UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE Mathématiques, Sciences de l'Information et de l'Ingénieur

Unité de recherche : **ICube – UMR 7357**

THÈSE présentée par :

Charline Grenier

soutenue le : **13 Mars 2024**

Pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : **Informatique**

Génération procédurale et rendu en temps réel de motifs structurés

THÈSE dirigée par :

M. Sauvage Basile	Professeur des universités, Université de Strasbourg
M. Dischler Jean-Michel	Professeur des universités, Université de Strasbourg

RAPPORTEURS :

M. Gilet Guillaume	Professeur des universités, Université de Sherbrooke
M. Lefebvre Sylvain	Directeur de recherche, INRIA Nancy Grand Est

---

AUTRES MEMBRES DU JURY :

Mme Digne Julie	Directrice de recherche, Université de Lyon
M. Ribardière Mickaël	Maitre de conférence, Université de Poitiers

« The only way to achieve the impossible is to believe that it is possible. »
- Lewis Carol -

Remerciements

Avant de plonger dans le monde merveilleux des mathématiques appliquées à la synthèse d'images et au rendu de mondes virtuels, j'aimerais consacrer quelques lignes pour remercier les personnes qui m'ont soutenu au cours de cette thèse.

Je veux d'abord à remercier les membres de mon jury d'avoir accepté de rapporter et d'examiner mes travaux de recherche. Merci pour votre intérêt et votre temps.

Je tiens ensuite à remercier mon directeur de thèse, Basile Sauvage, et mon co-directeur, Jean-Michel Dischler, pour le temps passé à me former, leur soutient, leur confiance et les longues discussions sur des questionnements scientifiques parfois étranges, mais toujours intéressantes.
Je souhaite également remercier toute l'équipe IGG du laboratoire ICube pour l'ambiance de travail, les barbecues et les pauses "petits pains". Merci notamment à Dominique Bechmann pour sa présence féminine à la tête de l'équipe, à Sylvain Thery pour son soutient technique, à l'équipe administrative et à l'ensemble des doctorants et post-doctorants du laboratoire (entre autres Paul, Nicolas, Xavier, Cyril, Pascal, Quentin, Pauline, Romain, etc.).

Je veux enfin remercier ma famille pour leur présence et, en particulier, Stiven pour m'avoir soutenue, aidée et supportée tout au long de ma thèse.

Université de Strasbourg

Charline Grenier

iCUBE

Génération procédurale et rendu en temps réel de motifs structurés

Résumé

Dans un monde virtuel, les objets représentés sont décrits autant par leur géométrie que par le comportement lumineux de leur surface. Ce dernier est un élément important pour le réalisme de la scène créée. Pour simuler le comportement lumineux des surfaces, une méthode couramment utilisée consiste à recouvrir la géométrie des objets avec des textures.

Dans cette thèse, nous nous intéressons aux textures stochastiques structurées. Et plus particulièrement à leur génération et à leur rendu multi-échelle en temps réel. Ces textures se caractérisent par un caractère stochastique qui leur donne une apparence plus organique et naturelle, et de brusques variations de couleur ou de contraste, faisant apparaître des motifs distincts que nous appelons structures.

Nous proposons une nouvelle méthode de génération procédurale de motifs structurés. Celle-ci se base sur la composition d'un bruit procédural vectoriel par une fonction de transfert multi-variée. Cette méthode permet de tirer parti de la séparation entre l'information de structure, contenue dans la fonction de transfert, et le côté stochastique, apporté par le bruit.

Mots clés : informatique graphique, fonction de transfert, génération de textures, génération procédurale, motifs structurés, temps réel, filtrage, rendu.

Abstract

In a virtual world, objects represented are described as much by their geometry as by the luminous behaviour of their surface. The latter is an important element in the realism of the scene created. To simulate the luminous behaviour of surfaces, a commonly used method consists of covering the geometry of objects with textures.

In this thesis, we focus on structured stochastic textures. More specifically, we are interested in their generation and rendering at different level of detail, in real time. These textures are characterised by a stochastic nature which gives them a more organic and natural appearance, and by abrupt variations in colour or contrast giving rise to distinct patterns which we call structures.

We propose a new method for procedural generation of structured patterns. It is based on the composition of a procedural vector noise by a multivariate transfer function. This method takes advantage of the separation between the structural information contained in the transfer function and the stochastic information provided by the noise.

Keywords : computer graphics, transfer function, texture synthesis, procedural generation, structured patterns, real-time, filtering, rendering.

Table des matières

Chapitre 1

Introduction

L'informatique graphique est une branche de l'informatique qui s'intéresse, entre autres, à la synthèse d'images et au rendu de mondes virtuels. Ce domaine a connu une évolution très importante au cours des dernières décennies, l'augmentation rapide de la puissance de calcul des ordinateurs ayant permis des avancées significatives dans la recherche. Dû à la démocratisation de l'accès à cette importante puissance de calcul, les mondes virtuels font aujourd'hui partie de notre quotidien. Ainsi, de nombreux milieux industriels, tel que ceux du divertissement (par exemple les films ou les jeux vidéo), font un usage massif des effets numériques à leur disposition.
Dans ce chapitre, nous présentons le contexte global de la thèse, puis ses objectifs ainsi que l'organisation du reste du manuscrit.

Sommaire

1.1 Introduction générale

Les mondes virtuels sont constitués d'objets 3D le plus souvent décrits par leur maillage. Ces objets sont généralement recouverts par des textures. Ces dernières décrivent le comportement lumineux des surfaces (figure 1.1).

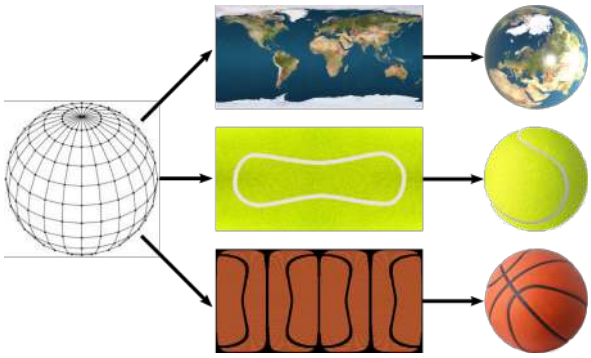


Figure 1.1 – *Une sphère, décrite par son maillage (à gauche), peut représenter différents objets 3D (à droite) selon la texture qui lui est appliquée (au centre).*

De nombreuses informations peuvent être contenues dans les textures d'un objet (figure 1.2). Elles permettent entre autres de décrire :

- des perturbations de hauteur ou de normale (pour enrichir la géométrie),
- la couleur de l'objet (appelée "albédo"),
- l'aspect brillant ou mat (différenciant par exemple les métaux des caoutchoucs),
- la présence d'irrégularités sur sa surface (comme des griffures ou des paillettes scintillantes)...

Figure 1.2 – *L'apparence de l'objet (à gauche), est enrichie par l'application de plusieurs textures (perturbation des normales, couleurs, aspect brillant). Image issue de [Sau18].*

Les créateurs de mondes virtuels cherchent de plus en plus à se rapprocher de l'aspect visuel de mondes réalistes. Cela passe par la création de mondes virtuels de plus en plus grands et détaillés. Cette surenchère au réalisme apporte des difficultés sur plusieurs aspects. Notamment en termes de temps de calcul et d'occupation de la mémoire. En effet, plus une scène se veut réaliste, plus elle doit comporter d'éléments et de détails et, en particulier, de détails très fins lorsque l'observateur s'en approche. Cependant, cette quantité très importante de détails représente aussi une quantité d'informations trop importante pour être intégralement conservée en mémoire. Il est donc nécessaire de ne traiter que les éléments visibles à un instant donné. Cela signifie également que, si un élément vient à entrer dans le champ de vision de l'utilisateur à l'instant suivant, toutes les informations le concernant doivent être traitées dans un temps le plus court possible. Ainsi, un compromis doit

être fait entre la précision du monde à toutes les échelles et le temps nécessaire pour le rendre, en prenant en compte la distance à laquelle se trouve la portion de scène rendue, de façon à ne pas passer beaucoup de temps à calculer des détails trop loin pour être visibles à l'écran.

Une autre difficulté apportée par l'augmentation du réalisme des mondes virtuel est le temps nécessaire à la création des éléments qui le composent. En effet, la représentation d'éléments de plus en plus détaillés signifie également que les textures qui simulent leur comportement lumineux doivent être extrêmement détaillées aussi. Ces éléments et leurs textures ne peuvent pas raisonnablement être créés entièrement à la main par des artistes (par exemple les micro-détail à la surface d'une roche ou les grains de sable d'un désert). Pour représenter ces éléments dans un monde virtuel sans les créer entièrement à la main, plusieurs méthodes peuvent être utilisées. Par exemple, certaines méthodes s'appuient sur la récupération d'informations du monde réel (au moyen de photographies par exemple) et utilisent des algorithmes de reconstruction pour recréer l'objet dans un monde virtuel (figure 1.3). Cependant, elles peuvent être coûteuses en mémoire et en temps. En effet, elles exigent une grande quantité de données d'entrée et peuvent nécessiter l'entraînement d'un réseau de neurones dans certain cas.

Figure 1.3 – *Reconstitution d'un objet en 3D à partir d'une série de photographies (à gauche) en utilisant un réseau de neurones. Image issue de [MST+21].*

Pour la création de textures, indépendamment de la forme 3D, il est, par exemple, possible d'utiliser le pavage répétitif d'un petit nombre de tuiles créées en amont. Cette technique utilise peu de mémoire dans le cas où peu de tuiles différentes sont utilisées, mais des répétitions vont alors apparaître dans le résultat (figure 1.4). Il est possible de définir plus de tuiles pour augmenter la variété dans le résultat, mais le coût en mémoire devient alors plus important.

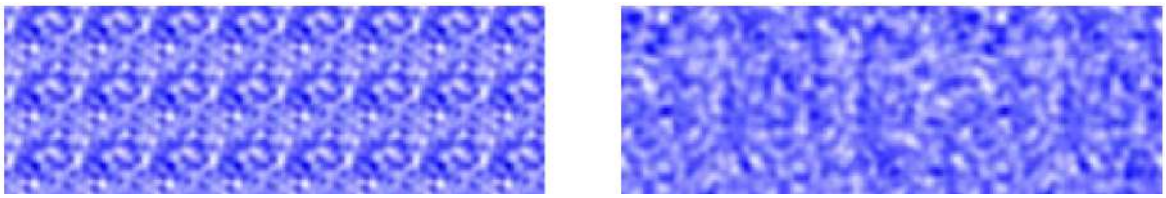


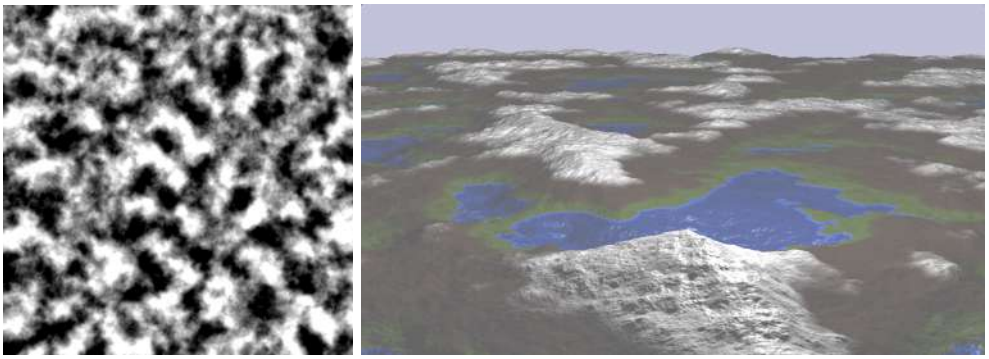
Figure 1.4 – *Comparaison entre le pavage répétitif d'une seule tuile (à gauche) et le pavage de 16 tuiles légèrement différentes. Image issue de [Sta97].*

D'autres méthodes proposent une alternative au stockage d'images en mémoire et se basent sur l'exécution d'une procédure. Ces méthodes peuvent être regroupées sous la qualification "d'approches procédurales". Dans leur livre *Texturing and Modeling : A Procedural Approach* [EMP+02] Ebert et collègues proposent une définition de l'approche procédurale. Selon leurs termes, « dans le cadre d'une approche procédurale, plutôt que de spécifier et stocker explicitement tous les détails complexes d'une scène ou d'une séquence, nous les abstrayons dans une fonction ou un algorithme (c'est-à-dire une procédure) et nous évaluons cette procédure au moment et à la localisation à laquelle c'est nécessaire. » [1] Dans ce manuscrit, nous nous intéressons à l'utilisation de l'approche procédurale pour la génération de textures. Le résultat est alors qualifié de "texture procédurale". Ces méthodes procédurales présentent de nombreux avantages pour la création de mondes réalistes. Par exemple, pour

1. Traduit de l'anglais.

le cas de la création de textures, chaque élément de texture (ou *texel*, contraction de l'anglais *texture element*) est calculé indépendamment de tous les autres. Cela permet une parallélisation à l'extrême de la génération, puisque tous les éléments peuvent être calculés en même temps, et donc un gain conséquent en temps de calcul. De plus, la texture résultante est de taille non bornée et peut donc couvrir des surfaces arbitrairement grandes. Cela représente un gain de place en mémoire, en effet, la place nécessaire pour stocker la chaîne de caractères correspondant à la procédure est très inférieure à celle nécessaire pour stocker une image de grande taille. Nous développons plus en détails les caractéristiques de la génération procédurale dans l'état de l'art de ce manuscrit (section 2.1).

Pour imiter des éléments plus organiques, la génération procédurale s'appuie sur des processus simulant l'aléatoire. Ces processus, appelés bruits en informatique graphique, sont utilisés pour ajouter une part d'aléatoire dans le résultat de la génération. Nous pouvons définir un bruit comme un signal qui ne se répète pas à l'identique lors de la réitération de l'expérience qui le produit. En particulier, les bruits dits gaussiens permettent de simuler des processus gaussiens, c'est-à-dire dont toutes les statistiques de répartition suivent des lois normales. L'étude des bruits remonte à avant l'informatique graphique avec, entre autres, la présentation des bruits fractals par Benoit Mandelbrot en 1968 [MVN68]. Ceux-ci sont encore utilisés aujourd'hui dans certains domaines d'informatique graphique comme la modélisation de terrains (figure 1.5).



(a) *Bruit fractal* **(b)** *Terrain*

Figure 1.5 – *Utilisation d'un bruit fractal pour générer un terrain procéduralement. Le champ scalaire 1.5a est utilisé comme information de hauteur pour créer le terrain 1.5b.*

Ces bruits procéduraux ont connu une poussée d'utilisation après la présentation du bruit de Perlin [Per85] par Ken Perlin en 1985. La présentation de ce bruit vient de la nécessité d'ajouter des imperfections aux images créées numériquement pour le film *Tron*, et jugées trop lisses par rapport aux images en prise de vue réelle (figure 1.6).

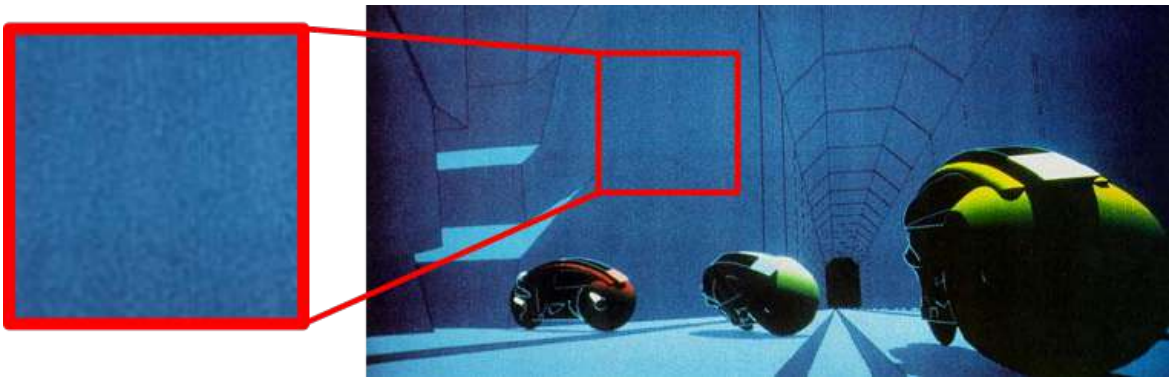


Figure 1.6 – *Les lumicycles,* Tron *(©*Disney*). Le zoom sur une portion du mur (à gauche) permet d'observer le bruit ajouté sur l'image numérique dans le but d'imiter le grain des pellicules.*

Dans cette thèse, nous nous intéressons au cas des textures générées procéduralement. Une très grande variété d'apparences peuvent être observées dans le monde réel. Une classification de ces apparences a été proposée par Lin et collègues [LHW+06] en se reposant sur des caractéristiques de contenu (figure 1.7). Nous pouvons ainsi distinguer les textures dites régulières, présentant des répétitions périodiques d'un motif élémentaire, sans modification d'intensité, de couleur ou de forme. Celles-ci se différencient des textures irrégulières, qui sont complètement aléatoires et où aucune répétition de motifs ne se distingue.

Figure 1.7 – *Classification de textures selon la caractéristique régulière ou irrégulière de leur contenu. Les textures intermédiaires, dites quasi-régulières, présentent à la fois des éléments présentant des similitudes, mais aussi une part d'aléatoire. Image issue de [LHW+06].*

Les textures intermédiaires, dites quasi-régulières, présentent à la fois des motifs se répétant, mais aussi une part d'aléatoire, soit dans la disposition des motifs, soit dans les caractéristiques internes aux motifs (comme des variations de forme ou de couleur par exemple).

Nous pouvons également caractériser les textures selon la présence ou non de motifs dits structurés (figure 1.8). Nous définissons ici la structure comme la présence de discontinuités ou de variations franches dans la texture (comme des variations d'intensité ou de couleurs).

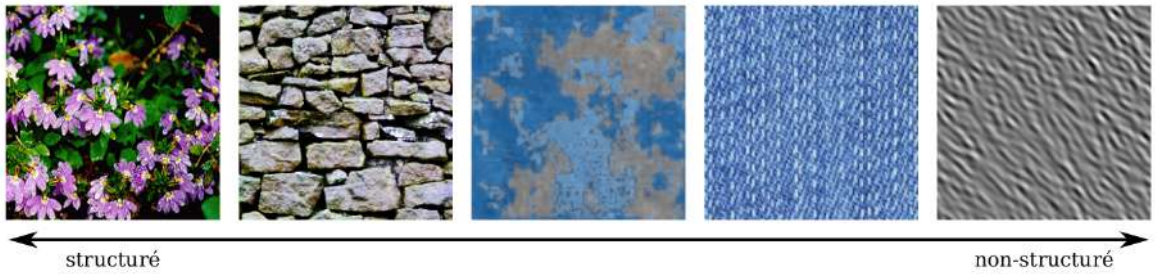


Figure 1.8 – *Classification de textures selon la présence ou non de structure dans leur contenu. Les fortes variations de couleurs ou de contraste entre les fleurs ou les pierres mettent en avant la structure dans les textures correspondantes. Contrairement aux variations moins franches dans les mailles du tissu. Image issue de [Sau18].*

Ces deux éléments de caractérisation n'entrent pas en conflit et il est possible de considérer un plan regroupant les deux axes précédents (figure 1.9). Ainsi, les motifs d'une texture peuvent tout à fait être structurés et réguliers (comme pour les pavages du plan), structurés et irréguliers (par exemple un mur de pierre). Les motifs peuvent également être non structurés et réguliers ou quasi-réguliers (comme les mailles d'un tissu) ou non structurés et irréguliers (c'est le cas, par exemple, des bruits et des textures de terre ou de pierre).

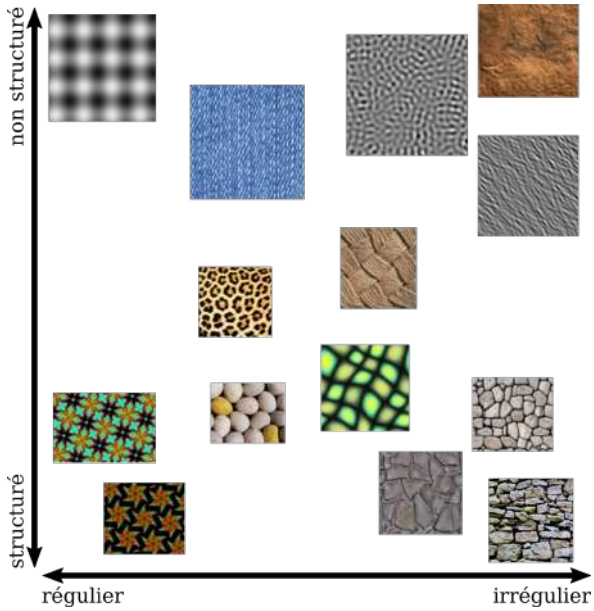


Figure 1.9 – *Classification de textures selon les deux axes précédents. Sur l'axe horizontal, le caractère régulier ou irrégulier et sur l'axe vertical, la présence ou non de structure.*

Les textures procédurales peuvent être définies au moyen de graphes de textures. La première version de ces graphes a été présentée par Robert Cook en 1984 [Coo84]. Ce sont des structures de données composées de champs scalaires en entrée et de nœuds d'opérations entre ces entrées. Ils sont aujourd'hui utilisés dans des logiciels professionnels tels que *Substance Designer* de Adobe ou le moteur *RenderMan* de Pixar. Les graphes permettent de proposer aux artistes une manipulation plus intuitive des notions mathématiques nécessaires à la définition des procédures de génération (figure 1.10).

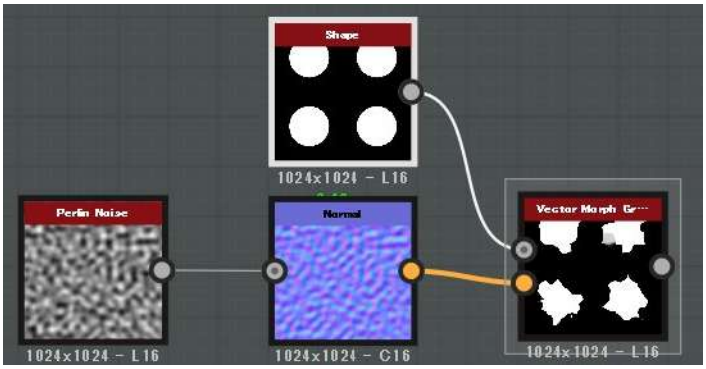


Figure 1.10 – *Un graphe simple permettant d'obtenir une texture procédurale (à droite) à partir de deux champs scalaires (un bruit de Perlin à gauche et une forme en haut). Capture d'écran du logiciel* Adobe Substance Designer.

Après avoir défini toutes les caractéristiques de la scène virtuelle (entre autres la géométrie des objets et leur texture), vient le rendu. Cette étape consiste à calculer l'image de la scène par une caméra virtuelle. Par exemple, la méthode de rendu par lancer de rayons (ou *ray tracing* en anglais) simule les rebonds de rayons lumineux sur les surfaces. Pour obtenir l'image finale, il faut pouvoir connaître l'information de couleur à afficher pour chaque *pixel* (contraction de l'anglais *picture element*). Cependant, un seul *pixel* de l'image peut contenir une portion plus ou moins grande de la scène virtuelle si la scène est éloignée de la caméra (figure 1.11).

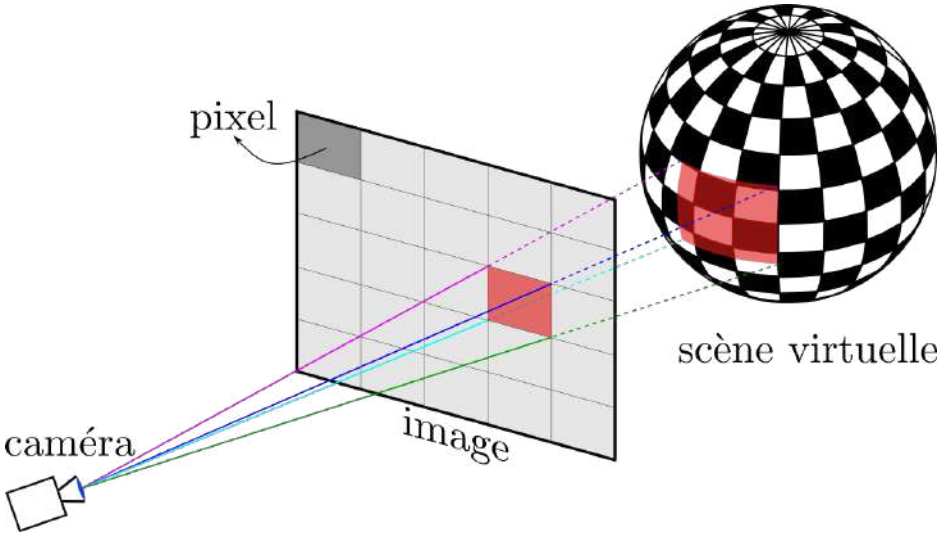


Figure 1.11 – *Projection de l'empreinte d'un pixel de l'image dans la scène virtuelle.*

Chaque *pixel* de l'image finale doit prendre la couleur moyenne des textures présentes dans son empreinte dans la scène virtuelle (figure 1.12). Le filtrage consiste à calculer cette couleur moyenne. Il nécessite l'évaluation d'une intégrale sur une surface non plane (la surface des objets couverte par l'empreinte du *pixel*). La difficulté pour le filtrage est que le pipeline graphique, permettant le rendu en temps réel, nécessite que les calculs soient complètement parallélisés (indépendants pour chaque *pixel*) et faits à la volée.

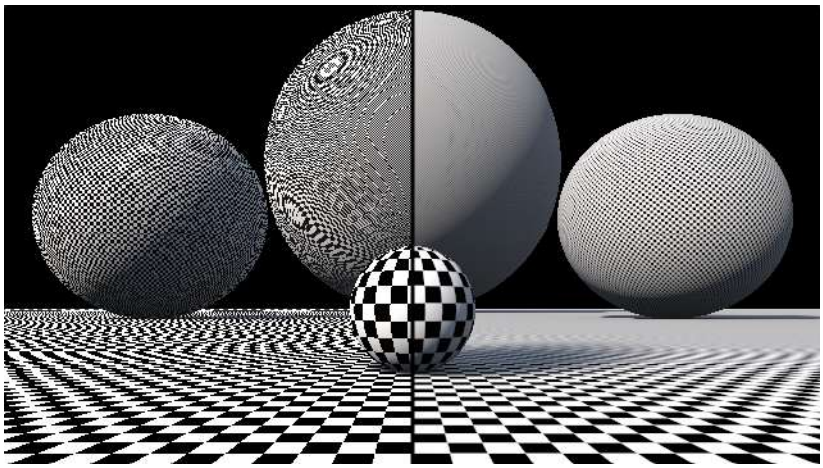


Figure 1.12 – *Rendu d'une scène virtuelle avec filtrage (à droite) et sans filtrage (à gauche). En l'absence de filtrage, des artéfacts visuels apparaissent sur le résultat. Image issue de* `iquilezles.org`[2].

1.2 Objectif de la thèse

L'objectif de cette thèse est de proposer un modèle permettant la génération et le rendu en temps réel d'un large éventail de motifs procéduraux structurés. Ce modèle s'inspire de la composition de bruits gaussiens par des cartes de couleurs (figure 1.13). Entre autres, nous nous sommes intéressés à la difficulté représentée par la présence de structure dans les textures procédurales, aussi bien lors de leur génération que lors de leur rendu.

2. https://iquilezles.org/articles/checkerfiltering/

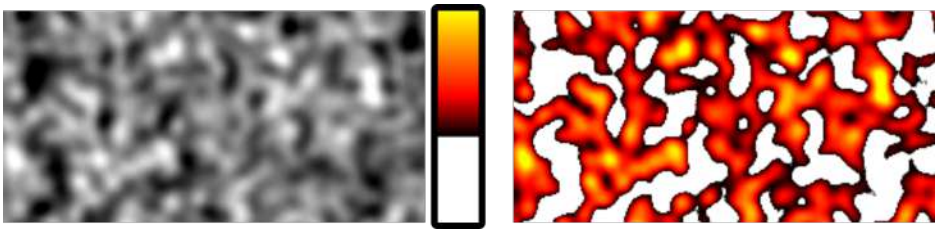


Figure 1.13 – *Composition d'un bruit par une carte de couleurs. La carte de couleurs (au centre) est appliquée sur le champ scalaire (à gauche) pour obtenir le résultat (à droite). Image inspirée de [HNPN13].*

Lors de la génération : Comme nous allons le voir dans l'état de l'art (section 2.4), les méthodes de génération procédurales peinent à associer la génération de motifs stochastiques structurés avec une large variété de motifs. En effet, les méthodes permettant la génération de structure sont généralement spécialisées pour un type de motifs et ne peuvent pas en créer d'autres.

Lors du rendu : La génération de motifs structurés se caractérise par la présence de contours à forts contrastes dans le résultat. Ces variations franches provoquent des discontinuités dans la description mathématique de la texture et vont créer des artéfacts comme du scintillement ou des effets de moiré lors du rendu d'une scène (figure 1.12, gauche).

Ce manuscrit porte sur la génération et le filtrage de textures à la fois stochastiques et structurées, et plus particulièrement, sur leur création, leur contrôle et leur filtrage.
La contribution centrale de cette thèse est la présentation d'un modèle basé non pas sur un bruit scalaire, mais sur un bruit vectoriel et sur la composition par une fonction multi-dimensionnelle. La montée en dimension par rapport aux cartes de couleurs traditionnelles (figure 1.13) offre une plus grande liberté dans l'agencement des couleurs. Cela permet la création d'une plus large variété de motifs structurés ou non (figure 1.14). L'idée clé de la génération de structure par cette méthode repose sur la séparation entre la partie stochastique de la texture (apportée par le bruit gaussien vectoriel) et la création de structure par l'introduction de discontinuités dans la carte de couleurs.

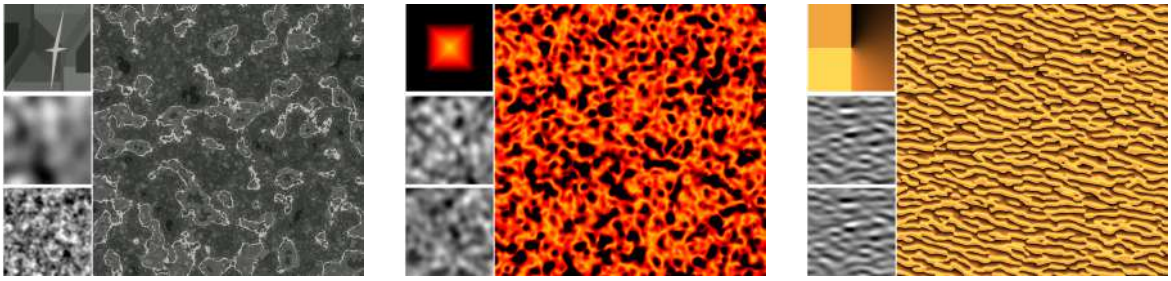


Figure 1.14 – *Composition de bruits vectoriels par des cartes de couleurs bi-dimensionnelles. Pour chaque texture, la carte de couleurs (en haut à gauche) est appliquée sur les composantes du bruit vectoriel (au milieu et en bas à gauche) pour obtenir le résultat (à droite). Images issues de [GSDT22].*

1.3 Organisation du manuscrit

Dans ce manuscrit, nous commençons par le traditionnel chapitre d'état de l'art (chapitre 2) au cours duquel nous présentons en détail les définitions et les principales notations utilisées avant de faire un tour d'horizon des méthodes présentes dans la littérature pour la génération et le rendu de motifs procéduraux structurés. Nous présentons ensuite notre travail sur la composition par des cartes de couleurs bi-dimensionnelles (chapitre 3), puis leur application à la création de terrains virtuels (chapitre 4). Enfin, nous concluons par une synthèse et une présentation des perspectives de recherche (chapitre 5).

Chapitre 2

État de l'art

Nous avons présenté en introduction différents aspects de la synthèse de textures (chapitre 1). En particulier, nous nous sommes intéressés à l'importance de la génération procédurale et aux difficultés que représente la génération de structure, autant lors de la génération que du rendu. Dans ce chapitre, nous commençons par définir les termes et notations importantes pour la suite de ce manuscrit. Nous faisons ensuite un état de l'art des méthodes de génération et de rendu de textures procédurales, au sens de Ebert et collègues [EMP+02].
Nous présentons d'abord des algorithmes de génération procédurale de bruits (section 2.2). Puis, nous nous intéressons aux méthodes de contrôle du résultat de la génération procédurale (section 2.3). Ensuite, nous développons la génération procédurale de structure (section 2.4). Enfin, nous regardons quelques méthodes de rendu et de filtrage (section 2.5). Nous concluons ce chapitre par un positionnement des contributions de cette thèse par rapport aux méthodes précédentes et à leurs limitations (section 2.6).

Sommaire

2.1 Définitions et notations

Les textures. Dans un monde virtuel, les textures sont utilisées pour simuler le comportement lumineux des éléments qui le compose. Elles sont généralement appliquées sur la surface des maillages 3D représentant les objets. Mathématiquement, une texture est donc une fonction qui associe un comportement lumineux à tous les points $\mathbf{u}$ de l'espace texture.

$$\begin{aligned} T : \mathbb{X} &\to \mathbb{K} \\ \mathbf{u} &\mapsto T(\mathbf{u}) \end{aligned} \tag{2.1}$$

Le comportement lumineux décrit peut être une intensité ($\mathbb{K} = \mathbb{R}$), une couleur ($\mathbb{K} = \mathbb{R}^3$ dans le cas d'une décomposition trichromatique) ou une autre propriété du comportement lumineux (comme la rugosité, la brillance...). L'espace de définition $\mathbb{X}$ peut être une surface ($\mathbb{X} \subset \mathbb{R}^2$), un volume ($\mathbb{X} \subset \mathbb{R}^3$) ou un espace de dimension supérieur (pour des textures volumiques variant temporellement par exemple). Les textures sont souvent stockées en mémoire sous la forme d'un tableau représentant une image discrète, dans ce cas, l'espace de définition $\mathbb{X}$ est un intervalle de $\mathbb{N}^2$.

La génération procédurale. Comme nous l'avons vu dans l'introduction (chapitre 1.1), il existe plusieurs façons de créer des textures. Ainsi, elles peuvent être décrites par une structure de données stockée en mémoire (comme une image) ou par une procédure. Dans ce second cas, la procédure constitue une abstraction de la texture et est évaluée uniquement lorsque c'est nécessaire. Par abus de langage, les textures obtenues au moyen d'un algorithme décrivant une procédure sont souvent appelées "textures procédurales". La génération procédurale, au sens de Ebert et collègues [EMP+02], tend à avoir de nombreux avantages :

- **Compacité en mémoire** : les instructions des procédures sont stockées en mémoire sous forme de texte et ne sont exécutées qu'aux moments où elles sont nécessaires. Cela occupe moins de mémoire comparé au stockage d'une image.
- **Continuité et indépendance de la résolution** : la texture ne dépend pas d'une structure de donnée discrète, elle peut donc être évaluée à n'importe quelle échelle de résolution.
- **Infinie et non périodique** : la texture peut être appliquée sur un espace arbitrairement grand sans faire apparaître de répétitions non souhaitées.
- **Contrôle paramétrique** : les caractéristiques de la texture (comme la rugosité ou la couleur) peuvent être associées à des paramètres de la procédure, qui peuvent être modifiés pour obtenir différentes variations du résultat.
- **Parallélisation** : la procédure peut être évaluée en temps constant et indépendamment pour n'importe quelle position de son espace de définition, cela permet de générer le résultat à la volée et uniquement au moment et à l'endroit où il doit être affiché à l'écran.

Notons cependant que la définition de la génération procédurale reste assez floue. Ainsi, tout ce qui est fait par un ordinateur présente un aspect procédural à un certain niveau. Remarquons également que certaines méthodes de génération procédurale ne remplissent pas parfaitement tous les avantages précédents. Par exemple, celles basées sur l'utilisation d'un échantillon de texture stocké en mémoire sont dépendantes de la résolution de cet échantillon et de la discrétisation de sa structure de données. Finalement, selon Ebert et collègues [EMP+02], la caractéristique principale permettant de définir la génération procédurale est que le résultat est généré par un programme ou un modèle et ne repose pas uniquement sur une image numérisée.

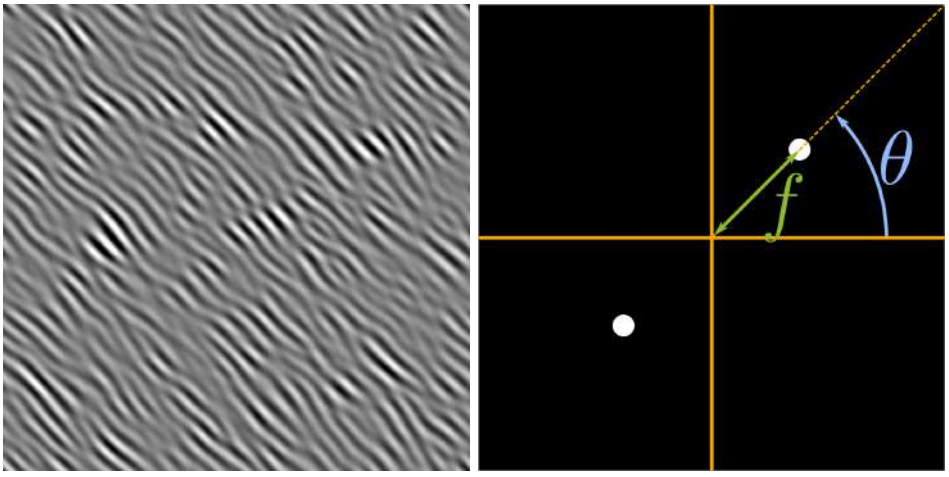
Les bruits. Pour obtenir des résultats plus proches de la réalité, une part d'aléatoire est introduite dans la génération procédurale au moyen de bruits. Mathématiquement, le bruit est un processus aléatoire (ou fonction aléatoire), c'est-à-dire un modèle probabiliste permettant d'étudier un phénomène aléatoire en fonction d'une variable (par exemple le temps ou l'espace). En traitement du signal, les bruits sont des modifications non souhaitées (et en général inconnues) que subit un signal lors de son acquisition, de son enregistrement, de son traitement ou de sa transmission. Dans ce domaine, l'étude des bruits a pour objectif de parvenir à les éliminer pour retrouver un signal "lisse" (non bruité).

En informatique graphique, au contraire, les bruits sont utilisés pour ajouter une part d'aléatoire dans des signaux. Par exemple, en 1985, Ken Perlin [Per85] a présenté une méthode de génération de bruit qui a été plusieurs fois reprise depuis. Son objectif était alors d'imiter le grain des pellicules et d'ajouter du bruit sur les images créées par ordinateur pour le film *Tron*, car celles-ci étaient trop lisses et dénotaient avec les images en prise de vue réelle.
Nous pouvons donc définir un bruit comme un signal aléatoire (ou processus stochastique), c'est-à-dire un signal qui ne se répète pas à l'identique lors de la réitération de l'expérience qui le produit. Ce signal peut être continu ou non et peut dépendre d'une seule dimension (par exemple le temps) ou de plusieurs (par exemple l'espace 2D pour les bruits surfaciques ou 3D pour les bruits volumiques). Dans ce manuscrit, nous nous intéressons principalement aux bruits tels que définis dans l'étude de Lagae et collègues [LLC+10]. C'est-à-dire un processus aléatoire stationnaire (dont les propriétés statistiques sont indépendantes de l'instant ou de la position initiale choisie), ergodique (dont les statistiques peuvent être approchées par l'étude d'une seule réalisation de taille suffisante) suivant une loi normale, entièrement décrit et contrôlé par sa densité spectrale de puissance. Ce bruit gaussien est noté N et est défini en tous points $\mathbf{u}$ de l'espace de texture.

Le domaine de Fourier. Dans le domaine temporel ou spatial, un signal est représenté par les variations de sa valeur en fonction du temps ou de l'espace. La transformée de Fourier d'un signal temporel ou spatial permet d'obtenir sa représentation fréquentielle. Celle-ci est une fonction à valeur complexe représentant l'amplitude et la phase des sinusoïdes composant le signal en fonction de leur fréquence. Ainsi, pour une fonction f définie pour tout t, sa transformée de Fourier $\mathcal{F}$ est définie pour toute fréquence ξ par

$$\mathcal{F}(\xi) = \int_{-\infty}^{+\infty} e^{-i\xi t} f(t) \mathrm{d}t \tag{2.2}$$

Dans le domaine fréquentiel, un signal peut être représenté par sa densité spectrale. Cette représentation est calculée avec le carré du module de la transformée de Fourier $|\mathcal{F}(\xi)|^2$. La densité spectrale de puissance (ou PSD, de l'anglais *Power Spectral Density*) représente la répartition de la puissance d'un signal en fonction des fréquences qui le composent. Elle permet de caractériser les signaux aléatoires, gaussiens, stationnaires et ergodiques tels que N. Dans le cas d'un bruit au sens mathématique, l'espace des fréquences est continu et infini. En informatique graphique, seule une discrétisation d'une portion finie de ce domaine est utilisée.
Dans le cas d'un bruit N défini sur l'espace 2D, les sinusoïdes qui le composent sont caractérisées par une fréquence f et une orientation θ sur le plan 2D. La densité spectrale de N représente donc la répartition des puissances sur un plan 2D ayant pour coordonnées polaires le couple (f, θ) (figure 2.1).



(a) *Représentation spatiale* **(b)** *Représentation spectrale*

FIGURE 2.1 – *Un bruit gaussien et sa représentation spectrale dans le domaine de Fourier*

Les graphes de textures. Les graphes sont des structures de données couramment utilisées dans de nombreux domaines scientifiques. Dans le cadre de la génération procédurale de textures, leur utilisation a été théorisée par Robert Cook en 1984 [Coo84]. Ils se présentent sous la forme de graphes directs et acycliques. Les nœuds sources représentent les données d'entrée, sous la forme de champs scalaires ou de fonctions mathématiques. Les nœuds internes au graphe réalisent les opérations entre les données d'entrée en parallèle pour chaque position, à la façon d'un *fragment shader*. Enfin, le ou les nœuds puits contiennent les éléments de la texture finale. La procédure de génération de la texture consiste alors à parcourir le graphe des sources vers les puits. Ces graphes permettent de représenter les algorithmes de génération de textures (figure 2.2).

Algorithm 1: Génération procédurale

Fonctions:
Perlin_Noise : $\mathbf{u} \mapsto N$;
Normal : $n \mapsto$ vecteur normal ;
Shape : $\mathbf{u} \mapsto$ valeur $\in \{0, 1\}$;
Sortie : valeur *résultat*
Pour tous *coordonnées* **u** *(en parallèle)* **:**
  $N \leftarrow$ Perlin_Noise(**u**) ;
  vecteur_normal $\leftarrow$ Normal(N) ;
  résultat $\leftarrow$ Shape(**u** + vecteur_normal) ;
Fin

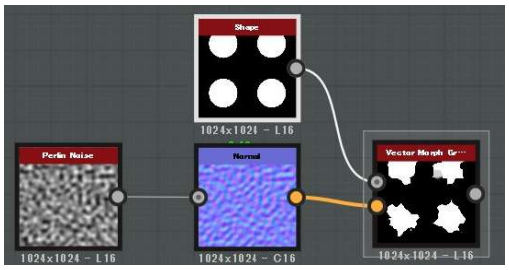


Figure 2.2 – *Représentation en graphe de l'algorithme 1. Capture d'écran du logiciel* Adobe Substance Designer.

Les textures structurées. Le terme "structure" peut désigner plusieurs caractéristiques différentes selon le contexte dans lequel il est utilisé. Ainsi, il peut désigner la régularité ou la répétition des éléments présents dans une texture. Dans ce manuscrit, nous considérons la définition proposée par Sauvage [Sau18]. Nous disons des motifs d'une texture qu'ils sont structurés s'ils présentent des arêtes vives ou des contours à fort contraste (figure 2.3).

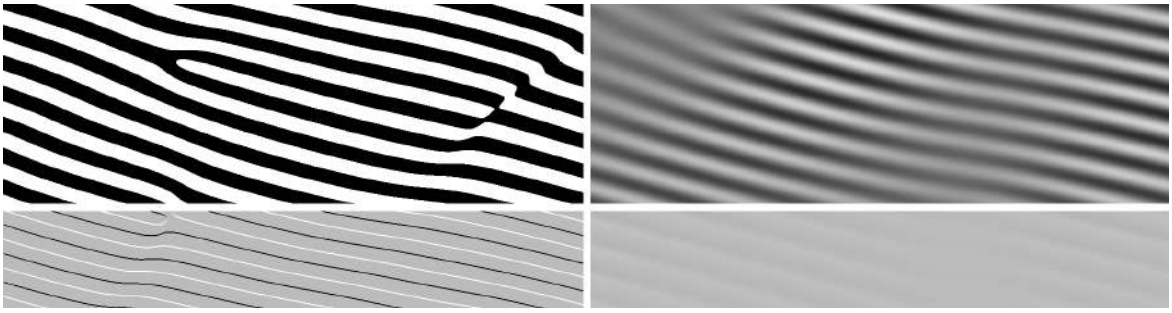


Figure 2.3 – *Un exemple de texture procédurale. La partie gauche de la texture présente des motifs structurés. L'utilisation d'un filtre de Sobel (en bas) met en avant la présence de structure (à gauche) ou son absence (à droite).*

Comme nous l'avons vu dans l'introduction (section 1.1), cette notion de structure est indépendante du caractère régulier ou irrégulier de la texture. Ainsi, une texture peut être structurée et régulière (comme un pavage) ou structurée et irrégulière (comme la partie gauche de la figure 2.3).

Le rendu. Après avoir entièrement décrit un monde virtuel, il faut en faire un rendu pour obtenir une ou plusieurs images. Le rendu consiste à calculer l'image d'une scène, capturée par une caméra virtuelle, en simulant les interactions lumineuses. Le résultat du rendu est ensuite affiché à l'écran à la volée ou stocké en mémoire sous la forme d'un tableau. Dans les deux cas, une discrétisation de l'image résultat doit être faite. Ainsi, le résultat du rendu est découpé en un nombre fini de *pixels*. Lors de cette discrétisation, chaque *pixel* est associé à une surface plus ou moins étendue du monde virtuel. En effet, si un objet est très près de la caméra, il occupera une place importante sur le rendu, donc les *pixels* le représentant ne couvriront chacun qu'une petite portion de l'objet. Cependant, dans le cas d'objets plus éloignés de la caméra, le nombre de *pixels* les représentant est plus faible

et chacun couvre donc une surface plus grande. La projection, dans le monde virtuel, de la surface couverte par chaque *pixel* est appelée "empreinte de *pixel*" et nous la notons $\mathcal{P}$ (figure 2.4).

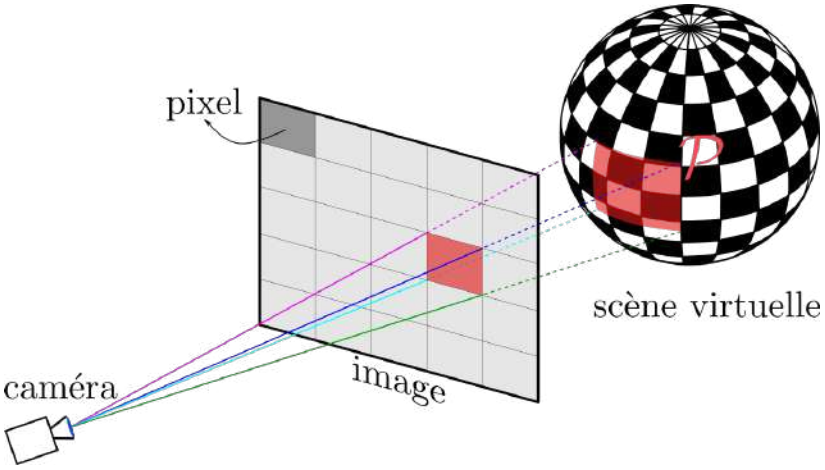


Figure 2.4 – *Projection de l'empreinte $\mathcal{P}$ d'un pixel de l'écran dans la scène virtuelle. Le pixel surligné en rouge dans l'image doit prendre la couleur moyenne des éléments de la scène virtuelle présents dans son empreinte $\mathcal{P}$.*

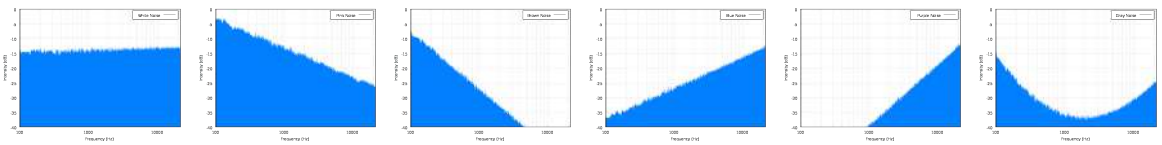
Pour éviter les artéfacts visuels, comme les effets de scintillement ou de moiré, chaque *pixel* du résultat doit prendre la couleur moyenne des textures présentes dans son empreinte $\mathcal{P}$. Ce calcul de moyenne s'appelle le filtrage.

2.2 Les bruits procéduraux

Comme nous l'avons vu dans la partie définition (section 2.1), le terme "bruit" peut désigner de nombreuses choses. Dans cette section, nous nous intéressons à des algorithmes permettant la génération en temps réel de bruits gaussiens stationnaires ergodiques. C'est-à-dire des processus aléatoires suivant une loi normale dont les propriétés statistiques sont indépendantes des conditions initiales et peuvent être approchées par l'étude d'une seule réalisation de taille suffisante.

2.2.1 Vue d'ensemble

Les bruits colorés. Un bruit est un signal aléatoire, il peut être caractérisé par ses propriétés statistiques, et notamment par sa densité spectrale de puissance (PSD). Dans le cas d'un bruit à une dimension (signal temporel), cette densité représente la répartition de la puissance pour chaque fréquence de sinusoïde présente dans le bruit. Elle permet de distinguer différents types de bruits, chacun associé à une couleur. L'idée d'association entre le spectre d'un bruit et une couleur vient d'une analogie entre les ondes sonores et lumineuses. Nous pouvons donc distinguer plusieurs "couleurs" de bruits en fonction de leur représentation spectrale (figure 2.5).



(a) *Blanc* **(b)** *Rose* **(c)** *Brun* **(d)** *Bleu* **(e)** *Violet* **(f)** *Gris*

Figure 2.5 – *Représentation spectrale de bruits colorés. Images issues de* `Wikipedia`[1].

1. https://fr.wikipedia.org/wiki/Bruits_colorés

Par exemple, le bruit blanc est caractérisé par un spectre dit "plat" (figure 2.5a). C'est-à-dire que la puissance est répartie également entre toutes les fréquences $\xi \in \mathbb{R}$.
En informatique graphique, les bruits sont utilisés pour ajouter de l'aléatoire dans les images créées. Les différents algorithmes existants tendent donc à contrôler ces bruits de façon à obtenir le résultat souhaité lors de la phase de création d'une image par un utilisateur.

Les bruits par interpolation. Ces méthodes sont basées sur l'interpolation de valeurs aléatoires ou de gradients définis sur les sommets d'un maillage discret périodique. L'exemple le plus connu de cette méthode est le bruit de Perlin [Per85] (figure 2.6). Pour cela, Perlin propose de considérer la grille discrète des entiers, définie sur l'espace de la texture. Le bruit est ensuite évalué en tous points de l'espace en calculant l'interpolation entre les gradients pseudo-aléatoire des huit sommets les plus proches dans la grille. Les gradients pseudo-aléatoires sont donnés par une fonction de hachage appliquée en chaque point de la grille. Cet algorithme a été grandement utilisé depuis sa présentation, notamment dans l'industrie, en raison de sa grande simplicité.

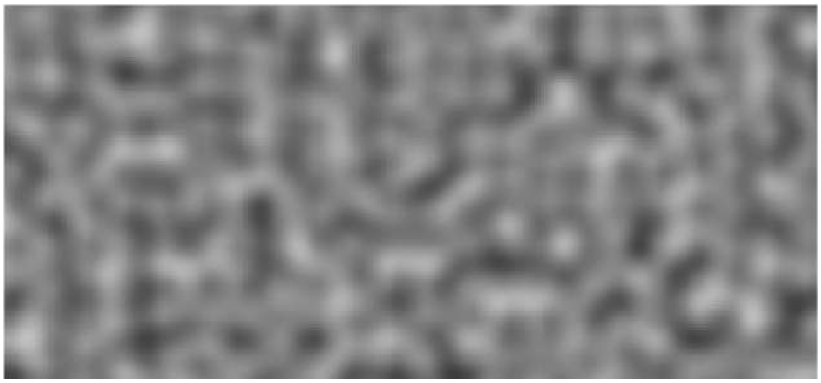


Figure 2.6 – *Une réalisation du bruit de Perlin. Image issue de [Per02].*

Plusieurs variations et améliorations du bruit de Perlin ont été présentées comme le *flow noise* [PN01], le *simplex noise* [OAH+04], le *curl noise* [BHN07], le *better gradient noise* [KKS+08] ou, plus récemment, le *prime gradient noise* [TSP21].

Les bruits explicites. Les bruits dits "explicites" sont basés sur le pré-calcul et le stockage d'une réalisation d'un bruit, sous la forme d'une image, en vue de la génération d'un bruit similaire. Ils ne sont pas vraiment procéduraux dans le sens strict du terme, en effet, ils exigent une phase de pré-calcul, de stockage et le résultat de la génération est borné dans l'espace. Cependant, ils restent pertinents pour de la synthèse de bruits. En effet, ils ne consistent pas à calculer intégralement le résultat en amont. Ils font cependant le choix d'augmenter le coût en mémoire pour augmenter la qualité du résultat. Parmi ces méthodes, nous pouvons citer le *wavelet noise* [CD05] (figure 2.7) ou le *anisotropic noise* [GZD08].

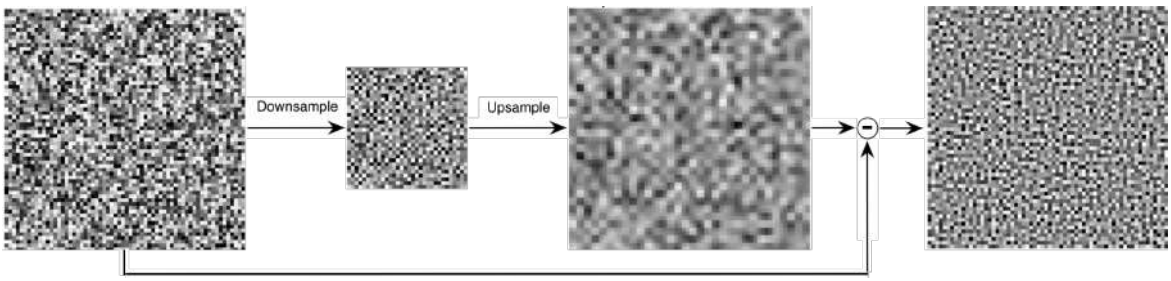


Figure 2.7 – *Calcul d'une réalisation du wavelet noise. Image issue de [CD05].*

Les bruits par convolution. Les bruits par convolution sont décrits, dans le domaine spatial, par la convolution d'un bruit blanc par un élément k appelé noyau. Mathématiquement, il s'agit donc du filtrage d'un bruit blanc par un noyau. Par transformée de Fourier, une convolution dans le domaine spatial est équivalente à la multiplication des spectres dans le domaine spectral. Or, le spectre d'un bruit blanc étant une constante, le spectre de la convolution correspond donc exactement au spectre du noyau k. Intuitivement, la convolution d'un bruit blanc (spectre constant) par un noyau (spectre contrôlé) permet de générer un signal dont l'aléatoire est apporté par le bruit blanc et dont le contrôle est fait par le noyau k. Cependant, le produit de convolution sur un bruit blanc continu est une opération trop coûteuse pour être réalisée en temps réel.

Dans une série d'articles [Lew84, Lew86, Lew89], John-Peter Lewis a présenté les bases de la génération par convolution parcimonieuse. Cette méthode consiste à calculer la convolution avec le noyau k en remplaçant le bruit blanc continu par un processus ponctuel de Poisson τ.

$$N(\mathbf{u}) = \int_{\mathbb{R}^2} \tau(\mathbf{t}) k(\mathbf{u} - \mathbf{t}) \mathrm{d}\mathbf{t} \tag{2.3}$$

Le processus de Poisson τ est composé d'impulsions de Dirac δ d'intensités et de positions aléatoires :

$$\tau(\mathbf{t}) = \sum_i a_i \delta(\mathbf{t} - \mathbf{t}_i) \tag{2.4}$$

De cette façon, la convolution devient possible à calculer en temps réel. Le noyau k est généralement choisi à support fini, de façon à ce que sa contribution soit nulle au-delà d'une certaine distance. Ainsi, pour connaître le bruit résultat N aux coordonnées $\mathbf{u}$, il faut évaluer les noyaux qui se recouvrent pour ce point. Cette évaluation peut être accélérée en considérant une grille dont la taille des cellules est égale au rayon du noyau. Seuls les noyaux dont le centre se situe dans la cellule étudiée ou dans les cellules voisines sont alors évalués. Finalement, la convolution parcimonieuse s'écrit de façon discrète par

$$N(\mathbf{u}) = \sum_{\mathbf{x}_i \in \tau} w_i k(\mathbf{u} - \mathbf{x}_i) \tag{2.5}$$

avec w_i les pondérations et k les noyaux centrés en $\mathbf{x}_i$. Plusieurs algorithmes ont été proposés autour de cette formulation.

2.2.2 Convolution parcimonieuse de noyaux

Convolution de noyaux de Gabor. En 2009, Lagae et collègues [LLDD09], ont proposé d'utiliser les noyaux décrit par Dennis Gabor en 1946 [Gab46]. Les avantages de ces noyaux sont qu'ils sont facilement paramétrables et à support compact dans le domaine spatial et dans le domaine spectral. Ces derniers éléments permettent respectivement une évaluation efficace du résultat et un contrôle précis par la densité spectrale de puissance. Les noyaux de Gabor sont définis par le produit d'une sinusoïde, paramétrée par une fréquence f et une orientation θ, appelée *Harmonique* et d'une fonction gaussienne tronquée :

$$\begin{aligned} Harmonique(\mathbf{x}) &= \sin\left(2\pi f \begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix} \cdot \mathbf{x}\right) \\ Gaussienne(\mathbf{x}) &= K \exp^{\left(-\frac{||\mathbf{x}||^2}{2\sigma^2}\right)} \end{aligned} \tag{2.6}$$

Le noyau k est donc défini par

$$k(\mathbf{x}) = K \exp^{\left(-\frac{||\mathbf{x}||^2}{2\sigma^2}\right)} \sin\left(2\pi f \begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix} \cdot \mathbf{x}\right) \tag{2.7}$$

Dans le domaine de Fourier, k est alors représenté par deux lobes de coordonnées polaires : l'orientation θ et la fréquence f (figure 2.8).

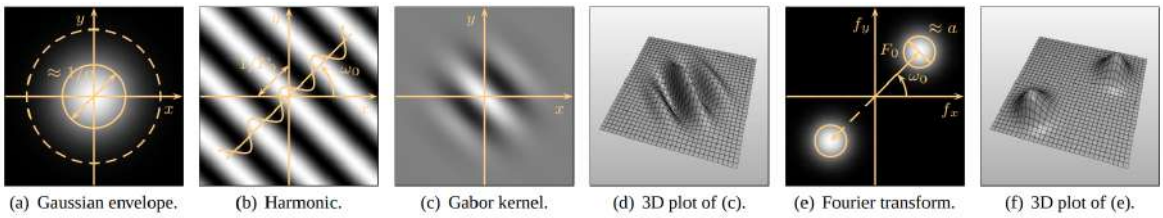


Figure 2.8 – *Décomposition d'un noyau de Gabor. L'harmonique (b) est pondérée par l'enveloppe gaussienne (a) pour obtenir le noyau (c). La représentation en 3 dimensions du noyau (d) permet de bien observer l'effet de la pondération. La représentation spectrale du noyau (e, f) est un bi-lobe dont les coordonnées polaires sont la fréquence et l'orientation du noyau. Image issue de [LLDD09].*

Ces noyaux sont ensuite placés à des positions aléatoires dans le domaine spatial (figure 2.9), suivant le principe de la convolution parcimonieuse (équation (2.5)), pour obtenir le bruit final N :

$$N(\mathbf{u}) = \sum_{\mathbf{x}_i \in \tau} w_i(\mathbf{u}) k(\mathbf{u} - \mathbf{x}_i) \tag{2.8}$$

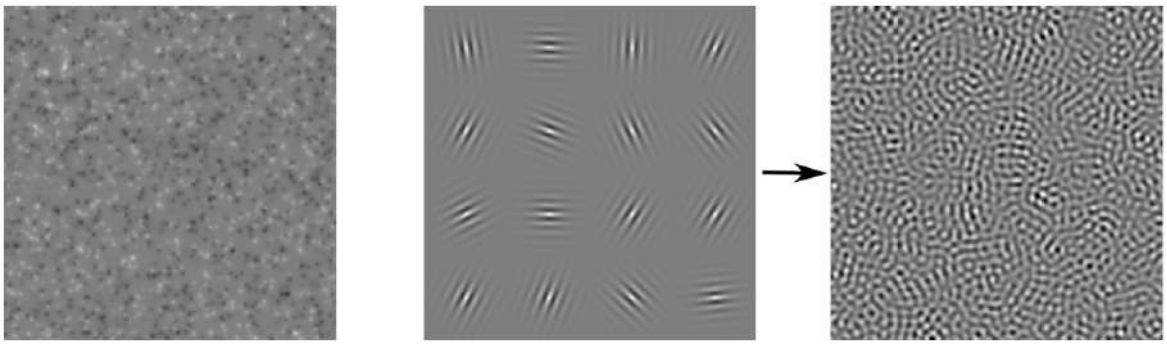


Figure 2.9 – *À gauche, les positions aléatoires et leur pondération dans le domaine spatial. Au centre, le résultat avec quelques noyaux. À droite, le résultat final de la convolution. Image issue de [GSV+14].*

Par linéarité de la transformée de Fourier, le spectre du résultat est composé de la somme des spectres des noyaux.

Le bruit par *texton*. Galerne et collègues [GLM17] ont proposé en 2017 une méthode de génération par convolution parcimonieuse utilisant un noyau stocké en mémoire. À partir d'une texture exemple de taille finie, les auteurs définissent le noyau k, qu'ils appellent "*texton*" (figure 2.10). Ce noyau est une représentation condensée du contenu fréquentiel de la texture d'entrée. Un champ aléatoire n est obtenu par convolution parcimonieuse en utilisant un processus de Poisson τ

$$n(\mathbf{u}) = \sum_{\mathbf{x}_i \in \tau} k(\mathbf{u} - \mathbf{x}_i) \tag{2.9}$$

Soit λ l'intensité du processus de Poisson τ. Le bruit final N est obtenu par normalisation du champ aléatoire précédent :

$$N(\mathbf{u}) = \frac{n(\mathbf{u}) - \mathbb{E}(n(\mathbf{u}))}{\sqrt{\lambda}} \tag{2.10}$$

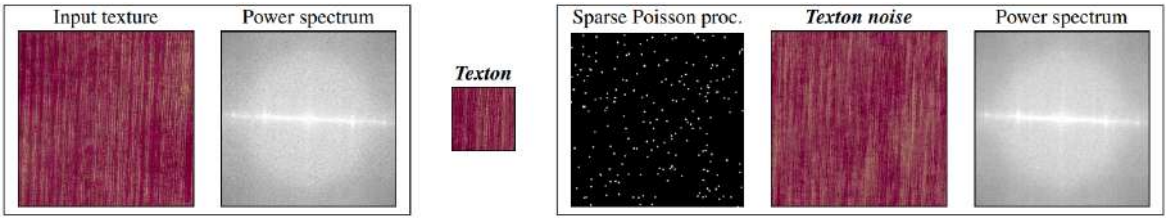


Figure 2.10 – *L'exemple d'entrée (à gauche) est analysé pour obtenir sa représentation spectrale. Les informations sont ensuite condensées dans un texton (au centre). Puis le résultat (à droite) est obtenu par convolution parcimonieuse. La représentation spectrale du résultat est très proche de celle de l'exemple. Image issue de [GLM17].*

2.2.3 Mélange de noyaux sur une grille

Bruit local à phase aléatoire. En 2014, Gilet et collègues [GSV+14] ont proposé une autre approche, non plus basée sur des noyaux mono-fréquences positionnés aléatoirement, mais sur des bruits locaux (pouvant être vus comme des noyaux multi-fréquences) positionnés sur une grille régulière. Les noyaux k_i sont donc composés d'une somme d'harmoniques contrôlées par leur amplitude A, fréquence f, orientation θ et phase φ.

$$k_i(\mathbf{u}) = \sum_j A_{i,j} \cos\left(2\pi f_{i,j} \begin{bmatrix}\cos(\theta_{i,j})\\ \sin(\theta_{i,j})\end{bmatrix} \cdot \mathbf{u} + \varphi_{i,j}\right) \tag{2.11}$$

Ces noyaux décrivent localement le bruit final. Ils sont positionnés sur une grille régulière (figure 2.11). Pour conserver la description locale, ils sont chacun pondérés par une fonction w représentant une fenêtre de Kaiser-Bessel, de largeur Δ et centrée à la position $\mathbf{x}_i$ du noyau correspondant. Où Δ dépend du nombre d'harmoniques utilisées pour les noyaux et de la résolution de la grille sur laquelle ils sont positionnés.
Le bruit final N est ensuite obtenu par la somme pondérée des noyaux :

$$N(\mathbf{u}) = \sum_{\mathbf{x}_i \in \tau} w\left(\frac{||\mathbf{u} - \mathbf{x}_i||}{\Delta}\right) k_i(\mathbf{u}) \tag{2.12}$$

Dans ce modèle, le spectre du résultat est entièrement représenté par le spectre des noyaux.

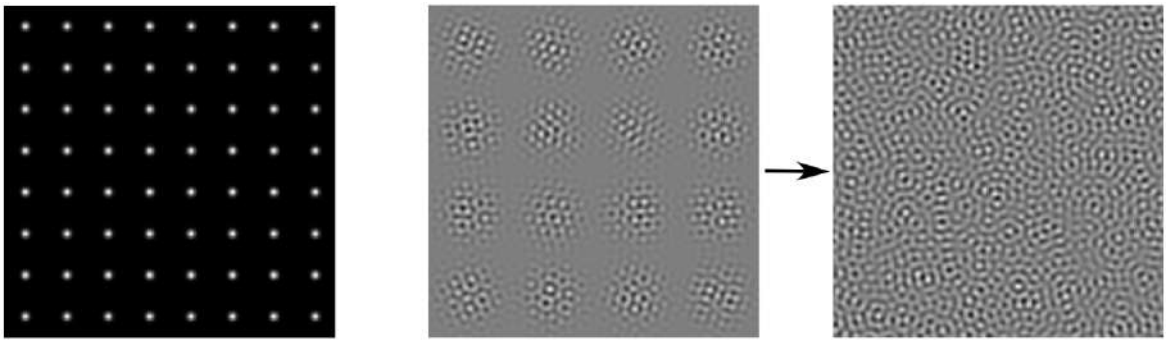


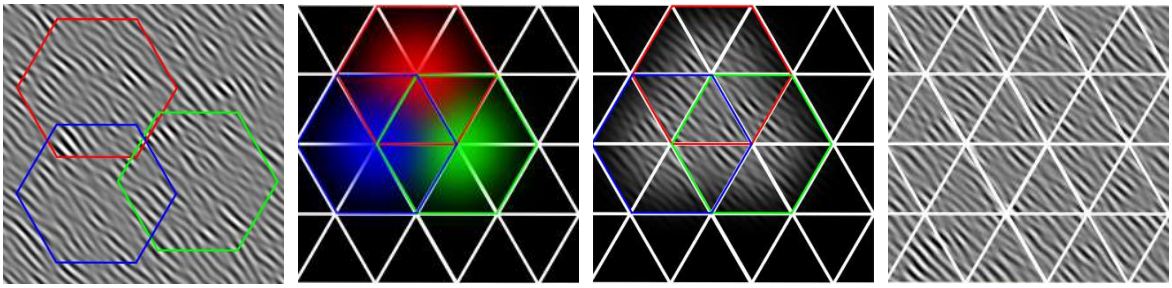
Figure 2.11 – *À gauche, les positions sur une grille régulière dans le domaine spatial. Au centre, le résultat avec quelques noyaux. À droite, le résultat final de la convolution. Image issue de [GSV+14].*

Le pavage et mélange. Une autre méthode, utilisant une grille régulière et des bruits locaux, a été proposée en 2018 par Heitz et collègues [HN18]. Cette méthode prend en entrée une texture exemple E de taille finie, décrivant un bruit gaussien de moyenne nulle. L'hypothèse de moyenne nulle peut être considérée sans perte de généralité étant donné que l'ajout d'une constante aux valeurs du résultat d'un processus gaussien ne modifie par ses caractéristiques spectrales.

Contrairement au bruit par *texton*, la méthode de Heitz et collègues utilise des tuiles k_i directement tirées dans l'exemple d'entrée (figure 2.12a).

$$k_i(\mathbf{x}) = E(\mathbf{x} + h_i(\mathbf{x})) \tag{2.13}$$

Avec h_i une fonction de hachage permettant le tirage de la tuile i centrée à une position aléatoire dans l'exemple E.



(a) *Tirage des E_i* **(b)** *Pavage* **(c)** *Mélange* **(d)** *Résultat*

FIGURE 2.12 – *Synthèse par pavage et mélange. Les tuiles E_i sont tirées dans une texture exemple 2.12a et placées sur les sommets d'une grille de triangles équilatéraux. Image inspirée de [HN18].*

Le bruit N est obtenu par mélange de ces tuiles. Pour cela, l'espace de sortie est divisé en un maillage régulier de triangles équilatéraux et une tuile k_i est associée à chaque sommet (figure 2.12b). Pour un triangle donné dans l'espace de sortie, les auteurs définissent le poids w_i de chacune des trois tuiles le recouvrant en utilisant les coordonnées barycentriques b_1, b_2, b_3 dans ce triangle.

$$w_i(\mathbf{x}) = \frac{b_i(\mathbf{x})}{\sqrt{\sum_{j=1}^{3} b_j^2(\mathbf{x})}} \tag{2.14}$$

Le bruit final N est donc défini en tous points par le mélange des trois tuiles recouvrant le triangle associé (figure 2.12c).

$$N(\mathbf{u}) = \sum_{i=1}^{3} w_i(\mathbf{u}) k_i(\mathbf{u}) \tag{2.15}$$

Ce mélange permet de conserver la variance des valeurs du processus gaussien décrit dans E.

2.3 Le contrôle

Un élément important lors de la génération procédurale d'un bruit, ou plus généralement d'une texture, est de pouvoir contrôler le résultat. À ce sujet, nous distinguons deux éléments. D'une part, le contrôle de l'apparence (section 2.3.1), qui vise à contraindre localement les caractéristiques spectrales du bruit. D'autre part, le contrôle de l'organisation (section 2.3.2), qui vise à faire varier cette contrainte de caractéristiques spatialement.

2.3.1 Contrôle de l'apparence

Contrôle du spectre par le noyau. Comme nous l'avons vu un peu plus tôt en section 2.2.1, dans le cas des méthodes par convolution parcimonieuse, le spectre du résultat reflète le spectre du noyau utilisé. En effet, ces méthodes de génération se reposent sur la convolution, dans le domaine spatial, d'un bruit blanc (apportant l'aléatoire) par un noyau (définissant l'apparence). Dans le domaine spectral, cette convolution est équivalente à la multiplication du spectre du noyau (qui est contrôlé) par le spectre du bruit blanc (qui est une constante). Par exemple, en 1991, Van Wijk présente le bruit ponctuel (ou *spot noise* en anglais) [VW91], pour lequel l'apparence du résultat est liée à la forme du noyau choisi, dans le domaine spatial (figure 2.13).

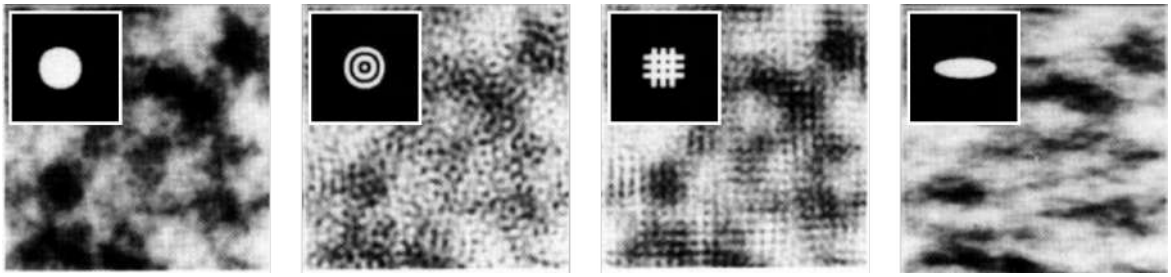
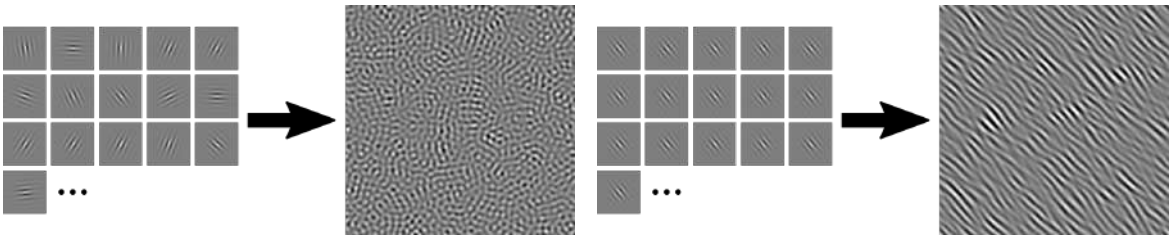


Figure 2.13 – *Plusieurs résultats du spot noise selon la forme du noyau utilisé (en vignette en haut à gauche). Image issue de [VW91].*

Dans le cas du bruit par composition de noyaux de Gabor [LLDD09], les caractéristiques de la texture générée sont également décrites par le noyau utilisé. Cependant, dans ce cas, le noyau ne contient qu'une seule harmonique, pondérée par une fonction gaussienne (équation (2.8)). Par linéarité de la transformée de Fourier, le spectre du bruit généré est composé de la somme des spectres des réalisations du noyau utilisées. Ainsi, le spectre du résultat est contrôlé par les plages d'orientations et de fréquences possibles pour chaque réalisation du noyau (figure 2.14).



(a) *Résultat isotrope* **(b)** *Résultat anisotrope*

Figure 2.14 – *Impact de l'orientation des noyaux sur l'apparence du résultat. À gauche, toutes les orientations de noyau sont possibles lors du tirage aléatoire des orientations, le bruit final est donc isotrope. À droite, une seule orientation est possible lors du tirage aléatoire des orientations, le bruit final est donc anisotrope.*

Dans le cas du bruit local à phase aléatoire [GSV+14], le spectre du noyau représente complètement le spectre du résultat. En effet, les noyaux utilisés sont composés de sommes d'harmoniques. Ils sont ensuite placés sur une grille régulière et pondérés de façon à conserver leur description locale. Ainsi, le sceptre du résultat est directement contrôlé par le spectre du noyau.

Les méthodes par l'exemple. Les méthodes de contrôle basées sur les caractéristiques spectrales des noyaux ont l'inconvénient d'être peu intuitives pour des utilisateurs non scientifiques, et donc difficiles à utiliser par des artistes. Pour palier à cette difficulté, il existe des algorithmes permettant d'extraire les caractéristiques du noyau à partir d'un exemple de taille finie, au cours d'une étape de pré-calcul. Cela permet de contrôler finement le résultat de façon plus intuitive. Voici une liste non exhaustive de quelques méthodes de génération par l'exemple.

Ghazanfarpour et Dischler [GD95] proposent une méthode pour analyser le spectre d'une texture 2D. Leur objectif est de créer automatiquement une texture 3D en effectuant une somme de cosinus paramétrée par les caractéristiques obtenues par la phase d'analyse. Plus récemment, Galerne et collègues [GLLD12] ont proposé une méthode pour généraliser le bruit par convolution de noyaux de Gabor [LLDD09] de façon à utiliser une texture exemple pour définir les caractéristiques du noyau. Pour cela, leur méthode se base sur une décomposition d'un spectre de puissance complexe en une somme de gaussiennes dans le domaine de Fourier (figure 2.15). De cette façon, en partant d'un exemple dont la répartition des intensités est gaussienne, ils parviennent à discrétiser le spectre de puissance en une somme de spectres plus simples pour guider la génération finale.

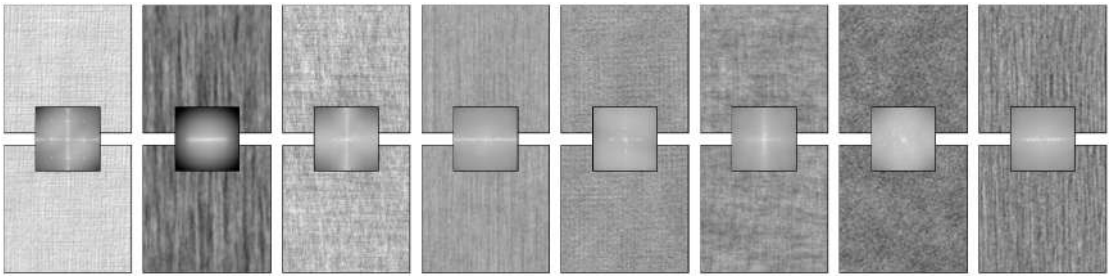


Figure 2.15 – *En haut, des textures exemples. En bas, le résultat de la génération. Les vignettes au milieu représentent les densités spectrales estimées des exemples. Image issue de [GLLD12].*

La même année, Gilet et collègues [GDG12] présentent une autre méthode de décomposition du spectre de puissance par des sous-ensembles rectangulaires. Pour cela, ils proposent l'utilisation de noyaux en sinus cardinal, qui ont une représentation spectrale idéale rectangulaire (figure 2.16). Là encore, la méthode vise à décomposer un spectre complexe pour guider les caractéristiques de noyaux en sinus cardinal.

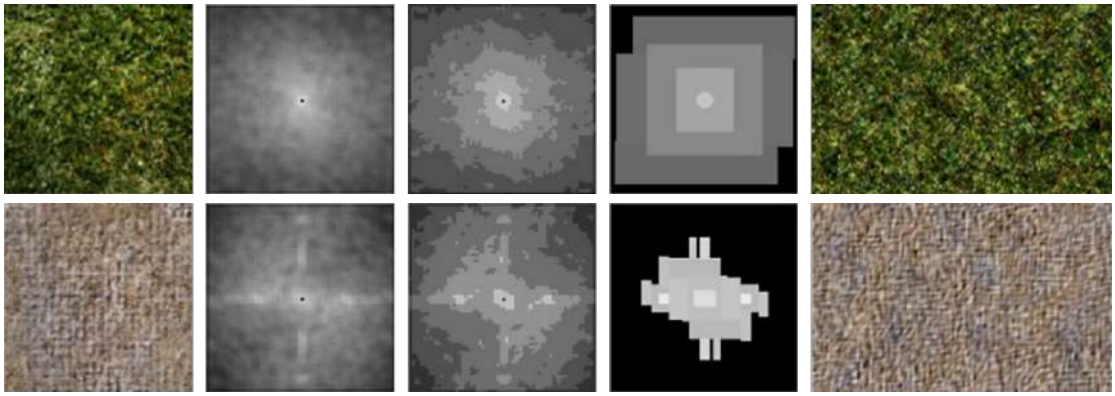


Figure 2.16 – *Pour chaque ligne, de gauche à droite : exemple d'entrée, spectre de puissance de l'exemple, niveau d'énergie du spectre, décomposition en rectangle, résultat de la génération. Images issues de [GDG12].*

Le bruit local à phase aléatoire [GSV+14] utilise un bruit local comme noyau. Comme nous l'avons vu (section 2.2.3), les caractéristiques de celui-ci se retrouvent, assez naturellement, dans le résultat final. Une autre approche est également proposée pour étendre cette méthode à de la synthèse par l'exemple. Pour cela, la PSD de l'exemple est séparée en plusieurs strates, puis les fréquences et les amplitudes des bruit locaux sont tirées aléatoirement dans ces strates (figure 2.17).

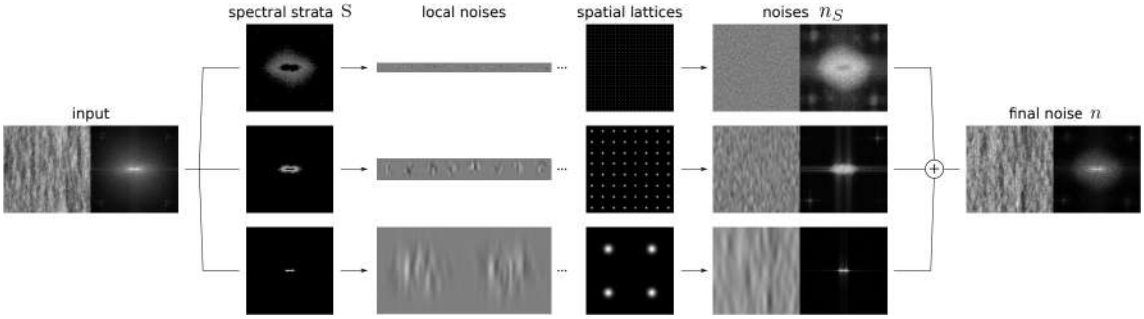


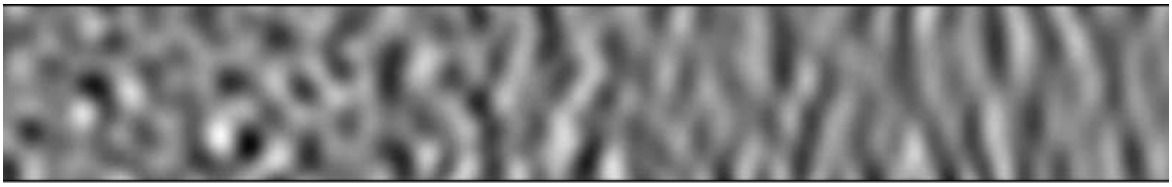
Figure 2.17 – *Décomposition du spectre d'un exemple d'entrée (à gauche) pour reproduire ses caractéristiques dans la texture de sortie (à droite). Image issue de [GSV+14].*

Parmi les méthodes par convolution parcimonieuse, celles utilisant un noyau discret reposent entièrement sur l'utilisation d'une texture exemple à partir de laquelle le noyau va être défini. Le bruit par *texton* [GLM17] est basé sur l'analyse d'une texture d'entrée pour en extraire les caractéristiques spectrales. Ces caractéristiques sont ensuite stockées dans un *texton* qui est utilisé comme noyau pour la génération du résultat. Pour le pavage et mélange [HN18], les noyaux ne sont pas créés à partir des caractéristiques spectrales de l'exemple d'entrée, mais directement tirés dans celui-ci. Le mélange est ensuite défini de façon à préserver les statistiques de premier et second ordre, et obtenir un résultat similaire.

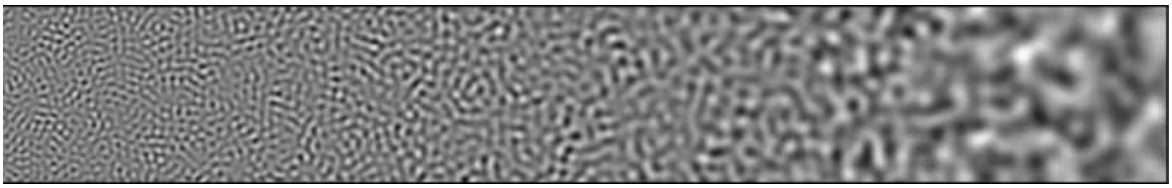
2.3.2 Contrôle de l'organisation

Jusque-là, nous n'avons parlé que de bruits dont le résultat est stationnaire et ergodique. C'est-à-dire que leurs propriétés statistiques peuvent être obtenues par l'observation d'une seule réalisation. Cependant, une texture n'est pas forcément homogène spatialement et peut présenter des variations d'orientation et de fréquence par exemple. Nous présentons ici quelques méthodes permettant de contrôler les caractéristiques d'un bruit en fonction de la position dans l'espace.

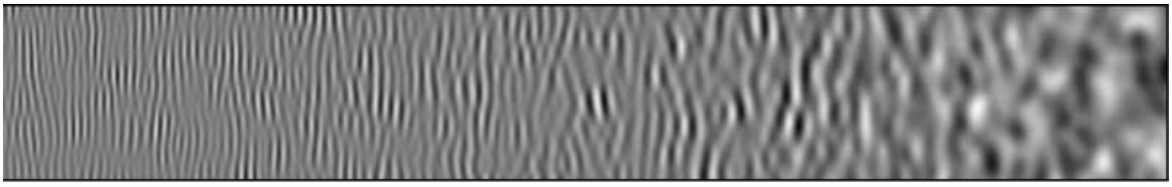
Les cartes de contrôle. Les bruits procéduraux sont générés indépendamment en tout point de l'espace. Il est donc possible de contrôler l'apparence du résultat pour chaque position sans interférer avec les autres. Comme nous l'avons vu juste avant (section 2.3.1), l'apparence d'un bruit est déterminée par son spectre. Donc, en contrôlant spatialement les caractéristiques du spectre, nous pouvons contrôler spatialement l'apparence du résultat. Ces variations spatiales du spectre peuvent être matérialisées sous la forme de cartes de contrôle. Il s'agit de champs scalaires ou vectoriels, définis sur l'espace de sortie, et qui représentent la valeur d'une ou plusieurs caractéristiques du spectre dans cet espace. L'article de Charpenay et collègues [CSM14] présente l'impact de la variation spatiale de différents paramètres du bruit par convolution de noyaux de Gabor avec une même carte de contrôle (figure 2.18).



(a) *Variation de l'anisotropie*



(b) *Variation de la fréquence (bruit isotrope)*



(c) *Variation de la fréquence (bruit anisotrope)*

Figure 2.18 – *Résultat de variations spatiales des caractéristiques du spectre appliquées sur un bruit par convolution de noyaux de Gabor. La variation de l'anisotropie est obtenue par le choix des orientations possibles pour les noyaux (voir figure 2.14), la variation de la fréquence est obtenue par la modification de la fréquence des harmoniques. Images issues de [CSM14].*

La cyclo-stationnarité est un cas particulier de l'utilisation des cartes de contrôle. Dans ce cas, les caractéristiques du spectre du résultat vont se répéter à intervalles réguliers dans l'espace. Cette répétition est obtenue par contrôle cyclo-stationnaire du spectre des noyaux. Ainsi, des motifs similaires vont apparaître à intervalles réguliers dans le résultat. Cependant, ces motifs ne seront pas identiques grâce à la convolution par le bruit blanc qui apporte l'aléa et n'est pas impacté par le contrôle du spectre des noyaux. Dans l'article de Lutz et collègues [LSD21], ce principe de cyclo-stationnarité est appliqué à différentes méthodes de génération de bruits (figure 2.19).

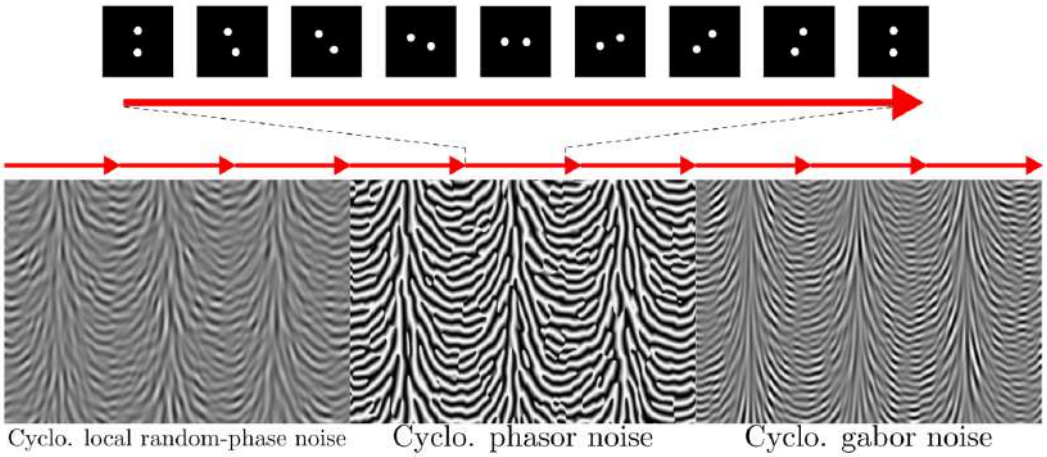


FIGURE 2.19 – *Variations cyclo-stationnaires des caractéristiques spectrales de bruits. Vignettes en haut : représentation spectrale du résultat sur un cycle. Image issue de [LSD21].*

Les textures multi-couches. Les variations spatiales au sein d'une texture peuvent également être représentées comme la superposition de différentes couches de textures. Guingo et collègues [GSDC17] ont présenté un modèle de décomposition de textures en une couche de structure et une couche de bruit (figure 2.20). L'objectif de cette décomposition est de permettre le traitement indépendant des informations de structure et de bruit contenus dans un exemple d'entrée. Ainsi, la couche de structure représente les informations de structure globale et locale. Elle contient le positionnement spatial et la forme des motifs sous-jacents de la texture, mais également leurs couleurs. La couche de bruit représente le "grain" des différents matériaux de la texture. Elle est divisée en plusieurs sous-couches se partageant l'espace. Ce partage de l'espace est réalisé au moyen de masques binaires.

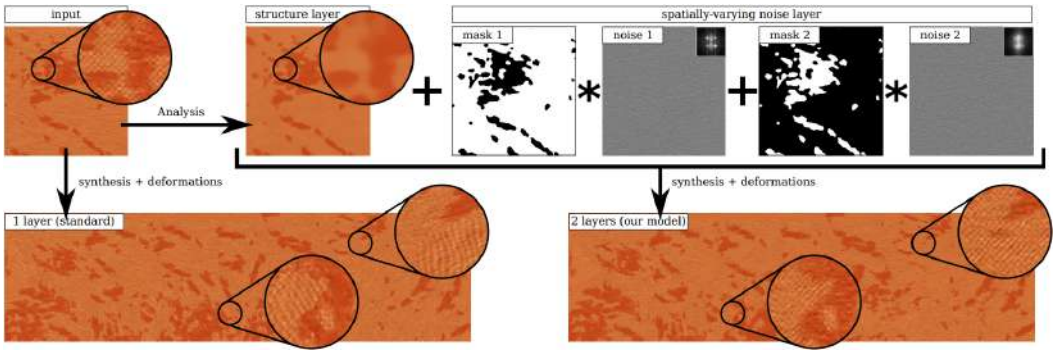


FIGURE 2.20 – *Décomposition d'un exemple d'entrée (en haut à gauche) en plusieurs couches. Cette décomposition permet d'applique des déformations sans observer d'étirement des micro-détails (en bas). Image issue de [GSDC17].*

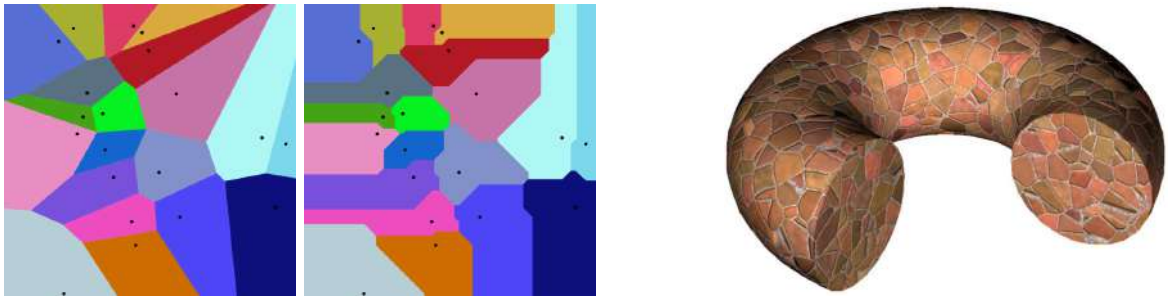
Cette séparation en différentes couches permet également de mieux représenter les déformations appliquées sur une texture. En effet, il devient possible de déformer la couche de structure indépendamment du contenu de la couche de bruit et donc de conserver l'apparence fine du matériau.

2.4 Génération procédurale de structure

La génération procédurale de bruits permet de créer des motifs aléatoires dont les caractéristiques sont contrôlables. Lors de la création d'une texture, la présence de structure est importante. Aussi bien pour représenter des éléments créés par l'homme (comme les pavés d'une route ou des taches de peinture) que des éléments naturels (comme des empilements de feuilles ou les motifs de la peau d'un animal).

2.4.1 Les processus ponctuels sans mélange

Les bruits cellulaires. Une première méthode permettant de générer des éléments structurés est l'utilisation d'un partitionnement de l'espace en cellules. À partir d'un nombre fini de graines, réparties sur la surface, ces méthodes permettent de générer des textures organiques s'approchant de la peau ou des écailles. Ainsi, les diagrammes de Voronoi associent chaque point de l'espace à la graine la plus proche suivant une mesure de distance donnée. Une même disposition de graines initiale donne différents résultats en fonction de la formule choisie pour mesurer la distance (figure 2.21a).



(a) *Diagramme de Voronoi* **(b)** *Méthode de Worley*

Figure 2.21 – *Figure 2.21a, la construction de partitions de Voronoi à partir de 20 graines. Les résultats sont différents suivant la mesure utilisée : à gauche distance euclidienne, à droite distance de Manhattan. Images issues de* `Wikipedia`[2]. *Figure 2.21b, les texture cellulaire obtenue par la méthode de Worley sont inspirées des diagrammes de Voronoi. Image issue de [Wor96].*

En 1996, Steven Worley [Wor96] présente une méthode inspirée des diagrammes de Voronoi (figure 2.21b). Celle-ci consiste à utiliser des combinaisons linéaires de la distance de chaque point aux n graines voisines les plus proches. Là aussi, l'apparence du résultat varie selon la fonction utilisée pour mesurer la distance. Plus tard, Qin et Yang [QY01] ont présenté une extension à la méthode de Worley en ajoutant des combinaisons non-linéaires de fonctions de distance. Cela leur permet d'élargir l'éventail des résultats possibles.

Le tamponnage. La synthèse par tamponnage [SA79, Pea85] est une famille de méthodes regroupant plusieurs algorithmes basés sur l'impression de noyaux K_i prédéfinit, à des positions aléatoires $\mathbf{x}_i$ (figure 2.22). Ces méthodes peuvent se rapprocher de la synthèse par convolution parcimonieuse par leur utilisation de processus ponctuels pour définir la position des noyaux. Cependant, elles ont la particularité d'utiliser un petit nombre de noyaux K, décrivant un motif élémentaire. De plus, ces noyaux sont appliqués sans poids de mélange, de façon à se recouvrir partiellement. L'ordre d'application a donc une importante, en effet, le dernier noyau appliqué masque ceux situés sous lui.
Praun et collègues [PFH00] ont présenté une méthode basée sur la définition de *texture patches* à partir d'un exemple. Ceux-ci sont ensuite répétés sur la surface d'un objet jusqu'à son recouvrement. Les *patches* ainsi utilisés peuvent être mis à l'échelle ou orientés selon un champ définit à la surface de l'objet, de façon à correspondre au mieux à la géométrie.
Dischler et collègues [DMLG02] proposent de décomposer une texture existante en un nombre fini de particules. L'organisation spatiale des particules entre-elles est faite en analysant les voisinages dans la texture étudiée. Enfin, la synthèse est faite directement sur la surface de l'objet en 3D, de façon à

2. https://en.wikipedia.org/wiki/Voronoi_diagram

[PFH00] [DMLG02] [LHN05]

FIGURE 2.22 – *Résultats de méthodes par tamponnage, images issues des articles correspondants.*

tenir compte des voisinages précédemment déterminés.
Lefebvre et collègues [LHN05] présentent en 2005 une méthode permettant d'appliquer de petits éléments, des *texture sprites*, sur la surface en utilisant une structure de données en *octree* stockée sur la carte graphique. Cette optimisation leur permet de composer une texture complexe à l'aide d'éléments atomiques dont les attributs (taille, couleur, position) peuvent être modifiés dynamiquement.
De nombreuses autres méthodes existent, étudiant, entre autres, l'importance de l'ordre des motifs appliqués, mais nous ne les développerons pas ici.

2.4.2 Adaptation de procédures de génération de bruits gaussiens

Certains algorithmes de génération procédurale de bruits peuvent être repris et adaptés pour faire apparaitre de la structure dans leur résultat.

Le bruit local à phase aléatoire. Le bruit local à phase aléatoire [GSV+14] permet également de faire apparaître de la structure dans son résultat. Cette méthode est d'abord conçue pour générer des processus gaussiens (section 2.2.3). Mais, en appliquant un contrôle sur les phases, des éléments structurés apparaissent dans le résultat. En effet, l'alignement des phases des sinusoïdes de différentes fréquences permet de fortes variations d'intensité dans le résultat.
Cependant, aligner les phases sur une portion de la texture provoque une périodicité du résultat. Il est donc nécessaire d'ajouter des déphasages aléatoires en "bloc" pour casser ces répétitions (figure 2.23).

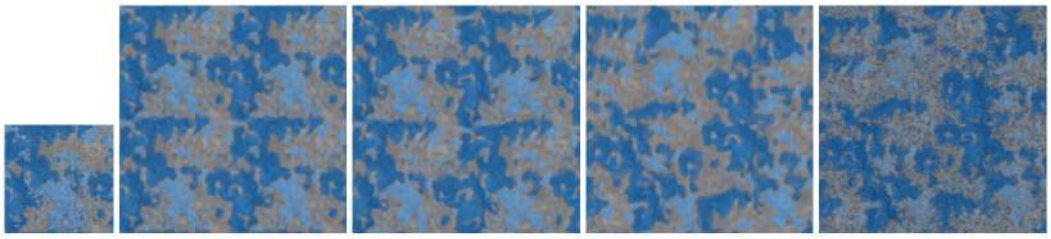


FIGURE 2.23 – *Génération d'une texture à partir d'un exemple (a) présentant des éléments structurés. Une forte périodicité est observable dans le premier résultat (b). Celle-ci est cassée par l'ajout de turbulences (c), en ajoutant de l'aléatoire dans le positionnement des noyaux (d) et en introduisant une part d'aléatoire dans les phases. Image issue de [GSV+14].*

Le pavage et mélange. L'algorithme de pavage et mélange présenté en 2018 [HN18] et dont nous avons parlé plus haut (section 2.2.3) s'applique sur un exemple d'entrée dont l'histogramme est gaussien. Cependant, les auteurs ont présenté, dans le même article, une extension à cette méthode pour l'appliquer à des exemples E non gaussiens. Pour parvenir à cela, ils réalisent un pré-calcul pour rendre l'histogramme de E gaussien (figure 2.24). Cette étape de "gaussiennisation" est réalisée

hors-ligne. L'exemple résultant, E_G, est stocké en mémoire, ainsi que l'opérateur de transformation $\mathcal{T}$ tel que $\mathcal{T}(E) = E_G$. Puis la génération par pavage et mélange est faite en temps réel sur E_G. Enfin, la transformation de l'histogramme est inversée grâce aux informations sur $\mathcal{T}$ stockées en amont.

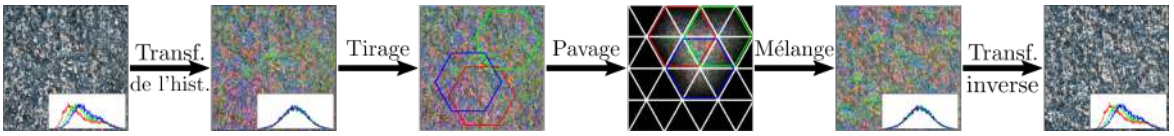


Figure 2.24 – *Étape de transformation de l'histogramme de l'image d'entrée. La première étape de transformation est pré-calculée pour pouvoir appliquer le pavage et le mélange sur une texture dont l'histogramme est gaussien. Image issue de [HN18].*

Une reprise de cette méthode a été présentée par Burley [Bur19]. Afin d'améliorer le résultat du pavage et mélange, il met en place une exponentiation des poids de mélange w_i, pour mieux préserver les détails structurés de la texture. Pour éviter les anomalies de couleurs dans le résultat, il propose également de réaliser le mélange dans l'espace de couleur `YCbCr` (Luminance, Chrominance) plutôt que dans l'espace `RGB` (Rouge, Vert, Bleu), et de concentrer la préservation de l'histogramme sur la luminance plutôt que sur les couleurs. Cette méthode reste cependant limitée pour reproduire des motifs trop structurés ou répétitifs (figure 2.25).

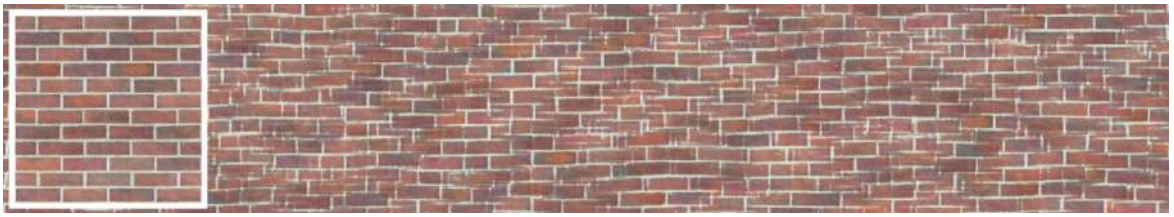


Figure 2.25 – *Limitation du pavage et mélange pour les textures présentant des répétitions de structures. Image issue de [DH19].*

Le pavage hexagonal. En 2022, Mikkelsen propose une modification du pavage et mélange qu'il appelle *Hex-tiling* [Mik22]. Cette méthode se base toujours sur le tirage de tuiles hexagonales dans un exemple d'entrée et leur mélange sur les sommets d'une grille régulière de triangles. Cependant, Mikkelsen applique une modification sur les poids de mélange entre les tuiles en fonction du contenu de celles-ci. Ainsi, pour générer une texture à partir d'un exemple E, les poids de mélange w_i sont calculés de façon à favoriser les hautes luminances. Cette méthode de mélange permet l'utilisation du pavage et mélange sur un exemple non gaussien sans passer par la phase de pré-calcul (figure 2.26). Cependant, elle ne garantit pas la préservation des statistiques de l'exemple d'entrée, en particulier sur les zones de mélange entre les tuiles.

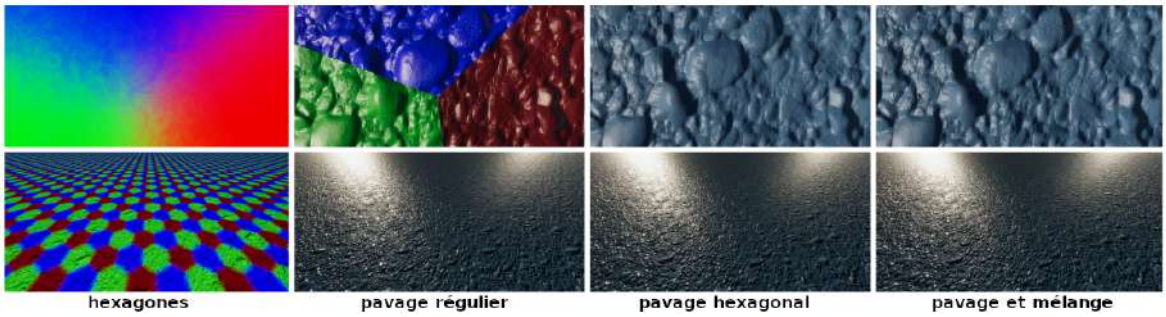


Figure 2.26 – *Le pavage hexagonal permet d'obtenir des résultats équivalent à ceux du pavage et mélange, mais sans réaliser de modification de l'histogramme en amont de la génération. Image issue de [Mik22].*

La cyclo-stationnarité. Malgré l'adaptation des poids de mélange, la méthode du pavage et mélange ne permet pas de reproduire de motifs présentant des répétitions régulières. En 2021, Lutz et collègues [LSD21] ont proposé une autre amélioration permettant la génération de motifs avec une régularité cyclique, tels que les carrelages. L'idée de cette méthode est d'impacter les tirages dans l'exemple d'entrée E de façon à coller à une grille représentant le motif à répéter (figure 2.27).

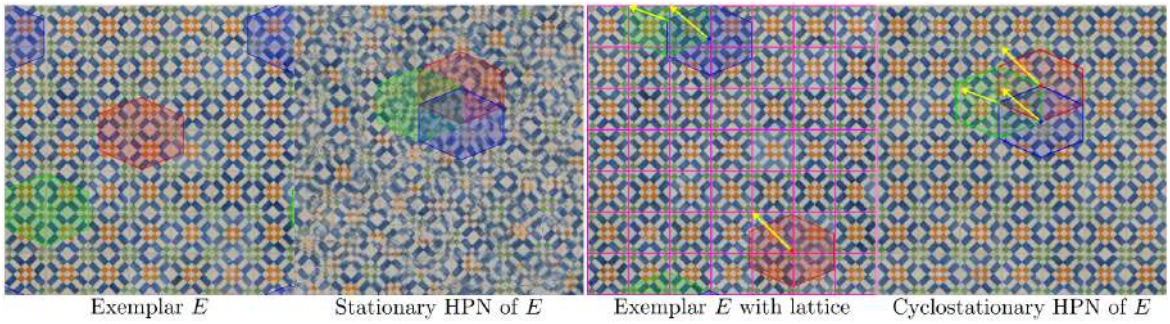


Figure 2.27 – *À gauche, le tirage stationnaire dans l'exemple E ne permet pas de reproduire les motifs présents. À droite, un tirage cyclo-stationnaire permet de retrouver ces structures dans le résultat. Image issue de [LSD21].*

Cela permet de générer des textures présentant une structure cyclo-stationnaire, comme des murs de briques, mais pas de reproduire des motifs aléatoires structurés.

La préservation de l'auto-covariance. Dernièrement, Luzt et collègues [LSD23] ont proposé une nouvelle amélioration des méthodes de génération par pavage pour permettre la préservation de l'auto-covariance de la texture d'exemple. Leur méthode s'appuie sur l'utilisation d'un échantillonnage préférentiel lors du tirage des tuiles de l'algorithme du pavage et mélange [HN18]. L'objectif de cette modification du tirage est de conserver au mieux la PSD de la texture d'exemple (figure 2.28). Cela permet de générer des textures dont le contenu est caractérisé pas une forte auto-covariance.

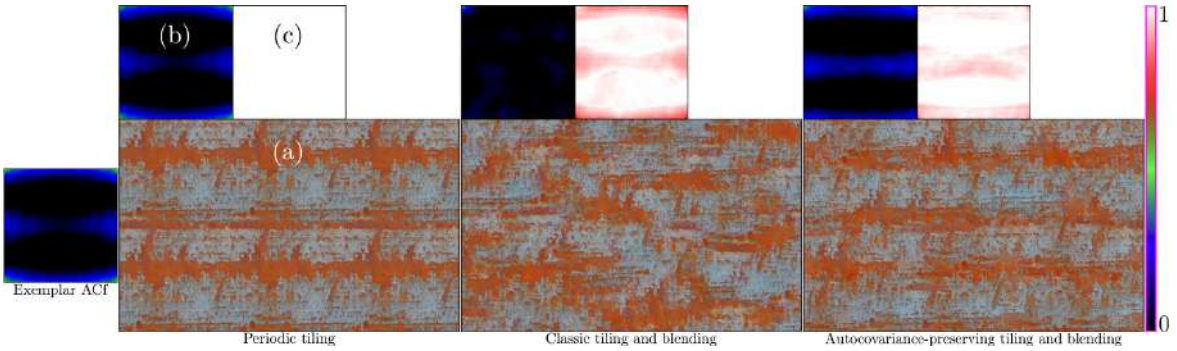


Figure 2.28 – *À gauche, le pavage périodique préserve bien l'auto-covariance, mais fait apparaitre des répétitions. Au centre, le pavage et mélange [HN18] crée de la variété par le tirage aléatoire des tuiles dans l'exemple. À droite, le tirage basé sur un échantillonnage préférentiel permet de mieux préserver l'auto-covariance de l'exemple. Pour chaque vue : (a) résultat de la synthèse, (b) auto-covariance du résultat, (c) similarité avec l'auto-covariance de l'exemple. Image issue de [LSD23].*

2.4.3 Les fonctions de transfert

D'autres méthodes visant à générer de la structure proposent plutôt d'utiliser les bruits procéduraux comme source d'aléatoire et de leur appliquer des fonctions de transfert apportant la structure.

Le bruit à crête. Une première fonction de transfert simple et très utilisée pour la génération de structure est la valeur absolue. Elle permet de faire apparaître des discontinuités à partir d'un

champ gaussien sans perdre les avantages de la génération procédurale de ce champ. Cette fonction de transfert est souvent utilisée sur des bruits de type mouvement Brownien fractionnaire (ou fBm, de l'anglais *Fractional Brownian motion*). Ceux-ci ont été introduits en 1940 par le mathématicien Andreï Nikolaïevitch Kolmogorov [Kol40] puis étudiés, entre autres, par Norbert Wiener et Benoit Mandelbrot [MVN68]. Ils ont la particularité de présenter une propriété d'auto-similarité. C'est-à-dire que, lors d'un zoom avant dans une réalisation du bruit, les caractéristiques spectrales restent les mêmes (figure 2.29).

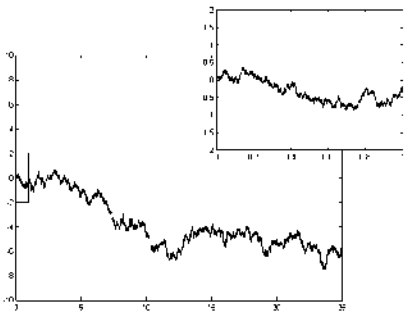


Figure 2.29 – *Réalisation d'un processus de Wiener à une dimension. L'encadré à droite montre un zoom sur le processus pour des valeurs entre* 0 *et* 1. *Image issue de* `Wikipedia`[3].

En informatique graphique, cette propriété est obtenue en sommant plusieurs réalisations d'un bruit, chacune de fréquence croissante et d'amplitude décroissante par rapport à la précédente (figure 2.30). Cela permet de faire apparaître des éléments haute fréquence.

Figure 2.30 – *Le bruit fractal (à droite) est composé d'une somme de réalisations d'un même bruit à des fréquences croissantes et des amplitudes décroissantes.*

L'application d'une valeur absolue sur un bruit fBm permet d'obtenir un bruit à crête (ou *ridge noise* en anglais) et est utilisée, entre autres, pour la génération de crêtes sur des terrains virtuels (figure 2.31).

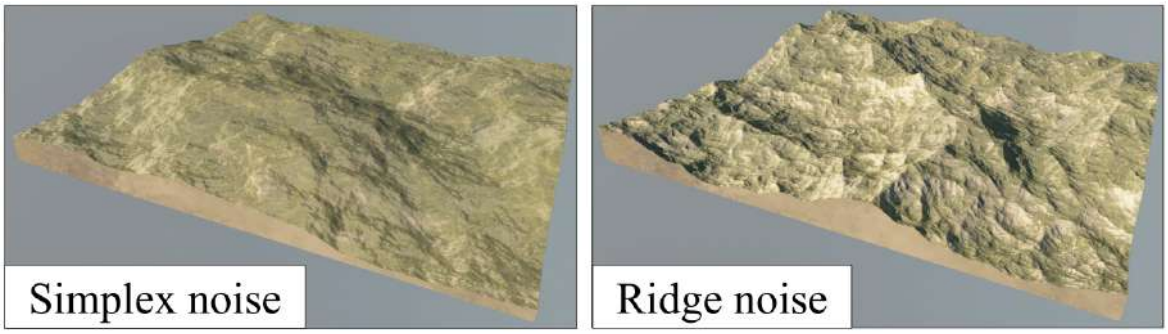
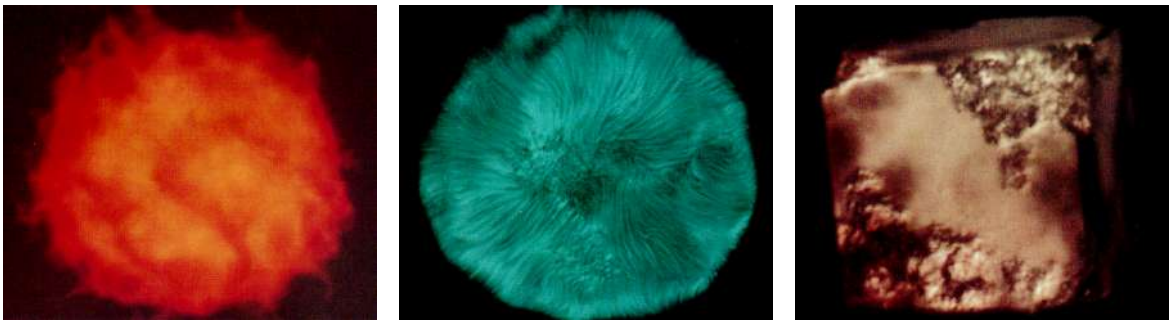


Figure 2.31 – *À gauche, un terrain généré en utilisant un simplex noise (dérivé du bruit de Perlin). À droite, un terrain généré en utilisant un bruit à crête. Image issue de [GGP+19].*

3. https://en.wikipedia.org/wiki/Wiener_process

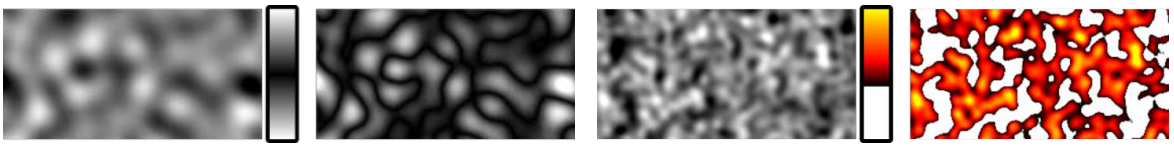
Les hypertextures. Perlin et collègues [PH89] ont présenté une généralisation des textures afin d'y ajouter une information de géométrie. Cette généralisation, appelée "hypertexture", tend à représenter des phénomènes comme la fourrure, le feu ou l'érosion. Pour cela, les auteurs utilisent des fonctions de transferts appliquées sur un bruit procédural. Par exemple, la fonction de turbulence, basée sur le bruit à crête, permet de reproduire un effet de flamme (figure 2.32a). Pour créer un effet de fourrure, une fonction de projection est utilisée (figure 2.32b), et pour l'érosion, un opérateur d'intersection entre la géométrie et un bruit fractal (figure 2.32c).



(a) *Feu* **(b)** *Fourrure* **(c)** *Érosion*

FIGURE 2.32 – *Représentation de phénomènes par des hypertextures. Images issues de [PH89].*

Les cartes de couleurs. L'utilisation d'une fonction de transfert, telles que la valeur absolue, pour obtenir de la structure à partir d'un champ gaussien, peut être généralisée par les cartes de couleurs (figure 2.33). Dans ce cas, les valeurs d'intensité du champ gaussien généré sont utilisées comme données d'entrée pour une fonction quelconque, stockée dans une image appelée carte de couleurs.



(a) *Bruit à crête* **(b)** *Carte quelconque*

FIGURE 2.33 – *Figure 2.33a, représentation du bruit à crête sous la forme d'une carte de couleurs. Figure 2.33b, utilisation d'une carte de couleurs quelconque.*

De cette façon, le champ gaussien apporte la partie stochastique et la présence de discontinuités dans la carte de couleurs fait apparaitre de la structure dans le résultat. Ainsi, dans la figure 2.33b, le passage discontinu de noir à blanc au centre de la carte fait apparaître des éléments structurés dans le résultat final.
Cependant, ces cartes de couleurs sont limitées dans la variété de motifs qu'elles permettent de produire. En effet, La couleur est appliquée uniquement selon l'intensité du champ scalaire utilisé en entrée. Cela induit que les couleurs dans le résultat sont appliquées de façon concentrique. Cette limitation est observable dans le résultat de la figure 2.33b (à droite). Dans cette texture, les régions blanches ne peuvent être entourées que d'un liseré noir et ne peuvent pas être voisines des zones jaunes, par construction de la carte utilisée.

Le bruit de vecteurs de phase. Une méthode, baptisée bruit *phasor*, a été proposée en 2019 par Tricard et collègues [TEZ+19] pour obtenir des éléments structurés à partir d'un champ gaussien. Cette méthode se base sur la représentation des fonctions sinusoïdales par leur phaseur (ou vecteur de phase, de l'anglais *phasor*, contraction de *phase vector*). La représentation par phaseur utilise la formule d'Euler, indiquant que les ondes sinusoïdales peuvent être mathématiquement représentées

comme la partie réelle d'un complexe.

$$A\cos(2\pi f\theta\mathbf{x}) = \Re\left(A\exp^{(i(2\pi f\theta\mathbf{x}))}\right) \tag{2.16}$$

Le phaseur $\phi(\mathbf{x})$ désigne le nombre complexe $A\exp^{(i(2\pi f\theta\mathbf{x}))}$.
Soit le phaseur $\phi_j(\mathbf{x})$, définit pour toute position $\mathbf{x}$. Il a pour module $A_j(\mathbf{x})$ et pour argument $\varphi_j(\mathbf{x}) = 2\pi f_j\theta_j\mathbf{x}$. C'est-à-dire :

$$\phi_j(\mathbf{x}) = A_j(\mathbf{x})\exp^{(i\varphi_j(\mathbf{x}))} \tag{2.17}$$

La somme de plusieurs phaseurs est un phaseur. Son module est alors la racine de la somme des carrés des parties réelles et imaginaires. Son argument est l'angle polaire entre la somme des parties réelles et la somme des parties imaginaires.
Nous pouvons donc écrire la représentation complexe d'un bruit gaussien généré par convolution de noyaux de Gabor, comme la somme des phaseurs des noyaux :

$$\begin{aligned}\widetilde{N}(\mathbf{u}) &= A(\mathbf{u})\ \exp^{(i\Phi(\mathbf{u}))} \\ &= \sum_j \phi_j(\mathbf{u}-\mathbf{x}_j)\end{aligned} \tag{2.18}$$

Avec le champ de module $A(\mathbf{u})$

$$A(\mathbf{u}) = \sqrt{\left(\sum_j \Re\left(\phi_j(\mathbf{u})\right)\right)^2 + \left(\sum_j \Im\left(\phi_j(\mathbf{u})\right)\right)^2} \tag{2.19}$$

et le champ d'argument $\Phi(\mathbf{u})$

$$\Phi(\mathbf{u}) = \mathrm{Atan2}\left(\sum_j \Im\left(\phi_j(\mathbf{u})\right), \sum_j \Re\left(\phi_j(\mathbf{u})\right)\right) \tag{2.20}$$

Le bruit de vecteurs de phase (ou bruit *phasor*) utilise le champ d'arguments $\Phi(\mathbf{u})$ pour générer un champ de vagues sur lequel est appliqué un profil γ (figure 2.34).

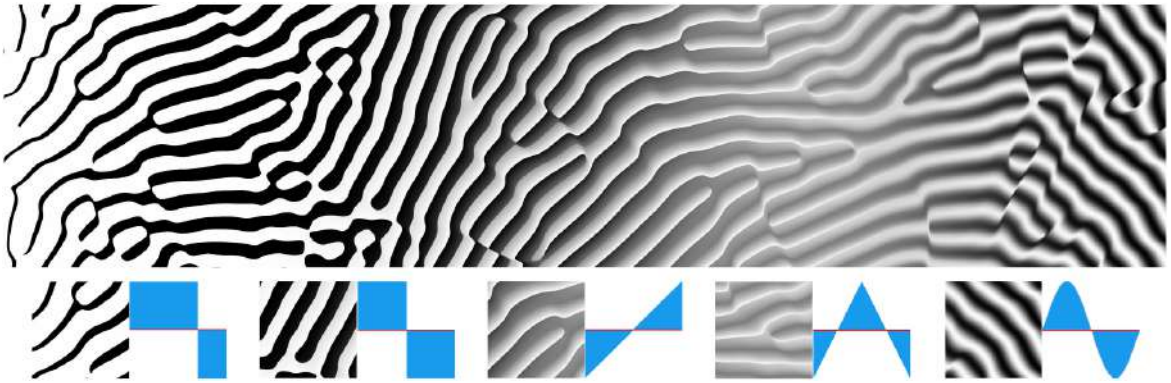
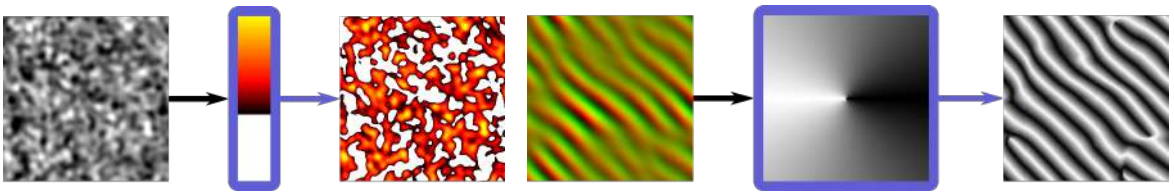


Figure 2.34 – *Résultat de vagues profilées générées par le bruit phasor. Les vignettes en bas représentent le profil γ utilisé pour chaque portion du résultat. Image issue de [TEZ+19].*

L'avantage de cette méthode est qu'elle permet de séparer la partie stochastique de la création de structure. En effet, la première est obtenue par le champ gaussien généré par convolution de noyaux de Gabor. Alors que la seconde est obtenue par l'utilisation de la fonction Atan2 et contrôlée par la fonction de profil γ. Le bruit *phasor* est donc obtenu par l'application d'une fonction de transfert de $\mathbb{C}$ dans $\mathbb{R}$ à un champ gaussien complexe. Cette fonction peut également être écrite sous la forme d'une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$ en utilisant un champ gaussien vectoriel. Cette formulation est développée plus loin dans ce manuscrit dans les chapitres 3 et 4.

2.4.4 Les graphes des textures

Généralisation des fonctions de transfert. Le raisonnement par composition de fonctions est intrinsèquement présent dans les graphes de textures (figure 2.35). En effet, leur principe de fonctionnement est de définir des opérations entre des champs donnés en entrée et dont les résultats sont ensuite utilisés comme entrées pour d'autres opérations. Les méthodes présentées juste avant (section 2.4.3) peuvent donc être regroupées en une seule méthode générale par leur représentation en graphe. Ainsi, les fonctions décrivant la couleur, pour les cartes de couleurs, ou le calcul de l'argument, pour le bruit *phasor*, peuvent être écrites sous la forme de nœuds d'opération dans un graphe.



(a) *Carte de couleur 1D* **(b)** *Carte de couleur 2D*

Figure 2.35 – *Représentation des fonctions de transfert précédentes sous forme de composition dans un graphe de texture. Dans la figure 2.35a la composition par une fonction de transfert 1D (cerclée en bleu) permet de retrouver la composition par une carte de couleurs. Dans la figure 2.35b, la composition d'un champ vectoriel par une fonction de transfert 2D (cerclée en bleu) permet de reproduire le bruit phasor.*

Cette représentation en graphe nous permet de formaliser une méthode générale pour la génération de structure procédurale (chapitre 3). Cette formalisation représente la base des contributions présentées dans la suite de ce manuscrit.

Autres nœuds d'opération. Les graphes de textures permettent de représenter tous les algorithmes de génération au moyen de nœuds d'opérations. D'autres opérations que les fonctions de transfert permettent de générer de la structure localement (comme l'utilisation locale de matrices de transformation par exemple). Dans cette thèse, nous nous intéressons aux possibilités offertes par les fonctions de transfert bi-dimensionnelles. Celles-ci permettent de modéliser un grand nombre d'opérations et présentent donc un potentiel intéressant.

2.5 Le rendu

Le rendu consiste à calculer une image d'une scène par une caméra virtuelle. Plusieurs algorithmes existent pour cela, présentant chacun leurs avantages et limitations. Un algorithme va donc être préféré aux autres selon les contraintes définies pour le résultat (rendu en temps réel ou non, photo-réaliste ou non, ...). Dans cette thèse, nous ne nous intéressons pas aux algorithmes de rendu en eux-mêmes, mais seulement aux contraintes d'informations qu'une texture doit contenir pour être utilisée par ces algorithmes. Nous présentons donc ici deux algorithmes de rendu connu, sans entrer dans les détails, de façon à introduire la raison de ces contraintes.

2.5.1 Les méthodes de rendu

Le lancer de rayons. Cet algorithme est plutôt utilisé dans le cas où l'objectif est d'obtenir des résultats de haute qualité visuelle (tel que des rendus photo-réalistes) [PJH23]. Le lancer de rayons s'applique dans l'espace de l'image résultat. Pour chaque *pixel* de cette image, un ou plusieurs rayons sont lancés depuis la caméra, vers la scène virtuelle. L'algorithme teste ensuite l'intersection entre le rayon et les objets de la scène pour déterminer quelle portion des objets de la scène est visible dans ce *pixel*. Des rayons sont ensuite lancés en direction des sources lumineuses, pour déterminer si les objets trouvés sont éclairés ou non. Cette méthode peut être complexifiée afin de pouvoir rendre des effets de lumière plus complexes (comme la réflexion et la réfraction dans des objets transparents) ou les inter-réflexions entre objets (illumination globale).

Lorsqu'il est lancé vers la scène virtuelle, un rayon n'est pas d'épaisseur nulle. En effet, il couvre un *pixel* complet ou une partie d'aire non nulle. Ainsi, l'intersection avec la scène virtuelle ne se fait pas en un point précis, mais sur une empreinte $\mathcal{P}$. Cette empreinte peut avoir une taille et une forme arbitraire, qui est fonction de la position des objets dans la scène.

La rastérisation. Cet algorithme est plutôt utilisé pour obtenir le résultat du rendu en temps réel. La rastérisation s'applique dans l'espace des objets de la scène virtuelle. Pour chaque objet, les sommets constituant le maillage sont projetés dans le plan représentant l'image résultant du rendu, avec leurs informations de texture et de connexité. L'information de connexité est utilisée pour retrouver les arêtes reliant les sommets. Le plan de l'image est ensuite discrétisé en un nombre fini de *pixels*. La couleur de chaque *pixel* est déterminée en fonction des informations de texture des éléments présents.
Lors de la projection puis de la discrétisation de l'espace image, plusieurs objets de la scène virtuelle peuvent être présents dans le même *pixel* final. Cependant, un *pixel* ne peut être que d'une seule couleur. Il convient donc de déterminer quelle est la couleur à afficher sur toute la surface $\mathcal{P}$ du *pixel*.

La micro-géométrie. Les algorithmes de rendu, tels que le lancer de rayons ou la rastérisation, calculent les interactions lumineuses et l'éclairage global de la scène. À une échelle plus petite, c'est la micro-géométrie de la surface des objets qui détermine l'interaction locale avec la lumière et donc la couleur (figure 2.36). Ces interactions lumineuses locales dépendent d'un grand nombre de paramètres

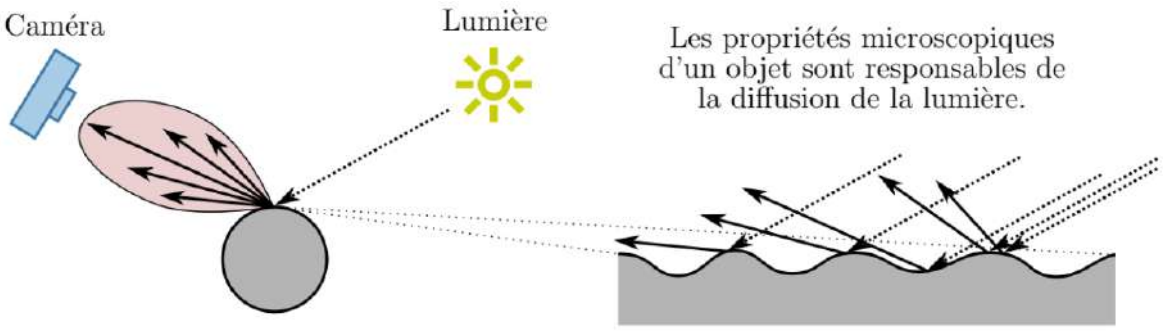


Figure 2.36 – *La micro-géométrie de la surface (à droite) impacte l'interaction lumineuse et détermine sa couleur. Image issue de [Che19].*

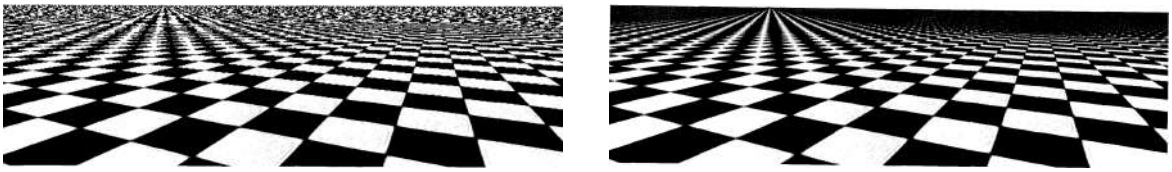
dont la direction d'éclairage, la direction d'observation et la normale locale de la surface. Les fonctions bidirectionnelles de distribution de la réflectance ou BRDF (de l'anglais *Bidirectional Reflectance Distribution Function*) sont une représentation de l'apparence d'une surface en fonction des directions d'observation et d'éclairage [Gla14].

La problématique du filtrage. Comme nous l'avons vu juste avant, un *pixel* de l'image finale peut contenir une large portion de la scène 3D à rendre (en fonction de la position des objets par rapport à la caméra). Cependant, chaque *pixel* ne pouvant afficher qu'une seule couleur, il est nécessaire de connaitre la couleur moyenne sur la surface de l'empreinte $\mathcal{P}$ du *pixel* dans la scène. Ce calcul de moyenne est le filtrage. Dans le cas d'un rendu en temps réel, le défi est de calculer cette couleur moyenne à coût constant quelle que soit l'empreinte $\mathcal{P}$.
Dans le cadre de cette thèse, nous nous intéressons principalement au filtrage de textures ne contenant que des informations de couleur. Nous ne traitons donc pas les problématiques liées à l'ajout de géométrie sur la surface, telles que les problématiques de masquage ou d'ombrage.

2.5.2 Les méthodes de filtrage

De nombreuses méthodes de filtrage peuvent être utilisées [BN11]. Le choix de la méthode la plus appropriée doit être fait en fonction de la nature de la texture et des caractéristiques connues en amont. Dans cette section, nous présentons quelques méthodes connues.

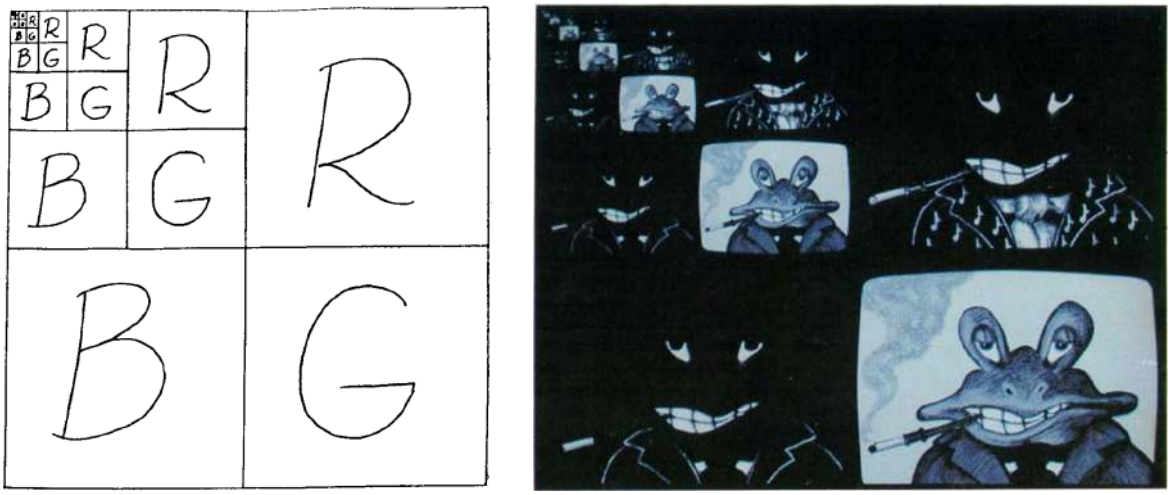
La limitation des fréquences. Une des toutes premières méthodes de filtrage de bruits procéduraux consiste à limiter les fréquences utilisées en fonction de la position dans l'espace 3D de la scène virtuelle [NRS82]. Cette méthode s'inspire du théorème d'échantillonnage de Shannon qui stipule qu'un signal peut être reconstruit précisément à partir d'échantillons si la fréquence de prélèvement de ceux-ci est plus de deux fois supérieurs à la fréquence la plus élevée du signal. La réciproque de ce théorème indique donc que, pour une fréquence d'échantillonnage fixe, le signal qu'on souhaite rendre doit avoir une fréquence maximale au moins deux fois plus petite que la fréquence d'échantillonnage pour être reconstitué précisément. L'idée de cette méthode de filtrage est donc de travailler sur la représentation spectrale de la texture à rendre et de couper les fréquences trop élevées (figure 2.37).



(a) *Rendu sans limitation des fréquences* **(b)** *Rendu avec limitation des fréquences*

FIGURE 2.37 – *Rendu d'un motif de damier sans et avec une limitation de fréquences. Image issue de [NRS82].*

Les cartes MIP . Du latin *Multum In Parvo*, qui signifie "beaucoup de choses dans un petit endroit". Cette méthode de filtrage a été présentée en 1983 par Williams [Wil83]. Son idée est de pré-calculer et de stocker en mémoire un nombre fini de niveaux de détail d'une texture donnée (figure 2.38). Les niveaux intermédiaires sont ensuite calculés par interpolation entre les niveaux connus. Les

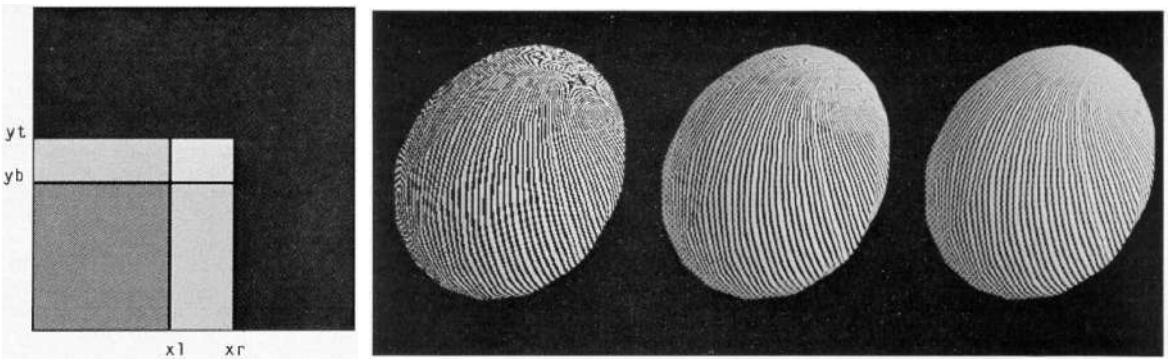


(a) *Structure de la carte MIP* **(b)** *Exemple de carte MIP*

FIGURE 2.38 – *Description de la carte MIP . Les trois canaux (Rouge, Vert, Bleu) de l'image d'entrée sont considérés et stockés indépendamment dans la carte. Images issues de [Wil83].*

limitations de cette méthode sont qu'elle nécessite de connaître entièrement une version discrète de la texture en amont, ce qui n'est pas le cas pour les textures générées procéduralement. De plus, elle utilise une place conséquente en mémoire, ce qui est problématique dans le cas de l'utilisation de nombreuses textures de grande taille. Cette méthode est cependant encore utilisée de façon adaptée pour certaines méthodes, telles que le pavage et mélange que nous avons vu plus haut (section 2.2.3) et qui se repose sur l'utilisation d'une texture exemple de petite taille et entièrement connue en amont.

Les tables d'aire totale. En 1984, Crow [Cro84] propose une alternative aux cartes MIP , les tables d'aire totale, ou SAT (contraction de l'anglais *Summed-Area Table*). Mathématiquement, le calcul de la valeur moyenne d'un signal est équivalent au calcul de l'aire sous la courbe de ce signal. De plus, par relation de Chasles, l'intégrale $\int_a^b$ entre deux bornes a et b est égale l'intégrale entre une troisième borne c et b moins l'intégrale entre c et a ($\int_a^b = \int_c^b - \int_c^a$). L'idée de cette méthode est donc, pour chaque coordonnée $[i, j]$ dans un tableau, de stocker l'intégrale du signal entre $[0, 0]$ et $[i, j]$. L'aire sous le signal pour des bornes quelconques est ensuite calculée par relation de Chasles. Ainsi, sur la figure 2.39a, nommons T la table des aires. L'aire sous la courbe du signal définit sur le rectangle le plus clair est égale à $T[x_r, y_t] - T[x_l, y_t] - T[x_r, y_b] + T[x_l, y_b]$.



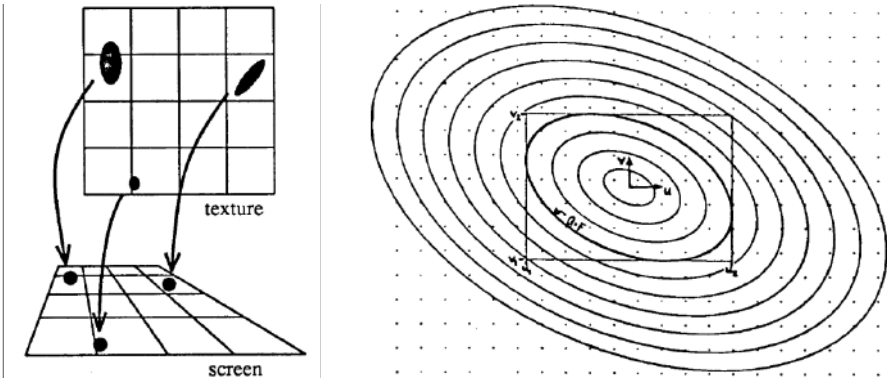
(a) *Structure de la table T* **(b)** *Résultat du filtrage*

FIGURE 2.39 – *Utilisation d'un tableau d'aire totale pour le filtrage. Dans la figure 2.39b, à gauche : sans filtrage, au milieu : filtrage par carte MIP , à droite : utilisation d'un tableau stockant les valeurs d'aire sous la courbe du signal. Images issues de [Cro84].*

Cette méthode a l'avantage de permettre l'évaluation de la moyenne pour n'importe quelle taille d'empreinte en interrogeant seulement quatre positions dans la table des aires. Elle nécessite également moins de mémoire que la méthode par cartes MIP . En effet, les cartes MIP nécessitent de stocker la texture dans sa taille initiale et ses niveaux de filtrage, alors que la table des aires peut être stockée seule, la texture initiale pouvant être retrouvée grâce à elle. Cependant, elle nécessite toujours de connaître une version discrète de la texture à rendre en amont. Dans le cas des textures générées procéduralement, elle est donc limitée à celle dont on connaît les informations avant la génération. De plus, les surfaces sur lesquelles il est possible de connaître l'aire sous la courbe du signal sont obligatoirement des rectangles dont les côtés sont alignés sur les axes de la texture. Or, dans le rendu d'une scène, les textures sont appliquées sur des surfaces quelconques et les empreintes de *pixel* ne sont jamais parfaitement rectangulaires, cela entraîne donc des approximations lors de l'utilisation de cette méthode.

Le filtrage elliptique pondéré. En 1986, Greene et Heckbert [GH86] présentent une méthode de filtrage utilisant des ellipses pondérées que Heckbert reprend et affine dans son manuscrit de thèse en 1989 [Hec89]. Les travaux de thèse de Heckbert se concentrent beaucoup sur l'impact des déformations sur le rendu des textures. Cette méthode de filtrage se base donc sur une fonction biquadratique $Q(x, y) = Ax^2 + Bxy + Cy^2$ permettant d'approcher la forme d'une gaussienne à deux dimensions (figure 2.40). L'avantage de cette formulation est qu'elle permet de déformer le filtre Q pour s'approcher au mieux des déformations imposées à la texture lors de l'application sur des surfaces quelconques. Le filtrage est ensuite réalisé par convolution de la texture avec le filtre elliptique.

Le filtrage de la géométrie. Le filtrage de la géométrie ou de la micro-géométrie présente des difficultés supplémentaires par rapport au filtrage de couleurs. Ainsi, des zones de masquage et des ombres supplémentaires apparaissent et impactent la couleur moyenne à afficher. De plus, la moyenne des normales à une surface ne peut pas être faite de la même façon que la moyenne des couleurs. En effet, en ne considérant que la normale moyenne, un matériau rugueux va prendre une apparence très lisse, la répartition des normales autour de la normale moyenne doit donc également être considérée.



(a) *Déformations des empreintes* **(b)** *Représentation du filtre elliptique*

FIGURE 2.40 – *Représentation elliptique des empreintes de pixel. Figure 2.40a la texture observée dans l'espace écran (en bas) est déformée par rapport à l'espace texture (en haut) et des empreintes qui apparaissent rondes sont en réalité elliptiques. Figure 2.40b, Représentation du filtre Q. Cette présentation permet une meilleure estimation de la moyenne que ne le ferait le calcul sur la boite englobante (rectangle sur l'image). Images issues de [Hec89].*

En 2010, Olano et Baker [OB10] présentent une méthode de filtrage des cartes de normales, basée sur l'estimation de la moyenne et de la variance des directions des normales (figure 2.41). En considérant une répartition gaussienne, la connaissance de la moyenne et de la variance leur permet de connaître la répartition. La répartition "moyenne" dans de grandes empreintes est ensuite estimée en combinant les répartitions dans des empreintes plus petites.

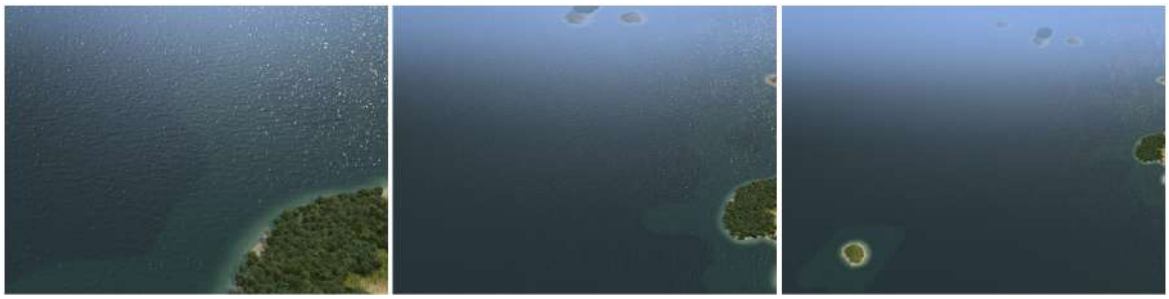


FIGURE 2.41 – *Filtrage des normales à un océan. Image issue de [OB10].*

En 2013, Dupuy et collègues [DHI+13] proposent une méthode de filtrage des cartes de déplacement et de réflectance inspirée de la méthode de Olano et Baker (figure 2.42). Pour cela, ils utilisent le pré-calcul des moments d'ordre 1 et 2 de la dérivée de la carte de déplacement.

FIGURE 2.42 – *Filtrage des écailles d'un dinosaure (modèle 3D ©Disney). Image issue de [DHI+13].*

2.5.3 Le cas des compositions

Dans le cadre de cette thèse, nous nous intéressons à des textures obtenues par la composition d'un champ vectoriel par une fonction de transfert (section 2.4.3). Dans cette section, nous nous attardons sur deux méthodes qui ont été proposées pour filtrer spécifiquement les textures obtenues par ce type de composition.

Fréquence de Nyquist. En 2006, Bergner et collègues [BMWM06] ont présenté une étude sur l'analyse spectrale du résultat d'une composition de fonctions. Leur objectif est d'estimer la fréquence nécessaire à l'échantillonnage d'une composition. Pour cela, ils cherchent à étudier la fréquence de Nyquist du résultat de la composition en fonction des fréquences de Nyquist des fonctions utilisées.

Pré-filtrage de la fonction de transfert. En 2013 et 2014, Heitz et collègues [HNPN13, HNPN14] ont présenté une méthode de filtrage du résultat de la composition d'un champ gaussien par une fonction de transfert quelconque. Leurs travaux étudient le filtrage dans les cas où la couleur du résultat est corrélée à la micro-géométrie (hauteur ou orientation) de la surface et dans le cas où aucune corrélation n'existe. Dans le cadre de cette thèse, nous nous limitons au cas où aucune corrélation entre la couleur et la géométrie n'est présente.
Dans chacun des cas, leur méthode se base sur l'observation que la couleur moyenne sur une empreinte est égale à la moyenne de toutes les couleurs possibles, chacune pondérée par sa présence dans l'empreinte. Or, dans le cas de la composition d'un champ gaussien par une fonction quelconque, les couleurs possibles sont entièrement décrites par la fonction de transfert et les quantités de présence dépendent uniquement du champ gaussien.
Cette méthode de filtrage est la principale méthode que nous utilisons pour le rendu en temps réel des motifs structurés que nous cherchons à générer. Son application mathématique est présentée dans la section 3.2.2 dans le chapitre suivant.

2.6 Résumé et positionnement

2.6.1 La génération procédurale

Comme nous l'avons vu au cours de cet état de l'art, les bruits procéduraux présentent de nombreux avantages pour la création de textures. En effet, ils permettent une génération et un filtrage en temps réel. De plus, ils introduisent une part d'aléatoire qui rend les résultats plus proches de textures naturelles. Ils sont également assez faciles à contrôler. Leur apparence peut être contrôlée par les caractéristiques de leur spectre et les méthodes "par l'exemple" permettent de définir ce spectre de façon intuitive. Les variations spatiales de l'apparence peuvent également être contrôlées, en faisant varier spatialement les caractéristiques spectrales ou en définissant plusieurs couches de bruit, par exemple. Cependant, ces bruits gaussiens produisent uniquement des motifs peu contrastés et ne permettent pas de générer des textures structurées.

2.6.2 La création de structure

Plusieurs familles de méthodes existent pour générer des motifs structurés. Les méthodes par processus ponctuels sans mélange comme les bruits cellulaires ou les méthodes par tampons parviennent à créer des textures très structurées. Cependant, les bruits cellulaires sont limités à la création de motifs similaires à des cellules. De leur côté, les méthodes par tampons, sont basées sur l'utilisation d'un petit nombre de motifs. Elles sont donc limitées dans leurs résultats ou deviennent coûteuses en mémoire pour augmenter le nombre de motifs pouvant être représentés.
Adapter les méthodes de génération de bruits gaussiens est une alternative intéressante. En effet, les textures ainsi générées présentent les mêmes avantages que les bruits gaussiens, notamment en termes d'aléatoire, de filtrage et de contrôle. Les algorithmes comme le bruit local à phase aléatoire ou le pavage et mélange peuvent être utilisés pour obtenir des textures structurées. Cependant, ces algorithmes ont des difficultés à générer des motifs très structurés comme peuvent le faire les processus sans mélanges.

L'utilisation de fonctions de transfert, telles que le bruit à crête, les cartes de couleurs ou le calcul de la phase, permettent d'obtenir un bon intermédiaire entre les deux groupes de méthodes précédents. Ces méthodes sont basées sur la composition d'un bruit gaussien par une fonction quelconque. Ainsi, elles présentent les avantages des méthodes de génération de bruits et permettent de créer des motifs très structurés en utilisant des fonctions de transfert discontinues. Cependant, les motifs obtenus sont limités par la façon dont les fonctions de transfert sont utilisées. En effet, les cartes de couleurs uni-dimensionnelles ne permettent pas un libre agencement des couleurs et les motifs obtenus sont obligatoirement concentriques. Le bruit *phasor*, quant à lui, est limité à des motifs de vagues profilées.

2.6.3 Positionnement de la thèse

Dans cette thèse, nous présentons une méthode permettant de générer et de filtrer à la volée une grande variété de motifs procéduraux fortement structurés. Pour cela, nous utilisons un bruit gaussien vectoriel composé par une fonction de transfert multi-dimensionnelle. Cette idée est inspirée des cartes de couleurs, mais permet un plus libre agencement des couleurs. L'utilisation de fonctions de transfert sépare la partie stochastique de la création de structure. Cela permet de conserver les avantages de la génération de bruits gaussiens (aléatoire, filtrage, contrôle) et de créer de la structure très contrastée et diversifiée.

Chapitre 3

Génération procédurale et rendu de micro-motifs structurés

Comme nous l'avons vu précédemment (section 2.4.3), la composition d'un bruit par une carte de couleurs permet de générer des motifs structurés en temps réel. Cependant, les cartes de couleurs à une seule dimension produisent un agencement limité entre les différentes couleurs présentes dans la texture finale. Ainsi, les couleurs des motifs obtenus sont obligatoirement concentriques. Dans ce chapitre, nous nous intéressons à l'utilisation de cartes de couleurs multi-dimensionnelles. Plus particulièrement, nous proposons un modèle définit par la composition d'un bruit vectoriel par une fonction de transfert bi-dimensionnelle (stockée comme une carte de couleurs). Ce modèle reproduit les résultats d'algorithmes déjà existant, mais permet également de générer une large variété de motifs structurés ou non. Nous présentons également le filtrage en temps réel de ce modèle.
Ce travail a fait l'objet d'une publication et d'une présentation en conférence lors de *Pacific Graphics 2022*, ainsi que d'une soumission au *Replicability stamp*, validée en janvier 2023.

Sommaire

3.1 Contexte

Dans ce chapitre, nous nous intéressons à la génération de motifs structurés. Nous conservons ici la définition de "structure" donnée dans la section 2.1, à savoir des motifs présentant des arêtes vives, ou des contours à fort contraste. Notre objectif ici est de générer procéduralement et de filtrer des motifs stochastiques structurés en temps réel. Comme nous avons pu le constater dans l'état de l'art, la génération procédurale de structure est un défi complexe. En effet, les méthodes de génération procédurale de bruits (section 2.2), permettent d'obtenir des textures stochastiques et un bon contrôle de l'apparence. Cependant, elles ne permettent pas de faire apparaître de la structure franche. Certaines méthodes ont été adaptées pour générer de la micro-structure (section 2.4.2). Par exemple, le bruit local à phase aléatoire [GSV+14] permet un contrôle partiel des phases et le pavage et mélange [HN18] passe par une transformation d'histogramme pour reproduire des histogrammes non gaussien. Cependant, ces méthodes sont limitées pour la génération de structure. Le bruit local à phase aléatoire nécessite d'ajouter des turbulences pour casser l'apparition de répétitions et le pavage et mélange provoque l'apparition de flou de mélange lorsqu'il est appliqué à des motifs très structurés. D'autres méthodes proposent d'utiliser des fonctions de transfert pour générer de la structure par-dessus un bruit procédural (section 2.4.3). Par exemple, le bruit *phasor* [TEZ+19] extrait la phase instantanée d'un bruit gaussien complexe pour créer des vagues dont le profile est contrôlable. Cependant, les motifs obtenus se limitent à ces vagues profilées. De plus, cette méthode ne permet pas le filtrage en temps réel des motifs obtenus, ce qui limite son utilisation pour du rendu en temps réel (section 2.5). Enfin, la composition d'un bruit procédural par une fonction de transfert contenue dans une carte de couleurs permet la création de structure. Cette méthode permet bien la création de motifs stochastiques structurés et une méthode de filtrage de la composition en temps réel a été présentée [HNPN13]. Cependant, les motifs obtenus se limitent à des zones concentriques et ne permettent pas plus de deux voisinages pour une couleur.

Dans ce chapitre, nous présentons une nouvelle approche de l'utilisation de la composition par une carte de couleurs. La contribution principale est l'utilisation d'un bruit gaussien vectoriel et de cartes de couleurs à deux dimensions. Cette approche permet d'élargir l'éventail des motifs possibles en donnant plus de liberté dans l'agencement des différentes couleurs dans la carte.
Nous commençons par présenter ce modèle ainsi que la méthode permettant son filtrage (section 3.2). Puis, nous développons l'estimation en temps réel des statistiques nécessaires pour utiliser ce filtrage en temps réel (section 3.3). Nous exposons ensuite les choix faits pour l'implémentation concrète du modèle et de son filtrage (section 3.4). Enfin, nous présentons une intuition de l'impact des différents paramètres du modèle à travers une série d'exemples de résultats (section 3.5).

3.2 Modèle

3.2.1 Composition

Le modèle de génération est défini par la composition

$$S(\mathbf{u}) = H \circ \mathbf{N}(\mathbf{u}) \tag{3.1}$$

avec $\mathbf{N}$ un bruit vectoriel, à valeur pour toutes les coordonnées de texture $\mathbf{u} \in \mathbb{R}^2$.

$$\mathbf{N}(\mathbf{u}) = \begin{bmatrix} N_x(\mathbf{u}) \\ N_y(\mathbf{u}) \end{bmatrix} \tag{3.2}$$

Ce bruit vectoriel peut être considéré comme deux bruits scalaires générés de façon complètement indépendante. La figure 3.1 illustre cette composition.

La fonction de transfert H est définie sur un intervalle de $\mathbb{R}^2$. Elle peut être encodée comme une carte de couleurs. Dans ce cas, elle est définie sur un intervalle fini $\mathbb{H} \subset \mathbb{R}^2$, et est à valeur dans $\mathbb{R}^c$ avec c le nombre de canaux. Une carte H en nuances de gris est donc à valeurs dans $\mathbb{R}$ et une carte en couleur, encodée sur les trois canaux Rouge, Vert, Bleu, est à valeurs dans $\mathbb{R}^3$. Pour obtenir la texture finale, les composantes du bruit $\mathbf{N}$ sont évaluées à la coordonnée texture $\mathbf{u}$ et les intensités

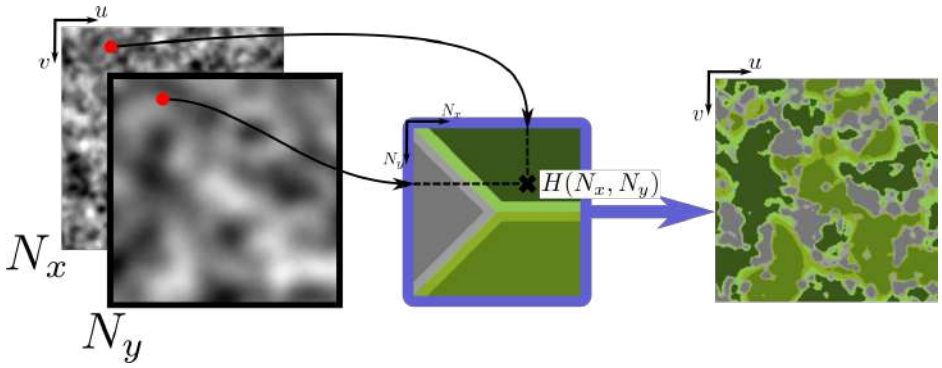
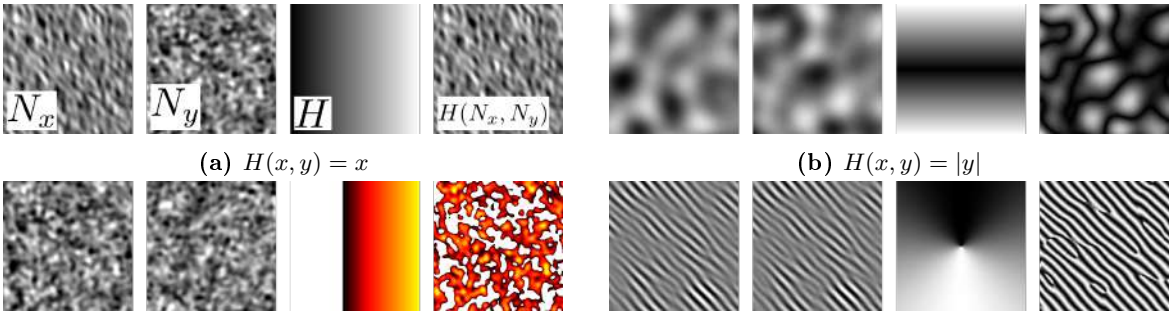


FIGURE 3.1 – *Le modèle est défini par une composition. À gauche, le bruit vectoriel* **N** *est représenté sous la forme de deux bruits scalaires pour des raisons de visibilité. Au centre, la fonction de transfert est stockée sous la forme d'une carte de couleurs. À droite, le résultat de la composition de* **N** *par* H.

obtenues sont utilisées comme coordonnées dans la carte H. Intuitivement, nous pouvons remarquer qu'un déplacement dans les coordonnées texture **u** va induire des variations dans les intensités des composantes de **N**. Ces variations d'intensités vont provoquer un déplacement dans la carte H. Nous observons également que l'agencement des couleurs dans la carte à une forte influence sur la texture finale. En effet, les composantes de **N** sont continues dans $\mathbb{R}^2$, ainsi, les voisinages présents dans la carte H se retrouvent complètement dans la texture S.

Ce modèle présente l'avantage d'inclure les définitions de plusieurs bruits procéduraux déjà connus. Ainsi, il est possible de retrouver un bruit gaussien (figure 3.2a) en considérant deux entrées N_x et N_y gaussiennes et une fonction H faisant une combinaison linéaire (par exemple $H(x, y) = x$ ou $H(x, y) = \frac{x+y}{2}$). Le "bruit à crête" (ou *ridged noise*) correspond à une fonction de transfert de la forme $H(x, y) = |x|$ (figure 3.2b). Les cartes de couleurs traditionnelles, à une dimension, correspondent à l'utilisation d'un seul des deux axes (figure 3.2c). Enfin, le bruit *Phasor* est reproduit par l'utilisation d'une somme de sinus et d'une somme de cosinus en entrée d'une fonction de transfert de la forme $H(x, y) = \gamma \circ \text{Atan2}(y, x)$ avec γ une fonction de profil périodique (figure 3.2d).



(a) $H(x, y) = x$ **(b)** $H(x, y) = |y|$

(c) $H(x, y) = couleur(x)$ **(d)** $H(x, y) = \sin(\text{Atan2}(y, x))$

FIGURE 3.2 – *Reproduction de résultats déjà existant par une composition avec une carte de couleurs à deux dimensions.*

Nous retrouvons l'utilisation des cartes de couleurs à une dimension dans le cas où la fonction de transfert H ne dépend que d'une seule de ses données d'entrée. Cependant, l'utilisation de la seconde dimension de H permet de générer une bien plus grande variété de motifs. En effet, dans le cas des cartes de couleurs à une dimension, une couleur choisie ne peut être voisine que de deux autres couleurs au maximum. Cela restreint le résultat à des motifs concentriques. Dans le cas des cartes à deux dimensions, une couleur peut être voisine d'une infinité d'autres (figure 3.3). Ce qui accroît la liberté d'agencement des motifs.

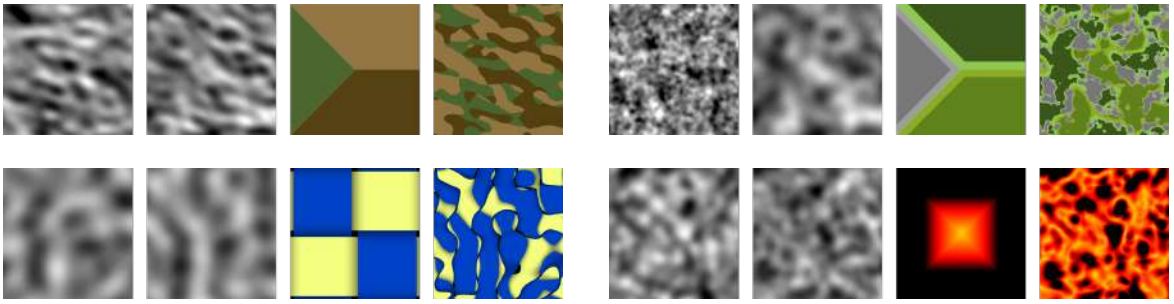


Figure 3.3 – *L'utilisation des deux dimensions de la carte de couleurs permet une plus grande liberté dans l'agencement des couleurs.*

3.2.2 Filtrage

Dans le cas de l'utilisation de motifs procéduraux pour un rendu en temps réel, il est nécessaire de pouvoir faire le filtrage de ces motifs lors du rendu (et donc en temps réel). L'objectif du filtrage est de calculer la moyenne μ_S de la texture S, sur une empreinte de *pixel* $\mathcal{P}$ quelconque, par l'intégrale

$$\mu_S(\mathcal{P}) = \int_{\mathbf{u}\in\mathbb{R}^2} H \circ \mathbf{N}(\mathbf{u}) w_{\mathcal{P}}(\mathbf{u}) \mathrm{d}\mathbf{u} \tag{3.3}$$

Nous considérons que cette empreinte $\mathcal{P}$ est représentée par une fonction de poids $w_{\mathcal{P}}$ positive, à support compact et dont l'intégrale est égale à 1. Les formes usuelles pour $w_{\mathcal{P}}$ sont une fonction boite ou une gaussienne tronquée.

La difficulté est de pouvoir calculer cette intégrale (3.3) en temps constant. En effet, H est non linéaire et $\mathbf{N}$ est procédural, ce qui pose plusieurs difficultés. Ainsi, dans le cas de la composition par une fonction f linéaire, il est possible d'écrire $\int f \circ n(x)\mathrm{d}x = f(\int n(x)\mathrm{d}x)$. Cependant, dans notre cas, H est une fonction arbitraire. Elle a donc toutes les chances d'être non linéaire et peut même ne pas avoir d'écriture mathématique simple dans le cas où elle est représentée par une carte de couleurs dessinées à la main. Il n'est donc pas possible d'envisager de faire la composition par H de l'intégrale du bruit vectoriel $\mathbf{N}$.

D'autre part, en considérant une fonction H stockée en mémoire, nous pourrions être tentés de pré-calculer le résultat de $H \circ \mathbf{N}$ et de calculer l'intégrale ensuite. Cependant, $\mathbf{N}$ est un bruit procédural, cela signifie qu'il n'est pas connu en amont, mais évalué à la volée lors de la génération. Faire un pré-calcul et un stockage en mémoire de $\mathbf{N}$ est possible, mais va à l'encontre des avantages apportés par la génération procédurale. Car, dans ce cas, $\mathbf{N}$ ne sera plus aussi compacte en mémoire (le stockage d'une image en haute définition étant plus conséquent que celui de la procédure de génération). Il sera également dépendant de la résolution dans laquelle il a été pré-calculé (là où la génération procédurale est indépendante de la résolution). Enfin, il sera de dimension finie, car la mémoire d'un ordinateur est finie par contrainte matérielle (là où la génération procédurale permet de générer des textures arbitrairement grandes). Ainsi, il n'est donc possible ni de sortir H de l'intégrale, ni de pré-filtrer $\mathbf{N}$.

Pour évaluer l'intégrale de la composition, nous utilisons un changement de variable d'intégration. L'objectif ici est de pouvoir calculer l'intégrale non pas sur $\mathbf{u}$, qui est à valeur dans l'espace $\mathbb{R}^2$ (infini et non dénombrable) et n'est pas connu en amont, mais sur l'espace des couleurs qui est connu et fini, car complètement décrit par la carte de couleurs H. Cette idée s'appuie sur les travaux de Heitz et collègues [HNPN13].
La constatation permettant le changement de variable est la suivante : la couleur moyenne dans une empreinte est égale à la moyenne de toutes les couleurs pouvant être présentes, pondérées par leur présence dans l'empreinte. Dans le cas de la composition par une carte de couleurs, toutes les couleurs pouvant être présentes sont décrites dans la carte, et leur présence dépend de l'intensité du bruit procédural utilisé. Cela signifie que la couleur moyenne dans une empreinte est égale à la moyenne de toutes les couleurs décrites dans la carte, pondérées par la distribution des intensités du bruit dans cette même empreinte.

Mathématiquement, cela correspond à faire un changement de variable dans l'intégrale précédente. Ainsi, nous ne considérons plus l'espace $\mathbb{R}^2$ sur lequel sont définies les coordonnées $\mathbf{u}$, mais l'espace des intensités que $\mathbf{N}$ peut atteindre et qui correspond à l'espace de définition $\mathbb{H}$ de H. De cette façon, nous couvrons l'ensemble des valeurs d'entrée possibles de H et donc l'ensemble des couleurs présentes dans la carte. Nous avons donc

$$\mu_S(\mathcal{P}) = \int_{\mathbf{N}\in\mathbb{H}} H(\mathbf{N})\mathcal{D}_{\mathbf{N},\mathcal{P}}(\mathbf{N})\mathrm{d}\mathbf{N} \tag{3.4}$$

Avec $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ la fonction de distribution des intensités de $\mathbf{N}$ dans $\mathcal{P}$. Cette distribution tient également compte de la pondération due à la forme de l'empreinte $\omega_{\mathcal{P}}$. L'ensemble des couleurs contenues dans H est parfaitement connu en amont, il faut donc maintenant évaluer la distribution $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ pour connaître la couleur moyenne dans $\mathcal{P}$.
Cette équation (3.4) s'écrit sous la forme d'un produit scalaire entre la carte de couleurs et la distribution des intensités :

$$\mu_S(\mathcal{P}) = \langle H, \mathcal{D}_{\mathbf{N},\mathcal{P}}(.)\rangle \tag{3.5}$$

Pour estimer la distribution $\mathcal{D}_{\mathbf{N},\mathcal{P}}$, nous posons l'hypothèse que le champ d'entrée $\mathbf{N}$ est un processus gaussien. Puis, nous considérons l'approximation que les statistiques de ce processus gaussien suivent une loi normale, quelle que soit l'échelle d'observation. En effet, par définition, un processus gaussien est une collection de variables aléatoires (ici indexées spatialement), telles que chaque sous-ensemble fini de ces variables suit une loi normale. De cette façon, nous approximons la distribution sur une empreinte $\mathcal{P}$ par une gaussienne $\mathcal{G}_{\mu_{\mathbf{N}}(\mathcal{P}),\Sigma_{\mathbf{N}}(\mathcal{P})}$, avec $\mu_{\mathbf{N}}(\mathcal{P})$ et $\Sigma_{\mathbf{N}}(\mathcal{P})$ respectivement la moyenne et la matrice de covariance du champ $\mathbf{N}$ dans $\mathcal{P}$.

Cette approximation fonctionne bien pour de larges empreintes $\mathcal{P}$, mais introduit une erreur dans le cas d'empreintes plus petites (figure 3.4). En effet, si nous considérons une empreinte couvrant une large partie d'un bruit, la distribution des intensités (représentée par l'histogramme) sera une gaussienne. Mais, si nous considérons une empreinte très petite devant la fréquence de variation des intensités, la zone couverte ne sera plus réellement représentative de la distribution globale, la propriété d'ergodicité du bruit n'est donc plus vérifiée.

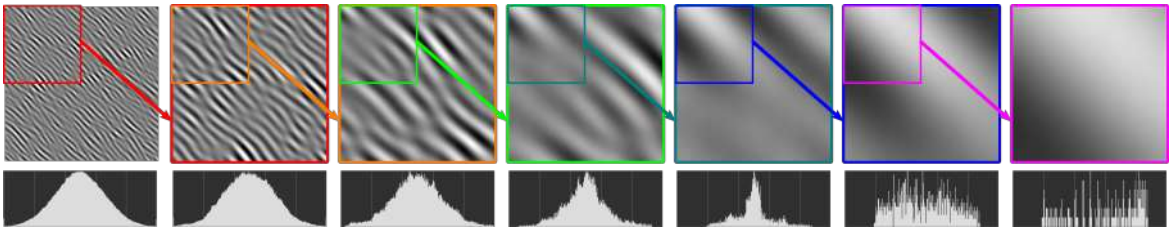


Figure 3.4 – *Zooms successifs dans un bruit initial gaussien. Dans la ligne supérieure, pour chaque étape, la zone encadrée correspondant à l'étape suivante. La ligne inférieure montre le tracé de l'histogramme du bruit pour l'étape actuelle. Tout à gauche, le bruit initial est obtenu par une convolution éparse de noyaux de Gabor, l'histogramme initial est une gaussienne. Dans les étapes suivantes, la forme gaussienne de l'histogramme se dégrade peu à peu, la répartition des intensités n'étant plus réellement représentative de la distribution globale.*

Cette approximation permet cependant de décrire $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ très simplement. En effet, une fonction gaussienne est entièrement décrite par sa moyenne μ et sa variance σ^2. En considérant cela, nous pouvons écrire l'approximation de l'équation (3.5) sous la forme :

$$\mu_S(\mathcal{P}) \approx \left\langle H, \mathcal{G}_{\mu_{\mathbf{N}}(\mathcal{P}),\Sigma_{\mathbf{N}}(\mathcal{P})}\right\rangle \tag{3.6}$$

Finalement, le produit scalaire entre une fonction H et une gaussienne de moyenne μ correspond à la convolution entre la fonction H et une gaussienne de moyenne nulle, évaluée en μ. Nous avons donc :

$$\left\langle H, \mathcal{G}_{\mu_{\mathbf{N}}(\mathcal{P}),\Sigma_{\mathbf{N}}(\mathcal{P})}\right\rangle = \left[H * \mathcal{G}_{\mathbf{0},\Sigma_{\mathbf{N}}(\mathcal{P})}\right](\mu_{\mathbf{N}}(\mathcal{P})) \tag{3.7}$$

Soit $\widehat{\mu_S}$ l'approximation de la couleur moyenne μ_S. Pour une empreinte de *pixel* $\mathcal{P}$, cette approximation dépend de la moyenne $\mu_{\mathbf{N}}(\mathcal{P})$ et de la matrice de covariance $\Sigma_{\mathbf{N}}(\mathcal{P})$ de $\mathbf{N}$ dans l'empreinte. Soit :

$$\widehat{\mu_S}(\mu_{\mathbf{N}}(\mathcal{P}), \Sigma_{\mathbf{N}}(\mathcal{P})) = \left[H * \mathcal{G}_{\mathbf{0},\Sigma_{\mathbf{N}}(\mathcal{P})}\right](\mu_{\mathbf{N}}(\mathcal{P})) \tag{3.8}$$

Cette écriture permet de séparer le calcul du filtrage en deux parties. D'une part, le calcul de la convolution entre la fonction de transfert H et une gaussienne de variance Σ et, d'autre part, l'estimation de la moyenne et de la matrice de covariance de $\mathbf{N}$ dans l'empreinte $\mathcal{P}$. En effet, pour une matrice de covariance donnée, l'évaluation de la convolution est complètement indépendante de $\mathcal{P}$.
Comme la fonction de transfert peut être stockée en mémoire sous la forme d'une carte de couleurs, nous pré-calculons la convolution de H par un nombre fini de fonctions gaussiennes de variance croissante (figure 3.5). C'est-à-dire que nous pré-filtrons la carte H par des noyaux gaussiens de plus en plus grands et stockons les filtrages intermédiaires dans une matrice de cartes $\overline{H}$. Partant de la carte de couleur H initiale, cette opération peut être réalisée en calculant la convolution par des noyaux gaussiens de taille croissante ou par convolutions successives par le même noyau gaussien (détails dans la section 3.4.2). De cette façon, nous obtenons une matrice de cartes de couleurs pré-filtrées, indexées sur la taille du noyau gaussien (taille contenue dans la matrice de covariance) :

$$\overline{H_i}(\mu, \Sigma_i) \tag{3.9}$$

Où $\overline{H_i}$ est la carte H convoluée par un noyau gaussien de matrice de covariance Σ_i.

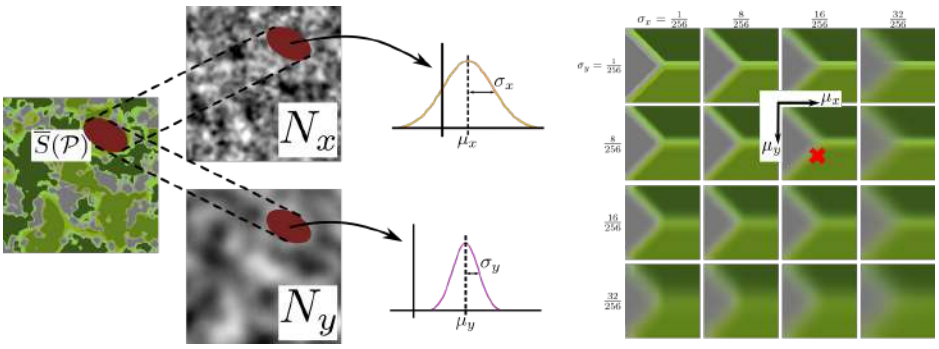


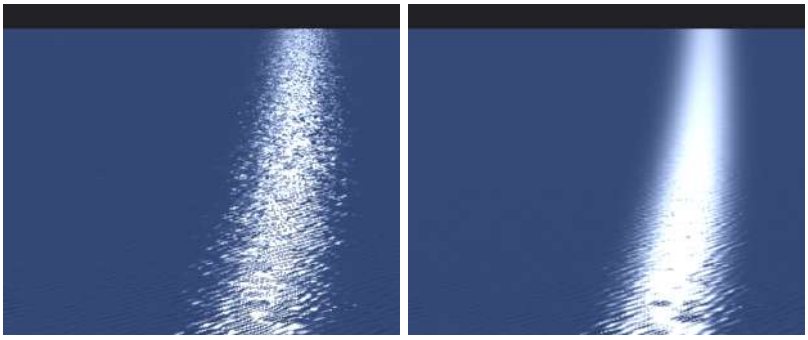
Figure 3.5 – *L'estimation de la couleur moyenne dans une empreinte dans le résultat de la composition (à gauche) est fait en estimant la répartition des intensités des bruits dans cette empreinte (au centre). Ces répartitions sont utilisées dans une version pré-filtrée de la carte H (à droite).*

Nous utilisons ensuite ces cartes pré-filtrées lors du rendu en fonction des caractéristiques statistiques de $\mathbf{N}$. Ainsi, pour une empreinte de *pixel* $\mathcal{P}$, la distribution $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ est caractérisée par une moyenne $\mu_{\mathbf{N}}(\mathcal{P})$ et une matrice de covariance $\Sigma_{\mathbf{N}}(\mathcal{P})$. La matrice $\Sigma_{\mathbf{N}}(\mathcal{P})$ détermine le niveau de filtrage de H et donc la carte $\overline{H_i}$ à utiliser. Dans cette carte, les composantes du vecteur des moyennes $\mu_{\mathbf{N}}(\mathcal{P})$, sont utilisées comme coordonnées pour obtenir la couleur moyenne recherchée.

3.3 Estimateurs temps réel des statistiques dans une empreinte

Pour pouvoir faire le filtrage de la texture finale en temps réel, il reste à estimer, à coût constant, la moyenne $\mu_{\mathbf{N}}(\mathcal{P})$ et la matrice de covariance $\Sigma_{\mathbf{N}}(\mathcal{P})$ du bruit vectoriel $\mathbf{N}$ pour une empreinte de *pixel* quelconque $\mathcal{P}$.

Une estimation de la variance a déjà été proposée par Deliot et collègues [DH19]. Leur constatation est que les valeurs de variance sont très similaires entre elles pour un niveau de filtrage donné. L'approximation qu'ils proposent est donc de considérer que la variance est constante pour chaque niveau de filtrage.
Cependant, les valeurs de variance estimées par cette méthode ne dépendent que du niveau de filtrage et pas de l'empreinte étudiée. Cela conduit à une forte perte de détails pour les empreintes appartenant à un même niveau de filtrage (figure 3.6).



(a) *Référence* **(b)** *[DH19]*

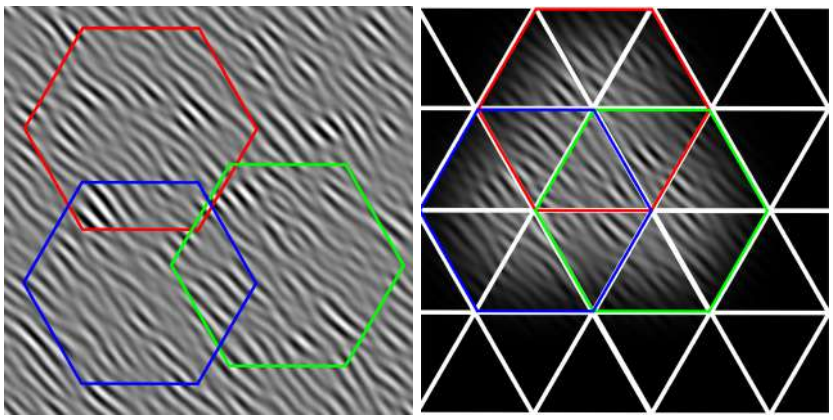
Figure 3.6 – *Filtrage de vagues sur un plan horizontal. à gauche la référence, à droite la matrice de covariance est constante par niveau de filtrage ce qui provoque une perte de détails dans le résultat.*

la méthode que nous développons ici s'appuie sur l'algorithme de synthèse de bruit par pavage et mélange, présenté par Heitz et collègues [HN18].

3.3.1 La synthèse par pavage et mélange

L'algorithme de pavage et mélange de Heitz et collègues est une méthode de génération procédurale basée sur le mélange de tuiles tirées dans un exemple. Cet algorithme est présenté en état de l'art (section 2.2.3). Nous reprenons ici les principales notations qui seront nécessaires pour la suite de cette section.

Cet algorithme de pavage et mélange est une méthode de génération par l'exemple. Il prend en entrée une texture exemple E, sous la forme d'une image de taille finie, et génère une texture de taille infinie dont les statistiques d'ordre 1 et 2 sont similaires à celles de E.
Soit $\mathbf{u} \in \mathbb{R}^2$ les coordonnées dans l'espace de sortie. Cet espace de sortie est pavé par une grille régulière de triangles équilatéraux. Chaque point de l'espace est associé à un triangle et peut être repérés par ses coordonnées barycentriques b_1, b_2, b_3. Nous considérons ensuite une texture d'exemple E représentant un processus gaussien de moyenne nulle. Pour chaque sommet de la grille, une tuile E_i est tirée à une position aléatoire dans l'exemple E. De cette façon, chaque triangle est recouvert par trois tuiles (figure 3.7).



(a) *Tirage des E_i* **(b)** *Pavage et mélange*

Figure 3.7 – *La méthode par pavage et mélange s'appuie sur le tirage de tuiles hexagonales E_i dans la texture exemple puis le pavage de ces tuiles sur les sommets d'une grille de triangles équilatéraux. Image inspirée de [HN18].*

Le bruit N est obtenu en mélangeant les tuiles E_i à l'intérieur des triangles, chacune étant pondérée par un poids w_i de façon à conserver l'histogramme de E.

$$N(\mathbf{u}) = \sum_{i=1}^{3} w_i(\mathbf{u}) E_i(\mathbf{u}) \tag{3.10}$$

Avec les poids w_i définis tels que $w_i = 1$ au centre et $w_i = 0$ au bord des triangles, et tels que $\sum_i w_i^2(\mathbf{u}) = 1$ pour tous $\mathbf{u}$

$$w_i(\mathbf{u}) = \frac{b_i(\mathbf{u})}{\sqrt{\sum_{j=1}^{3} b_j^2(\mathbf{u})}} \tag{3.11}$$

3.3.2 Estimation de la moyenne

Nous cherchons ici à obtenir l'intensité moyenne μ_N de N sur une empreinte $\mathcal{P}$. Analytiquement, cette moyenne est égale à

$$\begin{aligned} \mu_N(\mathcal{P}) &= \int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) N(\mathbf{u}) \mathrm{d}\mathbf{u} \\ &= \int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) \sum_{i=1}^{3} w_i(\mathbf{u}) E_i(\mathbf{u}) \mathrm{d}\mathbf{u} \end{aligned} \tag{3.12}$$

En pratique, $\mathcal{P}$ est une empreinte bornée, avec $w_{\mathcal{P}}(\mathbf{u})$ la pondération due à l'empreinte $\mathcal{P}$ (fonction de poids à support compacte). Les formes les plus communes pour $w_{\mathcal{P}}$ sont la boite et la gaussienne tronquée. La somme est une opération linéaire, nous pouvons donc considérer

$$\mu_N(\mathcal{P}) = \sum_{i=1}^{3} \int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) w_i(\mathbf{u}) E_i(\mathbf{u}) \mathrm{d}\mathbf{u} \tag{3.13}$$

Cependant, $\int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) w_i(\mathbf{u}) E_i(\mathbf{u}) \mathrm{d}\mathbf{u}$ est difficile à calculer en temps constant. En effet, la taille de la zone d'intégration dépend de l'empreinte $\mathcal{P}$. Or celle-ci est de forme et de taille quelconque et croissante lorsque la texture s'éloigne de la caméra. Cela signifie que le temps de calcul sera plus long pour les zones plus éloignées (et donc moins visibles).

Dans le pipeline graphique classique, le GPU transmet deux informations pour permettre le filtrage pour le rendu : la position $\mathbf{u}$ de l'empreinte et le niveau de détail l correspondant. Pour estimer la moyenne du bruit N en temps réel, nous procédons en deux temps. Dans un premier temps, nous exprimons le calcul de moyenne de façon à nous placer dans un cas de calcul de cartes MIP sur la texture d'exemple E pour le niveau l. Pour cela, nous cherchons à estimer la moyenne sur des empreintes de taille quelconque, mais de forme carrée, alignées sur les axes de la texture d'exemple E, et notées $\mathbb{P}$. Dans un second temps, nous décomposons les empreintes quelconques $\mathcal{P}$ en une union d'empreintes carrées $\mathbb{P}$ de façon à estimer la moyenne pour n'importe quelle empreinte.

Empreinte carrée. Soit $\mathbb{P}$ une empreinte carrée dans l'espace $\mathbf{u}$, alignée sur les axes de la texture d'exemple E. Dans un premier temps, nous cherchons à nous placer dans un cas où il nous est possible d'utiliser le pré-calcule du filtrage de E. Pour cela, nous avons besoin de pouvoir considérer que tous les éléments de filtrage sont constants dans une empreinte. De la même façon que pour l'équation (3.13), nous pouvons écrire

$$\mu_N(\mathbb{P}) = \sum_{i=1}^{3} \int_{\mathbf{u}\in\mathbb{P}} w_{\mathbb{P}}(\mathbf{u}) w_i(\mathbf{u}) E_i(\mathbf{u}) \mathrm{d}\mathbf{u} \tag{3.14}$$

Dans le cas d'une empreinte carrée, $w_{\mathbb{P}}(\mathbf{u}) = 1/|\mathbb{P}| \ \forall \mathbf{u} \in \mathbb{P}$, avec $|\mathbb{P}|$ la taille de l'empreinte $\mathbb{P}$. Cela nous permet de simplifier le calcul de la moyenne en sortant la fonction de pondération de l'empreinte du calcul d'intégrale dans l'équation (3.14) :

$$\mu_N(\mathbb{P}) = \sum_{i=1}^{3} \frac{1}{|\mathbb{P}|} \int_{\mathbf{u}\in\mathbb{P}} w_i(\mathbf{u}) E_i(\mathbf{u}) \mathrm{d}\mathbf{u} \tag{3.15}$$

Les poids de mélange w_i varient doucement à l'intérieur de chaque triangle, pavant l'espace de sortie. Nous posons l'hypothèse que w_i varie peu dans une empreinte $\mathbb{P}$. Cette hypothèse nous permet de considérer l'approximation que w_i est constant à l'intérieur d'une empreinte $\mathbb{P}$. Ainsi, nous approximons la valeur de w_i en fonction de la valeur au centre $\dot{\mathbb{P}}$ de l'empreinte $\mathbb{P}$

$$w_i(\mathbf{u}) \approx w_i(\dot{\mathbb{P}}) \ \forall \mathbf{u} \in \mathbb{P} \tag{3.16}$$

Ainsi, nous obtenons une approximation $\widehat{\mu_N}(\mathbb{P})$ de $\mu_N(\mathbb{P})$, en reprenant l'équation (3.15), avec l'approximation de l'équation (3.16) :

$$\widehat{\mu_N}(\mathbb{P}) = \sum_{i=1}^{3} \frac{w_i(\dot{\mathbb{P}})}{|\mathbb{P}|} \int_{\mathbf{u}\in\mathbb{P}} E_i(\mathbf{u})\mathrm{d}\mathbf{u} \tag{3.17}$$

Dans cette formulation, l'intégrale n'est plus calculée que sur les tuiles E_i tirées dans l'exemple E. Or, celui-ci est entièrement connu en amont, étant donné qu'il s'agit d'une donnée d'entrée stockée en mémoire. Nous considérons actuellement des empreintes $\mathbb{P}$ carrées, il est donc possible de pré-calculer $\int_{\mathbf{u}\in\mathbb{P}} E_i(\mathbf{u})\mathrm{d}\mathbf{u}$ pour un nombre fini de niveaux de filtrage par un calcul de carte MIP . Les niveaux de filtrage intermédiaires sont ensuite estimés par interpolation entre les deux niveaux pré-calculés les plus proches. Soit $\{E^0, E^1, E^2, \ldots\}$ la hiérarchie MIP de l'exemple E, avec $E^0 = E$ le niveau de filtrage le plus fin. Pour un niveau de filtrage l, nous avons :

$$\begin{aligned} E^l(\mathbb{P}) &= \frac{1}{|\mathbb{P}|} \int_{\mathbf{u}\in\mathbb{P}} E(\mathbf{u})\mathrm{d}\mathbf{u} \\ &= \frac{1}{|\mathbb{P}|} \sum_{\mathbf{u}\in\mathbb{P}} E(\mathbf{u}) \end{aligned} \tag{3.18}$$

Ainsi, nous obtenons l'approximation $\widehat{\mu_N}(\mathbb{P}, l)$ de la moyenne $\mu_N(\mathbb{P})$ pour une empreinte carrée $\mathbb{P}$, à un niveau l, en injectant la carte MIP de l'équation (3.18) dans l'équation (3.17) :

$$\widehat{\mu_N}(\mathbb{P}, l) = \sum_{i=1}^{3} w_i(\dot{\mathbb{P}}) E_i^l(\mathbb{P}) \tag{3.19}$$

Où la tuile E_i^l est de la même forme que la tuile E_i mais tirée dans E^l. De cette façon, nous obtenons une approximation du filtrage du bruit N généré pour des empreintes $\mathbb{P}$ carrées.

Empreinte quelconque. Lors d'un rendu, les empreintes de *pixel* sont rarement parfaitement carrées, il est donc nécessaire de pouvoir estimer le filtrage sur une empreinte quelconque $\mathcal{P}$. Pour cela, nous considérons $\mathcal{P} = \bigcup \mathbb{P}$ une union de plusieurs empreintes carrées à un niveau l (figure 3.8). Par linéarité, la moyenne d'une union est la moyenne des moyennes des éléments qui la compose.

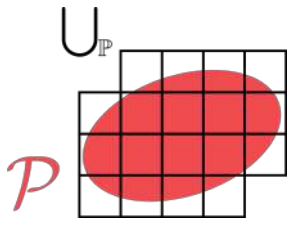


Figure 3.8 – *Découpage d'une empreinte quelconque $\mathcal{P}$ (en rouge) en une union de plusieurs empreintes carrées $\mathbb{P}$ (en noir).*

Les formes les plus classiques pour $\mathcal{P}$ sont la boite ou la gaussienne tronquée, couvrant plusieurs empreintes $\mathbb{P}$. Ses variations à l'intérieur d'une empreinte $\mathbb{P}$ sont donc faibles et de basse fréquence. Nous pouvons donc considérer $w_\mathcal{P}$ comme une fonction constante par morceau et, en particulier, constante à l'intérieur des empreintes $\mathbb{P}$. Soit la fonction de poids $w_\mathcal{P}(\mathbb{P}) > 0$ tel que $\sum_{\mathbb{P}\in\mathcal{P}} w_\mathcal{P}(\mathbb{P}) = 1$, constante à l'intérieur d'une empreinte $\mathbb{P}$. Nous obtenons alors :

$$\widehat{\mu_N}(\mathcal{P}, l) = \sum_{\mathbb{P}\in\mathcal{P}} w_\mathcal{P}(\mathbb{P}) \widehat{\mu_N}(\mathbb{P}, l). \tag{3.20}$$

3.3.3 Estimation de la variance

Nous cherchons ici à estimer, en temps constant, la valeur de la variance σ_N^2 du bruit N sur une empreinte quelconque $\mathcal{P}$:

$$\sigma_N^2(\mathcal{P}) = \int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) \left(N(\mathbf{u}) - \mu_N(\mathcal{P})\right)^2 \mathrm{d}\mathbf{u} \tag{3.21}$$

Contrairement à la moyenne, la variance d'une combinaison linéaire de plusieurs variables n'est pas la combinaison linéaire des variances. Nous ne pouvons donc pas estimer la variance sur une empreinte quelconque en calculant la moyenne des variances des empreintes carrées qui la compose.

Le bruit N est obtenu par un mélange pondéré de tuiles tirées dans E. Nous avons donc :

$$\sigma_N^2(\mathcal{P}) = \int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) \left(\sum_{i=1}^{3} w_i(\mathbf{u}) E_i(\mathbf{u}) - \mu_N(\mathcal{P})\right)^2 \mathrm{d}\mathbf{u} \tag{3.22}$$

Cette formulation met en évidence la mise au carré de la somme des tuiles tirées dans l'exemple. Or, ce passage au carré fait apparaitre des produits croisés entre des tuiles tirées aléatoirement et indépendamment dans l'exemple E. Encore une fois, cette intégrale est difficile à calculer en temps constant, d'autant plus que la zone d'intégration est, là aussi, fonction de la taille de l'empreinte $\mathcal{P}$. Cependant, cette fois, il n'est pas possible d'échanger la somme et l'intégrale en raison du passage au carré.

Comme pour la moyenne, nous estimons la variance en travaillant en deux temps. Un premier temps avec des empreintes carrées $\mathbb{P}$ puis, un second temps avec des empreintes quelconques $\mathcal{P}$ décomposées en union d'empreintes carrées.

Empreintes carrées. Soit $\mathbb{P}$ une empreinte carrée dans l'espace $\mathbf{u}$, alignée sur les axes de la texture d'exemple E. En reprenant l'équation (3.22), nous avons donc :

$$\sigma_N^2(\mathbb{P}) = \int_{\mathbf{u}\in\mathbb{P}} w_{\mathbb{P}}(\mathbf{u}) \left(\sum_{i=1}^{3} w_i(\mathbf{u}) E_i(\mathbf{u}) - \mu_N(\mathbb{P})\right)^2 \mathrm{d}\mathbf{u} \tag{3.23}$$

De nouveau, dans le cas d'une empreinte carrée, $w_{\mathbb{P}}(\mathbf{u}) = 1/|\mathbb{P}|\ \forall \mathbf{u} \in \mathbb{P}$, avec $|\mathbb{P}|$ la taille de l'empreinte $\mathbb{P}$. D'où :

$$\sigma_N^2(\mathbb{P}) = \frac{1}{|\mathbb{P}|} \int_{\mathbf{u}\in\mathbb{P}} \left(\sum_{i=1}^{3} w_i(\mathbf{u}) E_i(\mathbf{u}) - \mu_N(\mathbb{P})\right)^2 \mathrm{d}\mathbf{u} \tag{3.24}$$

Nous utilisons maintenant l'approximation $\widehat{\mu_N}(\mathbb{P}, l)$ calculée dans l'équation (3.19) pour $\mu_N(\mathbb{P})$ et considérons de nouveau que w_i est constant à l'intérieur d'une empreinte $\mathbb{P}$ (*i.e.* $w_i(\mathbf{u}) \approx w_i(\dot{\mathbb{P}})\ \forall \mathbf{u} \in \mathbb{P}$). Ainsi, nous obtenons l'approximation $\widehat{\sigma_N^2}$ de σ_N^2 par :

$$\widehat{\sigma_N^2}(\mathbb{P}) = \frac{1}{|\mathbb{P}|} \int_{\mathbf{u}\in\mathbb{P}} \left(\sum_{i=1}^{3} w_i(\dot{\mathbb{P}}) (E_i(\mathbf{u}) - E_i^l(\mathbb{P}))\right)^2 \mathrm{d}\mathbf{u} \tag{3.25}$$

Pour continuer les calculs, nous développons à part la mise au carré de la somme à l'intérieur de l'intégrale :

$$\begin{aligned} \left(\sum_{i=1}^{3} w_i(\dot{\mathbb{P}}) (E_i(\mathbf{u}) - E_i^l(\mathbb{P}))\right)^2 = & \sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}}) (E_i(\mathbf{u}) - E_i^l(\mathbb{P}))^2 \\ & + 2 \sum_{i\neq j, i<j} w_i(\dot{\mathbb{P}}) w_j(\dot{\mathbb{P}}) (E_i(\mathbf{u}) - E_i^l(\mathbb{P}))(E_j(\mathbf{u}) - E_j^l(\mathbb{P})) \end{aligned} \tag{3.26}$$

Dans ce développement, l'intégrale du produit croisé,

$$\frac{2}{|\mathbb{P}|} \int_{\mathbf{u}\in\mathbb{P}} \sum_{i\neq j, i<j} w_i(\dot{\mathbb{P}}) w_j(\dot{\mathbb{P}}) (E_i(\mathbf{u}) - E_i^l(\mathbb{P}))(E_j(\mathbf{u}) - E_j^l(\mathbb{P})) \mathrm{d}\mathbf{u}$$

calcule la covariance entre les tuiles résultant de deux tirages i et j. Or, le tirage des tuiles est complètement indépendant et l'exemple E représente un bruit, la covariance est donc nulle. Nous pouvons donc reprendre l'équation (3.25) par :

$$\widehat{\sigma_N^2}(\mathbb{P}) = \frac{1}{|\mathbb{P}|}\int_{\mathbf{u}\in\mathbb{P}}\sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}})(E_i(\mathbf{u}) - E_i^l(\mathbb{P}))^2\mathrm{d}\mathbf{u} \tag{3.27}$$

Maintenant que la somme n'est plus mise au carré, nous pouvons écrire :

$$\widehat{\sigma_N^2}(\mathbb{P}) = \sum_{i=1}^{3}\frac{w_i^2(\dot{\mathbb{P}})}{|\mathbb{P}|}\int_{\mathbf{u}\in\mathbb{P}}(E_i(\mathbf{u}) - E_i^l(\mathbb{P}))^2\mathrm{d}\mathbf{u} \tag{3.28}$$

Dans cette formulation, l'intégrale ne porte plus que sur les données connues en amont par la texture d'exemple E. Il est donc possible de pré-calculer $\int_{\mathbf{u}\in\mathbb{P}}(E_i(\mathbf{u}) - E_i^l(\mathbb{P}))^2\mathrm{d}\mathbf{u}$ pour un nombre fini de niveaux de filtrage au moyen d'une carte MIP . Soit $\{V^0, V^1, V^2, \ldots\}$ la hiérarchie MIP de la variance de l'exemple E. Pour un niveau de filtrage l, nous avons :

$$\begin{aligned} V^l(\mathbb{P}) &= \frac{1}{|\mathbb{P}|}\int_{\mathbf{u}\in\mathbb{P}}(E(\mathbf{u}) - E^l(\mathbb{P}))^2\mathrm{d}\mathbf{u} \\ &= \frac{1}{|\mathbb{P}|}\sum_{\mathbf{u}\in\mathbb{P}}\left(E(\mathbf{u}) - E^l(\mathbb{P})\right)^2. \end{aligned} \tag{3.29}$$

Ainsi, nous obtenons l'approximation $\widehat{\sigma_N^2}(\mathbb{P}, l)$ de la variance $\sigma_N^2(\mathbb{P})$ pour une empreinte carrée $\mathbb{P}$, pour un niveau l, en utilisant la carte MIP (3.29) dans l'équation (3.28) :

$$\widehat{\sigma_N^2}(\mathbb{P}, l) = \sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}})V_i^l(\mathbb{P}) \tag{3.30}$$

Empreintes quelconques. Le raisonnement jusque-là est très similaire à celui mené pour la moyenne. Nous pourrions donc être tentés de le poursuivre et d'estimer la variance sur une empreinte quelconque $\mathcal{P}$ par l'union des variances pour des empreintes carrées. Cependant, contrairement à la moyenne, la variance n'est pas linéaire. La variance d'une union n'est donc pas égale à la moyenne des variances. Nous considérons donc plutôt :

$$\sigma_N^2(\mathcal{P}) = \mu_{N^2}(\mathcal{P}) - (\mu_N(\mathcal{P}))^2 \tag{3.31}$$

L'estimation de $\mu_N(\mathcal{P})$ a déjà été traitée plus tôt dans l'équation (3.20), nous nous concentrons donc sur l'estimation de $\mu_{N^2}(\mathcal{P})$. Par définition, nous avons :

$$\mu_{N^2}(\mathcal{P}) = \int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u})N^2(\mathbf{u})\mathrm{d}\mathbf{u} \tag{3.32}$$

De nouveau, nous considérons $\mathcal{P} = \bigcup\mathbb{P}$ une union de plusieurs empreintes carrées à un niveau l. Pour couvrir complètement l'empreinte $\mathcal{P}$, les empreintes $\mathbb{P}$ sont pondérées par une fonction de poids $w_{\mathcal{P}}(\mathbb{P}) > 0$ tel que $\sum_{\mathbb{P}\in\mathcal{P}} w_{\mathcal{P}}(\mathbb{P}) = 1$. Nous avons donc :

$$\mu_{N^2}(\mathcal{P}) = \sum_{\mathbb{P}\in\mathcal{P}}\int_{\mathbf{u}\in\mathbb{P}} w_{\mathcal{P}}(\mathbf{u})N^2(\mathbf{u})\mathrm{d}\mathbf{u} \tag{3.33}$$

Or, en considérant $w_{\mathcal{P}}(\mathbf{u})$ constant par morceau, *i.e.* constant à l'intérieur de chaque empreinte $\mathbb{P}$

$$\begin{aligned} \int_{\mathbf{u}\in\mathbb{P}} w_{\mathcal{P}}(\mathbf{u})N^2(\mathbf{u})\mathrm{d}\mathbf{u} &\approx w_{\mathcal{P}}(\mathbb{P})\int_{\mathbf{u}\in\mathbb{P}} N^2(\mathbf{u})\mathrm{d}\mathbf{u} \\ &= w_{\mathcal{P}}(\mathbb{P})\mu_{N^2}(\mathbb{P}) \end{aligned} \tag{3.34}$$

Par la formule de Koenig-Huygens, nous avons :

$$\mu_{N^2}(\mathbb{P}) = (\mu_N(\mathbb{P}))^2 + \sigma_N^2(\mathbb{P}) \tag{3.35}$$

L'estimation de $\mu_N(\mathbb{P})$ ainsi que celle de $\sigma_N^2(\mathbb{P})$ ont été présentées plus haut (respectivement dans les équations (3.19) et (3.30)). Donc, en utilisant la formule (3.35) et l'approximation (3.34) dans l'équation (3.33), nous obtenons l'approximation $\widehat{\mu_{N^2}}(\mathcal{P}, l)$ de $\mu_{N^2}(\mathcal{P})$ par :

$$\widehat{\mu_{N^2}}(\mathcal{P}, l) = \sum_{\mathbb{P} \in \mathcal{P}} w_{\mathcal{P}}(\mathbb{P}) \left((\widehat{\mu_N}(\mathbb{P}, l))^2 + \widehat{\sigma_N^2}(\mathbb{P}, l) \right). \tag{3.36}$$

Finalement, en injectant cela dans l'équation (3.31), nous obtenons l'estimation de σ_N^2 sur une empreinte $\mathcal{P}$ par :

$$\widehat{\sigma_N^2}(\mathcal{P}, l) = \widehat{\mu_{N^2}}(\mathcal{P}, l) - (\widehat{\mu_N}(\mathcal{P}, l))^2 \tag{3.37}$$

3.3.4 Estimation de la covariance

Nous cherchons ici à estimer en temps constant la valeur de la covariance entre un premier bruit N et un second bruit M. Le raisonnement pour obtenir cet estimateur va être très similaire à celui suivit pour l'estimation de la variance de N. En effet, par définition, la variance d'une variable aléatoire est égale à la covariance de cette variable avec elle-même ($\sigma_{XX} = \sigma_X^2$).
Soit un second bruit gaussien scalaire M obtenu par la méthode de synthèse du pavage et mélange à partir d'un exemple F. La covariance σ_{NM} entre N et M sur une empreinte quelconque $\mathcal{P}$ s'écrit :

$$\sigma_{NM}(\mathcal{P}) = \int_{\mathbf{u} \in \mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) \left(N(\mathbf{u}) - \mu_N(\mathcal{P})\right) \left(M(\mathbf{u}) - \mu_M(\mathcal{P})\right) \mathrm{d}\mathbf{u} \tag{3.38}$$

Les deux bruits N et M sont obtenus par des sommes pondérées de tuiles tirées respectivement dans les exemples E et F. Nous avons donc

$$\sigma_{NM}(\mathcal{P}) = \int_{\mathbf{u} \in \mathcal{P}} w_{\mathcal{P}}(\mathbf{u}) \left(\sum_{i=1}^{3} w_i(\mathbf{u}) E_i(\mathbf{u}) - \mu_N(\mathcal{P}) \right) \left(\sum_{i=1}^{3} w_i(\mathbf{u}) F_i(\mathbf{u}) - \mu_M(\mathcal{P}) \right) \mathrm{d}\mathbf{u} \tag{3.39}$$

Comme pour la variance, cette formulation met en évidence des produits croisés, cette fois entre les différents tirages dans les deux textures exemple. Or, si deux tuiles de même indice i sont tirées de la même façon dans leur exemple respectif, les autres tuiles sont tirées aléatoirement et indépendamment. Encore une fois, cette intégrale est difficile à calculer en temps constant, d'autant plus que la zone d'intégration est, là aussi, fonction de la taille de l'empreinte $\mathcal{P}$. Comme pour la variance, il n'est pas possible d'échanger les sommes et l'intégrale en raison du produit d'éléments non indépendants.

Comme pour la moyenne et la variance, nous étudions la covariance en deux temps. Un premier temps sur des empreintes carrées puis, dans un second temps, sur des empreintes quelconques.

Empreintes carrées. Soit $\mathbb{P}$ une empreinte carrée dans l'espace $\mathbf{u}$, alignée sur les axes des textures d'exemple E et F. En reprenant l'équation (3.39), nous avons donc :

$$\sigma_{NM}(\mathbb{P}) = \int_{\mathbf{u} \in \mathbb{P}} w_{\mathbb{P}}(\mathbf{u}) \left(\sum_{i=1}^{3} w_i(\mathbf{u}) E_i(\mathbf{u}) - \mu_N(\mathbb{P}) \right) \left(\sum_{i=1}^{3} w_i(\mathbf{u}) F_i(\mathbf{u}) - \mu_M(\mathbb{P}) \right) \mathrm{d}\mathbf{u} \tag{3.40}$$

Notons $\{F^0, F^1, F^2, \ldots\}$ la hiérarchie MIP de l'exemple F utilisé pour générer M, avec $F^0 = F$ le niveau de filtrage le plus fin. Nous utilisons l'estimation de $\mu_N(\mathbb{P})$ et $\mu_M(\mathbb{P})$ obtenue par l'équation (3.19). Et, de nouveau, dans le cas d'une empreinte carrée, $w_{\mathbb{P}}(\mathbf{u}) = 1/|\mathbb{P}|$ $\forall \mathbf{u} \in \mathbb{P}$, avec $|\mathbb{P}|$ la taille de l'empreinte $\mathbb{P}$. Nous avons donc :

$$\sigma_{NM}(\mathbb{P}) = \frac{1}{|\mathbb{P}|} \int_{\mathbf{u} \in \mathbb{P}} \left(\sum_{i=1}^{3} w_i(\mathbf{u}) (E_i(\mathbf{u}) - E_i^l(\mathbb{P})) \right) \left(\sum_{i=1}^{3} w_i(\mathbf{u}) (F_i(\mathbf{u}) - F_i^l(\mathbb{P})) \right) \mathrm{d}\mathbf{u} \tag{3.41}$$

En considérant les w_i constant à l'intérieur d'une empreinte (*i.e.* $w_i(\mathbf{u}) \approx w_i(\dot{\mathbb{P}})$ $\forall \mathbf{u} \in \mathbb{P}$), nous obtenons l'approximation $\widehat{\sigma_{NM}}$ de σ_{NM} par :

$$\widehat{\sigma_{NM}}(\mathbb{P}) = \frac{1}{|\mathbb{P}|} \int_{\mathbf{u} \in \mathbb{P}} \left(\sum_{i=1}^{3} w_i(\dot{\mathbb{P}}) (E_i(\mathbf{u}) - E_i^l(\mathbb{P})) \right) \left(\sum_{i=1}^{3} w_i(\dot{\mathbb{P}}) (F_i(\mathbf{u}) - F_i^l(\mathbb{P})) \right) \mathrm{d}\mathbf{u} \tag{3.42}$$

Nous développons à part le produit des deux sommes à l'intérieur de l'intégrale :

$$\begin{aligned}\left(\sum_{(E_i)}\dots\right)\left(\sum_{(F_i)}\dots\right) =& \sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}})(E_i(\mathbf{u})-E_i^l(\mathbb{P}))(F_i(\mathbf{u})-F_i^l(\mathbb{P})) \\ &+\sum_{i\leq j} w_i(\dot{\mathbb{P}})w_j(\dot{\mathbb{P}})(E_i(\mathbf{u})-E_i^l(\mathbb{P}))(F_j(\mathbf{u})-F_j^l(\mathbb{P}))\end{aligned} \tag{3.43}$$

Dans ce développement, l'intégrale des produits croisés entre des tuiles d'indices différents,

$$\frac{1}{|\mathbb{P}|}\int_{\mathbf{u}\in\mathbb{P}}\sum_{i\leq j} w_i(\dot{\mathbb{P}})w_j(\dot{\mathbb{P}})(E_i(\mathbf{u})-E_i^l(\mathbb{P}))(F_j(\mathbf{u})-F_j^l(\mathbb{P}))\mathrm{d}\mathbf{u}$$

calcule la covariance entre un tirage i dans le premier exemple E et un autre tirage j dans le second exemple F. Or, ces tirages sont complètement indépendants, la covariance est donc nulle. Nous pouvons donc reprendre l'équation (3.41) par :

$$\widehat{\sigma_{NM}}(\mathbb{P}) = \frac{1}{|\mathbb{P}|}\int_{\mathbf{u}\in\mathbb{P}}\sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}})(E_i(\mathbf{u})-E_i^l(\mathbb{P}))(F_i(\mathbf{u})-F_i^l(\mathbb{P}))\mathrm{d}\mathbf{u} \tag{3.44}$$

Nous pouvons donc maintenant intervertir la somme et l'intégrale et écrire :

$$\widehat{\sigma_{NM}}(\mathbb{P}) = \sum_{i=1}^{3}\frac{w_i^2(\dot{\mathbb{P}})}{|\mathbb{P}|}\int_{\mathbf{u}\in\mathbb{P}}(E_i(\mathbf{u})-E_i^l(\mathbb{P}))(F_i(\mathbf{u})-F_i^l(\mathbb{P}))\mathrm{d}\mathbf{u} \tag{3.45}$$

Dans cette formulation, l'intégrale ne porte plus que sur des données connues en amont. Il est donc possible de pré-calculer $\int_{\mathbf{u}\in\mathbb{P}}(E_i(\mathbf{u})-E_i^l(\mathbb{P}))(F_i(\mathbf{u})-F_i^l(\mathbb{P}))\mathrm{d}\mathbf{u}$ dans une carte MIP . Soit $\{C^0, C^1, C^2, \dots\}$ la hiérarchie MIP de la covariance entre E et F. Pour un niveau de filtrage l, nous avons :

$$\begin{aligned}C^l(\mathbb{P}) &= \frac{1}{|\mathbb{P}|}\int_{\mathbf{u}\in\mathbb{P}}(E(\mathbf{u})-E^l(\mathbb{P}))(F(\mathbf{u})-F^l(\mathbb{P}))\mathrm{d}\mathbf{u} \\ &= \frac{1}{|\mathbb{P}|}\sum_{\mathbf{u}\in\mathbb{P}}\left(E(\mathbf{u})-E^l(\mathbb{P})\right)\left(F(\mathbf{u})-F^l(\mathbb{P})\right)\end{aligned} \tag{3.46}$$

Ainsi, nous obtenons l'approximation $\widehat{\sigma_{NM}}(\mathbb{P}, l)$ de la covariance $\sigma_{NM}(\mathbb{P})$ entre N et M sur une empreinte carrée $\mathbb{P}$ en utilisant la carte MIP (3.46) dans l'équation (3.45) :

$$\widehat{\sigma_{NM}}(\mathbb{P}, l) = \sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}})C_i^l(\mathbb{P}) \tag{3.47}$$

Empreintes quelconques. Pour estimer la covariance sur une empreinte quelconque, nous suivons le même raisonnement que pour la variance. En effet, la covariance n'étant pas linéaire, la covariance d'une union n'est pas égale à la moyenne des covariances. Nous considérons donc :

$$\sigma_{NM}(\mathcal{P}) = \mu_{NM}(\mathcal{P}) - \mu_N(\mathcal{P})\mu_M(\mathcal{P}) \tag{3.48}$$

Les estimations des moyennes $\mu_N(\mathcal{P})$ et $\mu_M(\mathcal{P})$ ont été traitées dans l'équation (3.20), nous nous concentrons donc sur l'estimation de $\mu_{NM}(\mathcal{P})$. Par définition, nous avons :

$$\mu_{NM}(\mathcal{P}) = \int_{\mathbf{u}\in\mathcal{P}} w_{\mathcal{P}}(\mathbf{u})N(\mathbf{u})M(\mathbf{u})\mathrm{d}\mathbf{u} \tag{3.49}$$

Nous considérons $\mathcal{P} = \bigcup\mathbb{P}$ une union de plusieurs empreintes carrées à un niveau l. Pour couvrir complètement l'empreinte $\mathcal{P}$, les empreintes $\mathbb{P}$ sont pondérées par une fonction de poids $w_{\mathcal{P}}(\mathbb{P}) > 0$ tel que $\sum_{\mathbb{P}\in\mathcal{P}} w_{\mathcal{P}}(\mathbb{P}) = 1$. Nous avons donc :

$$\mu_{NM}(\mathcal{P}) = \sum_{\mathbb{P}\in\mathcal{P}}\int_{\mathbf{u}\in\mathbb{P}} w_{\mathcal{P}}(\mathbf{u})N(\mathbf{u})M(\mathbf{u})\mathrm{d}\mathbf{u} \tag{3.50}$$

Or, en considérant $w_{\mathcal{P}}(\mathbf{u})$ constant par morceau, *i.e.* constant à l'intérieur de chaque empreinte $\mathbb{P}$

$$\begin{aligned}\int_{\mathbf{u}\in\mathbb{P}} w_{\mathcal{P}}(\mathbf{u})N(\mathbf{u})M(\mathbf{u})\mathrm{d}\mathbf{u} &\approx w_{\mathcal{P}}(\mathbb{P})\int_{\mathbf{u}\in\mathbb{P}} N(\mathbf{u})M(\mathbf{u})\mathrm{d}\mathbf{u} \\ &= w_{\mathcal{P}}(\mathbb{P})\mu_{NM}(\mathbb{P})\end{aligned} \tag{3.51}$$

De plus, par la formule de Koenig-Huygens :

$$\mu_{NM}(\mathbb{P}) = \mu_N(\mathbb{P})\mu_M(\mathbb{P}) + \sigma_{NM}(\mathbb{P}) \tag{3.52}$$

En reprenant les estimations de $\mu_N(\mathbb{P})$, $\mu_M(\mathbb{P})$ par l'équation (3.20) et l'estimation de $\sigma_{NM}(\mathbb{P})$ par l'équation (3.47), nous obtenons l'approximation $\widehat{\mu_{NM}}(\mathcal{P},l)$ de $\mu_{NM}(\mathcal{P})$ par :

$$\widehat{\mu_{NM}}(\mathcal{P},l) = \sum_{\mathbb{P}\in\mathcal{P}} w_{\mathcal{P}}(\mathbb{P})\left(\widehat{\mu_N}(\mathbb{P},l)\widehat{\mu_M}(\mathbb{P},l) + \widehat{\sigma_{NM}}(\mathbb{P},l)\right) \tag{3.53}$$

Donc, en injectant dans l'équation (3.48), la covariance $\sigma_{NM}(\mathcal{P})$ sur une empreinte quelconque est finalement estimée par :

$$\widehat{\sigma_{NM}}(\mathcal{P},l) = \widehat{\mu_{NM}}(\mathcal{P},l) - \widehat{\mu_N}(\mathcal{P},l)\widehat{\mu_M}(\mathcal{P},l) \tag{3.54}$$

3.3.5 Résumé des estimateurs

Moyenne. La moyenne $\mu_N(\mathcal{P})$ d'un bruit N, sur une empreinte $\mathcal{P}$, à un niveau l, est estimée par :

$$\widehat{\mu_N}(\mathcal{P},l) = \sum_{\mathbb{P}\in\mathcal{P}} w_{\mathcal{P}}(\mathbb{P})\widehat{\mu_N}(\mathbb{P},l). \tag{3.55}$$

Avec, pour $\mathbb{P}$ une empreinte carrée,

$$\widehat{\mu_N}(\mathbb{P},l) = \sum_{i=1}^{3} w_i(\dot{\mathbb{P}})E_i^l(\mathbb{P}) \tag{3.56}$$

Et $\{E^0, E^1, E^2, \ldots\}$ la hiérarchie MIP de l'exemple E telle que :

$$E^l(\mathbb{P}) = \frac{1}{|\mathbb{P}|}\sum_{\mathbf{u}\in\mathbb{P}} E(\mathbf{u}) \tag{3.57}$$

Variance. La variance $\sigma_N^2(\mathcal{P})$ d'un bruit N, sur une empreinte $\mathcal{P}$, à un niveau l, est estimée par :

$$\widehat{\sigma_N^2}(\mathcal{P},l) = \widehat{\mu_{N^2}}(\mathcal{P},l) - \left(\widehat{\mu_N}(\mathcal{P},l)\right)^2 \tag{3.58}$$

Avec $\widehat{\mu_N}(\mathcal{P},l)$ obtenu par l'équation (3.55) et

$$\widehat{\mu_{N^2}}(\mathcal{P},l) = \sum_{\mathbb{P}\in\mathcal{P}} w_{\mathcal{P}}(\mathbb{P})\left(\left(\widehat{\mu_N}(\mathbb{P},l)\right)^2 + \widehat{\sigma_N^2}(\mathbb{P},l)\right). \tag{3.59}$$

Avec, pour $\mathbb{P}$ une empreinte carrée, $\widehat{\mu_N}(\mathbb{P},l)$ obtenu par l'équation (3.56) et

$$\widehat{\sigma_N^2}(\mathbb{P},l) = \sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}})V_i^l(\mathbb{P}) \tag{3.60}$$

Et $\{V^0, V^1, V^2, \ldots\}$ la hiérarchie MIP de la variance de l'exemple E telle que :

$$V^l(\mathbb{P}) = \frac{1}{|\mathbb{P}|}\sum_{\mathbf{u}\in\mathbb{P}}\left(E(\mathbf{u}) - E^l(\mathbb{P})\right)^2 \tag{3.61}$$

Covariance. La covariance $\sigma_{NM}(\mathcal{P})$ entre deux bruits N et M, sur une empreinte $\mathcal{P}$, à un niveau l, est estimée par :

$$\widehat{\sigma_{NM}}(\mathcal{P},l) = \widehat{\mu_{NM}}(\mathcal{P},l) - \widehat{\mu_N}(\mathcal{P},l)\widehat{\mu_M}(\mathcal{P},l) \tag{3.62}$$

Avec $\widehat{\mu_N}(\mathcal{P},l)$ et $\widehat{\mu_M}(\mathcal{P},l)$ obtenu par l'équation (3.55) et

$$\widehat{\mu_{NM}}(\mathcal{P},l) = \sum_{\mathbb{P}\in\mathcal{P}} w_{\mathcal{P}}(\mathbb{P})\left(\widehat{\mu_N}(\mathbb{P},l)\widehat{\mu_M}(\mathbb{P},l) + \widehat{\sigma_{NM}}(\mathbb{P},l)\right) \tag{3.63}$$

Avec, pour $\mathbb{P}$ une empreinte carrée, $\widehat{\mu_N}(\mathbb{P},l)$ et $\widehat{\mu_M}(\mathbb{P},l)$ obtenu par l'équation (3.56) et

$$\widehat{\sigma_{NM}}(\mathbb{P},l) = \sum_{i=1}^{3} w_i^2(\dot{\mathbb{P}})C_i^l(\mathbb{P}) \tag{3.64}$$

Et $\{C^0, C^1, C^2, \ldots\}$ la hiérarchie MIP de la covariance entre E et F telle que :

$$C^l(\mathbb{P}) = \frac{1}{|\mathbb{P}|}\sum_{\mathbf{u}\in\mathbb{P}}\left(E(\mathbf{u}) - E^l(\mathbb{P})\right)\left(F(\mathbf{u}) - F^l(\mathbb{P})\right) \tag{3.65}$$

3.4 Implémentation

3.4.1 Dimension du filtrage de H

Nous définissons une texture S par la composition d'un bruit vectoriel $\mathbf{N}$ par une fonction de transfert bi-dimensionnelle H en tous points $\mathbf{u}$ de l'espace :

$$S(\mathbf{u}) = H \circ \mathbf{N}(\mathbf{u}) \tag{3.66}$$

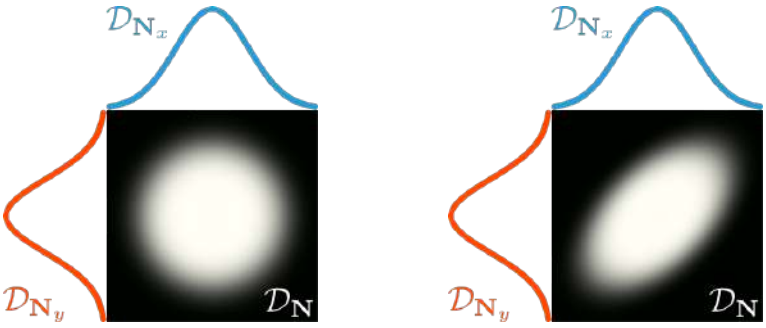
Le filtrage de S consiste à calculer la couleur moyenne μ_S dans une empreinte $\mathcal{P}$ arbitraire. Dans la section 3.2.2, nous avons montré que cette couleur moyenne est égale à la moyenne des couleurs présente dans H, pondérée par la distribution $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ des intensités du bruit $\mathbf{N}$ dans l'empreinte de *pixel* $\mathcal{P}$.

$$\mu_S(\mathcal{P}) = \langle H, \mathcal{D}_{\mathbf{N},\mathcal{P}}(.)\rangle \tag{3.67}$$

Où la distribution $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ représente l'histogramme à deux dimensions de $\mathbf{N}$, soit l'histogramme conjoint des deux composantes de $\mathbf{N}$. Pour réaliser ce filtrage à la volée, nous approximons $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ par une gaussienne à deux dimensions.

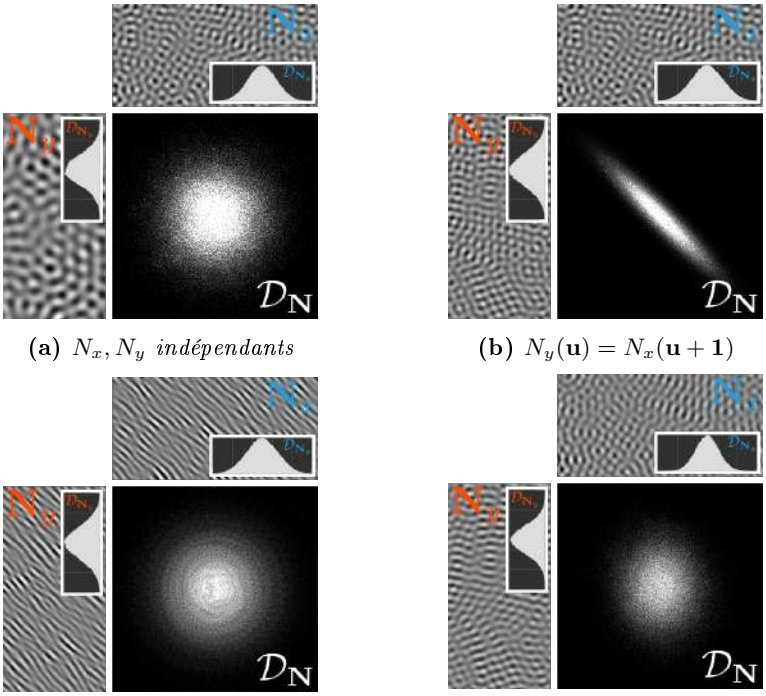
$$\mu_S(\mathcal{P}) \approx \left\langle H, \mathcal{G}_{\mu_{\mathbf{N}}(\mathcal{P}),\Sigma_{\mathbf{N}}(\mathcal{P})}\right\rangle \tag{3.68}$$

L'avantage de cette approximation gaussienne est qu'elle permet de décrire entièrement $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ uniquement avec le vecteur moyen $\mu_{\mathbf{N}}$ et la matrice de covariance $\Sigma_{\mathbf{N}}$.
Dans le cas où les composantes $\mathbf{N}_x$ et $\mathbf{N}_y$ de $\mathbf{N}$ sont indépendantes, alors la covariance $\sigma_{\mathbf{N}_x\mathbf{N}_y}$ entre elles est nulle. Cela permet de réduire le nombre de dimensions de la carte pré-filtrée $\overline{H}$ de 5 à 4 et donc de réduire la taille de $\overline{H}$ en mémoire. Mathématiquement, dans ce cas, la distribution conjointe $\mathcal{D}_{\mathbf{N},\mathcal{P}}$ est égale au produit des distributions marginales $\mathcal{D}_{\mathbf{N}_x,\mathcal{P}}$ et $\mathcal{D}_{\mathbf{N}_y,\mathcal{P}}$. Ainsi, l'indépendance ou non de $\mathbf{N}_x$ et $\mathbf{N}_y$ peut être observée sur le tracer de la distribution conjointe. En effet, dans le cas où $\mathbf{N}_x$ et $\mathbf{N}_y$ sont indépendants, cette distribution est une gaussienne présentant deux axes de symétries selon les deux axes de coordonnées de la carte de couleurs (figure 3.9).
Cette vérification visuelle peut être faite sur une réalisation réelle de $\mathbf{N}$, en traçant son histogramme (figure 3.10). Ainsi, en traçant les histogrammes des couples d'intensités pour les bruits utilisés, nous remarquons empiriquement qu'ils se rapprochent du produit des histogrammes marginaux (à l'exception des cas où la corrélation entre les bruits est forcée, figure 3.10b). Nous faisons également cette constatation dans des cas où les bruits utilisés ne sont pas explicitement indépendants par leur construction, comme dans le cas de l'utilisation d'une somme de sinus et d'une somme de cosinus (cas du bruit *Phasor*, figure 3.10c) ou lors de l'utilisation des dérivées partielles d'un unique bruit N comme composantes de $\mathbf{N}$ (figure 3.10d).



(a) $\mathbf{N}_x$ *et* $\mathbf{N}_y$ *indépendants* **(b)** $\mathbf{N}_x$ *et* $\mathbf{N}_y$ *corrélés*

FIGURE 3.9 – *Tracer théoriques de distributions conjointes de couples de variables aléatoires. Les distributions marginales sont représentées en haut (bleu) et à gauche (rouge) de la distribution conjointe. Dans le cas où les variables sont indépendantes (figure 3.9a) la distribution conjointe est le produit des distributions marginales. Dans le cas contraire (figure 3.9b), la distribution conjointe est déformée.*



(a) N_x, N_y *indépendants* **(b)** $N_y(\mathbf{u}) = N_x(\mathbf{u}+\mathbf{1})$

(c) $N_x = \sum \cos, N_y = \sum \sin$ **(d)** $N_x = \frac{dN}{\mathrm{d}x}$, $N_y = \frac{dN}{\mathrm{d}y}$

FIGURE 3.10 – *Tracer réels de distributions conjointes de bruits gaussiens. Lorsque les bruits N_x et N_y sont parfaitement indépendants (figure 3.10a), la distribution conjointe est le produit des distributions marginales. Lorsque les bruits ne sont pas indépendants (figure 3.10b), la distribution conjointe est déformée en raison de la covariance entre les bruits. L'hypothèse d'indépendance peut être considérée à partir de la distribution conjointe également dans le cas où l'indépendance des bruits d'entrée n'est pas triviale à vérifier (figure 3.10c et 3.10d).*

3.4.2 Échantillonnage des niveaux de filtrage de H

Comme nous venons de le voir, nous pouvons nous contenter de faire le filtrage de H dans un espace de quatre dimensions $(\mu_{N_x}, \mu_{N_y}, \sigma^2_{N_x}, \sigma^2_{N_y})$. La carte filtrée $\overline{H}$ est obtenue par produit de convolution avec des filtres gaussiens à deux dimensions, de largeur croissante. Ainsi, pour un niveau de filtrage l

$$\overline{H_l}(x,y) = \left[H * \mathcal{G}_{0,\sigma_l^2}\right](x,y) \tag{3.69}$$

Avec σ_l^2 un vecteur représentant $\sigma_{N_x}^2, \sigma_{N_y}^2$ et dont les deux composantes varient indépendamment. En pratique, la convolution par des noyaux gaussiens de taille croissante peut être faite par plusieurs convolutions successives par un seul noyau de taille fixe. En effet, si on considère deux fonctions gaussiennes $\mathcal{G}_1$ (de moyenne μ_1 et de variance σ_1^2) et $\mathcal{G}_2$ (de moyenne μ_2 et de variance σ_2^2). Alors le produit de convolution $\mathcal{G}_1 * \mathcal{G}_2$ est une fonction gaussienne, de moyenne $\mu = \mu_1 + \mu_2$ et de variance $\sigma^2 = \sigma_1^2 + \sigma_2^2$. Soit $\mathcal{G}_{0,1}$ un noyau gaussien à une dimension, de moyenne nulle et de variance 1. Le niveau de filtrage l sur un axe de la carte H est obtenu en utilisant le niveau de filtrage $l-1$ sur le même axe :

$$H_l(x,y) = \left[H_{l-1} * \mathcal{G}_{0,1}\right](x,y) \tag{3.70}$$

Algorithm 2: Filtrage par convolutions successives

Entrée : $H = \overline{H_{0,0}}$
$\mathcal{G}_x \leftarrow \mathcal{G}_{0,1}$ un noyau gaussien 1D selon l'axe x;
$\mathcal{G}_y \leftarrow \mathcal{G}_{0,1}$ un noyau gaussien 1D selon l'axe y;
Pour tous *niveaux de filtrage l_x de l'axe x* **:**
  $\overline{H_{l_x,l_y}} \leftarrow \overline{H_{l_x-1,l_y}} * \mathcal{G}_x$;
  Pour tous *niveaux de filtrage l_y de l'axe y* **:**
    $\overline{H_{l_x,l_y}} \leftarrow \overline{H_{l_x,l_y-1}} * \mathcal{G}_y$;
  Fin
Fin

De cette façon, nous obtenons une carte filtrée stockée sous la forme d'une image de la même taille que H pour chaque niveau de filtrage selon les deux dimensions. Cependant, le filtrage successif de H tend vers une image de couleur unie correspondant à la couleur moyenne de H. Ainsi, au-delà d'un certain niveau de filtrage, les résultats sont très similaires (figure 3.11).

(a) $\sigma_x = 63, \sigma_y = 63$ **(b)** $\sigma_x = 64, \sigma_y = 64$

Figure 3.11 – *Deux niveaux de filtrage de la carte H par des filtres gaussiens isotrope d'écart-type respectif $\sigma = 63$ (3.11a) et $\sigma = 64$ (3.11b).*

De plus, les différences entre deux niveaux de filtrage successifs sont de moins en moins visibles alors que $\overline{H}$ tend vers la couleur moyenne de H (figure 3.12).

(a) $\sigma_y = 1$ **(b)** $\sigma_y = 4$ **(c)** $\sigma_y = 60$ **(d)** $\sigma_y = 64$

Figure 3.12 – *Filtrage de la carte H par un filtre gaussien anisotrope, avec $\sigma_x = 1$ pour tous les niveaux. La différence entre les niveaux de filtrage bas (3.12a et 3.12b) est plus visible qu'entre les niveaux de filtrage haut (3.12c et 3.12d) alors que la différence entre les écart-types est la même.*

Nous pouvons donc alléger l'occupation mémoire du pré-filtrage de H en ne conservant qu'une partie des niveaux de filtrage. Pour conserver les variations entre les bas niveaux de filtrage et ne pas stocker trop de niveaux de haut filtrage, nous pouvons baser le stockage en mémoire sur une échelle logarithmique. De cette façon, plus de bas niveaux sont conservés et moins de hauts niveaux de filtrage. Lors du rendu, les niveaux intermédiaires sont retrouvés par interpolation entre les niveaux stockés.

3.4.3 Estimation des statistiques

Méthode. Nous étudions ici l'erreur due aux approximations faites lors du calcul des statistiques (moyenne et variance) pour des empreintes carrées $\mathbb{P}$. Pour cela, nous considérons une texture exemple E, de taille $|E|$. À partir de cet exemple, nous générons un bruit N, stocké dans une image de même taille $|E|$. Les valeurs de référence pour la mesure sont obtenues par le calcul de la hiérarchie MIP de N. Les valeurs approximées de moyenne et de variance sont obtenues en utilisant respectivement les équations (3.56) et (3.60) sur l'exemple E.
Pour chaque niveau de filtrage et chaque estimateur, nous obtenons deux images de taille

$$\mathbf{S} = \left(\frac{|E|}{|\mathbb{P}|}\right)^2$$

contenant chacune les valeurs de références et les valeurs estimées.
Nous mesurons la différence moyenne entre des valeurs de référence X et des valeurs estimées $\widehat{X}$

$$\mathbf{D}_a(X) = \frac{\sum_i |\widehat{X}_i - X_i|}{\mathbf{S}} \tag{3.71}$$

Puis le rapport entre cette différence moyenne et la moyenne des valeurs de référence

$$\begin{aligned} \mathbf{D}_r(X) &= \frac{\frac{\sum_i |\widehat{X}_i - X_i|}{\mathbf{S}}}{\frac{\sum_i X_i}{\mathbf{S}}} \\ &= \frac{\sum_i |\widehat{X}_i - X_i|}{\sum_i X_i} \end{aligned} \tag{3.72}$$

Afin de mesurer l'erreur relativement à la plage de valeurs de la référence.

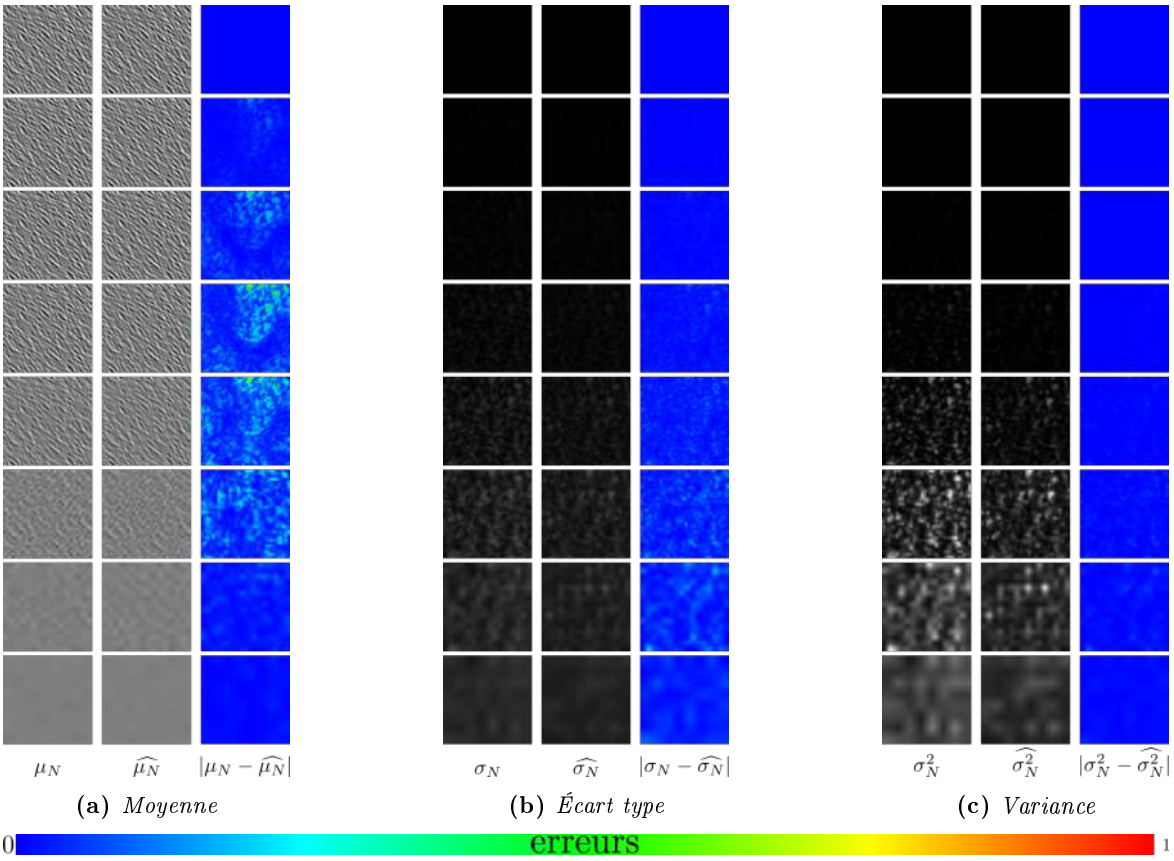
Résultat. Afin d'observer l'impact du contenu du bruit sur les valeurs d'erreur, nous réalisons les mesures sur deux exemples différents : le premier représente un bruit basse fréquence isotrope et le second un bruit haute fréquence anisotrope. Pour faciliter les comparaisons des grandeurs, la mesure est faite sur l'écart-type et non sur la variance.

Dans le tableau (3.1), pour chaque niveau de filtrage (représentés dans les lignes), nous mesurons l'erreur absolue $\mathbf{D}_a$ et l'erreur relative $\mathbf{D}_r$ (entre parenthèse) pour la moyenne et l'écart type. Les texture d'exemple ont une taille de $|E| = 1024 \times 1024$, les images correspondants aux niveaux de filtrage ont donc des tailles allant de $\mathbf{S} = 512 \times 512$ (pour $|\mathbb{P}| = 2 \times 2$) à $\mathbf{S} = 8 \times 8$ (pour $|\mathbb{P}| = 128 \times 128$). Nous remarquons que les erreurs absolues sont faibles autant pour la moyenne que pour l'écart-type. Dans le cas de l'écart-type, l'erreur relative est plus élevée en raison des valeurs de références très faibles. Cependant, l'écart-type est utilisé pour le choix du niveau de filtrage de la carte de couleurs H. Or, pour une position donnée dans la carte, la différence de couleur entre deux niveaux de filtrage est très faible, l'erreur d'estimation de l'écart-type a donc un faible impact sur la précision finale du filtrage.

Nous pouvons également regarder la localisation des erreurs dans le résultat du filtrage (figure 3.13). Ainsi, nous remarquons que les erreurs se concentrent principalement sur les niveaux de filtrage intermédiaires. Un élément très important pour le rendu est la cohérence temporelle. Cette cohérence est assurée par la méthode de synthèse par pavage et mélange qui assure que le résultat est déterministe pour une position donnée dans l'espace texture.

				
	$\widehat{\mu_N}$	$\widehat{\sigma_N}$	$\widehat{\mu_N}$	$\widehat{\sigma_N}$
$\lvert\mathbb{P}\rvert = 2 \times 2$	0.0108 (0.0230)	0.0014 (0.3269)	0.0455 (0.0910)	0.0111 (0.3973)
$\lvert\mathbb{P}\rvert = 4 \times 4$	0.0181 (0.0383)	0.0027 (0.2902)	0.1131 (0.2261)	0.0283 (0.4534)
$\lvert\mathbb{P}\rvert = 8 \times 8$	0.0330 (0.0698)	0.0056 (0.3085)	0.1018 (0.2035)	0.0394 (0.3345)
$\lvert\mathbb{P}\rvert = 16 \times 16$	0.0453 (0.0959)	0.0117 (0.3265)	0.0322 (0.0643)	0.0459 (0.2820)
$\lvert\mathbb{P}\rvert = 32 \times 32$	0.0818 (0.1732)	0.0248 (0.3660)	0.0063 (0.0127)	0.0477 (0.2793)
$\lvert\mathbb{P}\rvert = 64 \times 64$	0.0851 (0.1801)	0.0355 (0.3194)	0.0017 (0.0033)	0.0441 (0.2493)
$\lvert\mathbb{P}\rvert = 128 \times 128$	0.0619 (0.1310)	0.0344 (0.2444)	0.0005 (0.0009)	0.0342 (0.1882)

Table 3.1 – *Mesures des erreurs des estimateurs de la moyenne $\widehat{\mu_N}$ et de l'écart-type $\widehat{\sigma_N}$. Dans chaque cellule, nous trouvons l'erreur absolue $\mathbf{D}_a$ (3.71) et relative $\mathbf{D}_r$ (3.72) (entre parenthèse). En ligne, de haut en bas : des tailles d'empreinte $\lvert\mathbb{P}\rvert$ croissante (indiquées dans la colonne de gauche). En colonne : un bruit isotrope basse fréquence et un bruit anisotrope haute fréquence sont testés.*



(a) *Moyenne* **(b)** *Écart type* **(c)** *Variance*

Figure 3.13 – *Comparaison entre la référence et l'estimation des moyenne, écart-type et variance pour différents niveaux de filtrage. En ligne, les différents niveaux de filtrage en fonction de la taille de l'empreinte $\mathbb{P}$. De $\lvert\mathbb{P}\rvert = 1 \times 1$, sur la première ligne, à $\lvert\mathbb{P}\rvert = 128 \times 128$, sur la dernière ligne.*

3.4.4 Résultat du filtrage

Nous évaluons l'impact des différentes approximations sur la qualité du résultat final. Pour cela, nous mesurons l'erreur entre le filtrage de référence (figure 3.14a) et le résultat de chaque étape d'approximation.

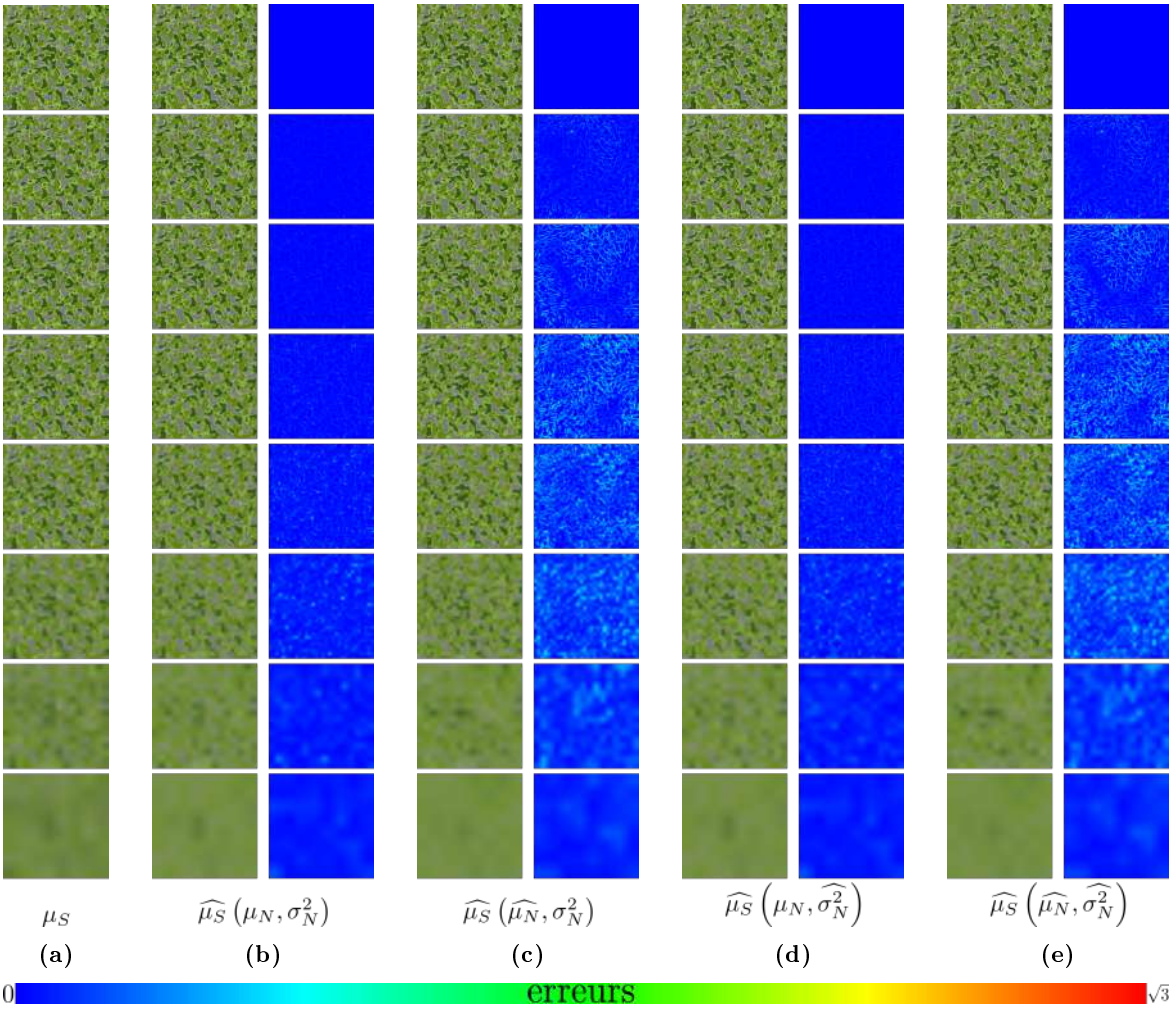


Figure 3.14 – *Utilisation des estimateurs des statistiques pour faire le filtrage de la composition. Pour chaque étape d'approximation, nous mesurons la distance avec le filtrage de référence dans la colonne de droite. En ligne, les différents niveaux de filtrage en fonction de la taille de l'empreinte $\mathbb{P}$. De $|\mathbb{P}| = 1 \times 1$, sur la première ligne, à $|\mathbb{P}| = 128 \times 128$, sur la dernière ligne.*

Approximation de la distribution. La première approximation que nous faisons est de considérer que la distribution des intensités des bruits dans une empreinte est une gaussienne (section 3.2.2, équation (3.6)). En effet, comme nous l'avons vu dans la figure 3.4, l'histogramme d'un bruit gaussien ne peut pas être considéré comme gaussien à tous les niveaux de filtrage. Cela est dû à la perte de la propriété d'ergodicité du bruit. Ainsi, si l'empreinte considérée devient trop petite par rapport à la fréquence maximale du bruit, la distribution dans cette empreinte n'est plus représentative de la distribution globale.
Dans la figure 3.14b, le filtrage est réalisé en calculant exactement la moyenne et la variance du bruit vectoriel $\mathbf{N}$. Ces informations sont utilisées pour approximer la distribution par une gaussienne à chaque niveau.

Approximation des statistiques. Pour réaliser le filtrage en temps réel, nous utilisons des estimations de la moyenne et de la variance à la volée. Comme nous l'avons vu juste avant (section 3.4.3), ces deux estimations ne sont pas exactes. Pour mesurer l'impact des deux estimateurs de moyenne et de variance séparément, nous mesurons la différence entre le filtrage de référence et les résultats utilisant les estimateurs.
Dans la figure 3.14c, la distribution des intensités dans les empreintes est approximée pas une gaussienne dont la moyenne est estimée en utilisant la méthode développée dans la section 3.3.2 et dont la variance est calculée exactement. Dans la figure 3.14d, la distribution des intensités dans les empreintes est approximée par une gaussienne dont la moyenne est calculée exactement et la variance estimée par la méthode développée dans la section 3.3.3.
Nous avons remarqué plus haut (Tableau 3.1) que les erreurs relatives étaient plus élevées pour les estimations de variances que de moyennes. Cependant, nous pouvons voir ici que l'erreur d'estimation de la variance à un impact plus faible que l'erreur d'estimation de la moyenne.

Utilisation de toutes les approximations. Finalement, dans la figure 3.14e, nous utilisons l'approximation de la distribution par une gaussienne et les estimations des moyenne et variance à la volée pour réaliser le filtrage en temps réel. Nous pouvons voir que l'erreur finale vient majoritairement de l'estimation de la moyenne, mais que celle-ci reste faible et que le résultat est visuellement très proche de la référence.

3.5 Résultats

3.5.1 Impact du choix de la carte de couleurs

Analyse. La définition du modèle comme la composition d'un bruit gaussien vectoriel continue $\mathbf{N}$ par une carte de couleurs à deux dimensions H permet de déduire empiriquement plusieurs propriétés du résultat S en fonction du contenu de la carte. Le caractère continu de $\mathbf{N}$ assure que tous les voisinages entre les régions colorées dans S sont dus à des voisinages entre les cellules colorées de H. Ainsi, si deux cellules colorées ne sont pas voisines dans H, les régions correspondantes dans S ne seront jamais voisines (figure 3.15).

Figure 3.15 – *Les voisinages présents dans la carte H (à gauche) se retrouvent dans le résultat de la composition.*

Le caractère gaussien de $\mathbf{N}$, quant à lui, nous donne des informations sur l'utilisation des différentes cellules de la carte H. Ainsi, les bordures de la carte correspondent aux plus hautes et plus basses intensités, alors que le centre correspond aux valeurs proches de l'intensité moyenne. Cela donne une intuition sur les formes prise par les régions colorées de S en fonction de la position des cellules dans la carte H (figure 3.16). Ainsi, les cellules correspondant aux hautes et basses intensités forment des régions plus compactes et convexes (en vert foncé sur la figure 3.16), alors que les cellules correspondant aux intensités moyennes forment des régions plus allongées et semblables à un réseau (en gris foncé sur la figure 3.16).

Ce caractère gaussien nous donne également une information sur la probabilité d'utilisation des différentes cellules de la carte et donc sur la probabilité de présence des différentes couleurs dans S (figure 3.17). Ainsi, la probabilité d'apparition d'une couleur dans S correspond au volume de la cellule de cette couleur dans H, pondéré par la probabilité que cette cellule soit utilisée (c'est-à-dire l'histogramme de $\mathbf{N}$ dans cette partie de la carte H).

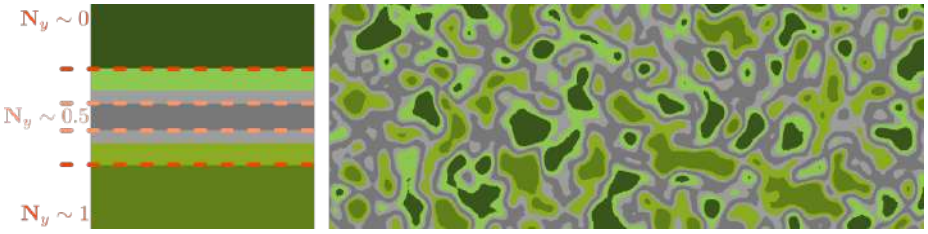


FIGURE 3.16 – *La position des cellules colorées dans la carte H (à gauche) impact la forme prise par les régions colorées dans le résultat de la composition.*

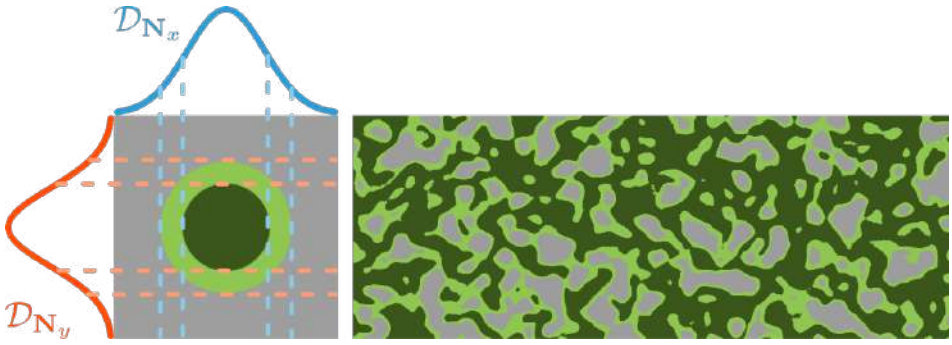
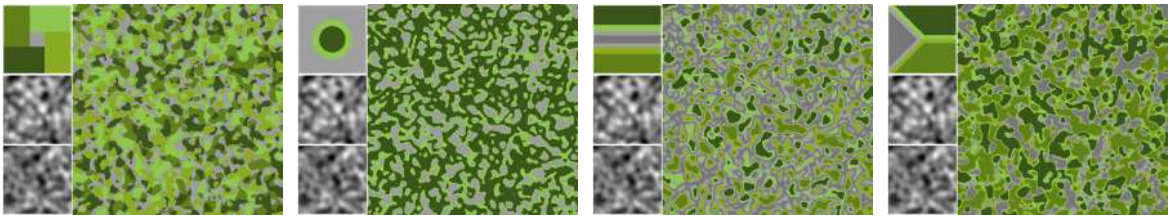


FIGURE 3.17 – *Le volume d'apparition d'une couleur dans le résultat de la composition dépend de la taille des cellules dans la carte H (à gauche) et de la distribution des intensités des bruits utilisés.*

Agencement des cellules. Les analyses précédentes nous permettent d'expérimenter la génération de différentes textures en utilisant le même bruit vectoriel $\mathbf{N}$ et les mêmes couleurs dans la carte H. Pour chacun de ces résultats (figure 3.18), seul l'agencement des cellules dans H est modifié.

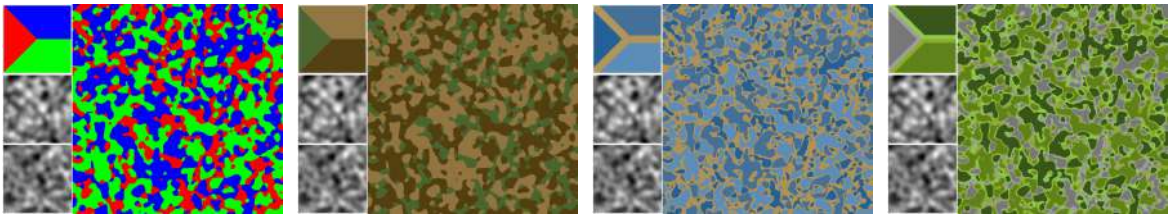


(a) (b) (c) (d)

FIGURE 3.18 – *Différents résultats obtenus en modifiant uniquement l'agencement des cellules dans H. Les composantes du bruit $\mathbf{N}$ (en bas à gauche), ainsi que les couleurs utilisées sont inchangées.*

Nous remarquons que ces résultats reflètent bien l'analyse faites dans le paragraphe précédent. Dans le cas où les bordures de H sont colorées (figure 3.18a), la texture finale présente des régions de couleurs éparses, qui semblent se superposer. Avec des cellules de couleurs disposées de façon concentrique (figure 3.18b), la couleur centrale correspond à la valeur moyenne des deux composantes de $\mathbf{N}$. Les régions de cette couleur dans S vont donc tracer une sorte de réseau, partiellement connecté. En s'éloignant du centre, la cellule en anneau entourant la cellule centrale crée des contours aux motifs précédents. Nous remarquons également que les cellules grises et vert foncé ne sont pas voisines dans H et que les régions grises et vert foncé ne sont donc jamais voisines dans S. De plus, le volume de la cellule grise est plus conséquent que celui de la cellule verte dans H, mais les bords de H ayant une probabilité plus faible d'être utilisés, le volume total de gris dans S est plus faible que le volume de vert foncé. Une carte complètement parallèle à un des axes de la carte H (figure 3.18c) reproduit une carte de couleur uni-dimensionnelle. Donc, de la même façon que ces dernières, cela créé des régions concentriques dans lesquels une couleur ne peut être voisine qu'à deux autres couleurs maximum. Enfin, des frontières de cellules obliques permettent de générer des régions semblant être sur un même plan (figure 3.18d).

Couleur des cellules. Outre la disposition des cellules dans la carte, la palette de couleurs utilisée a également une importance (figure 3.19). En effet, pour une même disposition et un même bruit **N**, les régions obtenus ont les mêmes formes, mais S n'évoque pas le même matériau.



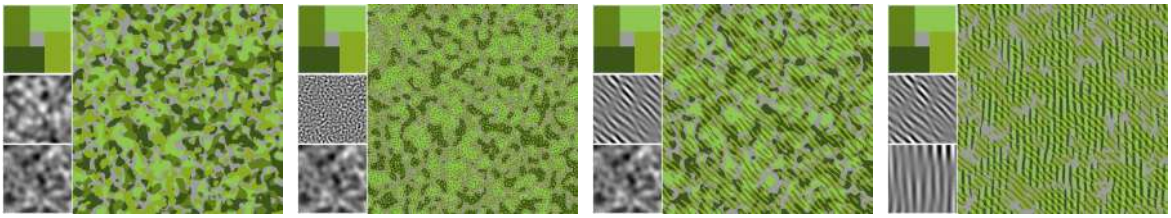
(a) (b) (c) (d)

Figure 3.19 – *Variation de résultat selon la palette de couleurs utilisée. Les composantes du bruit* **N** *(en bas à gauche de chaque sous figure), ainsi que l'agencement des cellules colorées sont inchangées.*

Prenons un découpage de la carte en trois cellules toutes voisines (figure 3.19a). Cet agencement permet de générer des régions contiguës. En utilisant des couleurs dans les tons bruns et verts (figure 3.19b), nous obtenons des motifs évoquant les camouflages militaires. Ajouter une cellule sous la forme d'une bande de couleur unie à l'intersection entre les trois cellules précédentes (figure 3.19c) crée un liseré uni entre les régions et peut évoquer des marbrures. Enfin, colorer les bordures des cellules avec différentes couleurs (figure 3.19d) fait ressortir les contours des régions de différentes façons et peut évoquer de la mousse.

3.5.2 Impact du choix du spectre du bruit d'entrée

La position et le volume des cellules dans la carte H permet de contrôler partiellement la forme et le volume des régions dans S (figure 3.18). Cependant, les caractéristiques spectrales des composantes de **N** ont également un impact direct sur la forme des régions dans S (figure 3.20). Ainsi, pour des bruits **N** présentant le même histogramme, la fréquence, l'orientation et le caractère isotrope ou anisotrope des composantes de **N** influe sur le résultat.



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

Figure 3.20 – *Variation de résultat selon la densité spectrale de* **N**. *L'agencement des cellules dans H, ainsi que les couleurs utilisées sont inchangées.*

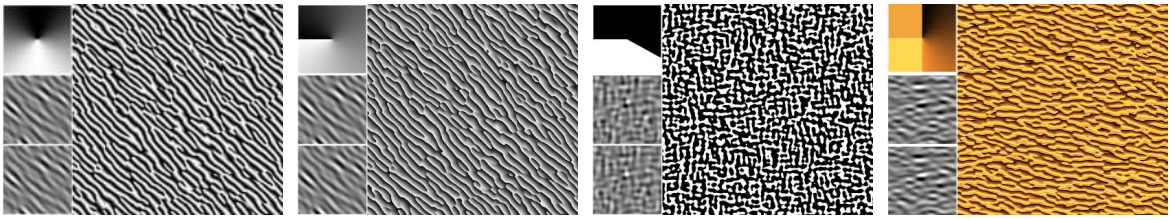
Si les deux composantes de **N** sont isotropes et de basse fréquence (figure 3.20a), les motifs dans le résultat obtenu sont également isotropes et de basse fréquence. Par contre, si l'une des composantes de **N** est de basse fréquence et l'autre de haute fréquence (figure 3.20b), le résultat présente à la fois des motifs basse et haute fréquence. De façon similaire, utiliser une composante isotrope et une composante anisotrope crée des motifs isotropes et des motifs anisotropes dans le résultat (figure 3.20c). Enfin, utiliser deux composantes anisotropes de directions différentes permet d'obtenir des motifs orientés (figure 3.20d). Dans le cas présent, la carte H utilisée permet d'obtenir des bandes orientées entrelacées, comme pour des matériaux tressés.

3.5.3 Filtrage du bruit *phasor*

Notre modèle par composition d'un bruit vectoriel par une fonction à deux dimensions est également capable de reproduire le bruit *phasor* [TEZ+19] (figure 3.21). Comme nous l'avons vu dans l'état de l'art (section 2.4.3) le bruit *phasor* peut être vu comme l'argument d'un bruit gaussien complexe.
Nous reformulons donc cette méthode en considérant un bruit vectoriel $\mathbf{N}$ dont les composantes sont les parties réelle et imaginaire d'un complexe, soit une somme de sinus et une somme de cosinus. Pour maintenir la cohérence entre les phases instantanées des éléments des deux sommes, nous utilisons simplement la méthode de génération par convolution parcimonieuse de noyaux de Gabor avec des harmoniques produites par un cosinus et d'autres par un sinus :

$$\mathbf{N}_{f,\theta}(\mathbf{u}) = \begin{cases} N_x(\mathbf{u}) = \sum_i w_i \; Gaussienne(\mathbf{u} - \mathbf{x}_i) \;\; \cos\left(2\pi f \Theta \cdot (\mathbf{u} - \mathbf{x}_i)\right) \\ N_y(\mathbf{u}) = \sum_i w_i \; Gaussienne(\mathbf{u} - \mathbf{x}_i) \;\; \sin\left(2\pi f \Theta \cdot (\mathbf{u} - \mathbf{x}_i)\right) \end{cases} \tag{3.73}$$

Ensuite, nous considérons une carte de couleurs $H = \gamma \circ \text{Atan2}$, avec γ le profil des vagues.



(a) $\gamma = sinus$ **(b)** $\gamma = dents\ de\ scie$ **(c)** $\gamma = seuil$ **(d)** $\gamma = couleur \circ scie$

Figure 3.21 – *Reproduction du bruit Phasor par la composition d'un bruit vectoriel par une carte contenant le profil et la couleur des vagues.*

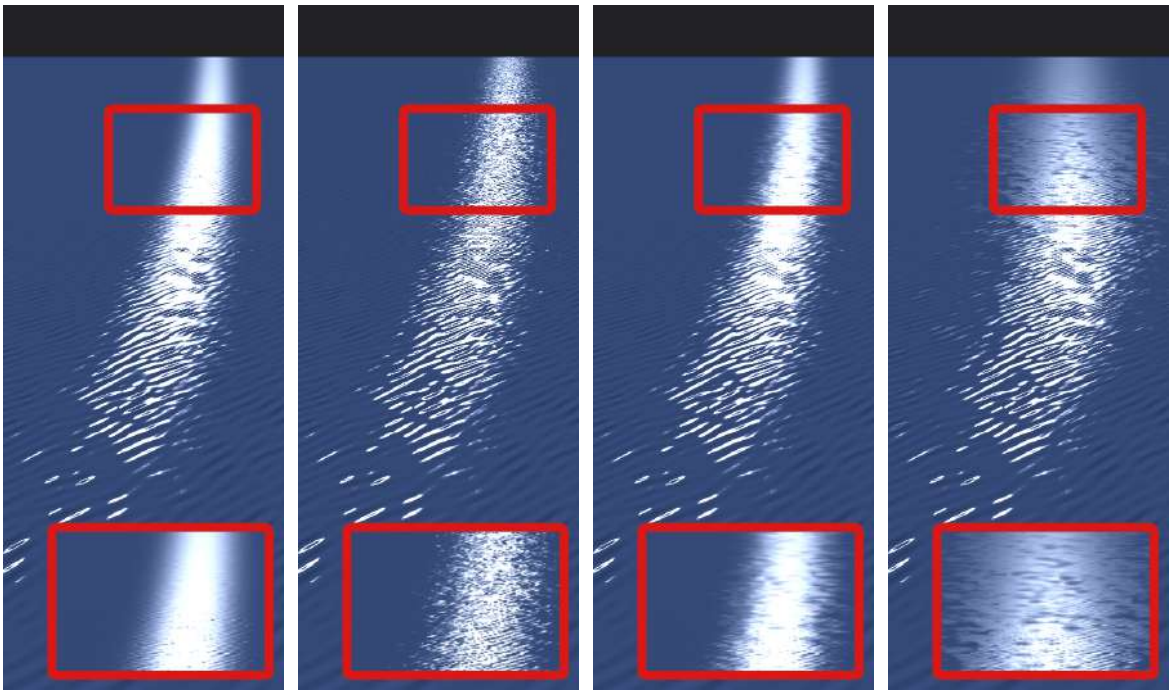
Cette reformulation permet de séparer la génération du champ vectoriel gaussien du calcul de l'arc tangente et l'application du profil. En utilisant une carte $H = \gamma \circ \text{Atan2}$, nous pouvons pré-filtrer cette fonction. En conséquence, cela permet de filtrer le bruit *Phasor* par le pré-filtrage de la carte H comme expliqué précédemment. Des résultats de filtrage sont observables dans la figure 3.23, à droite.

3.5.4 Filtrage de cartes de normales procédurales

Les estimateurs de moyenne, de variance et de covariance présentés en section 3.3, peuvent être utilisés pour le filtrage de textures contenant des informations autres que des couleurs. Par exemple pour le filtrage de cartes de normales générées en utilisant un bruit procédural (figure 3.22). Nous nous plaçons dans le cas où le champ gaussien $\mathbf{N}$ représente le champ des normales à des vagues. Pour cette application, nous cherchons uniquement à utiliser les estimateurs des statistiques pour filtrer ce champ de normales, il n'y a donc pas de composition par une carte de couleurs H.

Un algorithme de filtrage en temps réel des cartes de normales, présenté en 2010 par Olano et Baker [OB10] (appelé *LEAN mapping*) utilise les informations de moyenne et de variance des normales dans une empreinte pour déterminer la rugosité d'un matériau vu de loin. En effet, dans le cas du filtrage de couleurs, en observant le matériau de loin, la couleur de celui-ci va tendre vers la moyenne de ses couleurs. Cependant, dans le cas du filtrage de normales, si nous nous contentons d'estimer la normale moyenne, cela va lisser la surface du matériau. Ainsi, des normales de directions différentes, formant un matériau rugueux, vont se regrouper en une seule normale moyenne, formant un matériau lisse. Pour palier à ce problème, Olano et Baker proposent un algorithme utilisant la moyenne et la matrice de covariance du champ de normales pour estimer la rugosité.

Les différentes méthodes de génération procédurale pour le bruit **N** permettent l'estimation de la moyenne en temps réel. Cependant, jusque-là, la matrice de covariance ne pouvait pas être estimée de façon convaincante en temps réel. L'approximation couramment utilisée consiste à considérer que les composantes de cette matrice sont constantes à l'intérieur d'un niveau de détails [DH19]. Cependant, cette méthode tend à lisser les lobes spéculaires et provoque une perte de détails (figure 3.22a). En utilisant nos estimateurs de variance et de covariance, nous obtenons une meilleure approximation des statistiques dans une empreinte, ce qui permet de retrouver des détails à mi-distance (figure 3.22c).



(a) *[DH19]* **(b)** *Référence* **(c)** *Notre méthode* **(d)** *Covariance nulle*

Figure 3.22 – *Comparaison du filtrage de cartes de normales représentant des vagues sur l'océan. Notre méthode permet de retrouver des détails à mi-distance et de se rapprocher de la référence. En utilisant une covariance nulle (approximation faite pour les couleurs dans la section 3.4.1), les détails à mi-distance sont conservés, seul la forme du lobe globale est modifiée (figure 3.22d).*

3.6 Conclusion et perspectives

Notre objectif était de générer procéduralement et de filtrer un large éventail de motifs stochastiques structurés en temps réel. Dans cette optique, nous avons présenté un modèle inspiré de l'utilisation de cartes de couleurs, mais basé sur un bruit vectoriel **N** et une carte de couleurs bi-dimensionnelle H (section 3.2.1). Comme pour les cartes de couleurs classiques, cette composition permet de séparer la génération procédurale (à travers le bruit **N**) de la création de structure (à travers la carte H). Cependant, la définition d'une carte à deux dimensions permet une plus grande liberté dans l'agencement des couleurs et donc une plus grande variété de résultats possibles (figure 3.23). De plus, nous avons présenté une méthode de filtrage de la composition, en considérant la moyenne de toutes les couleurs présentes dans H, pondérées par la distribution des intensités du bruit **N** (section 3.2.2). Pour pouvoir utiliser ce filtrage en temps réel, nous avons développé une estimation en temps réel de la distribution, à travers l'estimation de la moyenne et la matrice de covariance (section 3.3). À travers une série de tests sur le contenu de H et les caractéristiques spectrales de **N** (section 3.5), nous avons pu présenter une intuition de l'impact des différents paramètres du modèle sur les motifs résultant. Ainsi (entre autres), la position, la surface et le voisinage des différentes cellules colorées dans H vont avoir un impact sur la forme, la taille et les voisinages des régions dans le résultat.

Ces informations dans H sont également à mettre en relation avec les caractéristiques spectrales du bruit $\mathbf{N}$ utilisé, comme sa fréquence, son orientation et son isotropie. L'impact joint de tous ces paramètres est difficile à contrôler manuellement, nous étudions donc une piste de méthode de contrôle en utilisant un exemple de résultat dans le chapitre 5.

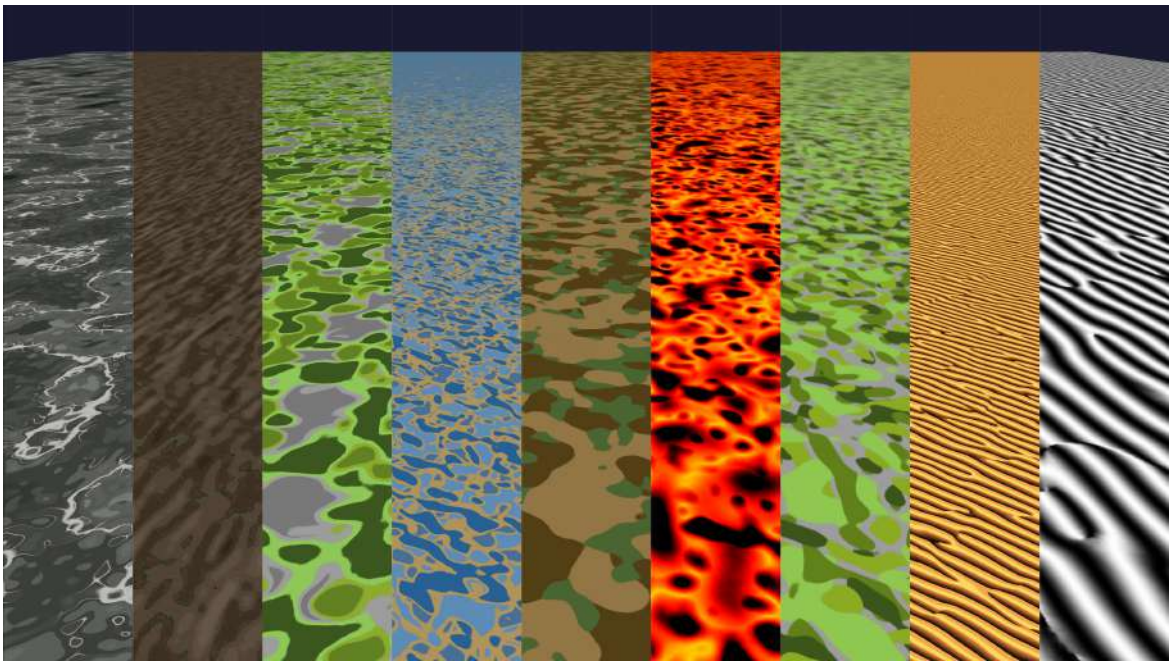


Figure 3.23 – *Résultat du filtrage de différents motifs, appliqués sur un plan dans un espace 3D.*

Chapitre 4

Augmentation de terrains par des motifs procéduraux

Le modèle que nous avons développé dans le chapitre précédent permet de générer à la volée des motifs colorés structurés, mais il peut également être utilisé dans d'autres contextes. Dans ce chapitre, nous explorons l'application de ce modèle à la génération de détails à ajouter à un terrain virtuel pour en augmenter le réalisme. Plus particulièrement, nous nous intéressons à la création de motifs d'érosion hydraulique sur un terrain déjà existant et donné en entrée. Pour cela, nous utilisons le modèle du chapitre précédent pour générer de la géométrie sous la forme d'un champ de hauteurs. Nos contraintes sont de définir une méthode générant en temps réel des motifs géologiquement cohérents avec le terrain initial. De plus, nous voulons que notre méthode permette un contrôle intuitif par des artistes.
Ce travail a fait l'objet d'un article, publié au journal *Computer Graphics Forum* en novembre 2023.

Sommaire

4.1 Contexte

Les terrains jouent un rôle important dans l'atmosphère et le réalisme des scènes virtuelles. Ils sont indispensables dans des domaines tels que la simulation ou l'industrie du divertissement (films, jeux vidéo). Le réalisme de ces terrains est déterminé aussi bien par les éléments à grande échelle (montagnes, plaines, canyons...) que par les détails de relief et d'environnement (ravines, végétation, neige...).

De nombreuses méthodes ont été présentées pour l'augmentation de terrains virtuels [GGP+19]. L'érosion hydraulique est une technique commune pour ajouter des détails naturels sur un terrain virtuel. Celle-ci peut être faite dans une étape de post-traitement, via une simulation de systèmes de particules hydrauliques, par exemple [KBKŠ09]. La méthode de modélisation parcimonieuse, présentée en 2016 [GDGP16], propose l'utilisation de données haute définition, stockées dans un dictionnaire. Ces méthodes produisent des résultats proches de terrains réels, mais présentent des inconvénients. Ainsi, elles sont coûteuses en temps de calcul et doivent être appliquées sur la globalité du terrain, ce qui empêche leur utilisation en temps réel. Elles ne permettent également pas de contrôle artistique par l'utilisateur.
À l'opposé, l'approche procédurale calcule les détails localement en parallèle pour toutes les positions, ce qui permet une utilisation en temps réel. La méthode la plus communément utilisée [EMP+02] se base sur le bruit fractal étudié par Mandelbrot et Van Ness [MVN68], évoqué dans l'état de l'art (section 2.4.2). Cependant, il se pose alors la question de la cohérence globale des détails générés par rapport au terrain. En effet, les méthodes procédurales ne prennent pas en compte certaines informations nécessaires au réalisme d'un terrain, comme l'écoulement de l'eau par exemple.

Nous nous intéressons ici à une représentation des terrains sous la forme de champs de hauteurs. Soit h le champ scalaire définissant le terrain initial. Cette fonction d'élévation est à valeur dans $\mathbb{R}$ pour toute coordonnées $\mathbf{u} \in \mathbb{R}^2$ et modélise un terrain ne présentant que des détails basse résolution (c'est-à-dire, des éléments de grande échelle et pas de détails de relief). Notre objectif est d'ajouter des ravines imitant de l'érosion hydraulique sur ce terrain (figure 4.1).

(a) *Bardenas, Espagne* **(b)** *Pays Glaoua, Maroc* **(c)** *Piau Engaly, Pyrenées* **(d)** *La roche fendue, Jura*

FIGURE 4.1 – *Photographies de ravines dues à de l'érosion hydraulique sur des terrains réels*[1].

Notre modèle d'augmentation ajoute une fonction r, représentant les détails d'érosion, au terrain basse résolution pour obtenir le terrain final T (figure 4.2).

$$T(\mathbf{u}) = h(\mathbf{u}) + r(\mathbf{u}) \tag{4.1}$$

Où r est une fonction procédurale qui varie spatialement, représentant les motifs d'érosion.
Dans un premier temps, nous présentons la modélisation des ravines r (section 4.2). Puis, nous développons la définition de l'apparence des ravines (section 4.3) ainsi que leur guidage spatial par des cartes de contrôle (section 4.4). Enfin, nous proposons une amélioration du rendu par l'ajout en cascade d'un second niveau de ravines (section 4.5) et d'un bruit sur les lignes de crête des ravines (section 4.6). Finalement, nous commentons des détails d'implémentation (section 4.7) et la validation de notre modèle (section 4.8).

1. Crédits photographies : (a) https://loeildeos.com/desert-des-bardenas-reales-espagne/, (b) https://planet-terre.ens-lyon.fr/ressource/degradation-sols.xml, (c) https://coursgeologie.com/1-erosion-221/, (d) http://champa-over-blog-com.over-blog.com/article-la-roche-fendue-41891414.html.

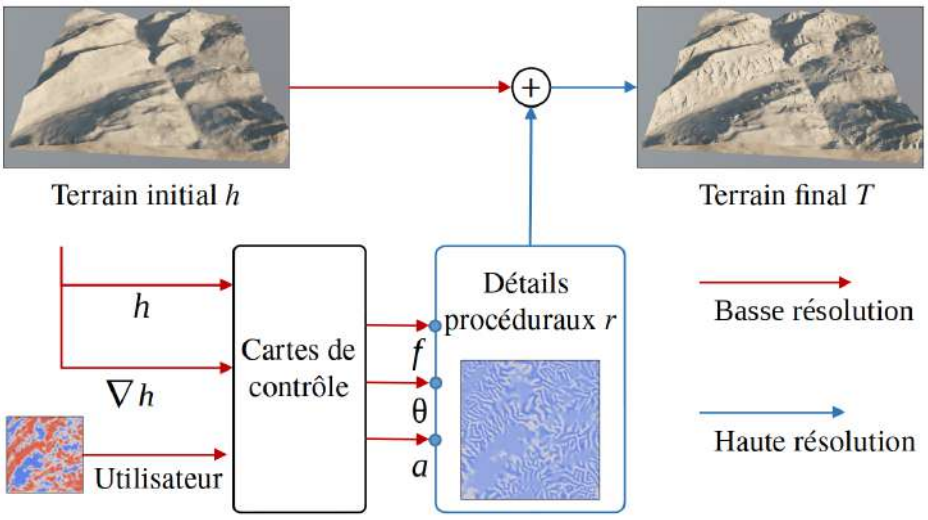


Figure 4.2 – *Pipeline du modèle. Le terrain initial h (en haut à gauche) est augmenté pour obtenir le terrain final T (en haut à droite). Les détails d'augmentation r sont guidés par des cartes de contrôle dépendant du terrain initial, de son gradient et d'éventuelles cartes données par l'utilisateur.*

4.2 Modélisation des ravines

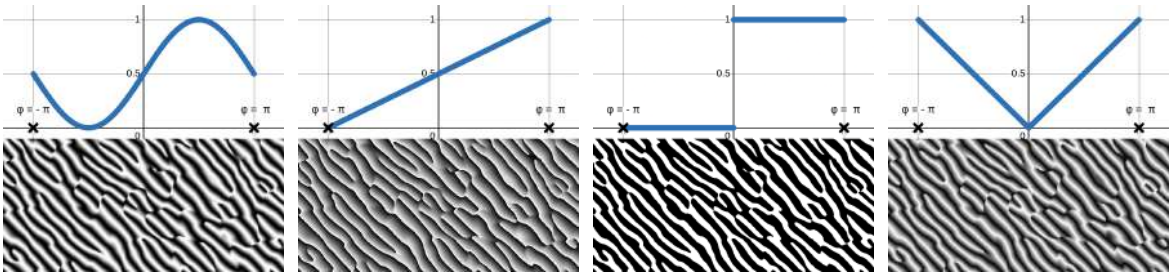
Pour décrire les ravines $r(\mathbf{u})$, nous nous inspirons du bruit *phasor* [TEZ+19] présenté dans l'état de l'art (section 2.4.3). En effet, ce modèle permet de décrire des vagues dont le profil et les caractéristiques spatiales sont contrôlables.

4.2.1 Formulation initiale

Le bruit *phasor* [TEZ+19] permet de générer des vagues profilées et peut être formulé comme la composition d'un champ de phases Φ par une fonction de profil, γ, 2π-périodique.

$$\text{Phasor}(\mathbf{u}) = \gamma \circ \Phi(\mathbf{u}) \tag{4.2}$$

Pour une variable φ représentant une phase, la fonction $\gamma(\varphi)$ décrit le profil transversal des vagues générées (figure 4.3).



(a) $\gamma =$ *sinus* **(b)** $\gamma =$ *dents de scie* **(c)** $\gamma =$ *seuil* **(d)** $\gamma =$ *triangle*

Figure 4.3 – *Exemples de profils γ, tracés pour $\varphi \in [-\pi, \pi]$ en haut, et les résultats de $\gamma \circ \Phi(\mathbf{u})$ correspondants (en bas).*

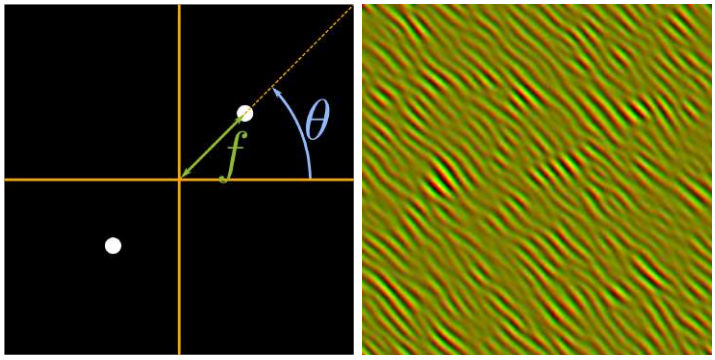
Arbitrairement, nous considérons une fonction $\gamma(\varphi)$ définie de $]-\pi, \pi]$ dans $[0, 1]$, et nous plaçons le fond de la vague en $\varphi = 0$ et la crête en $\varphi = \pm\pi$.
Le champ de phase $\Phi(\mathbf{u})$ est défini pour toute position $\mathbf{u}$ comme l'argument d'un champ gaussien complexe $\widetilde{N}$:

$$\widetilde{N}(\mathbf{u}) = A(\mathbf{u})\ \exp^{(i\Phi(\mathbf{u}))} \tag{4.3}$$

Cet argument peut être obtenu en utilisant la fonction arc tangente entre la partie réelle et la partie imaginaire du bruit complexe $\widetilde{N}$.

$$\begin{aligned}\Phi(\mathbf{u}) &= \text{Atan2} \circ \widetilde{N}(\mathbf{u}) \\ &= \text{Atan2}\left(\Im\left(\widetilde{N}(\mathbf{u})\right), \Re\left(\widetilde{N}(\mathbf{u})\right)\right)\end{aligned} \tag{4.4}$$

Le champ gaussien $\widetilde{N}$ est entièrement contrôlé par sa densité spectrale de puissance (PSD). Pour obtenir des vagues allongées et orientées dans une direction, nous considérons une PSD concentrée autour d'une seule fréquence et une seule orientation. Cette forme de PSD est souvent appelée "bilobe" (figure 4.4a). Notons (f, θ) les coordonnées polaires dans la PSD, le champ gaussien correspondant (figure 4.4b) présente des vagues de fréquence f et d'orientation θ.



(a) *Représentation spectrale* **(b)** *Représentation spatiale*

FIGURE 4.4 – *Un bruit gaussien complexe et sa représentation spectrale dans le domaine de Fourier.*

Le champ de phase est donc guidé en fréquence et en orientation par le champ gaussien complexe :

$$\Phi_{f,\theta}(\mathbf{u}) = \text{Atan2} \circ \widetilde{N_{f,\theta}}(\mathbf{u}) \tag{4.5}$$

4.2.2 Reformulation

Le bruit *phasor*, contrôlé en fréquence et en orientation, est donc obtenu par :

$$\text{Phasor}_{f,\theta}(\mathbf{u}) = \gamma \circ \text{Atan2} \circ \widetilde{N_{f,\theta}}(\mathbf{u}) \tag{4.6}$$

Cependant, il possède certaines limites. En effet, la fonction Atan2 présente une singularité au point $(x, y) = (0, 0)$, qui ne peut pas être traitée avec cette formulation. Cette singularité risque d'altérer la qualité visuelle de nos résultats en introduisant des singularités aux intersections des ravines générées. De plus, elle rend le résultat de la génération non dérivable en certains points.
Pour pouvoir palier à cela, nous reformulons le bruit *phasor* en utilisant le modèle du chapitre 3, de la même façon que dans la section 3.5.3. Soit $\mathbf{N}$ l'écriture vectorielle du champ complexe $\widetilde{N}$. Nous considérons que $\mathbf{N}$ est généré par une méthode de synthèse permettant un contrôle fin en fréquence et en orientation. Ce champ vectoriel est défini de façon à ce que les composantes N_x et N_y soit respectivement une somme de sinus et une somme de cosinus.

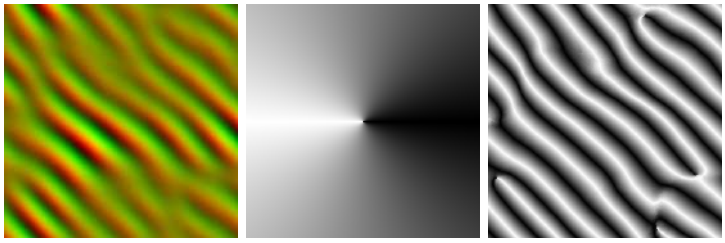
$$\mathbf{N}_{f,\theta}(\mathbf{u}) = \begin{cases} N_x(\mathbf{u}) = \sum_i a_i \cos\left(2\pi f \theta \cdot \mathbf{u}\right) \\ N_y(\mathbf{u}) = \sum_i a_i \sin\left(2\pi f \theta \cdot \mathbf{u}\right) \end{cases} \tag{4.7}$$

De cette façon, le bruit *phasor* s'écrit sous la forme (figure 4.5) :

$$\text{Phasor}_{f,\theta}(\mathbf{u}) = H \circ \mathbf{N}_{f,\theta}(\mathbf{u}) \tag{4.8}$$

Avec

$$H(x, y) = \gamma \circ \text{Atan2}(y, x). \tag{4.9}$$



(a) *Champ* $\mathbf{N}_{f,\theta}$ **(b)** *Fonction* H **(c)** $H \circ \mathbf{N}_{f,\theta}$

FIGURE 4.5 – *Décomposition du bruit phasor en un champ gaussien vectoriel* $\mathbf{N}_{f,\theta}$ *et une fonction de transfert* H*.*

Cette formulation nous permet de mettre en place deux améliorations plus facilement : d'abord, le bruit *phasor* d'origine s'appuie uniquement sur l'argument du champ complexe $\widetilde{N}(\mathbf{u})$ et n'utilise pas son module. Dans notre reformulation, nous utilisons ce dernier pour enrichir le profil des ravines et, en particulier, atténuer la singularité de l'Atan2 en $(x, y) = (0, 0)$ (section 4.3.2). Ensuite, cette écriture nous permet aussi de faire facilement le calcul de la dérivée du résultat de la composition en temps réel (section 4.5.1). Cette information de dérivée est nécessaire pour mettre en place plusieurs niveaux de ravines (section 4.5) mais également pour faire le rendu du résultat final (section 4.7.1). Finalement, nous introduisons une carte $a(\mathbf{u})$ définissant l'amplitude en tout point. Ainsi, le modèle des ravines r est de la forme :

$$r(\mathbf{u}) = a(\mathbf{u}) H \circ \mathbf{N}_{f,\theta}(\mathbf{u}) \tag{4.10}$$

4.3 Définition du profil

Dans cette section, nous cherchons à définir la fonction $H(x, y)$. Celle-ci permet de décrire l'apparence que les ravines $r(\mathbf{u})$ vont prendre. De nombreux choix de H sont possibles, nous développons dans cette section un choix possible pour obtenir une forme de ravine plausible.

4.3.1 Profil transversal des ravines

Le profil transversal des ravines peut être défini arbitrairement selon l'apparence souhaitée. Afin d'imiter des ravines d'érosion hydraulique, nous nous inspirons de l'apparence de terrain réel (figure 4.6) pour définir une forme globale triangulaire avec un fond arrondi.

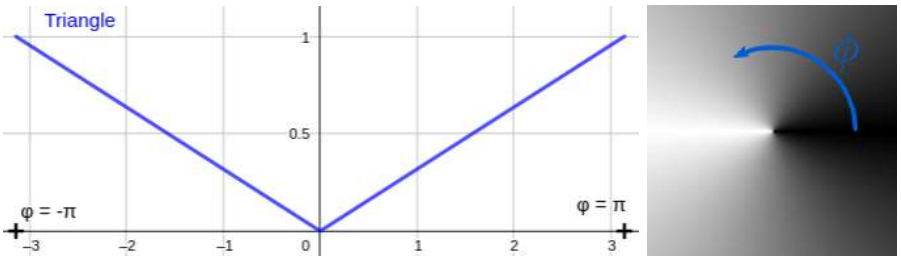
(a) *Piau Engaly, Pyrénées* **(b)** *Profil transversal*

FIGURE 4.6 – *Un profil transversal possible pour des ravines d'érosion dues à de l'écoulement d'eau.*

Dans notre modèle, le profil transversal des ravines est défini par la fonction γ, définie sur $\varphi \in]-\pi, \pi]$ et à valeur dans $[0, 1]$. Le paramètre φ représente la position transversale dans la ravine (figure 4.7a). Arbitrairement, nous situons le fond de la ravine au milieu du profil, en $\varphi = 0$. Mathématiquement, la forme triangulaire peut être décrite par une valeur absolue :

$$\gamma(\varphi) = \left| \frac{\varphi}{\pi} \right| \tag{4.11}$$

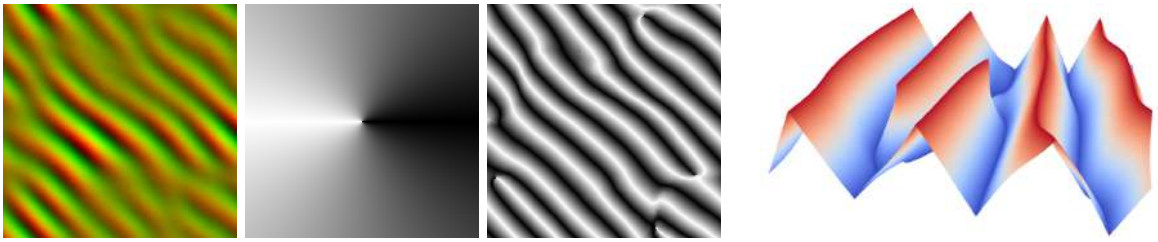
Ce profil γ peut être tracé en 1D en fonction de la coordonnée φ (figure 4.7a) et en 2D selon la coordonnée angulaire de la fonction H (figure 4.7b).



(a) *1D* **(b)** *2D*

FIGURE 4.7 – *Tracé de la fonction de profil γ en fonction de φ. À gauche en 1D et à droite selon la coordonnée angulaire de la carte représentant la fonction H.*

L'application de ce profil transversal permet d'obtenir des ravines parfaitement triangulaires (figure 4.8).



(a) *Champ* $\mathbf{N}_{f,\theta}$ **(b)** *Fonction* H **(c)** $H \circ \mathbf{N}_{f,\theta}$ **(d)** *Vue 3D du résultat*

FIGURE 4.8 – *La composition du champ vectoriel $\mathbf{N}_{f,\theta}$ par la fonction de transfert H permet d'obtenir des motifs de ravines dont le profil transversal, définit par γ, est contenu dans H.*

La nature crée rarement des angles tels que celui de la fonction valeur absolue. Pour rendre les ravines moins artificiellement pointues, nous ajoutons un léger arrondi au fond de la valeur absolue. En terme industriel, cette forme arrondie est appelée "congé de raccordement" et consiste en une transition douce entre deux surfaces (figure 4.9). Dans le cas des ravines d'érosion, cette transition douce entre les deux versants est due au dépôt de sédiments transportés par le ruissellement de l'eau.

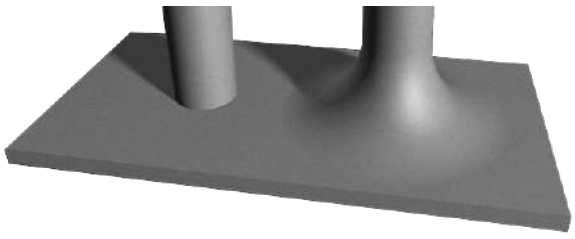


FIGURE 4.9 – *Jonction entre un plan et un cylindre, sans congé de raccordement (à gauche) et avec un congé de raccordement (à droite). Image issue de* `Wikipedia`[2].

Mathématiquement, pour obtenir une transition douce autour de $\varphi = 0$, nous pouvons remplacer la valeur absolue par une fonction racine carrée, biaisée par une valeur ε :

$$\gamma(\varphi) = \sqrt{\left(\frac{\varphi}{\pi}\right)^2 + \varepsilon}, \text{ avec } \varepsilon \ll 1. \tag{4.12}$$

2. https://fr.wikipedia.org/wiki/Arrondi_et_congé

Cette fonction permet d'obtenir un profil proche de celui de la valeur absolue en s'éloignant de $\varphi = 0$, avec une dérivée tendant rapidement vers 1 pour φ croissant (respectivement -1 pour φ décroissant). Elle permet, en outre, d'arrondir le profil pour les valeurs de φ proche de 0 (figure 4.10).

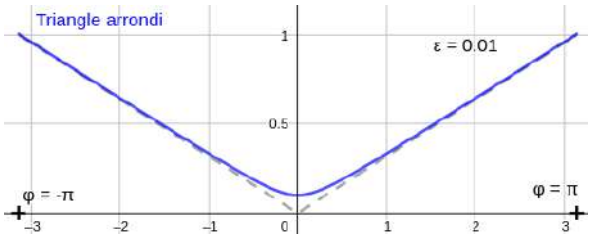
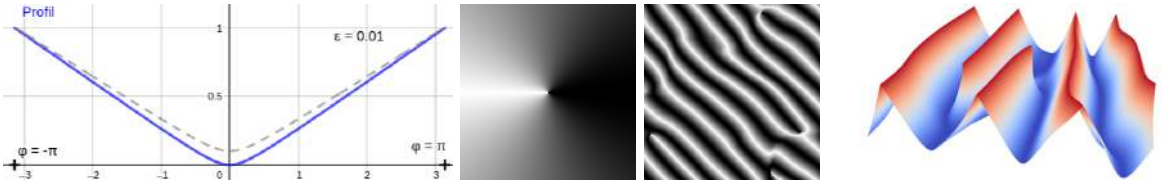


FIGURE 4.10 – *Ajout d'un congé de raccordement pour arrondir le fond du profil des ravines.*

Pour que γ soit à valeur de $]-\pi, \pi]$ dans $[0, 1]$, l'équation (4.12) devient :

$$\gamma(\varphi) = \sqrt{(1 + 2\sqrt{\varepsilon})\left(\frac{\varphi}{\pi}\right)^2 + \varepsilon} - \sqrt{\varepsilon} \tag{4.13}$$

Cette fonction de profil permet de représenter une apparence transversale plausible de ravines (figure 4.11).

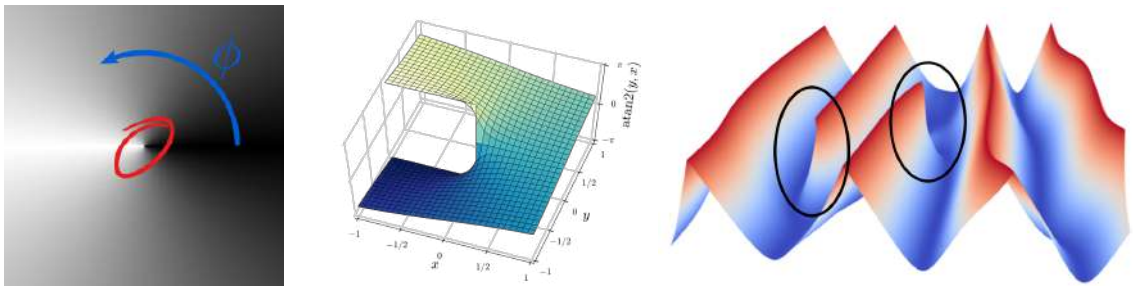


(a) *Profil* $\gamma(\varphi)$ **(b)** *Fonction* H **(c)** $H \circ \mathbf{N}_{f,\theta}$ **(d)** *Vue 3D du résultat*

FIGURE 4.11 – *Tracé du profil γ final que nous utilisons pour définir l'apparence transversale des ravines. Ce profil est contenu dans la fonction H sur la coordonnée angulaire et permet d'obtenir des ravines avec un fond légèrement arrondi.*

4.3.2 Traitement des singularités

Comme nous pouvons le voir dans la figure 4.12, l'Atan2 fait apparaitre une singularité au point de coordonnées $(x, y) = (0, 0)$. Celle-ci est visible au centre du tracer de la fonction H (entourée en rouge dans la figure 4.12a). Mathématiquement, la fonction Atan2 n'est simplement pas définie au point $(x, y) = (0, 0)$. Le problème est que cela risque d'ajouter des singularités dans le champ de hauteurs du terrain final (entourées en noir dans la figure 4.12c).



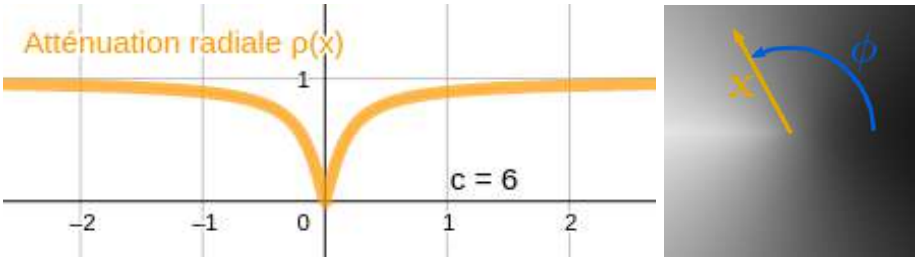
(a) $\gamma \circ \text{Atan2}(y, x)$ **(b)** $\text{Atan2}(y, x)$, *vue 3D* **(c)** *Vue 3D des ravines*

FIGURE 4.12 – *La fonction* Atan2 *présente une singularité au point* $(0, 0)$ *qui provoque l'apparition de variations brusques aux intersections des ravines générées.*

Pour estomper ces singularités dans le résultat, nous travaillons sur la singularité présente au centre de la fonction H. Dans son utilisation classique, le bruit *phasor* exploite principalement l'argument du champ complexe, représenté sur la coordonnée angulaire de H. Pour atténuer la singularité, nous utilisons l'information de module. Visuellement, celle-ci est contenue dans la coordonnée radiale de la carte H. Nous introduisons une fonction d'atténuation ρ, définie sur le rayon de la carte H. Celle-ci doit être à 0 au point $(x, y) = (0, 0)$ pour supprimer la singularité, croître rapidement en s'en éloignant, et tendre vers 1 pour ne pas perturber le profil angulaire γ ailleurs. Plusieurs fonctions pourraient correspondre à ces contraintes, nous pouvons par exemple définir :

$$\rho(\mathbf{x}) = \frac{2}{\pi}\text{Atan}(c\|\mathbf{x}\|), \text{ avec } c \gg 1. \tag{4.14}$$

Où le paramètre c correspond à la vitesse à laquelle l'amplitude diminue en approchant du centre de la carte H (figure 4.13).



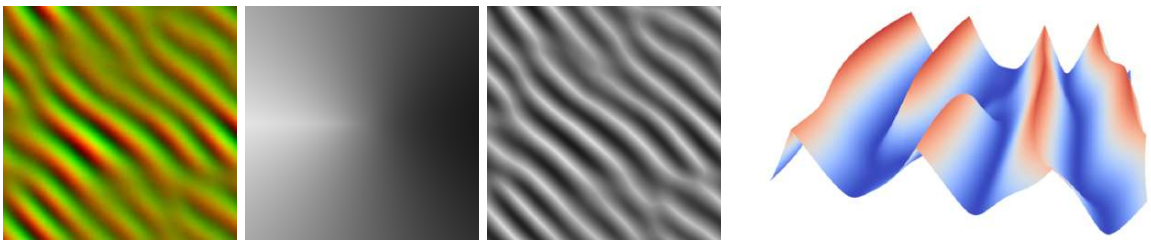
(a) *1D* **(b)** *2D*

Figure 4.13 – *Tracer de la fonction de profil ρ en fonction de $\mathbf{x}$. À gauche en 1D et à droite selon la coordonnée radiale de la carte représentant la fonction H.*

La fonction d'atténuation ρ est prise en compte dans la carte de transfert H à la façon d'une fonction de pondération. Enfin, pour des facilités de manipulation, nous voulons que la fonction de transfert finale H tende, au point $(x, y) = (0, 0)$, vers la moyenne du profil γ (figure 4.14).

$$H(\mathbf{x}) = \rho(\mathbf{x}) \times \gamma \circ \text{Atan2}(\mathbf{x}) + (1 - \rho(\mathbf{x})) \times \mu_\gamma \tag{4.15}$$

De cette façon, contrairement au bruit *phasor* initial, la fonction H ne dépend plus uniquement de la phase du champ complexe, mais du produit entre une fonction appliquée au module et une fonction appliquée à la phase.



(a) *Champ* $\mathbf{N}_{f,\theta}$ **(b)** *Fonction* H **(c)** $H \circ \mathbf{N}_{f,\theta}$ **(d)** *Vue 3D du résultat*

Figure 4.14 – *La modification de la fonction H par le profil radial ρ atténue la singularité au centre de la fonction. La composition du champ $\mathbf{N}_{f,\theta}$ par cette nouvelle carte H permet d'obtenir un résultat ne présentant pas de singularités aux intersections entre les ravines.*

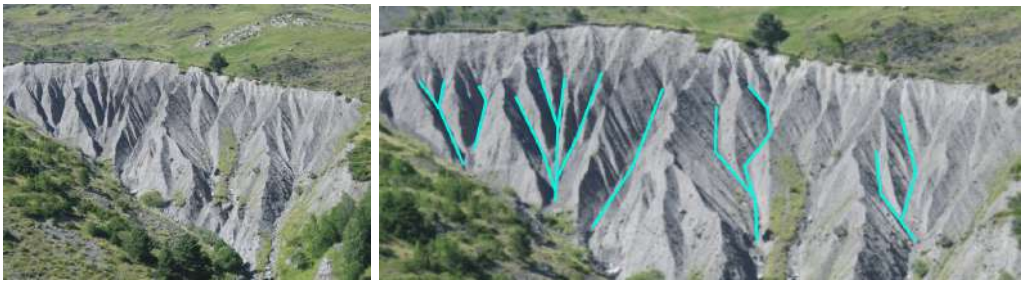
4.4 Contrôle spatial

Comme nous l'avons vu plus haut (équation (4.10)), les caractéristiques de fréquence et d'orientation des ravines r sont guidées par le contrôle de la fréquence et de l'orientation du champ gaussien

N. De plus, l'amplitude des ravines est contrôlée par un champ scalaire a utilisé comme coefficient multiplicateur. Le champ **N** est généré à la volée par un algorithme procédural. Il est donc défini pour chaque position **u** indépendamment et peut être contrôlé spatialement, comme nous l'avons vu dans l'état de l'art (section 2.3.2). Dans cette section, nous montrons comment guider spatialement les caractéristiques des ravines à travers le contrôle spatial du champ gaussien **N** par des cartes de contrôle d'amplitude, de fréquence et d'orientation.

4.4.1 Orientation

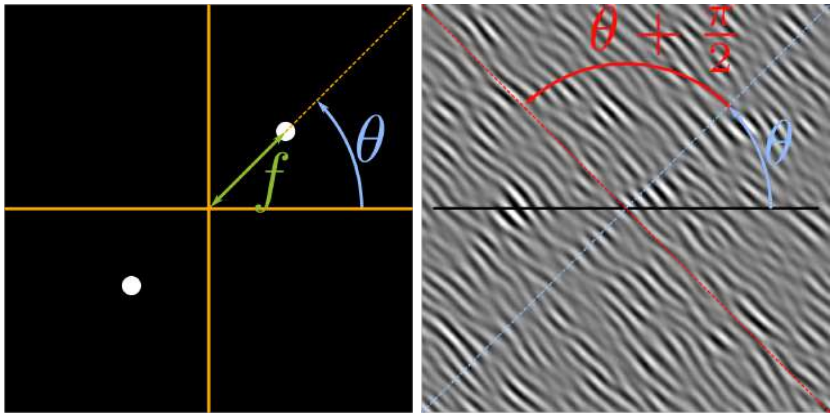
L'écoulement de l'eau sur une pente se fait du sommet vers le bas (figure 4.15). Dans notre cas, le terrain est défini par le champ de hauteur h. La pente du terrain est donc obtenue en tout point grâce au gradient ∇h.



(a) *Piau Engaly, Pyrénées* **(b)** *Direction des ravines*

FIGURE 4.15 – *Tracé de la direction des ravines d'érosion sur un terrain réel.*

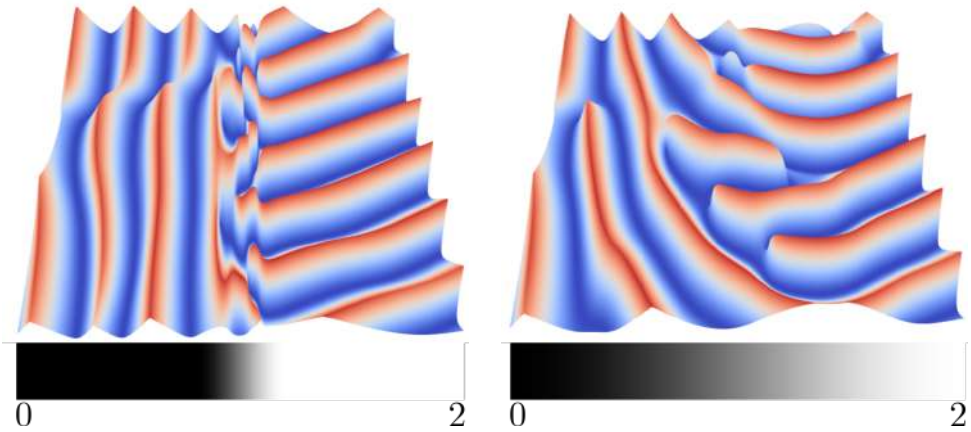
Le champ $\theta(\mathbf{u})$ décrivant l'orientation d'un champ gaussien N correspond à la direction perpendiculaire aux motifs allongés qu'il produit (figure 4.16). Le champ $\theta(\mathbf{u})$ détermine donc la direction orthogonale à celle des ravines.



(a) *Représentation spectrale* **(b)** *Représentation spatiale*

FIGURE 4.16 – *Un bruit gaussien et sa représentation spectrale dans le domaine de Fourier. L'orientation visible dans la représentation spectrale correspond à la direction perpendiculaire à la direction d'élongation des motifs.*

Dans le cadre de notre application à la génération de ravines d'écoulement d'eau, nous avons besoin que le champ d'orientation soit suffisamment "lisse". C'est-à-dire qu'il ne présente pas de variations trop brusque de valeur (figure 4.17). Les méthodes de génération de bruits gaussiens acceptent l'utilisation de champs d'orientation avec des variations brusques, mais cela provoque des discontinuités visibles sous la forme de cisaillements (figure 4.17a).



(a) *Variations brusques* **(b)** *Variations lisses*

FIGURE 4.17 – *Impact de la vitesse de variation du champ d'orientation sur l'apparence des ravines, pour des valeurs d'orientation* $\theta \in [0, 2]$.

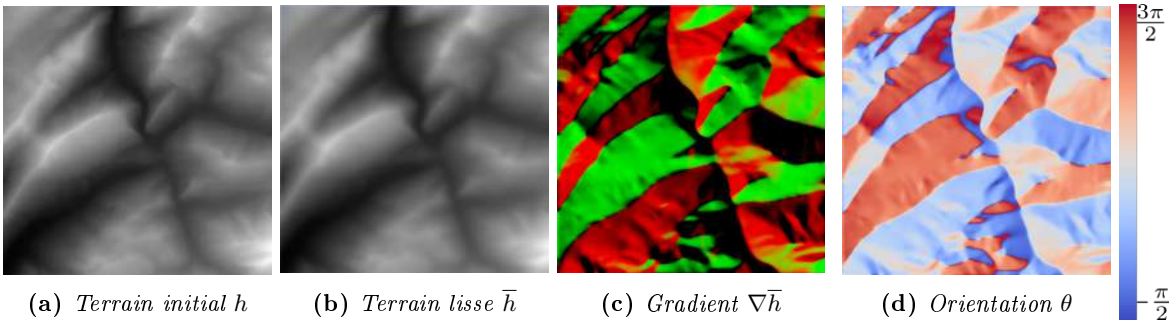
Dans notre cas, nous voulons assurer que l'écoulement de l'eau est possible dans les ravines générées. Celles-ci doivent donc suivre l'orientation de la pente sans s'interrompre ou se croiser brusquement. Dans le cas contraire, cela provoquerait l'apparition de bassins de rétention de l'eau et empêcherait son ruissellement.

Pour éviter ce problème, nous calculons donc une version $\overline{h}$ lissée du terrain initial au moyen d'un filtre gaussien.

$$\overline{h}(\mathbf{u}) = h * \mathcal{G}(\mathbf{u}) \tag{4.16}$$

Finalement, le champ d'orientation θ est obtenu par l'orientation du gradient (figure 4.18) :

$$\theta(\mathbf{u}) = \text{Atan2}\left(\nabla\overline{h}(\mathbf{u})\right) + \frac{\pi}{2} \tag{4.17}$$



(a) *Terrain initial* h **(b)** *Terrain lisse* $\overline{h}$ **(c)** *Gradient* $\nabla\overline{h}$ **(d)** *Orientation* θ

FIGURE 4.18 – *Calcul du champ d'orientation* $\theta(\mathbf{u})$ *des ravines en fonction du gradient du terrain initial lissé* $\nabla\overline{h}(\mathbf{u})$.

Le champ d'orientation peut également être modifié à la main, mais doit prendre en compte le terrain pour être réaliste.

4.4.2 Fréquence

Le champ de fréquence $f(\mathbf{u})$ du bruit gaussien $\mathbf{N}$ influence l'espacement entre les ravines. Le modèle décrit par l'équation (4.10) permet à un utilisateur de définir la carte $f(\mathbf{u})$ entièrement à la main.

Nos expérimentations nous ont menées à considérer qu'un champ de fréquence constant, $f(\mathbf{u}) = \alpha$, donnait des résultats visuellement convaincants. La valeur de la fréquence doit être choisie en fonction de la fréquence globale des montagnes présentes sur le terrain initial (figure 4.19).

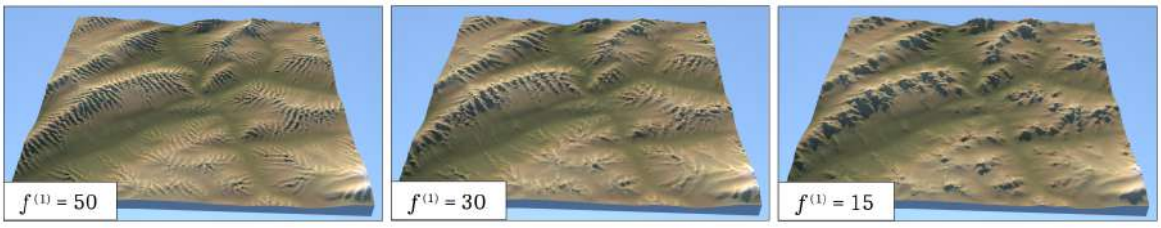
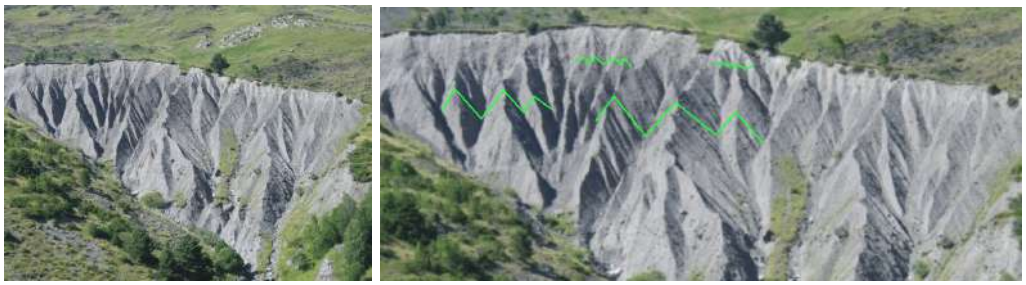


Figure 4.19 – *Le choix de la valeur choisie pour la fréquence à une forte influence sur l'apparence du résultat. Cette valeur doit donc être cohérente avec la fréquence des éléments présents dans le terrain initial.*

Il est également possible d'utiliser un champ de fréquence non constant pour tenir compte des variations dans la fréquence des éléments déjà présents sur le terrain. Intuitivement, nous pouvons considérer que les ravines sont plus nombreuses et moins larges en haut des pentes, puis se regroupent progressivement pour former des ravines moins nombreuses et plus larges en descendant la pente. Ainsi, une valeur de fréquence élevée crée de petits motifs d'érosion (avec une courte longueur d'onde) alors qu'une valeur plus faible génère de larges ravines (avec une plus longue longueur d'onde) comme nous pouvons le voir dans la figure 4.20.



(a) *Piau Engaly, Pyrénées* **(b)** *Fréquence et amplitude des ravines*

Figure 4.20 – *Observation de la fréquence de ravines sur un terrain réel.*

4.4.3 Amplitude

L'amplitude $a(\mathbf{u})$ à une position spatiale donnée, correspond à la profondeur des ravines à cette position. Elle est utilisée en tant que coefficient multiplicateur après avoir défini le profil (équation (4.10)).
La carte $a(\mathbf{u})$ peut être entièrement créée à la main par un utilisateur ou définit en fonction du terrain. Pour proposer un calcul automatique cohérent avec le terrain initial (figure 4.21), notre observation est que, sur un terrain réel, la profondeur des ravines dépend du ruissellement de l'eau.

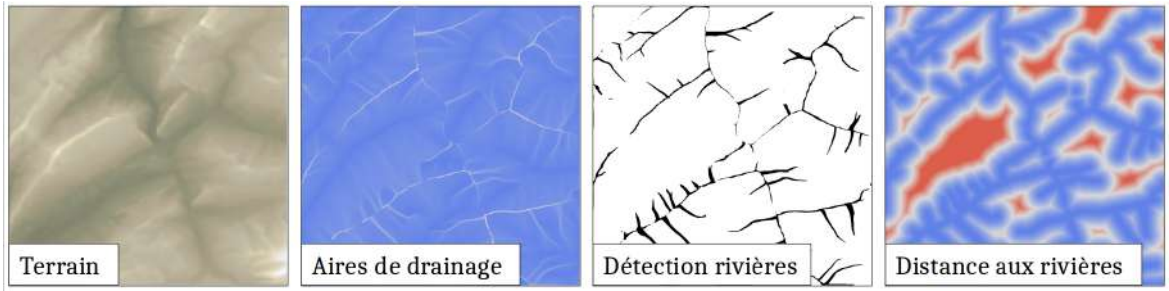


Figure 4.21 – *Calcul automatique d'une carte d'amplitude en utilisant la distance aux rivières présentes dans le terrain d'entrée.*

Ainsi, le ruissellement est plus important lorsque la pente est forte, il va donc transporter plus de matière et creuser des ravines plus profondes. À l'opposé, lorsque la pente devient plus faible, à l'approche du lit des rivières, le ruissellement est moins important et les ravines sont donc moins profondes. D'autre part, la profondeur des ravines dépend également du drainage, c'est-à-dire de la quantité d'eau déplacée. Nous proposons donc une définition automatique de l'amplitude basée sur la distance aux rivières présentes sur le terrain initial h . Pour cela, nous considérons que la carte d'aire de drainage a été calculée en amont sur le terrain h. Cette carte décrit la quantité d'eau déplacée à chaque position. Les valeurs les plus élevées correspondent donc aux rivières et les valeurs les plus faibles aux sommets des pentes. Nous détectons les rivières en appliquant un seuillage sur la carte d'aire de drainage. Puis un champ de distance est appliqué sur le squelette du tracé des rivières. Ce champ de distance est appliqué en tant que carte d'amplitude $a(\mathbf{u})$.

Cette carte d'amplitude calculée automatiquement peut être modifiée localement pour ajuster le résultat (figure 4.22).

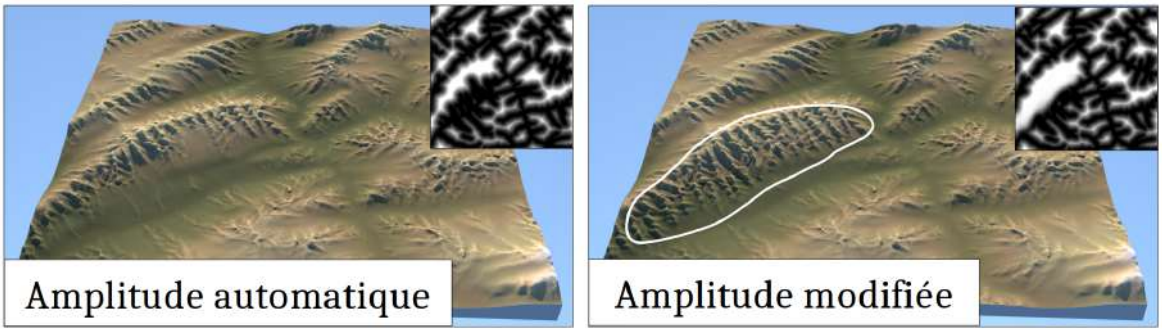


Figure 4.22 – *Utilisation de l'amplitude calculée automatiquement (à gauche) et modification de cette amplitude afin d'ajouter plus de ravines sur une des pentes (entourée en blanc, à droite).*

4.5 Mise en cascade

Le modèle décrit par l'équation (4.10) permet de générer des ravines d'érosion dont nous avons défini l'apparence à travers la fonction H, représentant le profil transversal (section 4.3) et dont nous pouvons contrôler les caractéristiques de fréquence, d'orientation et d'amplitude par le contrôle du champ gaussien $\mathbf{N}$ (section 4.4).

Cependant, les parois des ravines r obtenues avec le premier niveau sont lisses. Sur des terrains réels, l'intérieur des ravines présente des rigoles dont le profil est similaire à celui des ravines principales, un peu à la façon des fractales. Nommons $r^{(1)}$ les ravines principales, $h^{(1)}$ le champ de hauteurs du terrain initial et $\theta^{(1)}, f^{(1)}, a^{(1)}$ les cartes de contrôle correspondantes. Pour simuler les rigoles, nous utilisons le modèle de l'équation (4.10) pour générer un second champ de hauteurs $r^{(2)}(\mathbf{u})$.

$$r^{(2)}(\mathbf{u}) = a^{(2)}(\mathbf{u}) H \circ \mathbf{N}_{f^{(2)},\theta^{(2)}}(\mathbf{u}) \tag{4.18}$$

Ce second champ est ajouté au premier pour créer des rigoles sur les pentes des ravines.

4.5.1 Orientation

Tous comme les ravines du premier niveau, les rigoles du second niveau doivent permettre l'écoulement de l'eau dans le sens de la pente. Cependant, cette fois-ci, l'orientation doit tenir compte de la pente du terrain initial h et de celle des ravines $r^{(1)}$. Ainsi, nous cherchons à avoir

$$\theta^{(2)}(\mathbf{u}) = \text{Atan2}\left(\nabla \overline{h}^{(2)}(\mathbf{u})\right) + \frac{\pi}{2} \tag{4.19}$$

Avec

$$\overline{h}^{(2)}(\mathbf{u}) = \overline{h}^{(1)}(\mathbf{u}) + r^{(1)}(\mathbf{u}) \tag{4.20}$$

Nous avons donc besoin de connaitre le gradient $\nabla\overline{h}^{(2)}(\mathbf{u})$ de cette somme.

Le gradient $\nabla\overline{h}^{(1)}(\mathbf{u})$ du terrain initial est déjà utilisé pour orienter le premier niveau de ravines et est donc déjà connu. Les ravines $r^{(1)}(\mathbf{u})$ sont générées procéduralement en temps réel, nous devons donc calculer leur gradient au moment de cette génération.
Le premier niveau de ravines est défini par la composition

$$r^{(1)}(\mathbf{u}) = a^{(1)}(\mathbf{u}) H \circ \mathbf{N}_{f^{(1)},\theta^{(1)}}(\mathbf{u})$$

Le champ d'amplitude $a^{(1)}$ varie très peu localement, nous le considérons donc comme localement constant et sa dérivée localement nulle. Cela nous permet de simplifier la dérivation du produit par la carte d'amplitude. La dérivée d'une composée de fonctions multi-variées s'écrit, par la règle de la chaine, sous la forme :

$$\nabla r(\mathbf{u}) = a(\mathbf{u}) \mathbf{J}_{\mathbf{N}_{f,\theta}}(\mathbf{u}) \times \nabla H(\mathbf{N}_{f,\theta}(\mathbf{u})) \tag{4.21}$$

Avec $\mathbf{J}_{\mathbf{N}_{f,\theta}}(\mathbf{u})$ la matrice Jacobienne du champ $\mathbf{N}$, ∇H le gradient de la fonction H et $\times$ représentant l'opérateur de produit matriciel.

Dans cette équation, la fonction de transfert H est connue et stockée en amont selon la définition de l'équation (4.15). Cette fonction ne dépend ni de la position $\mathbf{u}$ sur le terrain ni des cartes de contrôle $a(\mathbf{u})$, $f(\mathbf{u})$, $\theta(\mathbf{u})$. Donc son gradient ∇H peut être calculé et stocké en amont par un calcul analytique ou numérique (par des différences finies par exemple).

Le champ gaussien vectoriel $\mathbf{N}$ est généré procéduralement à la volée. Pour calculer analytiquement la dérivée de ce champ, nous considérons que la méthode de synthèse utilisée est une convolution parcimonieuse de noyaux de Gabor [LLDD09]. Mathématiquement, ce champ vectoriel s'écrit sous forme complexe par :

$$\widetilde{N}(\mathbf{u}) = \sum_j w_j \exp^{\left(-\frac{||\mathbf{u}-\mathbf{x}_j||^2}{2\sigma^2}\right)} \exp^{(i2\pi f \Theta \cdot (\mathbf{u}-\mathbf{x}_j))} \tag{4.22}$$

avec $\Theta = \begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix}$.

Par dérivation de l'exponentielle, la matrice Jacobienne complexe du champ gaussien est obtenue par la somme des gradients complexes :

$$\widetilde{\mathbf{J}}_{\widetilde{N}}(\mathbf{u}) = \sum_j w_j \exp^{\left(-\frac{||\mathbf{u}-\mathbf{x}_j||^2}{2\sigma^2}\right)} \exp^{(i2\pi f \Theta \cdot (\mathbf{u}-\mathbf{x}_j))} \left(\frac{-(\mathbf{u}-\mathbf{x}_j)}{\sigma^2} + i2\pi f \Theta\right) \tag{4.23}$$

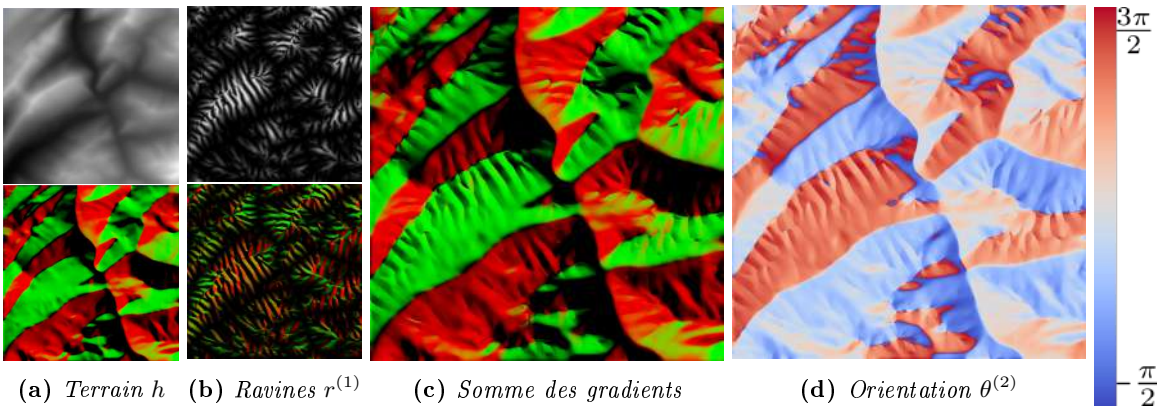
Ce calcul de dérivation est mathématiquement exact dans le cas où les noyaux de Gabor sont dérivables en tous points. Cependant, dans le cas de la génération par convolution parcimonieuse, les gaussiennes utilisées sont tronquées de façon à être à support compact, elles ne sont donc pas dérivables aux bords des noyaux. Le résultat de la dérivation reste toute fois une bonne estimation de la dérivée de la convolution parcimonieuse. En effet, les valeurs de la dérivée d'une fonction gaussienne non tronquée, au-delà du bord des noyaux, sont quasiment nulles. Ainsi, l'impact de ces valeurs est négligeable par rapport aux valeurs des dérivées des autres noyaux de la somme de l'équation (4.23).

Finalement, la matrice Jacobienne réelle de $\mathbf{N}$ est donnée par :

$$\mathbf{J}_{\mathbf{N}_{f,\theta}}(\mathbf{u}) = \begin{bmatrix}\Re\left(\widetilde{\mathbf{J}}_{\widetilde{N}}(\mathbf{u})\right)\\ \Im\left(\widetilde{\mathbf{J}}_{\widetilde{N}}(\mathbf{u})\right)\end{bmatrix}^T \tag{4.24}$$

En injectant la matrice Jacobienne (4.23) et le gradient ∇H dans l'équation (4.21), nous obtenons le gradient du premier niveau de ravines $r^{(1)}$ à la volée.
Grâce à cela, nous avons le champ d'orientation à la volée (figure 4.23) :

$$\theta^{(2)}(\mathbf{u}) = \text{Atan2}\left(\nabla\overline{h}^{(1)}(\mathbf{u}) + \nabla r^{(1)}(\mathbf{u})\right) + \frac{\pi}{2} \tag{4.25}$$



(a) *Terrain* h **(b)** *Ravines* $r^{(1)}$ **(c)** *Somme des gradients* **(d)** *Orientation* $\theta^{(2)}$

Figure 4.23 – *Calcul du champ d'orientation* $\theta^{(2)}$ *du second niveau de ravines en utilisant le gradient du terrain initial (figure 4.23a bas) et celui du premier niveau de ravines (figure 4.23b bas).*

4.5.2 Fréquence

Pour définir la fréquence $f^{(2)}$, nous nous inspirons des travaux de l'hydrologue Robert Horton [Hor45]. En hydrologie, les flux d'un réseau peuvent être classés suivant une valeur appelée "ordre", qui indique leur niveau dans la hiérarchie. Horton attribue une valeur d'ordre 1 au flux se trouvant les plus en altitude et des valeurs croissantes aux flux résultant de regroupement en descendant vers l'aval (figure 4.24).

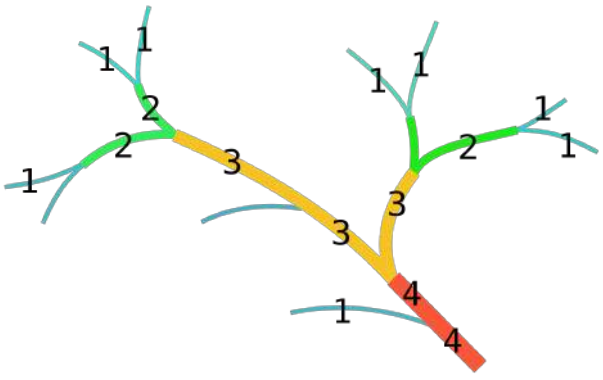


Figure 4.24 – *Ordre des flux d'un réseau en fonction de leur position dans la hiérarchie. Les flux d'ordre* 1 *sont situés les plus en altitude. Image issue de* `Wikipedia`[3].

Son étude montre que, sur une portion de terrain délimitée, la série décrivant le nombre de flux en fonction de leur ordre tend à être une série géométrique dont la raison dépend du nombre de bifurcations. La fréquence des flux d'un certain ordre correspondant au nombre de flux de cet ordre divisé par l'aire de la portion de terrain étudiée, cela montre que la série décrivant les fréquences tend également à être une série géométrique. Ces observations nous indiquent que la fréquence du

3. https://fr.wikipedia.org/wiki/Ordre_des_cours_d'eau

second niveau de ravines $r^{(2)}$ est proportionnelle à celle du premier niveau $r^{(1)}$. Dans la pratique, nous utilisons le rapport

$$f^{(2)}(\mathbf{u}) \approx 3f^{(1)}(\mathbf{u}) \tag{4.26}$$

4.5.3 Amplitude

Pour définir l'amplitude $a^{(2)}$, nous nous basons sur deux arguments géomorphologiques. Le premier est que les ravines $r^{(1)}$ et les rigoles $r^{(2)}$ doivent apparaître dans les mêmes zones et être approximativement proportionnelles. Le second est que les rigoles doivent disparaître au fond des ravines dans lesquelles l'eau ruisselle. Nous proposons ici de considérer

$$a^{(2)}(\mathbf{u}) = \alpha r^{(1)}(\mathbf{u}) \tag{4.27}$$

Le champ de hauteur $r^{(1)}$ contient à la fois l'amplitude déterminée à partir de la distance aux rivières du terrain initial et la hauteur des ravines principales. En effet, il est de valeur maximale au sommet des ravines principales et diminue pour devenir nul aussi bien au fond des ravines qu'au fond des rivières du terrain initial. Le paramètre α est une constante de proportionnalité. Empiriquement, nous utilisons $\alpha = 0.3$. Cette mise en cascade génère des rigoles sur les pentes des ravines principales (figure 4.25).

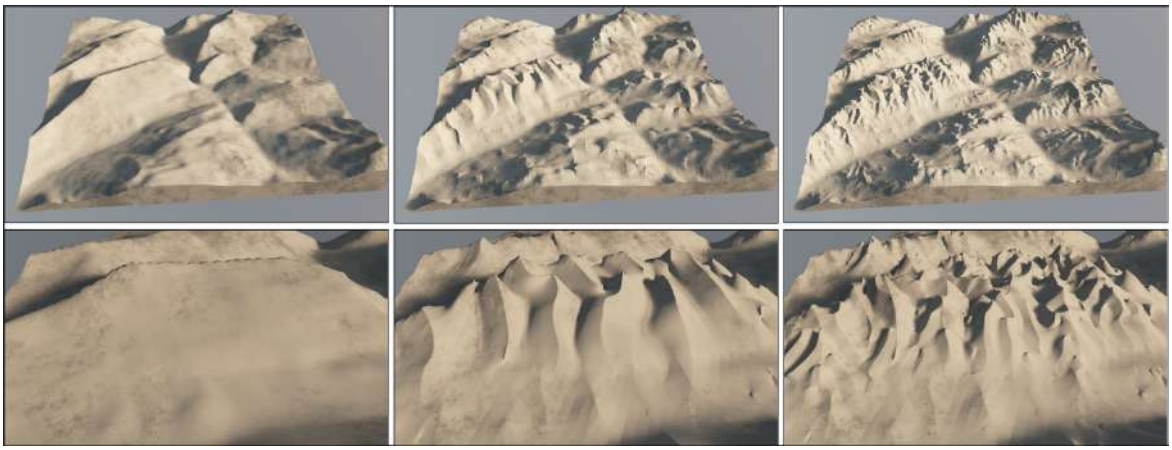


Figure 4.25 – *Le terrain initial h (à gauche) est augmenté avec un premier niveau de ravines $r^{(1)}$ (au centre), un second niveau $r^{(2)}$ est ajouté en cascade (à droite) pour améliorer le réalisme. La ligne du haut montre le terrain dans son ensemble et celle du bas une vue plus rapprochée.*

4.6 Les lignes de crête

Le modèle d'augmentation que nous avons obtenu est constitué de la somme de deux niveaux de motifs d'érosion : un niveau de ravines $r^{(1)}$ et un réseau de rigoles secondaires $r^{(2)}$. Le réseau de rigoles permet de rendre les pentes des ravines plus réalistes en y ajoutant des détails. Cependant, les sommets des ravines restent trop lisses. Comme il s'agit de parties du terrain où l'eau ne s'écoule pas, des irrégularités devraient être présentes. Nous ajoutons donc un bruit fractal au sommet des ravines pour améliorer le résultat final. Pour cela, nous utilisons un mouvement Brownien fractionnaire ou fBm (de l'anglais *Fractional Brownian motion*), noté fBm($\mathbf{u}$). Celui-ci permet la génération de motifs aléatoires présentant des caractéristiques fractales. Par addition de plusieurs octaves d'un bruit de Perlin de fréquence croissante et d'amplitude décroissante.

$$\text{fBm}(\mathbf{u}) = \sum_i 0.5^i a_0 \text{bruit}(2^i f_0 \mathbf{u}) \tag{4.28}$$

Avec a_0 et f_0 une amplitude et une fréquence initiale choisies en fonction du terrain.

Cette méthode permet de générer un champ aléatoire présentant des détails fins très similaires à ceux observables sur les crêtes de montagnes. Dans la figure 4.26, la ligne de crête d'une montagne réelle (à gauche) est comparée avec un bruit fractal. Pour cela, une segmentation est faite pour séparer la crête du ciel, puis cette segmentation est utilisée comme un signal sonore pour en analyser le spectre. Le spectre de la crête (en violet) présente une décroissance d'amplitude en fonction de la fréquence, comparable à celle d'un bruit brun (en bleu). Cette décroissance (entre 6 et 9 décibels par octave) est couramment utilisée pour la génération de bruit fractal tel que le fBm.

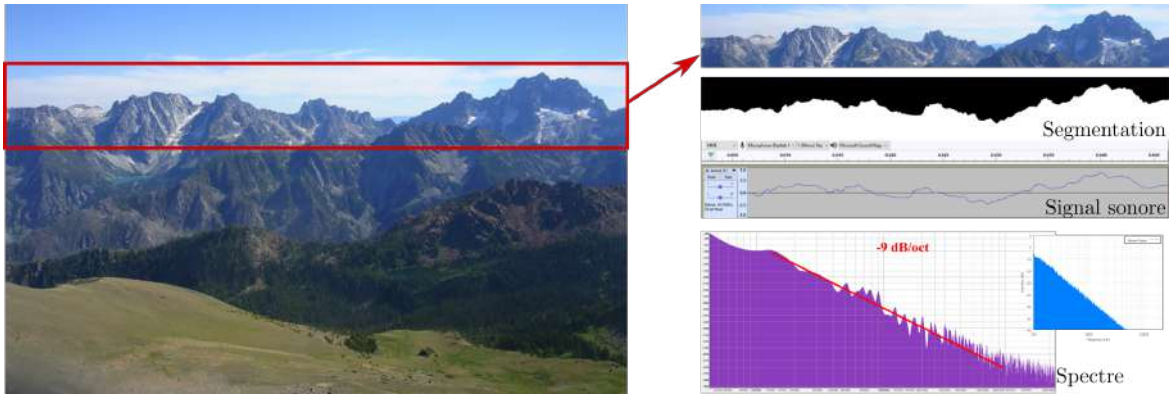


Figure 4.26 – *Comparaison spectrale entre les crêtes d'une montagne réelle et un bruit brun. Image issue de* `iquilezles.org`[4].

Nous voulons ajouter ces détails uniquement au sommet des ravines, nous devons donc discriminer les différentes parties des ravines. Comme nous l'avons vu lors de la définition du profil des ravines (section 4.3.1), le paramètre de phase φ, définissant le profil angulaire γ, correspond à la position transversale dans une ravine. Nous pouvons donc l'utiliser pour distinguer le sommet des ravines (en $\varphi = \pm\pi$) du fond (en $\varphi = 0$). Pour que le bruit des crêtes apparaisse progressivement dans la pente des ravines (pour ne pas provoquer de discontinuités), nous utilisons une fonction de transition lisse pour contrôler son amplitude (figure 4.27). Cette fonction est à valeur entre 0 et 1, symétrique par rapport à l'axe des ordonnées et croissante sur $[0, \pi]$.

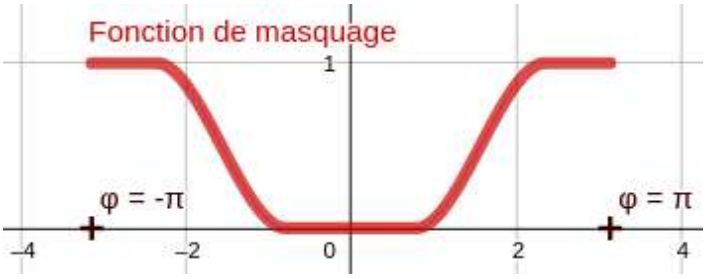


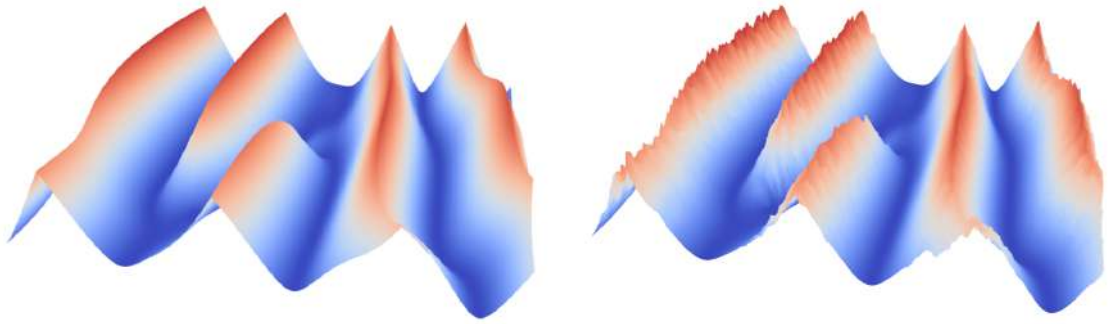
Figure 4.27 – *Profil de la fonction de masquage* $\text{Smoothstep}(|\text{Atan2} \circ \mathbf{N}_{f,\theta}|)$ *permettant de contrôler l'amplitude du bruit ajouté au sommet des crêtes.*

Finalement, les motifs aléatoires $C(\mathbf{u})$ au sommet des crêtes sont définis par :

$$C(\mathbf{u}) = \text{Smoothstep}(|\text{Atan2} \circ \mathbf{N}_{f,\theta}|)\text{fBm}(\mathbf{u}) \tag{4.29}$$

De cette façon, le bruit fractal est ajouté au sommet des ravines principales, sans affecter le reste du profil (figure 4.28).

4. https://iquilezles.org/articles/fbm/. Crédit photographie : https://en.wikipedia.org/wiki/Stuart_Range



(a) *Ravines lisses* **(b)** *Bruit sur les crêtes*

Figure 4.28 – *Comparaison des crêtes du premier niveau de ravines sans (gauche) et avec (droite) le bruit C. L'ajout de ce bruit permet de casser le côté lisse des lignes de crêtes et d'ajouter du réalisme.*

4.7 Implémentation

4.7.1 Implémentation du modèle complet

La méthode que nous avons développée dans les sections précédentes nous permet d'augmenter le réalisme d'un terrain virtuel en lui ajoutant des ravines d'érosion procéduralement (figure 4.29). Ces ravines $r^{(1)}$ sont décrites par l'application d'une fonction de transfert bi-dimensionnelle H (section 4.2) décrivant le profil, sur un champ vectoriel gaussien guidé par des cartes de contrôle f, θ, a (section 4.4). Afin d'améliorer la qualité visuelle du résultat, un second niveau de rigoles $r^{(2)}$ est ajouté sur leurs pentes (section 4.5) et un bruit fractal C est ajouté au sommet des ravines (section 4.6).

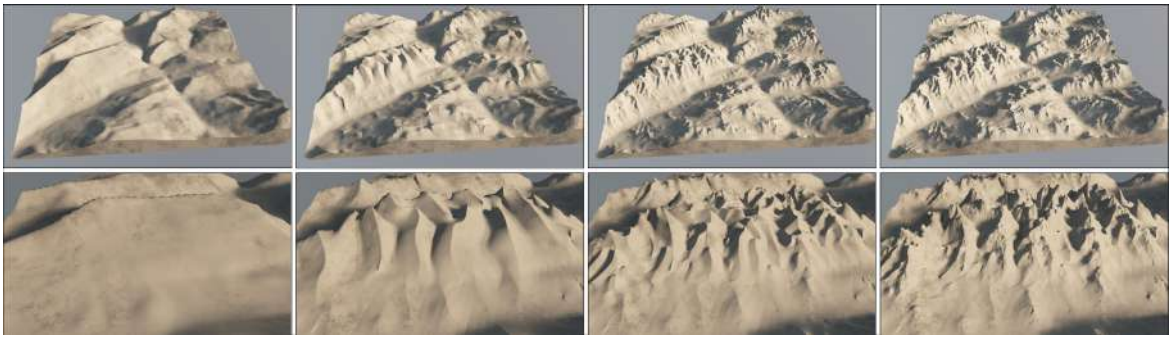


Figure 4.29 – *Résultat final de notre augmentation de terrain. La première ligne montre le terrain dans sa globalité alors que la seconde montre une vue de plus près. De gauche à droite, les vignettes montrent : le terrain initial, l'ajout du premier niveau de ravines, l'ajout du second niveau de ravines et l'ajout du bruit au sommet de crêtes.*

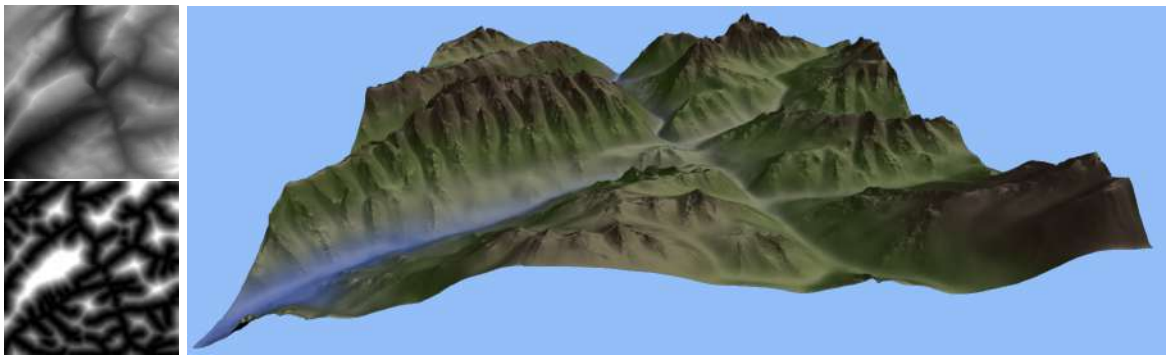
Pour permettre une animation et la manipulation en temps réel, nous avons réalisé une implémentation en WebGL de notre modèle. De façon à ne pas re-calculer des éléments inchangés à chaque image, nous avons divisé le code en plusieurs étapes en utilisant des *framebuffer objects*.

Initialisation. Nous utilisons deux champs scalaires comme données d'entrée : le champ de hauteurs h décrivant le terrain et un champ de distances décrivant le réseau de rivières correspondant (utilisé pour le contrôle de l'amplitude). Ces deux champs sont stockés sous forme d'images en nuances de gris. Notre modèle s'applique mieux sur des champs ne présentant pas de variation haute fréquence, ce qui nous permet d'utiliser des images de taille raisonnable (400^2 *pixels* pour le champ de hauteur et 100^2 *pixels* pour le champ de distance). Afin de s'assurer que le champ de hauteurs ne présente pas de haute fréquence, un champ lissé $\overline{h}$ est calculé par convolution de h avec un noyau gaussien. Le gradient $\nabla\overline{h}$ du terrain est également calculé, par différences finies.

Génération. La génération procédurale des détails est ensuite réalisée en deux étapes, utilisant les données de l'initialisation. La première étape calcule les ravines $r^{(1)}$, les rigoles $r^{(2)}$ et leurs gradients $\nabla r^{(1)}$ et $\nabla r^{(2)}$, ainsi que le bruit sur le sommet des crêtes C. La seconde étape calcule le champ de hauteur du terrain augmenté T et son gradient ∇T.

Rendu. Enfin, le champ de hauteur du terrain augmenté est appliqué sur un maillage de grille régulière à la façon d'une carte de déplacement (ou *displacement map* en anglais) et les interactions lumineuses sont définies en utilisant le champ de hauteurs T et son gradient ∇T.

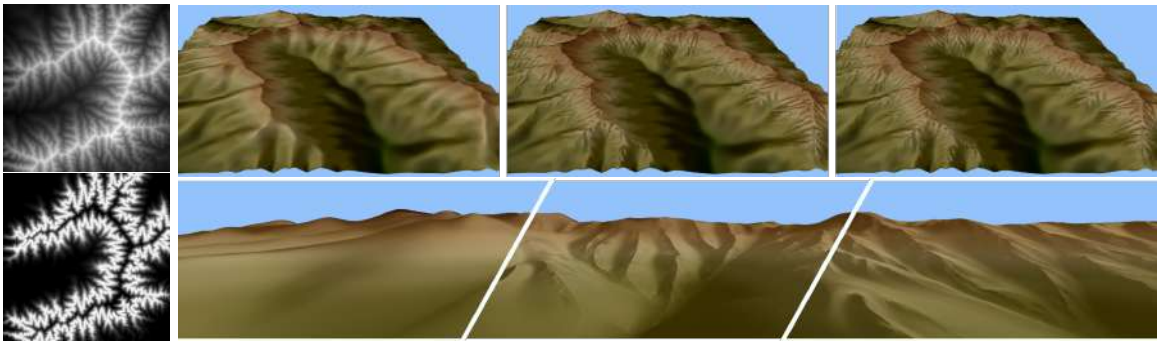
Les figures 4.30 et 4.31 montrent des résultats de l'implémentation faite en WebGL. Pour chacune des figures, les vignettes à gauche (figure 4.30a et 4.31a) montrent les données d'entrée utilisées. Il s'agit du champ de hauteurs décrivant le terrain (en haut) et du champ d'amplitude $a(\mathbf{u})$, définit selon les caractéristiques du terrain.



(a) *Entrées* **(b)** *Vue 3D du résultat*

Figure 4.30 – *Résultat de l'augmentation d'un terrain. À gauche (4.30a), les données d'entrée comprennent le champ de hauteurs décrivant le terrain (en haut) et la distance aux rivières (en bas) utilisée comme carte de contrôle de l'amplitude.*

Dans la figure 4.31, la vue 3D du résultat montre trois étapes de l'amplification : le terrain initial, le premier niveau de ravines avec le bruit au sommet des crêtes et enfin le second niveau de ravines. La ligne du haut montre une vue d'ensemble du terrain et la ligne du bas une vue à la première personne.



(a) *Entrées* **(b)** *Vue 3D du résultat*

Figure 4.31 – *Résultat de l'augmentation d'un terrain. À gauche (4.31a), les données d'entrée comprennent le champ de hauteurs décrivant le terrain (en haut) et la distance aux rivières (en bas) utilisée comme carte de contrôle de l'amplitude. À droite (4.31b), le résultat du rendu montre une vue générale (en haut) et une vue rapprochée (en bas). Les différentes vignettes montrent les étapes de l'amplification : terrain initial, premier niveau de ravines avec du bruit au sommet des crêtes, second niveau de ravines.*

Le pipeline que nous utilisons pour réaliser l'augmentation nous permet de ne faire l'étape d'initialisation qu'une seule fois, de générer les détails à la demande lors de la mise à jour des paramètres du modèle et de ne faire que l'étape de rendu à chaque image. Pour mesurer le temps de calcul nécessaire avec notre implémentation en WebGL, nous avons réalisé 100 fois l'augmentation d'un terrain initial avec des détails d'une résolution de 2048^2 *pixels*. Sur une carte NVidia GTX 980, ces 100 générations de détails sont faites en 4 millisecondes. À titre de comparaison, le calcul de 100 générations de détails par un bruit fractal à 6 octaves prend 3 millisecondes sur la même carte graphique.

4.7.2 Méthode de synthèse

Attardons-nous un instant sur le choix de la méthode de synthèse pour le champ gaussien vectoriel $\mathbf{N}$. Dans le chapitre précédent (section 3.3) nous avons fait le choix de la méthode par pavage et mélange présentée par Heitz et collègues [HN18]. L'utilisation de cette méthode nous a permis de présenter des estimateurs pour la moyenne, la variance et la covariance en temps réel. L'estimation de ces statistiques était nécessaire pour le filtrage en temps réel du résultat de la composition $H \circ \mathbf{N}$. Ici, nous choisissons d'utiliser la méthode de synthèse par convolution parcimonieuse de noyaux de Gabor présentée par Lagae et collègues [LLDD09]. L'avantage de cette méthode, dans ce cadre, est qu'elle utilise une convolution parcimonieuse de noyaux continus. En effet, la méthode par pavage et mélange utilise des tuiles tirées dans un exemple discret. Cette discrétisation risquerait de poser problèmes, notamment pour la précision du calcul de dérivation.

4.7.3 Normale au terrain augmenté

Au moment du rendu du terrain final, les informations nécessaires sont le champ de hauteurs T mais également sa normale en tous points $\mathbf{u}$. En effet, l'information de normale permet de déterminer l'effet de la lumière sur la géométrie du terrain. Dans le cas d'un champ de hauteurs entièrement connu en mémoire, il est possible de calculer la normale par dérivation de ce champ. Dans notre cas, les détails d'érosion ajoutés sont calculés à la volée et ne sont donc pas connus en mémoire en amont du rendu. Nous devons donc calculer la normale au terrain augmenté à la volée, lors de la génération des détails. Le terrain final T est obtenu par la somme du terrain initial h, des ravines $r^{(1)}$ et rigoles $r^{(2)}$, et du bruit C au sommet des crêtes des ravines. Par linéarité de la dérivée, le gradient ∇T est égal à la somme des gradients ∇h, $\nabla r^{(1)}$, $\nabla r^{(2)}$ et ∇C.
Le champ de hauteurs h est entièrement connu en amont de l'ajout de détails et sa dérivée est déjà utilisée pour orienter les deux niveaux de ravines (sections 4.4.1 et 4.5.1). Nous considérons donc que ∇h est pré-calculé et stocké en mémoire (figure 4.32a). La méthode permettant d'obtenir le gradient des ravines $r^{(1)}$ a été présentée dans la section 4.5.1 pour permettre d'orienter les rigoles $r^{(2)}$ en tenant compte de la pente de ravines (figure 4.32b). En reprenant l'équation (4.21), nous avons donc :

$$\nabla r^{(1)}(\mathbf{u}) = a^{(1)}(\mathbf{u}) \mathbf{J}_{\mathbf{N}_{f^{(1)},\theta^{(1)}}}(\mathbf{u}) \times \nabla H(\mathbf{N}_{f^{(1)},\theta^{(1)}}(\mathbf{u})) \tag{4.30}$$

Le bruit C, ajouté au sommet des crêtes du premier niveau de ravines, obtenu par le produit d'un bruit fractal fBm par une amplitude variant spatialement (équation (4.29)). Le bruit fractal fBm est entièrement indépendant de la génération des ravines, son gradient peut donc être calculé à la volée en accumulant les gradients des différentes octaves. Par dérivation de l'équation (4.28), nous obtenons

$$\nabla \text{fBm}(\mathbf{u}) = \sum_i 0.5^i a_0 2^i f_0 \nabla \text{bruit}(2^i f_0 \mathbf{u}) \tag{4.31}$$

Le bruit utilisé est obtenu par interpolation entre des valeurs aléatoires placées sur une grille discrète, le gradient ∇bruit peut donc être calculé en utilisant la dérivée de la fonction d'interpolation. De plus, les variations spatiales de l'amplitude ont une fréquence basse en comparaison de la fréquence de fBm, nous pouvons donc les considérer comme localement constantes (et donc considérer la dérivée de l'amplitude comme nulle). Donc, en reprenant l'équation (4.29), nous obtenons (figure 4.32c) :

$$\nabla C(\mathbf{u}) = \text{Smoothstep}(|\text{Atan2} \circ \mathbf{N}_{f,\theta}|) \nabla \text{fBm}(\mathbf{u}) \tag{4.32}$$

Pour obtenir le gradient du second niveau de ravines $r^{(2)}$ (figure 4.32d), nous appliquons la même méthode que pour $\nabla r^{(1)}$. En effet, celles-ci sont générées de façon similaire. Ainsi, la fonction de

transfert H définissant le profil des rigoles est la même que celle des ravines et la synthèse du champ gaussien $\mathbf{N}$ repose également sur une convolution parcimonieuse de noyaux de Gabor. Seules les cartes de contrôle diffèrent de celles utilisées pour la génération des ravines. Nous avons donc :

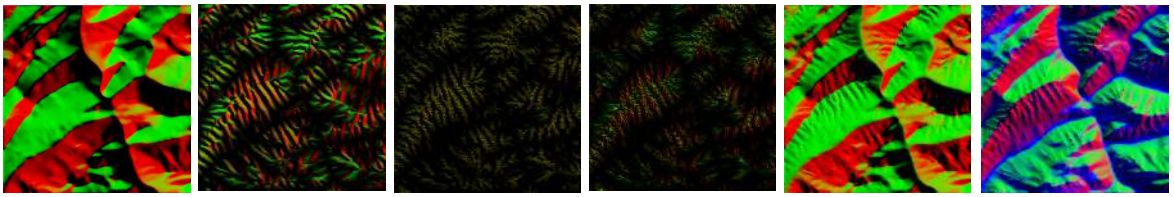
$$\nabla r^{(2)}(\mathbf{u}) = a^{(2)}(\mathbf{u})\mathbf{J}_{\mathbf{N}_{f^{(2)},\theta^{(2)}}}(\mathbf{u}) \times \nabla H(\mathbf{N}_{f^{(2)},\theta^{(2)}}(\mathbf{u})) \tag{4.33}$$

Finalement, le gradient ∇T du terrain augmenté est obtenu par (figure 4.32e) :

$$\nabla T(\mathbf{u}) = \nabla h(\mathbf{u}) + \nabla r^{(1)}(\mathbf{u}) + \nabla r^{(2)}(\mathbf{u}) + \nabla C(\mathbf{u}) \tag{4.34}$$

Et la normale ΛT au terrain est obtenue par (figure 4.32f) :

$$\Lambda T(\mathbf{u}) = \frac{1}{\sqrt{||\nabla T(\mathbf{u})||^2 + 1}} \begin{bmatrix} -\nabla T(\mathbf{u})_x \\ -\nabla T(\mathbf{u})_y \\ 1 \end{bmatrix} \tag{4.35}$$

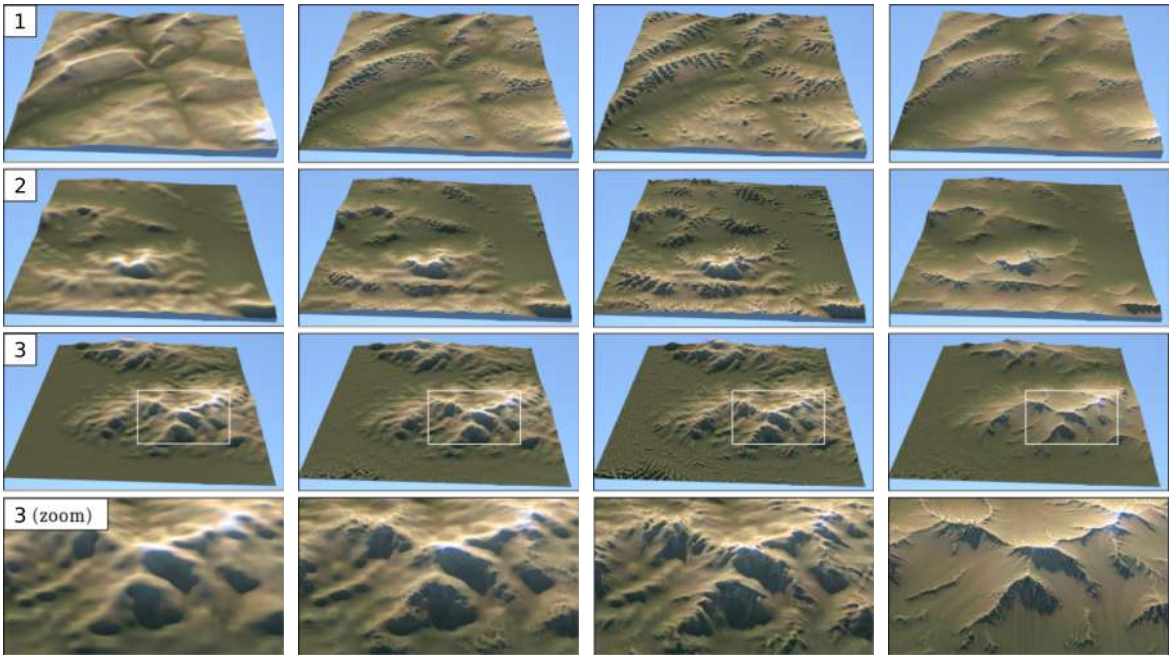


(a) ∇h **(b)** $\nabla r^{(1)}$ **(c)** ∇C **(d)** $\nabla r^{(2)}$ **(e)** ∇T **(f)** ΛT

FIGURE 4.32 – *Étapes de calcul de la normale au terrain augmenté T par la somme des gradients.*

4.8 Validation

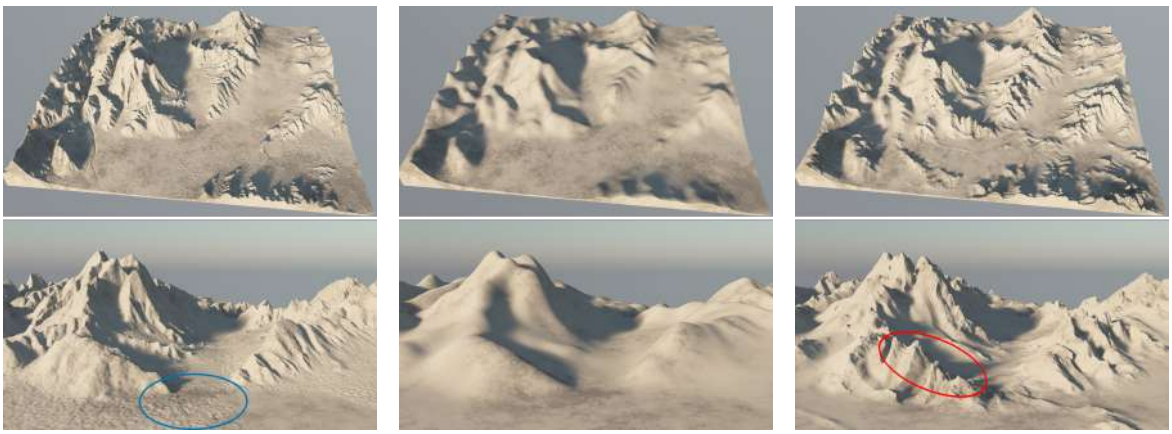
Notre méthode de génération de $r^{(1)}$ et $r^{(2)}$ nous permet de calculer ces champs de hauteurs en temps réel et en parallèle en tout point sur le processeur graphique. Dans le cadre de l'amplification de terrain, seules les méthodes basées sur des bruits procéduraux [EMP+02] présentes des caractéristiques de performance similaires. Nous réalisons donc une comparaison entre les résultats de ces méthodes et la nôtre. Pour cela, nous utilisons un même champ de hauteurs en entrée pour les deux méthodes, ainsi qu'une même carte d'amplitude basée sur la distance aux rivières, afin que les détails soient ajoutés aux mêmes endroits.
Dans la figure 4.33, les trois premières lignes présentent trois terrains initiaux testés et la quatrième ligne montre un zoom sur les détails de la troisième ligne. La première colonne représente les champs de hauteurs utilisés en entrée. La deuxième colonne montre les résultats obtenus par utilisation de la valeur absolue d'un bruit fractal (figure 4.33b). Cette méthode, appelée "bruit à crête" (ou *ridge noise* en anglais), est couramment utilisée pour obtenir des motifs structurés simulant des éléments naturels, comme des crêtes de montagnes. La troisième colonne montre les résultats obtenus avec notre méthode (figure 4.33c). Le bruit à crêtes permet d'ajouter des motifs structurés, mais ceux-ci sont isotropes et ne sont pas orientés dans le sens des pentes du terrain initial. De son côté, notre méthode adapte l'orientation des ravines en fonction du terrain d'entrée.
Dans la quatrième colonne (figure 4.33d), nous effectuons également une comparaison entre nos résultats et ceux obtenus par une simulation d'érosion. Cette méthode de simulation est réputée pour générer des détails réalistes, mais ne les calcule pas en temps réel. De plus, lorsqu'elle est appliquée sur un terrain lisse, cette méthode tend à creuser le terrain sans créer des motifs similaires à des ravines. En effet, la simulation calcul les déplacements de matière qui vont être provoqués par le ruissellement de l'eau. Ainsi, elle a besoin de la présence de détails haute fréquence pour simuler le déplacement de matière. Pour pouvoir effectuer une comparaison, nous réalisons une étape de pré-calcul pour ajouter un bruit haute fréquence sur le terrain d'entrée, localisé selon notre carte d'amplitude. Les résultats de la simulation montrent que cette méthode modifie significativement la forme du terrain initial lors de l'ajout des motifs d'érosion. De son côté, notre méthode garantit la conservation de la topographie du terrain initial, étant donné que celui-ci n'est pas modifié, mais seulement augmenté.



(a) *Terrain initial* **(b)** *Bruit* fBm **(c)** *Notre augmentation* **(d)** *Simulation*

Figure 4.33 – *Comparaison du résultat de l'augmentation des terrains avec différentes méthodes. Pour chaque ligne, la première colonne montre le terrain initial, les colonnes suivantes montrent le résultat de l'augmentation avec un bruit* fBm *en bruit à crête, notre méthode et une simulation d'érosion. Les lignes 1, 2 et 3 montrent trois terrains d'entrée différents ; la dernière ligne montre un zoom sur le troisième terrain (encadré en blanc).*

Nous souhaitons également comparer nos résultats avec un terrain réel. Dans la figure 4.34, la première colonne présente le terrain de référence avec des détails haute définition. Nous lissons ensuite ce terrain pour obtenir le champ de hauteur basse définition qui est représenté dans la deuxième colonne. Ce champ lissé est enfin utilisé comme donnée d'entrée pour notre méthode, dont le résultat est présenté dans la troisème colonne. Pour chaque colonne, la première ligne montre le terrain complet et la seconde ligne montre une vue plus rapprochée.

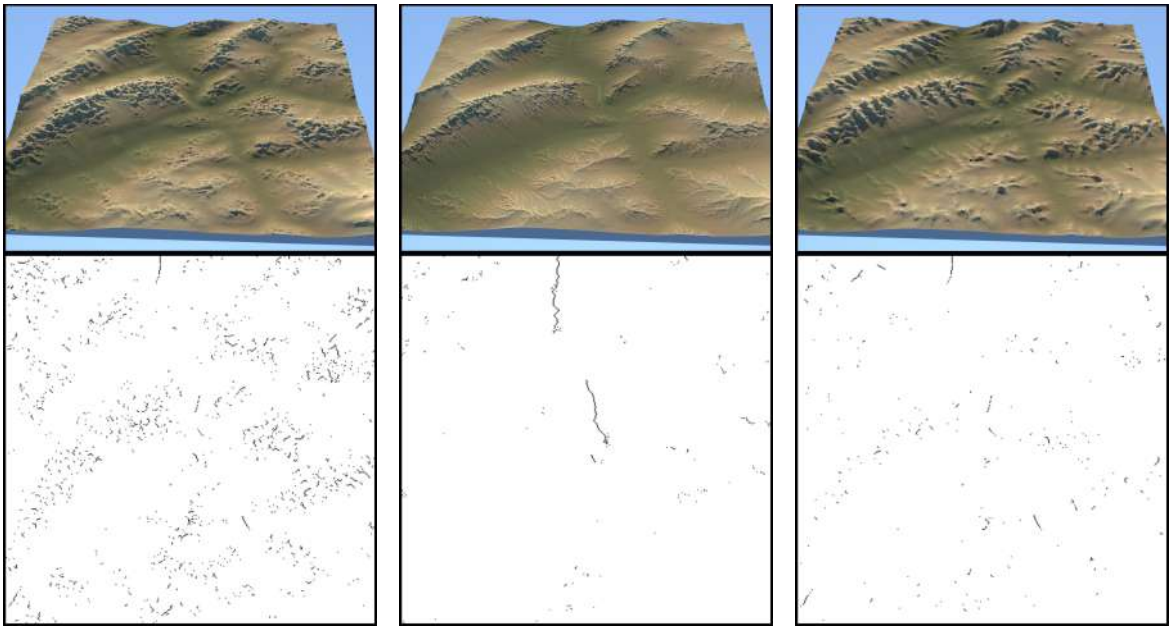


(a) *Terrain initial* **(b)** *Terrain lissé* **(c)** *Notre augmentation*

Figure 4.34 – *Ré-augmentation d'un terrain réel pour comparaison. Notre augmentation parvient à retrouver des détails cohérents avec le terrain lisse et d'esprit similaire à ceux du terrain initial.*

Les détails d'amplification obtenus par notre méthode ne sont évidemment pas identiques à ceux initialement présents sur le terrain de référence. Cependant, ils sont visuellement similaires. Nous pouvons également remarquer que notre méthode ajoute des motifs abrupts sur des éléments du terrain qui étaient initialement arrondis (figure 4.34c, entouré en rouge). Cela est dû à la méthode de calcul de la carte d'amplitude. En effet, dans notre implémentation, celle-ci ne prend en compte que la distance aux rivières. L'amplitude est donc maximale au sommet des pentes. Cet effet pourrait être corrigé en retouchant manuellement la carte contrôlant l'amplitude. Enfin, les détails initialement présents sur les zones plates du terrain ne sont pas dues à de l'érosion (figure 4.34a, entouré en bleu). Ils ne peuvent donc pas être reproduits par notre méthode, étant donné qu'elle vise à ajouter des détails principalement dans les pentes.

Finalement, nous mesurons l'impact des ravines générées par notre méthode sur l'écoulement d'eau. Pour cela, nous utilisons un algorithme dit "de rupture de charge". Celui-ci vise à analyser un terrain virtuel pour en retirer tous les éléments de matière qui empêcherait un volume d'eau de s'écouler entièrement jusqu'en dehors du domaine délimité [SPF+23]. Pour quantifier la qualité de nos résultats, nous utilisons comme métrique la quantité de matière retirée par cet algorithme. La figure 4.35 montre la localisation de la matière retirée par l'algorithme pour les différentes méthodes comparées précédemment (amplification par un bruit, simulation d'érosion et notre méthode) en utilisant le terrain initial de la figure 4.33(1).



(a) *Bruit* fBm **(b)** *Simulation* **(c)** *Notre méthode*

Figure 4.35 – *Reprise du premier terrain de la figure 4.33. Pour chaque méthode d'amplification, la carte en bas montre la localisation de la matière retirée pour permettre l'écoulement de l'eau.*

La simulation est utilisée comme référence. L'augmentation par un bruit fBm est décorrélée des pentes du terrain et ne favorise donc pas l'écoulement, le résultat nécessite donc de retirer de la matière de façon assez uniforme dans toutes les zones qui ont été amplifiées. Notre méthode nécessite également de retirer de la matière, mais en quantité plus faible et de façon plus localisée.

Le tableau 4.1 présente la mesure des quantités de matière retirées, pour chacun des cas présentés dans la figure 4.33. Plus la quantité de matière retirée est faible, plus le terrain testé est considéré comme réaliste. La méthode par simulation d'érosion est la seule visant directement à favoriser l'écoulement de l'eau. Sans surprise, c'est donc elle qui obtient les meilleurs résultats. Nous l'utilisons donc comme référence pour comparer les résultats de la méthode par bruit fractal et la nôtre. La méthode ajoutant du bruit fractal est celle donnant les moins bons résultats. En effet, elle tend seulement à

ajouter des détails haute fréquence et ne prend pas en compte la possibilité d'écoulement d'eau sur les pentes. Notre méthode obtient des résultats intermédiaires à l'ajout de bruit et à la simulation.

Visuel correspondant	Bruit fractal	Simulation d'érosion	Notre méthode
Figure 4.33 (1)	7.4	2	2.1
Figure 4.33 (2)	8.9	4.8	6.8
Figure 4.33 (3)	128	25	86

Table 4.1 – *Mesure de la quantité de matière retirée pour permettre l'écoulement de l'eau jusqu'en dehors du terrain délimité.*

4.9 Discutions et limitations

Notre objectif était de présenter une méthode d'augmentation de terrains utilisant des motifs procéduraux. Dans cette optique, nous avons présenté un modèle basé sur le bruit *phasor* permettant d'imiter des ravines d'érosion hydraulique. La reformulation du bruit *phasor* en utilisant une fonction de transfert à deux dimensions nous permet de contrôler simplement les différents aspects du résultat. Ainsi, l'apparence des ravines est contrôlée à travers la fonction de transfert H. Indépendamment de cela, leurs direction, fréquence et amplitude sont contrôlées par le guidage du champ gaussien $\mathbf{N}$ par des cartes de contrôle. Plusieurs éléments peuvent être intéressants à étudier pour améliorer encore ce modèle.

4.9.1 Variations spatiales

Fréquence. Nos études pour la validation se concentrent principalement sur le contrôle de l'orientation des motifs, de façon à favoriser l'écoulement de l'eau. Au cours de nos tests, nous avons choisi de définir la fréquence constante sur tout le terrain. Cependant, la carte contrôlant la fréquence peut également être définie avec des variations spatiales. Le ratio de fréquence entre le premier et le second niveau pourrait également varier spatialement.

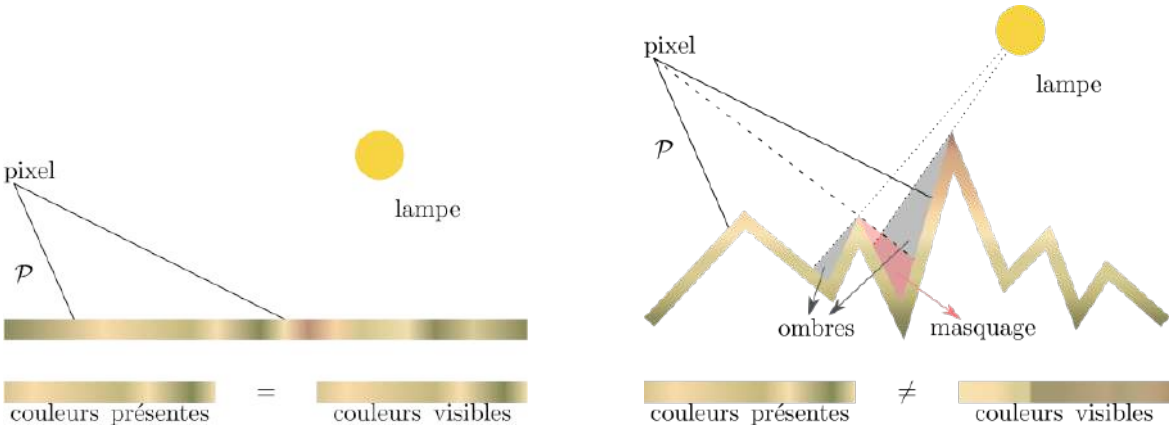
Profil. La fonction γ décrit le profil transversal des ravines. Dans ce chapitre, nous avons choisi d'utiliser un profil triangulaire au fond arrondi. Cependant, d'autres formes de profil peuvent être envisagées. Il serait également intéressant d'étudier la possibilité de faire varier le profil spatialement sur le terrain. Cependant, cette variation apporterait une difficulté supplémentaire lors du calcul du gradient $\nabla r^{(1)}$. En effet, pour ce calcul, la fonction de transfert H, contenant le profil γ, est considérée comme connue en mémoire et son gradient calculé en amont. D'autre part, utiliser un profil variant spatialement nécessiterait de s'assurer de la possibilité de transition d'un profil à un autre sans provoquer de discontinuités ni dans les ravines principales, ni dans les ravines secondaires.

Contrôle artistique. Une partie importante de ces variations spatiales est qu'elles puissent être contrôlées facilement par un artiste. Le modèle permet à un utilisateur de créer ou de modifier les cartes de contrôle, comme nous l'avons vu pour l'amplitude dans la section 4.4.3. Réaliser une série de tests d'utilisations du modèle par des artistes permettrait de déterminer si ces paramètres de contrôle sont intuitifs à utiliser.

4.9.2 Limitations

Limites de la modélisation. Une limitation intrinsèque à ce modèle est le choix d'utiliser des champs de hauteurs pour représenter les terrains. En effet, cette modélisation ne permet pas de représenter l'intégralité du spectre de terrains possibles. Ainsi, il n'est pas possible de représenter des grottes, des surplombs ou des réseaux de cavernes. Cependant, le ruissellement de l'eau sur ce type de terrain se fait de façon très différente. Le modèle ne serait donc pas applicable dans ces cas.

Limite de la génération. Enfin, nous nous intéressons à la génération procédurale de détails en temps réel, mais pas à leur filtrage. La méthode de filtrage de la composition par une fonction de transfert, présentée dans le chapitre 3, est applicable dans le cas de textures contenant des informations de couleurs. Dans ce chapitre, nous avons utilisé cette fonction pour créer de la géométrie. Le filtrage du résultat nécessiterait donc de prendre en compte les effets de masquage et d'ombrage apportés par la géométrie générée (figure 4.36).



(a) *Surface plate* **(b)** *Présence de géométrie*

FIGURE 4.36 – *L'ajout de géométrie nécessite de prendre en compte les ombres et les zones masquées pour déterminer la couleur moyenne dans l'empreinte d'un pixel.*

Ainsi, dans le cas de l'estimation de la couleur moyenne sur une surface plate (figure 4.36a), les couleurs présentes dans l'empreinte $\mathcal{P}$ sont les mêmes que les couleurs visibles. Cependant, après l'ajout de géométrie (figure 4.36b) certaines zones sont masquées par la géométrie et d'autre sont assombries par des ombres, si bien que les couleurs présentes sur le terrain dans l'empreinte $\mathcal{P}$ ne sont pas les mêmes que les couleurs visibles.

Chapitre 5

Conclusion

Dans cette thèse, nous avons présenté une méthode originale de génération de structure procédurale. Cette méthode est basée sur la composition d'un champ gaussien vectoriel **N** par une fonction de transfert à deux dimensions H. L'utilisation d'une fonction de transfert permet de séparer les informations de couleurs et de structure (contenues dans la fonction H) de la génération procédurale du bruit **N** apportant l'aléatoire. Ce modèle permet la génération procédurale d'une large variété de motifs, dont des motifs structurés.
Dans ce chapitre, nous faisons un bilan des contributions apportées et présentons des idées de directions de travaux futurs.

Sommaire

5.1 Génération procédurale de motifs structurés

La première contribution de cette thèse est la présentation d'une méthode de génération de motifs procéduraux basée sur la composition d'un champ vectoriel $\mathbf{N}$ par une fonction de transfert bi-dimensionnelle H.

5.1.1 Modèle par composition

Comme avec les cartes de couleurs uni-dimensionnelles classiques, cette composition permet de séparer l'information de structure (contenue dans la fonction H) de la génération de l'aléatoire (par le bruit $\mathbf{N}$). Ainsi, ces deux éléments peuvent être contrôlés individuellement et le résultat de la composition, $S = H \circ \mathbf{N}$ (figure 5.1), dépend des interactions entre eux.

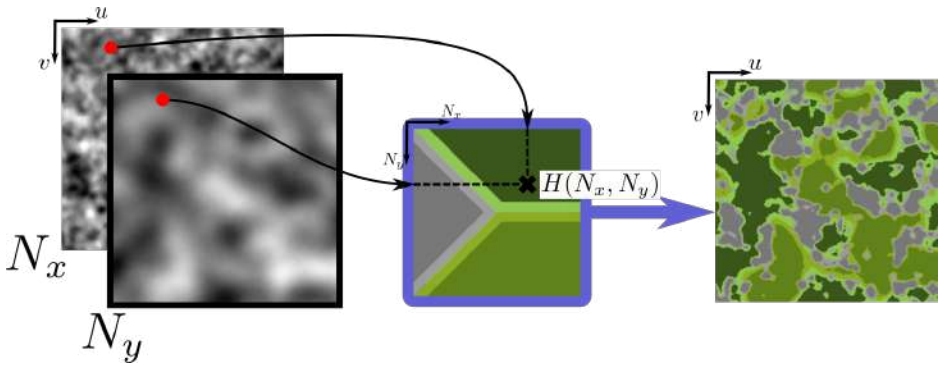


Figure 5.1 – *Le modèle est défini par une composition. À gauche, le bruit vectoriel* $\mathbf{N}$ *est représenté sous la forme de deux bruits scalaires pour des raisons de visibilité. Au centre, la fonction de transfert est stockée sous la forme d'une carte de couleurs. À droite, le résultat de la composition de* $\mathbf{N}$ *par* H.

Outre cette similitude de séparation, les fonctions de transfert bi-dimensionnelles offrent une plus grande liberté dans les fonctions qu'elles permettent de manipuler. Ainsi, la diversité de résultats possibles est plus importante. Cela permet également de reformuler des méthodes de génération déjà existantes, comme le bruit *phasor*.

5.1.2 Manipulation de fonctions bi-variées

Une limitation contraignante des cartes de couleurs traditionnelles est qu'elles ne permettent de manipuler que des fonctions uni-variées. Ainsi, le résultat de la composition ne dépend que d'un seul champ scalaire. Visuellement, dans le cas d'un champ scalaire N continu, cela signifie que les motifs générés seront obligatoirement concentriques, par construction. Cette limitation est très visible dans la figure 5.2, dans laquelle la texture à droite est le résultat de la composition du champ scalaire à gauche par la fonction de transfert uni-variée, représentée par la carte de couleurs au centre. Dans cette texture, les couleurs sont appliquées de façon concentrique. Ainsi, les zones de couleur rouge orangé sont toutes entourées d'un liseré noir, correspondant à la bande noire au centre de la carte de couleurs.

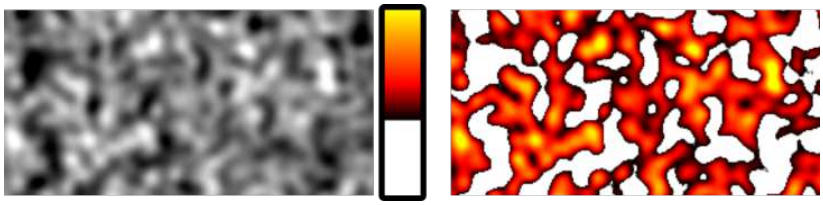
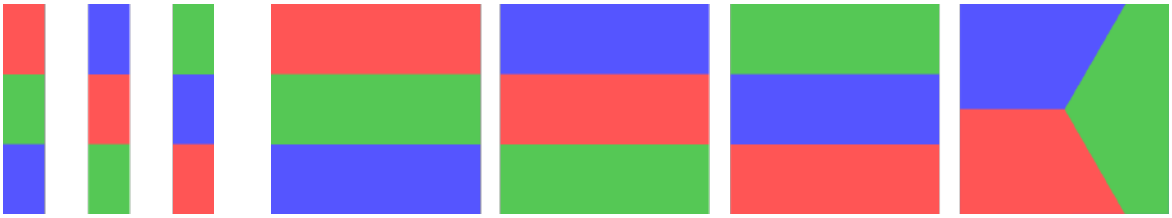


Figure 5.2 – *Composition d'un bruit par une carte de couleur. La carte de couleur (au centre) est appliquée sur le champ scalaire (à gauche) pour obtenir le résultat (à droite). Image inspirée de [HNPN13].*

L'utilisation de fonctions bi-variées offre une plus grande liberté et permet la création de motifs non concentriques. Visuellement, dans la carte représentant la fonction H, cela permet une plus large variété d'agencements possibles entre les régions de différentes couleurs.
Prenons, par exemple, une carte comportant 3 régions colorées de même taille. Dans une carte uni-dimensionnelle, il n'est possible de mettre qu'une seule couleur au centre de la carte. Cette couleur centrale est alors voisine des deux autres couleurs, mais ces dernières ne peuvent pas être voisines entre elles. Par construction, une carte uni-dimensionnelle à 3 zones ne peut donc avoir que 3 agencements possibles en termes de voisinages, comme nous pouvons le voir dans la figure 5.3a.
Dans une carte bi-dimensionnelle, il est possible de retrouver toutes les configurations de voisinages précédentes (avec une couleur voisine des deux autres, elles-mêmes non voisines entre elles). Mais il est également possible d'agencer les couleurs de façon à ce qu'elles soient toutes voisines entre elles, comme le montre la figure 5.3b. De plus, la forme et la position des régions dans la carte peuvent varier tout en conservant la contrainte de taille égale des trois zones.



(a) *Cartes 1D* **(b)** *Cartes 2D*

FIGURE 5.3 – *En tenant compte de la symétrie, il n'existe que trois agencements possibles de voisinages pour une carte uni-dimensionnelle comportant 3 régions colorées (5.3a). Dans une carte bi-dimensionnelle, il est possible de rendre toutes les régions voisines entre elles, mais également de leur donner différentes formes.*

Cette différence devient plus marquée en augmentant le nombre de régions colorées. Soit une carte composée de n zones colorées, toutes de même taille. Dans une carte uni-dimensionnelle, il existe $n!$ ordres possibles pour les couleurs, soit $\frac{n!}{2}$ agencements de voisinages (en tenant compte des symétries). Dans une carte bi-dimensionnelle, le nombre d'agencements de voisinage possibles correspond au nombre de manières possibles de relier n sommets d'un graphe entre eux de façon à ne pas avoir de sommet isolé, soit le nombre de graphes connexes à n sommets.
Comme pour les cartes de couleurs 1D, les brusques variations de couleurs ou de contraste délimitant les motifs structurés dans le résultat S sont dues aux discontinuités entre les zones colorées de H.

5.1.3 Effet du choix du bruit N

Outre l'agencement des régions colorées dans la carte de couleurs, les caractéristiques spectrales du bruit vectoriel **N** influent sur l'apparence du résultat (figure 5.4).

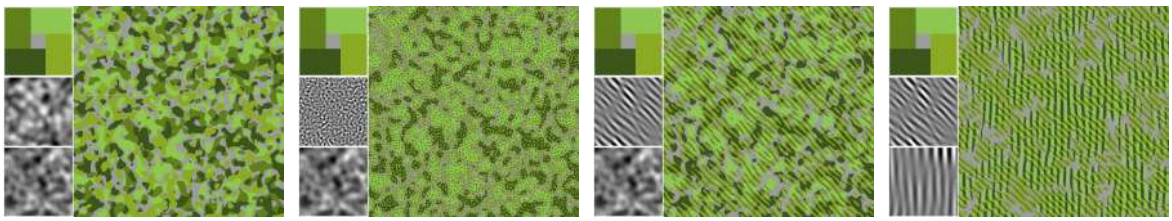


FIGURE 5.4 – *Variation de résultat selon le spectre de* **N**. *Pour chaque sous-figure, les vignettes à gauche montrent la carte de couleur et les composantes du bruit* **N**. *La carte de couleurs est inchangée pour les quatre sous-figures.*

Ainsi, la fréquence, l'amplitude ou l'anisotropie de **N** modifie, entre autres, la fréquence, la taille ou la forme des zones colorées dans le résultat S.

5.1.4 Le filtrage

Le filtrage des textures est une étape importante lors du rendu. En effet, elle est nécessaire pour assurer la cohérence spatiale et temporelle de la couleur des pixels à l'écran. Filtrer directement le résultat S de la composition présente des difficultés en raison de la non-linéarité de la fonction H. Le filtrage d'une composition est un problème qui a été précédemment étudié, en particulier dans l'article de Heitz et collègues [HNPN13]. Ainsi, l'intégrale, sur une empreinte, du résultat de la composition est égale à l'intégrale de toutes les valeurs possibles de H, pondérées par leur présence dans l'empreinte. Cette observation permet de séparer le filtrage de la texture entre le filtrage de la fonction H et celui du champ $\mathbf{N}$ (détails des calculs dans la section 3.2.2).

5.2 Rendu en temps réel

Faire le rendu en temps réel d'une texture procédurale implique de pouvoir évaluer à la volée toutes les informations nécessaires dans l'empreinte d'un *pixel*. Au cours de cette thèse, nous avons présenté deux types d'informations dont nous avons eu besoin. D'une part, les statistiques décrivant la répartition des intensités de $\mathbf{N}$ dans une empreinte, et d'autre part, la dérivée spatiale du résultat de la composition.

5.2.1 Estimation des statistiques à la volée

Le résultat S de la composition peut être filtré à la volée en pré-filtrant la carte de couleurs représentant la fonction H (figure 5.5, à droite). Ce filtrage nécessite également de connaître la distribution des intensités du bruit $\mathbf{N}$ dans l'empreinte étudiée. Dans le cas où $\mathbf{N}$ est un bruit gaussien, la distribution de ses intensités dans une empreinte peut être approximée par une gaussienne. Les fonctions gaussiennes sont entièrement décrites par leur moyenne et leur matrice de covariance (figure 5.5, au centre).

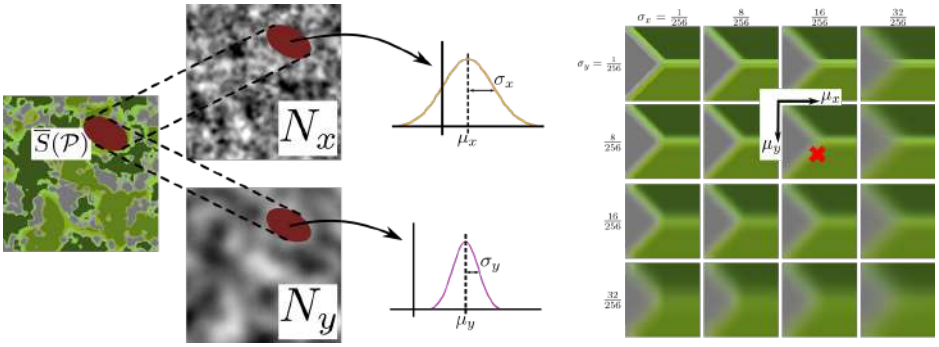


Figure 5.5 – *L'estimation de la couleur moyenne dans une empreinte dans le résultat de la composition (à gauche) est fait en estimant la répartition des intensités des bruits dans cette empreinte (au centre). Ces répartitions sont utilisées dans une version pré-filtrée de la carte H (à droite).*

Estimation de la moyenne. Les méthodes de génération de bruits procéduraux sont toutes accompagnées de leur méthode de filtrage, permettant d'estimer la moyenne sur une empreinte à la volée. Par exemple, dans le cas de la méthode de pavage et mélange de Heitz et collègues [HN18], l'application de la synthèse sur une hiérarchie MIP de l'exemple d'entrée permet d'obtenir différents niveaux de détail de la moyenne du bruit. Cependant, l'estimation des statistiques d'ordre 2 présente plus de difficultés.

Estimation de la matrice de covariance. Dans la section 3.3 de cette thèse, nous avons présenté des estimateurs de la matrice de covariance d'un bruit gaussien vectoriel $\mathbf{N}$ à la volée. Nos estimateurs sont basés sur une méthode de génération par pavage et mélange, telle que celle de Heitz et collègues [HN18]. Ainsi, l'utilisation d'un exemple d'entrée de taille restreinte nous permet d'avoir des

connaissances à priori sur les statistiques du bruit $\mathbf{N}$. Une difficulté est que la matrice de covariance d'une combinaison linéaire de variables n'est pas égale à la combinaison linéaire des matrices. Pour palier à cette difficulté, notre méthode consiste à utiliser la combinaison des moments d'ordre 2 non centrés (détails des calculs dans la section 3.3).

5.2.2 Calcul des dérivées

La connaissance des dérivées partielles d'un champ scalaire est une information importante, notamment lors du rendu de géométrie ou de micro-géométrie. En effet, c'est le vecteur normal à la surface, déterminé à partir de la dérivée, qui permet de déterminer la direction de réflexion de la lumière reçue. Notre modèle nous permet de séparer la structure et le champ gaussien, il nous permet donc également de calculer les dérivées du résultat par dérivée de la composition. Comme nous l'avons vu dans la section 4.5.1, nous pouvons calculer le gradient d'un bruit généré par convolution parcimonieuse (comme le bruit par convolution de noyaux de Gabor [LLDD09]) avec la somme des gradients des noyaux. Puis, par la règle de la chaîne, le gradient de la composition est obtenu par le produit de la Jacobienne du bruit vectoriel $\mathbf{N}$ et du bruit composé par le gradient de la fonction H. Ainsi, la fonction de transfert H peut être dérivée à part, de façon analytique ou numérique et stockée en mémoire.

5.3 Reprise de méthodes existantes

5.3.1 Le bruit *Phasor*

La fonction arc tangente. La fonction Atan2 est une variation de la fonction arc tangente, prenant 2 arguments en entrée. Elle est intéressante à étudier parce qu'elle permet un changement de repère de la fonction de transfert H. Ainsi, elle permet de passer de l'utilisation en coordonnées cartésiennes à celle en coordonnées polaires.

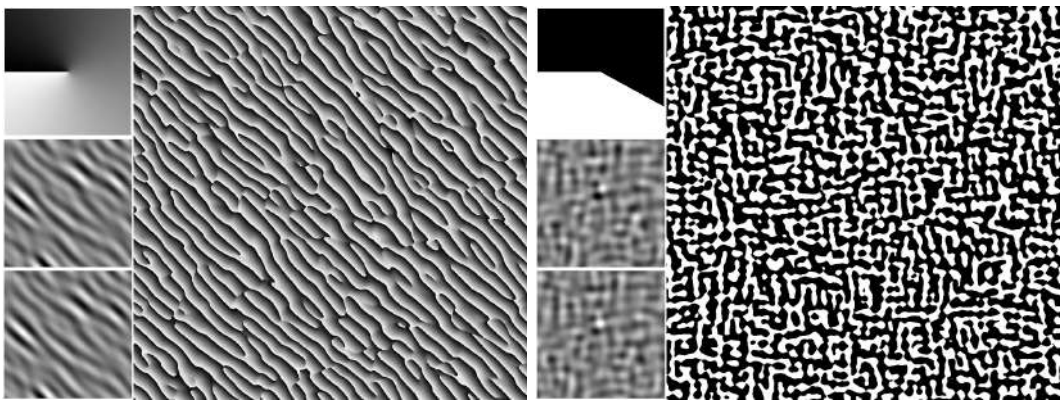


Figure 5.6 – *Reproduction du bruit Phasor par la composition d'un bruit vectoriel par une carte contenant le profil des vagues (à gauche des dents de scie, à droite un profil carré).*

Ce changement de système de coordonnées est en particulier intéressant dans le cas où le champ vectoriel $\mathbf{N}$ représente les parties réelle et imaginaire d'un unique champ complexe $\widetilde{N}$. Dans ce cas, la fonction de transfert H permet alors de manipuler le module et l'argument de $\widetilde{N}$. Cela nous permet de reformuler le bruit *phasor* sous la forme Phasor $= H \circ \mathbf{N}$, comme le montre la figure 5.6 (détails des calculs dans la section 3.5.3). Cette reformulation nous permet, entre autres, de traiter certains défauts du bruit *phasor* [TEZ+19]. Ainsi, si sa formulation initiale ne s'appuie que sur l'argument du champ complexe (représenté par la coordonnée angulaire de H), la reformulation offre la possibilité d'utiliser la coordonnée radiale de H et donc le module du champ. De cette façon, il devient possible de traiter la singularité apportée par la fonction Atan2 en $(x, y) = (0, 0)$, comme nous l'avons vu dans la section 4.3.2.

Le filtrage. Une des limites du bruit *phasor* [TEZ+19] était l'impossibilité d'en faire le filtrage à la volée et donc d'utiliser son plein potentiel de méthode procédurale lors d'un rendu en temps réel. Notre reformulation de cette méthode sous la forme d'une composition entre une fonction H et un champ $\mathbf{N}$ nous permet de considérer son filtrage comme celui d'une simple composition. De cette façon, la fonction H, contenant l'Atan2, peut être filtrée en amont, ce qui permet de faire le filtrage des motifs du bruit *phasor* à la volée (figure 5.7).

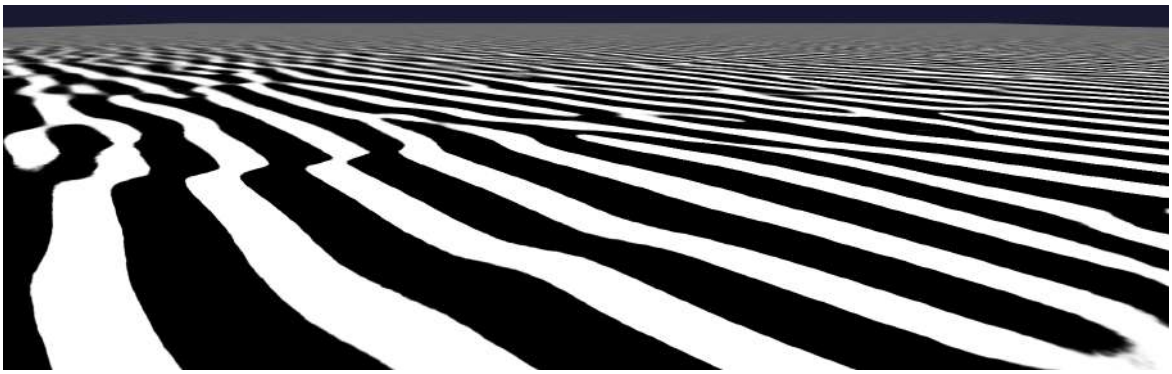


Figure 5.7 – *Filtrage du bruit phasor en réalisant le pré-filtrage de la fonction de transfert H contant le profil γ et la fonction* Atan2.

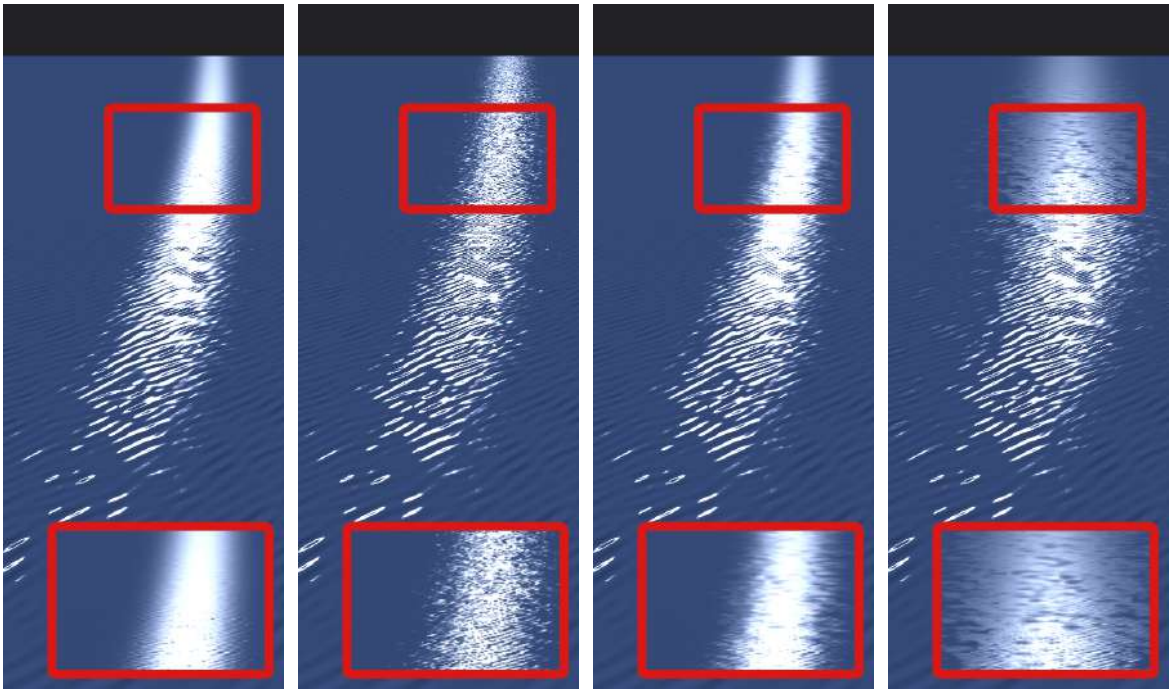
La dérivation. De la même façon que pour le filtrage, la reformulation comme la composition d'un bruit vectoriel par une fonction bi-dimensionnelle nous permet d'appliquer le résultat précédent au bruit *phasor*. Ainsi, la fonction H, contenant le profil des vagues et la fonction Atan2, est dérivée à part puis appliquée sur le bruit pour obtenir le gradient du résultat (figure 5.8).

(a) *Bruit phasor* **(b)** *Notre méthode* **(c)** *Dérivation numérique*

Figure 5.8 – *Dérivation du bruit phasor 5.8a par notre méthode analytique 5.8b et par différences finies 5.8c. La méthode numérique est très dépendante de la résolution choisie, en particulier lors de forte variation dans les valeurs du champ scalaire.*

5.3.2 Le filtrage des cartes de normales

L'estimation à la volée de la matrice de covariance d'un champ vectoriel nous permet également de présenter une implémentation en temps réel du filtrage de cartes de normales. En effet, l'algorithme de filtrage de cartes de normales de Olano et collègues [OB10] utilise les informations de moyenne, de variance et de covariance du champ de normales pour déterminer la rugosité d'une surface. La méthode précédente, proposée par Deliot et collègues [DH19], provoquait une perte de détails. Nos estimateurs des composantes de la matrice de covariance permettent de retrouver des détails à mi-distance par rapport à leur approximation de la variance constante par niveau de détail (figure 5.9).



(a) *[DH19]* **(b)** *Référence* **(c)** *Notre méthode* **(d)** *Covariance nulle*

Figure 5.9 – *Comparaison du filtrage de cartes de normales représentant des vagues sur l'océan. La précédente méthode utilisait des valeurs de variance constantes par niveau de détail (figure 5.9a). Notre méthode permet de retrouver les détails à mi-distance et de se rapprocher de la référence (figure 5.9c). En utilisant une covariance nulle (approximation faite pour les couleurs dans la section 3.4.1), les détails à mi-distance sont conservés, seul la forme du lobe globale est modifiée (figure 5.9d).*

5.4 Perspective de recherche : la synthèse par l'exemple

Le résultat S de la composition de $\mathbf{N}$ par H présente des caractéristiques héritées du bruit et de la fonction de transfert, ce qui permet de le contrôler. Ainsi, nous pouvons contrôler les couleurs présentes en utilisant la fonction H et, par exemple, l'orientation des motifs en guidant le champ $\mathbf{N}$. Cependant, il n'est pas possible de séparer complètement les impacts du bruit $\mathbf{N}$ et de la fonction H sur l'apparence du résultat S. La conséquence est qu'il est complexe de prévoir l'apparence du résultat S à partir de H. Pour les résultats du chapitre 3, les cartes de couleurs présentées ont été créées par tâtonnement de façon itérative, mais il est laborieux de définir H pour obtenir un résultat souhaité.
De nombreux modèles de génération souffrent de ce manque de contrôle intuitif de leurs paramètres. Une solution couramment utilisée pour rendre les contrôles plus intuitif est de permettre l'utilisation d'un exemple de taille limitée pour définir les paramètres du modèle.

5.4.1 Présentation du problème

Objectifs. En définissant une méthode de synthèse par l'exemple, l'objectif global est de permettre à un utilisateur d'utiliser un échantillon de texture pour configurer les paramètres de la méthode de synthèse. C'est-à-dire que, à partir d'une texture S de taille finie, échantillonnée et stockée en mémoire sous la forme d'une image, nous cherchons à retrouver les données d'entrées du modèle. Dans notre cas, les données d'entrées sont le bruit vectoriel $\mathbf{N}$ et la fonction bi-variée représentée par la carte de couleurs bi-dimensionnelle H. Ainsi, l'objectif d'une méthode de synthèse par l'exemple, dans notre cas, est de retrouver une décomposition possible de S en un bruit $\mathbf{N}$ et une carte H.

Difficultés. Nous avons présenté une partie des interactions entre $\mathbf{N}$ et H dans la section 3.5. Ainsi, l'agencement, la couleur et la taille des cellules dans H, et le spectre des composantes de $\mathbf{N}$ modifient le résultat. Cependant, pour obtenir des résultats plus riches, ces interactions sont encore plus entrelacées (figure 5.10).

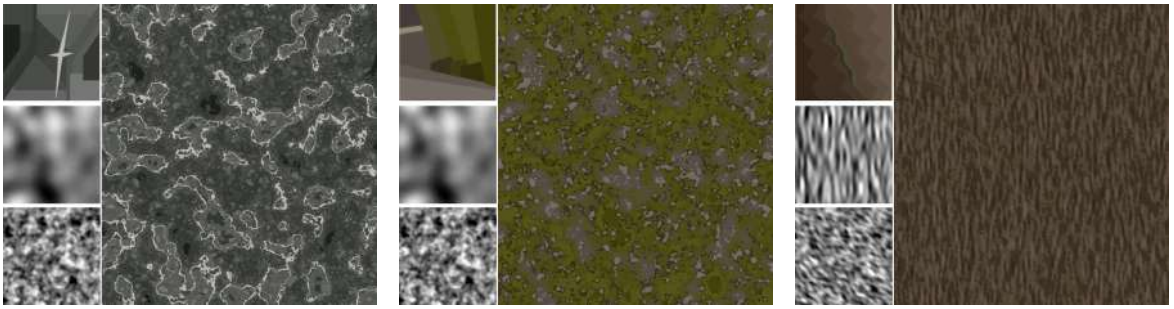


Figure 5.10 – *Exemples de cartes de couleurs plus travaillées pour obtenir des résultats plus riches. Ces cartes de couleurs ont été créées par tâtonnement en ajoutant ou en retirant itérativement des cellules colorées.*

La conséquence de cet entrelacement est que les caractéristiques individuelles des données d'entrée (bruit $\mathbf{N}$ et carte de couleurs H) sont difficiles à extraire en ne connaissant que le résultat de la composition.

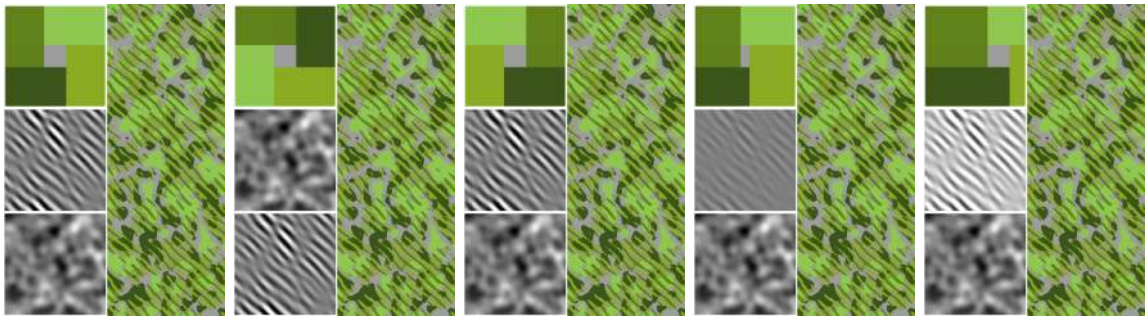
5.4.2 Délimitation du problème

Hypothèses. Pour réduire le domaine des données d'entrée possible, nous considérons que le bruit vectoriel $\mathbf{N}$ utilisé pour obtenir le résultat S possède certaines propriétés. D'abord, nous nous plaçons dans le cas où les composantes de ce bruit suivent des processus gaussiens. Cela nous permet de connaître à priori la forme de la fonction de probabilité des intensités des bruits (des fonctions gaussiennes). Ensuite, nous posons l'hypothèse que ces processus sont indépendants. De cette façon, la fonction de probabilité du bruit vectoriel est obtenu par le produit des deux fonctions de probabilité des composantes. Nous avons vu dans la section 3.4.1 que cette hypothèse pouvait raisonnablement être faite. Enfin, nous considérons que ces composantes sont continues. C'est-à-dire que la précision de l'échantillonnage de l'image représentant le résultat S est suffisante. Cette hypothèse nous assure que toutes les couleurs présentes et tous les voisinages entre couleurs présents dans S existe dans la carte de couleurs H, et qu'aucune information n'a pu être perdue par l'échantillonnage.

Éléments de simplification. En considérant les hypothèses précédentes, nous pouvons remarquer que certains paramètres du bruit et de la carte peuvent être modifiés en miroir de manière à obtenir le même résultat avec des données d'entrée différentes.

- **Transposée de H :** utiliser la transposée de la carte de couleurs correspond à inverser ses axes x et y, cette opération est équivalente à inverser les deux composantes du bruit $\mathbf{N}$ (figure 5.11b).
- **Symétrie de H selon un axe :** considérons que les composantes de $\mathbf{N}$ sont de moyenne nulle, utiliser le champ $-N_x$ à la place de N_x est équivalent à réaliser une symétrie de la carte H selon son axe y (figure 5.11c).
- **Mise à l'échelle d'un axe de H :** multiplier une des composantes de $\mathbf{N}$ par une valeur constante permet de modifier son amplitude, cela a pour impact de modifier la largeur de la plage de valeurs atteintes. Ainsi, utiliser le champ αN_x au lieu de N_x est équivalent à mettre à l'échelle l'axe x de H (figure 5.11d).
- **Translation d'un axe de H :** ajouter une constante à une des composantes de $\mathbf{N}$ permet de modifier sa valeur moyenne. Ainsi, utiliser $N_x + \alpha$ au lieu de N_x est équivalent à translater d'autant la carte H le long de l'axe x (figure 5.11e).

D'autres opérations comme des combinaisons linéaires entre les composantes de $\mathbf{N}$ peuvent être définies de façon à être équivalentes à une rotation de la carte H.



(a) **(b)** *Transposée* **(c)** *Inverse* **(d)** *Mise à l'échelle* **(e)** *Translation*

Figure 5.11 – *Modification en miroir des données d'entrée du modèle de la figure 5.11a de façon à obtenir le même résultat après la composition.*

Ces opérations en miroir sur le bruit et la carte de couleurs nous montre qu'il n'y a pas une unique décomposition pour une texture exemple donnée. Mais elles nous permettent également de fixer certaines caractéristiques du bruit pour étudier la carte de couleurs. Ainsi, nous pouvons fixer la forme de la fonction de probabilité des intensités de $\mathbf{N}$ en fixant sa moyenne et sa matrice de covariance. En effet, ces paramètres peuvent être corrigés après avoir défini une carte H possible en appliquant une translation et une mise à l'échelle des axes de H.

5.4.3 Analyse de l'exemple

Un processus possible pour décrire $\mathbf{N}$ et H à partir de S est de commencer par décrire une carte de couleurs possible, puis de définir les composantes du bruit. En effet, en tenant compte des hypothèses et des éléments de simplification présentés, nous pouvons fixer une valeur moyenne et une matrice de covariance pour $\mathbf{N}$, ce qui nous donne des informations sur la probabilité d'utilisation de chaque région de la carte H. De cette façon, l'observation d'une texture exemple S permet de déduire des informations sur la fonction de transfert H.
L'objectif ici est donc de réaliser des mesures dans S et de les mettre en lien avec des mesures dans H en tenant compte des informations statistiques fixées pour $\mathbf{N}$. Ces mesures en parallèle dans S et H sont ensuite utilisées dans un processus d'optimisation pour tendre vers une carte H possible.

Proportion de présence. La première mesure qui peut être facilement réalisée sur S est la proportion de présence de chaque couleur. Dans S, cette proportion se mesure simplement en comparant le nombre de *pixels* de chaque couleur avec le nombre de *pixels* total.
Dans la carte de couleurs, cette proportion peut être mesurée en utilisant la fonction de probabilité des intensités de $\mathbf{N}$ définie par les statistiques fixée en amont. En effet, le volume obtenu par l'intégrale de cette fonction de probabilité, dans une portion du domaine de définition de H, correspond à la probabilité d'utilisation de cette cellule, et donc à la proportion de présence de la couleur correspondante (figure 5.12).

Proportion de voisinage. La proportion de voisinage entre chaque paire de couleurs est également une information importante. Dans S, cette proportion se mesure en comptant le nombre de *pixels* de chaque couleur voisin à chacune des autres couleurs présentes. Cette analyse de S n'est plus ponctuelle, mais locale étant donné qu'elle dépend du voisinage des *pixels*. Nous cherchons donc à étudier la façon dont des *pixels* voisins co-varient.
La fonction de probabilité des intensités ne contient pas d'information de voisinage dans l'espace de la texture. Une solution pour mesurer les proportions de voisinages dans la carte de couleurs serait de réaliser une montée en dimension, en considérant les valeurs de $\mathbf{N}$ pour des coordonnées voisines. De cette façon, il serait possible de poser une hypothèse sur la co-variance des valeurs de $\mathbf{N}$ entre plusieurs coordonnées.

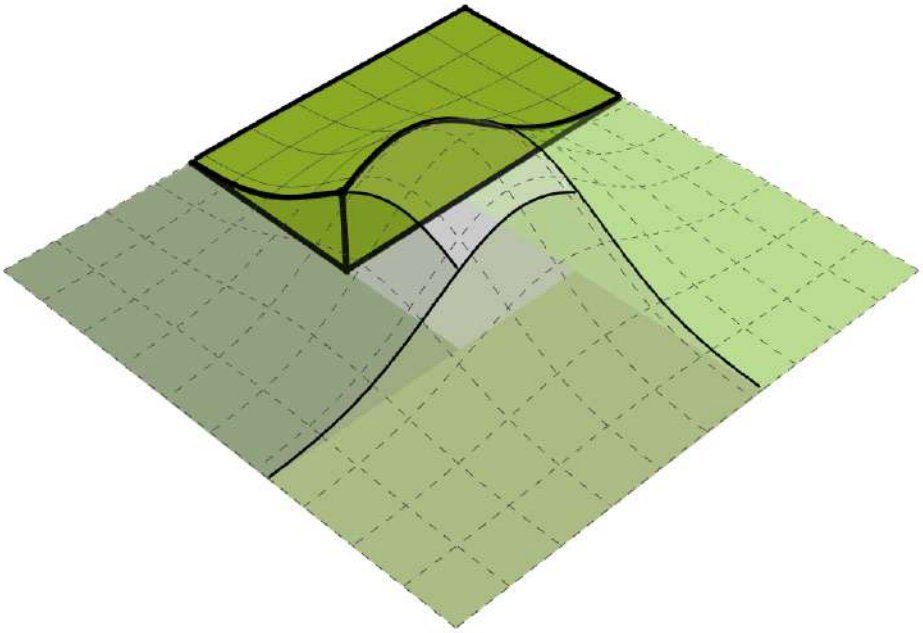


Figure 5.12 – *La fonction de probabilité des intensités (en pointillé) renseigne sur la probabilité d'utilisation des différentes cellules de la carte de couleurs. La proportion d'utilisation d'une couleur correspond au volume sous la fonction de probabilité (lignes continues épaisses).*

Autres mesures. De nombreuses informations sont par ailleurs difficiles à obtenir par la seule observation de S. Par exemple, la présence de plusieurs cellules de même couleur dans H, utilisée avec un bruit $\mathbf{N}$ basse fréquence, peut être difficile à distinguer de la présence d'une seule cellule et l'utilisation d'un bruit haute fréquence. Une autre information que nous pourrions être tentés de mesurer dans S est la forme des régions colorées. Par exemple leur concavité ou leur compacité pour chaque couleur. Cette mesure pourrait donner des informations sur la position d'une cellule colorée dans H. Intuitivement, une cellule proche du centre de H générera plutôt des régions semblables à un réseau plus ou moins connecté reliant les valeurs moyennes de $\mathbf{N}$. À l'inverse, une cellule proche du bord de la carte générera des régions plus compactes, dont la forme va également dépendre de l'anisotropie de $\mathbf{N}$. Cependant, de telles mesures sont difficiles à faire dans S et il est également complexe de définir une métrique dans H pouvant être utilisée dans un processus d'optimisation.

5.5 Autres perspectives

5.5.1 Graphes plus avancés

Notre modèle de génération, par composition d'un bruit vectoriel par une fonction de transfert bi-dimensionnelle, permet de synthétiser une large variété de motifs procéduraux structurés. Cependant, les graphes de textures créés par des artistes pour les besoins de jeux vidéo ou de films sont beaucoup plus conséquents pour permettre de créer des textures plus riches et détaillées.

Succession de nœuds. Une première piste pour étendre le modèle à des graphes plus conséquents est de permettre la mise en cascade de plusieurs fonctions de transfert. Ainsi, le résultat d'une composition peut être utilisé comme donnée d'entrée pour la composition suivante. Comme dans la figure 5.13. Cependant, cette extension apporte une difficulté pour le filtrage du résultat final du graphe. En effet, la méthode de filtrage de la composition, présentée dans la section 3.2.2, repose sur l'hypothèse que le bruit vectoriel $\mathbf{N}$ est gaussien. Cette hypothèse nous donne des informations sur la distribution des intensités du bruit. C'est également grâce à cette hypothèse que nous pouvons approximer la distribution des intensités dans une empreinte quelconque par une gaussienne. Dans le cas où une des composantes d'entrée est le résultat d'une composition réalisée en amont, l'hypothèse

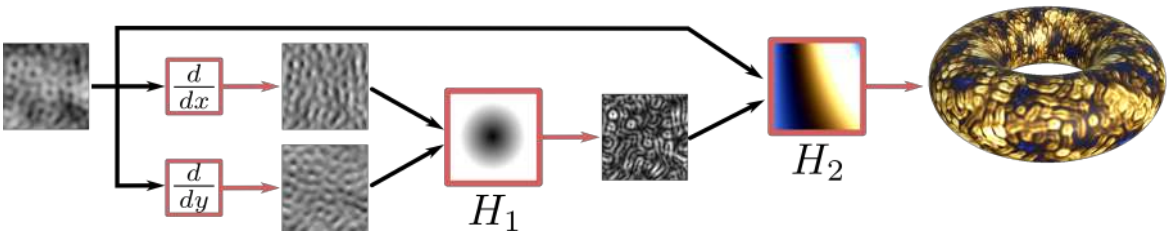


Figure 5.13 – *Utilisation successive de deux fonctions de transfert H_1 et H_2.*

que la distribution de ses intensités est gaussienne ne peut pas être faite dans toutes les situations. La majorité du temps, cette hypothèse sera fausse. C'est le cas par exemple dans la figure 5.13, les données d'entrée de la première fonction H_1 ont bien des distributions d'intensités gaussiennes, mais ce n'est pas le cas des données d'entrée de la seconde fonction H_2.

Fonction de transfert nD. Il serait tentant de proposer l'utilisation de cartes de couleurs de dimension supérieure à 2, pour représenter des fonctions multi-variées. Cependant, la montée en dimension de la carte de couleurs implique également la montée en dimension de la matrice des niveaux de pré-filtrage de cette carte. Ainsi, avec une carte H de dimension 2, la matrice du pré-filtrage est de dimension 5 (2 moyennes, 2 variances et 1 covariance) ou de dimension 4 si les composantes du bruit sont indépendantes (la covariance est alors nulle). Dans le cas d'une carte à n dimensions, la matrice du pré-filtrage est alors de dimension $2n + \binom{n}{2}$ (n moyennes, n variances et 2 parmi n covariances) ou de dimension $2n$ si toutes les composantes du bruit sont toutes indépendantes 2 à 2. Cette montée en dimension risque donc d'être coûteuse en mémoire. Étudier le potentiel des cartes à n dimensions peut être intéressant, il faudrait cependant s'assurer en amont que celles-ci apportent un réel intérêt en comparaison du coût en mémoire et dans quels domaines et situations les utiliser.

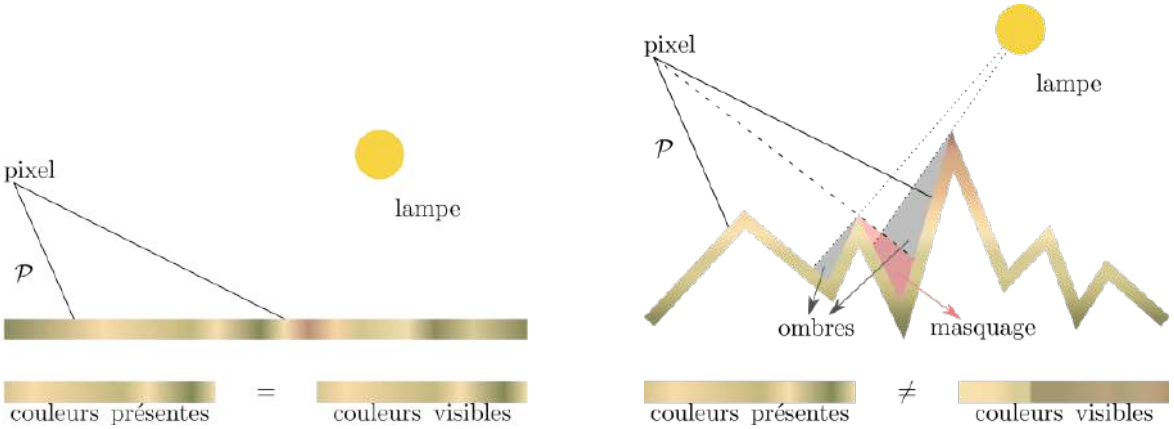
5.5.2 Variations de texture

Variations spatiales. L'apparence des objets du monde réel peut présenter des variations spatiales. Par exemple, une pièce de tissus usée par endroits va avoir des variations spatiales de couleurs. Ces variations sont difficiles à faire avec notre modèle parce qu'elles impliquent de pouvoir faire des transitions entre plusieurs fonctions de transfert. De plus, pour pouvoir réaliser le rendu du résultat en temps réel, il est nécessaire de pouvoir également faire les transitions lors du filtrage à la volée. Ainsi, si une empreinte de *pixel* se situe dans une zone de transition de fonctions, la distribution des intensités du bruit vectoriel $\mathbf{N}$ est répartie entre les différentes fonctions de façon inconnue à priori. Les récents travaux de Fournier et Sauvage [FS23] proposent une méthode de transition entre plusieurs textures ainsi que le filtrage à la volée du résultat. Leur méthode repose sur des textures entièrement connues en amont, cependant, les informations apportées par la carte de couleurs de notre méthode pourraient être suffisante pour adapter leur méthode.

Variations temporelles. L'apparence peut aussi évoluer en fonction du temps. Par exemple sous l'effet d'éléments naturels, comme de la rouille due à l'humidité, ou de la terre qui s'assèche sous l'effet du soleil, ou encore sous l'effet d'une force appliquée sur l'objet, comme pour les déformations observées sur les motifs d'un tissu étiré. Les travaux de Guingo et collègues [GLS+20] présentent une méthode de déformation dynamique de textures. Le contrôle des déformations qu'ils proposent est réalisé localement et repose sur l'utilisation d'un champ de déformation pré-calculé. Il serait intéressant d'étudier l'adaptation de notre modèle à leur méthode.

5.5.3 Génération de géométrie

Filtrage de géométrie. La génération des détails de géométrie du chapitre 4 est faite à la volée. Cependant, nous n'avons pas étudié le filtrage à la volée de ces détails. Le filtrage est une étape délicate lors de la génération procédurale de détails géométriques. En effet, ceux-ci vont provoquer des effets d'ombrage et masquer certaines zones (figure 5.14).



(a) *Surface plate* **(b)** *Présence de géométrie*

FIGURE 5.14 – *L'ajout de géométrie nécessite de prendre en compte les ombres et les zones masquées pour déterminer la couleur moyenne dans l'empreinte d'un pixel.*

Génération de micro-géométrie. Un autre domaine pour lequel la génération procédurale de géométrie est intéressante à étudier est la définition de la micro-géométrie à la surface des objets. En effet, un modèle très utilisé pour créer des rendus d'objets photo-réalistes est celui des micro-facettes. Ce modèle est une représentation statistique de la BRDF et permet donc le rendu des objets de loin. Une extension intéressante de notre modèle serait de pouvoir générer procéduralement une micro-géométrie explicite, contrôlée par ces statistiques. Cela permettrait de faire le lien entre la représentation statistique et la micro-géométrie explicite sans avoir besoin de stocker cette dernière. De cette façon, il serait possible de faire la transition entre les différentes échelles de représentation en utilisant un seul modèle, cohérent à chaque niveau de détails. Cette idée nécessite de pouvoir contrôler la répartition des pentes du résultat, cependant cela pourrait être possible grâce au contrôle offert par la fonction de transfert H.

5.5.4 Délimitation de l'espace des possibles

Les idées de perspectives précédentes sont des cas particuliers d'une question plus générale, qui est de savoir quel est le domaine des textures qu'il est possible de créer en utilisant le modèle des fonctions bi-dimensionnelles. Ainsi, nous avons vu dans la section 5.4 de cette conclusion qu'il est difficile de déterminer des paramètres d'entrée possibles pour décomposer une texture (même en sachant qu'elle a été obtenue par notre modèle). Il est donc d'autant plus difficile de déterminer si une texture peut, ou ne peut pas, être décomposée de cette façon, et pourquoi. Cette question se pose aussi bien pour la génération de textures de couleurs (qui sont celles que nous avons principalement étudiées dans le chapitre 3) que pour les autres caractéristiques lumineuses de la surface d'un objet (comme la géométrie, comme nous l'avons vu dans le chapitre 4 et la section 5.5.3).

Bibliographie

[BHN07] Robert Bridson, Jim Houriham, and Marcus Nordenstam. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics (ToG)*, 26(3) :46–es, 2007.

[BMWM06] Steven Bergner, Torsten Moller, Daniel Weiskopf, and David J Muraki. A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE transactions on visualization and computer graphics*, 12(5) :1353–1360, 2006.

[BN11] Eric Bruneton and Fabrice Neyret. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics*, 18(2) :242–260, 2011.

[Bur19] Brent Burley. On histogram-preserving blending for randomized texture tiling. *Journal of Computer Graphics Techniques*, 8(4), 2019.

[CD05] Robert L Cook and Tony DeRose. Wavelet noise. *ACM Transactions on Graphics (TOG)*, 24(3) :803–811, 2005.

[Che19] Xavier Chermain. Rendu basé physique de micro-reflet. *Thèse de Doctorat*, 2019.

[Coo84] Robert L Cook. Shade trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 223–231, 1984.

[Cro84] Franklin C Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, 1984.

[CSM14] Victor Charpenay, Bernhard Steiner, and Przemyslaw Musialski. Sampling gabor noise in the spatial domain. In *Proceedings of the 30th Spring Conference on Computer Graphics*, pages 79–82, 2014.

[DH19] Thomas Deliot and Eric Heitz. Procedural stochastic textures by tiling and blending. *GPU Zen*, 2, 2019.

[DHI+13] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. Linear efficient antialiased displacement and reflectance mapping. *ACM Transactions on Graphics (TOG)*, 32(6) :1–11, 2013.

[DMLG02] J-M Dischler, Karl Maritaud, Bruno Lévy, and Djamchid Ghazanfarpour. Texture particles. *Computer Graphics Forum*, 21(3) :401–410, 2002.

[EMP+02] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling : A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2002.

[FS23] Romain Fournier and Basile Sauvage. Mix-Max : a content-aware operator for real-time texture transitions. *Journées Française d'Informatique Graphique*, Novembre 2023.

[Gab46] Dennis Gabor. *Theory of communication*. J. Int. Electrical Engineers, 1946.

[GD95] Djamchid Ghazanfarpour and Jean-Michel Dischler. Spectral analysis for automatic 3-d texture generation. *Computers & Graphics*, 19(3) :413–422, 1995.

[GDG12] Guillaume Gilet, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Multiple kernels noise for improved procedural texturing. *The Visual Computer*, 28 :679–689, 2012.

[GDGP16] Eric Guérin, Julie Digne, Eric Galin, and Adrien Peytavie. Sparse representation of terrains for procedural modeling. *Computer Graphics Forum (proceedings of Eurographics 2016)*, 35(2) :177–187, 2016.

[GGP+19] Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. A review of digital terrain modeling. *Computer Graphics Forum*, 38(2) :553–577, 2019.

[GH86] Ned Greene and Paul S Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6) :21–27, 1986.

[Gla14] Andrew S Glassner. *Principles of digital image synthesis*. Elsevier, 2014.

[GLLD12] Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. Gabor noise by example. *ACM Transactions on Graphics (ToG)*, 31(4) :1–9, 2012.

[GLM17] B. Galerne, A. Leclaire, and L. Moisan. Texton noise. *Computer Graphics Forum*, 36(8) :205–2018, 2017.

[GLS+20] Geoffrey Guingo, Frédéric Larue, Basile Sauvage, Nicolas Lutz, Jean-Michel Dischler, and Marie-Paule Cani. Content-aware texture deformation with dynamic control. *Computers & Graphics*, 91 :95–107, 2020.

[GSDC17] Geoffrey Guingo, Basile Sauvage, Jean-Michel Dischler, and Marie-Paule Cani. Bi-layer textures : a model for synthesis and deformation of composite textures. *Computer Graphics Forum*, 36(4) :111–122, 2017.

[GSDT22] Charline Grenier, Basile Sauvage, Jean-Michel Dischler, and Sylvain Thery. Color-mapped noise vector fields for generating procedural micro-patterns. *Computer Graphics Forum*, 40(7), 2022.

[GSV+14] Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler, and Djamchid Ghazanfarpour. Local random-phase noise for procedural texturing. *ACM Transactions on Graphics*, 33(6) :195 :1–195 :11, November 2014. (Proceedings of Siggraph Asia'14).

[GZD08] Alexander Goldberg, Matthias Zwicker, and Frédo Durand. Anisotropic noise. *ACM Transactions on Graphics (TOG)*, 27(3) :1–8, 2008.

[Hec89] Paul S Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, University of California, Berkeley, CA, June 1989.

[HN18] Eric Heitz and Fabrice Neyret. High-performance by-example noise using a histogram-preserving blending operator. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2) :1–25, 2018.

[HNPN13] Eric Heitz, Derek Nowrouzezahrai, Pierre Poulin, and Fabrice Neyret. Filtering Color Mapped Textures and Surfaces. *I3D'13 - ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 129–136, March 2013.

[HNPN14] Eric Heitz, Derek Nowrouzezahrai, Pierre Poulin, and Fabrice Neyret. Filtering Non-Linear Transfer Functions on Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 20(7) :996–1008, July 2014.

[Hor45] Robert E Horton. Erosional Development of Streams and their Drainage Basins ; Hydrophysical Approach to Quantitative Morphology. *GSA Bulletin*, 56(3) :275–370, 03 1945.

[KBKŠ09] Peter Krištof, Bedrich Benes, Jaroslav Křivánek, and Ondřej Šťava. Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum*, 28(2) :219–228, 2009.

[KKS+08] Andrew Kensler, Aaron Knoll, Peter Shirley, A Kensler, A Knoll, and P Shirley. Better gradient noise. Technical report, Tech. Rep. UUSCI-2008-001, SCI Institute, University of Utah, 2008.

[Kol40] Andrei Nikolaevitch Kolmogorov. The wiener spiral and some other interesting curves in hilbert space. *Dokl. Akad. Nauk SSSR*, 26(2) :115–118, 1940.

[Lew84] John-Peter Lewis. Texture synthesis for digital painting. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 245–252, 1984.

[Lew86] JP Lewis. Methods for stochastic spectral synthesis. In *Proceedings on Graphics Interface'86/Vision Interface'86*, pages 173–179, 1986.

[Lew89] John-Peter Lewis. Algorithms for solid noise synthesis. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 263–270, 1989.

[LHN05] Sylvain Lefebvre, Samuel Hornus, and Fabrice Neyret. Texture sprites : Texture elements splatted on surfaces. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 163–170, 2005.

[LHW+06] Wen-Chieh Lin, James Hays, Chenyu Wu, Yanxi Liu, and Vivek Kwatra. Quantitative evaluation of near regular texture synthesis algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 427–434. IEEE, 2006.

[LLC+10] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony DeRose, George Drettakis, David S Ebert, John P Lewis, Ken Perlin, and Matthias Zwicker. A survey of procedural noise functions. *Computer Graphics Forum*, 29(8) :2579–2600, 2010.

[LLDD09] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. Procedural noise using sparse gabor convolution. *ACM Transactions on Graphics (TOG)*, 28(3) :1–10, 2009.

[LSD21] Nicolas Lutz, Basile Sauvage, and Jean-Michel Dischler. Cyclostationary gaussian noise : theory and synthesis. *Computer Graphics Forum*, 40(2) :239–250, 2021.

[LSD23] Nicolas Lutz, Basile Sauvage, and Jean-Michel Dischler. Preserving the autocovariance of texture tilings using importance sampling. *Computer Graphics Forum*, 42(2) :347–358, 2023.

[Mik22] Morten S. Mikkelsen. Practical real-time hex-tiling. *Journal of Computer Graphics Techniques (JCGT)*, 11(3) :77–94, August 2022.

[MST+21] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf : Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1) :99–106, 2021.

[MVN68] Benoit B Mandelbrot and John W Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM review*, 10(4) :422–437, 1968.

[NRS82] Alan Norton, Alyn P Rockwood, and Philip T Skolmoski. Clamping : A method of antialiasing textured surfaces by bandwidth limiting in object space. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, pages 1–8, 1982.

[OAH+04] Marc Olano, Kurt Akeley, John C Hart, Wolfgang Heidrich, Michael McCool, Jason L Mitchell, and Randi Rost. Real-time shading. In *ACM SIGGRAPH 2004 Course Notes*, pages 1–es. ACM, 2004.

[OB10] Marc Olano and Dan Baker. Lean mapping. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, page 181–188, New York, NY, USA, 2010. Association for Computing Machinery.

[Pea85] Darwyn R Peachey. Solid texturing of complex surfaces. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 279–286, 1985.

[Per85] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3) :287–296, 1985.

[Per02] Ken Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, 2002.

[PFH00] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, 2000.

[PH89] Ken Perlin and Eric M Hoffert. Hypertexture. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 253–262, 1989.

[PJH23] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering : From theory to implementation*. MIT Press, 2023.

[PN01] Ken Perlin and Fabrice Neyret. Flow noise. In *28th International Conference on Computer Graphics and Interactive Techniques (Technical Sketches and Applications)*, page 187. Siggraph, 2001.

[QY01] Xuejie Qin and Yee-Hong Yang. A generalized cellular texture basis function. *Department of Computer Science, University of Saskatchewan*, 2001.

[SA79] Bruce Schachter and Narendra Ahuja. Random pattern generation processes. *Computer Graphics and Image Processing*, 10(2) :95–114, 1979.

[Sau18] Basile Sauvage. Contributions à l'analyse et à la synthèse de l'apparence d'ob jets 3d numériques. *Mémoire d'habilitation à diriger des recherches*, 2018.

[SPF+23] Hugo Schott, Axel Paris, Lucie Fournier, Eric Guérin, and Eric Galin. Large-scale terrain authoring through interactive erosion simulation. *ACM Transactions on Graphics*, 42, 2023.

[Sta97] Jos Stam. *Aperiodic texture mapping.* European Research Consortium for Informatics and Mathematics, 1997.

[TEZ+19] Thibault Tricard, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, and Sylvain Lefebvre. Procedural phasor noise. *ACM Transactions on Graphics*, 38(4) :Article No. 57 :1–13, July 2019.

[TSP21] Sheldon Taylor, Owen Sharpe, and Jiju Peethambaran. Prime gradient noise. *Computational Visual Media*, 7 :349–362, 2021.

[VW91] Jarke J Van Wijk. Spot noise texture synthesis for data visualization. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 309–318, 1991.

[Wil83] Lance Williams. Pyramidal parametrics. In *10th annual conference on Computer graphics and interactive techniques*, pages 1–11, 1983.

[Wor96] Steven Worley. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294, 1996.