$$

Soit $u \in S^{d-1}$. Un champ aléatoire intégrable $f \in L^1(U)$ est un champ aléatoire à variation directionnelle bornée *dans U dans la direction u si il existe une mesure de Radon aléatoire à valeurs réelles $D_u f$ telle que $|D_u f|(U)$ est p.s. finie et telle que pour tout $\varphi \in \mathcal{C}_c^\infty(U, \mathbb{R})$,*

$$\int_{\mathbb{R}^d} f(x) \frac{\partial \varphi}{\partial u}(x) dx = -\int_{\mathbb{R}^d} \varphi(x) D_u f(dx) \text{ a.s.}$$

Un champ aléatoire localement intégrable $f \in L^1_{\text{loc}}(U)$ est un champ aléatoire à variation localement bornée *dans U (resp. un* champ aléatoire à variation directionnelle localement bornée *dans U dans la direction u) si pour tout $V \Subset U$ la restriction de f à V est un champ aléatoire à variation bornée dans V (resp. un champ aléatoire à variation directionnelle bornée dans V dans la direction u).*

On s'intéressera principalement par la suite à l'espérance de la variation totale des champs. On parle alors de mesure d'intensité de variation (directionnelle).

Définition 4.2 (Mesures d'intensité de variation (directionnelle))**.** *Pour tout champ f à variation localement bornée dans U, la mesure d'intensité de la mesure aléatoire $|Df|$, à savoir la mesure $A \mapsto \mathbb{E}(|Df|(A))$, est appelée* mesure d'intensité de variation *de f et est notée $\Theta_V(f, \cdot)$. De même, pour tout champ f à variation directionnelle localement bornée dans U dans la direction $u \in S^{d-1}$, la mesure $A \mapsto \mathbb{E}(|D_u f|(A))$ est appelée* mesure d'intensité de variation directionnelle *de f dans la direction u et est notée $\Theta_{V_u}(f, \cdot)$.*

Il est immédiat de vérifier que la formule (4.5) se traduit par

$$\Theta_V(f, A) = \frac{1}{2\omega_{d-1}} \int_{S^{d-1}} \Theta_{V_u}(f, A) \mathcal{H}^{d-1}(du).$$

Par ailleurs, (4.6) et (4.7) permettent d'établir un théorème de caractérisation simple : $f \in BV_u(U)$ p.s. si et seulement si $\liminf_{r \to 0} \int_{U \ominus [0, ru]} \frac{|f(x+ru) - f(x)|}{|r|} dx$ est finie p.s., et alors la limite existe et on a

$$\Theta_{V_u}(f, U) = \lim_{r \to 0} \int_{U \ominus [0, ru]} \frac{\mathbb{E}(|f(x+ru) - f(x)|)}{|r|} dx. \tag{4.8}$$

Comme discuté en introduction, à notre connaissance, avant 2010 aucun article ne traite des champs aléatoires BV mis à part le court article [Ibragimov, 1995] qui établi que si un champ aléatoire est lipschitzien en moyenne, c'est-à-dire $\mathbb{E}(|f(x) - f(y)|) \leq K\|x - y\|$ pour un certain $K > 0$, alors f est à variation localement bornée et son intensité de variation $\Theta_V(f, \cdot)$ est bornée par un multiple $C\mathcal{L}^d$, $C > 0$, de la mesure de Lebesgue. Grâce aux expressions donnant la variation totale directionnelle en fonction des taux d'accroissement, nous avons pu améliorer le résultat d'Ibragimov d'une part en l'étendant aux fonction à variation bornée directionnelle et d'autre part en déterminant la valeur optimale $C = \frac{d\omega_d}{2\omega_{d-1}} K$ pour la constante de domination (voir [J9, Proposition 4]). Ainsi, un champ aléatoire

lipschitzien en moyenne est à variation bornée. Une des conséquences des résultats qui suivent est que la réciproque est vraie si le champ aléatoire est à accroissements stationnaires (voir [J9, Corollary 1]), notion que nous définissons à présent.

Soit $f : \Omega \times \mathbb{R}^d \to \mathbb{R}$ un champ aléatoire mesurable pour la tribu produit. On dit que f est à accroissements stationnaires si pour tout $y \in \mathbb{R}^d$, les deux champs aléatoires $x \mapsto f(x+y) - f(y)$ et $x \mapsto f(x) - f(0)$ ont les même lois finies dimensionnelles. Le théorème suivant est le principal résultat de l'article [J9].

Théorème 4.2 (Définition et calcul de l'intensité de variation (directionnelle) d'un champ aléatoire à accroissements stationnaires). *Soit $u \in S^{d-1}$ et soit $f : \Omega \times \mathbb{R}^d \to \mathbb{R}$ un champ aléatoire mesurable pour la tribu produit et à accroissements stationnaires tel que $f \in L^1_{\mathrm{loc}}(\mathbb{R}^d)$ p.s. Alors les assertions suivantes sont équivalentes :*

(i) $f \in BV_{u,\mathrm{loc}}(\mathbb{R}^d)$ *p.s. et sa mesure d'intensité de variation directionnelle* $\Theta_{V_u}(f,\cdot)$ *est localement finie.*

(ii) $f \in BV_{u,\mathrm{loc}}(\mathbb{R}^d)$ *p.s. et sa mesure d'intensité de variation directionnelle* $\Theta_{V_u}(f,\cdot)$ *est proportionnelle à la mesure de Lebesgue : il existe une constante* $\theta_{V_u}(f) \geq 0$ *telle que pour tout* $A \in \mathcal{B}(\mathbb{R}^d)$, $\Theta_{V_u}(f,A) = \theta_{V_u}(f)\mathcal{L}^d(A)$.

(iii) $\liminf_{r\to 0} \dfrac{\mathbb{E}(|f(x+ru)-f(x)|)}{|r|} < +\infty$.

(iv) $\lim_{r\to 0} \dfrac{\mathbb{E}(|f(x+ru)-f(x)|)}{|r|}$ *existe et est finie.*

Si une des assertion ci-dessous est vérifiée, alors la constante de proportionnalité $\theta_{V_u}(f)$ est donnée par

$$\theta_{V_u}(f) = \liminf_{r\to 0} \frac{\mathbb{E}(|f(x+ru)-f(x)|)}{|r|} = \lim_{r\to 0} \frac{\mathbb{E}(|f(x+ru)-f(x)|)}{|r|} \tag{4.9}$$

et est appelée intensité de variation directionnelle *de f dans la direction u.*

Concernant la variation non directionnelle, $f \in BV_{\mathrm{loc}}(\mathbb{R}^d)$ p.s. avec une mesure d'intensité de variation $\Theta_V(f,\cdot)$ localement finie si et seulement si pour tout $u \in S^{d-1}$ la limite $\lim_{r\to 0} \dfrac{\mathbb{E}(|f(x+ru)-f(x)|)}{|r|}$ existe et est finie. Dans ce cas $\Theta_V(f,\cdot)$ est proportionnelle à la mesure de Lebesgue, et la constante de proportionnalité $\theta_V(f)$, appelée intensité de variation *de f, est donnée par*

$$\begin{aligned}\theta_V(f) &= \frac{1}{2\omega_{d-1}} \int_{S^{d-1}} \theta_{V_u}(f)\mathcal{H}^{d-1}(du) \\ &= \frac{1}{2\omega_{d-1}} \int_{S^{d-1}} \lim_{r\to 0} \frac{\mathbb{E}(|f(x+ru)-f(x)|)}{|r|}\mathcal{H}^{d-1}(du) \\ &= \lim_{r\to 0} \frac{1}{2\omega_{d-1}} \int_{S^{d-1}} \frac{\mathbb{E}(|f(x+ru)-f(x)|)}{|r|}\mathcal{H}^{d-1}(du).\end{aligned} \tag{4.10}$$

Ainsi, pour les champs à accroissements stationnaires, le comportement moyen de la variation totale se résume à la constante $\theta_V(f)$ qui est facilement calculable

grâce aux expressions (4.9) et (4.10). Nous allons illustrer par la suite l'intérêt de ces résultats généraux en s'intéressant au cas particulier des ensembles aléatoires (Section 4.2.2) puis des champ de type germe-grain (Section 4.2.3). Nous pouvons dès à présent discuter du cas des champs gaussiens stationnaires. En effet, on montre aisément qu'un champ gaussien stationnaire f_{G} est à variation directionnelle localement bornée dans la direction u si et seulement si sa fonction de covariance est deux fois dérivable dans la direction u, et alors $\theta_{V_u}(f_{\mathrm{G}}) = \sqrt{\frac{-2C''_u(0)}{\pi}}$ (et un résultat semblable porte sur les ensembles d'excursions de f_{G} [J9, Proposition 7]). Par conséquent, f_{G} est à variation localement bornée si sa covariance est deux fois dérivable en 0 dans chaque direction u. Mais alors les réalisations de f_{G} sont nécessairement très régulières parmi les fonctions BV. D'une part la différentiabilité d'ordre deux en 0 de la covariance implique que les réalisations sont s-hölderienne pour tout $s < 1$ (voir [Biermé et al., 2007, Proposition 5.2] qui résume des résultats dus à Adler [Adler, 1981]), et donc continues. D'autre part, d'après [Scheuerer, 2010] la double différentiabilité en 0 de la covariance dans chaque direction implique que les réalisations sont dans l'espace de Sobolev $W^{1,2}_{\mathrm{loc}}(\mathbb{R}^d) \subset W^{1,1}_{\mathrm{loc}}(\mathbb{R}^d)$. Donc $f_{\mathrm{G}} \in BV_{\mathrm{loc}}(\mathbb{R}^d)$ implique $f_{\mathrm{G}} \in W^{1,1}_{\mathrm{loc}}(\mathbb{R}^d)$, ce qui montre que le modèle de champs aléatoires BV n'a aucun intérêt pour les champs gaussiens. Ceci est sûrement une des raisons pour lesquelles les champs BV n'ont jamais été étudiés auparavant.

4.2.2 Le cas des ensembles aléatoires

On s'intéresse maintenant au cas particulier des ensembles aléatoires. On discute des liens entre périmètre moyen des ensembles et dérivées directionnelles en 0 de leur covariogramme, puis on aborde rapidement le problème de réalisabilité pour une fonction covariogramme traité dans [J8].

4.2.2.1 Covariogramme d'un ensemble mesurable de périmètre fini

On rappelle que le périmètre (au sens des variations) d'un ensemble mesurable $A \subset \mathbb{R}^d$ de mesure finie est défini par la variation totale de sa fonction indicatrice $\mathrm{Per}(A) = |D\mathbb{1}_A|(\mathbb{R}^d)$ si $\mathbb{1}_A \in BV(\mathbb{R}^d)$ et vaut $+\infty$ sinon. On définit de même la variation directionnelle de A par $V_u(A) = |D_u\mathbb{1}_A|(\mathbb{R}^d)$. Par ailleurs, le covariogramme de A est défini par

$$g_A(y) = \mathcal{L}^d(A \cap (y + A)) = \int_{\mathbb{R}^d} \mathbb{1}_A(x)\mathbb{1}_A(x-y)dx.$$

Nous avons montré dans [J3] que pour tout ensemble mesurable A de mesure finie,

$$\lim_{r\to 0} \frac{g_A(0) - g_A(ru)}{|r|} = \frac{1}{2}V_u\left(A\right),\ u \in S^{d-1}. \tag{4.11}$$

et en notant $\left(g^u_A\right)'(0) := \lim_{r\to 0^+} \dfrac{g_A(ru) - g_A(0)}{r}$

$$\mathrm{Per}(A) = -\frac{1}{\omega_{d-1}} \int_{S^{d-1}} \left(g^u_A\right)'(0)\mathcal{H}^{d-1}(du). \tag{4.12}$$

Ce résultat est assez remarquable puisqu'il montre que cette formule bien connue sous certaines hypothèses de régularité sur A est un fait toujours vraie dès lors que l'on définit convenablement la notion de périmètre.

Les preuves de ces formules sont immédiates en utilisant les équations (4.7) et (4.5) à partir de la constatation suivante due à Matheron [Matheron, 1986]

$$g_A(0) - g_A(y) = \frac{1}{2} \int_{\mathbb{R}^d} |\mathbb{1}_A(x+y) - \mathbb{1}_A(x)| \, dx$$

qui s'obtient au passant au carré dans l'intégrale (car 0 et 1 sont égaux à leur carré !).

Enfin, on mentionne que (4.6) et l'inégalité (étonnamment inédite !)

$$|g_A(y) - g_A(z)| \leq g_A(0) - g_A(y-z) \quad y, z \in \mathbb{R}^d.$$

permettent de déterminer la constante de Lipschitz du covariogramme d'un ensemble mesurable A qui est

$$\mathrm{Lip}\,(g_A) = \frac{1}{2} \sup_{u \in S^{d-1}} V_u(A).$$

L'expression de $\mathrm{Lip}\,(g_A)$ n'avait jamais été établie et même dans le cas d'un ensemble A convexe, la borne sur $\mathrm{Lip}\,(g_A)$ due à Matheron était surestimée d'un facteur 2. Grâce à cette expression de $\mathrm{Lip}\,(g_A)$, les auteurs de [Bianchi et al., 2011] ont pu diviser par deux certaines majorations d'erreurs intervenant dans leur méthode de reconstruction d'un ensemble convexe à partir de son covariogramme (qui est un problème de restitution de phase de la transformée de Fourier de $\mathbb{1}_A$).

4.2.2.2 Ensembles mesurables aléatoires

Il est naturel de vouloir étendre les résultats précédents aux ensembles aléatoires. Cependant on se heurte là encore à des problèmes de définition des objets aléatoires. Le cadre usuel pour l'étude des ensembles aléatoires est celui des ensembles aléatoires fermés [Matheron, 1975, Molchanov, 2005] abrégés par RACS pour *random closed sets*. Les résultats sur le covariogramme s'étendent au RACS comme présenté dans l'article [J3]. Cependant, là encore ni la notion de périmètre moyen $\mathbb{E}(\mathrm{Per}(X))$ d'un RACS X, ni la notion de covariogramme $\gamma_X(y) = \mathbb{E}(\mathcal{L}^d(X \cap (y+X)))$ ne font appel à la topologie de X. En revanche, ce sont tous deux des invariants de la classe de Lebesgue de X. Ceci invite à définir des ensembles aléatoires mesurables, abrégés RAMS pour *random measurable sets*, comme des variables aléatoires dans l'ensemble $\mathcal{B}(\mathbb{R}^d)$ muni de la tribu borélienne pour la convergence dans L^1_{loc} des fonctions indicatrices (on parle de convergence locale de mesure [Ambrosio et al., 2000]).

La définition formelle des RAMS et la comparaison de cette notion d'ensembles aléatoires avec la notion classique de RACS est l'objet de la Section 2 de [J8]. On y montre notamment qu'un RAMS de périmètre localement fini est toujours confondu avec un RACS si la dimension d vaut 1, alors que ce n'est généralement pas le cas dès lors que $d \geq 2$.

Finalement pour tout RAMS X, $\mathrm{Per}(X)$ et $V_u(X)$ sont des variables aléatoires bien définies et on a

$$\lim_{r\to 0} \frac{\gamma_X(0)-\gamma_X(ru)}{|r|} = \frac{1}{2}\mathbb{E}(V_u(X)), \quad u \in S^{d-1}.$$

et en notant $(\gamma_X^u)'(0) = \lim_{r\to 0^+} \dfrac{\gamma_X(ru)-\gamma_X(0)}{r}$,

$$-\frac{1}{\omega_{d-1}} \int_{S^{d-1}} (\gamma_X^u)'(0)\mathcal{H}^{d-1}(du) = \mathbb{E}\left(\mathrm{Per}(X)\right).$$

4.2.2.3 Intensité de variation et variogramme des ensembles aléatoires stationnaires

Discutons maintenant rapidement de l'application du Théorème 4.2 dans le cas où f est la fonction indicatrice d'un ensemble X stationnaire (mesurable pour la tribu produit en reprenant les hypothèses du théorème). Alors,

$$\mathbb{E}(|f(ru)-f(0)|) = \mathbb{P}(ru \in X,\ 0 \notin X)+\mathbb{P}(ru \notin X,\ 0 \in X) = 2\mathbb{P}(ru \in X,\ 0 \notin X) = 2\nu_X(ru)$$

où $\nu_X(y) = \mathbb{P}(y \in X,\ 0 \notin X)$ est le variogramme de l'ensemble X. En notant, $(\nu_X^u)'(0) = \lim_{r\to 0} \dfrac{1}{|r|}\nu_X(ru)$, on obtient que

$$\theta_{V_u}(X) = 2\lim_{r\to 0} \frac{\mathbb{E}(|f(ru)-f(0)|)}{|r|} = 2(\nu_X^u)'(0),$$

et ainsi en intégrant selon chaque direction,

$$\theta_V(X) = \frac{1}{\omega_{d-1}} \int_{S^{d-1}} (\nu_X^u)'(0)\mathcal{H}^{d-1}(du). \tag{4.13}$$

Cette formule avait déjà été étendue pour tout RACS stationnaire dans [J3], mais le Théorème 4.2 montre qu'elle est valable pour tout ensemble stationnaire mesurable pour la tribu produit. Là encore cette formule était déjà établie selon certaines hypothèses de régularité depuis les premiers travaux de Matheron [Matheron, 1967, p. 30] (on renvoie à la discussion correspondante de notre article [J3] où l'on a tenté d'être exhaustif au niveau des généralisations de cette formule).

En conclusion, ces résultats montrent que la bonne notion de périmètre associé aux dérivées directionnelles du covariogramme ou du variogramme est toujours la notion de périmètre au sens des variations, et que finalement le rôle des hypothèses techniques de régularité sur les ensembles X rencontrées dans la littérature permettaient simplement d'assurer que les différentes notions de périmètre (comme contenu de Minkowski ou la mesure de Hausdorff de la frontière) coïncidaient avec le périmètre au sens des variations.

4.2.2.4 Problème de réalisabilité d'un covariogramme

Nous terminons notre discussion sur le lien entre covariogramme et périmètre moyen d'un ensemble mesurable aléatoire (abrégé RAMS pour rappel) en évoquant le résultat principal de notre article [J8]. Ce travail est issu d'une collaboration avec R. Lachièze-Rey et fait suite aux travaux de R. Lachièze-Rey et I. Molchanov [Lachièze-Rey and Molchanov, 2015] sur le problème de réalisabilité. On renvoie à cet article pour une introduction formelle et générale au problème de réalisabilité que nous allons illustré ici uniquement lorsque la fonctionnelle d'intérêt est le covariogramme spécifique γ_X^s des ensembles stationnaires (qui est en fait le complémentaire du variogramme $\gamma_X^s = 1 - \nu_X$).

Le problème de réalisabilité du covariogramme spécifique est le suivant : Etant donné une fonction $S_2 : \mathbb{R}^d \to \mathbb{R}$, existe-t-il un RAMS stationnaire X tel que $S_2(y) = \gamma_X^s$ pour tout $y \in \mathbb{R}^d$? En suivant [Lachièze-Rey and Molchanov, 2015], notre résultat principal apporte une caractérisation de ce problème lorsque l'on s'intéresse aux RAMS d'intensité de variation fini (*i.e.* de périmètre moyen fini).

Afin de présenter ce résultat on doit introduire la notion d'admissibilité d'une fonction covariogramme spécifique. Pour cela on passe par la notion d'admissibilité d'un covariogramme local. Etant donné un ensemble mesurable A (déterministe), on appelle covariogramme local de A l'application

$$\delta_{y;W}(A) = \mathcal{L}^d(A \cap (y + A) \cap W), (y, W) \in \mathbb{R}^d \times \mathcal{W},$$

où $\mathcal{W}$ désigne l'ensemble des fenêtres d'observation défini par

$$\mathcal{W} = \left\{ W \subset \mathbb{R}^d \text{ ouvert borné t.q. } \mathcal{L}^d(\partial W) = 0 \right\}.$$

Pour un RAMS X, le covariogramme local (moyen) de X est $\gamma_X(y; W) = \mathbb{E}(\delta_{y;W}(X))$. Si X est stationnaire, $W \mapsto \gamma_X(y; W)$ est invariant par translation et se prolonge en une mesure proportionnelle à la mesure de Lebesgue. La constante de proportionnalité est obtenue pour $W =]0, 1[^d$, c'est le covariogramme spécifique de X, noté $\gamma_X^s(y) = \gamma_X(y;]0, 1[^d)$. On introduit alors la définition d'admissibilité pour le covariogramme.

Définition 4.3 (Fonctions admissibles pour le covariogramme)**.** *Une fonction* $\gamma : \mathbb{R}^d \times \mathcal{W} \to \mathbb{R}$ *est dite* admissible pour le covariogramme local*, ou simplement* admissible*, si pour tous les 5-uplets* $(q \geq 1, (a_i) \in \mathbb{R}^q, (y_i) \in (\mathbb{R}^d)^q, (W_i) \in \mathcal{W}^q, c \in \mathbb{R})$*,*

$$\left[\forall A \in \mathcal{B}(\mathbb{R}^d), \quad c + \sum_{i=1}^{q} a_i \delta_{y_i;W_i}(A) \geq 0 \right] \Rightarrow c + \sum_{i=1}^{q} a_i \gamma(y_i; W_i) \geq 0.$$

Une fonction $S_2 : \mathbb{R}^d \to \mathbb{R}$ *est dite* admissible pour le covariogramme spécifique*, ou simplement* admissible*, si la fonction* $(y; W) \mapsto S_2(y)\mathcal{L}^d(W)$ *est admissible pour le covariogramme local.*

Par linéarité et positivité de l'espérance, il est immédiate de constater que toute fonction covariogramme réalisable est bien admissible. En revanche, la réciproque est

fausse. Etant donnée une fonction S_2 admissible pour le covariogramme spécifique, il n'est pas toujours possible de construire un RAMS X tel que $S_2 = \gamma_X^s$. Un des principaux résultats de [Lachièze-Rey and Molchanov, 2015] est de montrer que si l'on accompagne le problème de réalisabilité avec le contrôle des valeurs d'un *module de régularité* satisfaisant certaines propriétés, alors la réciproque attendue devient vraie. Dans notre cas, c'est l'intensité de variation θ_V qui joue ce rôle de régularité et nous avons pu démontrer le résultat suivant.

Théorème 4.3. *Soit $S_2 : \mathbb{R}^d \to \mathbb{R}$ une fonction. Alors S_2 est le covariogramme spécifique d'un RAMS stationnaire $X \in \mathcal{B}(\mathbb{R}^d)$ ayant une intensité $\theta_V(X)$ finie si et seulement si S_2 est admissible et Lipschitzien en 0 dans chacune des d directions canoniques.*

Notons qu'il n'était pas possible d'établir ce résultat dans le cadre classique des ensembles fermés aléatoires (RACS), car pour la topologie des ensembles fermés aléatoires le périmètre ne satisfait pas les hypothèses de semi-continuité inférieure et de compacité des ensembles de niveaux nécessaires pour être un module de régularité [Lachièze-Rey and Molchanov, 2015] [J8]. Cependant, nous avons montré que le résultat pouvait s'étendre au RACS en dimension $d = 1$, car alors la géométrie des ensembles de périmètre fini et celle des ensembles fermés est la même [J8]. Une fois encore, ce résultat démontre la pertinence de l'utilisation des RAMS et de la notion de périmètre au sens des variations pour la géométrie stochastique.

4.2.3 Variation totale moyenne des modèles germe-grain

Nous avons défini les champs aléatoires à variation borné, établi une formule pour le calcul de leur intensité de variation d'un champ à accroissement stationnaire, et nous avons traité le cas particulier des ensembles aléatoires en reliant le périmètre moyen aux dérivées directionnelles en 0 du covariogramme. Nous disposons donc désormais de tous les éléments nécessaires pour revenir à la motivation initiale de cette série de travaux, à savoir déterminer la variation totale moyenne des champs aléatoires de type germe-grain. Ces illustrations sont toutes établies avec les hypothèses minimales sur les grains. Leur intérêt et de montrer que pour l'intensité de variation, seuls le périmètre moyen et le volume moyen des grains interviennent. Les modèles pour lesquels nous donnons les formules d'intensité de variation sont tous illustrés pas la Figure 4.11 (on omet l'exemple des partitions aléatoires également traité dans [J9, Section 6.2]). Afin de ne pas alourdir la présentation, on considère des grains X_i qui sont des RACS $X \in \mathcal{F}(\mathbb{R}^d)$ même si les résultats s'étendent à priori à des RAMS. On renvoie également à [J9, Section 6.2] pour les énoncés précis correspondant aux formules associées.

Remarque (Modèles gaussiens associés aux covariances des modèles germe-grain)**.** Tous les modèles germe-grain ont une covariance dont la régularité en 0 dépend de la régularité du covariogramme des grains. Or, le covariogramme n'est jamais dérivable en 0 (au mieux il est Lipschitz en 0 avec des dérivées directionnelles non

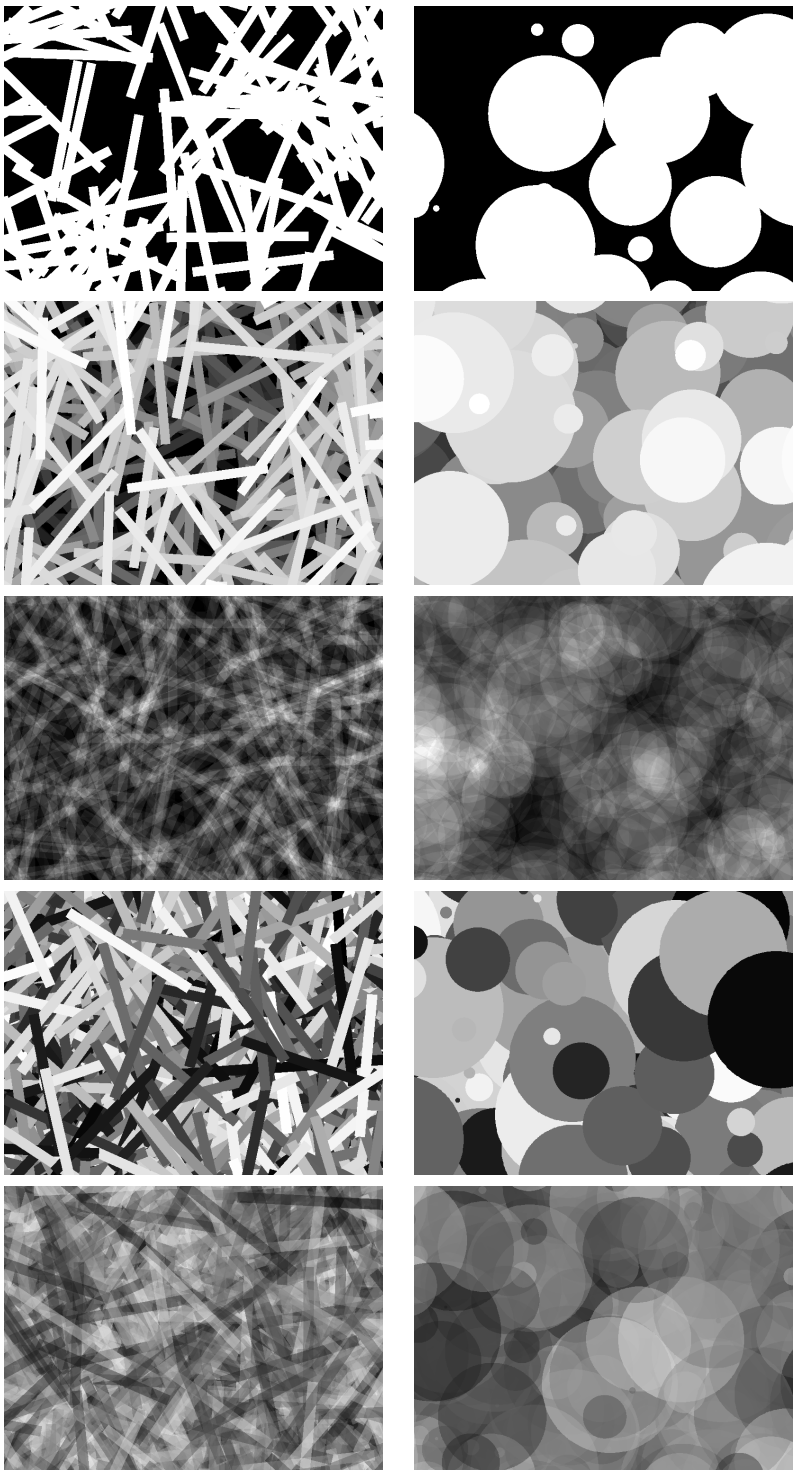


FIGURE 4.11 – Quelques réalisations de champs aléatoires de type germe-grain. De haut en bas : ensemble booléen, champ booléen, shot noise poissonnien, modèle feuilles mortes, modèles feuilles mortes transparentes. Pour la colonne de gauche, les grains sont des rectangles avec des orientations aléatoires uniformes, pour la colonne de droite, les grains sont des disques avec un rayon uniforme dans un intervalle $[0, R_{\max}]$.

nulles). En particulier, les champs gaussiens associés à ces covariances ont tous des variations totales moyennes infinies alors que les modèles germe-grain ont eux une variation totale moyenne finie lorsque les grain sont de périmètre moyen fini. Cette observation est un point supplémentaire pour souligner la pertinence de l'utilisation de la variation totale pour les indices de *sharpness* [Blanchet et al., 2008, Blanchet and Moisan, 2012, Leclaire and Moisan, 2015].

Ensembles booléens et fonctions booléennes Partant d'un processus de Poisson marqué $\Phi = \{(x_i, X_i)\} \subset \mathbb{R}^d \times \mathcal{F}(\mathbb{R}^d)$ de mesure d'intensité $\lambda\mathcal{L}^d \otimes P_X$, l'ensemble booléen associé à Φ est

$$Z_{\mathrm{B}} = \bigcup_{i\in\mathbb{N}} x_i + X_i,$$

comme introduit à la Section 4.1.3. L'expression du variogramme d'un modèle booléen est bien connu, et en appliquant nos différents résultats on obtient que

$$\theta_{V_u}(Z_{\mathrm{B}}) = \lambda\mathbb{E}\left(V_u(X)\right)\exp\left(-\lambda\mathbb{E}\left(\mathcal{L}^d\left(X\right)\right)\right), \quad u \in S^{d-1},$$

$$\theta_{V}(Z_{\mathrm{B}}) = \lambda\mathbb{E}\left(\mathrm{Per}(X)\right)\exp\left(-\lambda\mathbb{E}\left(\mathcal{L}^d\left(X\right)\right)\right). \tag{4.14}$$

Encore une fois, comme discuté dans [J3], l'équation (4.14) est satisfaite pour une loi quelconque P_X et elle généralise donc les résultats connus pour les modèles booléens ayant des grains convexes [Schneider and Weil, 2008, p. 386]. Des généralisations similaires faisant intervenir des notions d'intensités surfaciques dérivant de la formule de Steiner ont aussi été récemment proposées [Hug et al., 2004, Villa, 2010].

Si l'on attache maintenant un niveau de gris indépendant $a_i \in \mathbb{R}+$ à chaque point de Φ on peut définir la fonction booléenne f_{B} par [Serra, 1988]

$$f_{\mathrm{B}}(y) = \sup\left(\{0\} \cup \{a_i,\ y \in x_i + X_i\}\right).$$

Comme chaque ensemble de niveau $\{y,\ f_{\mathrm{B}}(y) > t\}$ de f_{B} est un ensemble booléen de mesure d'intensité $\mathbb{P}_a(a > t)\lambda\mathcal{L}^d \otimes P_X$, en appliquant la formule de la coaire pour les intensités de variations (voir [J9, Proposition 6] on obtient les formules suivantes :

$$\theta_{V_u}\left(f_{\mathrm{B}}\right) = \lambda\mathbb{E}\left(V_u(A)\right)\int_0^{+\infty} P_a(\{a > t\})\exp\left(-\lambda\mathbb{E}\left(\mathcal{L}^d\left(X\right)\right)P_a(\{a > t\})\right)dt, \quad u \in S^{d-1},$$

$$\theta_{V}\left(f_{\mathrm{B}}\right) = \lambda\mathbb{E}\left(\mathrm{Per}(A)\right)\int_0^{+\infty} P_a(\{a > t\})\exp\left(-\lambda\mathbb{E}\left(\mathcal{L}^d\left(X\right)\right)P_a(\{a > t\})\right)dt.$$

Shot noise poissonniens de fonctions indicatrices On considère ici un processus de Poisson $\Phi = \{(x_i, X_i, a_i)\} \subset \mathbb{R}^d \times \mathcal{F}(\mathbb{R}^d) \times \mathbb{R}$ de mesure d'intensité $\lambda\mathcal{L}^d \otimes P_X \otimes P_a$ et le processus shot noise associé

$$f_{\mathrm{SN}}(x) = \sum_{(x_i,a_i,X_i)\in\Phi} a_i \mathbb{1}(x \in x_i + X_i).$$

On a alors pour ces champs

$$\theta_V(f_{\mathrm{SN}}) = \lambda\mathbb{E}(|a|)\mathbb{E}(\mathrm{Per}(X)) \quad \text{and} \quad \theta_{V_u}(f_{\mathrm{SN}}) = \lambda\mathbb{E}(|a|)\mathbb{E}(V_u(X)), \quad u \in S^{d-1}.$$

L'expression de $\theta_V(f_{\mathrm{SN}})$ est une conséquence de [Biermé and Desolneux, 2016, Theorem 3]. On en déduit l'expression de $\theta_{V_u}(f_{\mathrm{SN}})$ en remarquant que l'inégalité $\theta_{V_u}(f_{\mathrm{SN}}) \leq \lambda\mathbb{E}(|a|)\mathbb{E}(V_u(X))$, qui est valable pour tous les champs shot noise (somme de fonctions intégrables et pas seulement d'indicatrices) [J9], est ici saturée.

Modèles feuilles mortes et feuilles mortes transparentes Comme présenté en Section 4.1.1, les modèles feuilles mortes et feuilles mortes transparentes sont construits à partir d'un processus de Poisson indépendamment marqué $\Phi = \{(t_i, x_i, X_i, a_i),\ i \in \mathbb{N}\} \subset]-\infty, 0] \times \mathbb{R}^d \times \mathcal{F}(\mathbb{R}^d) \times \mathbb{R}$ de mesure d'intensité $\mathcal{L}^1 \otimes \mathcal{L}^d \otimes P_X \otimes P_a$. Le modèle FMT est alors le champ aléatoire

$$f_{\mathrm{FMT}}(y) = \sum_{i\in\mathbb{N}} \mathbb{1}\left(y \in x_i + X_i\right) \alpha a_i \left(1-\alpha\right)^{\left(\sum_{j\in\mathbb{N}} \mathbb{1}(t_j \in (t_i,0) \text{ and } y \in x_j + X_j)\right)}.$$

et le modèle feuilles mortes correspond au cas $\alpha = 1$. Toutefois, mentionnons qu'il est plus facile et naturel de définir le modèle feuilles mortes à l'aide de la partition de $\mathbb{R}^d$ en "parties visibles" (voir [Bordenave et al., 2006]). On obtient alors

$$\theta_{V_u}(f_{\mathrm{FMT}}) = C_\alpha \frac{E\left(V_u(X)\right)}{E\left(\mathcal{L}^d(X)\right)}, \quad u \in S^{d-1},$$

où C_α est le contraste moyen entre une couleur indépendante a_1 et le modèle FMT, c'est-à-dire,

$$C_\alpha = \mathbb{E}(|a - f_{\mathrm{FMT}}(0)|) = E\left(\left|a - \sum_{k=0}^{+\infty} \alpha a_k (1-\alpha)^k\right|\right),$$

où les v.a. a et $(a_k)_{k\in\mathbb{N}}$ sont i.i.d. de loi P_a. Par conséquent,

$$\theta_V(f_{\mathrm{FMT}}) = C_\alpha \frac{E\left(\mathrm{Per}(X)\right)}{E\left(\mathcal{L}^d(X)\right)}.$$

En particulier, pour $\alpha = 1$ le terme de contraste C_α devient $\mathbb{E}(|a_1 - a_2|)$, soit le contraste moyen entre deux feuilles adjacentes. On remarque que le rapport $\frac{\mathbb{E}(\mathrm{Per}(X))}{\mathbb{E}\left(\mathcal{L}^d(X)\right)}$ est connu pour être la longueur moyenne de frontière entre les cellules de la partition du modèle feuilles mortes lorsque les RACS sont des polygones [Cowan and Tsang, 1994]. L'intensité de variation est donc facilement interprétable ici : c'est la longueur moyenne de sauts multipliée par la hauteur moyenne d'un saut.

Chapitre 5

Quelques perspectives de recherches

Mise à part la rédaction d'un article long sur l'inpainting de textures gaussiennes (voir Section 2.4 et [C14]), j'envisage désormais d'arrêter de travailler sur la synthèse de textures gaussiennes. En effet, d'une part grâce au *texton noise* [J11] je pense que nous sommes arrivés à une limite de performance pour la génération de ces textures. D'autre part, l'ensemble des textures gaussiennes reste assez restreint et ne semble plus susciter un grand intérêt dans la communauté du *computer graphics*.

Je compte toutefois poursuivre mes recherches sur des problématiques similaires, à savoir l'étude et l'élaboration de méthodes numériques pour la génération d'images reposant sur des modèles stochastiques, mais en faisant appel à des outils mathématiques différents. Je détaille principalement ci-dessous deux perspectives, le transport optimal numérique pour la synthèse de textures solides et l'étude des processus ponctuels déterminantaux pour la synthèse d'images.

5.1 Transport optimal pour la synthèse de textures

Le transport optimal entre lois de probabilités est un outil naturel pour imposer à une image de respecter certaines statistiques. Il a notamment été utilisé en synthèse de textures pour des méthodes par ondelettes (plutôt des steerable pyramide comme pour [Heeger and Bergen, 1995, Portilla and Simoncelli, 2000]) dans [Rabin et al., 2012].

Le projet Texto (Projet Jeunes Chercheurs du GdR ISIS) en collaboration avec Julien Rabin (GREYC, Université de Caen) et Arthur Leclaire (CMLA, ENS Cachan) a pour but d'apporter des garanties de préservation de statistiques globales dans les méthodes de synthèse de textures par patchs. En effet, un défaut majeur de la plupart des algorithmes de synthèse par patchs tels que [Kwatra et al., 2005] est qu'ils cherchent à respecter des structures locales mais pas à préserver des statistique globales des images. Pour cela nous avons l'ambition d'utiliser des algorithmes pour le transport optimal numérique afin de préserver certaines distributions de l'image d'entrée. Notre travail a jusqu'ici essentiellement consisté en un état de l'art sur les méthodes numériques pour le transport optimal dont aucune n'est réellement adapté à notre problème en dimension élevée. Il existe en effet de nombreux contextes pour le transport optimal numérique, notamment selon la nature discrète ou continue des distributions ou selon les approximations du problème de transport. Les méthodes

numériques sont nombreuses et ce domaine connaît actuellement de nombreux développements avec notamment la régularisation entropique du transport optimal introduite par M. Cuturi [Cuturi, 2013] (voir par exemple les travaux de la nouvelle équipe INRIA MOKAPLAN). Après m'être familiarisé avec ces nouveaux outils, les deux perspectives suivantes me semblent les plus prometteuses.

5.1.1 Imposition de multi-histogrammes par transport optimal numérique dans les méthodes par patchs pour la synthèse de textures solides

La synthèse de textures solides par l'exemple a pour le but de synthétiser un volume de texture dont chaque coupe correspond à une texture donnée en entrée. Une des méthodes de l'état de l'art [Kopf et al., 2007] se base sur des méthodes heuristiques pour préserver l'histogramme des couleurs de l'image d'entrée afin de garantir une synthèse de qualité. En effet une des spécificités de la synthèse de texture solide par l'exemple est que l'on dispose de voisinage assez pauvre et sans volume (union de trois patchs dans chaque plan principal) ce qui implique que l'intersection des voisinages de deux voxels voisins est très réduite. Il est donc important d'imposer une cohérence globale dans le volume synthétisé. Nous espérons que des outils de transport optimal numérique puissent permettre de mieux formaliser et d'augmenter les garanties de préservation des statistiques pour les volumes synthétisés par rapport à l'approche heuristique de [Kopf et al., 2007]. Cette perspective fait actuellement l'objet du stage de master de Jorge Gutierrez (étudiant en Master 2 de l'Université de Bordeaux) co-encadré avec J. Rabin et Thomas Hurtut (école polytechnique de Montréal) et se déroulant au GREYC.

La synthèse de textures solides et ses liens avec les méthodes de synthèse de textures procédurales sera plus largement le sujet de la thèse de J. Gutierrez qui débutera à la rentrée 2016 à Montréal sous la direction de T. Hurtut et toujours avec J. Rabin et moi-même comme co-encadrants.

5.1.2 Transport optimal numérique semi-discret par méthode Monte-Carlo

Un autre outil pour le transport optimal numérique sont les algorithmes spécifiques au cadre dit semi-discret où l'on cherche à transporter une mesure à densité vers une mesure discrète de somme de Dirac (avec poids). Il est connu que la solution de ce type de problème correspond à assigner à chaque masse de Dirac les points d'une cellule d'une partition de Voronoï généralisée appelé diagramme de puissance [Aurenhammer et al., 1998].

Les approches numériques usuelles [Mérigot, 2011, Lévy, 2015] reposent sur la minimisation d'une fonctionnelle convexe par méthode de quasi-Newton (et une preuve de convergence a récemment été obtenue pour un algorithme de Newton amorti [Kitagawa et al., 2016]). Le calcul de la fonctionnelle repose sur le calcul explicite de la géométrie des cellules de Voronoï généralisée, en faisant appel à

des méthodes de géométrie algorithmique. Ainsi, ces méthodes sont limitées à la dimension 2 et à la dimension 3, et il faut déjà faire appel à des librairies spécialisées de géométrie computationnelle pour les implémenter efficacement (voir les résultats impressionnants de B. Lévy [Lévy, 2015]).

Nous avons cependant remarqué que le calcul de la fonctionnelle, qui repose essentiellement sur le calcul d'intégrale sur chaque cellule, pourrait tout à fait être calculé de manière approchée par une simulation Monte Carlo. L'intérêt d'une telle approche est qu'elle remplace le calcul d'une structure géométrique complexe par la simple recherche de distance minimale effectuée un très grand nombre de fois. En outre, cette approche Monte Carlo doit permettre a priori de traiter des dimensions supérieures à 3. En revanche, nos premiers tests montrent que la limitation de cette approche se situera dans le nombre N de sommes de Dirac car chaque évaluation de la fonctionnelle et de son gradient nécessite a un coût de calcul de l'ordre de N^2. Cependant, cette évaluation d'intégrales de Monte Carlo est fortement parallélisable et nous espérons démontrer avec A. Leclaire et J. Rabin que cette approche numérique est compétitive pour certaines applications. Une des applications envisagées est la correction non linéaire de la distribution couleur gaussienne de bruit procéduraux de type *texton noise*.

5.2 Processus ponctuels déterminantaux pour la synthèse d'images

J'ai commencé à travailler en collaboration avec Agnès Desolneux sur l'utilisation des processus ponctuels déterminantaux pour la modélisation d'images pendant mon année de CRCT (2015-2016). Cette première étude nous a confirmé que ce sujet était très prometteur et nous avons décidé de proposer un sujet de thèse regroupant nos problématiques. Ce sujet de thèse a intéressé Claire Launay, une étudiante de Paris Descartes lauréate de la bourse d'excellence PGSM master de la Fondation Sciences Mathématiques de Paris (FSMP) et ayant suivi en 2015-2016 le master MVA à l'ENS Cachan. Claire Launay vient de commencer sa thèse sous notre direction en octobre 2016 grâce à l'obtention d'une bourse financée par le Réseau de Recherche Doctoral en Mathématiques de l'Île de France (RDM-IdF), après sélection par la FSMP.

La description qui suit de cette perspective de recherche est adaptée du sujet de thèse rédigé avec Agnès Desolneux.

5.2.1 Contexte

Les processus ponctuels déterminantaux (PPD) sont des modèles d'ensembles de points aléatoires qui suscitent actuellement un grand intérêt dans plusieurs communautés mathématiques. Initialement étudiés en probabilités, notamment pour la modélisation des fermions en mécanique quantique ou encore comme processus décrivant le spectre de certaines matrices aléa-

toires [Hough et al., 2009, Soshnikov, 2000], de récentes contributions en statistiques spatiales [Lavancier et al., 2015] et en apprentissage statistique (*i.e. machine learning*) [Kulesza and Taskar, 2012] ont démontré que ces modèles ont un fort potentiel applicatif.

L'intérêt premier des PPD est qu'ils fournissent une famille de modèles de configurations aléatoires de points ayant un caractère répulsif, au sens où la probabilité d'observer deux points proches l'un de l'autre est plus faible que dans le cas du processus de Poisson dont les points sont indépendants [Biscio and Lavancier, 2016]. Les PPD sont complètement déterminés par leur fonction de corrélation. Contrairement aux processus ponctuels de Gibbs tels que les processus de Strauss, les moments des PPD sont tous connus à partir de leur fonction de corrélation. De plus, les PPD peuvent être simulés séquentiellement de manière exacte et ne nécessitent pas d'avoir recourt à des algorithmes itératifs approchés de type MCMC (même si une simulation par MCMC peut avoir un intérêt pratique dans certains contextes en *machine learning* [Kang, 2013]).

5.2.2 Processus déterminantaux dans le cadre des images

On se propose d'étudier et de développer les processus ponctuels déterminantaux (PPD) dans un cadre bidimensionnel stationnaire qui sera discret (grille de pixels) ou continu (plan $\mathbb{R}^2$ ou tore $\mathbb{T}^2$) selon le contexte. Le lien entre les modèles discrets et continus sera une des problématiques de ces travaux puisque la question de l'échantillonnage selon une grille discrète est centrale en traitement d'images. Dans ce cadre, l'algorithme de simulation spectrale [Hough et al., 2009, Lavancier et al., 2015] est aisément mis en œuvre puisque l'opérateur de corrélation est diagonalisé par la transformée de Fourier. Pour les mêmes raisons, dans ce contexte stationnaire les opérateurs de corrélations commutent et il est envisageable de comparer différents PPD entre eux. Pour cela on déterminera une expression de la distance de transport optimal (distance de Wasserstein) entre deux mesures empiriques de PPD et on étudiera l'existence de barycentres de Wasserstein [Agueh and Carlier, 2011] pour ces ensembles de points.

5.2.3 Shot noise basé sur des processus déterminantaux

Comme on l'a vu aux Chapitres 2 et 3, les modèles shot noise basés sur des processus de Poisson permettent de synthétiser des textures gaussiennes, aussi bien en discret [J1] [C13] qu'en continu avec des textures procédurales [van Wijk, 1991] [J5, J11].

Intuitivement, un modèle shot noise basé sur un PPD sera moins sujet aux effets de moyennage de la fonction noyau que l'on observe très vite pour une processus shot noise poissonnien, et qui résulte par des images de textures ne présentant aucun contour net. On peut espérer qu'un PPD ayant une fonction de corrélation adaptée au support de la fonction noyau du shot noise permette de créer des textures présentant plus de détails localement orientés. Or ces détails sont bien souvent très

importants visuellement pour la synthèse de textures naturelles, mais aussi pour les textures issues de différentes modalités d'imagerie médicale et de science des matériaux.

Afin d'étudier les shot noise reposant sur des PPD, nous allons dans un premier temps établir les différences théoriques et visuelles avec le cas bien connu des shot noise poissonniens. Sur un plan plus théorique, on envisagera une étude géométrique des ensembles de niveaux (excursions) de ces champs [Biermé and Desolneux, 2016] ainsi qu'une étude des moments d'ordre supérieur à deux des champs shot noise et leur différence avec les moments des champs gaussiens.

5.2.4 Processus ponctuels déterminantaux pour l'échantillonnage stochastique

L'antialiasing stochastique est une technique couramment utilisée en *computer graphics* (infographie) et qui consiste à échantillonner une scène selon un ensemble de points aléatoires répulsifs plutôt que sur une grille régulière. L'intérêt de cette approche est que l'aliasing résultant de cet échantillonnage aléatoire est incohérent et est perçu comme un bruit haute fréquence moins gênant pour la perception que l'aliasing structuré induit par l'échantillonnage sur une grille régulière [Cook, 1986, D. Heck and Deussen, 2013]. L'antialiasing stochastique repose essentiellement sur des considérations perceptuelles. Notre premier objectif concernant ce sujet sera d'apporter un formalisme mathématique qui justifie l'intérêt de cette approche en étudiant la distribution du champ aléatoire résultant de l'échantillonnage stochastique. Cette étude reposera sur les théorèmes de calcul des moments des processus ponctuels (formule de Campbell et de Slyvniak-Mecke) [Schneider and Weil, 2008]. Notre second objectif sera d'étudier les possibilités offertes par les PPD pour l'antialisaing stochastique, en proposant notamment des critères permettant de calculer le PPD optimal pour cette application. D'un point de vue pratique, si un tel résultat d'optimalité était établi, cela permettrait de proposer des critères d'évaluation objectifs des nombreuses approches heuristiques issues de la littérature en *computer graphics* où il est d'usage d'introduire de nombreuses approximations pour assurer un temps d'exécution minimal.

5.3 Synthèse de textures de bois

Une dernière perspective de recherche est une nouvelle collaboration avec A. Desolneux, J.-M. Morel et L. Raad. Ces membres du CMLA ont été contactés par une société industrielle afin de proposer des algorithmes de synthèse pour différent type de sols. Il s'est avéré qu'un type de textures était très important pour cet industriel, le bois en vue de faire des sols de type parquet. Or, les textures de bois présentent des structures géométriques à différentes échelles et présentant de fortes contraintes topologiques. Je les ai rejoint récemment pour travailler sur ce problème à l'interface entre la synthèse de texture par l'exemple et la synthèse procédurale [Liu et al., 2015]. Cette première expérience de recherche en partenariat

avec un industriel sera assurément très enrichissante, d'autant plus qu'elle attestera de l'"utilité" de la synthèse de textures.

Références

Je reproduis ici la liste de mes publications suivies des références aux articles cités dans le manuscrit. Toutes mes publications sont disponibles sur ma page web : `http://www.math-info.univ-paris5.fr/~bgalerne/publications.html`

Articles de journaux internationaux

[J1] B. Galerne, Y. Gousseau, and J.-M. Morel. Random phase textures : Theory and synthesis. *IEEE Trans. Image Process.*, 20(1) :257 – 267, 2011.

[J2] B. Galerne, Y. Gousseau, and J.-M. Morel. Micro-texture synthesis by phase randomization. *Image Processing On Line*, 2011.

[J3] B. Galerne. Computation of the perimeter of measurable sets via their covariogram. Applications to random sets. *Image Anal. Stereol.*, 30 :39–51, 2011.

[J4] B. Galerne and Y. Gousseau. The transparent dead leaves process. *Adv. Appl. Probab.*, 44(1) :1–20, 2012.

[J5] B. Galerne, A. Lagae, S. Lefebvre, and G. Drettakis. Gabor noise by example. *ACM Trans. Graph.*, 31(4) :73 :1–73 :9, jul 2012.

[J6] P.-E. Landes, B. Galerne, and T. Hurtut. A shape-aware model for discrete texture synthesis. *Computer Graphics Forum (Proc. EGSR)*, 32, 2013.

[J7] T. Briand, J. Vacher, B. Galerne, and J. Rabin. The heeger & bergen pyramid based texture synthesis algorithm. *Image Processing On Line*, 4 :276–299, 2014.

[J8] B. Galerne and R. Lachièze-Rey. Random measurable sets and covariogram realisability problems. *Adv. Appl. Probab.*, 47 :611–639, 2015.

[J9] B. Galerne. Random fields of bounded variation and computation of their variation intensity. *Adv. Appl. Probab.*, in press, 2016.

[J10] L. Raad and B. Galerne. Efros and Freeman image quilting algorithm for texture synthesis. *Image Processing On Line*, (accepted).

[J11] B. Galerne, A. Leclaire, and L. Moisan. Texton noise. *Computer Graphics Forum*, (accepted).

Actes de Conférences avec comité de lecture

[C12] B. Galerne. Variation totale moyenne de modèles stochastiques de texture. In *GRETSI*, Bordeaux, 2011.

[C13] B. Galerne, A. Leclaire, and L. Moisan. A texton for fast and flexible Gaussian texture synthesis. In *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, pages 1686–1690, 2014.

[C14] B. Galerne, A. Leclaire, and L. Moisan. Microtexture inpainting through Gaussian conditional simulation. In *Proceedings of IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP 2016)*, 2016.

Thèse de doctorat

[T15] B. Galerne. *Modèles d'image aléatoires et synthèse de texture*. PhD thesis, Ecole Normale Supérieur de Cachan, Déc. 2010.

Articles soumis

[S16] A. Newson, B. Galerne, and J. Delon. Stochastic modeling and resolution-free rendering of film grain. submitted to Computer Graphics Forum, 2016.

Bibliographie

[Adler, 1981] Adler, R. (1981). *The Geometry of Random Fields*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, Ltd., Chichester.

[Agueh and Carlier, 2011] Agueh, M. and Carlier, G. (2011). Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2) :904–924.

[Aguerrebere et al., 2013] Aguerrebere, C., Gousseau, Y., and Tartavel, G. (2013). Exemplar-based texture synthesis : the Efros-Leung algorithm. *Image Processing On Line*, 3 :223–241.

[Alvarez et al., 2015] Alvarez, L., Gousseau, Y., Morel, J.-M., and Salgado, A. (2015). Exploring the space of abstract textures by principles and random sampling. *Journal of Mathematical Imaging and Vision*, 53(3) :332–345.

[Ambrosio et al., 2000] Ambrosio, L., Fusco, N., and Pallara, D. (2000). *Functions of Bounded Variation and Free Discontinuity Problems*. Oxford mathematical monographs. The Clarendon Press, Oxford University Press, New York.

[Andreu-Vaillo et al., 2004] Andreu-Vaillo, F., Caselles, V., and Mazon, J. M. (2004). *Parabolic Quasilinear Equations Minimizing Linear Growth Functionals*, volume 223 of *Progress in mathematics*. Birkhäusser.

[Arias et al., 2011] Arias, P., Facciolo, G., Caselles, V., and Sapiro, G. (2011). A variational framework for exemplar-based image inpainting. *International Journal of Computer Vision*, 93(3) :319–347.

[Aujol et al., 2010] Aujol, J., Ladjal, S., and Masnou, S. (2010). Exemplar-based inpainting from a variational point of view. *SIAM Journal on Mathematical Analysis*, 42(3) :1246–1285.

[Aujol et al., 2005] Aujol, J.-F., Aubert, G., Blanc-Féraud, L., and Chambolle, A. (2005). Image decomposition into a bounded variation component and an oscillating component. *J. Math. Imaging Vis.*, 22 :71 – 88. 10.1007/s10851-005-4783-8.

[Aujol and Chambolle, 2005] Aujol, J.-F. and Chambolle, A. (2005). Dual norms and image decomposition models. *Int. J. Comput. Vis.*, 63 :85–104. 10.1007/s11263-005-4948-3.

[Aurenhammer et al., 1998] Aurenhammer, F., Hoffmann, F., and Aronov, B. (1998). Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20(1) :61–76.

[Baddeley, 2007] Baddeley, A. (2007). Spatial point processes and their applications. In Weil, W., editor, *Stochastic Geometry, Lectures given at the C.I.M.E. Summer School held in Martina Franca, Italy, September 13-18, 2004*, volume 1892 of *Lecture Notes in Mathematics*, pages 1–75. Springer.

[Baddeley and Turner, 2000] Baddeley, A. and Turner, R. (2000). Practical maximum pseudolikelihood for spatial point patterns. *Australian & New Zealand Journal of Statistics*, 42(3) :283–322.

[Barla et al., 2006] Barla, P., Breslav, S., Thollot, J., Sillion, F., and Markosian, L. (2006). Stroke pattern analysis and synthesis. In *Computer Graphics Forum (Proc. of Eurographics 2006)*, volume 25.

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

[Bertalmio et al., 2000] Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. (2000). Image inpainting. In *Proc. of SIGGRAPH*, pages 417–424.

[Bianchi et al., 2011] Bianchi, G., Gardner, R. J., and Kiderlen, M. (2011). Phase retrieval for characteristic functions of convex bodies and reconstruction from covariograms. *J. Amer. Math. Soc.*, 24(2) :293–343.

[Biermé and Desolneux, 2012a] Biermé, H. and Desolneux, A. (2012a). Crossings of smooth shot noise processes. *Ann. Appl. Probab.*, 22(6) :2240–2281.

[Biermé and Desolneux, 2012b] Biermé, H. and Desolneux, A. (2012b). A Fourier approach for the level crossings of shot noise processes with jumps. *J. Appl. Probab.*, 49(1) :100–113.

[Biermé and Desolneux, 2016] Biermé, H. and Desolneux, A. (2016). On the perimeter of excursion sets of shot noise random fields. *Ann. Probab.*, 44(1) :521–543.

[Biermé et al., 2007] Biermé, H., Meerschaert, M. M., and Scheffler, H.-P. (2007). Operator scaling stable random fields. *Stochastic Processes and their Applications*, 117(3) :312 – 332.

[Biermé et al., 2015] Biermé, H., Moisan, L., and Richard, F. (2015). A turning-band method for the simulation of anisotropic fractional Brownian fields. *Journal of Computational and Graphical Statistics*, 24(3) :885–904.

[Biscio and Lavancier, 2016] Biscio, C. and Lavancier, F. (2016). Quantifying repulsiveness of determinantal point processes. *Bernoulli*, (to appear).

[Blanchet and Moisan, 2012] Blanchet, G. and Moisan, L. (2012). An explicit sharpness index related to global phase coherence. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 1065–1068. IEEE.

[Blanchet et al., 2008] Blanchet, G., Moisan, L., and Rougé, B. (2008). Measuring the global phase coherence of an image. In *ICIP 2008*, pages 1176 – 1179.

[Bordenave et al., 2006] Bordenave, C., Gousseau, Y., and Roueff, F. (2006). The dead leaves model : a general tessellation modeling occlusion. *Adv. Appl. Probab.*, 38(1) :31–46.

[Buades et al., 2010] Buades, A., Le, T. M., Morel, J.-M., and Vese, L. A. (2010). Fast cartoon + texture image filters. *IEEE Trans. Image Process.*, 19(8) :1978 –1986.

[Cardoso and Souloumiac, 1996] Cardoso, J.-F. and Souloumiac, A. (1996). Jacobi angles for simultaneous diagonalization. *SIAM J. Mat. Anal. Appl.*, 17(1) :161–164.

[Chainais, 2007] Chainais, P. (2007). Infinitely divisible cascades to model the statistics of natural images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12) :2105–2119.

[Chambolle, 2004] Chambolle, A. (2004). An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20(1-2) :89–97.

[Chan and Shen, 2002] Chan, T. and Shen, J. (2002). Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3) :1019–1043.

[Chan and Esedoglu, 2005] Chan, T. F. and Esedoglu, S. (2005). Aspects of total variation regularized l^1 function approximation. *SIAM J. Appl. Math.*, 65(5) :1817–1837.

[Chan and Wong, 1998] Chan, T. F. and Wong, C. K. (1998). Total variation blind deconvolution. *IEEE Trans. Image Process.*, 7(3) :370 – 375.

[Chellappa and Kashyap, 1985] Chellappa, R. and Kashyap, R. L. (1985). Texture synthesis using 2-D noncausal autoregressive models. In *Acoustics, Speech and Signal Processing, IEEE Transactions on*, volume 33, pages 194–203. IEEE.

[Chiu et al., 2013] Chiu, S. N., Stoyan, D., Kendall, W. S., and Mecke, J. (2013). *Stochastic geometry and its applications*. Wiley series in probability and statistics. John Wiley & Sons, third edition.

[Cook, 1986] Cook, R. (1986). Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1) :51–72.

[Cook and DeRose, 2005] Cook, R. L. and DeRose, T. (2005). Wavelet noise. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 803–811, New York, NY, USA. ACM.

[Cowan and Tsang, 1994] Cowan, R. and Tsang, A. (1994). The falling-leaves mosaic and its equilibrium properties. *Adv. Appl. Probab.*, 26 :54–62.

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

[Cuturi, 2013] Cuturi, M. (2013). Sinkhorn distances : Lightspeed computation of optimal transport. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 26*, pages 2292–2300. Curran Associates, Inc.

[D. Heck and Deussen, 2013] D. Heck, T. S. and Deussen, O. (2013). Blue noise sampling with controlled aliasing. *ACM Trans. on Graphics*.

[Dereudre et al., 2012] Dereudre, D., Drouilhet, R., and Georgii, H.-O. (2012). Existence of Gibbsian point processes with geometry-dependent interactions. *Probability Theory and Related Fields*, 153(3) :643–670.

[Desolneux et al., 2012] Desolneux, A., Moisan, L., and Ronsin, S. (2012). A compact representation of random phase and Gaussian textures. In *ICASSP'12*, pages 1381–1384.

[Desolneux et al., 2016] Desolneux, A., Moisan, L., and Ronsin, S. (2016). A texton for random phase and Gaussian textures. in preparation.

[Dietrich and Newsam, 1997] Dietrich, C. R. and Newsam, G. N. (1997). Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix. *SIAM Journal on Scientific Computing*, 18(4) :1088–1107.

[Digne et al., 2011] Digne, J., Audfray, N., Lartigue, C., Mehdi-Souzani, C., and Morel, J.-M. (2011). Farman institute 3d point sets - high precision 3d data sets. *Image Processing On Line*, 1.

[Doob, 1953] Doob, J. L. (1953). *Stochastic Processes*. John Wiley & Sons.

[Duval et al., 2009] Duval, V., Aujol, J.-F., and Gousseau, Y. (2009). The tvl1 model : A geometric point of view. *Multiscale Model. Simul.*, 8(1) :154–189.

[DxO, 2016] DxO (2016). Dxo film pack 5.

[Efros and Freeman, 2001] Efros, A. A. and Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 341–346, New York, NY, USA. ACM.

[Efros and Leung, 1999] Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *ICIP 1999*, pages 1033 – 1038.

[Evans and Gariepy, 1992] Evans, L. C. and Gariepy, R. F. (1992). *Measure theory and fine properties of functions*. Studies in advanced mathematics. CRC Press.

[Geyer and Møller, 1994] Geyer, C. and Møller, J. (1994). Simulation procedures and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics*, pages 359–373.

[Gilet et al., 2010] Gilet, G., Dischler, J.-M., and Soler, L. (2010). Procedural descriptions of anisotropic noisy textures by example. In *Eurographics (Short)*.

[Gilet et al., 2014] Gilet, G., Sauvage, B., Vanhoey, K., Dischler, J.-M., and Ghazanfarpour, D. (2014). Local random-phase noise for procedural texturing. *ACM Trans. Graph.*, 33(6) :195 :1–195 :11.

[G'MIC, 2016] G'MIC (2016). Greyc's magic for image computing.

[Goldberg et al., 2008] Goldberg, A., Zwicker, M., and Durand, F. (2008). Anisotropic noise. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 54 :1–54 :8, New York, NY, USA. ACM.

[Grubba Software, 2015] Grubba Software (2015). Truegrain.

[Guichard and Malgouyres, 1998] Guichard, F. and Malgouyres, F. (1998). Total variation based interpolation. In *Eusipco'98*, pages 1741–1744. Elsevier North-Holland, Inc.

[Hayes, 1982] Hayes, M. (1982). The Reconstruction of a Multidimensional Sequence from the Phase or Magnitude of its Fourier Transform. *IEEE Trans. on Acoustics, Speech, Sign. Proc.*, 30 :2 :140–154.

[Heeger and Bergen, 1995] Heeger, D. J. and Bergen, J. R. (1995). Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95*, pages 229–238, New York, NY, USA. ACM.

[Hough et al., 2009] Hough, J. B., Krishnapur, M., Peres, Y., and Virág, B. (2009). *Zeros of Gaussian Analytic Functions and Determinantal Point Processes*, volume 51 of *University Lecture Series*. Providence, RI.

[Hug et al., 2004] Hug, D., Last, G., and Weil, W. (2004). A local Steiner-type formula for general closed sets and applications. *Math. Z.*, 246(1-2) :237–272.

[Hurtut et al., 2009] Hurtut, T., Landes, P.-E., Thollot, J., Gousseau, Y., Drouillhet, R., and Coeurjolly, J.-F. (2009). Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '09, pages 51–60, New York, NY, USA. ACM.

[Ibragimov, 1995] Ibragimov, I. A. (1995). Remarks on variations of random fields. *J. Math. Sci.*, 75(5) :1931–1934.

[Ijiri et al., 2008] Ijiri, T., Mêch, R., Igarashi, T., and Miller, G. (2008). An example-based procedural system for element arrangement. *Computer Graphics Forum (Proc. of Eurographics 2008)*, 27(2) :429–436.

[Kang, 2013] Kang, B. (2013). Fast determinantal point process sampling with application to clustering. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 26*, pages 2319–2327.

[Kendall and Thönnes, 1999] Kendall, W. S. and Thönnes, E. (1999). Perfect simulation in stochastic geometry. *Pattern Recognition*, 32(9) :1569–1586.

[Kim et al., 2007] Kim, S.-J., Koh, K., Lustig, M., Boyd, S., and Gorinevsky, D. (2007). An interior-point method for large-scale l_1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4).

[Kitagawa et al., 2016] Kitagawa, J., Mérigot, Q., and Thibert, B. (2016). A Newton algorithm for semi-discrete optimal transport. ArXiv e-prints.

[Kopf et al., 2007] Kopf, J., Fu, C.-W., Cohen-Or, D., Deussen, O., Lischinski, D., and Wong, T.-T. (2007). Solid texture synthesis from 2D exemplars. *ACM Trans. Graph. (Proc. SIGGRAPH 2007)*, 26(3).

[Kronmal and A. V. Peterson, 1979] Kronmal, R. A. and A. V. Peterson, J. (1979). On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4) :214–218.

[Kulesza and Taskar, 2012] Kulesza, A. and Taskar, B. (2012). Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3) :123–286.

[Kunisch and Pock, 2013] Kunisch, K. and Pock, T. (2013). A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences*, 6(2) :938–983.

[Kwatra et al., 2005] Kwatra, V., Essa, I., Bobick, A., and Kwatra, N. (2005). Texture optimization for example-based synthesis. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 795–802. ACM.

[Lachièze-Rey and Molchanov, 2015] Lachièze-Rey, R. and Molchanov, I. (2015). Regularity conditions in the realisability problem with applications to point processes and random closed sets. *Ann. Appl. Probab.*, 25(1) :116–149.

[Lagae and Drettakis, 2011] Lagae, A. and Drettakis, G. (2011). Filtering solid Gabor noise. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, 30(4) :51 :1–51 :6.

[Lagae et al., 2010a] Lagae, A., Lefebvre, S., Cook, R., DeRose, T., Drettakis, G., Ebert, D., Lewis, J., Perlin, K., and Zwicker, M. (2010a). A survey of procedural noise functions. *Computer Graphics Forum*, 29(8) :2579–2600.

[Lagae et al., 2009] Lagae, A., Lefebvre, S., Drettakis, G., and Dutré, P. (2009). Procedural noise using sparse gabor convolution. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 54 :1–54 :10, New York, NY, USA. ACM.

[Lagae et al., 2010b] Lagae, A., Vangorp, P., Lenaerts, T., and Dutré, P. (2010b). Procedural isotropic stochastic textures by example. *Computers & Graphics (Special issue on Procedural Methods in Computer Graphics)*.

[Lantuéjoul, 2002] Lantuéjoul, C. (2002). *Geostatistical Simulation : Models and Algorithms*. Springer-Verlag, Berlin.

[Lavancier et al., 2015] Lavancier, F., Møller, J., and Rubak, E. (2015). Determinantal point process models and statistical inference. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 77(4) :853–877.

[Leclaire, 2015] Leclaire, A. (2015). *Random Phase Fields and Gaussian Fields for Image Sharpness Assessment and Fast Texture Synthesis*. PhD thesis, Université Paris Descartes.

[Leclaire and Moisan, 2015] Leclaire, A. and Moisan, L. (2015). No-reference image quality assessment and blind deblurring with sharpness metrics exploiting Fourier phase information. *Journal of Mathematical Imaging and Vision*, 52(1) :145–172.

[Lefebvre and Hoppe, 2005] Lefebvre, S. and Hoppe, H. (2005). Parallel controllable texture synthesis. In *SIGGRAPH '05*, pages 777–786. ACM.

[Lévy, 2015] Lévy, B. (2015). A numerical algorithm for L2 semi-discrete optimal transport in 3D. *ESAIM : M2AN*, 49(6) :1693–1715.

[Lewis, 1984] Lewis, J.-P. (1984). Texture synthesis for digital painting. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 245–252, New York, NY, USA. ACM.

[Liu et al., 2015] Liu, A. J., Marschner, S. R., and Dye, V. E. (2015). Procedural wood textures.

[Ma et al., 2011] Ma, C., Wei, L.-Y., and Tong, X. (2011). Discrete element textures. *ACM Trans. Graph. (Proc. SIGGRAPH 20110)*, 30(4) :62 :1–62 :10.

[Mairal et al., 2011] Mairal, J., Jenatton, R., Obozinski, G., and Bach, F. (2011). Convex and network flow optimization for structured sparsity. *J. Machine Learning Research*, 12 :2681–2720.

[Masnou and Morel, 1998] Masnou, S. and Morel, J.-M. (1998). Level lines based disocclusion. In *Proceedings of ICIP*, volume 3, pages 259–263.

[Matheron, 1967] Matheron, G. (1967). *Éléments pour une théorie des mileux poreux*. Masson.

[Matheron, 1968] Matheron, G. (1968). Schéma booléen séquentiel de partition aléatoire. Technical Report 89, CMM.

[Matheron, 1975] Matheron, G. (1975). *Random Sets and Integral Geometry*. Wiley series in probability and mathematical statistics. John Wiley & Sons, New York-London-Sydney.

[Matheron, 1986] Matheron, G. (1986). Le covariogramme géométrique des compacts convexes de $\mathbb{R}^2$. Technical Report N/2/86/G, Centre de Géostatistique, Ecole des mines de Paris.

[Mees, 1942] Mees, C. E. K. (1942). *The Theory of the Photographic Process*. The MacMillan Company, New York, USA.

[Mérigot, 2011] Mérigot, Q. (2011). A multiscale approach to optimal transport. *Computer Graphics Forum*, 30(5) :1583–1592.

[Meyer, 2001] Meyer, Y. (2001). *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations. The Fifteenth Dean Jacqueline B. Lewis Memorial Lectures*, volume 22 of *Universtity lecture series*. American Mathematical Society, Providence, RI.

[Moisan, 2011] Moisan, L. (2011). Periodic plus smooth image decomposition. *J. Math. Imag. Vis.*, 39 :161–179.

[Molchanov, 2005] Molchanov, I. (2005). *Theory of Random Sets*. Probability and Its Applications. Springer-Verlag London, Ltd., London.

[Nikolova, 2004] Nikolova, M. (2004). A variational approach to remove outliers and impulse noise. *J. Math. Imag. Vis.*, 20 :99–120.

[Papoulis, 1971] Papoulis, A. (1971). High density shot noise and Gaussianity. *J. Appl. Probab.*, 8(1) :118–127.

[Perlin, 1985] Perlin, K. (1985). An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 287–296, New York, NY, USA. ACM.

[Portilla and Simoncelli, 2000] Portilla, J. and Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comp. Vis.*, 40(1) :49–71.

[Raad et al., 2014] Raad, L., Desolneux, A., and Morel, J.-M. (2014). Locally Gaussian exemplar based texture synthesis. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4667–4671.

[Raad et al., 2015] Raad, L., Desolneux, A., and Morel, J.-M. (2015). Conditional and multiscale locally gaussian models for texture synthesis. preprint.

[Rabin et al., 2012] Rabin, J., Peyré, G., Delon, J., and Bernot, M. (2012). Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 435–446. Springer Berlin / Heidelberg.

[Rataj, 2015] Rataj, J. (2015). Random sets of finite perimeter. *Mathematische Nachrichten*, 288(8-9) :1047–1056.

[Reed, 2013] Reed, N. (2013). Quick and easy GPU random numbers in D3D11. Blogpost, `http://www.reedbeta.com/blog/2013/01/12/quick-and-easy-gpu-random-numbers-in-d3d11/`.

[Reyes et al., 2016] Reyes, J. D. L., Schönlieb, C.-B., and Valkonen, T. (2016). The structure of optimal parameters for image restoration problems. *Journal of Mathematical Analysis and Applications*, 434(1) :464 – 500.

[Ronsin et al., 2016] Ronsin, S., Biermé, H., and Moisan, L. (2016). The Billard theorem for multiple random Fourier series. *Journal of Fourier Analysis and Applications*. to appear.

[Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D : Nonlinear Phenomena*, 60(1-4) :259 – 268.

[Scheuerer, 2010] Scheuerer, M. (2010). Regularity of the sample paths of a general second order random field. *Stoch. Process. Appl.*, 120(10) :1879 – 1897.

[Schneider and Weil, 2008] Schneider, R. and Weil, W. (2008). *Stochastic and Integral Geometry*. Probability and Its Applications. Springer-Verlag, Berlin.

[Serra, 1988] Serra, J., editor (1988). *Image Analysis and Mathematical Morphology, volume 2 : theoretical advances*. Academic press, London.

[Soshnikov, 2000] Soshnikov, A. (2000). Determinantal random point fields. *Russian Mathematical Surveys*, (55) :923–975.

[Tartavel et al., 2014] Tartavel, G., Gousseau, Y., and Peyré, G. (2014). Variational texture synthesis with sparsity and spectrum constraints. *Journal of Mathematical Imaging and Vision*, 52(1) :124–144.

[van Wijk, 1991] van Wijk, J. J. (1991). Spot noise texture synthesis for data visualization. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 309–318, New York, NY, USA. ACM.

[Vese and Osher, 2003] Vese, L. A. and Osher, S. J. (2003). Modeling textures with total variation minimization and oscillating patterns in image processing. *Journal of Scientific Computing*, 19 :553–572.

[Villa, 2010] Villa, E. (2010). Mean densities and spherical contact distribution function of inhomogeneous Boolean models. *Stoch. Anal. Appl.*, 28 :480–504.

[Walker, 1977] Walker, A. J. (1977). An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, 3 :253–256.

[Wei and Levoy, 2000] Wei, L. Y. and Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00*, pages 479–488. ACM Press/Addison-Wesley Publishing Co.

[Wood and Chan, 1997] Wood, A. T. A. and Chan, G. (1997). Simulation of stationary Gaussian processes in $[0, 1]^d$. *Journal of Computational and Graphical Statistics*, 3(4) :409–432.

[Worley, 1996] Worley, S. (1996). A cellular texture basis function. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 291–294, New York, NY, USA. ACM.

[Xia et al., 2014] Xia, G.-S., Ferradans, S., Peyré, G., and Aujol, J.-F. (2014). Synthesizing and mixing stationary Gaussian texture models. *SIAM J. on Imaging Science*, 8(1) :476–508.