HAL
open science

Exemplar based texture synthesis : models and applications

Lara Raad Cisa

To cite this version:

Lara Raad Cisa. Exemplar based texture synthesis : models and applications. General Mathematics [math.GM]. Université Paris Saclay (COmUE), 2016. English. NNT : 2016SACLN042 . tel-01493810

HAL Id: tel-01493810

https://theses.hal.science/tel-01493810v1

Submitted on 22 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNT : 2016SACLN042

THESE DE DOCTORAT
DE
L'UNIVERSITE DE PARIS-SACLAY
PREPAREE A
L'ECOLE NORMALE SUPERIEURE DE CACHAN
(ECOLE NORMALE SUPERIEURE DE PARIS-SACLAY)

ÉCOLE DOCTORALE N°574
Eocle doctorale de mathématiques Hadamard (EDMH)

Spécialité de doctorat : Mathématiques appliquées

Par

Mme Lara Raad

Exemplar Based Texture Synthesis: Models and Applications

Thèse présentée et soutenue à Cachan, le 3 octobre 2016:

Composition du Jury:

M. Yann Gousseau, Professeur, Télécom ParisTech, Président du Jury
M. Jean-François Aujol, Professeur, Institut de Mathématiques de Bordeaux, Rapporteur
M. Gabriel Peyré, Directeur de recherche CNRS, Université Paris Dauphine, Rapporteur
M. Javier Portilla, Professeur,CSIC, Rapporteur M. Pablo Musé, Professeur, Universidad de la Republica, Examinateur
Mme. Agnès Desolneux, Directrice de recherche CNRS, ENS Paris-Saclay, Directrice de thèse
M. Jean-Michel Morel, Professeur des Universités, ENS Paris-Saclay, Co-directeur de thèse

Version of March 20, 2017 at 16:28.

To Petra and Pierre

Contents

1 Introduction **19**

2 Locally Gaussian models **31**
2.1 Introduction 31
2.2 The patch Gaussian model 33
2.3 The locally Gaussian texture synthesis algorithm 37
2.4 Experiments 40
2.5 Conclusion 41

3 Conditional Gaussian model: a multiscale algorithm **45**
3.1 Introduction 45
3.2 Gaussian patches 47
3.3 Conditional Gaussian models 49
3.4 A multiscale generalization 53
3.5 Experiments 58
3.6 Conclusion 68

4 Midway patch blending **69**
4.1 Introduction 69
4.2 Optimal transport - midway equalization algorithm 71
4.3 Midway blending 72
4.4 Experiments 85
4.5 Conclusion 87

5 Can we emulate large textures? **99**
5.1 Introduction 99
5.2 Large natural textures 100
5.3 Exemplar based texture synthesis methods 101
5.4 Applying the classical methods 107
5.5 Masking repetitions with anamorphosis 111
5.6 Random mosaics 126
5.7 Synthesizing low frequencies 126
5.8 Mixing texture images 130
5.9 Conclusion 131

A Efros and Freeman Image Quilting Algorithm for Texture Synthesis **133**
A.1 Introduction 133
A.2 Algorithm Description 134
A.3 Implementation 142
A.4 Experiments 146
A.5 Conclusion 150

Bibliography **155**

Introduction

Motivation

La synthèse de texture par l'exemple a pour but de générer à partir d'un échantillon de texture de nouvelles images qui sont perceptuellement équivalentes à celle de départ. Les méthodes peuvent se regrouper en deux catégories : les méthodes paramétriques et les méthodes non-paramétriques à base de patchs. Le premier groupe a pour but de caractériser une image de texture à partir d'un ensemble de statistiques qui définissent un processus stochastique sous-jacent. Les résultats visuels de ces méthodes sont satisfaisants, mais seulement pour un groupe réduit de types de texture. La synthèse pour des images de textures ayant des structures très contrastées peut échouer. La deuxième catégorie d'algorithme découpe, puis recolle de manière consistente des voisinages locaux de l'image de départ pour générer de nouvelles configurations plausibles de ces voisinages (ou patchs). Les résultats visuels de ces méthodes sont impressionnants. Néanmoins, on observe souvent des répétitions verbatim de grandes parties de l'image d'entrée qui du coup peuvent être reproduites plusieurs fois. De plus, ces algorithmes peuvent diverger, reproduisant de façon itérative une partie de l'image de l'entrée en négligeant le reste.

Dans cette thèse on présente une approche combinant des idées des méthodes paramétriques et des méthodes non paramétriques à base de patchs. On l'appelle *synthèse localement Gaussienne* et elle est présentée dans le Chapitre 2. On préserve dans cette nouvelle méthode les aspects positifs de chaque approche : la capacité d'innover des méthodes paramétriques, et la capacité de générer des textures fortement structurées des méthodes non paramétriques à base de patchs. Pour ce faire, on construit un modèle Gaussien multidimensionnel des auto-similarités d'une image de texture. Une nouvelle image est générée patch par patch, où pour chaque patch une loi normale multidimensionnelle est estimée à partir des patchs de l'image d'entrée qui lui ressemble le plus, et ensuite échantillonné à partir de cette loi. Les images synthétisées sont donc différentes point à point de la texture de départ. Ainsi, on obtient des résultats qui sont visuellement supérieurs à ceux obtenus avec les méthodes paramétriques et qui sont comparables à ceux obtenus avec les méthodes non-paramétriques à base de patchs tout en utilisant une paramétrisation locale de l'image. La thèse s'attache aussi à résoudre une autre difficulté des méthodes à base de patchs : le choix de la taille du patch. Afin de réduire significativement cette dépendance, on propose une extension multi-échelle de la méthode au Chapitre 3.

Les méthodes à bases de patchs supposent une étape de recollement. En effet, les patchs de l'image synthétisée se superposent entre eux, il faut donc gérer le recollement dans ces zones. La première approche qu'on a considérée consiste à prendre en compte cette contrainte de superposition dans la modélisation des patchs. Cette approche est décrite au Chapitre 3. Les expériences montrent que cela est satisfaisant pour des images de textures périodiques ou pseudo-périodiques

et qu'en conséquence l'étape de recollement peut être supprimée pour ces textures. Cependant, pour des images de textures plus complexes ce n'est pas le cas, ce qui nous a menée à proposer une nouvelle méthode de recollement inspirée du transport optimal qui est détaillée au Chapitre 4.

Cette thèse se termine avec une étude complète de l'état de l'art en génération d'images de textures naturelles dans le Chapitre 5, qui tient lieu aussi de conclusion. L'étude que nous présentons montre que, malgré les progrès considérables des méthodes de synthèse à base d'exemples proposées dans la vaste littérature, et même en les combinant astucieusement, celles-ci sont encore incapables d'émuler des textures complexes et non stationnaires.

Dans cette introduction on donne une vue d'ensemble de tous les chapitres de la thèse avec leur résultats principaux et conclusions. On termine avec la liste de publications en lien avec ces travaux.

Chapitre 2 : Modèles localement Gaussien

Les principales approches de la synthèse d'images de textures sont les méthodes paramétriques et les méthodes non-paramétriques à base de patch. Le premier modèle de texture caractérise l'image par un ensemble de statistiques. L'algorithme de synthèse associé estime ces statistiques à partir de l'échantillon de texture et génère une nouvelle image. La texture générée est en général moins précise que celle d'entrée. Le second modèle se résume à une démarche intelligente de "copier-coller" qui met ensemble des morceaux de la texture d'entrée. La texture générée peut présenter des répétitions verbatim de grandes parties de l'échantillon de texture. Au Chapitre 2 nous proposons de mélanger les deux approches mentionnées en utilisant un modèle de texture localement Gaussien dans l'espace des patchs. Cela permet de synthétiser des textures qui sont partout différentes de celle d'entrée tout en ayant une meilleur qualité visuelle par rapport aux résultats de méthodes paramétriques.

Inspiré de la méthode d'Efros et Freeman [15] nous synthétisons une nouvelle texture patch par patch. A différence de [15] on va supposer que chaque nouveau patch synthétisé suit une loi normale multidimensionnelle qui est apprise à partir d'un ensemble de patchs pris dans l'échantillon de texture. Cet ensemble est constitué des patchs les plus semblables, pour la norme L^2, au patch qui est en cours de synthèse. Les modèles localement Gaussien ont été utilisés pour le débruitage d'image [38] avec des résultats très satisfaisants. Pour la synthèse de texture, cette approche nous permets de maintenir la cohérence entre patchs par rapport à l'image d'entrée tout en créant de nouveaux patchs qui n'existent a priori pas dans l'échantillon de texture mais qui leur sont visuellement semblables.

Le modèle de patch Les lois normales multidimensionnelles impliquées sont définies par leur vecteur moyen $\mu^{(x,y)}$ et leur matrice de covariance $\Sigma^{(x,y)}$. Pour un patch donné $p_u^{(x,y)}$, sous sa forme vectorielle, et qui est de taille $1 \times n^2$, les paramètres de la Gaussienne sont estimés à partir d'un ensemble de patchs que l'on dénote $\mathcal{U} = \{p_u^{(x_i,y_i)}, i = 1, \ldots, m\}$, où les $p_u^{(x_i,y_i)},\ i = 1, \ldots, m$ sont les m patchs les plus semblables à $p_u(x, y)$ selon la norme L^2. Le vecteur $\tilde{p}_u^{(x,y)}$ défini comme dans l'équation (1.1) suit une loi normale $\mathcal{N}(\mu^{(x,y)}, \Sigma^{(x,y)})$. On observe que les variances de ces modèles confirment un degré d'innovation raisonnable pour les vecteurs échantillonnés.

$$\tilde{p}_u^{(x,y)} = \frac{1}{\sqrt{m-1}} \sum_{i=1}^{m} a_i (p_u^{(x_i,y_i)} - \mu^{(x,y)}) + \mu^{(x,y)},\ a_i \sim \mathcal{N}(0,1),\ i = 1, \ldots, m \tag{1}$$

L'algorithme de synthèse localement Gaussienne On enchaîne sur la description d'un algorithme de synthèse qui génère une nouvelle image de texture en collant entre eux des patchs échantillonnés selon des lois normales multidimensionnelles (1.1). On note cet algorithme par LG. Cette méthode est itérative : les patchs sont synthétisés en *raster-scan* (de haut en bas, de gauche à droite). Le but de chaque itération est de générer un nouveau patch $p_w^{(x,y)}$ qui est partiellement défini sur une région qu'on appelle "région d'intersection" (voir Figure 1). La partie connue du patch est utiliser pour définir l'ensemble $\mathcal{U}$ qui déterminera la loi normale du nouveau patch. Le patch $p_w^{(x,y)}$ est généré comme défini dans (1.1). La dernière étape consiste à recoller le nouveau patch dans l'image de sortie. Pour cela la méthode utilisée est celle proposée dans [15]. En conclusion, l'algorithme de synthèse proposé génère une nouvelle image de texture w qui est visuellement semblable à la texture d'entrée et pourtant elle n'est pas composée des patchs existant dans l'image d'entrée. Par conséquent, la méthode proposée est capable de diminuer quelques uns des aspects négatifs des méthodes de synthèse dites paramétriques et non-paramétriques à base de patchs.

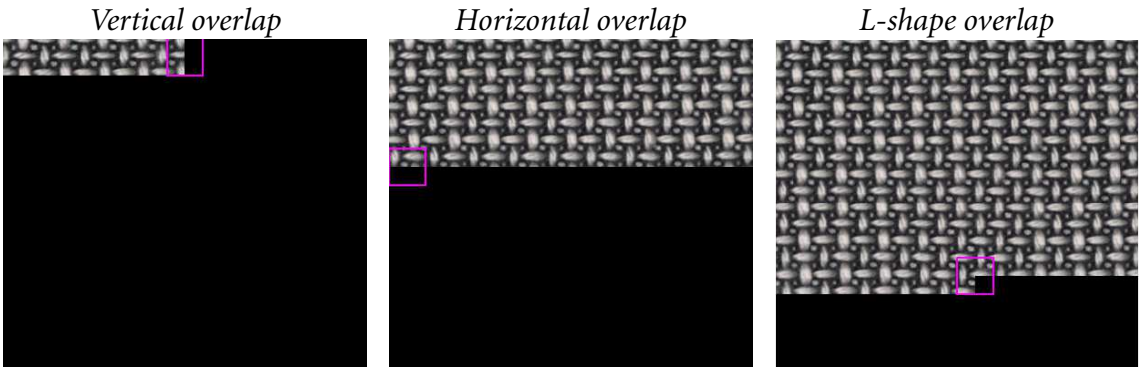


FIGURE 1 – On montre trois itérations différentes de la synthèse d'une texture. A chaque itération un patch est synthétisé. Il est représenté en rose à chaque itération. De gauche à droite on observe les trois cas d'intersection : vertical, horizontal et en forme de L.

L'algorithme reste cependant dépendant du choix de la taille du patch n et de m la taille de l'ensemble $\mathcal{U}$, comme on peut observer dans la Figure 2 et qui devrait être ajuster selon l'échantillon detexture. Notre algorithme a une faible complexité comparé aux méthodes de débruitage à base de patch comme [39], [9]. Une possibilité pour réduire la dépendance de la méthode à la taille du patch est de travailler avec une approche multi-échelle. Cette extension est présentée au Chapitre 3. Contrairement aux méthodes de synthèse de texture paramétriques et tout comme les méthodes non-paramétriques à base de patchs, notre algorithme n'est pas forcé à respecter les statistiques globales de l'image d'entrée. Cet aspect est également considéré au Chapitre 3. Un dernier point tout aussi important est traité aux Chapitres 3 et 4 et qui concerne le recollement d'un nouveau patch dans une partie de l'image déjà synthétisée.

Chapitre 3 : Modèle conditionnel Gaussien : un algorithme multi-échelle

Dans ce chapitre on étend le modèle présenté au Chapitre 2 et on représente les auto-similarités d'une image de texture avec des lois normales conditionnelles aux valeurs de l'intersection des

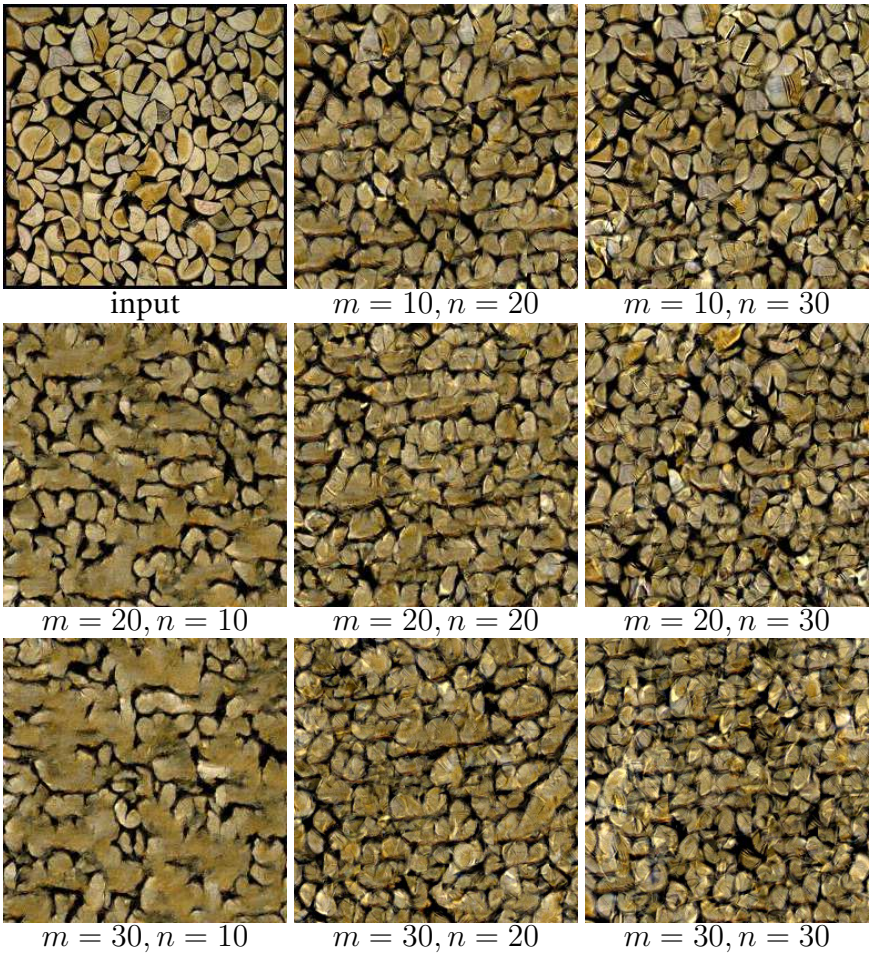
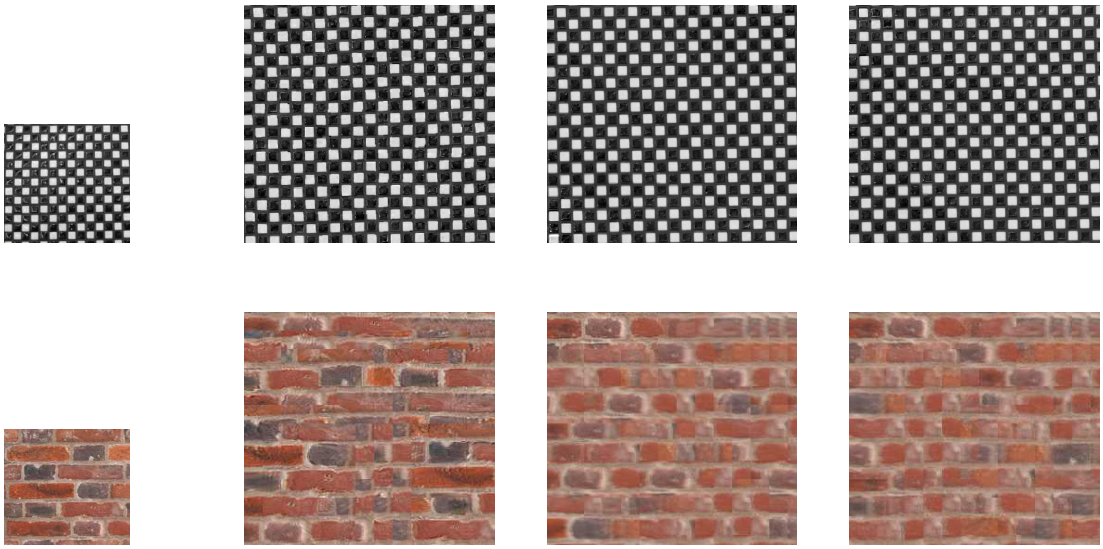


FIGURE 2 – Influence des paramétres. L'image du coin supérieur gauche et la texture d'entrée. Les images restantes sont des synthéses en variant la taille du patch n et m la taille de l'ensemble $\mathcal{U}$. De gauche à droite $n = 10, 20, 30$. Du haut vers le bas $m = 10, 20, 30$. Tous ces résultats correspondent à une intersection o de taille $n/2$.

patchs. Notre but est que le modèle considéré prenne en compte l'étape de recollement des patchs. Aussi, nous proposons de généraliser l'algorithme de synthèse localement Gaussienne (LG en utilisant une approche multi-échelle où la texture est considérée comme étant localement Gaussienne aux différentes échelles. Cette approche est évaluée en utilisant une grande variété d'images de texture (réelles et synthétiques) et les résultats montrent l'efficacité de la métode proposée pour une grande gamme de texture.

Modèle conditionnel Gaussien La première question qu'on s'est posé était si l'étape de recollement des patchs pouvait être supprimée et comment ? Dans le Chapitre 2, on a adopté l'approche d'Efros et Freeman [15] où pour chaque nouveau patch synthétisé son bord, sur la région d'intersection, est redéfini. Dans ce chapitre on considère deux modèles qui sont contraints aux valeurs de la région d'intersection : le modèle Gaussien conditionnel (CLG) et le modèle Gaussien conditionnel régularisé ($RCLG$). Ces deux solutions montrent des résultats satisfaisants pour des images de texture périodiques et pseudo-périodiques mais elles ont tendance a échoué pour des textures plus complexes. On peut voir dans la Figure 3 deux exemples de textures synthétisées en appliquant les

différents modèles.



input **LG model** **CLG model** **RCLG model**

FIGURE 3 – *Comparaison des modèles de patchs.* De gauche à droite : texture d'entrée, synthèse LG, synthèse CLG et synthèse $RCLG$. Aucune étape de recollement a été appliquée pour les trois modèles montrès. Les paramètres pour ces synthèses sont $n = 40$, $m = 30$ et $o = 0.5n$.

Généralisation multi-échelle La deuxième question qu'on s'est posé par rapport aux méthodes à base de patchs était comment gérer leur forte dépendance à la taille du patch utilisé en particulier quand on travaille avec des images fortement structurées. En effet, les textures structurées ont en général des détails à différentes échelles qui ne peuvent pas être capturés avec une seule taille de patch. L'approche multi-échelle que nous envisageons peut être résumée en quelques lignes et qu'on dénote $MSLG$. La méthode commence à l'échelle la plus grossière où on utilise l'algorithme présenté au Chapitre 2 avec une différence : l'étape de recollement des patchs est remplacée par une combinaison linéaire des valeurs sur la région d'intersection. Aux échelles suivantes la synthèse est faite telle qu'à chaque échelle les détails fins sont rajoutés à l'image synthétisée à l'échelle précédente. On suppose que les détails suivent une loi normale multidimensionnelle. En résumé, une image de plus faible résolution est synthétisée et utilisée comme guide à laquelle on rajoute les détails des échelles restantes. Une description complète est fournie dans le Chapitre 3. Dans la Figure 4 on montre quelques résultats de synthèse multi-échelle.

Chapitre 4 : Recollement *midway* de patchs

Dans le Chapitre 4 on s'intéresse à la phase de recollement de patchs des méthodes de synthèse à base de patchs. Un des défis de cette étape est de recoller des patchs qui diffèrent considérablement sur leur bord commun. Les techniques utilisées pour la synthèse de texture ont tendance à échouer dans des cas de fort contraste et par conséquent on observe la transition entre les patchs (voir Figure 5).

On propose une nouvelle méthode qui se base sur la théorie du transport optimal afin de fusionner deux patchs tout en ayant une transition lisse entre eux. Cette approche peut se décomposer

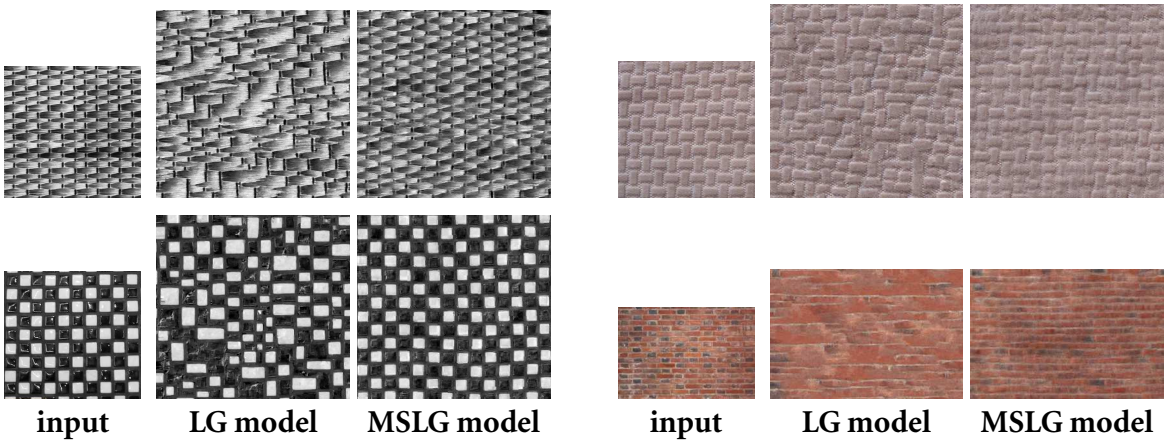


FIGURE 4 – *Comparaison de la synthèse localement Gaussienne (LG) à la synthèse multi-échelle ($MSLG$).* De gauche à droite : texture d'entrée, synthèse LG et synthèse $MSLG$. Les paramètres utilisés pour ces synthèses sont $n = 20$, $m = 50$ et $o = 0.5n$. Pour le multi-échelle le nombre d'échelles K utilisées est $K = 3$. Pour les deux modèles on a utilisé le même patch d'initialisation.

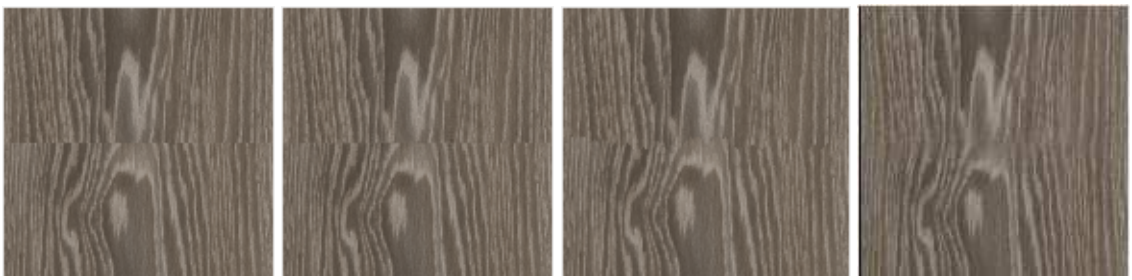


FIGURE 5 – Teaser. De gauche à droite : deux patchs d'entrée, deux patchs recollés avec Poisson editing [47], deux patchs recollés avec [15], deux patchs recollés avec le recollement *midway*.

en deux étapes. La première consiste à définir un bord commun aux deux patchs qui se trouve à mi-chemin entre leur bords initiaux. Pour obtenir ce bord à mi-chemin deux déformations différentes sont appliquées à chaque patch individuellement. La deuxième étape consiste à propager graduellement ces déformations sur chacun des patchs afin de générer un changement graduel et lisse sur chacun des patchs.

Le signal à mi-chemin La première étape consiste à trouver un signal de dimension un qui se trouve à mi-chemin entre deux signaux de dimension un qu'on dénote s_1 et s_2. Ces signaux, s_1 et s_2, correspondent aux bords des patchs p_1 et p_2 sur lesquels on veut les recoller (voir Figure 6). On considère s_1 et s_2 comme étant deux histogrammes qu'on veut transporter dans l'esprit de la méthode d'égalisation d'histogrammes midway [12]. Le signal s à mi-chemin est défini tel que son *histogramme cumulé* soit la moyenne harmonique des *histogrammes cumulés* de s_1 et s_2. Pour passer des signaux s_1 et s_2 au signal s on applique deux déformations différentes ϕ_1 et ϕ_2 à s_1 et s_2 qu'on appellera les déformations *midway*.

Propagation des déformations *midway* Une fois que les signaux s_1 et s_2 coïncident il faut propager les déformations ϕ_1 et ϕ_2 afin que dans les patchs p_1 et p_2 il n'y ait pas de discontinuités entre s et le reste du patch respectivement. Pour cela on considère deux alternatives. La

première consiste à propager linéairement et graduellement les déformations *midway* ϕ_1 et ϕ_2 dans chaque patch. La deuxième consiste à définir des signaux intermédiaires (pour chaque patch) qui résultent d'une combinaison convexe entre l'inverse de l'*histogramme cumulé* de s et l'inverse de l'*histogramme cumulé* du signal provenant du bord qui délimite la zone de propagation dans chacun des patchs. La première méthode de propagation rend des résultats où les déformations appliquées sont très lisses. Cependant, des problèmes de normalisation sont évidents et résultent du fait de considérer les signaux comme des histogrammes (qu'il faut donc normaliser pour qu'ils aient la même masse). La deuxième méthode est exempte de ces problèmes de normalisation. Néanmoins, les déformations sont bien plus brusques et peuvent être quelque fois plus visibles que dans le cas linéaire.

Nous avons évalué en détail les méthodes proposées. Dans une première partie nous avons analysé le recollement *midway* en sélectionnant deux patchs de manière aléatoire dans une même image de texture. Nous avons comparé le recollement *midway* à deux autres méthodes : Poisson editing [47] et le *quilting* d'Efros et Freeman [15]. Quelques exemples difficiles sont montrés dans la Figure 7. Dans une deuxième partie nous avons testé le recollement *midway* dans l'algorithme de synthèse d'Efros et Freeman [15] et comparé les résultats à ceux de la méthode originale. Ces résultats nous ont permis de conclure que la méthode proposée est une possible alternative pour le recollement des patchs dans les méthodes de synthèse à base de patchs.

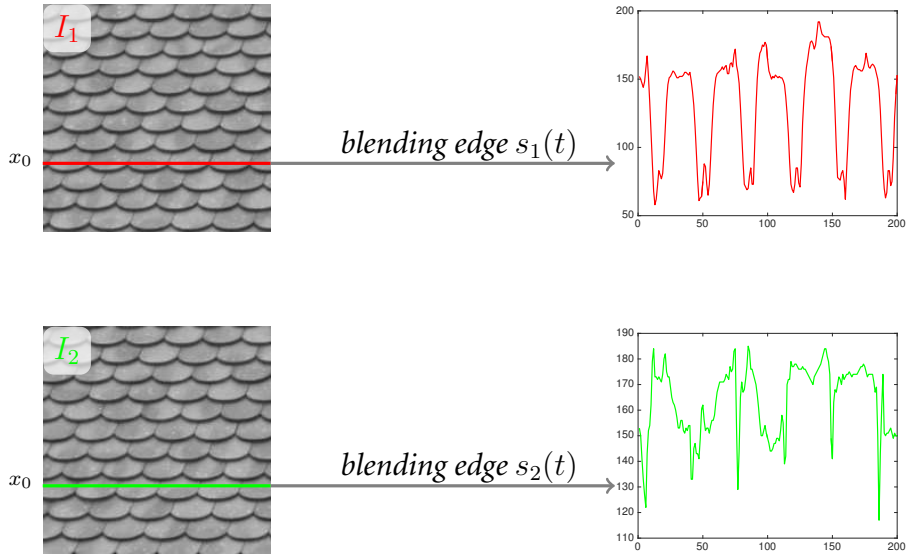


FIGURE 6 – Illustration des signaux de dimension un à recoller. Les patchs à recoller sont représentés par I_1 et I_2 et les bords de recollement sont représentés par s_1 et s_2.

Chapitre 5 : Peut-on faire de la synthèse à base de *grandes* textures ?

Dans ce chapitre conclusif on veut explorer quelles sont les limitations des méthodes de synthèse de texture à partir d'exemples quand on utilise de grandes images de texture en entrée (voir Figure 8). Contrairement aux méthodes de synthèse par l'exemple classique qui utilisent des images d'entrée très petites le but est de synthétiser de très grandes images de texture. On étudie ce problème avec

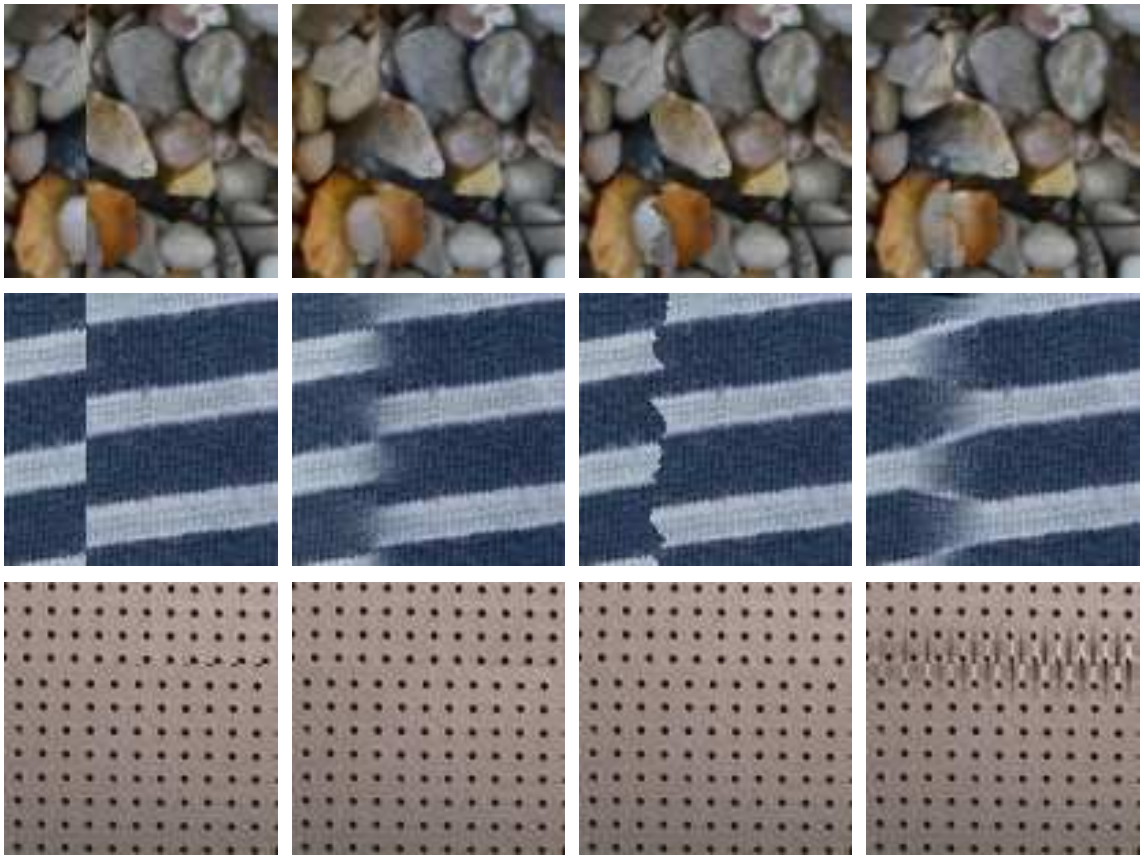


FIGURE 7 – Quelques exemples difficiles de recollement de patchs. De gauche à droite : entrées, recollement avec Poisson editing [47], recollement avec [15] et recollement *midway*.

les outils qui sont à notre disposition : les méthodes qui ressortent de la vaste littérature de la synthèse par l'exemple ainsi que nos contributions des Chapitres 2 et 3, et des combinaisons de ces méthodes.

Pour anticiper nos conclusions aucune des méthodes existantes n'est capable de synthétiser correctement de grandes images de texture. Néanmoins, nous explorons des combinaisons astucieuses de ces méthodes qui présentent des résultats intéressants mais pas parfaits. Les méthodes existantes montrent des résultats très satisfaisants quand elles sont appliquées à de tout petit bout des grandes texture d'entrée (voir Figure 9). Nous observons que ces grandes textures sont très différentes des textures académiques utilisées jusqu'à présent. Au fait elles montrent des variations locales très prononcées, qui peuvent être vues comme différentes textures, même si leur aspect global est celui d'une seule texture (Figure 9). Cette étude est une bonne expérience pour mettre en épreuve toutes les méthodes de synthèse à base d'exemple classiques.

Cette étude commence par tester différentes méthodes de synthèse de texture par l'exemple notamment celles d'Efros et Freeman [15], Portilla et Simoncello [50], Galerne et al. [21] et l'approche multi-échelle présenté au Chapitre 3 appliquées à différentes images de texture naturelles et de très grande taille comme du bois, de la pierre et du métal. On a observé que les résultats de synthèse sur les images de bois n'étaient pas satisfaisants. Néanmoins quand on a considéré des sous-images prises dans ces grandes textures de bois et où aucune structure saillante apparaît, les résultats de synthèse sont bons. On peut rapidement conclure que la perception qu'on a de ces images comme étant une seule texture n'est pas "vraie". Un exemple est montré avec les Figures 8 et 9. Le deuxième

FIGURE 8 – Texture de bois. L'image originale est de taille 7087×23622 pixels. Pour les expériences on a travaillé sur des images réduites d'un facteur 16.

FIGURE 9 – On montre quatre sous-images prise de l'image de la Figure 1.8 à la résolution originale. Ces sous-images sont de taille 500×500 pixels. On observe que chacune de ces quatre images représentent des textures bien différentes appartenants à une seule "grande" texture.

jeu d'expériences consiste à combiner ces méthodes de synthèse, en particulier :

- La méthode d'Efros et Freeman [15] combinée à la synthèse localement Gaussienne ($LG + EF$)
- La méthode de Portilla et Simoncelli [50] combinée à la synthèse localement Gaussienne ($LG + PS$)

Les combinaisons précédentes améliorent significativement les résultats obtenus dans la première étape en particulier la combinaison LG+PS. Pourtant cette alternative n'est pas suffisante dans le cas des images de bois. Par la suite on montre différentes tentatives pour synthétiser ces images comme par exemple l'application de déformation géométriques en espérant varier l'aspect des structures fortement saillantes. Une astuce qui est souvent utile est de permuter de manière aléatoire des sous-parties de l'image d'entrée afin de décomposer ses détails fortement reconnaissables comme dans le cas du bois et de les disposer de façon aléatoire dans la nouvelle image. Une dernière expérience que nous avons considérée est de varier les basses fréquences de l'image d'entrée en utilisant la méthode de Galerne et al. [21] tout en laissant inchangé les hautes fréquences de l'image d'entrée. Dans ces trois essais de synthèse nous observons des comportements intéressants pour les différentes textures. Pour illustration trois exemples sont montrés dans la Figure 10.

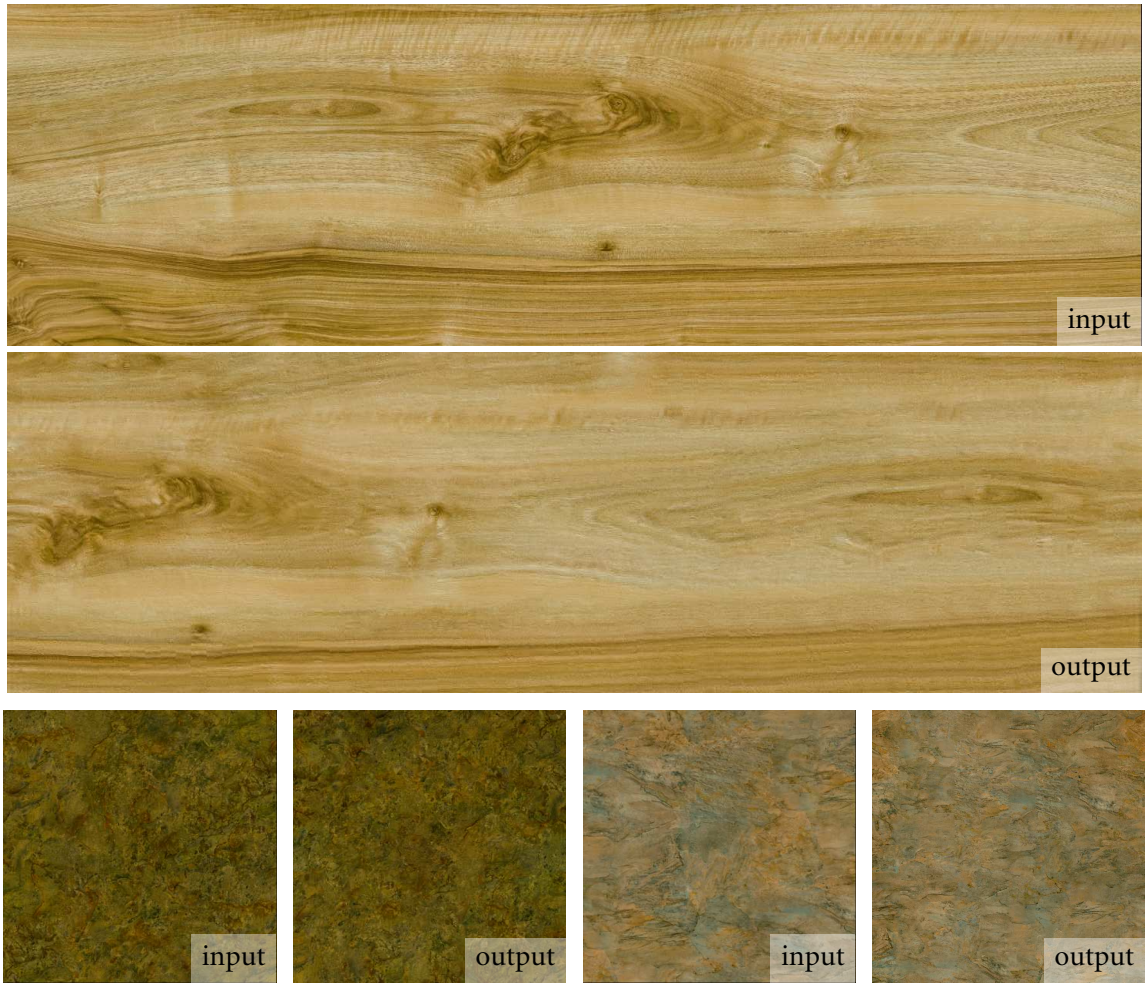


FIGURE 10 – Résultats de synthèse pour différentes “grande” textures.

Liste de Publications

- L. Raad, A. Desolneux and J.-M. Morel. *Locally Gaussian Exemplar Based Texture Synthesis*. In IEEE International Conference on Image Processing (ICIP), pages 4667 - 4671, 2014. IEEE.
- L. Raad, A. Desolneux and J.-M. Morel. *Conditional Gaussian Models for Texture Synthesis*. In International Conference on Scale Space and Variational Methods in Computer Vision (SSVM), pages 474 - 485, 2015. Springer International Publishing.
- L. Raad, A. Desolneux and J.-M. Morel. *Multiscale Exemplar Based Texture Synthesis by Locally Gaussian Models*. In IberoAmerican Congress on Pattern Recognition (CIARP),pages 435 - 443, 2015. Springer Iternational Publishing.
- L. Raad, A. Desolneux and J.-M. Morel. *Conditional and multiscale locally Gaussian models for texture synthesis*. In Journal of Mathematical Imaging and Vision (JMIV), pages 1 - 20, 2016.
- L. Raad and B. Galerne. *Efros and Freeman Image Quilting Algorithm for Texture Synthesis*. Submitted to Image Processing On Line (IPOL).

- L. Raad, A. Desolneux and J.-M. Morel. *Midway blending as a new technique for patch quilting*. To be submitted.
- L. Raad, A. Desolneux and J.-M. Morel. *Can exemplar based texture synthesis models emulate large textures ?* To be submitted.

1 Introduction

Motivation

Exemplar based texture synthesis is the process of generating, from an input texture sample, new texture images that are perceptually equivalent to the input. There are roughly two main categories of algorithms: the statistics-based methods and the non-parametric patch-based methods. The statistics-based methods aim at characterizing a given texture sample by estimating a set of statistics which will define an underlying stochastic process. The new images will then be samples of this stochastic process, i.e. they will have the same statistics as the input sample. The question here is what would be the appropriate set of statistics to yield a correct synthesis for the wide variety of texture images? The results of statistical methods are satisfying but only on a small group of textures, and often fail when important structures are visible in the input. The non-parametric patch-based methods reorganize local neighborhoods from the input sample in a consistent way to create new texture images. These methods return impressive visual results. Nevertheless, they often yield verbatim copies of large parts of the input sample. Furthermore, they can diverge, starting to reproduce iteratively one part of the input sample and neglecting the rest of it, thus growing what experts call "garbage".

In this thesis we propose a technique combining ideas from the statistical based methods and from the non-parametric patch-based methods. We call it the *locally Gaussian* method and it is presented in Chapter 2. The method retains the positive aspects of both categories: the innovation capacity of the parametric methods and the ability to synthesize highly structured textures of the non-parametric methods. To this aim, the self-similarities of a given input texture are modeled with conditional multivariate Gaussian distributions in the patch space. A new image is generated patch-wise, where for each given patch a multivariate Gaussian model is inferred from its nearest neighbors in the patch space of the input sample, and hereafter sampled from this model. The synthesized textures are therefore everywhere different from the original. In general, the results obtained are visually superior to those obtained with statistical based methods, which can be explained by the use of a local parametric model instead of a global one. On the other hand, our results are comparable to the visual results obtained with the non-parametric patch-based methods. This dissertation addresses in Chapter 3 another weakness of patch-based methods. They are strongly dependent on the patch size, which has to be decided manually. It is therefore crucial to fix a correct patch size for each synthesis. Since texture images have, in general, details at different scales, we extend the method to a multiscale approach which reduces the strong dependency of the method on the patch size.

Patch based methods involve a stitching step. Indeed, the patches used for the synthesis process overlap each other. This overlap must be taken into account to avoid any transition artifact

from patch to patch. Our first attempt to deal with it was to consider directly the overlap constraints in the local parametric model. This is described in Chapter 3. The experiments show that for periodic and pseudo-periodic textures, considering these constraints in the parametrization is enough to avoid the stitching step. Nevertheless, for more complex textures it is not sufficient. This leads us to suggest a new stitching technique inspired by optimal transport and midway histogram equalization which is detailed in Chapter 4.

This thesis ends with an extensive analysis of the generation of several natural textures in the conclusive Chapter 5. This study shows that, in spite of remarkable progress for local textures, the methods proposed in the extensive literature of exemplar based texture synthesis are still limited when dealing with complex and non-stationary textures.

In this introduction, we give a brief overview of all the chapters and their main results and conclusions. We end it with a list of publications linked to this PhD.

Chapter 2: Locally Gaussian models

The main approaches to texture modeling are the statistics-based methods and the non-parametric patch-based methods. In the first model the texture is characterized by a sophisticated statistical signature. The associated sampling algorithm estimates this signature from the example and produces a genuinely different texture. This texture nevertheless often loses accuracy. The second model boils down to a clever "copy-paste" procedure, which stitches together verbatim copies of large regions of the example. In Chapter 2 we propose to involve a locally Gaussian texture model in the patch space. It permits to synthesize textures that are everywhere different from the original but with better quality than the purely statistical methods.

We model each texture patch by a multivariate Gaussian distribution learned from its similar patches. Inspired by [15], we maintain the idea of searching for patches to stitch together in the original sample. However, instead of using the exact patch taken in the input texture, we sample the stitched patch from its Gaussian model. The locally Gaussian patch models have been proved very useful in image denoising [38]. Our approach permits to maintain the coherence between patches with respect to the input sample, while creating new patches that do not exist in the sample texture but are still perceptually equivalent to it.

The patch model The multivariate Gaussian models involved are defined by their mean vector $\mu^{(x,y)}$ and their covariance matrix $\Sigma^{(x,y)}$. For a given patch $p_u^{(x,y)}$, of size $n \times n$ pixels, these parameters are estimated from the set of the m nearest patches $\mathcal{U} = \{p_u^{(x_i,y_i)}, i = 1, \ldots, m\}$. The sampled vector $\tilde{p}_u^{(x,y)}$ defined in (1.1) follows the distribution $\mathcal{N}(\mu^{(x,y)}, \Sigma^{(x,y)})$. We observed that indeed these models have reasonable variances confirming that effectively the patches simulated have an acceptable degree of innovation.

$$\tilde{p}_u^{(x,y)} = \frac{1}{\sqrt{m-1}} \sum_{i=1}^{m} a_i (p_u^{(x_i,y_i)} - \mu^{(x,y)}) + \mu^{(x,y)},\ a_i \sim \mathcal{N}(0,1),\ i = 1, \ldots, m \tag{1.1}$$

The locally Gaussian synthesis algorithm We follow by describing a synthesis algorithm that generates a new texture image by stitching together patches sampled from multivariate Gaussian distributions (1.1) in the input sample patch space. This method is iterative: the patches are synthesized in a raster-scan order (top to bottom and left to right). The goal of each iteration is to

generate a new patch $p_{\tilde{u}}^{(x,y)}$ that is partially defined on a region called the overlap area (see Figure 1.1). The known part of the patch will define the set of patches $\mathcal{U}$ from which its Gaussian model is inferred. The generated patch $p_{\tilde{u}}^{(x,y)}$ is then sampled as defined in (1.1). The last step consists in stitching the patch into the output texture using the quilting method of [15]. This synthesis algorithm generates a texture that is perceptually equivalent to the sample texture yet not composed of patches existing in the input texture. Thus, this method reduces some of the drawbacks of the statistics-based and the patch-based methods.

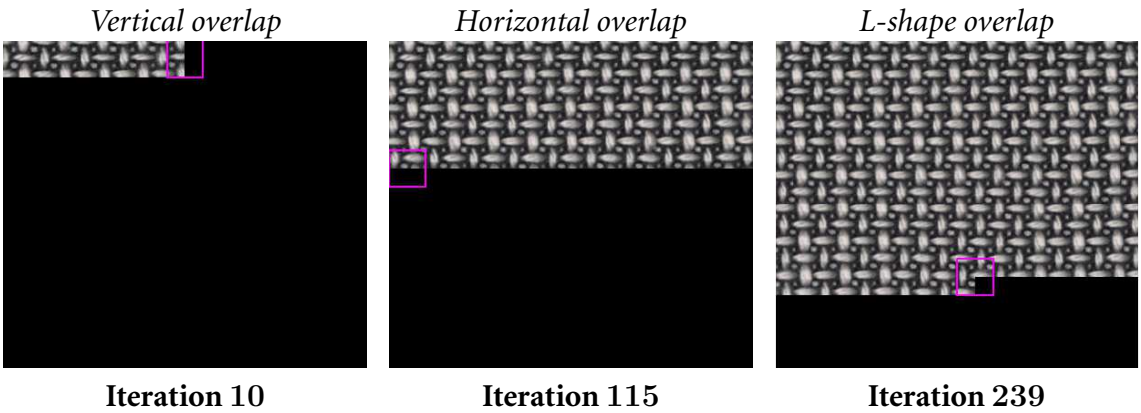


Figure 1.1 – Three different iterations of the synthesis process are shown. At each iteration a patch is being synthesized. This patch is represented by the pink square in the three iterations shown. From left to right the three overlap cases are represented: vertical, horizontal and L-shape.

The algorithm remains dependent on the choice of the patch size n and of the number of nearest neighbours m as can be observed in Figure 1.2, that may have to be adjusted for each texture sample. Our algorithm has low complexity, compared for instance with classic patch-based denoising algorithms [39], [9]. An alternative to reduce the dependency of the method to the patch size is to work in a multiscale approach. This extension is presented in Chapter 3. Unlike the statistics-based algorithms, but like the other patch-based methods, our algorithm is not forced to respect the global statistics of the texture sample. This issue is also considered in Chapter 3. A last and not minor aspect that will be addressed in Chapter 3 and Chapter 4 is the way the patches are actually stitched together.

Chapter 2 is structured as follows. In Section 2.2, we give an estimation method of the Gaussian model for each patch. An analysis of the model's variance is provided. In Section 2.3, the locally Gaussian synthesis algorithm is described. Section 2.4 compares the results with state of the art texture-synthesis methods. The concluding section 2.5 also states the limitations of the method.

Chapter 3: Conditional Gaussian model: a multiscale algorithm

In this chapter, we extend the model of Chapter 2 to model texture self-similarity with conditional Gaussian distributions in the patch space. Our goal is to extend the use of stitching techniques. In continuation the algorithm is also generalized by a multiscale procedure, where texture patches are modeled at each scale as spatially variable Gaussian vectors. This approach is tested over several real and synthetic texture images and its results show the effectiveness of the proposed technique for a wide range of textures.

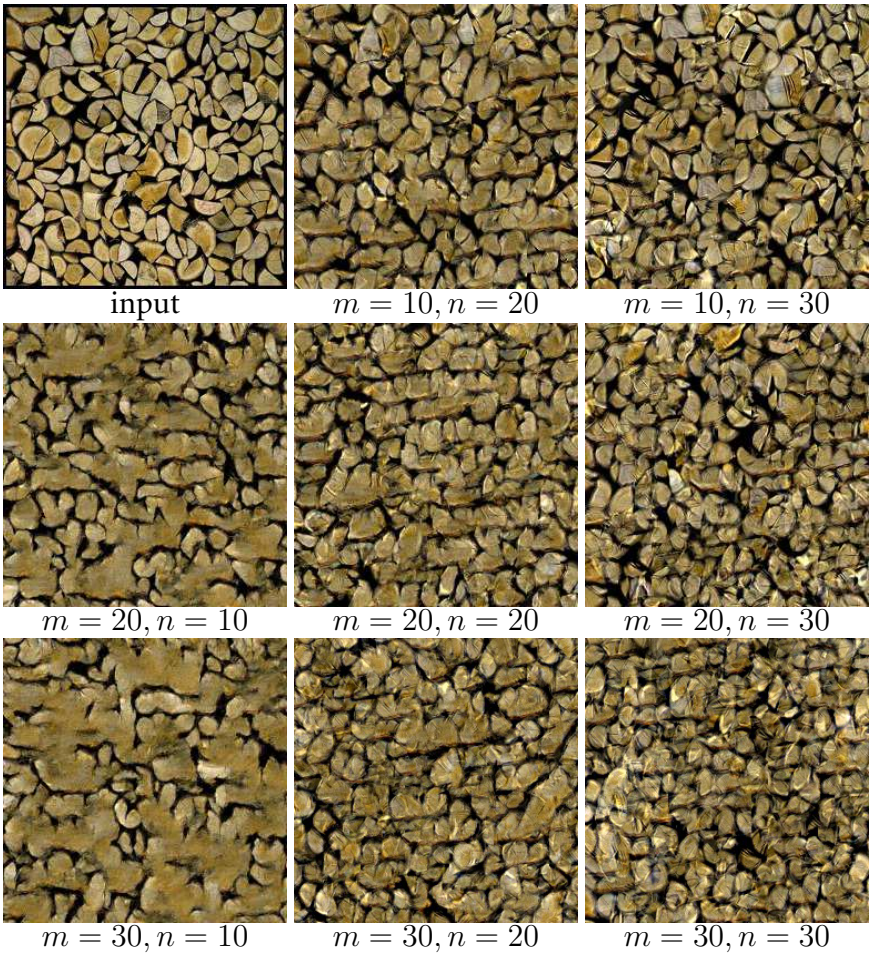


Figure 1.2 – Texture synthesis result for the left top corner texture image. We show the results obtained for different values of m (the number of similar patches) and n (the side patch size). From left to right $n = 10, 20, 0$. From top to bottom, the number of nearest neighbours is $m = 10, 20, 30$. All the results are obtained for an overlap of a half patch size $o = n/2$.

Conditional Gaussian models The first question that was brought up was how to stitch together the patches. In Chapter 2, the Efros and Freeman's approach [15] was adopted, where every new patch overlaps the previously synthesized one, and the overlapped parts are blended. In this chapter we consider two models constrained to respect the values of its overlapping zone: the Conditional Gaussian model (CLG) and the Regularized Conditional Gaussian model (RCLG). This solution shows satisfying results for periodic or pseudo-periodic textures but sometimes fails for more complex ones. Two texture examples are shown in 1.3 where the three models LG, CLG and RCLG are compared.

Multiscale generalization The second and central question brought up by non-parametric patch-based methods is: how to handle the strong dependency of the method on the patch size, in particular when dealing with macro-textures. Indeed, macro-textures show information at different scales that cannot be captured with a unique patch size. The multiscale approach can be summarized in a few sentences. The method begins by a synthesis at the coarsest scale using the local Gaussian method presented in Chapter 2 where the stitching step is replaced by a simple av-

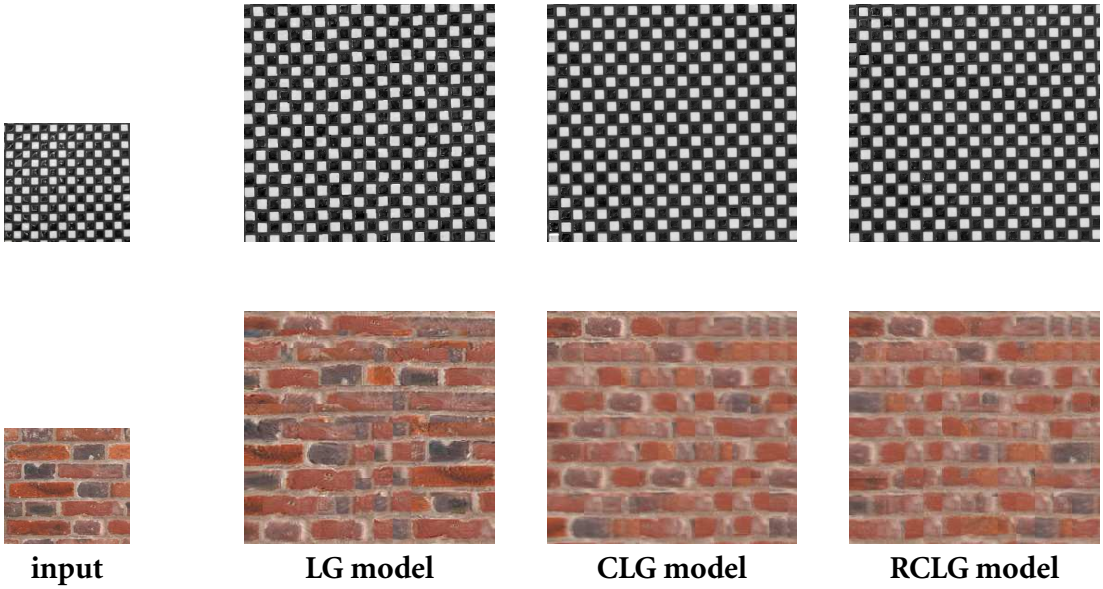


Figure 1.3 – *Patch model comparison.* From left to right: texture sample, synthesis result using LG, CLG and RCLG. No quilting technique was applied to stitch together the simulated patches for the three presented models. The parameters used for all examples are $n = 40$, $m = 30$ and $o = 0.5n$.

erage of the overlapping patches. For the remaining scales a synthesis is performed by using the result of the previous scale and the partial input at the same resolution. At each scale the synthesis is done patch by patch in a raster-scan order. Each new patch, added to the synthesized image, overlaps part of the previously synthesized patch and is the combination of a low resolution patch and a high resolution one sampled from a multivariate Gaussian distribution. The Gaussian distribution of the high frequencies of a given patch is estimated from the high frequencies of its m nearest neighbours in the corresponding scale input image. The synthesis result of the finer scale is the desired output image. A full description and discussion is provided in Chapter 3. In Figure 1.4 we show the strength of the multiscale approach compared to the results of the LG method.

Chapter 3 is structured as follows. In Section 3.2 the local Gaussian (LG) model is described as well as the description of its synthesis method. In Section 3.3 two new patch models are introduced: the conditional local Gaussian model (CLG) and the regularized conditional local Gaussian model (RCLG). Section 3.4 presents the multiscale approach (MSLG). Section 3.5 shows several experiments: a comparison of the three patch models proposed, a comparison to the results of state of the art texture synthesis methods and the influence of the parameters involved in the multiscale texture synthesis algorithm. Conclusions are presented in Section 3.6.

Chapter 4: Midway patch blending

The problem of stitching patches together is the subject of Chapter 4. One of the major challenges is stitching patches that differ considerably along the edges. Stitching methods used in texture synthesis usually fail in these cases, resulting in evident contrast changes between the patches (see Figure 1.5).

We suggest a novel method based on optimal transport theory that manages to merge both patches producing a smooth transition between them. This approach works in two steps: first by

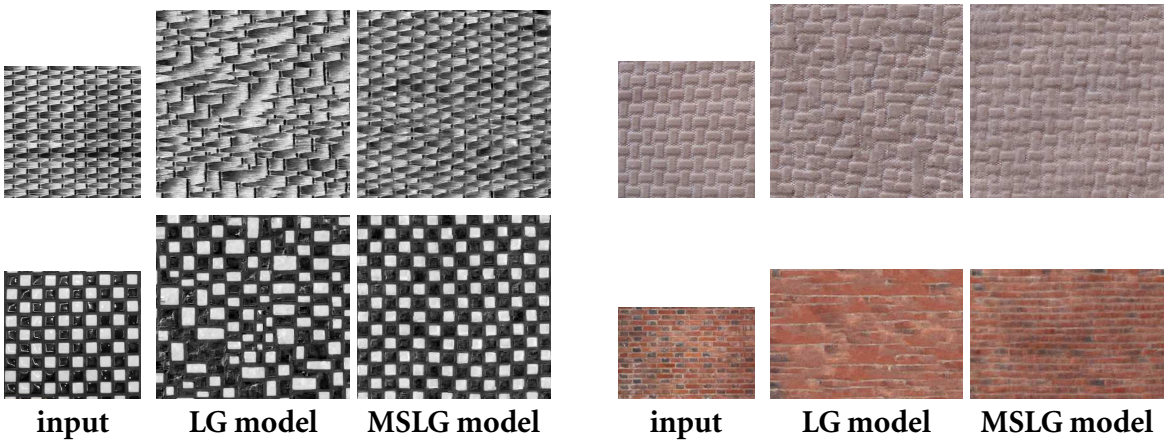


Figure 1.4 – *Comparison of LG (Locally Gaussian) to MSLG (Multiscale Locally Gussian).* From left to right: texture sample, synthesis result using LG, and MSLG. The parameters used for these examples are $n = 20$, $m = 50$ and $o = 0.5n$. For the multiscale synthesis results the number of scales K used is $K = 3$. For both methods we used the same seed patch for each texture sample.

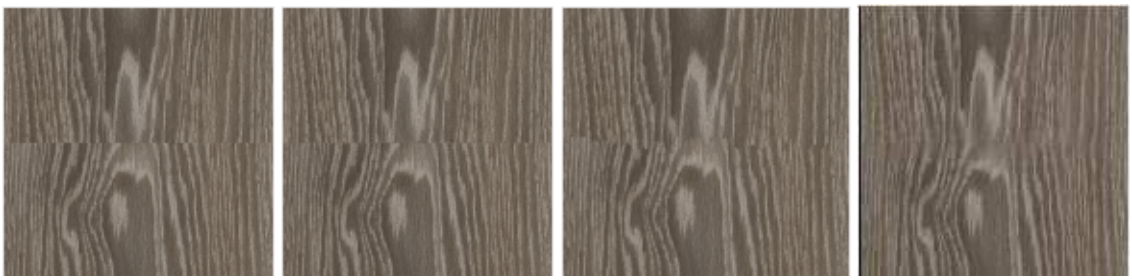


Figure 1.5 – Teaser. From left to right: input, input blended with Poisson editing [47], input blended with quilting [15], input blended with midway blending.

defining a "midway" common restriction for both patches on their common stitchig edge. Midway also defines a different deformation along the edge for each patch. Then this deforamtion is propagated into both patches to produce a smooth transition.

The midway signal The first step can be described as finding a common one dimensional signal which is midway, up to a minimal deformation, between two one dimensional signals denoted s_1 and s_2 (see Figure 1.6). The signals under construction are nothing but the restriction of the patches p_1 and p_2 to the blending edge. The two signals s_1 and s_2 are handled as two histograms on which we apply a geometric deformation derived from the midway histogram equalization [12]. The common intermediary signal s is thus defined as the underlying *histogram* of the harmonic mean of the cumulative functions of s_1 and s_2.

Propagating the midway signal For the propagation step two alternatives are considered. The first one consists in linearly propagating the deformations. The second one considers convex combinations of two inverse cumulative signals: the midway signal and the signal on the curve up to which the deformation is propagated. The first alternative yields smoother results but is problematic regarding the normalization step of the signals (to impose the same mass). The second option avoids the normalization issues yet the propagation is more brutal and sometimes more

conspicuous than the linear alternative.

We thoroughly evaluated the proposed method first by trying it directly on randomly selected patches and comparing it to two stitching techniques: the Poisson editing method [47] and the Efros and Freeman method [15]. A few challenging examples examples are shown in Figure 1.7. We then tried this method as a new stitching approach in the Efros and Freeman synthesis methods[15]. Based on the results, the proposed method proved to be a possible alternative to perform patch stitching in non-parametric patch-based methods.

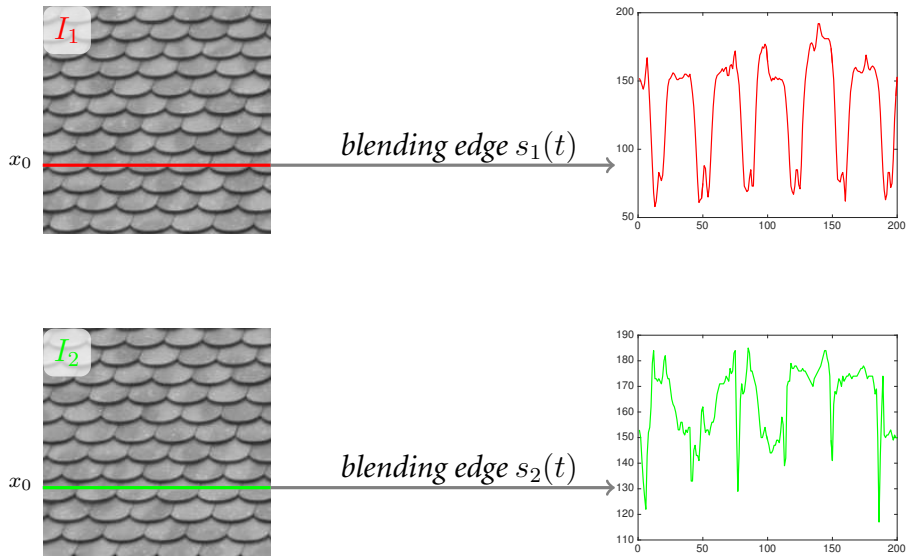


Figure 1.6 – Illustration of the *blending edges*. The two images to blend are represented by I_1 and I_2 and the edges on which the *midway blending* is applied are s_1 and s_2.

Chapter 4 is organized as follows. Section 4.2 recalls the midway equalization algorithm [12]. In section 4.3 the *midway blending* technique is introduced. At first a description on a simple blending curve is given. It is followed by a generalization of the method. A description of the algorithm is provided as well as a discussion on the extension to a color version. As a final contribution of this section we describe how to integrate this method into a simple and efficient texture synthesis method. In section 4.4 experimental results are organized in two parts. The first one compares the midway blending method to [15] and [47]. The second part compares synthesis results of [15] to the same synthesis method using the *midway blending*. Conclusions are given in section 4.5.

Chapter 5: Can we emulate large textures?

In this conclusive chapter we aim at analyzing the limits of exemplar-based texture methods when using large natural texture images (see Figure 1.8), a new problem which we might call Big Exemplar Based texture synthesis. Unlike exemplar-based methods that use small texture samples the goal is to synthesize large textures. We shall explore this problem with the tools at reach, namely the best methods from the extensive literature including our contributions of the previous chapters, and will not hesitate to combine them. To anticipate our conclusions, none of the existing methods manage by itself to satisfactorily emulate "big textures". Yet, we'll find that a clever combination of these methods yields interesting-if not perfect-results. Interestingly, the existing methods provide very good results on small crops (see Figure 1.9) of "big textures" images. But we discovered with

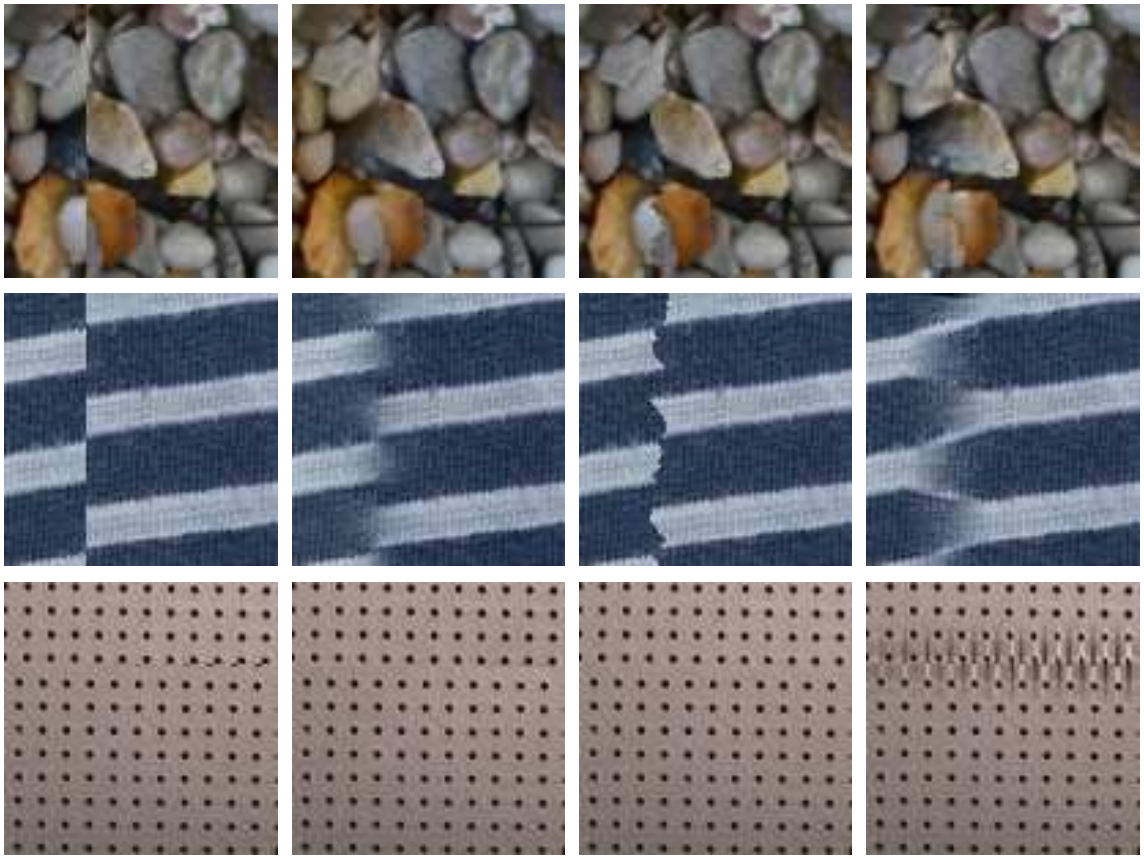


Figure 1.7 – Some challenging examples blending patches which do not match. Comparison with two other stitching methods. From left to right: input, Poisson editing [47] and the midway blending.

some dismay and shall illustrate on many examples that "big textures" are very different from academic texture crops. They show drastic local textural variations, even if their overall effect still is the one of a single texture (see the crops of a plank in Figure 1.9). This study is therefore a good test to put on trial all existing exemplar-based texture synthesis methods.

We began by trying some of the exemplar based texture synthesis approaches (Efros and Freeman [15], Portilla and Simoncelli [50], Galerne et al. [21] and the multiscale locally Gaussian method presented in Chapter 3) on different types of natural large texture images such as: wood, stone and metal. We observed that the synthesis results for the wood images were not satisfying. However for local crops of these textures where no strong global organization were evident the results obtained are impressive. One concludes obviously that, in spite of their perceptual unity, large textured images are more than *a texture* in the usual academic sense, as can be seen for the wood texture example shown in Figure 1.8 and its crops in Figure 1.9. We follow by showing results of a combination of the considered texture synthesis methods, namely:

- The Efros and Freeman method [15] combined to the locally Gaussian method (LG+EF)
- The Portilla and Simoncelli method [50] combined to the locally Gaussian method (LG+PS)

Combining the methods yields better visual results in particular for the combination PS + LG. Still this combination will not make it for wood textures, for example. We then show other emulation

Figure 1.8 – Wood texture 1. The original input provided was of size 7087×23622 pixels. For the experiments the image was reduced by a zoom out of factor 16.

Figure 1.9 – Four crops of different parts of Figure 1.8 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

attempts, like for example the application of geometric deformations to the inputs aiming at changing the aspect of the salient structures. An often successful trick is randomly switching sub-parts of the input to partition the salient structures and reorganize them in a way that visually mitigates the repeated patterns. As a last experiment we tried to vary the low frequency information using the method proposed by Galerne et al. in [21] while leaving unchanged the high frequencies with the same purpose. In all three emulation attempts we observed interesting behaviours for some of the textures. In Figure 1.10 we show three examples.

Chapter 5 is organized as follows. Section 5.2 describes the large natural texture images we worked on. In section 5.3 a brief analysis of some of the state of the art exemplar-based texture synthesis methods is provided listing the pros and cons of each of them. In section 5.4 a first set of experiments are showed on small crops of the big textures. All of the exemplar-based methods of section 5.3 are tested on these images. They are tested separately and then some of them are combined. The results on these images are very satisfying. The same experiments are performed on the big texture themselves to try to emulate their structures at low resolution. The results are not very satisfying since the typical drawbacks of exemplar based methods arise (verbatim copies and garbage growing). This brings us to test other alternatives presented in Sections 5.5, 5.6 and 5.7. We try the application of one and two dimensional anamorphoses to the inputs, a random combination of slices of the input texture and the application of the method [21] to randomize the low frequencies of the input.

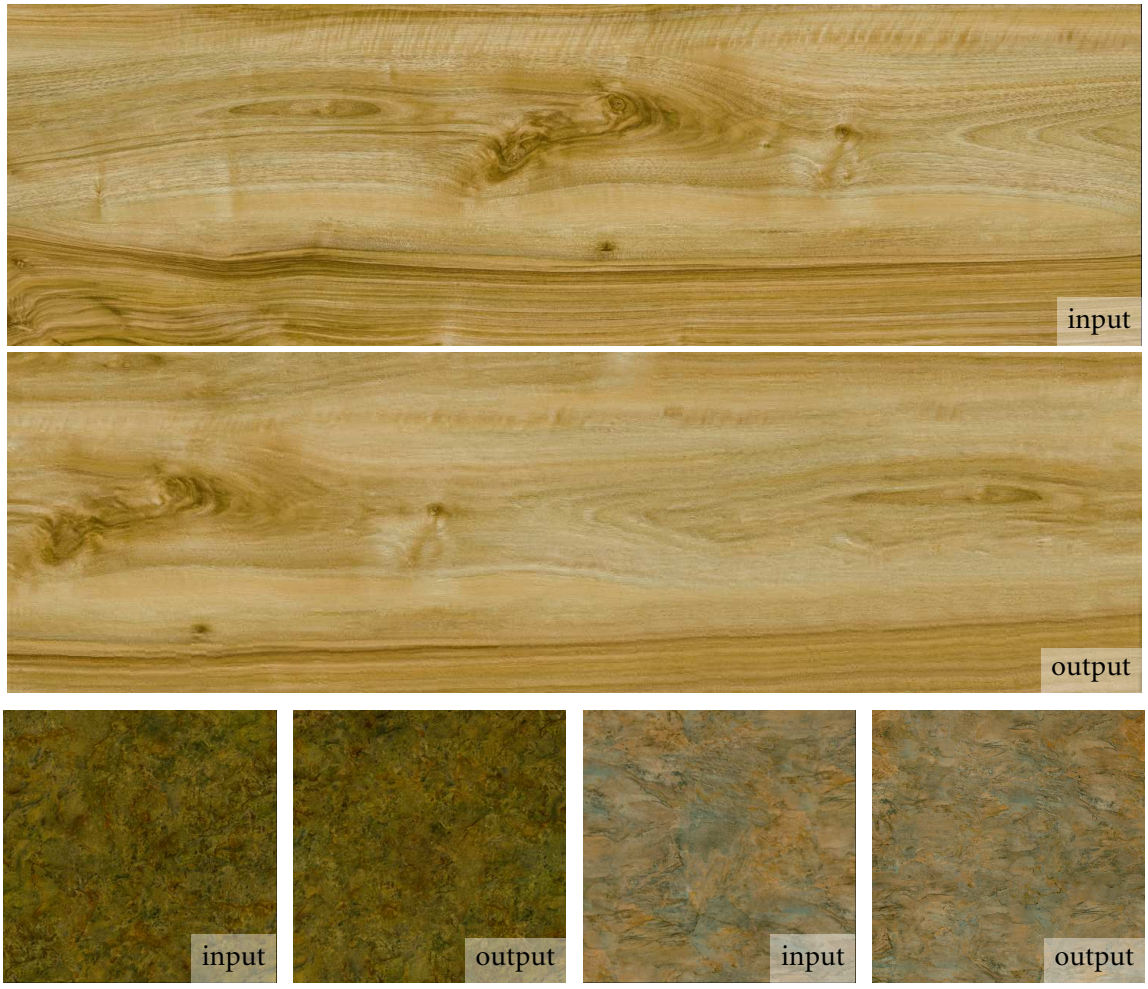


Figure 1.10 – Synthesis results of different natural large textures.

List of Publications

- L. Raad, A. Desolneux and J.-M. Morel. *Locally Gaussian Exemplar Based Texture Synthesis*. In IEEE International Conference on Image Processing (ICIP), pages 4667 - 4671, 2014. IEEE.
- L. Raad, A. Desolneux and J.-M. Morel. *Conditional Gaussian Models for Texture Synthesis*. In International Conference on Scale Space and Variational Methods in Computer Vision (SSVM), pages 474 - 485, 2015. Springer International Publishing.
- L. Raad, A. Desolneux and J.-M. Morel. *Multiscale Exemplar Based Texture Synthesis by Locally Gaussian Models*. In IberoAmerican Congress on Pattern Recognition (CIARP),pages 435 - 443, 2015. Springer Iternational Publishing.
- L. Raad, A. Desolneux and J.-M. Morel. *Conditional and multiscale locally Gaussian models for texture synthesis*. In Journal of Mathematical Imaging and Vision (JMIV), pages 1 - 20, 2016.
- L. Raad and B. Galerne. *Efros and Freeman Image Quilting Algorithm for Texture Synthesis*. Submitted to Image Processing On Line (IPOL).

- L. Raad, A. Desolneux and J.-M. Morel. *Midway blending as a new technique for patch quilting.* To be submitted.
- L. Raad, A. Desolneux and J.-M. Morel. *Can exemplar based texture synthesis models emulate large textures?* To be submitted.

2 Locally Gaussian models

The main approaches to texture modeling are the statistical psychophysically inspired model and the patch-based model. In the first model the texture is characterized by a sophisticated statistical signature. The associated sampling algorithm estimates this signature from the example and produces a genuinely different texture. This texture nevertheless often loses accuracy. The second model boils down to a clever copy-paste procedure, which stitches verbatim copies of large regions of the example. In this chapter we propose to involve a locally Gaussian texture model in the patch space. It permits to synthesize textures that are everywhere different from the original but with better quality than the purely statistical methods.

2.1 Introduction

Exemplar-based texture synthesis aims at generating new texture images inspired from a texture sample. Texture synthesis algorithms can be roughly divided in two categories: the statistics-based methods [26, 65, 11, 25, 50] and the non parametric patch based methods [14, 63, 2, 27, 42, 15, 37, 36].

Statistics-based methods estimate a set of the sample's statistics, which are then enforced in the synthesized texture. The statistical characterization of texture images was initiated by Julesz. Julesz was the first to point out that texture images could be reliably organized, according to their N-th order statistics, into groups of textures that are preattentively indistinguishable by humans [28]. In [31], Julesz demonstrated that many texture pairs sharing the same second-order statistics would not be discerned by human preattentive vision. This hypothesis constitutes the first Julesz axiom for texture perception. One consequence of this axiom is that two textures sharing the same Fourier modulus but with different phase should be perceptually equivalent. This motivates a class of algorithms (random phase methods) aiming at creating textures with a given second-order statistic. An example of such algorithms is [60] and its extension [21] in which a texture is generated by randomizing the Fourier phase while maintaining the Fourier modulus. The random phase method in [21] correctly synthesizes micro-texture images which adapt well to a Gaussian distribution, but it fails for more structured ones, as can be experimented in the executable paper [20]. It has then been showed that textures with the same second and even third order statistics were visually different [32, 8]. In [30, 29] Julesz proposed a second theory to explain texture preattentive discrimination theory by introducing the notion of textons (local conspicuous features). This new theory states that only the first order statistics of these textons are relevant for texture perception: images having the same texton densities (first ordre statistic) should not be discriminated. Relative to the texton theory, one axiom is that texture perception is invariant to random shifts of the textons [30] leading to the stochastic dead leaves models [43, 55, 4].

Heeger and Bergen [26] extended the Julesz approach to multiscale statistics. They characterized a texture sample by the histograms of its wavelet coefficients. By enforcing the same histograms on a white noise image, they obtained an exemplar based synthesis method. Yet this method only measures marginal statistics. It misses important correlations between pixels across scales and orientations. See the online execution of this method [6] where some success but many failures are evident, like for [20]. Within a similar range of results De Bonet [11] randomizes the initial texture image and preserves only a few statistics: the dependencies across scales of a multiresolution filter banch response. In [25] the authors propose to synthesize a texture by taking randomly patches from the sample texture and placing them randomly in the output texture image. A blending step is applied across the overlapping blocks to avoid edge artifacts. The results achieved are similar to [26, 11]. Other methods are also based on statistics of wavelet coefficients or more involved multiscale image representations [50, 48, 53]. The Heeger-Bergen method was extended by Portilla and Simoncelli [50] who proposed to evaluate on the sample some 700 cross-correlations, autocorrelations and statistical moments of the wavelet coefficients. Enforcing the same statistics on synthetic images achieves striking results for a very wide range of texture examples. This method, which represents the state of the art for psychophysically and statistically founded algorithms is nevertheless computationally heavy, and its convergence is not guaranteed. Its results, though generally indiscernible from the original samples in a pre-attentive examination, often present blur and phantoms.

Patch-based methods constitute a totally different category of texture synthesis algorithms. The initial Efros and Leung [14] method is based on Shannon's Markov random field model for the English language. In analogy with Shannon's algorithm for synthesizing sentences, the texture is constructed pixelwise. For each new pixel in the reconstructed image, a patch centered in the pixel is compared to all the patches of the input sample. The patches in the sample that are closer help predict the pixel value in the synthetic image. Several optimizations have been proposed to accelerate this algorithm. Among them Wei and Levoy [63], who managed to fix the shape and size of the learning patch and Ashikhmin [2] who proposed to extend existing patches whenever possible instead of searching in the entire sample texture. Yet, as already pointed out in the original paper [14], an iterative procedure may fail by producing "garbage" when the neigborhood's size is too small. On the other hand it can lead to a trivial verbatim reproduction of big pieces of the sample when the neighborhood is too large. This can be experimented in [1]. Several extensions of [14] have been proposed that manage to accelerate the procedure and resolve the "garbage" problem by stitching entire patches instead of pixels. In [42] the authors propose to synthesize the image by quilting together patches that were taken from the input image among those who best match the patch under construction. A blending step was also added to overcome some edge artifacts. Efros and Freeman [15] proposed an extension of the latter introducing for the quilting method a blending algorithm that computes a path with minimal contrast across overlapping patches, thus mitigating the transition effect from patch to patch. Kwatra et al. in [37] extend [15] by using a graphcut algorithm to define the edges of the patch to quilt in the synthesis image. Another extension of [14] is proposed by Kwatra et al. in [36] where to synthesize a texture image they improve the quality of the synthesis image sequentially by minimizing a patch-based energy function. For these methods the visual results are strikingly good, but they still retain the risk of verbatim copying large parts of the input sample. For practical applications this may result in the appearance of repeated patterns in the synthesized image.

An ideal exemplar-based texture synthesis algorithm should create a new texture image that is perceptually equivalent to the input sample, but this image should be as random as possible while maintaining the same statistics as the input. To the best of our knowledge, none of the methods

proposed in the literature are able to combine these advantages. On the one hand, statistics-based methods fail, in general, to be perceptually equivalent to the input, either because the statistics are not well enforced or because these statistics are not complete enough. On the other hand, patch-based methods are too close to a mere copy-paste. This effect can be observed in the synthesis maps of Figure 2.6.

In this chapter, we propose an algorithm that lies between statistics-based and patch-based algorithms. We gave up choosing a set of statistics from the input sample to enforce in the synthesized image. Instead, we chose to model each texture patch by a Gaussian distribution learned from its similar patches. Inspired by [15], we maintain the idea of searching for patches to stitch together in the original sample. However, instead of using the exact patch taken in the input texture, we sample the stitched patch from its Gaussian model. Locally Gaussian patch model have been proved useful in image denoising [38]. Our approach permits to maintain the coherence between patches with respect to the input sample, while creating new patches that do not exist in the sample texture but are still perceptually equivalent to it. In this way, we manage to combine the positive aspects of statistics-based and patch-based methods, while overcoming some of their drawbacks.

This chapter is structured as follows. In Section 2.2, we give an estimation method of the Gaussian model for each patch. An analysis of the model's variance is provided. In Section 2.3, the locally Gaussian synthesis algorithm is described. Section 2.4 compares the results with state of the art texture-synthesis methods. The concluding section 2.5 also states the limitations of the method.

u	$\Omega \to \mathbb{R}$: input texture image defined on the discrete domain $\Omega = I_M \times I_N$ of size $M \times N$ where I_c denotes the discrete interval $[0, \ldots, c-1]$
w	$\Omega_r \to \mathbb{R}$: output texture image defined on the discrete domain $\Omega_r = I_{rM} \times I_{rN}$ of size $rM \times rN$
d	is equal to 1 if u is grayscale and to 3 for color images
r	ratio (size of output image w)/(size of input image u)
n	side patch size
m	number of nearest neighbours used to learn the parameters of the Gaussian distribution
o	overlap size
$p_u^{(x,y)}$	square patch of size $n \times n$ from an image u of size $M \times N$ at position (x, y), $p_u^{(x,y)} = \{u\left((x,y)+(i,j)\right),\ (i,j) \in [0, \ldots, n-1]^2\}$, $(x,y) \in \mathcal{V}_u$ where $\mathcal{V}_u = I_{M-n+1} \times I_{N-n+1}$ denotes the discrete domain of the valid patches in u. The patch $p_u^{(x,y)}$ will be considered as a column vector of size $dn^2 \times 1$
$\mathbf{I}_m$	denotes the $m \times m$ identity matrix

Table 2.1 – Summary of the notations used in this chapter.

2.2 The patch Gaussian model

In this section we consider a first and simple model of the texture sample. We denote $u : \Omega \to \mathbb{R}$ the input texture image defined on the discrete domain $\Omega = I_M \times I_N$ of size $M \times N$ and $p_u^{(x,y)}$ a patch from u defined as a square block of size $n \times n$ among all possible overlapping patches in u. Each patch is defined by the position of its left top corner $(x, y) \in \Omega$. We assume that the

underlying distribution of a given patch $p_u^{(x,y)}$ in u follows a multivariate Gaussian distribution of mean $\mu^{(x,y)}$ and covariance matrix $\Sigma^{(x,y)}$. Hence to generate a new texture image $w : \Omega \to \mathbb{R}$ according to this model all the patches of u are replaced by samples of these multivariate Gaussian distributions.

Given the image u and fixing an overlap size $o \in [0,1]$ we will consider only the patches from u that are taken in a raster-scan order (left to right and top to bottom) and whose centers are separated by $(1-o)n$ pixels. We call them the *patches under construction*. They can overlap in three possible way: horizontally, vertically and forming an L-shape overlap (see Figure 2.3). Each one of these patches will be replaced by a new patch $\tilde{p}_u^{(x,y)}$ sampled from its corresponding multivariate Gaussian ditribution of parameters $\mu^{(x,y)}$ and $\Sigma^{(x,y)}$.

The parameters $\mu^{(x,y)}$ and $\Sigma^{(x,y)}$ of the Gaussian distribution of the patch $p_u^{(x,y)}$ are estimated from the set of nearest patches $\mathcal{U} = \{p_u^{(x_i,y_i)}, i = 1,\dots,m\}$. Here $p_u^{(x_i,y_i)}$ are the m nearest patches to $p_u^{(x,y)}$ in u according to the L^2 distance. The empirical statistics $\mu^{(x,y)}$ and $\Sigma^{(x,y)}$ are then defined as

$$\begin{aligned} \mu^{(x,y)} &= \tfrac{1}{m}\textstyle\sum_{i=1}^m p_u^{(x_i,y_i)} \\ \Sigma^{(x,y)} &= \frac{\sum_{i=1}^m \sum_{j=1}^m \left(p_u^{(x_i,y_i)} - \mu^{(x,y)}\right)\left(p_u^{(x_j,y_j)} - \mu^{(x,y)}\right)^t}{(m-1)}. \end{aligned} \tag{2.1}$$

To simplify the notation, we denote by P the matrix whose columns are the normalized patches in vector form $(p_u^{(x_i,y_i)} - \mu^{(x,y)}),\ i = 1,\dots,m$. Then we can reformulate the covariance matrix $\Sigma_{(x,y)}$

$$\Sigma^{(x,y)} = \frac{1}{(m-1)} P P^t.$$

Sampling step Sampling a vector $\tilde{p}_u^{(x,y)} \sim \mathcal{N}(\mu^{(x,y)}, \Sigma^{(x,y)})$ boils down to a linear combination of the m nearest patches $p_u^{(x_i,y_i)}$ such as

$$\tilde{p}_u^{(x,y)} = \frac{1}{\sqrt{m-1}} \sum_{i=1}^m a_i (p_u^{(x_i,y_i)} - \mu^{(x,y)}) + \mu^{(x,y)} = \frac{1}{\sqrt{m-1}} PA + \mu^{(x,y)} \tag{2.2}$$

where $a_i \sim \mathcal{N}(0,1),\ i = 1,\dots,m$ are independent random variables and A is a column vector of size $m \times 1$ whose elements are the standard normal independent random variables a_i. We denote by $\mathcal{N}(\mu, \Sigma)$ the multivariate normal distribution of mean μ and covariance matrix Σ. Indeed the sampled patch $\tilde{p}_u^{(x,y)}$ follows the mutltivariate normal distribution $\mathcal{N}(\mu^{(x,y)}, \Sigma^{(x,y)})$ where

$$\mathbb{E}(p_u^{(x,y)}) = \frac{1}{\sqrt{m-1}} \sum_{i=1}^m \mathbb{E}(a_i)(p_u^{(x_i,y_i)} - \mu^{(x,y)}) + \mu^{(x,y)} = \mu^{(x,y)}$$

and

$$\mathbb{E}((\tilde{p}_u^{(x,y)} - \mu^{(x,y)})(\tilde{p}_u^{(x,y)} - \mu^{(x,y)})^t) =$$
$$\frac{1}{(m-1)} \sum_{i=1}^{m} \sum_{j=1}^{m} \mathbb{E}(a_i a_j)(p_u^{(x_i,y_i)} - \mu^{(x,y)})(p_u^{(x_j,y_j)} - \mu^{(x,y)})^t =$$
$$\frac{1}{m-1} \sum_{i=1}^{m} (p_u^{(x_i,y_i)} - \mu^{(x,y)})(p_u^{(x_i,y_i)} - \mu^{(x,y)})^t = \Sigma^{(x,y)}.$$

Blending step Once a patch $\tilde{p}_u^{(x,y)}$ is sampled it is placed in image w at position (x, y) overlapping the part which has been synthesized previously. Across this overlap area the pixel values are blended using a convex combination. That is, at each position (i, j) in the overlap region two values are possible: the one from the previous synthesis g_{old} and the one from the actual synthesis g_{new}. We denote g the value assigned at position (i, j) defined as

$$g = (1 - \gamma(i, j))g_{\text{old}} + \gamma(i, j)g_{\text{new}}$$

where $\gamma = (1 - \alpha)(1 - \beta)$ and

$$\alpha(x, y) = \begin{cases} 1 - \frac{x-1}{on-1} & \text{if } 1 \leq x \leq on \\ 0 & \text{if } x > on \end{cases}$$

and

$$\beta = \begin{cases} 1 - \frac{y-1}{on-1} & \text{if } 1 \leq y \leq on \\ 0 & \text{if } y > on \end{cases} .$$

The procedure is summarized in Algorithm 1.

Algorithm 1: Naive locally Gaussian texture synthesis

Input: input texture sample u, side patch size n, number of nearest neighbours m, overlap size o
Output: reconstructed texture w

```
for x = 1 : n(1 − o) : M do
    for y = 1 : n(1 − o) : N do
        Compute U = {p_u^(x_i,y_i), i = 1, ..., m} the set of m closest patches to p_u^(x,y)
        Estimate μ^(x,y) ← (1/m) Σ_{i=1}^m p_u^(x_i,y_i)
        p̃_u^(x,y) ← (1/√(m−1)) (P − μ^(x,y)) A + μ^(x,y),  A ∼ N(0⃗, I_m) {sampling step}
        Blend p̃_u^(x,y) in the image w at position (x, y) {blending step}
    end for
end for
return w
```

The size n of the patches and the size of the neighborhood m are user-specified values. The results of this first model are shown in Figure 2.1. Observe that replacing the patches of the input texture by simulated ones (with the corresponding Gaussian models) achieves a faithful random variant of the original image, for reasonable values of n and m. For large values of m the model

no longer represents the patch faithfully because the set $\mathcal{U}$ will contain outliers. Nevertheless, reasonable values for n and m ensure a correct reconstruction when replacing patches by others simulated from multivariate Gaussian distributions. We observed that when using 30×30 patches, it is reasonable to consider a neighborhood of 10 to 20 patches. Even though the generated texture is a random and new image perceptually equivalent to the input one, both are visually too similar. We would like to observe more randomness in the generated texture. A second inconvenient of this model is that the generated texture has necessarily the same size as the input image. For these reasons we propose a second model which does not fix the patches and has no constraint on the size of the generated texture. This is explained in the next section.

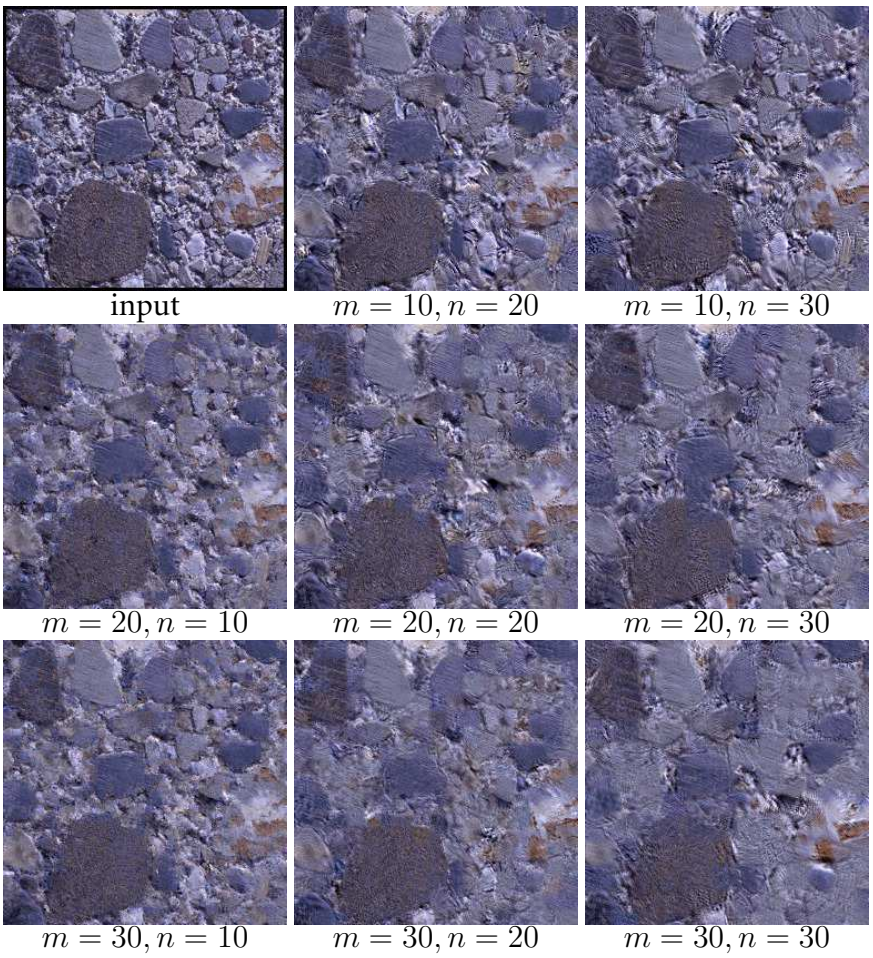


Figure 2.1 – Visual evaluation of the validity of a Gaussian patch model. This experiment performs a Gaussian substitution for each patch of the left top corner texture image (Algorithm 1). From left to right the patch size is $n = 10, 20, 30$. From top to bottom the number of nearest neighbours is $m = 10, 20, 30$.

The model's variance It is important to analyze how the variance of the Gaussian distribution varies with the patch size n and the number of nearest neighbours m used to estimate these distributions. We expect to have a variance that is not equal to zero since this would correspond to taking the best match in u and thus having no innovation at all. On the other hand a high variance would introduce too much variability and therefore end up with a blurry reconstruction of the texture. The ranges for m and n yielding a correct estimate for the variance cannot be determined *a priori*

and strongly depend on the texture sample. To measure the variability of the Gaussian model we observe the mean standard deviation per pixel of a given patch $p_u^{(x,y)}$ in the texture image which can be estimated by

$$\bar{\sigma} = \frac{1}{dn^2} \sum_{i=1}^{dn^2} \sqrt{\Sigma^{(x,y)}(i,i)} = \frac{1}{dn^2} \sum_{i=1}^{dn^2} \sigma_i$$

where d is the number of channels ($d = 1$ for grayscale images and $d = 3$ for color images).

In Figure 2.2, two texture examples are presented. As expected for a fixed patch size n the mean standard deviation per pixel $\bar{\sigma}$ increases when using more patches to estimate the Gaussian distribution. For a fixed value of m the same behaviour is observed when increasing n. The first example, a micro texture, shows that the variance of the patch is not negligible even for $m = 5$. Gaussian models are well suited for this kind of texture. The second example, a periodic texture, shows that the patch mean standard deviation varies between 8 and 18 for values of $m \in \{5, 10, 30, 50\}$. These are reasonable values confirming that effectively the patches simulated have an acceptable degree of innovation.

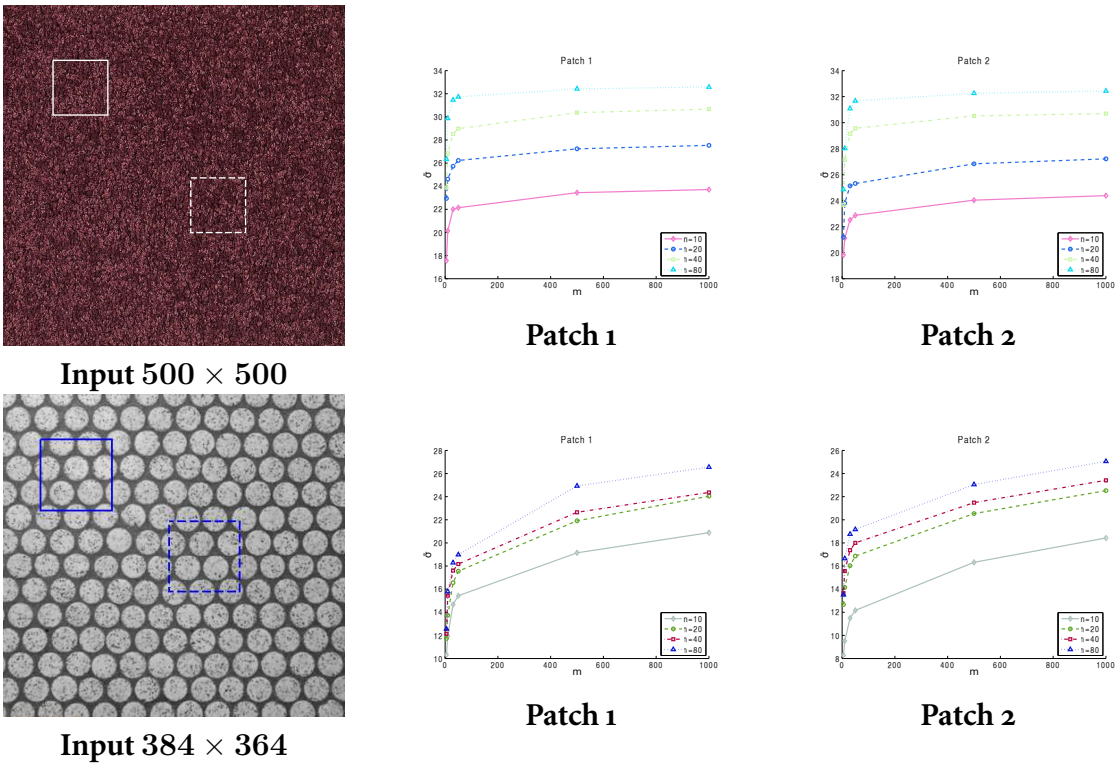


Figure 2.2 – Behaviour of the mean standard deviation of the patches as functions of the patch size n and the neighbourhood size m. For each texture example two patches have been selected (left column). For each of them the mean standard deviation is computed for several values of $n \in \{10, 20, 40, 80\}$ and $m \in \{5, 10, 30, 50, 500, 1000\}$. In the input image patch 1 is represented by a solid line and patch 2 by a dashed line.

2.3 The locally Gaussian texture synthesis algorithm

As mentioned in the previous section simply replacing the texture patches by Gaussian samples is not enough for synthesizing a texture. This model yields texture images that are visually too similar to the input sample and of fixed size. We aim at generating images that are perceptually equivalent to the input yet not visually identical. Inspired by the work of Efros and Freeman [15] we propose an iterative algorithm that models the self-similarities of the input sample as multivariate Gaussian distributions.

The proposed algorithm consists in generating a new texture image w patch by patch in a raster-scan order given the input sample u. At each iteration the goal is to generate a new patch $p_w^{(x,y)}$ that is partially defined on a region called the overlap region (see Figure 1.1). The known part of the patch will define the set of patches from u that will infer its Gaussian model. The generated patch $p_w^{(x,y)}$ will be a sample of this model. The overlap size o is fixed and for all the presented results o is half the size n of the patch.

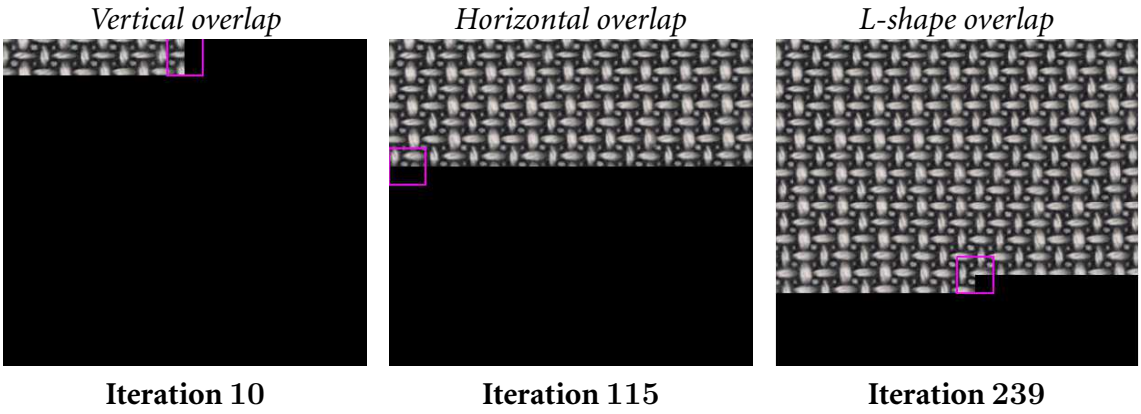


Figure 2.3 – Three different iterations of the synthesis process are shown. At each iteration a patch is being synthesized. This patch is represented by the pink square in the three iterations shown. From left to right the three overlap cases are represented: vertical, horizontal and L-shape.

We initialize the new texture image w placing a *seed patch* in its top left corner. For this we pick a random patch from the texture sample u, estimate its underlying Gaussian distribution and sample a new patch from it (the *seed patch*). In continuation, each synthesized patch is sampled from a Gaussian model learnt from the m patches with their left half most similar to the right half of the last synthesized texture patch. This new patch is then quilted on the current texture.

Patch stitching is an important step when adding the new patches at each iteration. In this chapter we propose to follow the procedure presented in [15]. In Chapters 3 and 4 we suggest other stitching alternatives. The method proposed in [15] assigns new edges to the patches on the overlap area in order to quilt them along the path on which they best match. These edges are selected as curves minimizing the error between the new patch and part of the synthesized image across the overlap region. For a detailed description of this step please refer to Annex A.2.3. The synthesis procedure is summarized in Algorithm 2.

The proposed method yields a new texture image perceptually similar to the original sample and whose patches have been randomized. The result obtained is visually similar to the input sample, i.e. we could consider that the input texture and the new texture are two different sub images of one same bigger texture. Initializing the input with the *seed patch* and synthesizing the

Gaussian patches in a raster-scan order using only their overlap to find the nearest neighbours from the input yields this variability regarding the first model we proposed.

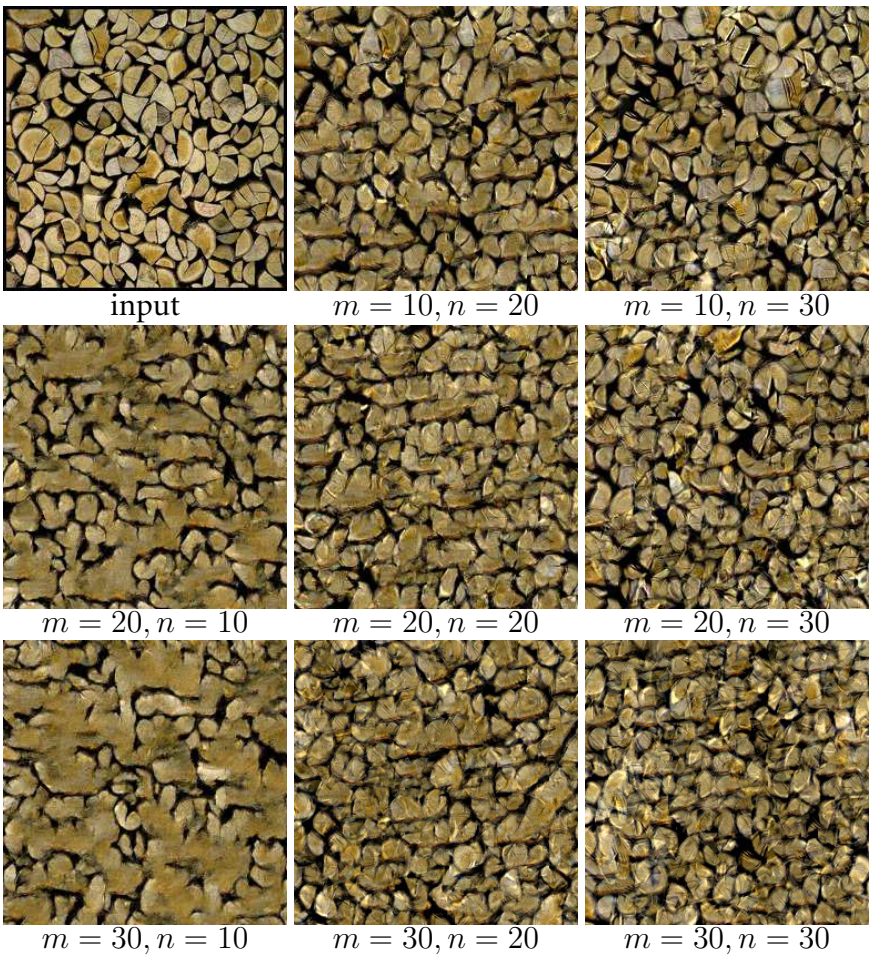


Figure 2.4 – Texture synthesis result for the left top corner texture image. We show the results obtained for different values of m (the number of similar patches) and n (the side patch size). From left to right $n = 10, 20, 0$. From top to bottom, the number of nearest neighbours is $m = 10, 20, 30$. All the results are obtained for an overlap of a half patch size $o = n/2$.

A discussion on the underlying mathematical model The underlying random field of a given input texture is assumed to be a Gaussian random field. Indeed, as illustrated with the experiment in Figure 2.1, a patch in the input is assumed to be a multivariate Gaussian vector. Furthermore, we assume the existence of a set of other patches following the same probability distribution, from which the parameters (mean and covariance matrix) are estimated. Thus the resulting reconstructed image is a sample of a Gaussian random field. Nevertheless, the same cannot be stated for the underlying random field of the synthesized textures when using Algorithm 2. These generated textures are not samples of a Gaussian random field. What can be affirmed is that the conditional distribution of the patches are Gaussian, meaning that if $p_w^0, p_w^1, p_w^2 \dots$ are the overlapping patches of the image taken in a raster scan order, then the conditional probability $\mathbb{P}(p_w^n | p_w^{n-1})$ is a Gaussian multivariate distribution. However, the mean and the covariance matrix

Algorithm 2: Locally Gaussian texture synthesis (LG)

Input: input texture sample u, side patch size n, overlap size o, number of nearest neighbours m, ratio (output size)/(input size) r

Output: synthesized texture w

1: Initialize w placing a seed patch in its top-left corner $(x = 1, y = 1)$. The image w is of size $rM \times rN$, where $M \times N$ is the size of u.
2: **for** $x = 1 : n(1-o) : rM$ **do**
3: **for** $y = 1 : n(1-o) : rN$ **do**
4: **if** $(x > 1$ **or** $y > 1)$ **then**
5: Find the m nearest patches $p_u^{(x_i,y_i)},\ i = 1, \ldots, m$ in the input sample u that best agree with the patch under construction $p_w^{(x,y)}$ along the overlap area
6: Estimate $\mu^{(x,y)} \leftarrow \frac{1}{m}\sum_{i=1}^{m} p_u^{(x_i,y_i)}$
7: Sample $\tilde{p}_w^{(x,y)} \leftarrow \left(\frac{1}{\sqrt{m-1}}PA + \mu\right) \sim \mathcal{N}\left(\mu^{(x,y)}, \Sigma^{(x,y)}\right)$ where $A \sim \mathcal{N}(\vec{0}, \mathbf{I}_m)$
8: Quilt $\tilde{p}_w^{(x,y)}$ in w at position (x, y) {see Annex A.2.3}
9: **end if**
10: **end for**
11: **end for**
12: **return** w

of this Gaussian distribution are computed from patches similar to p_w^{n-1} in the input sample. Therefore the joint probability distribution of the patches is not necessarily Gaussian.

Thus, while the algorithms are effective, they do not imply so far the existence of a complete texture model. Indeed, the input sample is characterized by a Gaussian distribution for each patch but also by *spreads*, defined as the spatial distribution of the patches belonging to the same Gaussian model. For some periodic textures the spread is deterministic (like in a chessboard for instance) but for general textures the spread itself is random. To model the whole texture as a random field a stochastic model for the spreads would therefore be necessary. For example, if the spread of the input sample is given and the patches of the generated texture are samples of conditional Gaussian distributions respecting exactly the same spread as the input's, then the underlying random field of the whole synthesized texture is Gaussian (as in the example of Figure 2.1). Nevertheless, to synthesize textures this is not very interesting in terms of variability among the different sampled textures. Another meaningful example where the model is complete is the case of periodic textures where the spread is deterministic. For a fixed seed patch (the one that initializes the generated texture) the underlying random field of the generated textures is a periodic Gaussian random field. But this is no longer the case when the seed patch is random.

To summarize, we cannot claim that the proposed synthesis algorithm corresponds to a complete texture model, because it would require a stochastic model for the *spread*, namely the spatial interaction between patches obeying the same Gaussian model. This is left for future investigation, perhaps in the spirit of [41].

2.4 Experiments

A first set of synthesis results can be seen in Figure 2.4 to illustrate the influence of the parameters n and m for a given texture example. We vary the values of the number of nearest neighbours m used to estimate the parameters of the Gaussian distributions and of the side patch size n. The first observation is that the preferable patch size has a sample-dependent lower bound. If the patch size is too small the algorithm does not capture the variability of the sample. This is the case for all non parametric patch-based methods. Yet, the larger the patch size, the fewer the neighbours that we can use to build a faithful model for the patches. The second observation relates to the size of the overlap between patches. If the proportion of the overlap area with respect to the patch size is too small, the obtained model may no longer be a correct representation of the patch to simulate. Indeed, the patches are compared on the overlap area but the model is learnt on the whole patch. We observed that for an overlap of half the patch size the model is faithful enough to achieve good results. The number of nearest neighbours used to infer the Gaussian models is strongly dependent of the input texture. For periodic or pseudo-periodic textures this is not an issue, as long as the patch size is not too large, since there are many similar patches to infer a good patch model. Whenever strong and isolated configurations are visible then this becomes tricky and the value of m is limited. Moreover a unique value of m may not be adapted in particular if the texture is not stationary. Another limitation on the value of m is indeed the size of the input image.

Figure 2.6 compares several state of the art texture synthesis methods. In general we can observe that for statistics-based methods (three first columns) the quality of our algorithm's visual results is considerably superior. However, when the size of the patch n is too large compared to the texture's structure (last row in Figure 2.6) blurring artifacts can appear in our results. On the other hand, like for the Efros-Freeman algorithm [15] our method is able to preserve the strong structures available in the input sample. We obtain similar visual results while creating an image whose patches are new configurations. Of course one can notice that for isolated configurations even if the generated patches are pixelwise different to the input ones, visually they are very similar and verbatim copy effects are still visible. We believe that being able to work with smaller patches reduces significantly this drawback.

In Figure 2.7 we compare the capacity of innovation of our method to the work in [15]. To illustrate this we represent the synthesis results using *position and synthesis maps* as can be seen in Figure 2.5 which shows the input sample u, the associated position map (Pmap), the synthesized texture w and the associated synthesis map (Smap). Each pixel position $\mathbf{x}$ in the sample texture u is associated to a different color from a continuous colormap. The resulting image is the position map. The synthesis map associated to the synthesized texture w is obtained by mapping each of its positions $\mathbf{x}'$ to the color value of the corresponding position $\mathbf{x}$ of u. This position $\mathbf{x}$ corresponds to the central pixel of the nearest patch in u to the patch centered in $\mathbf{x}'$ in w. The synthesis map permits to identify the tendency of an algorithm to generate verbatim copies and to visualize from which regions of the input texture are sampled the patches. To compute these synthesis maps we used the PatchMatch algorithm [3] an efficient method to approximate optimal correspondences. Patch match is an algorithm that finds approximate nearest neighbours matches between image patches. That is, given two input images A and B the patch match algorithm returns an offset map. This offset map indicates where, for each pixel $\mathbf{x}'_i$ in B considering the patch centered in $\mathbf{x}'_i$, is approximatively the nearest patch in A centered in $\mathbf{x}_j$. We used the implementation of the patch match algorithm provided in [16].

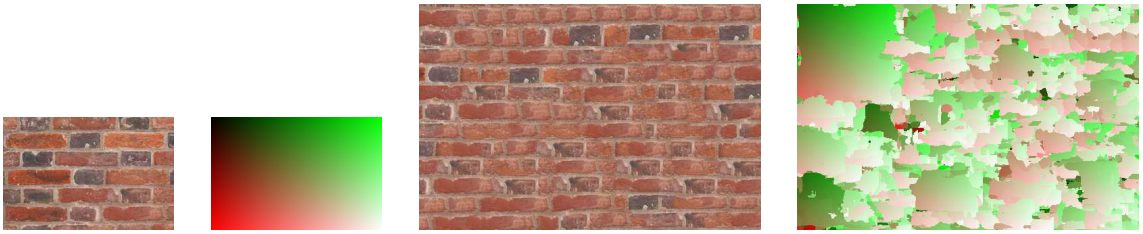


Figure 2.5 – *Results representation*. From left to right: texture sample, position map, synthesized image and synthesis map. The synthesis map shows for each synthesized patch its initial position in the texture sample. It allows then to identify exactly the verbatim copy regions (they correspond to continuous color areas of the map).

2.5 Conclusion

In this chapter we presented a synthesis algorithm that generates a new texture image by stitching together patches sampled from multivariate Gaussian distributions in the input sample patch space. For each patch under construction in the output image the underlying Gaussian model is inferred from the set of similar patches (to the patch under construction) from the input sample. Hence the algorithm synthesizes a texture that is perceptually equivalent to the sample texture yet not composed of patches existing in the input texture. This method reduces some of the drawbacks of the statistics-based and the patch-based methods. The stitching procedure is a bit complex and will be replaced in Chapter 3. Like the Efros-Leung or the Efros-Freeman methods, the algorithm remains dependent on the choice of n and m, that may have to be adjusted for each texture sample. In our opinion the texture samples used in the literature are too small, particularly for macro-textures like the ones presented here and in most papers. Thus, our local Gaussian model is forced to use only 10 to 20 degrees of freedom because only some 10 to 20 patches are similar enough. This estimate should improve with larger texture samples. Our algorithm has low complexity, compared for instance with classic patch-based denoising algorithms [39], [9]. Indeed, the patch covariance matrix needs not to be computed or inverted. The gaussian samples are simply obtained by linearly combining the nearest neighbours patches. Another alternative to reduce the dependency of the method to the patch size is to work in a multiscale approach. This extension is presented in the next chapter. Unlike the statistics-based algorithms, but like the other patch-based methods, our algorithm is not forced to respect the global statistics of the texture sample. This can be observed in the second row of Figure 2.6 where our result correctly reproduces an image of petals but fails to insert a correct proportion of the leaves in the synthetic image. In the next chapter we also deal with this issue.

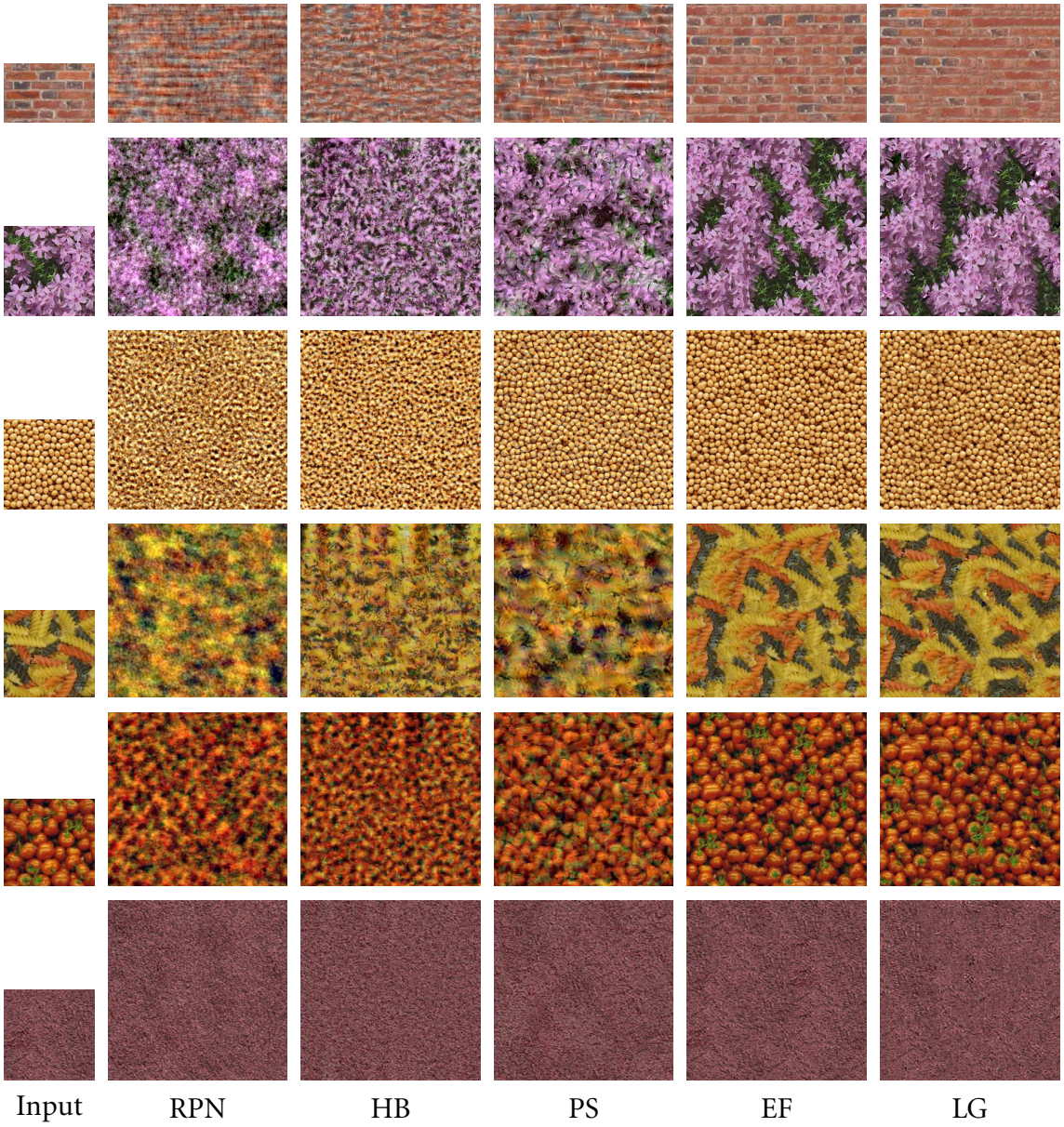


Figure 2.6 – Comparison with various other texture synthesis methods. From left to right: texture sample, synthesis results of [20], [6], [50], [15] and our algorithm the locally Gaussian synthesis method (LG). For all the examples $m = 10$, $n = 30$ except for the third row where $n = 10$.

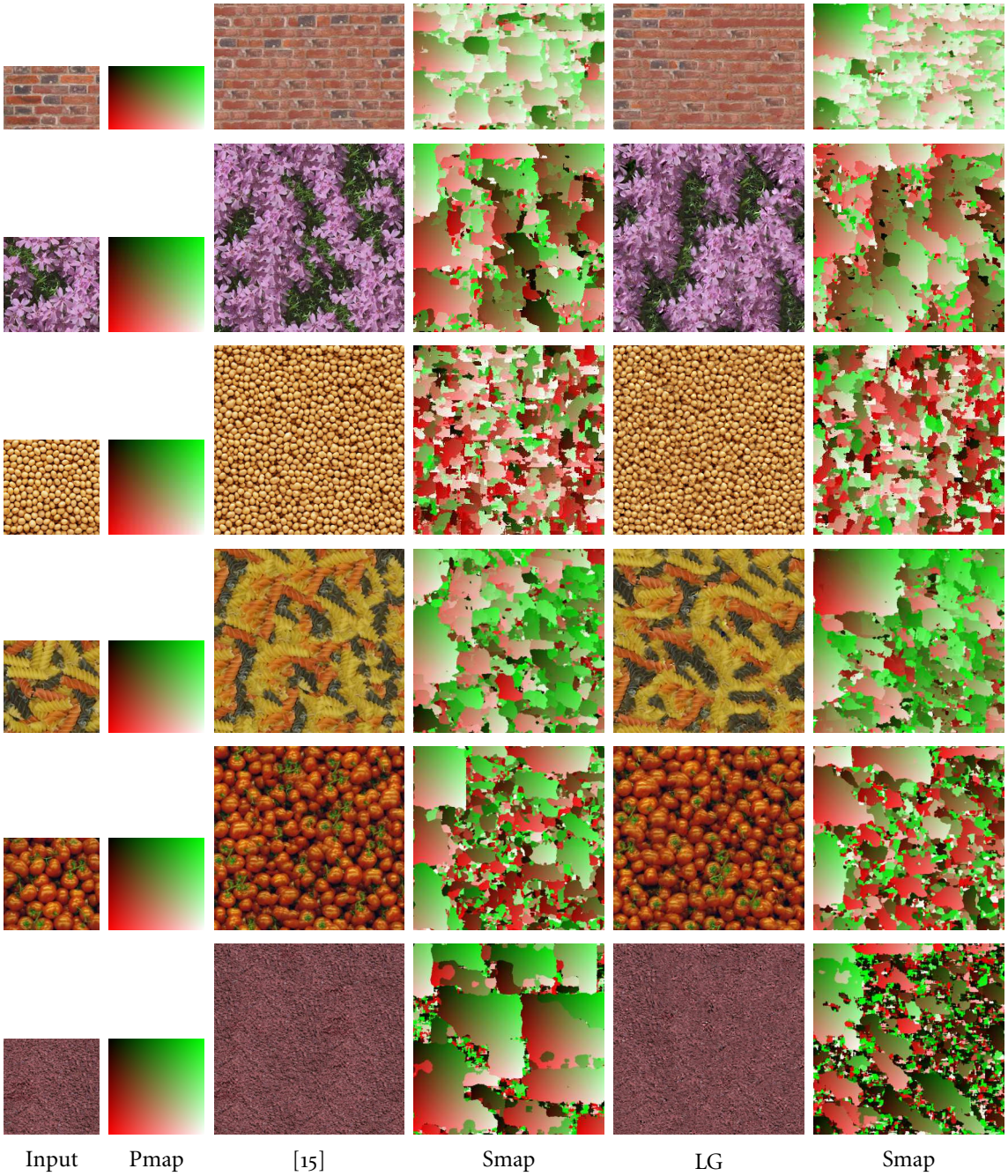


Figure 2.7 – Comparison of the synthesis maps. From left to right: texture sample, associated position map (Pmap), synthesis results of the Efros and Freeman method [15], associated synthesis map (Smap), results of LG and the associated synthesis map (Smap). The results of [15] and LG presented in this image are the same as the ones shown in Figure 2.6.

3 Conditional Gaussian model: a multiscale algorithm

Exemplar based texture synthesis is defined as the process of generating, from an input texture sample, new texture images that are perceptually equivalent to the input. In the present chapter, we model texture self-similarity with conditional Gaussian distributions in the patch space in order to extend the use of stitching techniques. Then, a multiscale texture synthesis algorithm is introduced, where texture patches are modeled at each scale as spatially variable Gaussian vectors in the patch space. The Gaussian distribution for each patch is inferred from the set of its nearest neighbours in the patch space obtained from the input sample. This approach is tested over several real and synthetic texture images and its results show the effectiveness of the proposed technique for a wide range of textures.

3.1 Introduction

In this chapter, an extension and detailed description of a texture synthesis framework is proposed which combines a multiscale approach to the locally Gaussian texture model in the patch space. Each texture patch of the synthesized image is sampled from its Gaussian distribution estimated on a set of similar patches taken in the input sample as defined in Chapter 2. Modeling patches with conditional Gaussian distributions is an approach that is also used in image processing as for instance in the denoising algorithm of Buades et al. [38]. The first question that was brought up was how to stitch together the patches. In Chapter 2, the Efros and Freeman's [15] was adopted, where every new patch overlaps the previously synthesized one, and the overlapped parts are blended. In this chapter we consider conditional Gaussian models constrained to respect the values of its overlapping zone. This solution shows satisfying results for periodic or pseudo-periodic textures but fails for more complex ones. The second and central question brought up by non paramatric patch based methods is: how to handle the strong dependency of the method on the patch size in particular when dealing with macro-textures. Macro-textures show information at different scales that cannot be captured with a unique patch size. To this aim, a multiscale approach is considered. A full description and discussion is provided here.

In this chapter, we aim at extending the Gaussian model proposed in Chapter 2 to take into account the stitching step. We first recall briefly the principle of non-parametric patch-based methods and review the most recent contributions. As we have seen in Chapter 2, non-parametric patch-based methods were initialized by Efros and Leung [14] who extended to images Shannon's Markov random field model initially devised to simulate text. In analogy with Shannon's algo-

rithm for synthesizing sentences, the texture is constructed pixelwise. For each new pixel in the reconstructed image, a patch centered at the pixel is compared to all patches with the same size in the input sample. The nearest matches in the input help predict the pixel value in the reconstructed image. Several optimizations have been proposed to extend and accelerate this algorithm. Among them, Wei and Levoy [63], managed to fix the shape and the size of the learning patch and Ashikmin [2] proposed to extend existing patches whenever possible instead of searching in the entire sample texture. As we mentioned, these pixelwise algorithms are not always satisfactory. They are known to grow "garbage" when the compared patches are too small or when the input texture is not stationary, or may lead to verbatim copies of significant parts of the input sample for large patches as can be verified in [1]. This explains why more recent methods stitch together entire patches instead of performing a synthesis pixel by pixel. The question then is how to blend a new patch in the existing texture. In [42] this is done by a smooth transition. Efros and Freeman [15] refined this process by stitching each new patch along a stitching path with minimal contrast between the new patch and the texture in their overlapping zone. Kwatra et al. in [37] extended the stitching procedure of [15] by a graph cut approach redefining the edges of the patches. In [36] the authors proposed to synthesize a texture image by sequentially improving the quality of a synthesis by minimizing a patch-based energy function. In the same spirit as [36], where texture optimization is performed, the authors in [40] proposed to synthesize textures in a multiscale framework using the coordinate maps of the sample texture at different scales. They introduced spatial randomness by applying a jitter function to the coordinates at each level combined to a correction step inspired by [2]. One of the key strengths of the method is that it is a parallel synthesis algorithm which makes it extremely fast. These non-parametric patch-based approaches often present satisfactory visual results. In particular they have the ability to reproduce highly structured textures. However, the risk remains of copying even several times verbatim large parts of the input sample. Furthermore, a fidelity to the global statistics of the initial sample is not guaranteed, in particular when the texture sample is not stationary. We refer to [62] for an extensive overview of the different non-parametric patch-based methods.

Recently methods such as [49, 59] combine patch-based and statistics-based methods to overcome the drawbacks mentioned previously. In [49], the author proposes to use a patch-based approach where all the patches of the synthesized image are created from a sparse dictionary learnt on the input sample. In [59], Tartavel et al. extend the work of [49] by minimizing an energy that involves a sparse dictionary of patches combined to constraints on the Fourier spectrum of the input sample. Still more recently, Gatys et al. [23] have introduced the neural network methodology into the field. Extending the parametric approach, they characterize a texture by the cross-correlations of kernels learnt from a convolutional neural network dedicated to shape recognition. This algorithm emulates complex textures containing large objects for which the Portilla and Simoncelli algorithm is not satisfactory. Contrarily to the Portilla and Simoncelli approach, no Occam's razor was applied to reduce the number of texture parameters. The number of kernel correlations involved is perhaps exceedingly large: it appears that parts of the input are being recombined in the output. Nevertheless this learning approach is definitely promising.

The rest of this chapter is structured as follows. In Section 3.2 the local Gaussian (LG) model is described. An analysis of the model's variance is provided as well as the description of a first synthesis method. In Section 3.3 two new patch models are introduced: the conditional local Gaussian model (CLG) and the regularized conditional local Gaussian model (RCLG). Section 3.4 presents the multiscale approach (MSLG). Section 3.5 shows several experiments: a comparison of the three patch models proposed, a comparison to the results of state of the art texture synthesis methods and the influence of the parameters involved in the multiscale texture synthesis algorithm.

Conclusions are presented in Section 3.6.

u	$\Omega \to \mathbb{R}$: input texture image defined on the discrete domain $\Omega = I_M \times I_N$ of size $M \times N$ where I_c denotes the discrete interval $[0, \dots, c-1]$
w	$\Omega_r \to \mathbb{R}$: output texture image defined on the discrete domain $\Omega_r = I_{rM} \times I_{rN}$ of size $rM \times rN$
d	is equal to 1 if u is grayscale and to 3 for color images
r	ratio (size of output image w)/(size of input image u)
n	side patch size
m	number of nearest neighbours used to learn the parameters of the Gaussian distribution
o	overlap size
K	number of scales (maximum factor of zoom out is $K-1$)
u_k	$\Omega^k \to \mathbb{R}$: zoom out of u of a factor 2^k, defined on the discrete domain $\Omega^k = I_{2^{-k}M} \times I_{2^{-k}N}$ of size $2^{-k}M \times 2^{-k}N$ for $k = 1 \dots K-1$
w_k	$\Omega_r^k \to \mathbb{R}$: synthesized texture at scale k, defined on the discrete domain $\Omega_r^k = I_{r2^{-k}M} \times I_{r2^{-k}N}$ of size $r2^{-k}M \times r2^{-k}N$ for $k = 0 \dots K-1$
v_k	$\Omega_r^k \to \mathbb{R}$: zoom in of w_{k+1} of factor 2. It is the initialization of the low resolution of the synthesized image w_k for $k = 0 \dots K-2$
$p_u^{(x,y)}$	square patch of size $n \times n$ from an image u of size $M \times N$ at position (x, y), $p_u^{(x,y)} = \{u\left((x,y) + (i,j)\right),\ (i,j) \in [0, \dots, n-1]^2\}$, $(x,y) \in \mathcal{V}_u$ where $\mathcal{V}_u = I_{M-n+1} \times I_{N-n+1}$ denotes the discrete domain of the valid patches in u. The patch $p_u^{(x,y)}$ will be considered as a column vector of size $dn^2 \times 1$
G_σ	Gaussian kernel of mean zero and standard deviation σ, $G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$
L_{u_k}	$\Omega^k \to \mathbb{R}$: low resolution of image u_k, $L_{u_k} = u_k * G_\sigma,\ k = 0 \dots K-2$
H_{u_k}	$\Omega^k \to \mathbb{R}$: high resolution of image u_k, $H_{u_k} = u_k - u_k * G_\sigma,\ k = 0 \dots K-2$
L_{w_k}	$\Omega_r^k \to \mathbb{R}$: low resolution of image w_k, $L_{w_k} = w_k * G_\sigma,\ k = 0 \dots K-2$
H_{w_k}	$\Omega_r^k \to \mathbb{R}$: high resolution of image w_k, $H_{w_k} = w_k - w_k * G_\sigma,\ k = 0 \dots K-2$
$\mathbf{I}_m$	denotes the $m \times m$ identity matrix

Table 3.1 – Summary of the notations used in this chapter.

3.2 Gaussian patches

For all notations used in this section we refer to Table 3.1 for a detailed definition.

3.2.1 The assumption of a Gaussian patch model

We recall in this section the locally Gaussian model suggested in Chapter 2. For a given texture image u, the underlying distribution of every patch $p_u^{(x,y)}$ is modeled as a multivariate Gaussian distribution of mean $\mu^{(x,y)}$ and covariance matrix $\Sigma^{(x,y)}$.

These parameters are estimated from the set of nearest patches $\mathcal{U} = \{p_u^{(x_i,y_i)}, i = 1, \dots, m\}$ where $p_u^{(x_i,y_i)}$ are the m nearest patches to $p_u^{(x,y)}$ in u according to the L^2 distance. The empirical

statistics $\mu^{(x,y)}$ and $\Sigma^{(x,y)}$ are then defined as

$$\begin{aligned} \mu^{(x,y)} &= \tfrac{1}{m}\textstyle\sum_{i=1}^{m} p_u^{(x_i,y_i)} \\ \Sigma^{(x,y)} &= \tfrac{1}{(m-1)} PP^t. \end{aligned} \tag{3.1}$$

We denote by P the matrix whose columns are the normalized patches in vector form $(p_u^{(x_i,y_i)} - \mu^{(x,y)}),\ i = 1, \ldots, m$.
The sampled vector $\tilde{p}_u^{(x,y)}$ is thus defined as a linear combination of the m nearest patches $p_u^{(x_i,y_i)}$

$$\tilde{p}_u^{(x,y)} = \frac{1}{\sqrt{m-1}} PA + \mu^{(x,y)}. \tag{3.2}$$

Here A is a vector whose elements are standar normal independent random variables $A \sim \mathcal{N}(\vec{0}, \mathbf{I}_m)$. Thus, the sampled vector has the correct Gaussian distribution $\mathcal{N}(\mu^{(x,y)}, \Sigma^{(x,y)})$.

In Chapter 2 we measured the variability of the Gaussian models with respect to the patch size n and the number of paches m. For this the mean standard deviation per pixel of a given patch in the texture image was estimated as

$$\bar{\sigma} = \frac{1}{dn^2} \sum_{i=1}^{dn^2} \sqrt{\Sigma^{(x,y)}(i,i)} = \frac{1}{dn^2} \sum_{i=1}^{dn^2} \sigma_i$$

where d is the number of channels(d=1 for grayscale images and $d = 3$ fop color images).We observed that indeed the models considered have reasonable variances confirming that effectively the patches simulated have an acceptable degree of innovation. From now on to simplify the notations when referring to $\mu^{(x,y)}$ and $\Sigma^{(x,y)}$ they will be denoted by μ and Σ respectively.

3.2.2 Texture synthesis algorithm

We remind the locally Gaussian texture synthesis algorithm (LG) presented in Chapter 2 with a different notation that will be useful for the following section. Given the input texture u, an output image w is synthesized sequentially patch by patch in a raster-scan order (left to right, top to bottom). Each new patch added to w overlaps part of the previously synthesized patch as can be seen in Figure 3.1. Depending on the stage of the synthesis three different cases of overlap can be observed: vertical (first row of raster-scan), horizontal (first column of raster-scan) and L-shape (everywhere else). This is also shown in Figure 3.1. Each patch is simulated following a multivariate Gaussian distribution of mean μ and covariance matrix Σ as defined in (3.1). When quilting the patch in w the same stitching step is applied as in Efros and Freeman's work [15] where the authors propose to compute an optimal boundary between the new patch and the previously synthesized one along their overlap region. This is done thanks to a linear programming optimization.

To define the set $\mathcal{U} = \{p_u^{(x_i,y_i)}, i = 1, \ldots, m\}$ of nearest patches to $p_w^{(x,y)}$, let us consider $p_w^{(x,y)}$ as the patch being currently synthesized and taken as a column vector of size $dn^2 \times 1$. The patch $p_w^{(x,y)}$ will overlap part of the previous synthesis. To synthesize $p_w^{(x,y)}$, we decompose it as

$$p_w^{(x,y)} = S^t S p_w^{(x,y)} + R^t R p_w^{(x,y)}. \tag{3.3}$$

Here, $S : \mathbb{R}^{dn^2} \to \mathbb{R}^{dn^2-k}$ and $R : \mathbb{R}^{dn^2} \to \mathbb{R}^k$ are projection operators such that $Rp_w^{(x,y)}$ is a vector of size $k \times 1$ with the values of $p_w^{(x,y)}$ on the overlap area and $Sp_w^{(x,y)}$ is a vector of size

$(dn^2 - k) \times 1$ with the other components of $p_w^{(x,y)}$. This decomposition will be useful in the following section.

The patches $p_u^{(x_i,y_i)}$ used to learn the parameters of the multivariate Gaussian distribution (3.1) are the m nearest neighbours in u to the current patch $p_w^{(x,y)}$, for the L^2 distance restricted to the overlap area, given by $\|Rp_u^{(x_i,y_i)} - Rp_w^{(x,y)}\|_2$. Once the patch $p_w^{(x,y)}$ is synthesized from the Gaussian model (3.1), the values of $Rp_w^{(x,y)}$ change. The texture synthesis algorithm is summarized in Algorithm 2 in section 2.3 of Chapter 2.

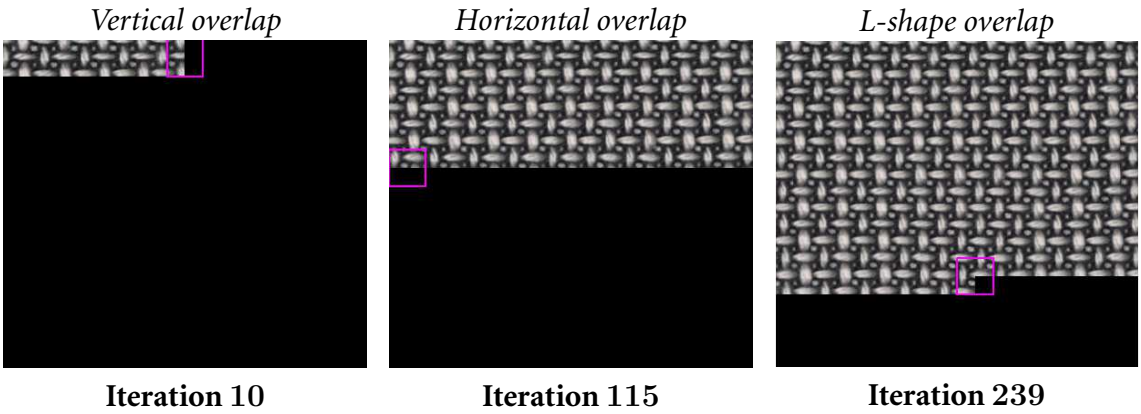


Figure 3.1 – Three different iterations of the synthesis process are shown. At each iteration a patch is being synthesized. This patch is represented by the pink square in the three iterations shown. From left to right the three overlap cases are represented: vertical, horizontal and L-shape.

In Figure 3.2 the synthesis results using the Gaussian sampling are compared to those of the quilting method [15] to illustrate that the Gaussian sampling achieves visual results that are comparable to those in [15] while providing a local parametric model, as shown in Chapter 2. With the Gaussian sampling some blur is introduced in the synthesis results but the effects of verbatim copies and garbage growing are reduced as can be observed in the second example in Figure 3.2. Regarding the uderlying mathematical model of the synthesized image, as discussed in Chapter 2, we cannot claim that the proposed synthesis algorithm corresponds to a complete texture model, because it would require a stochastic model for the *spread*, namely the spatial interaction between patches obeying the same Gaussian model.

3.3 Conditional Gaussian models

The purpose of this section is to analyze how to replace the stitching step by a conditional patch model constrained to the overlap zone. The natural idea is to condition the new Gaussian samples to the values observed in the overlap zone, thus allowing to maintain the same or slightly modifying the overlap pixels. We observed that conditional Gaussian models used in [22] as an inpainting method for microtextures achieved good results.

This requires modeling the transition effect between patches. Each new patch will be estimated as a Gaussian vector conditioned to the pixel values of its corresponding overlap region. In this way the simulated patch would naturally "agree" with w in the overlap area, thus avoiding a stitching procedure.

Such models will be considered in the following sections.

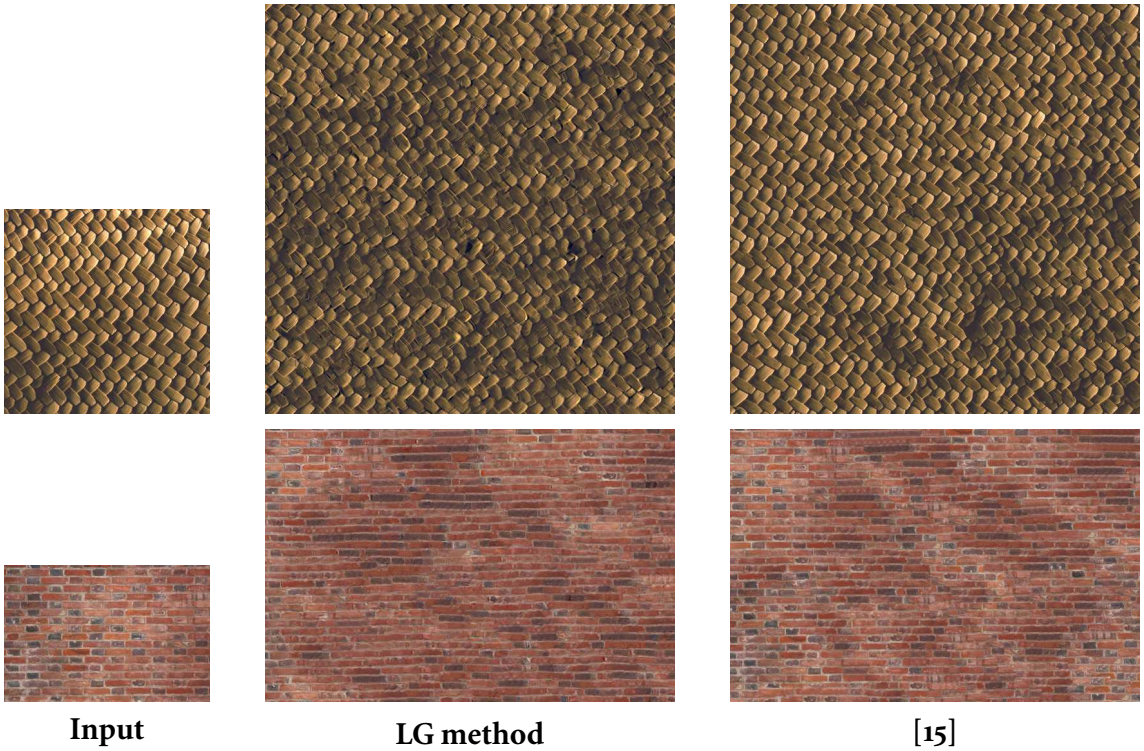


Figure 3.2 – Comparison of LG to the quilting method [15]. These results are done taking equal parameters and seed patch to initialize the synthesized imaged. The parameters used are $n = 40$, $o = n/2$ and $m = 30$ (for the LG model).

3.3.1 A first conditional Gaussian model

In this first patch model the idea is to model sample patches obeying a Gaussian distribution that exactly match with their overlap area pixels, avoiding the unwanted discontinuities. We shall call this model Conditional Locally Gaussian (CLG).

Each patch $p_w^{(x,y)}$ is taken as a column vector of size $dn^2 \times 1$ and can be partitioned as

$$p_w^{(x,y)} = S^t S p_w^{(x,y)} + R^t R p_w^{(x,y)}.$$

We assume throughout that the vector $p_w^{(x,y)}$ follows a Gaussian distribution of mean μ and covariance matrix Σ. These parameters can be expressed as follows :

$$\begin{aligned}
\mu =& S^t S\mu + R^t R\mu = S^t\mu_1 + R^t\mu_2 \\
&\text{and} \\
\Sigma =& S^t S\Sigma S^t S + S^t S\Sigma R^t R + R^t R\Sigma S^t S + R^t R\Sigma R^t R \\
=& S^t\Sigma_{1,1}S + S^t\Sigma_{1,2}R + R^t\Sigma_{2,1}R + R^t\Sigma_{2,2}R.
\end{aligned}$$

The problem can be formulated as finding the "best sample" $\tilde{p} = (\tilde{p}_1, \tilde{p}_2)$ conditioned to the overlap values q_0 that are known,

$$\tilde{p}_1 = \arg\max_{p_1} \mathbb{P}_{\mu,\Sigma}(P_1 = p_1 \mid P_2 = q_0). \tag{3.4}$$

Here P_1 corresponds to the unknown values of $p_w^{(x,y)}$ and P_2 to the values of $p_w^{(x,y)}$ on the overlap area, i.e. $P_1 = Sp_w^{(x,y)}$ and $P_2 = Rp_w^{(x,y)}$.
Using classic results on conditional multivariate Gaussian distributions [46], the distribution of P_1 conditioned to $P_2 = q_0$ is a multivariate Gaussian distribution of parameters $\bar{\mu}$ and $\bar{\Sigma}$ where

$$\begin{aligned}
&\bar{\mu} = \mu_1 + (S\Sigma R^t)(R\Sigma R^t)^{-1}(q_0 - \mu_2) \\
&\text{and} \\
&\bar{\Sigma} = (S\Sigma S^t) - S\Sigma R^t(R\Sigma R^t)^{-1}R\Sigma S^t.
\end{aligned}$$

Since the most probable sample of a multivariate Gaussian distribution is its mean, the solution to (3.4) is

$$\tilde{p}_w = R^t q_0 + S^t(\mu_1 + (S\Sigma R^t)(R\Sigma R^t)^{-1}(q_0 - \mu_2)). \tag{3.5}$$

This first result has the advantage that since the mean of the conditional distribution is taken, the risk of unwanted sharp transitions between patches for some types of textures will be attenuated. Another observation is that since this new model might not have very probable samples, working with the most probable one improves the visual aspect of the sampled patch. The negative aspects of taking the most probable sample is that on the one hand the result, for a given initialization, is deterministic. On the other hand the synthesis result loses details by displaying the mean of the conditional distribution.

To overcome both mentioned drawbacks we sample a patch from the underlying Gaussian distribution, that is,

$$\tilde{p} = R^t q_0 + S^t \tilde{p}_1,\ \tilde{p}_1 \sim \mathcal{N}(\bar{\mu}, \bar{\Sigma}). \tag{3.6}$$

Yet equations (3.5) and (3.6) show that these solutions do not make sense if $(R\Sigma R^t)$ is not invertible. This is unfortunately frequent, as the number of neighbours m used to build the Gaussian distribution is often very small compared to the dimension of the vectors we aim to model. Therefore the learnt Gaussian models are strongly degenerated.
The fact that Σ is not invertible implies that the Gaussian vectors $p_w^{(x,y)} \sim \mathcal{N}(\mu, \Sigma)$ live in a subspace of $\mathbb{R}^{dn^2}$. However it does not say that there is no solution to (3.4). Indeed there are two possible cases:

1. The subspace of Gaussian vectors ($\mathcal{N}(\mu, \Sigma)$) intersects the subspace of vectors $R^t q_0 + S^t p_1$. Thus the Gaussian distribution $\mathcal{N}(\bar{\mu}, \bar{\Sigma})$ exists and there is a solution to our problem.

2. The subspace of Gaussian vectors ($\mathcal{N}(\mu, \Sigma)$) does not intersect the subspace of Gaussian vectors $R^t q_0 + S^t p_1$. Therefore there is no solution.

To avoid the second case (having no solution) a small perturbation is added to the Gaussian distribution learnt for $p_w^{(x,y)}$ as follows

$$p_w^{(x,y)} \sim \mathcal{N}(\mu, \Sigma + \sigma^2 \mathbf{I}_{dn^2}),$$

where σ^2 is a real positive number and $\mathbf{I}_{dn^2}$ is the identity matrix of size $dn^2 \times dn^2$ ensuring that the problem is well conditioned. In that way the Gaussian vectors $p_w^{(x,y)}$ live in all $\mathbb{R}^{dn^2}$, and

this ensures the existence of the conditional multivariate Gaussian distribution sought. The new covariance matrix is denoted by $\Gamma = \Sigma + \sigma^2 \mathbf{I}_{dn^2}$.

If we are in the case where the Gaussian vectors subspace $p_w^{(x,y)} \sim \mathcal{N}(\mu, \Sigma)$ intersects the set of Gaussian vectors $R^t q_0 + S^t p_1$, using the new distribution $\mathcal{N}(\mu, \Gamma)$ will slightly modify the solutions in (3.5) and (3.6) when using small values of σ^2. Thus it is enough to take a low value for σ^2 and the solutions obtained in both cases ($\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu, \Gamma)$) will be very close to each other.

Adding this perturbation to the multivariate Gaussian model has the drawback of increasing the computational cost of sampling a Gaussian vector. For the model in Section 3.2.1 a Gaussian vector was sampled by a simple linear combination of m patches. There was no need in estimating the covariance matrix of the Gaussian distribution. The model presented in this section requires the computation of the Gaussian parameters, in particular the covariance matrix $\bar{\Gamma} = (S\Gamma S^t) - S\Gamma R^t (R\Gamma R^t)^{-1} R\Gamma S^t$. The covariance matrix $\bar{\Gamma}$ is symmetric and definite positive and then admits a Cholesky factorization, i.e. there is a unique lower triangular matrix C such as $\bar{\Gamma} = CC^t$. Thus the vector $s = Ct + \bar{\mu}$ yields a sample of the desired Gaussian distribution $\mathcal{N}(\bar{\mu}, \bar{\Gamma})$ with $t \sim \mathcal{N}(\vec{0}, \mathbf{I}_{dn^2})$. The complexity of computing the Gaussian samples with the cholesky decomposition is $\mathcal{O}(n^3)$ whereas the complexity of the first model is $\mathcal{O}(nm)$.

This synthesis method is summarized in Algorithm 3 and Algorithm 4.

Algorithm 3: CLG sampling

Input: set of nearest pathces $\mathcal{U}$, overlap values q_0, projection operators R and S
Output: Gaussian sample F

1: $\mu \leftarrow \frac{1}{m} \sum_{i=1}^{m} p_u^{(x_i,y_i)}$, mean of the Gaussian model
2: $\Sigma \leftarrow \frac{1}{(m-1)}(PP^t)$, covariance matrix of the Gaussian model
3: $\Gamma \leftarrow \Sigma + \theta^2 I_{n^2}$, covariance matrix of full rank
4: $\bar{\mu} \leftarrow S\mu + (S\Gamma R^t)(R\Gamma R^t)^{-1}(q_0 - R\mu)$, conditional Gaussian mean
5: $\bar{\Gamma} \leftarrow (S\Gamma S^t) - S\Gamma R^t (R\Gamma R^t)^{-1} R\Gamma S^t$, conditional covariance matrix
6: $C \leftarrow$ cholesky decomposition of matrix $\bar{\Gamma}$
7: $t \leftarrow$ Gaussian sample $\mathcal{N}(\vec{0}, \mathbf{I}_{n^2})$
8: $s \leftarrow Ct + \bar{\mu}$ Gaussian vector following $\mathcal{N}(\bar{\mu}, \bar{\Gamma})$
9: **return** F

3.3.2 The regularized conditional Gaussian model

In the previous section the patches' statistical models were conditioned to the exact values of the synthesized pixels across their overlap area. This is too restrictive for some types of textures and is at risk of creating unlikely samples. Instead of forcing each patch $p_w^{(x,y)}$ to take the exact same values on the previously synthesized part, it is therefore natural to allow the patch to vary slightly on the overlap area. This variation is rendered necessary by the scarcity of patch samples in a small texture sample. We shall call this model Regularized Conditional Locally Gaussian (RCLG). Consider the same patch model $\mathcal{N}(\mu, \Sigma)$, but let us now allow the overlap components $P_2 = Rp_w^{(x,y)}$ to take values $P_2 = q_0 + b$, where $b \sim \mathcal{N}(0, \theta^2 \mathbf{I}_k)$ and where θ is the variation allowed for the overlap values. The joint distribution of (P_1, P_2) is defined by:

Algorithm 4: CLG texture Synthesis

Input: input texture sample u, side patch size n, overlap size o, number of nearest neighbours m, ratio (output size)/(input size) r

Output: synthesized texture w

```
Initialize w placing a seed patch in its top-left corner (x = 1, y = 1). The image w is
  of size rM × rN, where M × N is the size of u.
for x = 1 : n − o : rM do
  for y = 1 : n − o : rN do
    if (x > 1 or y > 1) then
      q_0 ← known overlap values
      U ← set of m nearest neighbours to p_w^(x,y)
      p̃_w^(x,y) ← CLG sampling(U, q_0)
      Quilt p̃_w^(x,y) in w at position (x, y)
    end if
  end for
end for
return w
```

$$\begin{aligned}\mathbb{P}(p_1,p_2) =& \mathbb{P}_{\mu,\Sigma}(p_1|p_2)\mathbb{P}_{q_0,\theta^2 I_k}(p_2)\\ =& \frac{1}{Z}e^{-\frac{1}{2}((p_1,p_2)-(\bar{\mu},q_0))^t\Delta^{-1}((p_1,p_2)-(\bar{\mu},q_0))},\end{aligned} \tag{3.7}$$

where $Z = \sqrt{2\pi\det(\Delta)}$ and $\Delta = \begin{pmatrix} \bar{\Sigma} & \mathbf{0} \\ \mathbf{0} & \theta^2\mathbf{I}_k \end{pmatrix}$.

Proceeding as in the previous model (CLG) the most probable sample

$$\tilde{p} = \arg\max_{(p_1,p_2)} \mathbb{P}_{\mu,\Sigma}(p_1|p_2)\mathbb{P}_{0,\theta^2\mathbf{I}_k}(p_2 - q_0)$$

is exactly the same as in (3.5) and has the same drawbacks. On the other hand sampling from (3.7) allows to relax the overlap constraint.

Once again, to guarantee the existence of the solution, the Gaussian distribution of $p_w^{(x,y)}$ is slightly modified as done in the CLG model. A small perturbation is added to the covariance matrix Σ and the problem is then well conditioned. Thus, as in the CLG model, the computational cost of sampling from this new Gaussian distribution is higher compared to the LG sampling as mentioned for the CLG model since once again the parameters of the RCLG are estimated. The sampling process is similar to the one described in Algorithm 3. The difference is that the known values of the overlap are slightly modified to relax the constraint. Apart from that the procedure is the same where in one case the overlap values are exactly those from the previous synthesis (CLG) and in the second case they are slightly modified as $q_0 + b$ where $b \sim \mathcal{N}(\vec{0}, \theta^2\mathbf{I}_k)$.

3.4 A multiscale generalization

For all notations used in this section we refer to Table 3.1 for a detailed definition.

Macro textures present details at different scales: a coarse one that contains the global structure of the texture and finer ones containing the details. Small patch sizes may capture the finer details of the input but the resulting texture will lack global coherence. On the other hand using large patches will maintain better the global structures on the risk of a "copy-paste" effect. Furthermore with large patches it becomes impossible to model the patch variability due to the curse of dimensionality, in other terms the lack of sufficient samples. This is for example apparent in the examples of Chapter 2 where modeling patches as multivariate Gaussian vectors leads to a slightly blurry texture.

Multiscale approaches permit to contemplate several patch sizes within one synthesis, i.e. to capture the different levels of details. If we fix the patch size to be $n \times n$ and use K scales this is similar to using K different patch sizes going from $2^{K-1}n \times 2^{K-1}n$ to $n \times n$ for the coarsest to finer details within one synthesis.

In this section the potential of a multiscale approach is illustrated by improving the method described in Algorithm 2. Let us first introduce some notations. The input texture sample is denoted by u and $u_k,\ k = 1, \ldots, K-1$ are the zoomed out versions of u by a factor $2^k,\ k = 1, \ldots, K-1$. The synthesis result at each scale is denoted by $w_k,\ k = 1, \ldots, K-1$ and w is the synthesis result returned by the multiscale algorithm. An additional image needed at each scale is the low resolution of the result w_k that is denoted by v_k and its the result of zooming in w_{k+1}. The multiscale approach can be summarized in a few sentences. The method begins by a synthesis at the coarsest scale ($k = K-1$) using the local Gaussian method in Algorithm 2 where the quilting step is replaced by a simple average of the overlapping patches. For the remaining scales ($k = K-2, \ldots, 0$) a synthesis is performed by using the result of the previous scale ($k+1$) and the input of corresponding resolution. At each scale the synthesis is done patch by patch in a raster-scan order. Each new patch, added to the synthesized image, overlaps part of the previously synthesized patch and is the combination of a low resolution patch and a high resolution one sampled from a multivariate Gaussian distribution. The Gaussian distribution of the high frequencies of a given patch is estimated from the high frequencies of its m nearest neighbours in the corresponding scale input image. The synthesis result of the finer scale is the desired output image.

We shall call this method Multiscale Locally Gaussian (MSLG). In the following the different parts of the method described in Algorithm 5 are detailed.

Zoom-out The zoom-out operation is based on Gaussian filtering. It is performed as a smooth frequency cut-off followed by a sub-sampling of factor two. These operations are detailed below in (3.8), (3.9) and (3.10).

The smooth frequency cut-off is performed with the Gaussian kernel

$$G_\sigma(x, y) = \frac{1}{\sigma^2 2\pi} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.8}$$

where we chose $\sigma = 1.4$. Indeed as proved in [45] if we consider a Gaussian kernel of standard deviation $\sigma = \sqrt{(0.8\alpha)^2 - 0.8^2}$ to smooth the image and then sub-sample it by a factor α we can assume that no aliasing is generated.
The sub-sampling operation by a factor two applied to an image $u : I_M \times I_N \mapsto \mathbb{R}$ is defined by

$$\begin{aligned} \mathcal{S}_2 &: I_M \times I_N \mapsto I_{M/2} \times I_{N/2}, \\ \mathcal{S}_2(u)(i, j) &= u(2i, 2j),\ (i, j) \in I_{M/2} \times I_{N/2}. \end{aligned} \tag{3.9}$$

The zoom-out of an image u of factor two is then

$$\mathcal{Z}_2^{\text{out}}(u) = \mathcal{S}_2(u * G_\sigma). \tag{3.10}$$

The images obtained after applying this zoom-out operation have no aliasing and they are blurry enough to avoid ringing artifacts when applying the zero padding zoom-in.

Zoom-in For an image $u : I_M \times I_N \to \mathbb{R}$, $v = \mathcal{Z}_2^{\text{in}}(u) : I_{2M} \times I_{2N} \to \mathbb{R}$ is the zoom-in by a factor two. This operation is performed by a zero padding of $\hat{u} : \hat{I}_{2M} \times \hat{I}_{2N} \to \mathbb{C}$ where $\hat{u}$ denotes the discrete Fourier transform of u and $\hat{I}_c$ denotes the discrete interval $[-c/2, \ldots, c/2 - 1]$ for c even and $[-(c-1)/2, \ldots, (c-1)/2]$ for c odd. This is done as

$$\hat{v}(\xi_1, \xi_2) = \begin{cases} \hat{u}(\xi_1, \xi_2) & \text{if } |\xi_1| \leq \left\lfloor \frac{M-1}{2} \right\rfloor,\ |\xi_2| \leq \left\lfloor \frac{N-1}{2} \right\rfloor \\ 0 & \text{else} \end{cases} \tag{3.11}$$

where $\hat{v}$ is the discrete Fourier transform of v and $\lfloor x \rfloor$ denotes the integer part of x.

The relation between the zoom in and zoom out operators

$$\mathcal{Z}_2^{\text{in}}(\mathcal{Z}_2^{\text{out}}(u)) = u$$

is valid when the image u verifies

$$\hat{u}(\xi_1, \xi_2) = 0,\ \forall |\xi_1| > \left\lfloor \frac{M-1}{2} \right\rfloor, |\xi_2| > \left\lfloor \frac{N-1}{2} \right\rfloor.$$

We could have chosen other interpolation techniques as for example a spline interpolation. But a zero padding is well suited due to the nature of the zoomed out images which are blurry enough to avoid any ringing artifacts.

Distance between patches To estimate the parameters of the Gaussian distribution of the patch being processed, denoted by $p_{w_k}^{(x',y')}$, the set $\mathcal{U}$ of m nearest patches in u_k to $p_{w_k}^{(x',y')}$ is considered. These patches are those minimizing the distance to $p_{w_k}^{(x',y')}$ defined in (3.12) for $k = K-1$ and in (3.13) for the remaining scales $k = K-2, \ldots, 0$.

The size of patch overlap is fixed to half the patch size $n/2$. As mentioned in Section 3.2 there are three overlap cases: vertical (V.O.), horizontal (H.O.) and L-shape (L.O.). Here they are denoted as

$$Op_u^{(x,y)} = \{u\left((x,y) + (i,j)\right),\ (i,j) \in \mathcal{O}\}$$

where

$$\mathcal{O} = \begin{cases} [0, \ldots, n-1] \times \left[0, \ldots, \frac{n}{2} - 1\right] & \text{if V.O.} \\ \left[0, \ldots, \frac{n}{2} - 1\right] \times [0, \ldots, n-1] & \text{if H.O.} \\ \left[0, \ldots, \frac{n}{2} - 1\right] \times [0, \ldots, n-1] \cup & \\ \left[\frac{n}{2}, \ldots, n-1\right] \times \left[0, \ldots, \frac{n}{2} - 1\right] & \text{if L.O.} \end{cases}$$

When $k = K-1$, the m nearest neighbours in u_{K-1} to the current patch $p_{w_{K-1}}^{(x',y')}$ are those minimizing the L^2 distance restricted to the overlap area:

$$d(Op_{u_{K-1}}^{(x,y)}, Op_{w_{K-1}}^{(x',y')})^2 = \frac{1}{|\mathcal{O}|} \sum_{(i,j)\in\mathcal{O}} (u_{K-1}(x+i, y+j) - w_{K-1}(x'+i, y'+j))^2. \tag{3.12}$$

When $k = K-2, \ldots, 0$, the nearest neighbours in u_k to the patch $p_{w_k}^{(x',y')}$ are those minimizing a distance (3.13) similar to (3.12) with an additional term taking into account the low resolution v_k (the synthesis result of the previous scale $k+1$). In (3.13) L_{u_k} denotes the low resolution of the image u_k, $L_{u_k} = u_k * G_\sigma,\ k = 0, \ldots, K-2$. It is important to notice that when comparing $Op_{u_k}^{(x,y)}$ and $Op_{w_k}^{(x',y')}$ the low and the high resolution must be considered jointly, they are not independent.

$$d(p_{u_k}^{(x,y)}, p_{w_k}^{(x',y')})^2 = \frac{1}{|\mathcal{O}|} \sum_{(i,j)\in\mathcal{O}} (u_k(x+i, y+j) - w_k(x'+i, y'+j))^2 + \frac{1}{n^2} \sum_{i,j=0}^{n-1} (L_{u_k}(x+i, y+j) - v_k(x'+i, y'+j))^2 \tag{3.13}$$

Blending process The blending process consists in simply averaging the values across the overlap area as in (3.14). This step is applied only for the synthesis of scale $k = K-1$.

$$w_k(x+i, y+j) = \begin{cases} \frac{1}{2}(\tilde{p}_{w_{K-1}}^{(x,y)}(i,j) + p_{w_{K-1}}^{(x,y)}(i,j)) & \text{if } (i,j) \in \mathcal{O} \\ \tilde{p}_{w_{K-1}}^{(x,y)}(i,j) & \text{if } (i,j) \in I_n^2 - \mathcal{O} \end{cases} \tag{3.14}$$

Synthesizing patches at scales $k = K-2, \ldots, 0$ At each scale k a patch $p_{w_k}^{(x,y)}$ is synthesized as the combination of a low resolution patch with a high resolution one. It can be decomposed as

$$\begin{aligned} p_{w_k}^{(x,y)} &= p_{w_k * G_\sigma}^{(x,y)} + (p_{w_k}^{(x,y)} - p_{w_k * G_\sigma}^{(x,y)}) \\ &= p_{v_k}^{(x,y)} + (p_{w_k}^{(x,y)} - p_{v_k}^{(x,y)}) \\ &= p_{L_{w_k}}^{(x,y)} + p_{H_{w_k}}^{(x,y)}. \end{aligned}$$

Here L_{w_k} denotes the low resolution image of w_k defined as $L_{w_k} = w_k * G_\sigma,\ k = 0 \ldots K-2$ and H_{w_k} denotes the high resolution image of w_k defined as $H_{w_k} = w_k - w_k * G_\sigma,\ k = 0 \ldots K-2$. The set $\mathcal{U}$ defines the Gaussian distribution of $p_{L_{w_k}}^{(x,y)} \sim \mathcal{N}(\mu_L, \Sigma_L)$ and $p_{H_{w_k}}^{(x,y)} \sim \mathcal{N}(\mu_H, \Sigma_H)$ and therefore the distribution of the patch $p_{w_k}^{(x,y)} \sim \mathcal{N}(\mu, \Sigma)$ where

$$\mu = \mu_H + \mu_L$$

and

$$\Sigma = \Sigma_L + \Sigma_H + \mathrm{cov}\left(p_{L_{w_k}}^{(x,y)}, p_{H_{w_k}}^{(x,y)}\right) + \mathrm{cov}\left(p_{L_{w_k}}^{(x,y)}, p_{H_{w_k}}^{(x,y)}\right)^t.$$

Instead of sampling $p_{L_{w_k}}^{(x,y)}$ from its Gaussian distribution, $p_{v_k}^{(x,y)} \sim \mathcal{N}(\mu_L, \Sigma_L)$ is kept to conserve the low resolution synthesis from the previous scale. The high frequency patch $p_{H_{w_k}}^{(x,y)}$ is sampled form $\mathcal{N}(\mu_H, \Sigma_H)$ and then added to $p_{v_k}^{(x,y)}$. In this way the correlations between high and low resolution pixels are respected, using the low resolution synthesis v_k as initialization.

For all scales beside the coarsest one the synthesis is done by adding the high frequencies of the corresponding scale on the the low resolution basis image. It is the important to achieve a correct basis image in the coarsest scale on which the high frequencies will be added. The texture synthesis method is summarized in Algorithm 5.

Algorithm 5: Multiscale texture syntesis algorithm (MSLG)

Input: input texture sample u, side patch size n, number of nearest neighbours m, number of scales K, ratio (output size)/(input size) r

Output: synthesized texture w

1: Define $u_k \leftarrow \mathcal{Z}_2^{\text{out}}(u_{k-1}),\ k = 1 \ldots K-1$
2: Define $L_{u_k} \leftarrow u_k * G_\sigma,\ k = 0 \ldots K-2$
3: Synthesize $w_{K-1} \leftarrow \mathrm{LG}(u_{K-1}, n, m, r)$ {see Algorithm 2}
4: **for** $k = K-2 : 0$ **do**
5: $\quad v_k \leftarrow \mathcal{Z}_2^{\text{in}}(w_{k+1})$
6: $\quad$ Initialize w_k with zeros of same size as v_k
7: $\quad$ **for** $x = 1 : n/2 : (r2^{-k}M - n + 1)$ **do**
8: $\quad\quad$ **for** $y = 1 : n/2 : (r2^{-k}N - n + 1)$ **do**
9: $\quad\quad\quad$ Compute $d(p_{u_k}^{(x',y')}, p_{w_k}^{(x,y)})$, for all (x', y') in $\mathcal{V}_{u_k}$, where $\mathcal{V}_{u_k}$ denotes the discrete domain of the valid patches in u_k.
10: $\quad\quad\quad$ $\mathcal{U} \leftarrow \{p_{u_k}^{(x_i,y_i)},\ i = 1, \ldots, m\}$ set of m nearest patches in u_k that minimize $d(p_{u_k}^{(x',y')}, p_{w_k}^{(x,y)})$
11: $\quad\quad\quad$ $\mathcal{H} \leftarrow \{p_{u_k}^{(x,y)} - p_{u_k}^{(x,y)} * G_\sigma,\ \forall p_{u_k}^{(x,y)} \in \mathcal{U}\}$, high frequency of the corresponding patches in $\mathcal{U}$
12: $\quad\quad\quad$ Learn (μ, Σ) the parameters of the multivariate Gaussian distribution on the patches of $\mathcal{H}$
13: $\quad\quad\quad$ Sample $\tilde{p} \sim \mathcal{N}(\mu, \Sigma)$
14: $\quad\quad\quad$ $\tilde{p}_{w_k}^{(x,y)} \leftarrow p_{v_k}^{(x,y)} + \tilde{p}$
15: $\quad\quad\quad$ $w_k((x,y) + (i,j)) \leftarrow \tilde{p}_{w_k}^{(x,y)}$ for (i,j) in I_n^2
16: $\quad\quad$ **end for**
17: $\quad$ **end for**
18: **end for**
19: **return** w

3.5 Experiments

In this section, texture synthesis results are shown using the algorithm described in Algorithm 5. In Figure 3.3, general results of the multiscale method are shown with success and failure cases. In Figure 3.6 several texture synthesis methods are compared. In Figure 3.5 the innovation capacity of our method is compared to [15] using coordinate maps. In Figure 3.4 the patch models introduced in Section 3.2 and 3.3 are compared. Finally, the influence of the parameters is discussed in Figure 3.8. There are four of them: the patch size n, the number of neighbours m, the overlap size o and the number of scales K used in the multiscale approach. This is illustrated with two texture examples.

In general the results shown in Figure 3.3 are satisfying for a wide range of textures. The global structures are reproduced by the multiscale approach, while the local structures are maintained by the patch based approach. Using a Gaussian patch model allows to create new patches that do not exist in the input example while maintaining satisfying visual results. Based on the analysis of the patch's variance illustrated in Section 2.2 of Chapter 2 this guarantees that indeed the simulated patches are sufficiently different from the ones in the input sample. The examples of the last two rows illustrate some failure cases of the method. The main failure cause is the size of the input texture. It is obvious that the input must be large enough to provide us with a sufficient number of patch samples to estimate their distribution. If that is not the case, even though the patches are pixel-wise different of the input ones the visual aspect may remain too similar and cause a "copy-paste" effect when m is small enough. Furthermore, as in other non parametric methods,"garbage", namely the excessive use of a subset of patches in the input image is not fully avoided. This effect is nevertheless mitigated by the multiscale approach.

3.5.1 Model comparison

The results in Figure 3.4 show the effects of avoiding the use of the blending step in Algorithm 2. The three models LG, CLG and RCLG were tested on several types of textures. For the three of them the quilting step was omitted. This was done to achieve a better comparison of the capacity of respecting the overlap by modeling or not the restriction of the patches to the overlap values. The results are compared for the three models applied in the sampling mode. The results of the same models in the best sample patch mode are less interesting and are not shown here. The general conclusion of the results in Figure 3.4 is that the conditional patch models achieve a better transition between patches on their overlap region, as expected. Nevertheless the results for CLG and RCLG both loose some resolution compared to the LG results. One can observe in Figure 3.4 that blur appears progressively in the raster-scan order of the synthesis algorithm. Indeed, when moving forward in the synthesis the m nearest patches start being too different from each other (due to the strong restriction of keeping the values of the overlap) and the result is a much blurrier patch. The first five texture examples in Figure 3.4 show better stitching results for CLG and RCLG than for to LG. Looking carefully at the synthesis results of the LG model the edges between patches are more noticeable than in the other two models where almost no transition effect can be seen. Nevertheless the loss of resolution caused by the Gaussian distributions is increased in both conditional models, mostly for CLG. In the last row example the results are less convincing. There are not enough reliable samples to estimate a correct conditional model. The visual results of CLG and RCLG get more and more degraded in the order of the raster scan. This is not surprising since the sample patches respecting the overlap values become increasingly unlikely.

From the experiments in Figure 3.4 one can conclude that a quilting technique is still needed

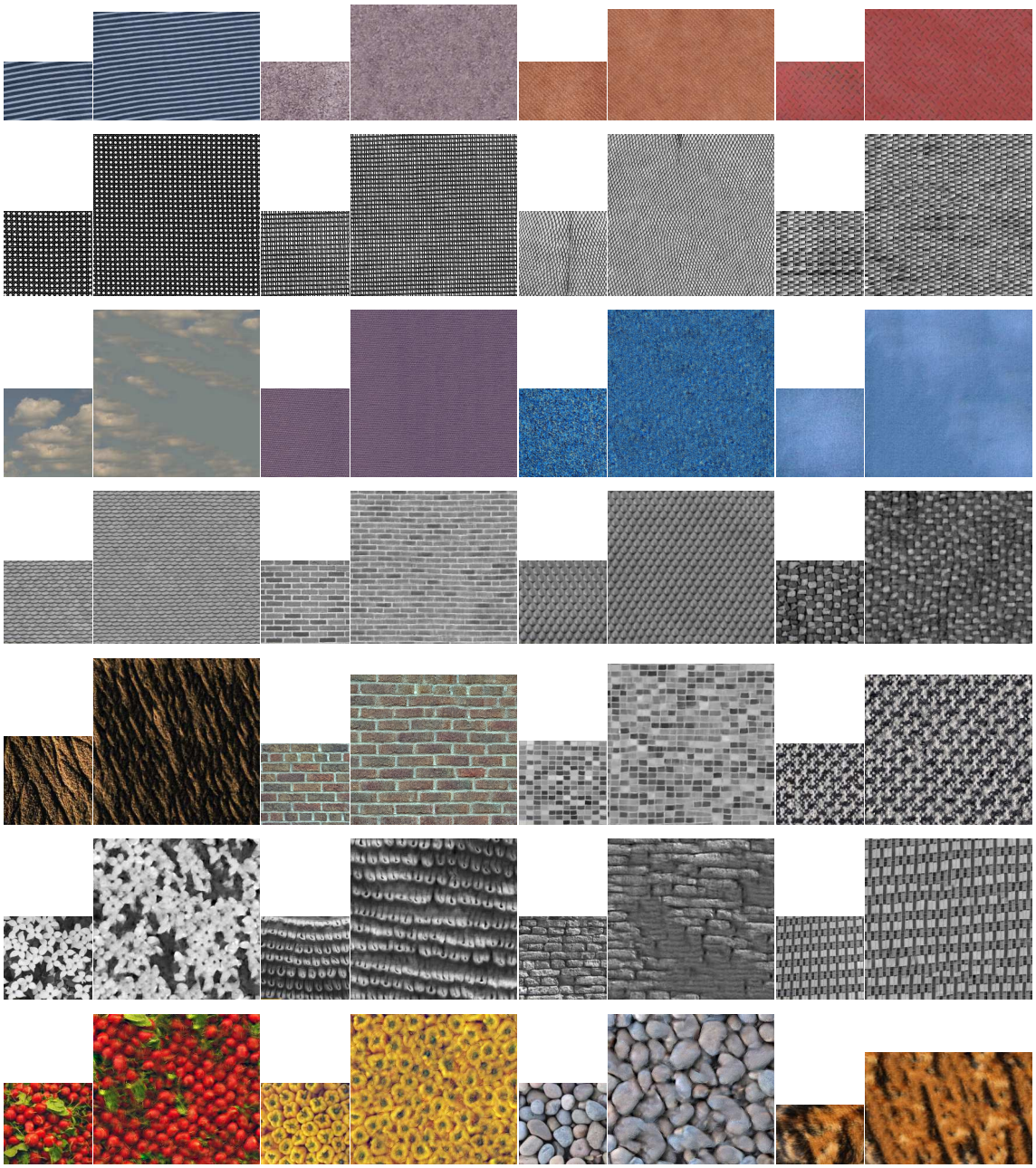


Figure 3.3 – *Synthesis results of the multiscale algorithm*. The small images correspond to the texture samples and the big ones are the corresponding synthesis results. The parameters used for all examples are $n = 20$, $m = 20$ and $K = 2$.

for complex textures and therefore the LG model is better to model locally the texture input. It generates less blur in the results and has a significant smaller computational cost. In the rest of the experiments only the LG model in considered. Nevertheless it could be interesting to test the use of conditional models in the multiscale version at the coarsest scale.

In Figure 3.5 the LG model is compared to the MSLG model. One can observe the strength of using the multiscale approach in terms of reproducing global arrangements that are not kept in the LG model for a fixed patch size. This allows to obtain satisfying results for small values of n and therefore gives more freedom to choose the number of nearest patches m. One can also conclude from the experiments in Figure 3.5 that the multiscale approach generates blurrier textures than the LG model.

3.5.2 Comparison to other texture synthesis methods

In this section the results of Algorithm 5 are compared to other synthesis methods such as [50, 59, 15]. In general, the results obtained with the multiscale locally Gaussian method are visually comparable to the non-parametric patch based method of Efros and Freeman [15], with the advantage that now, the patches are being sampled from their Gaussian model and therefore are different from their original patches. In Figure 3.6, the first column shows the result of the proposed method. A noticeable drawback of the method is the loss of resolution caused by the use of Gaussian distributions. The proposed method was therefore combined with the Portilla and Simoncelli's algorithm [50] as a first and simple approach to overcome this resolution loss. To combine both methods we first synthesize the texture example with MSLG and use this synthesis result as the initialization of the Portilla and Simoncelli algorithm instead of a "noise image". The result can be seen in the second column of Figure 3.6. The result of combining both algorithms is very satisfying. On the one hand the local and global structures are kept due to use of the patch based and multiscale method. On the other hand [50] allows to respect the global statistics of the input or at least be quite close to them. Comparing columns one and two to the third one shows how the combination of both methods improves the results of each method used separately. Of course this solution is limited in particular when the size of the synthesized image is too large. The fourth column shows the results of the Tartavel et al. method [59]. It is interesting to compare our results to this method since both approaches are multiscale, patch based, and create systematically new patches. The results for organized highly structured textures are comparable for both cases. Nevertheless for more complex textures like the flower example and the last two rows one notices a lack of sharpness when recreating the salient objects of the input with the method in [59]. Finally the last column shows results of [15]. One can observe that for MSLG the visual results are in general comparable to those of [15] while providing a local parametric model. The results of [15] are obviously excellent. But once again, in this kind of method, the algorithm ends up copying very large parts of the input. To illustrate this we represent our synthesis results and the ones obtained with [15] using the *position and synthesis maps* in section 2.4 of Chapter 2. The synthesis map permits to identify the tendency of an algorithm to generate verbatim copies and to visualize from which regions of the input texture are sampled the patches. To compute these synthesis maps we used the PatchMatch algorithm [3]. In Figure 3.7 one can observe that in general the synthesis maps associated to the results of MSLG are more "noisy" than those associated to the results of [15]. Also in the synthesis maps associated to [15] larger continuous zones are identified. This corresponds to the verbatim copies produced by the method. It is important to notice, that in both cases, for some texture examples in the synthesis maps there are some dominant colors represented in the synthesis maps. That reflects the discussion of Section 2.3 where we said that the spread of the input sample is not being

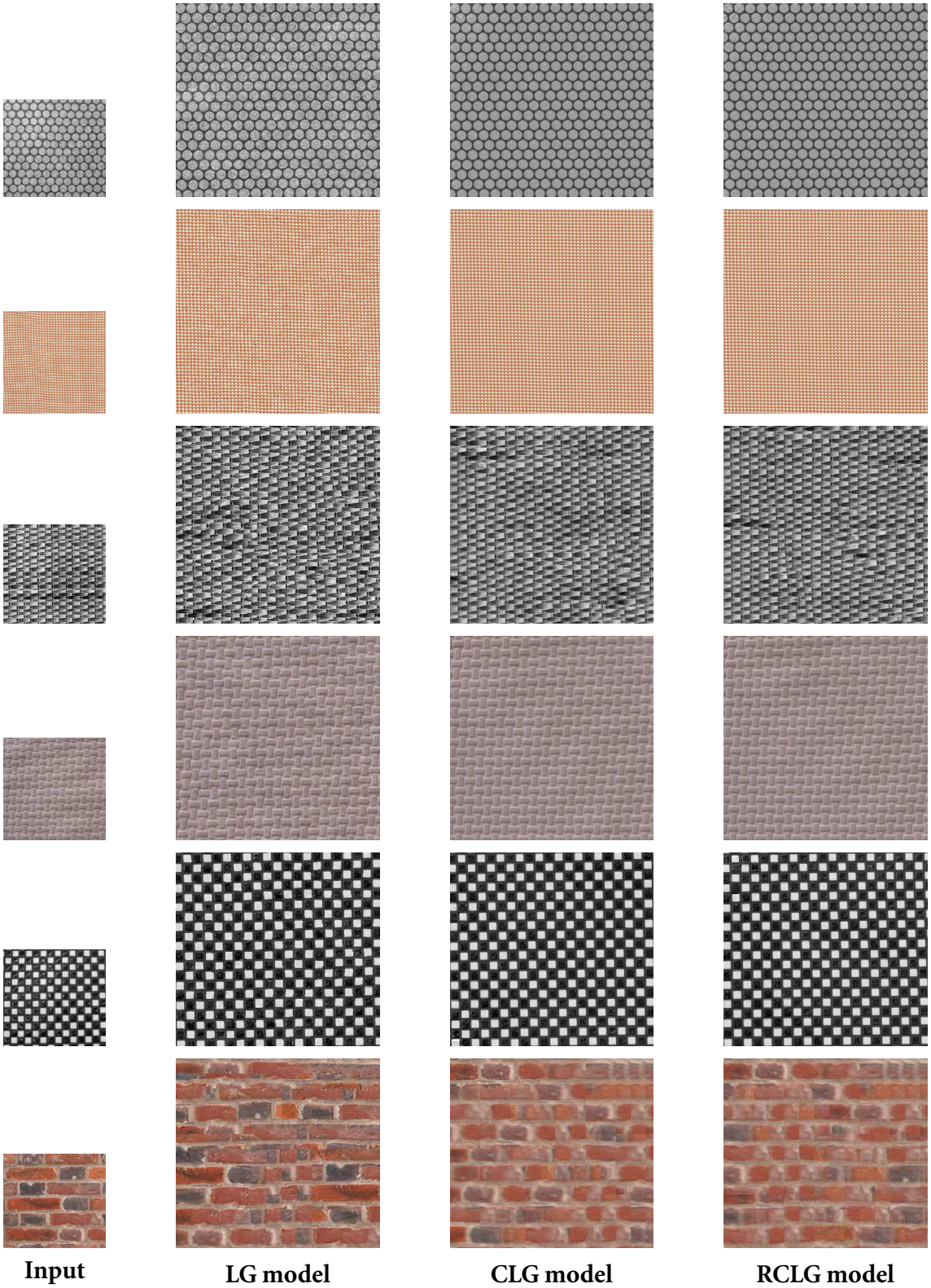


Figure 3.4 – *Patch model comparison.* From left to right: texture sample, synthesis result using LG, CLG and RCLG. No quilting technique was applied to stitch together the simulated patches for the three presented models. The parameters used for all examples are $n = 40$, $m = 30$ and $o = 0.5n$.

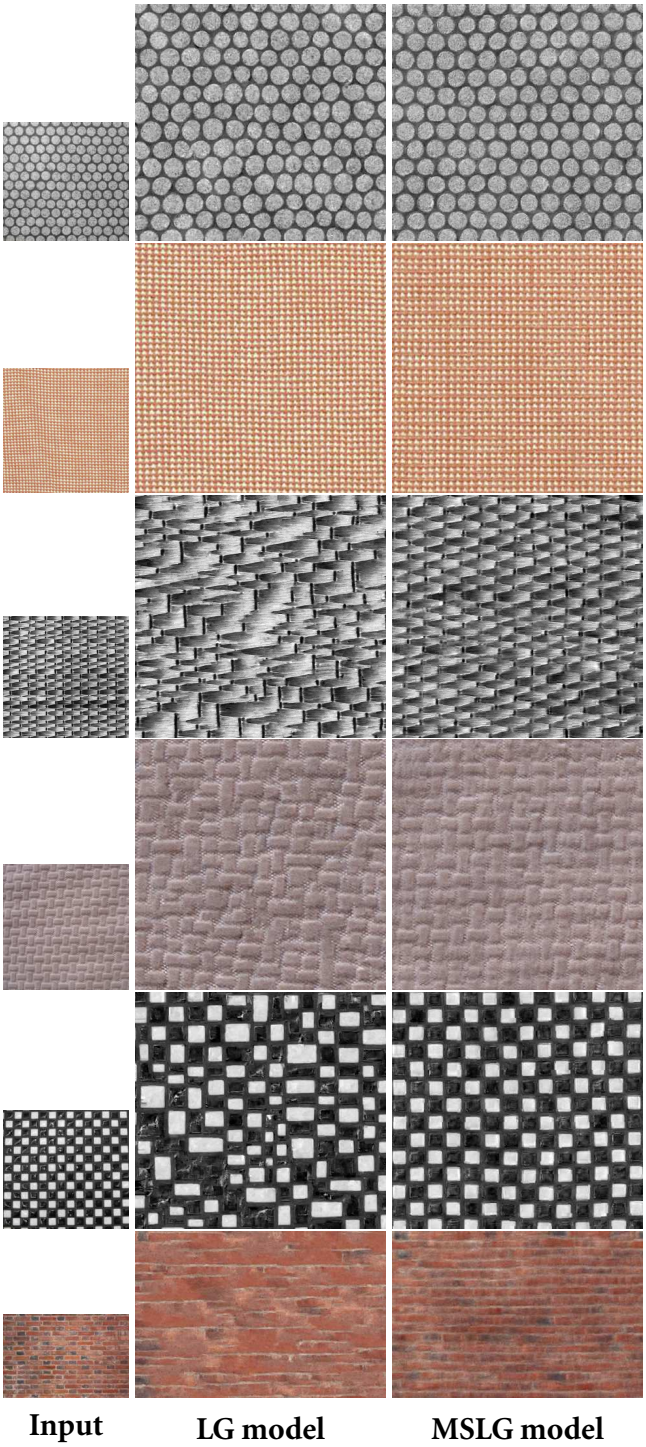


Figure 3.5 – *Comparison of LG to MSLG.* From left to right: texture sample, synthesis result using LG, and MSLG. The parameters used for these examples are $n = 20$, $m = 50$ and $o = 0.5n$. For the multiscale synthesis results the number of scales K used is $K = 3$. For both methods we used the same seed patch for each texture sample.

respected. For example, in the fifth row of Figure 3.7, the input sample is not stationary (there is a change of luminosity). Both methods fail in reproducing this. They tend to stay in one region (the darker one in this particular case) and may lead to garbage growing. This is well represented with the synthesis maps where green is dominant for the example in row number five. Combining MSLG to [50] used as post-processing significantly reduces this effect, although [50] has its own limitations.

3.5.3 Influence of the parameters

In Figure 3.8 and 3.9 the influence of the parameters is illustrated. There are four of them: the patch size $n \in \{10, 20, 40\}$, the number of neighbours $m \in \{10, 30, 50\}$, the overlap size $o \in \{0.25, 0.5\} \times n$ and the number of scales used $K \in \{1, 2, 3\}$.

Influence of the patch size The synthesis results are very sensitive to the patch size, in particular for macro textures that have details at different scales. Figure 3.8 clearly shows that if the patch size is too small then the synthesis will fail. Using the multiscale approach strongly reduces the dependency of the method on the patch size. For all examples shown in this paper, the multiscale method using patches of 20×20 pixels was enough to guarantee correct synthesis results. For the one scale version the patch size used should have been much larger to produce convenient results, on the cost of reducing the variability of the Gaussian models and even creating verbatim copy effects.

Influence of the number of neighbours This parameter corresponds to the number of patches used to estimate the patches' Gaussian distribution. As has been discussed in Section 3.2 the value of m in general should not be too small (> 5) or too large (< 50) to avoid patches of variance null or too big. These values are not general for all textures. The choice of m is linked to the amount of self-similarity in the image. Thus when m is too large the Gaussian sampling will blur up the image. In the experience of Figure 3.8 one can see that when $m = 150$ this leads to a texture which is too regular compared to the input sample. Otherwise since the sample texture in Figure 3.8 has many self similar patches the value of m can be large enough. The choice of m is a compromise between a copy-paste strategy and the risk of a blurry texture reconstruction.

Influence of the overlap size For the nearest patches only the overlap areas are compared. This implies that the variance of the model estimated on that set of patches will be controlled only on the overlap region, thus allowing more variety in the remaining pixels of the patch. If that region is not big enough then the complementary region of the overlap will not be correctly modeled, since outliers can be considered in the set of patches. In Figure 3.8 two overlap cases are considered: a quarter of the patch side and half the side patch size. This influence is more noticeable in the columns for $n = 20$ and $n = 40$.

Influence of the multiscale process In Figure 3.9 the results show that using a single scale for a fixed patch size is not enough to reproduce faithfully the global structure of the input sample for the three texture examples. To achieve satisfying results for a single scale a larger side patch size should be considered. Still this would lead to the limitations mentioned previously. Furthermore when the number of scales is increased the global arrangements are recovered as expected. This fact can be checked in the three examples of Figure 3.9. They also put in evidence how using a simple average of the values along the overlap area of the coarsest scale is sufficient to deal with the overlapping

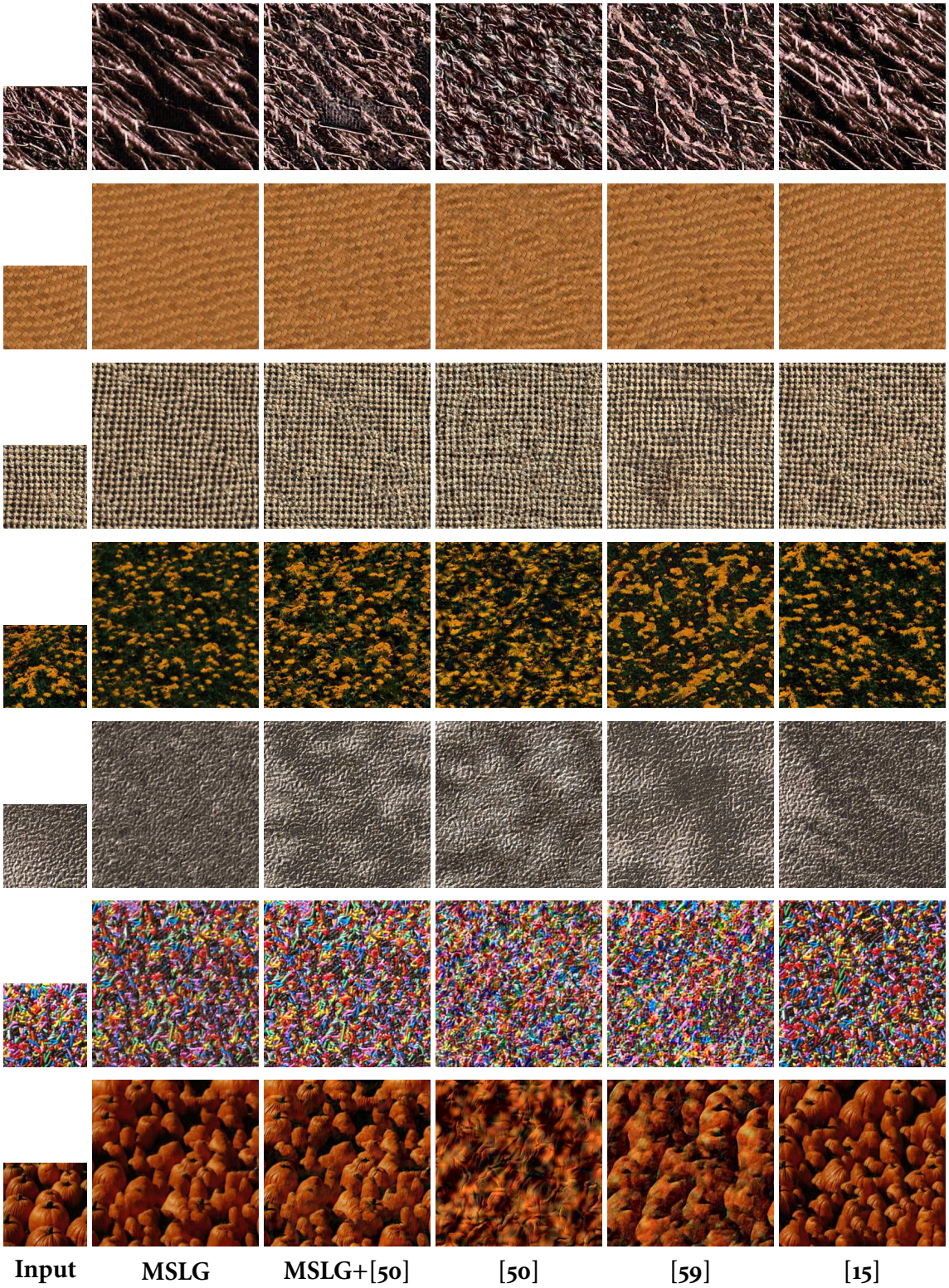


Figure 3.6 – *Comparison to several texture synthesis algorithms.* From left to right: input sample, MSLG, MSLG combined to Portilla and Simoncelli [50], Portilla and Simoncelli[50], Tartavel et al. [59] (source: `http://perso.telecom-paristech.fr/~tartavel/research/jmiv14.php`) and Efros and Freeman [15]. For MSLG the parameters used are $n = 20$, $m = 20$ and $K = 2$. For [50] four scales and orientations were used. For [59] the patch size is 12 and the number of scales 3. For [15] the patch size used is 20.

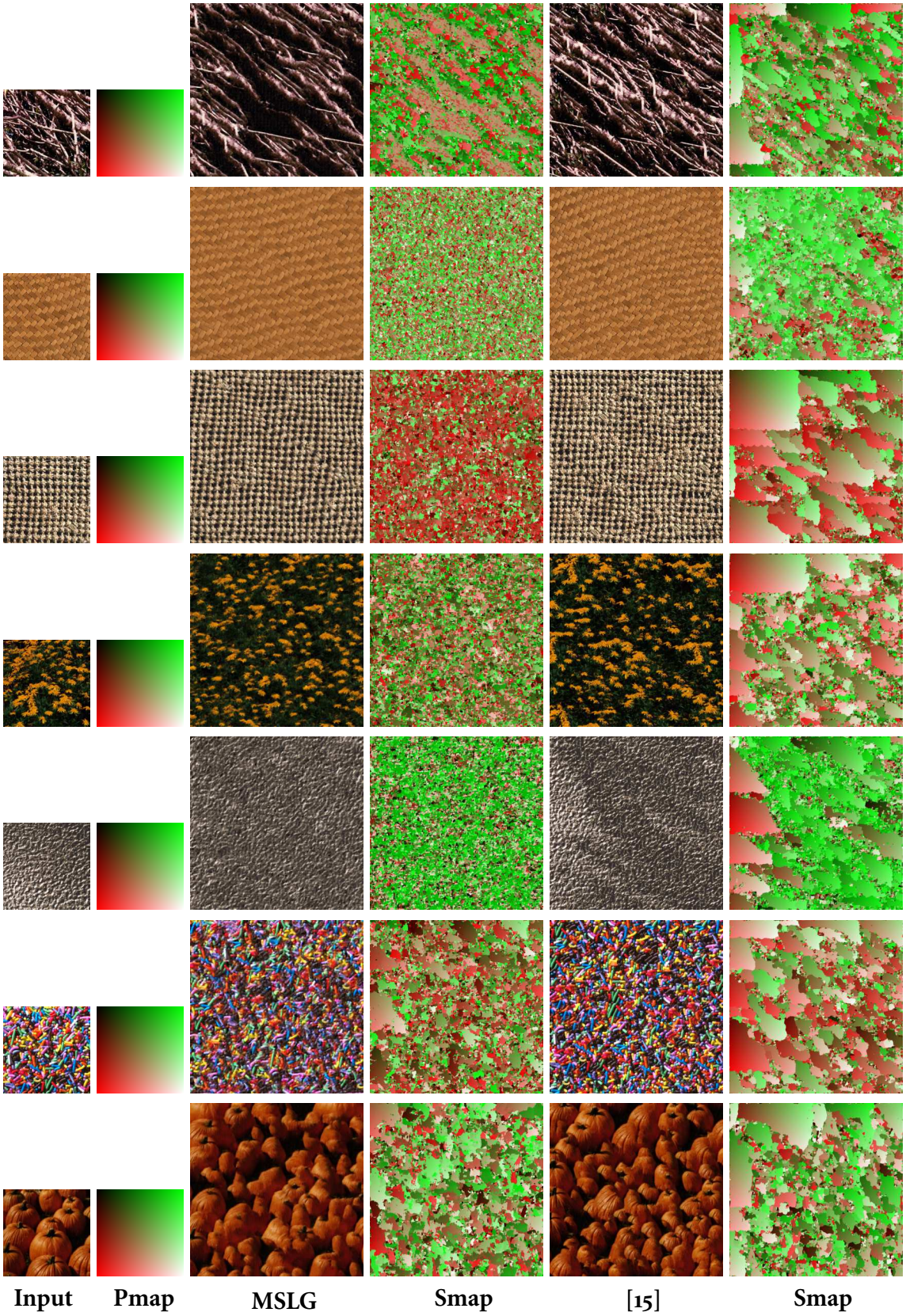


Figure 3.7 – *Innovation capacity of two texture synthesis algorithms.* From left to right: input sample, associated position map (Pmap), results of MSLG, associated synthesis map (Smap), results of [15] and associated synthesis map (Smap). The results of MSLG and [15] presented in this image are the same as the ones shown in Figure 3.6.

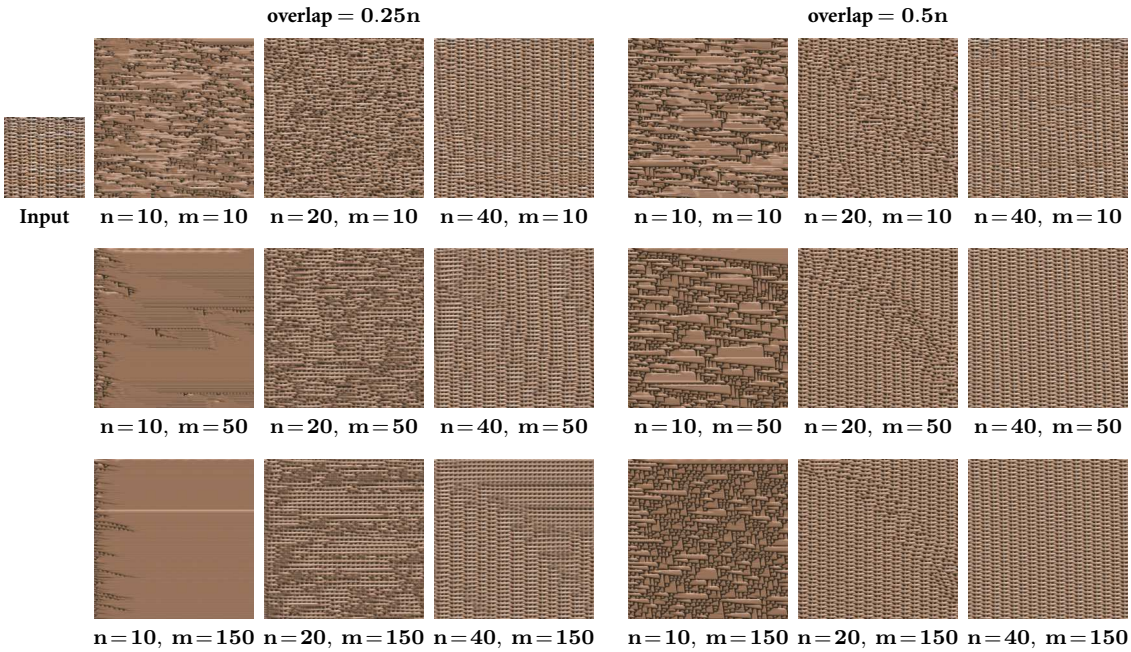


Figure 3.8 – *Influence of the choice of the patch size, number of nearest neighbours and overlap size.* For a given texture example several synthesis results are shown for different sets of parameters. For this experiment the MSLG method is used with $K = 2$ scales. Two columns of 3×3 results are presented. The left column corresponds to an overlap size $o = 0.25 \times n$ and the right column to an overlap size $o = 0.5 \times n$. For each column of 3×3, from top to bottom the number of nearest neighbours m takes the values $10, 50, 150$. From left to right the patch size n takes the values $10, 20, 40$. These results clearly show that for a fixed value of n and m using an overlap size $o = 0.25 \times n$ is not enough to determine a correct set of nearest neighbours. For a fixed patch size, for example $(n = 40, o = 0.5 \times n)$ one can observe that increasing the value of m smoothes the synthesis result.

patches. More complex quilting techniques are then avoided. The multiscale algorithm fosters a better respect of the spread (but it's not a guarantee).

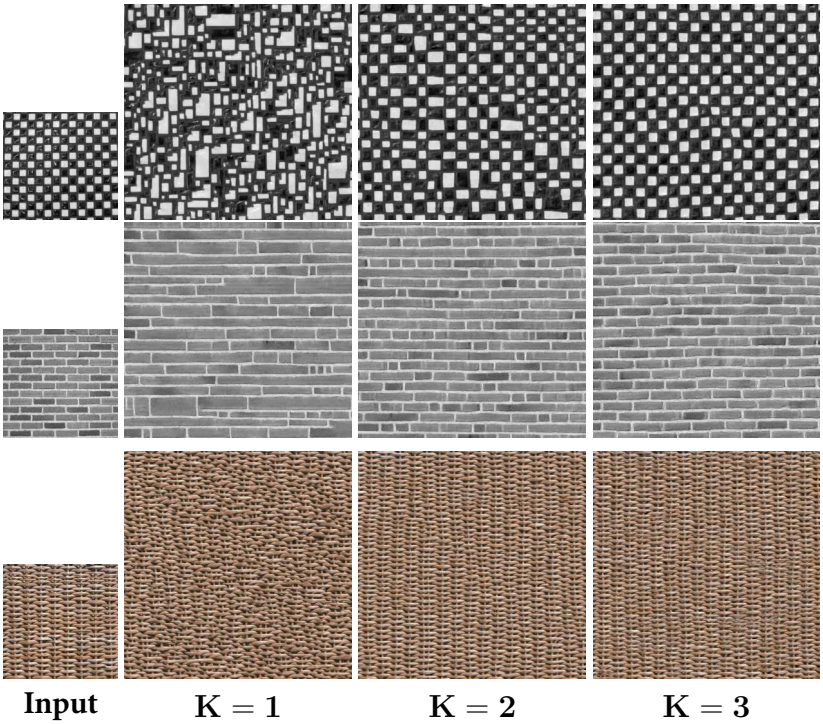


Figure 3.9 – *Positive effect of a multiscale procedure.* For each row from left to right: input sample, three synthesis results for $K = 1, 2, 3$ scales. For the three texture examples from top to bottom the parameters used were $(n = 10, m = 20), (n = 20, m = 20), (n = 20, m = 30)$. In the three examples the global arrangement for $K = 1$ is not respected while for $K = 2$ and $K = 3$ are correctly synthesized.

3.6 Conclusion

In this chapter, a local texture sampling method in the patch space using conditional Gaussian models was proposed. The motivation was to dispose of a patch stitching step by using a more robust local model for the texture. The Gaussian distribution of a patch was then conditioned to the values of its overlapping region. Two approaches were considered: CLG and RCLG models, and were compared to the local Gaussian model (LG). The results show that avoiding the stitching step is possible when dealing with periodic and pseudo-periodic textures. For more complex textures the conditional models are less performing and are fast limited by the size of the texture input sample. In general, the synthesis results using the conditional models were slightly smoother than the ones of the LG model.

The second contribution of this chapter was a multiscale texture synthesis algorithm using local Gaussian models (LG). In general, the experiments showed satisfying results for a wide variety of texture samples. Indeed, the patch-based approach conserves the local structures while sampling each patch from its Gaussian distribution, thus creating new patches that do not exist in the input sample. On the other hand the multiscale approach permits to synthesize the global arrangement of the salient structures of the input sample. The experiments also put in evidence that the use of Gaussian distributions, for some texture examples, have a tendency to slightly smooth the result

compared to the initial resolution of the sample. To overcome this effect the method was combined to Portilla and Simoncelli's algorithm [50] as a first solution.

Several aspects of this method are still open to elucidation, such as the way patches are compared, the adaptation of the number of neighbours to the patch being modeled and the conservation of the texture's global statistics.

4 Midway patch blending

In this chapter we analyze the problem of stitching patches together. One of the major challenges is stitching patches that differ considerably along the edges. Stitching methods used in texture synthesis usually fail in these cases, resulting in evident contrast changes between the patches. We suggest a novel method based on optimal transport theory that manages to merge both patches producing a smooth transition between them. This approach works in two steps: first by finding a common edge using the midway equalization methods, followed by propagating this edge into both images producing a smooth transition. We thoroughly evaluated the proposed method first by trying it directly on selected patches, followed by integrating it in the texture synthesis method of Efros and Freeman. Based on the results the proposed method proved to be a possible alternative to perform patch stitching in non-parametric patch-based methods.

4.1 Introduction

In this chapter we tackle the problem of stitching texture patches together. This problem arises repeatedly in the previous chapters where one of the steps of patch based texture synthesis methods is the fusion of overlapping patches. Given different patches of an input texture we want to stitch them together seamlessly. When the texture patches to stitch quite similar then existing techniques such as the quilting method in [15], Poisson editing [47], graph cuts algorithm [37] among others give a satisfactory result. Nevertheless when the patches to stitch are not that similar on the blending edge then these approaches do not solve conveniently the stitching step. This can be observed in the illustration of Figure 4.1. Here we address the question of deforming the content of the patches on a given curve in a way that both patches match on that curve. In other words we want to find a common signal such as the edges of the patches are transformed to match that signal. The problem is then composed of two steps. The first one consists in finding a common edge that is midway between the two edges. The second step consists in finding a way to progressively propagate this midway edge into both images. For the first step our method is strongly inspired in the midway equalization algorithm [12].

4.1.1 Antecedents

The Poisson editing method The Poisson editing method [47] is a method based on the manipulation of the image gradients for processing or combining images. It is used for seamless cloning, local illumination changes, seamless tiling *etc.*. In particular for seamless tiling the gra-

dients of the images to tile are given as a vector field and the Poisson equation is used to find the image gradient field that is the closest to the given vector field. This technique allows to follow the variations of each region to tile while removing the visible edges between regions. The drawback we encountered using this technique for the purpose of tiling two texture patches is the fact that when propagating the seamless tile, the Poisson equation decays very fast and for some cases, even if the "real edge" disappears since the propagation is quickly attenuated a visual effect of edge still remains (see Figure 4.1).

The quilting method The quilting method presented in [15] is a procedure to stitch a new patch in the sequentially built output texture by computing an optimal boundary cut between the patch and the synthesis area thanks to a linear programming optimization. This optimal boundary is the path on which the cumulative error between both patches is minimum. This technique presents very good results when the patches to tile are close to each other on their overlapping area. Nevertheless when this is not the case this optimal boundary is quite noticeable (see Figure 4.1).

Adaptive stitching methods Another work where texture merging is mentioned is in [37]. In this work Kwatra et al. propose to synthesize a new texture by copying irregularly shaped patches from the sample image into the output image. Once a candidate rectangular patch is selected an optimal portion of it is selected yielding an irregular shaped patch. The portion of the patch to copy is determined by using a graph cut algorithm, where the idea is to find out which patch each of the pixels in the overlapping area (of two patches A and B) come from. This amount to seek the minimum cost cut of the graph that separates the node A from the node B [19]. This method can work to merge patches. The method proposed in [37] proposes to use a graph-cut formulation that uses α-expansions [5] in order to take into account the previous seams and update them to deal with overlaps of more than two patches. They also propose an image merging technique using the same approach. Given a source image A and a target image B the idea is to constrain some pixels of image B where the image A will be inserted. Then the graph-cut algorithm finds the best seam that goes through the remaining unconstrained pixels of B setting the seam between both images. The results observed are satisfactory as long as both images have some common background. Otherwise, although the seam computed is optimal it will remain visible.

Our contribution In this chapter we suggest an original method to merge two texture patches along a common curve (that we call the blending curve). The problem can be seen as finding a common one dimensional signal which is midway, up to a minimal deformation, between two one dimensional signals denoted s_1 and s_2. The signals under construction are nothing but the restriction of the patches p_1 and p_2 to the blending curve. Afterwards the deformations sending s_1 and $_2$ to their midway signal are progressively propagated inside the corresponding patches. The signal registration step is inspired in the midway equalization method [12]. For this we considered that the two signals s_1 and s_2 were probability distributions on which we apply a geometric deformation derived from the midway equalization [12]. The common intermediary signal s is thus defined as the underlying *histogram* of the harmonic mean of the cumulative functions of s_1 and s_2. We noticed that the behaviour of the registered signals using the harmonic mean of the cumulative functions not always gives as result the intuitive registration one could expect. The signals are well registered but the intermediary signal can sometimes be "unnatural" and this could complicate the propagation step. For the propagation step two alternatives are considered. The first one consists in linearly propagating the deformations. The second one considers convex combinations of two

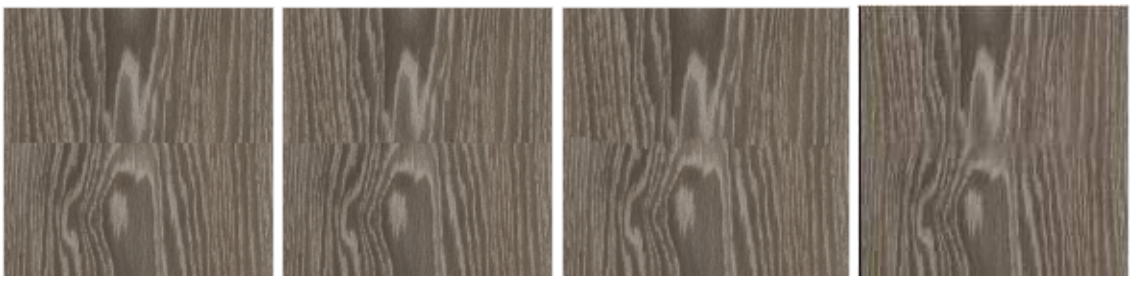


Figure 4.1 – Teaser. From left to right: input, input blended with Poisson editing [47], input blended with quilting [15], input blended with midway blending.

inverse cumulative signals: the midway signal and the signal on the curve up to which the deformation is propagated. The first alternative yields smoother results but is problematic regarding the normalization step of the signals (to impose the same mass). The second option avoids the normalization issues yet the propagation is more brutal and noticeable than the linear alternative. Results are shown for both options.

This chapter is organized as follows. Section 4.2 recalls the midway equalization algorithm [12]. In section 4.3 the *midway blending* technique is introduced. At first a description on a simple blending curve is given. It is followed by a generalization of the method. A description of the algorithm is provided as well as a discussion on the extension to a color version. As a final contribution of this section we describe how to integrate this method to a simple and efficient texture synthesis method. In section 4.4 experimental results are shown organized in two parts. The first one compares the midway blending method to [15] and [47]. The second part compares synthesis results of [15] to the same synthesis method using the *midway blending*. Conclusions are given in section 4.5.

4.2 Optimal transport - midway equalization algorithm

The theory of optimal transport [61] has been widely used in computer vision as a metric between statistical features [54]. This theory takes into account the spatial location of the density modes and provides a transport plan between them. Depending on the probability densities that are being transported, the transport plans can be very irregular. In [17] the authors propose a variational formulation to compute a regular transport map between two empirical densities. They show that the use of this regularized optimal transport improves visually the results for color image transfer.

The theory of optimal transport has been used in several applications, namely texture synthesis and texture mixing [53, 34, 64, 18], histogram equalization [12, 24], color normalization [13] and color transfer [12, 52, 51, 33] among others. We follow by describing with more details the histogram equalization method [12] which is strongly used in the remaining of the chapter.

Delon presented in [12] an image equalization method. The comparison of two images is always better when both images have the same dynamic ranges and luminance. For this given two images I_1 and I_2 a contrast change is applied obtaining two new images with the same gray or colour intensity histogram. It is desired that the common histogram stands midway between the histograms of I_1 and I_2 staying as close as possible to both of them avoiding to follow one of them.

Let $I : \Omega \to [0, 255]$ be an image and h the distribution of its values, i.e. the histogram of I.

The cumulative histogram of I is the increasing function $H : [0, 255] \to [0, 1]$ defined as

$$H(x) = \int_0^x h(t)dt. \tag{4.1}$$

For a given function $\phi : [0, 255] \to [0, 255]$ continuous and strictly increasing the cumulative histogram of the image $\phi(I)$ is $H \circ \phi^{-1}$. Thus, applying the change of contrast ϕ_1 and ϕ_2 defined in (4.2) to the images I_1 and I_2 respectively transports the respective cumulative histograms H_1 and H_2 to a common one H. In the definition (4.2) H^{-1} is the inverse function of H. In all of the paper every instance of H^{-1} will refer to the inverse as a function.

$$\phi_i = H^{-1} \circ H_i, \; i = 1, 2 \tag{4.2}$$

In other words the principle of histogram equalization methods is to apply two contrast functions ϕ_1 and ϕ_2 as in (4.2) to the images I_1 and I_2 to bring I_1 and I_2 to have the same histogram H. There are several ways to determine this common histogram H. Among them the classical equalization method where $H = id$, the specification method where the common histogram is equal to one of the images cumulative histogram $H = H_1$ (or $H = H_2$), taking the mean of both histograms $H = \frac{H_1+H_2}{2}$. All the previously mentioned methods are in general unsatisfactory. In the first method it might be to drastic to flatten both histograms (if these are far from being flat), the quantization noise is enhanced with this change of contrast. The second method is suitable if the dynamic of both histograms is similar otherwise the one that is modified may try to follow inconsistently the dynamic of the specified image. The third mentioned method tends to create unexisting structures in the resulting images. In [12] the author proposes to use the midway histogram H as the harmonic mean between the cumulative histograms H_1 and H_2 where

$$H^{-1} = \frac{H_1^{-1} + H_2^{-1}}{2} \tag{4.3}$$

and then the contrast changes involved are defined as

$$\phi_1 = \frac{id+H_2^{-1}\circ H_1}{2} \quad \text{and} \quad \phi_2 = \frac{id+H_1^{-1}\circ H_2}{2}. \tag{4.4}$$

In this way the equalized images have the same gray level distribution while preserving as much as possible their initial gray level dynamics. The midway histogram H is in fact the mid point of the geodesic in the Wassserstein distance [61] between the distributions H_1 and H_2.

4.3 Midway blending

We extended the midway equalization method previously described to use it as a blending method for texture patches. For this we propose to deform the edges on which the patches are stitched in the spirit of [12]. We will consider the signals on these edges as *histograms*.

Let $I_1 : \Omega \to [0, 255]$ and $I_2 : \Omega \to [0, 255]$ be two images. Here to illustrate the blending method we consider that the images are blended along a horizontal edge as illustrated in Figure 4.2. These edges will be referred to as the *blending edges*. Let $s_1 : \omega \to [0, 255]$ be the one dimensional edge of I_1 and $s_2 : \omega \to [0, 255]$ the one dimensional edge of I_2 with ω an interval $[a, b]$. We refer to $h_1 : \omega \to [0, 1]$ and $h_2 : \omega \to [0, 1]$ as the one dimensional *histograms* of s_1 and s_2 respectively. These *histograms* are nothing but the normalized one dimensional signals as defined in (4.5). The

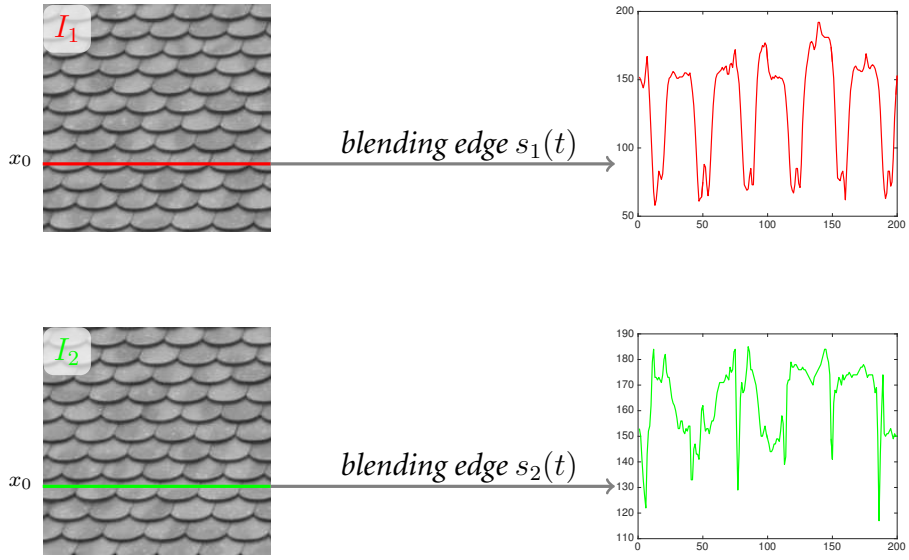


Figure 4.2 – Illustration of the *blending edges*. The two images to blend are represented by I_1 and I_2 and the edges on which the *midway blending* is applied are s_1 and s_2.

cumulative histograms H_1 and H_2 are then defined as in (4.1) for h_1 and h_2 respectively.

$$h_i(x) = \frac{s_i(x)}{\int_a^b s_i(t)dt} \tag{4.5}$$

The deformations we aim at will expand and/or contract the two signals on the *blending edges* in order to remove the visible edge between I_1 and I_2. We denote these two functions as $\phi_1 : \omega \to \omega$ and $\phi_2 : \omega \to \omega$ strictly increasing such as

$$H_1 \circ \phi_1 = H_2 \circ \phi_2 = H,$$

where the common cumulative histogram H is *midway histogram* defined in (4.3). The involved functions ϕ_1 and ϕ_2 are then defined as

$$\phi_1 = \left(\frac{\mathbf{I}+H_2^{-1}\circ H_1}{2}\right)^{-1} \quad \text{and} \quad \phi_2 = \left(\frac{\mathbf{I}+H_1^{-1}\circ H_2}{2}\right)^{-1}.$$

Let $\tilde{H}_1$ and $\tilde{H}_2$ be the deformed cumulative signals after applying ϕ_1 and ϕ_2 respectively

$$\tilde{H}_i(x) = H_i(\phi_i(x)),\ i = 1, 2. \tag{4.6}$$

where deriving (4.6) we obtain the deformed histograms

$$\tilde{h}_i(x) = h_i(\phi_i(x))\phi_i'(x), \forall x \in \omega,\ i = 1, 2$$

V
and the respective re-normalized deformed signals are

$$\tilde{s}_i(x) = \bar{m}\tilde{h}_i(x), \forall x \in \omega,\ i = 1, 2$$

where $\bar{m} = \frac{m_1+m_2}{2}$ with $m_i = \int_a^b s_i(t)dt$ for $i = 1, 2$. In this way $\tilde{s}_1$ and $\tilde{s}_2$ have the same mass.

Once the signals s_1 and s_2 are registered, i.e. the two images coincide on the blending edge, the deformations ϕ_1 and ϕ_2 have to be propagated. Otherwise we will create new discontinuities between the deformed signal $s_i,\ i = 1, 2$ and its corresponding image $I_i,\ i = 1, 2$. To avoid these inconsistencies we propagate the deformations ϕ_1 and ϕ_2 on a small part of the corresponding images I_1 and I_2. This propagation is done progressively to attenuate gradually the deformation on a portion of the image that we call the *propagation band* (see Figure 4.3). A first approach, that seems natural, is to use a linear propagation. That means that for each image $I_i,\ i = 1, 2$ the midway deformation $\phi_i,\ i = 1, 2$ is gradually attenuated as defined in (4.7).

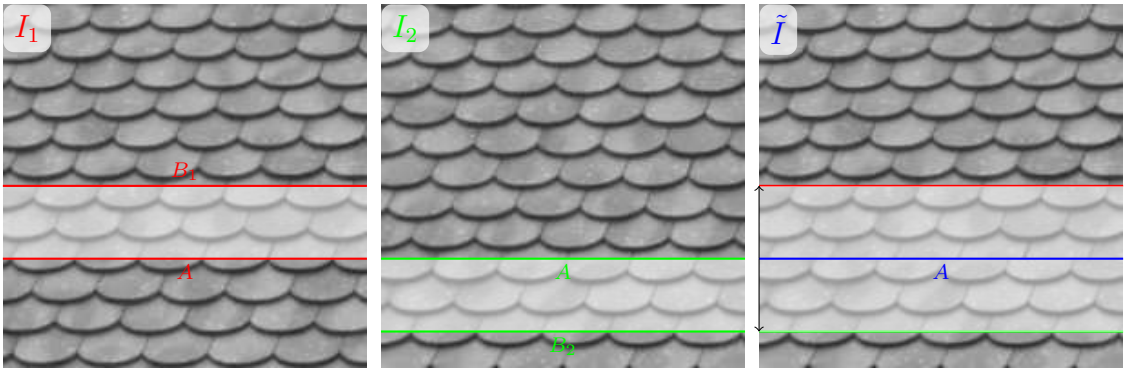


Figure 4.3 – Illustration of the *propagation band* of width W. Left: image I_1. The red rectangle marks out the propagation band of I_1. The propagation direction is from bottom to top from A to B_1. The edge in A represents the *blending edge* of I_1. Center: image I_2. The green rectangle marks out the propagation band of I_2. The propagation direction is from top to bottom from A to B_2. The edge in A represents the *blending edge* of I_2. Right: output image $\tilde{I}$ where the upper part is the corresponding portion in I_1 and the lower part is the corresponding portion of I_2. The blue edge is the registered edge that matches perfectly after applying the midway deformations.

Let $\Omega = [1, M] \times [1, N]$ be the rectangular domain of the images I_1 and I_2 and to illustrate the *blending edges* let us suppose that $s_1(y) = I_1(x_0, y),\ y \in [1, N]$ and $s_2(y) = I_1(x_0, y),\ y \in [1, N]$ (see Figure 4.2). We denote by W the width of the band on which we want to extend progressively the linear deformation. We denote by $\tilde{I}_i,\ i = 1, 2$ the images $I_i,\ i = 1, 2$ after deformation which are defined as

$$\tilde{I}_i(x, y) = I_i(\tilde{x}_i, \tilde{y}_i) J_i(x, y),\ \forall (x, y) \in \Omega,\ i = 1, 2$$

where $J_i(x, y) = f(x) + (1 - f(x))\phi_i'(y)$ and $(\tilde{x}, \tilde{y})$ are the deformed coordinates applying Φ_i

$$\begin{aligned} &(x, y) \to (\tilde{x}_i, \tilde{y}_i) = \Phi_i(x, y), \\ &\tilde{x}_i = x, \\ &\tilde{y}_i = f(x)y + (1 - f(x))\phi_i(y) \end{aligned} \tag{4.7}$$

The weight function f is defined as

$$f(x) = \begin{cases} 1 & \text{if } |x_0 - x| > W \\ \frac{|x_0 - x|}{W} & \text{if } |x_0 - x| \leq W \end{cases}.$$

Finally the images $\tilde{I}_1$ and $\tilde{I}_2$ are progressively re-normalized on the band on which the deformation was propagated to avoid any contrast change that may appear as an edge. The resulting images $\tilde{I}_1$ and $\tilde{I}_2$ are then

$$\tilde{I}_i(x,y) = \frac{\tilde{I}_i(x,y)}{m_i(x)} \left(f(x)m_i(x) + (1-f(x))\bar{m}\right) \tag{4.8}$$

where $m_i(x) = \int_a^b I_i(x,y)dy$. Nevertheless, we notice that propagating linearly the midway deformation yields some visual artifacts that comes of the normalization. Indeed, the normalization step is important since we want to transport signals of the same mass. However, propagating linearly the midway deformation does not guarantee that the signals after deformation conserve their mass. This normalization artifact can be seen in the example of Figure 4.4. A simple synthetic case is used to illustrate it where two binary images are to be blended. One can observe that the deformation achieved (in terms of shrinkage and dilatation) is coherent. Nevertheless, for the images $\tilde{I}_2$ and $\tilde{I}$ one can notice a shadowed area (marked with the red circles). This artifact in our opinion is the result of normalization step (see (4.8)). Thus a different normalization step should be considered if propagating linearly the midway deformations.

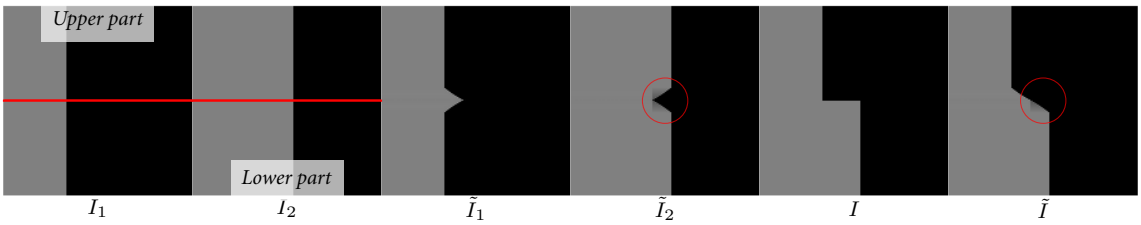


Figure 4.4 – Midway blending of two synthetic images using the linear propagation method. From left to right: image I_1, image I_2, deformed image $\tilde{I}_1$, deformed image $\tilde{I}_2$, image I composed of the upper part of I_1 (above the red line) and the lower part of I_2 (under the red line) and $\tilde{I}$ the blended version of I. Images $\tilde{I}_2$ and $\tilde{I}$ show the normalization artifact indicated within the red circles when propagating linearly the midway deformations.

The second propagation approach we considered for the midway deformations ϕ_1 and ϕ_2 consisted in a transition between the inverse of the cumulative histograms using a convex combination. We call it progressive midway. Let us suppose we applied the midway deformations to the signals s_1 and s_2 (blending edges of I_1 and I_2). Thus I_1 and I_2 match on their blending edge. Now for each image we aim at propagating the deformation on a small band (see Figure 4.3). For the propagation step each image is considered independently. Instead of working with the histograms we work with the inverse of the cumulative histograms. Thus for I_1 we consider H_A and H_{B_1} the cumulative histograms of the edges in A and B_1 shown in Figure 4.3. The edge in A is the blending edge after the midway deformation. The inverse cumulative histograms of the signals contained in the propagation band are a transition between H_A^{-1} and H_{B_1} using a convex combination (see (4.9)).

$$H_\delta^{-1}(y) = (1-\delta)H_A^{-1} + \delta H_{B_1}^{-1},\ 0 \leq \delta \leq 1 \tag{4.9}$$

The histograms h_δ are then recovered by differentiating the corresponding cumulative histogram H_δ and the signal s_δ yields from re-normalizing h_δ as in (4.10) where $m_A = \int_a^b \tilde{s}_1(y)dy$ and

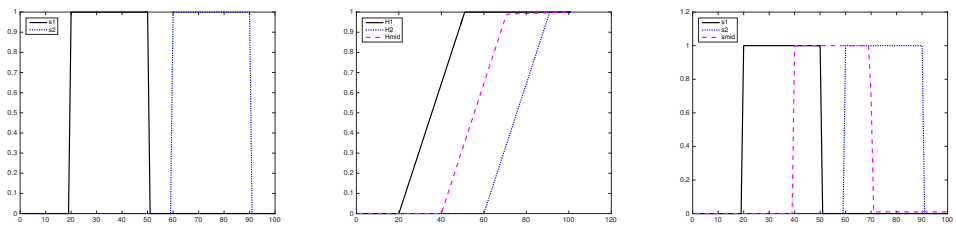


Figure 4.5 – Simple illustrative example 1. Left image: signals s_1 and s_2 to register. Middle image: cumulative histograms H_1 and H_2 and the midway histogram H. Right image: signals s_1 s_2 and the midway signal s. In this example one can observe that indeed s_1 and s_2 are translated to a signal that is halfway between them.

$m_{B_1} = \int_a^b I_1(x_0 - W, y)dy.$

$$s_\delta = h_\delta((1-\delta)m_A + \delta m_{B_1}),\ 0 \leq \delta \leq 1 \tag{4.10}$$

This approach deals with the normalization problems evoked for the linear propagation case. However one can observe in the results of Section 4.4 that the deformations can sometimes be too abrupt. In the next section we analyze the behaviour of the midway algorithm as a blending approach. We will consider the progressive midway propagation approach.

4.3.1 Interpretation of the method

Let us consider two signals s_1 and s_2 having the same mass as illustrated in Figure 4.5 where s_2 is simply a translation of s_1. We would expect the midway signal s_{mid} between s_1 and s_2 to be their translations to their halfway point. Indeed, that is exactly the result of applying the midway deformation to both signals as can be observed in Figure 4.5. As a second example let us consider another set of rectangular signals s_1 and s_2, where the flat zones values' are greater than zero. One would expect to have the same behaviour as in the example of Figure 4.5. However, the midway signal s is not a translation of s_1 and s_2 as one can observe in Figure 4.6. The cumulative histograms show that transporting the mass of both inputs signals s_1 and s_2 creates an additional intermediary flat zone in s. This redistribution of the mass is not intuitive. In this simple case a solution would be to substract the minimum value to both signals s_1 and s_2 before their registration. This is possible here since s_1 and s_2 have the same minimum value. A third example is shown in Figure 4.7 where once again the behaviour of the midway deformation is not intuitive. In Figures 4.8 and 4.9 we consider two signals s_1 and s_2 having different mass. Once again one can observe that if the values of the flat zones are greater than zero then the deformed signals have and additional flat zone yielding unnatural merging results. In Figures 4.10 - 4.14 five blending results are shown for synthetic images. The propagation step is done using the progressive midway approach. The results obtained are coherent with the examples shown in the previous Figures 4.5 - 4.9.

We conclude from these experiments that applying the *midway blending* does not always yield an intuitive deformation to blend two images. In Section 4.4 we show some results on real texture images where one can observe that it works for several textures and fail for others. In general when blending texture images these "unnatural" behaviour are less evident.

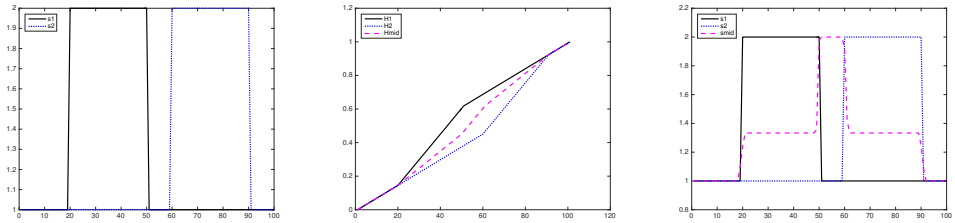


Figure 4.6 – Simple illustrative example 2. Left image: signals s_1 and s_2 to register. Middle image: cumulative histograms H_1 and H_2 and the midway histogram H. Right image: signals s_1 s_2 and the midway signal s. In this example one can observe that the midway signal is not the local translation of s_1 and s_2 that one would expect. The middle image shows that the resulting midway histogram H (i.e. the harmonic mean of H_1 and H_2) has an additional slope with respect to H_1 and H_2. This means that the midway signal s has an additional intermediary flat zone.

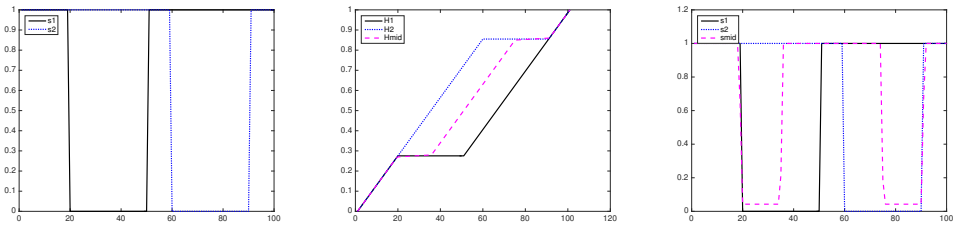


Figure 4.7 – Simple illustrative example 3. Left image: signals s_1 and s_2 to register. Middle image: cumulative histograms H_1 and H_2 and the midway histogram H. Right image: signals s_1 s_2 and the midway signal s. In this example one can observe that once again the deformation is not the expected translation. The signal is "split" in two and has an additional mode.

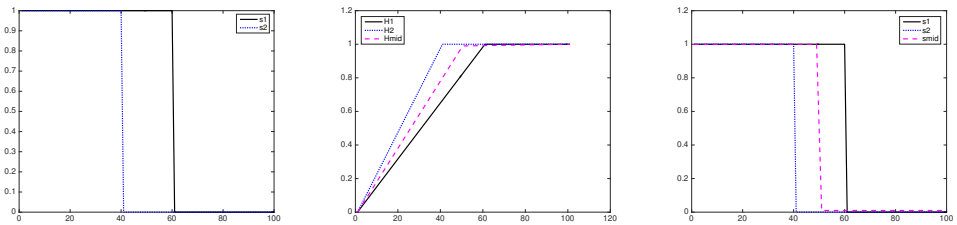


Figure 4.8 – Simple illustrative example 5. Left image: signals s_1 and s_2 to register. Middle image: cumulative histograms H_1 and H_2 and the midway histogram H. Right image: signals s_1 s_2 and the midway signal s. This is the case of two signals with different mass. After renormalizing the *midway* signal is a translation of s_1 and s_2 to a half way signal.

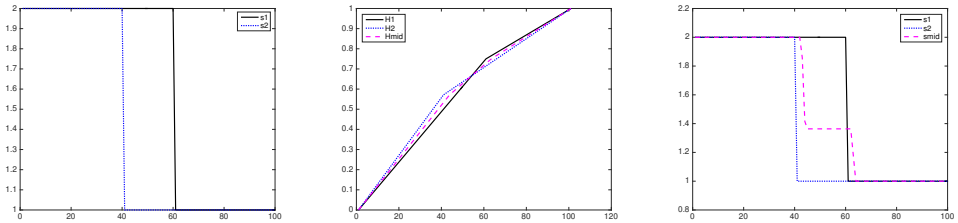


Figure 4.9 – Simple illustrative example 4. Left image: signals s_1 and s_2 to register. Middle image: cumulative histograms H_1 and H_2 and the midway histogram H. Right image: signals s_1 s_2 and the midway signal s. This is the case of two signals with different mass. The *midway* signal is not the intuitive result one expects. The redistribution of the mass is as shown in the left image with an extra intermediary flat zone.

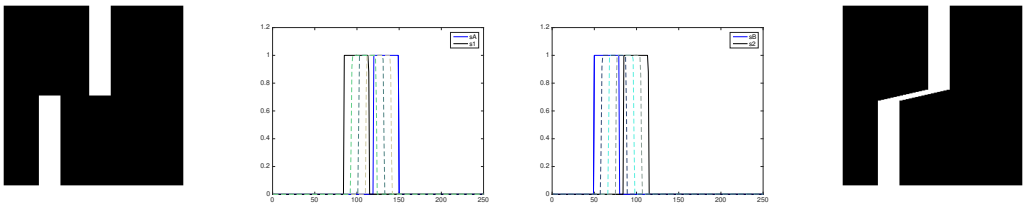


Figure 4.10 – This example illustrates a *midway blending* of two images I_1 and I_2 for a "successful case". The propagation of the deformation on each image is done using the *progressive midway* method. From left to right: inputs to blend, the deformed signals on I_1 and I_2 respectively, the blended output. The deformation on the two images is a progressive translation to a halfway signal.

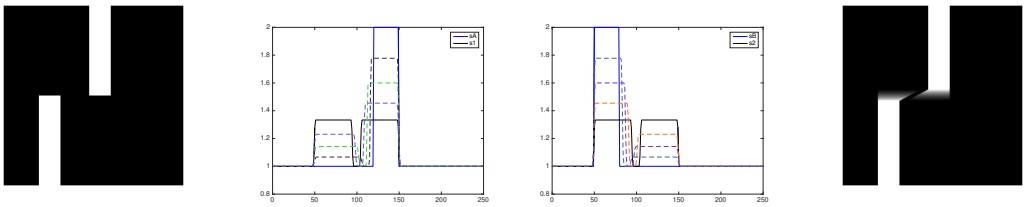


Figure 4.11 – This example illustrates a *midway blending* of two images I_1 and I_2 for a "failure case". The propagation of the deformation on each image is done using the *progressive midway* method on a band of 5 pixels of width. From left to right: inputs to blend, the deformed signals on I_1 and I_2 respectively, the blended output. As in the case of Figure 4.6 when the values of the grey levels of the flat zones are larger than zero an additional flat zone is created. The two middle images show how the successive deformation of the signals starting at s_1 (s_2) propagates to s_A (s_B). They show how the two initial flat zones are deformed to only one flat zone for each image. When the translation of the two initial bars is big than the mass is re-distributed in two equal modes as can be seen and this is what create the "passage" to zero from one bar to the other.

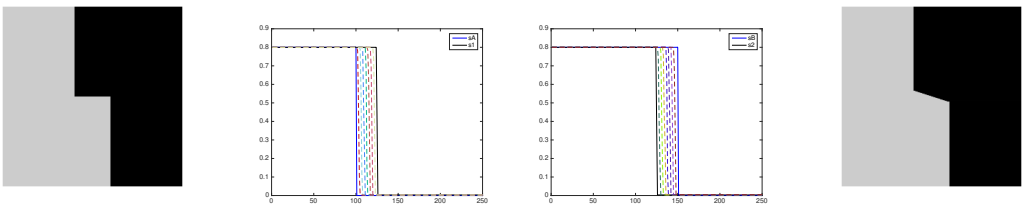


Figure 4.12 – This example illustrates a *midway blending* of two images I_1 and I_2 of different mass for a "successful case". The propagation of the deformation on each image is done using the *progressive midway* method on a band of 5 pixels of width. From left to right: inputs to blend, the deformed signals on I_1 and I_2 respectively, the blended output. The deformation on both images is a progressive translation from s_1 (s_2) to s_A (s_B).

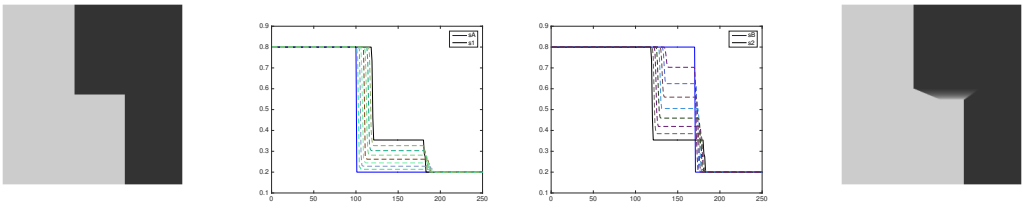


Figure 4.13 – This example illustrates a *midway blending* of two images I_1 and I_2 of different mass for a "failure case". The propagation of the deformation on each image is done using the *progressive midway* method on a band of 5 pixels of width. From left to right: inputs to blend, the deformed signals on I_1 and I_2 respectively, the blended output. The deformed signals once again creates additional flats zones and is the responsible of the unexpected dilatation of the bright zone in I_2.

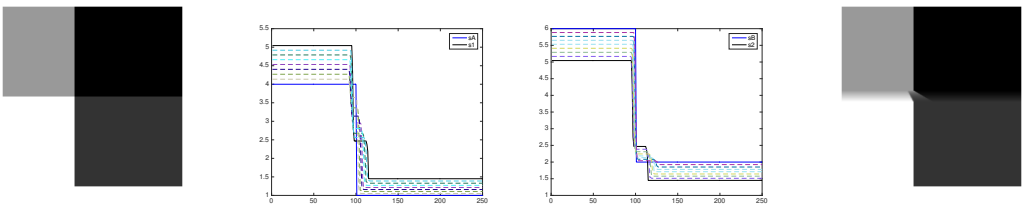


Figure 4.14 – This example illustrates a *midway blending* of two images I_1 and I_2 of different mass for a general case. The propagation of the deformation on each image is done using the *progressive midway* method on a band of 5 pixels of width. From left to right: inputs to blend, the deformed signals on I_1 and I_2 respectively, the blended output. Depending on the size of each flat regions and its values the resulting blended images can be as expected or not giving results similar to the ones obtained previously.

4.3.2 Midway blending algorithm

For the algorithm description let us suppose that I_1 and I_2 are two discrete images defined on $\Omega = \{1, \ldots, M\} \times \{1, \ldots, N\}$. The *blending edges* are now defined as $s_1(j) = I_1(x_0, j),\ j \in \{1, \ldots, N\}$ and $s_2(j) = I_2(x_0, j),\ j \in \{1, \ldots, N\}$. The respective *histograms* are then

$$h_i(j) = \frac{s_i(j)}{\sum_{k=1}^{N} s_i(k)},\ j \in \{1, \ldots, N\},\ i = 1, 2 \tag{4.11}$$

and the cumulative ones

$$H_i(j) = \sum_{k=1}^{j} h_i(k),\ j \in \{1, \ldots, N\},\ i = 1, 2. \tag{4.12}$$

A summary of the method is given in Algorithm 8, Algorithm 7 and Algorithm 9.

Algorithm 6: Pseudo inverse

Input: Function to invert H, x domain of H
Output: Inverse H^{-1}

```
k ← 2{re-sampling factor}
n ← length(H) {equal to length(x)}
for i = 1 to kn − 1 do
    z ← (i − 1)/(kn − 2)
    j ← Σ_{l=1}^{n} 1_{(H[l]<=z)}
    Ih[i] ← 1/(H[j+1]−H[j]) ((z − H[j])x[j + 1] + (H[j + 1] − z)x[j])
end for
for i = 1 to n do
    H^{-1}[i] ← Ih[1 + (i − 1)k]
end for
return H^{-1}
```

Algorithm 7: Midway histogram

Input: cumulative histograms H_1, H_2
Output: midway cumulative histogram H_{mid}

```
n ← lengthH_1 {equal to length(H_2)}
H_1^{-1} ← pseudo − inverse(H_1, {1, ..., n}) {see Algorithm 6}
H_2^{-1} ← pseudo − inverse(H_2, {1, ..., n}) {see Algorithm 6}
H_mid^{-1} ← (H_1^{-1} + H_2^{-1})/2
H_mid ← pseudo − inverse(H_mid^{-1}, {0, 1/(n − 1), 2/(n − 1), ..., 1}) {see Algorithm
6}
return H_mid
```

Algorithm 8: Midway registration of two signals

Input: Signal to register s_1 and s_2
Output: Midway signal s_{mid}

$n \leftarrow length(s_1)$ {equal to $length(s_2)$}
$m_1 \leftarrow \sum_{k=1}^{n} s_1(k)$
$m_2 \leftarrow \sum_{k=1}^{n} s_2(k)$
$h_1 \leftarrow s_1/m_1$ {1d *histogram* of s_1}
$h_2 \leftarrow s_2/m_2$ {1d *histogram* of s_2}
$H_1 \leftarrow zeros(n+1)$
$H_2 \leftarrow zeros(n+1)$
for $i = 2$ **to** $n+1$ **do**
  $H_1[i] \leftarrow H_1[i-1] + h_1[i-1]$
  $H_2[i] \leftarrow H_2[i-1] + h_2[i-1]$
end for
$H_{\text{mid}} \leftarrow midway(H_1, H_2)$ {see Algorithm 7}
$h_{\text{mid}} \leftarrow zeros(n)$
for $i = 1$ **to** n **do**
  $h_{\text{mid}}[i] \leftarrow H_{\text{mid}}[i+1] - H_{\text{mid}}[i]$
end for
$s_{\text{mid}} \leftarrow h_{\text{mid}}\left(\frac{m_1+m_2}{2}\right)$
return s_{mid}

Algorithm 9: Midway blending

Input: Images to blend I_1 and I_2, band width W, horizontal blending edge x_0
Output: Blended images $\tilde{I}$

$s_1[j] \leftarrow I_1[x_0, j]$ {get *blending edges* from I_1}
$s_2[j] \leftarrow I_2[x_0, j]$ {get *blending edges* from I_2}
$s_{\text{mid}} \leftarrow midway - registration(s_1, s_2)$ {see Algorithm 8}
$s_{B_1}[j] \leftarrow I_1[x_0 - W, j]$
$s_{B_2}[j] \leftarrow I_2[x_0 + W, j]$
$\tilde{I}_1 \leftarrow propagate(s_{\text{mid}}, s_{B_1})$ {propagation step defined in Section 4.3}
$\tilde{I}_2 \leftarrow propagate(s_{\text{mid}}, s_{B_2})$ {propagation step defined in Section 4.3}
$\tilde{I}[i,j] \leftarrow \begin{cases} \tilde{I}_1[i,j] & \text{if } i \leq x_0 \\ \tilde{I}_2[i,j] & \text{if } i > x_0 \end{cases}$
return $\tilde{I}$

4.3.3 Generalization of the midway blending

Let $I_1, I_2 : \Omega \to [0, 255]$ be two images. We consider $\gamma : \omega \to \Omega$ a parametric curve defined as $\gamma(t) = (x(t), y(t)),\ t \in \omega$ and $\omega = [a, b]$. Let $s_1 : \omega \to [0, 255]$ and $s_2 : \omega \to [0, 255]$ be the 1D *blending edges* defined as $s_i(t) = I_i(\gamma(t)),\ i = 1, 2$. Then the *histograms* h_1 and h_2 and the cumulative histograms H_1 and H_2 are defined as in (4.5) and (4.1). As explained in the previous

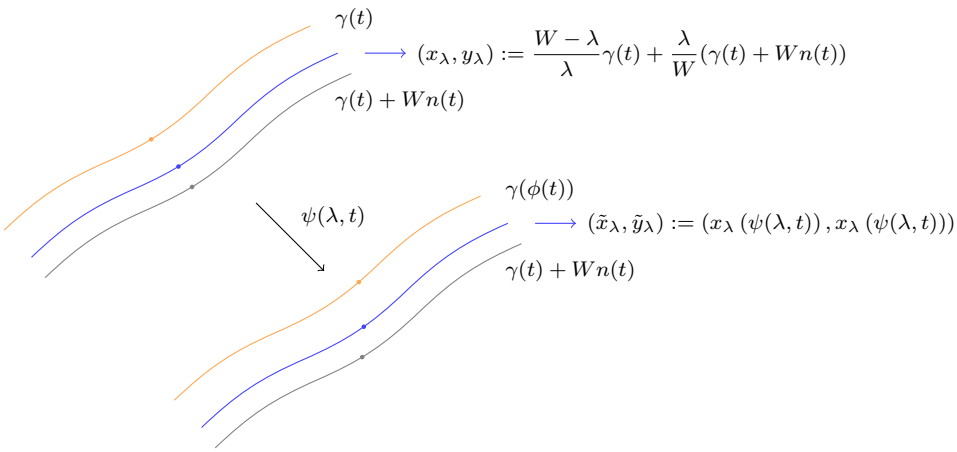


Figure 4.15 – Illustration of the linear propagation of the deformation ϕ on a band of width W. This figure shows how the three points in A are modified after applying the deformation $\psi(\lambda, t)$ yielding the three points in B. For the orange curve the point $(x(t), y(t))$ moves $(x(\phi(t)), y(\phi(t)))$. For the blue curve the point $(x_\lambda(t), y_\lambda(t))$ moves $(x_\lambda(\phi(\lambda, t)), y_\lambda(\phi(\lambda, t)))$. For the gray curve the point does not move.

section we aim at finding two functions ϕ_1 and ϕ_2 such as the new *histograms* $\tilde{h}_1$ and $\tilde{h}_2$ satisfy

$$\tilde{h}_i(t) = h_i(\phi_i(t))\phi_i'(t),\ i = 1, 2.$$

This implies then $\forall t \in \omega$

$$\tilde{I}_i(\gamma(t)) = \tilde{I}_i(x(t), y(t)) = \frac{\bar{m}}{m_i} I_i(x(\phi_i(t)), y(\phi_i(t)))\phi_i'(t)$$

where $m_i = \int_a^b s_i(t)dt,\ i = 1, 2$ and $\bar{m} = m_1 + m_2$.

We shall now consider all the curves $\gamma_{\alpha,\lambda},\ \alpha \in \{-1, 1\},\ \lambda \in [0, W]$ that are defined on a band of width $2W$ around γ. These curves are taken in the normal direction $n(t) = \frac{\gamma'(t)^\perp}{\|\gamma'(t)\|}$ and are defined as

$$\gamma_{\alpha,\lambda}(t) = \gamma(t) + \alpha\lambda n(t).$$

We assume that the considered curves and the width W are such that $\forall(\alpha, \lambda, t_1) \neq (\beta, \mu, t_2), \gamma_{\alpha,\lambda}(t_1) \neq \gamma_{\beta,\mu}(t_2)$ (see 4.15). Once again we consider two propagation approaches: the linear propagation and the progressive midway propagation. The first one, the linear propagation (see Figure 4.15), is defined as the deformation $\psi_i : \omega \times [0, W] \to \omega,\ i = 1, 2$

$$\psi_i(\lambda, t) = \frac{W - \lambda}{W}\phi_i(t) + \frac{\lambda}{W}t,\ \lambda \in [0, W],\ t \in \omega.$$

The resulting deformed images $\tilde{I}_1$ and $\tilde{I}_2$ are then

$$\begin{aligned}&\tilde{I}_i(x_{\alpha,\lambda}(t), y_{\alpha,\lambda}(t)) = \\ &= I_i(x_{\alpha,\lambda}(\psi_i(\lambda, t)), y_{\alpha,\lambda}(\psi_i(\lambda, t)))\left(\frac{W - \lambda}{W}\phi_i'(t) + \frac{\lambda}{W}\right),\end{aligned}$$

for $i = 1, 2$, and after re-normalization

$$\tilde{I}_i(x_{\alpha,\lambda}(t), y_{\alpha,\lambda}(t)) =$$
$$\frac{\tilde{I}_i(x_{\alpha,\lambda}(t), y_{\alpha,\lambda}(t))}{m_i(\alpha,\lambda)} \left((\frac{W-\lambda}{W})\bar{m} + \frac{\lambda}{W} m_i(\alpha,\lambda) \right),$$

where $m_i(\alpha,\lambda) = \int_\omega I_i(\gamma(u) + \alpha\lambda n(u))du$.

Let us consider the particular case of a circular arc $\gamma : \omega \to \Omega$. That is

$$\gamma(t) = (R\cos\theta(t), R\sin\theta(t)) = R(\cos(s), \sin(s)),\ t \in \omega$$

where $s = \theta(t) = \frac{\pi}{2}t, \omega = [0,1]$ and R is the radius of the circular curve. We consider the curve γ parametrized in the angles s. The normal $n(s)$ to the curve $\gamma(s)$ is defined as

$$n(s) = \frac{\gamma''(s)}{\|\gamma''(s)\|} = -\frac{\gamma(s)}{R} = -(\cos(s), \sin(s)),$$

where s denotes the arc length, i.e. $\|\gamma'(s)\| = 1$. The resulting images $\tilde{I}_1$ and $\tilde{I}_2$ are

$$\tilde{I}_i(R(\cos(s), \sin(s)) - \alpha\lambda(\cos(s), \sin(s))) =$$
$$\tilde{I}_i((R-\alpha\lambda)(\cos(s), \sin(s))) =$$
$$K_i(r) I_i\left((R-\alpha\lambda)(\cos(\psi_i(s)), \sin(\psi_i(s)))\right)\psi_i'(s),$$

where $K_i(r) = \left((\frac{W-\lambda}{W})\bar{m} + \frac{\lambda}{W} m_i(r)\right) / (m_i(r))$ and $m_i(r) = \int_0^{\pi/2} I_i(r(\cos\theta, \sin\theta))d\theta$.

We can verify that indeed for $\lambda = 0$ the deformation function is ϕ_i

$$\tilde{I}_i(R(\cos(s), \sin(s))) = K_i(R) I_i\left(R(\cos(\phi_i(s)), \sin(\phi_i(s)))\right)\phi_i'(s)$$

and for $\lambda = W,\ \alpha \in \{-1, 1\}$ the deformation function is the identity

$$\tilde{I}_i((R-\alpha W)(\cos(s), \sin(s))) =$$
$$K_i(R-\alpha W) I_i\left((R-\alpha W)(\cos(s), \sin(s))\right).$$

The second propagation method, the progressive midway, is as explained for the horizontal case: a convex combination of the inverse cumulative histograms of the blending edge and the second edge of the propagation area.

In practice we tested three curves: horizontal, vertical and circular arc. To simplify we transformed the vertical and circular case to a horizontal edge problem. A pseudo-code of the algorithm is provided in Algorithm 10. For this let us consider that I_1 and I_2 are two discrete images defined on $\Omega = \{1,\ldots,M\} \times \{1,\ldots,N\}$. The *blending edges* can be taken along a horizontal curve $\gamma : \{1,\ldots,N\} \to \Omega$ of length N defined as $\gamma[j] = (x_0, j)$ or along a vertical curve $\gamma : \{1,\ldots,M\} \to \Omega$ of length M defined as $\gamma[j] = (j, x_0)$ or along a circular curve $\gamma : \{1,\ldots,L\} \to \Omega$ of length L defined as $\gamma[j] = (M - x_0\cos\theta[j], N - x_0\sin\theta[j])$ where $\theta[j] = \left(\frac{j-1}{L-1}\right)\frac{\pi}{2},\ j \in \{1,\ldots,L\}$ and x_0 is the radius of the curve. The *blending edges* are then defined as $s_i[j] = I_i(\gamma[j])\ \ i = 1, 2$.

Algorithm 10: Generalized midway blending algorithm

1 **Input:** Images to blend I_1 and I_2, band width W, reference position x_0 of the blending curve
2 **Output:** Blended images $\tilde{I}$

```
if edge-type ≠ horizontal then
    I_1 ← T(I_1) {transform to a horizontal edge case}
    I_2 ← T(I_2) {transform to a horizontal edge case}
    x_0 ← t(x_0) {transform to the reference position of a horizontal edge case}
end if
Ĩ ← midway − blending(I_1, I_2, W, x_0) {see Algorithm 9}
if edge-type ≠ horizontal then
    Ĩ ← T^{-1}(Ĩ) {transform to the original edge case}
end if
return Ĩ
```

4.3.4 Colour midway blending

The first approach we considered for colour images was doing the transport on the grayscale image and then applying the same deformation to each channel. Thus no color artifact appears. Nevertheless for each colour channel the signals of both images do not match perfectly and an edge is still evident on the resulting colour image.

As a second alternative we considered treating each channel independently. This amounts to apply a midway blending deformation for each channel separately. In this case the edges do match perfectly but false colors may appear, each colour channel being processed independently from the other ones. Indeed, the colour components of the RGB color representation are strongly correlated. Thus, deforming each channel independently will not maintain this correlation. The solution is to work with a different color space where the components are less correlated.

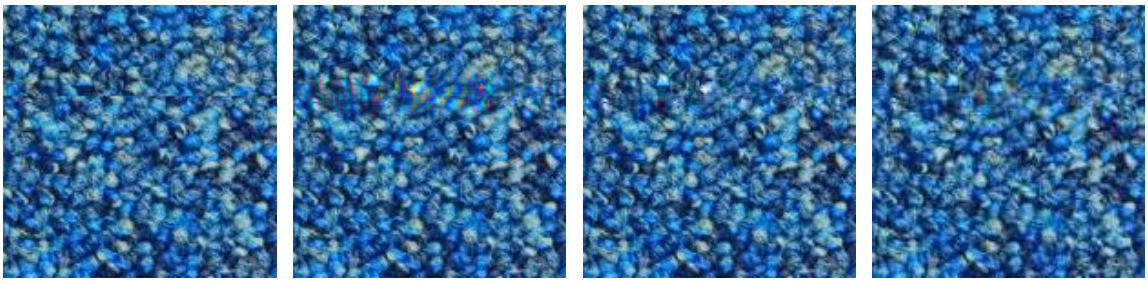
Instead of working with the RGB color spae we propose to use a DCT colour representation which is similar to the HSV colour space but easier to implement. Let us suppose a colour image $I : \Omega \to [0, 255]^3$ defined in the RGB space. Than the DCT representation of I is obtained from the RGB components applying the orthogonal linear map

$$T = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{6} & -2/\sqrt{6} & 1/\sqrt{6} \end{bmatrix}.$$

Notice that the DCT representation of I yields negative values in its second and third components (the first component is the average of the R, G and B channels up to a constant multiplier). Since the signals are considered as histograms and their cumulative functions are used, negative values are not desired. To avoid this after applying the orthogonal matrix T we substract its minimum from each channel, so that it has no negative values.

Another approach one could consider is the PCA colour space where the colour representation is not fixed and depends on the image. This approach yields a representation where the colour components are decorrelated. However when using a PCA colour space each image has a different diagonalization basis and the question that arises from this is what common basis should one use to apply the transport between signals? We observed that for natural images the DCT colour

representation yields components that are in average almost decorrelated. This representation is very close to the PCA colour space with the advantage that the same basis is used for all images. Nevertheless to our application the patches we aim at stitching come from the same texture image. Thus applying the PCA decomposition to both images yields an orthogonal transformation very similar to the one that we would obtain if the PCA decomposition was applied to each image separately. We compare the three approaches (RGB, DCT and PCA) in Figure 4.16.



Input RGB DCT PCA

Figure 4.16 – Different colour representations used to stitch two texture patches with the midway approach. From left to right: input, output using the RGB representation (notice the appearance of the colour artifacts along the stitching edge), output using the DCT representation and output using the PCA representation.

4.3.5 Texture synthesis

This work is originally motivated by patch based texture synthesis methods. Up to now several stitching techniques were used, among them the quilting method [15], graph cuts [37], Poisson blending [47], etc. However, these techniques do not yield an image deformation. They find the "best way" of putting together the different pieces. Methods [15] and [37] tend to find the best possible path in a given region without changing the content of the image. For instance in the example of Figure 4.1 the stripes are not placed one in front of the other. The visual impression of an existing edge between both images remains. A different approach would be to use the method to seamlessly edit image regions [47] to blend the patches. Still, one can notice that the edge is attenuated but remains evident and the stripes do not match. In this chapter we aimed at "matching these stripes". To that end we allow to stretch and/or dilate the images on their common edge as illustrated in Figure 4.1 where one can observe how the stripes "moved" and match.

In this section we detail the synthesis algorithm where the *midway blending* method is used. We shall compare three stitching techniques: the Poisson editing methods [47], the quilting method in [15] and the midway blending. In continuation, we shall compare two texture synthesis methods: one using the quilting technique [15] and another using the midway blending.

The image quilting method [15] computes a new texture by rearranging seamlessly patches of the input texture. The procedure to stitch a new patch in the sequentially built output texture is achieved by computing an optimal boundary cut between the patch and the synthesis area thanks to a linear programming optimization. This method is a sequential patch-based algorithm. I and $\tilde{I}$ denote the input and the output texture respectively. The output image $\tilde{I}$ is constructed patch by patch in a raster scan order. The goal of each iteration is to fill a patch of $\tilde{I}$ that is only partially defined on a region called overlap region. Note that there are three possible overlap regions: vertical overlap for the first row, horizontal overlap for the first column, and L-shaped

overlap everywhere else. To complete the patch P_A a patch P_B of I that matches P_A on the overlap region is randomly selected. An optimal boundary cut between P_A and P_B is then computed within the overlap region. This optimal boundary cut is used to construct the new patch P by blending P_A and P_B along the cut. We now want to compare the results of this texture synthesis method when replacing the stitching step by the *midway blending*. For details on the quilting method please refer to Annex A.2.3. The synthesis algorithm using the *midway blending* is detailed in Algorithm 11.

Algorithm 11: Texture synthesis

1 **Input:** Sample texture I, patch size n, overlap size o
2 **Output:** Synthesized texture $\tilde{I}$

```
Initialize Ĩ with a seed patch (randomly taken from I)
for each patch P_A in Ĩ do
    Select a compatible patch P_B ∈ I
    Construct P blending P_A and P_B using the midway blending algorithm (see
    Algorithm 9). Depending on the overlap type the (blending curve) can be a
    horizontal, vertical or circular curve. (see Algorithm 10)
    Replace P_A with P within Ĩ
    return Ĩ
end for
```

4.4 Experiments

4.4.1 Comparison of blending techniques

In this section we compare the blending methods that can be seen in Figures 4.17 to 4.20. Given two patches p_1 and p_2 we applied three different stitching techniques: the quilting method [15], Poisson blending [47] and the midway blending. For each texture example we blended the two patches along three different curves: horizontal, vertical and circular. For the quilting method the circular curve corresponds to the l-shape overlap. To compare the circular curve and the L-shape overlap we consider the same overlap size and the circular curve is taken such as its radius is contained in the l-shape overlap. For the midway blending results the propagation step used is the linear propagation approach. We noticed that the linear propagation yields better results for natural images and the normalization artifact is not evident as in Figure 4.4.

For the horizontal and vertical *blending edges* one can observe in general that the results are comparable to the other methods and sometimes better when we can observe the quilting path or where the Poisson blending "smoothes" the edge but this one is still evident. However some results using the midway blending may present deformations that are too abrupt (Figure 4.20) or may present color artifacts (Figure 4.20). For example in Figure 4.18 in the three fist examples one can clearly notice the quilting path. In Figure 4.19 the first example is an evident case that shows the advantages of the midway blending algorithm. This method tends to extend the structures when it can. Figure 4.20 shows failure examples. These failures are colour artifacts or highly abrupt deformations that are too evident. On the other hand, in the mentioned figures we attempted blending of strongly dissimilar patches, so that fusing them is close to a mission impossible. For

the circular edges however the results are not as convincing as for horizontal or vertical edges. This will depend on the curve γ used and the discretization of it. Nevertheless one can observe that when used in patch-based texture synthesis since the patches are quite similar on the overlap zones (it is intended to be this way) then using a circular curve is not problematic. One can also notice fewer or less color artifacts for the same reason.

4.4.2 Comparison of texture synthesis results

In this section we evaluate the results of synthesizing textures with patch based approaches using as stitching step the midway blending algorithm. We compare the results to [15] in Figures 4.21 to 4.25. The synthesis algorithm is the same in both cases, only the stitching step is modified. To compare the results the synthesized images are both initialized with the same seed patch and the parameters are exactly the same (patch size n and overlap size o).

In general the results of both methods yield satisfying and comparable visual results. Thus the midway blending approach could be a satisfying alternative to stitch patches together. This is particularly so for textures having a dominant structure direction, as for example the wood texture in Figure 4.1.

An interesting aspect of the stitching method would be to relax the constraint on the resemblance of the patches on the overlap area. This could be interesting since as pointed out in Annex A.4 when relaxing the error tolerance on the overlap resemblance of the patches more innovation is introduced in the result. Increasing this tolerance error value permits to increase the number of candidate patches used at an iteration of the synthesis method and so the chosen patch may vary more. As a consequence the patches P_{old} and P_{in} might be more distant to each other on the overlap area and so by just using the quilting method to stitch them the boundary might be noticeable since the values of thoses pixel are not alterated. Nevertheless with the *midway* technique a seamless fusion remains possible.

4.5 Conclusion

In this chapter we have presented a method for stitching texture patches inspired in the midway equalization algorithm [12]. This method consists in equalizing two images by transporting their gray levels cumulative histograms to a common one, the midway histogram. The cumulative midway histogram is the harmonic mean of the cumulative histograms of both images. The *midway blending* algorithm proceeds in the same way. Thus we consider the two signals we aim at registering as two distributions. The signals s_1 and s_2 on the blending edge are handled as *histograms*. The registered signals coincide with the midway signal defined as the *histogram* whose cumulative function is the harmonic mean of the cumulative functions of s_1 and s_2. The midway blending is composed of two steps. The first one consists in registering the signals obtained by restricting both patches to their common boundary. The second one consists in propagating the two midway deformations toward the interior of both patches. So far we have observed that the registration step achieves satisfying results whereas the propagation step is sometimes more challenging. We explored two approaches for this. The first one, the linear propagation, sometimes yields normalization artifacts. The second approach, the progressive midway, has no normalization issues. Nevertheless applying the second approach for natural images is too abrupt and the deformations are sometimes too evident. However for the linear propagation the results on natural images are satisfying and the normalization artifacts are almost invisible, unlike what we observed for the

synthetic image examples. The examples in section 4.3.1 were designed to illustrate that using the harmonic mean of the cumulative *histograms* does not always yield an intuitive solution.

In the first part of Section 4.4 we showed the result of blending two patches of the same texture by different stitching techniques. We observed our results are generally comparable or better. The quilting method [15] does not modify the content of the image. Instead, it looks for the edge where the patches best match. The Poisson editing method [47] yields a solution which mitigates the edges. Still, in challenging cases, both methods can leave behind conspicuous stitching edges. The midway blending avoids such edges, but it may sometimes require a patch deformation that is too abrupt. Besides color artifacts can be observed which are not present with the other techniques. In the second experimental section we tried the proposed technique in a patch based texture synthesis algorithm [15] where we replaced the quilting step by the *midway blending*. We observed that the results obtained with the midway blending are comparable to those obtained with the quilting technique. Since for such algorithms the patches to blend are similar on the overlapping area the artifacts caused by the attempts at blending too dissimilar patches no longer occur. We conclude that our technique could be used as a stitching step for this kind of texture synthesis algorithms.

The method as we have seen has some limitations, such as the normalization issue for the linear propagation. Other techniques could also be considered with the aim at stretching and/or expanding the images on the overlap area. We showed that the proposed method can be use as a stitching step in patch based texture synthesis methods and has room for improvement as for more complex textures to blend. It is clear at this point that the deformation technique we introduced should be explored further, and perhaps as generalized, by using other deformation and registration tools such as optical flow.

Figure 4.17 – Some challenging examples blending patches which do not match. Comparison with two other stitching methods. From left to right: input, Poisson blending [47], quilting method [15], midway blending.

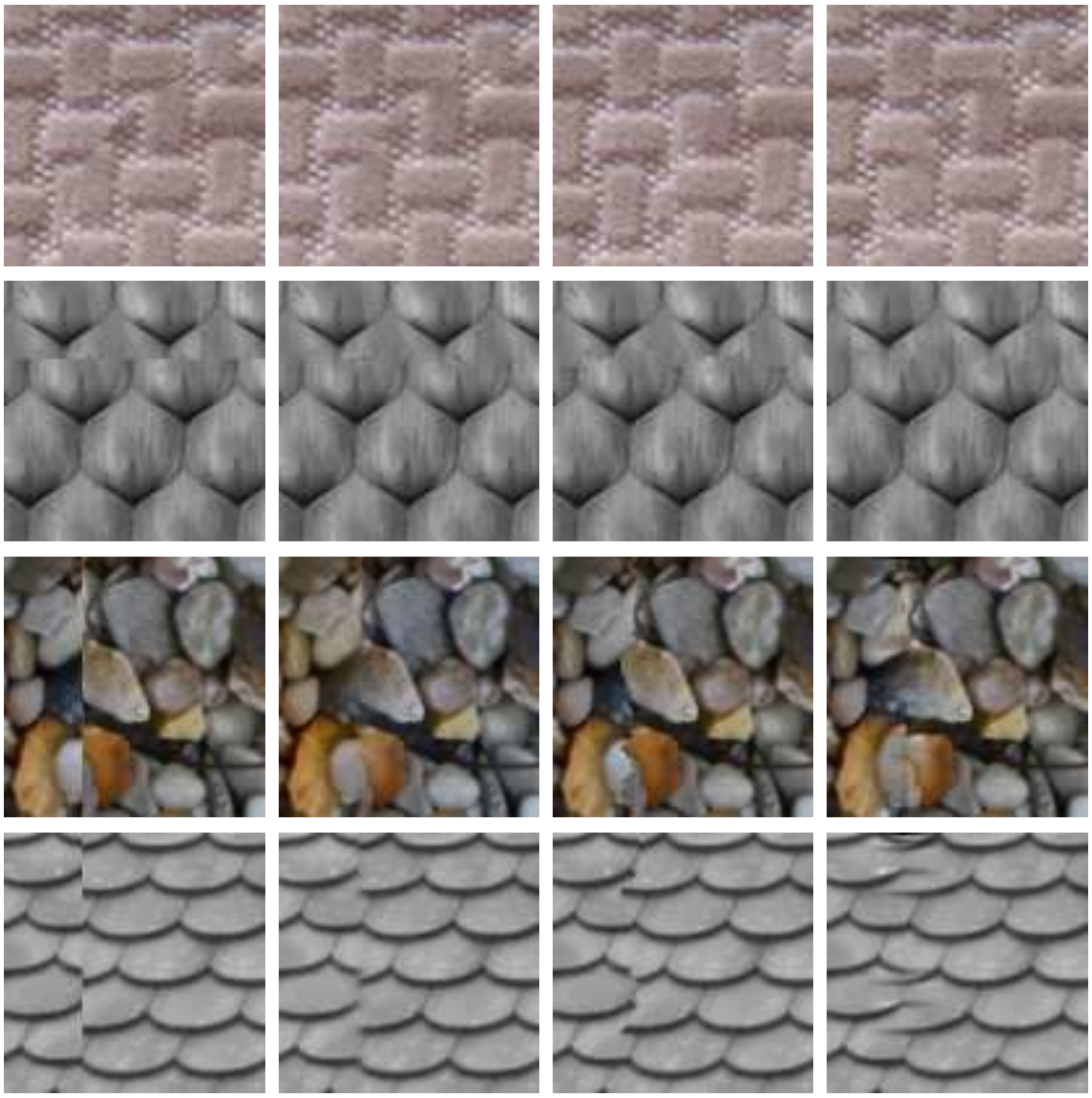


Figure 4.18 – Some challenging examples blending patches which do not match. Comparison with two other stitching methods. From left to right: input, Poisson blending [47], quilting method [15], midway blending.

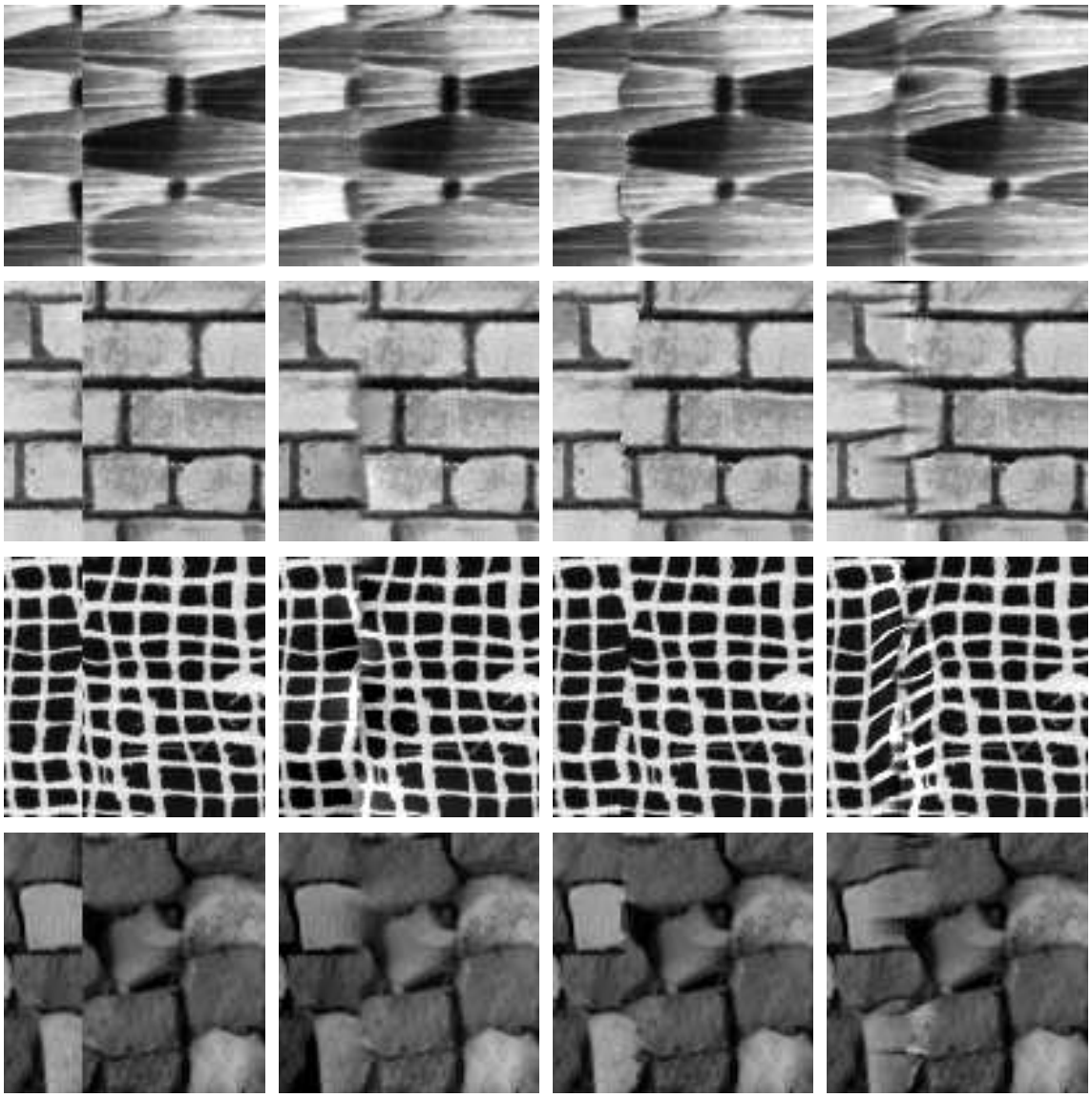


Figure 4.19 – Some challenging examples blending patches which do not match. Comparison with two other stitching methods. From left to right: input, Poisson blending [47], quilting method [15], midway blending.

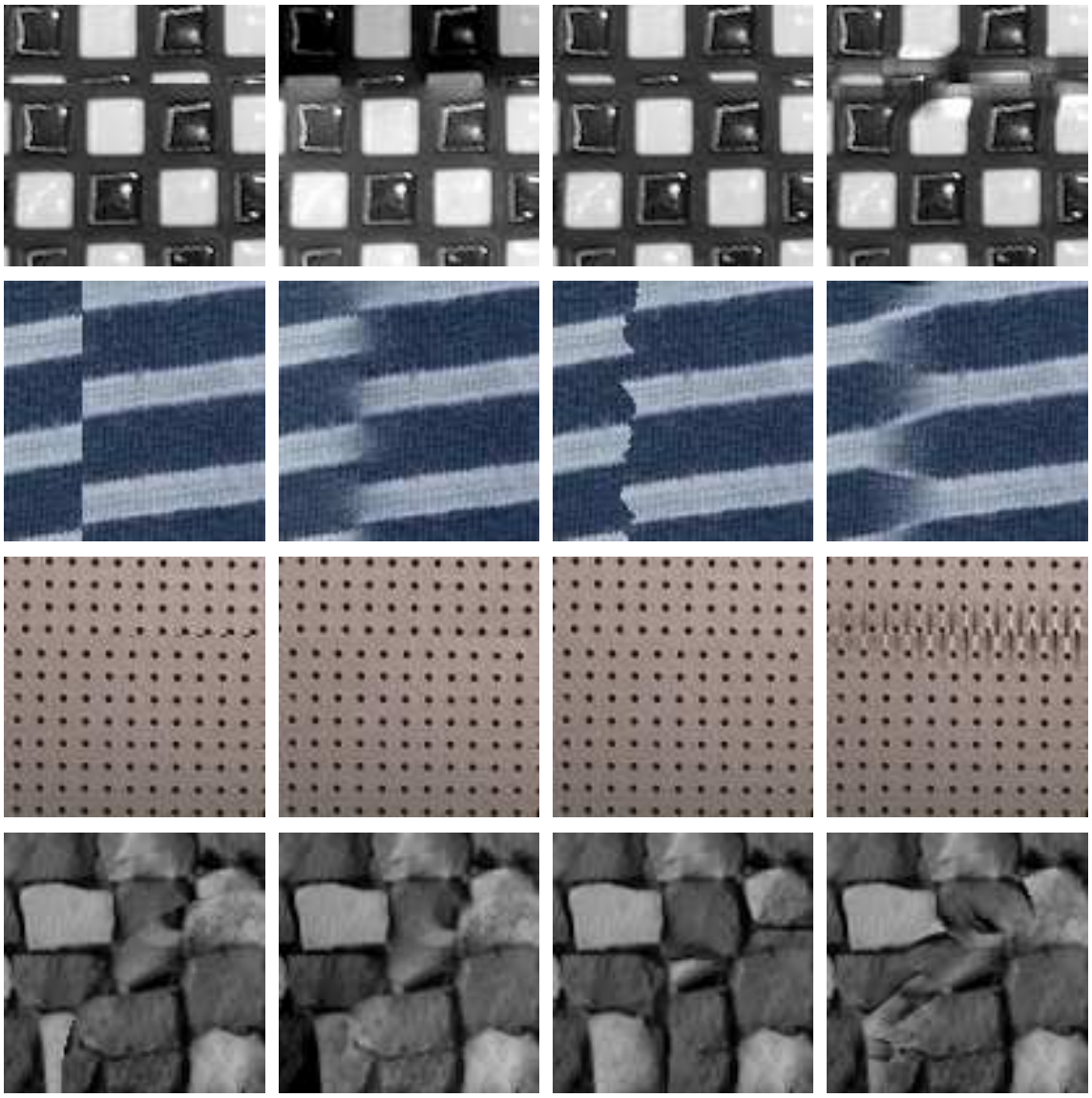


Figure 4.20 – Some nearly impossible examples blendig patches which do not match. Comparison with two other stitching methods. From left to right: input, Poisson blending [47], quilting method [15], midway blending.

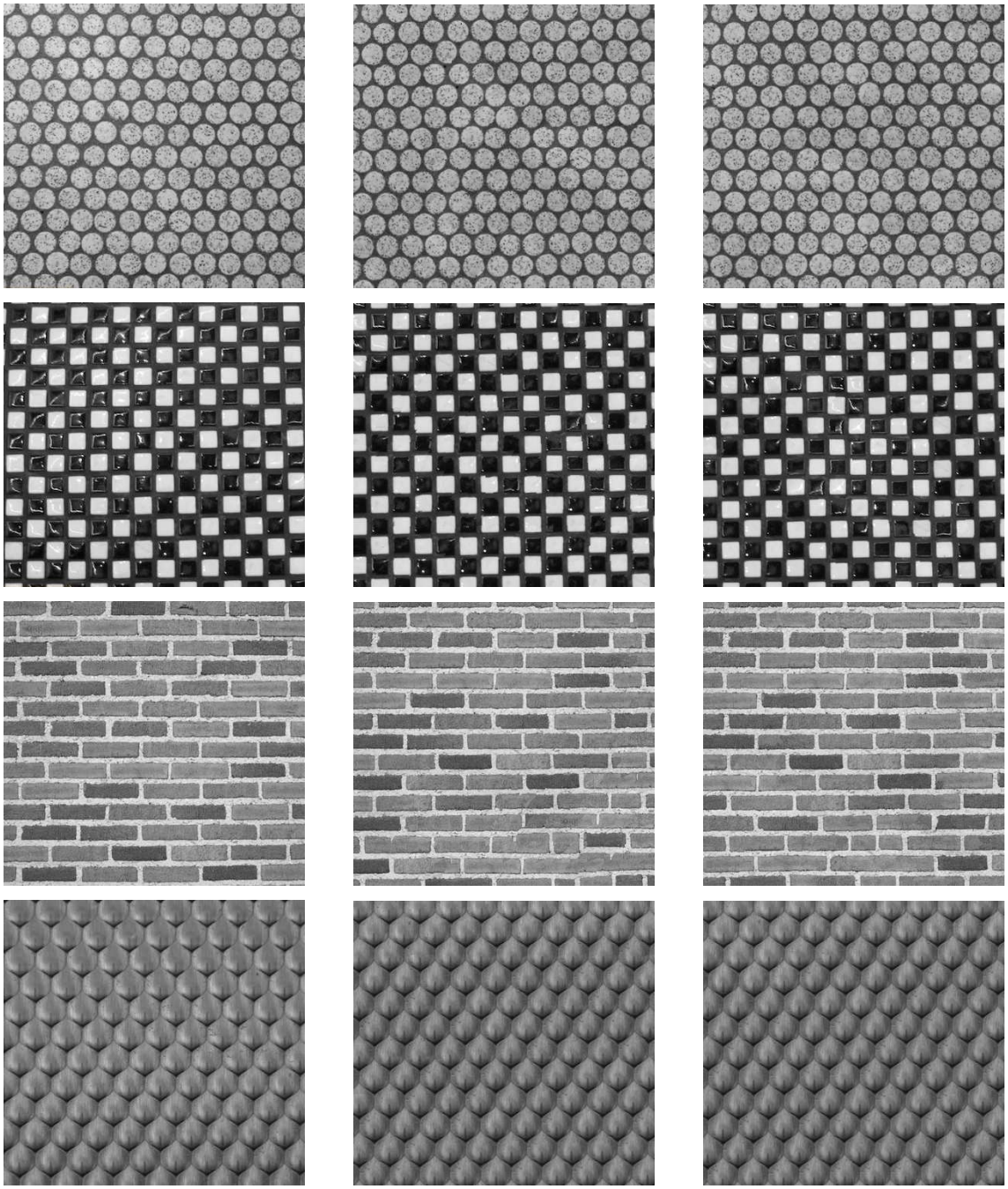


Figure 4.21 – Synthesis results 1. From left to right: input, texture synthesis results using *midway blending*, quilting algorithm [15]. For each example the patch size is $n = 40$, the same seed patch was taken to initialize the synthesized image, the overlap area is of size $o = 0.45n$ and the band width is $W = 10$.

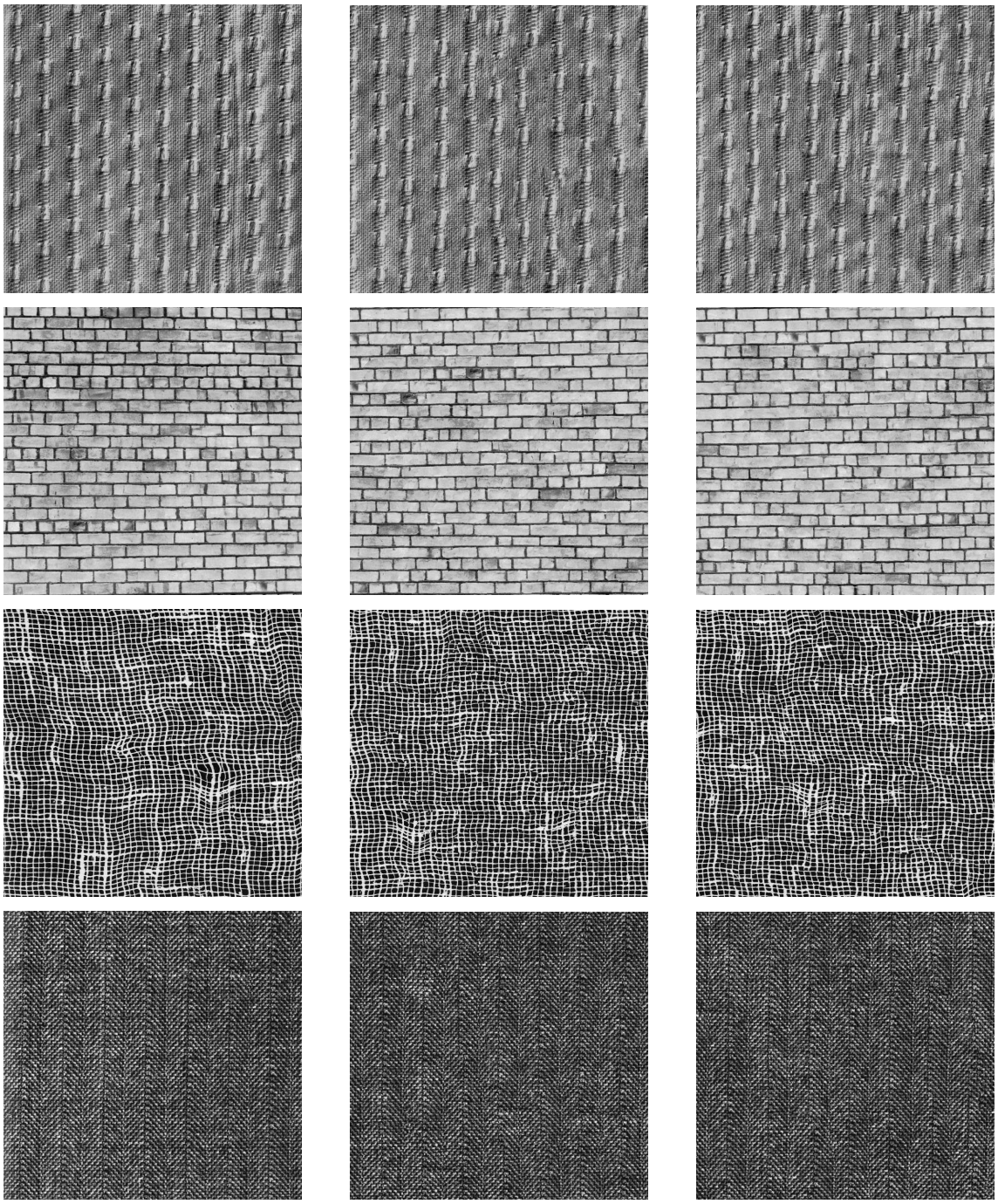


Figure 4.22 – Synthesis results 2. From left to right: input, texture synthesis results using *midway blending*, quilting algorithm [15]. For each example the patch size is $n = 40$, the same seed patch was taken to initialize the synthesized image, the overlap area is of size $o = 0.45n$ and the band width is $W = 10$.

Figure 4.23 – Synthesis results 3. From left to right: input, texture synthesis results using *midway blending*, quilting algorithm [15]. For each example the patch size is $n = 40$, the same seed patch was taken to initialize the synthesized image, the overlap area is of size $o = 0.45n$ and the band width is $W = 10$.

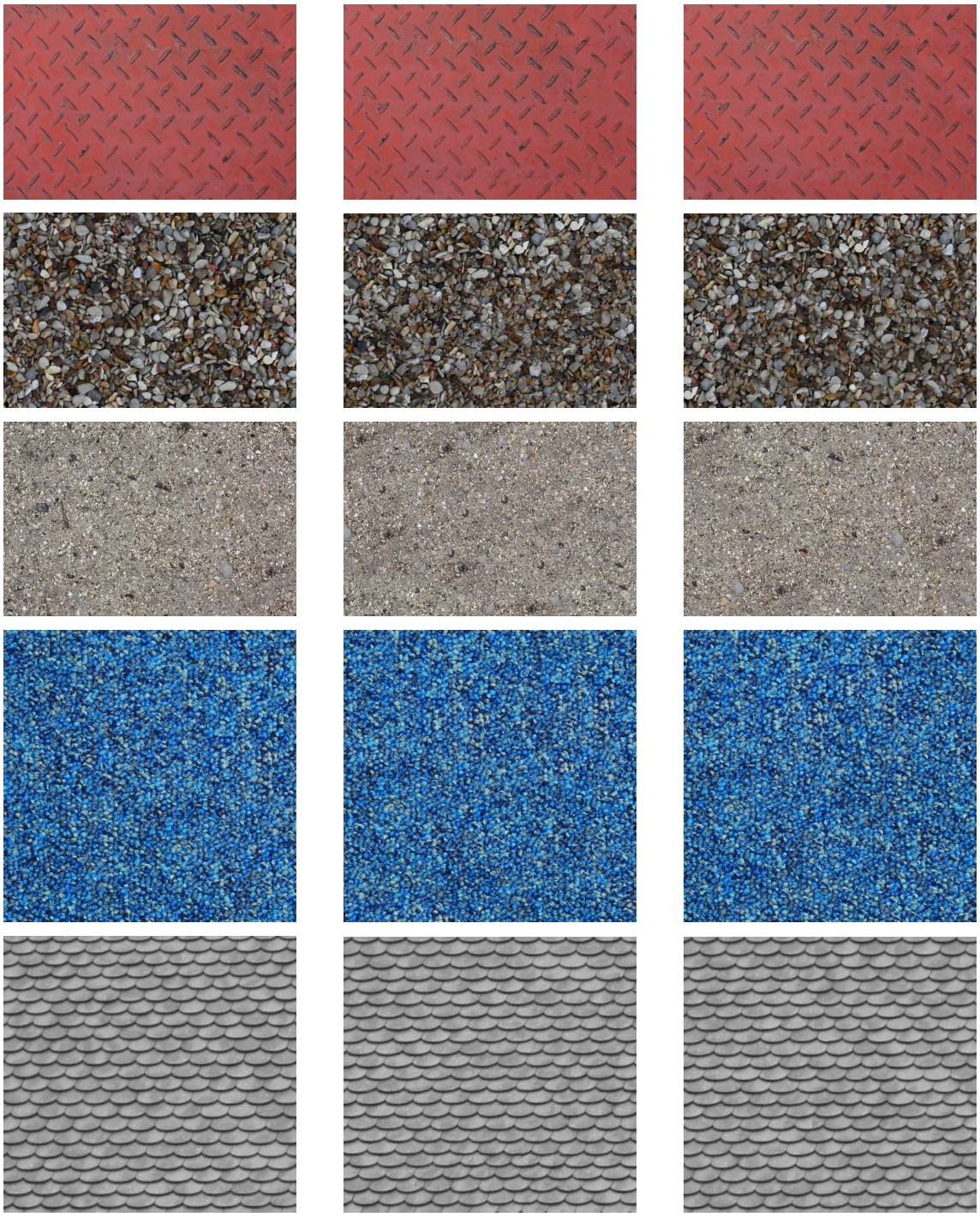


Figure 4.24 – Synthesis results 4. From left to right: input, texture synthesis results using *midway blending*, quilting algorithm [15]. For each example the patch size is $n = 40$, the same seed patch was taken to initialize the synthesized image, the overlap area is of size $o = 0.45n$ and the band width is $W = 10$.

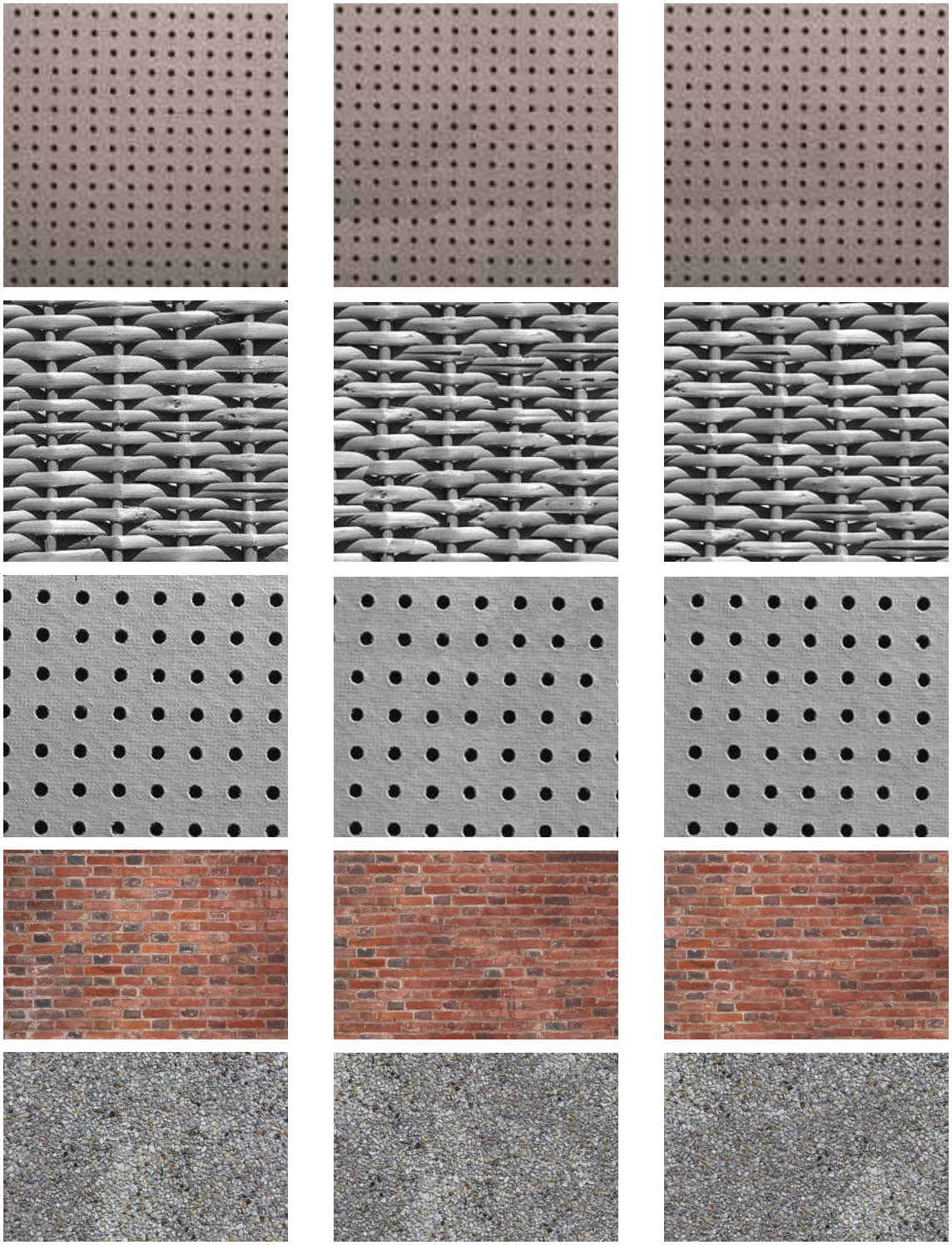


Figure 4.25 – Synthesis results 5. From left to right: input, texture synthesis results using *midway blending*, quilting algorithm [15]. For each example the patch size is $n = 40$, the same seed patch was taken to initialize the synthesized image, the overlap area is of size $o = 0.45n$ and the band width is $W = 10$.

5 Can we emulate large textures?

In this chapter we aim at analyzing the limits of exemplar based texture methods when using large natural texture images, a new problem which we might call Big Exemplar Based texture synthesis. Unlike exemplar-based methods that use small texture samples the goal is to synthesize large textures. We shall explore this problem with the tools at reach, namely the best methods from the extensive literature, and will not hesitate to combine them. To anticipate our conclusions, none of the existing methods manage by itself to satisfactorily emulate "big textures". Yet, we'll find that a clever combination of these methods yields interesting-if not perfect-results. Interestingly, the existing methods provide very good results on small crops of these images. But we discovered with some dismay and shall illustrate on many examples that "big textures" are very different from academic texture crops. They show drastic local textural variations, even if their overall effect still is the one of a single texture. This study is therefore a good tested to put on trial all existing exemplar-based texture synthesis methods.

5.1 Introduction

This chapter shows an extensive study on the limits of exemplar-based methods when used to synthesize large natural textures. The goal of these methods is to create from a given sample decor (for example a wood texture as in Figure 5.1) new, different samples that are perceptually equivalent to the input example. Current exemplar-based approaches use small input samples. When the texture example is not stationary, non-parametric methods often end up in a *garbage growing*, meaning that only a local piece of the texture is being reproduced to the detriment of the rest. The parametric methods present a different problem in presence of a non-stationary texture: when the input sample contains *more than one texture* global statistics are learnt on the input. It follows that the result is a homogenization of the input sample. The texture images we try to synthesize in this chapter are quite complex: large and non stationary.

We began by trying some of the exemplar based texture synthesis approaches (Efros and Freeman [15], Portilla and Simoncelli [50], Galerne et al. [21] and the multiscale locally Gaussian method presented in Chapter 3) on the following type of images: wood, stone and metal (Figures 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6). Among these textures the most challenging were the wood samples. We observed that the synthesis results for the wood images were not satisfying. However for local crops of these textures where no strong global organization were evident the results obtained are impressive. One concludes obviously that, in spite of their perceptual unity, large textured images are more than *a texture* in the usual academic sense. In Figures 5.7, 5.8, 5.9, 5.10, 5.11 different crops

of the inputs in Figures 5.1, 5.2, 5.3 are shown where one can clearly see that within a same texture very different textures are present. Wood planks for example are non-stationary images, containing several different textures and with very recognizable and unique structures such as ring and knots. For other complex surfaces such as the stones and metals the exemplar based techniques seem to give more satisfactory results. Again these textures are not as the academic textures considered for the experiments done to evaluate their performance. These images are more complex as can be seen in Figures 5.12, 5.13 and 5.14 which are crops of the inputs in Figures 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6. All the experiments done are showed in the following sections, including failures that permit to discard certain techniques for certain texture. We follow by showing results of a combination of the considered texture synthesis methods, there are two of them:

- The Efros and Freeman method [15] combined to the locally Gaussian method
- The Portilla and Simoncelli method [50] combined to the locally Gaussian method

Combining the methods yields better visual results in particular for the combination PS + LG. Still this combination will not make it for big textures. We then show other emulation attempts, like for example the application of geometric deformations to the inputs aiming at changing the aspect of the salient structures. Randomly switching sub-parts of the input to partition the salient structures and reorganize them in a way that in the outputs the repeated patterns are less evident. As a last experiment we tried to vary the low frequency information while leaving unchanged the high frequencies with the same purpose. In all three emulation attempts we observed interesting behaviours for some of the textures.

This chapter is organized as follows. Section 5.2 describes the type of inputs we worked on. In section 5.3 a description of some of the state of the art exemplar-based texture synthesis methods is provided listing the pros and cons of each of them. In section 5.4 a first set of experiments are showed on small crops of the big textures. All of the exemplar-based methods of section 5.3 are tested on these images. They are tested separately and then some of them are combined. The results on these images are very satisfying. The same experiments are performed on the big texture themselves to try to emulate the structures at low resolution. The results are not very satisfying since the typical drawbacks of exemplar based methods arise (verbatim copies and garbage growing). This brings us to test other alternatives presented in Sections 5.5, 5.6 and 5.7. We try the application of one and two dimensional anamorphosis to the inputs, a random combination of slices of the input texture and the application of the [21] method to randomize the low frequencies of the input.

5.2 Large natural textures

Wood, stone and metal textures In this section we present images of typical "big textures". These images are of very high resolution(more than $1GB$). For this reason the experiments were sometimes done on zoomed out versions of these inputs. We considered three types of classic natural textures: wood, stone and metals shown in Figures 5.1, 5.2, 5.3, 5.5, 5.6. As a general remark on these inputs one can notice that they hardly represent "real textures" in the academic sense. Indeed we observe on different crops of the inputs, of size 500×500 pixels, that within one of these large natural textures several different textures are present (Figures 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14). This is clearly not the case in the smaller academic textures.

Let us consider for instance the crops in Figure 5.7 which are very small portions (500×500 pixels) of the inputs in Figure 5.1 whose original size is of 7087×23633 pixels. The four crops

provided show four completely different textures. Still worse, these crops themselves are not stationary. This obviously illustrates how hard it will be to synthesize a new different textures from such an example.

Among the inputs that we considered the more challenging ones are the wood textures. Strong salient configurations are evident in these images that makes difficult to generate new variants of these structures instead of mere "copies" of them. The stone and metal decors are in this sense easier. They do not present strong salient patterns. Nevertheless they still are highly non stationary.

Figure 5.1 – Wood texture 1. The original input provided was of size 7087×23622 pixels. For the experiments the image was reduced by a zoom out of factor 16.

Figure 5.2 – Wood texture 2. The original input provided was of size 3500×13500 pixels. For the experiments the image was reduced by a zoom out of factor 16.

Figure 5.3 – Wood texture 3. The original input provided was of size 4000×14000 pixels. For the experiments the image was reduced by a zoom out of factor 16.

5.3 Exemplar based texture synthesis methods

Exemplar-based texture synthesis is defined as the process of generating from an input texture sample a perceptually equivalent larger one. Exemplar-based texture synthesis algorithms are commonly divided into two categories, the statistics-based methods and the non-parametric patch-based methods. The first category models a given texture sample by estimating statistical parameters that characterize the underlying stochastic process of the input. Although these methods can

Figure 5.4 – Wood texture 4. The original input provided was of size 3400×14600 pixels. For the experiments the image was reduced by a zoom out of factor 16.

Figure 5.5 – Stone textures 1, 2 and 3. The original inputs provided were of size 23622×23622, 6903×9983 and 21260×21260 pixels (left to right). For the experiments the images were reduced by a zoom out of factor 32, 8 and 32.

faithfully reproduce some of the global statistics of the sample and synthesize micro and pseudo-periodic textures, they generally do not yield high quality visual results for more structured ones, in particular when the sample is small and contains large objects. The second category rearranges local neighbourhoods of the input sample in a consistent way. These methods provide efficient algorithms able to reproduce highly structured textures. Even though they yield visual satisfactory results, they often turn into practising verbatim copies of large parts of the input sample and growing garbage.

Within these two groups we chose some of the state of the art methods to synthesize the given decors in section 5.2. These methods are:

- Portilla & Simoncelli [50], denoted *PS*
- Galerne et al. [21], denoted *RPN*
- Efros & Freeman [15], denoted *EF*
- Multiscale locally Gaussian (Chapter 3), denoted *MSLG*

We follow by giving a short explanation of these methods.

5.3.1 The Portilla and Simoncelli algorithm

The Portilla ans Simoncelli method [50] is performed in two steps: analysis and synthesis. First, the texture sample is characterized by a set of statistics estimated on the coefficients of its steerable wavelet decomposition [58]. The steerable pyramid is an overcomplete linear multi-scale and

Figure 5.6 – Metal texture 1 and 2. The original inputs provided were of size 23622×23622 and 6000×12390 pixels (left to right). For the experiments the images were reduced by a zoom out of factor 32 and 8 (left to right).

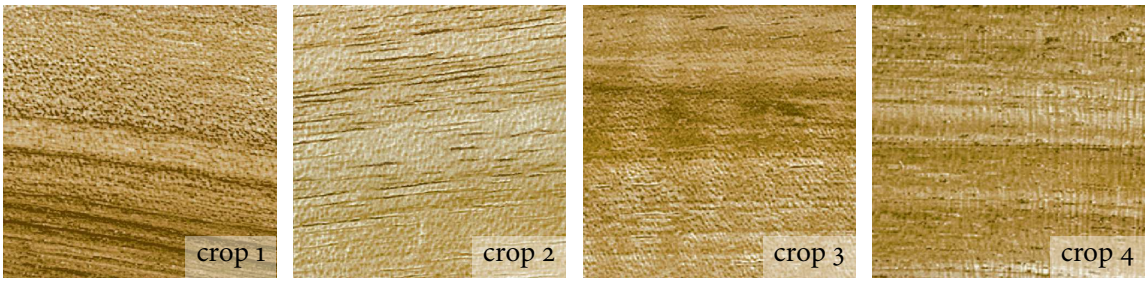


Figure 5.7 – Four crops of different parts of Figure 5.1 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

multi-orientation image decomposition. The statistics extracted on the coefficients of this decomposition are their autocorrelations, cross-correlations (inner and intra scales) as well as the statistical moments of order one, two three and four of the input sample's values. Second, these statistics are enforced on a white noise image obtaining like this a texture satisfying these constraints. To this end the authors in [50] suggest to project the synthesis image into the subspace of constraints iteratively until a stabilization of the image observed. The final image may not have exactly the same statistics as the input sample but reached a local minima where it stabilized. The synthesis results in a pre-attentive examination are in general indistinct from the original samples. Nevertheless to an attentive examination the synthesis results on structured textures often present blur and phantom effects. The structures are missed and the method tends to homogenize the output texture. We follow by showing in Figure 5.15 four synthesis results. The two first examples (top row in Figure 5.15) represent two microtextures where the method yields excellent results. Nevertheless for the last two examples (bottom row in Figure 5.15) even though we recognize the nature of the input sample in the synthesized image one can observe the strong structures are missing. In both cases it is impossible to recover the lined up tiles.

Figure 5.8 – Four crops of different parts of Figure 5.2 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

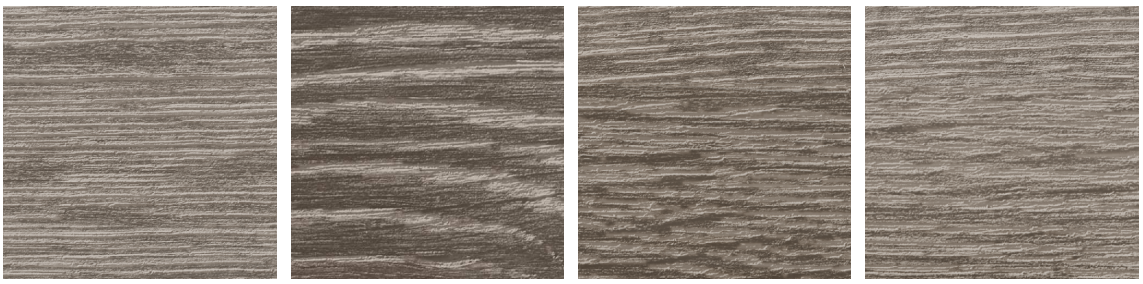


Figure 5.9 – Four crops of different parts of Figure 5.3 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

5.3.2 Micro-texture synthesis by phase randomization

The random phase method [21] synthesizes a texture by randomizing the phase of the Fourier coefficients of the input sample. The results are very satisfying for textures that are characterized by their Fourier modulus. This method will not grow garbage or verbatim copies of the input sample. It is however limited to micro textures. It will fail synthesizing structured textures. We follow by showing in Figure 5.16 four synthesis examples. The two first synthesis (top row in Figure 5.16) show outstanding results. These microtextures are indeed well represented by their Fourier modulus. However this is not the case for the last two examples (bottom row in Figure 5.16). Clearly, the knowledge of the modulus of the Fourier coefficients of these textures is not sufficient to recover the strong contrast of the inputs. For a detailed description of the method please refer to [20].

5.3.3 The Efros and Leung algorithm

The Efros and Leung algorithm relies on Claude Shannons's Markov random field model for the English language [56]. He proposed to use a Markov chain to model the underlying stochastic process of a sequence of letters in a piece of English text. The same idea was used by Efros and Leung in [14] to synthesize a texture image by considering that a pixel value depends on the values of its neighbouring pixels. The method is illustrated in Figure 5.17 and works as follows. For a given input texture a new image is synthesized pixelwise. For a pixel p being synthesized the algorithm finds all the neighbourhoods in the input image that are similar to the neighbourhood of p. Then one of this neighbourhoods is randomly chosen and the central pixel value is affected to the pixel being synthesized. The neighbourhood of p is a square patch centered in p and only the known pixels of this patch are considered when comparing to the neighbourhoods of the input. In general

Figure 5.10 – Four crops of different parts of Figure 5.4 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

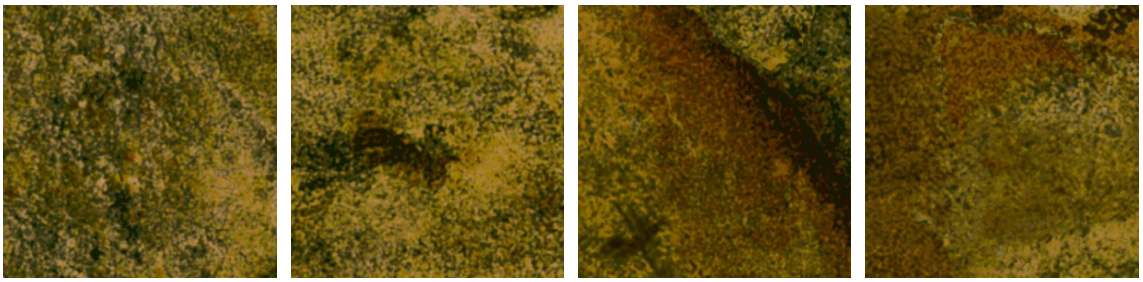


Figure 5.11 – Four crops of different parts of Figure 5.5 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

the visual results are very impressive most of all for structured textures. Nevertheless this algorithm suffers from two important drawbacks: verbatim copies of the input and garbage growing (the algorithm starts reproducing iteratively one part of the example and neglects the rest). We invite the reader to refer [1] for a detailed description of the method.

5.3.4 The Efros and Freeman algorithm

Efros and Freeman's method [15] is an extension of Efros and Leung's method. It is based on the same principle where the pixel values are conditioned to their neighbourhood values. Efros and Freman propose to generate a new image patch by patch in a raster scan order. At each iteration a patch that is only partially defined on a region called *overlap region* is completed. This is the patch under construction. To do so a patch of the input image that matches the patch under construction on the overlap region is randomly selected (*patch selection* step). An optimal boundary cut between the chosen patch and the one under construction is then computed within the overlap region (*stitching* step). This optimal boundary cut is used to construct the new patch by blending the chosen patch and the patch under construction along the cut. The results obtained are very impressive, particularly for highly structured textures. The patch size being larger, the risk of garbage growing is reduced comparing to the Efros-Leung algorithm. Nevertheless the risk of verbatim copies remains ans is even amplified. Moreover, the respect of the global statistics of the input is not guaranteed and this is quite visible when the input texture is not stationary (for example if there is a change of illumination across the image). In Figure 5.18 we show four synthesis examples. For the first two microtextures (top row in Figure 5.18) one can observe that the generated image does not respect the global statistic of the input creating "stains" of different illuminations. This follows the fact that the input texture is not homogeneously illuminated. However for the second example where this illumination effect is not evident the synthesis is excellent. The last two

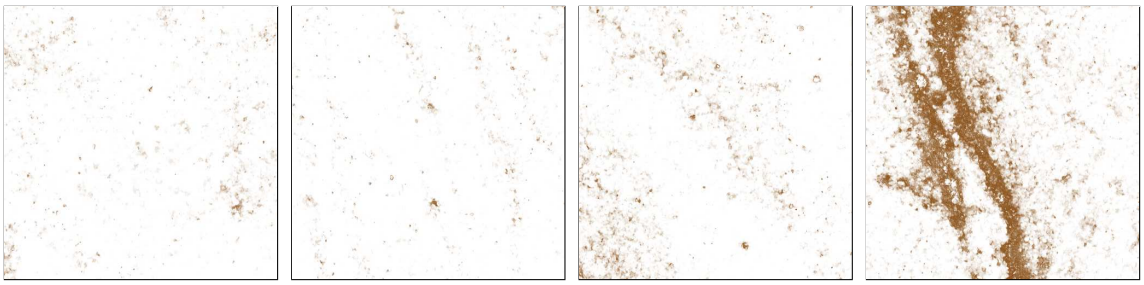


Figure 5.12 – Four crops of different parts of Figure 5.5 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

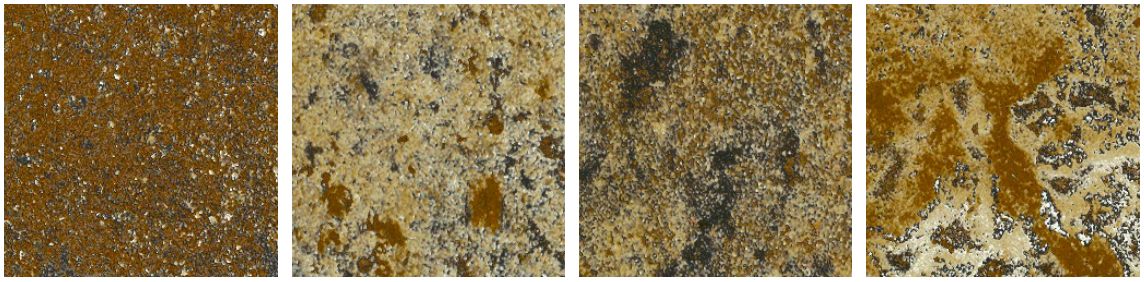


Figure 5.13 – Four crops of different parts of Figure 5.6 at its original resolution. The cropped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

examples (bottom row in Figure 5.18) present impressive results recovering perfectly the strong structures of the input. We invite the reader to refer to Annex A for a detailed description of the method.

5.3.5 The multiscale locally Gaussian algorithm

The multiscale locally Gaussian method presented in Chapter 3 is a parametric patch based method. The output image is constructed patch by patch, in a raster scan order. In contrast with the Efros and Freeman work [15] this method involves a multiscale local Gaussian model in the patch space. Every patch used to construct the output texture is sampled form a multivariate Gaussian distribution (a different one for each patch). For a given patch p under construction the set of m patches from the input texture that are similar (to the one under construction) to p on their overlap area is used to estimate the parameters of the multivariate Gaussian distribution used to sample a new patch. This is done for each patch in the output image. Figure 5.19 illustrates an iteration of the method. The multiscale approach permits to use "different patch sizes" within one synthesis and hence capture the details at different scales. The results are visually satisfying. The quality of the results is higher than those of statistics based methods. Nevertheless, even though the patches generated do not exist in the input texture the risk of visual verbatim copy remains. Four synthesis examples are shown in Figure 5.20 where once again we can notice, like for the Efros and Freeman results, that when the input is not stationary the generated output might not respect the global statistics of the input sample. For the other presented examples the results are excellent. For a more detailed description and extensive results please refer to Chapters 2 and 3.

Figure 5.14 – Four crops of different parts of Figure 5.6 at its original resolution. The croped images are of size 500×500 pixels. Each one represents a different texture belonging to a single "big texture".

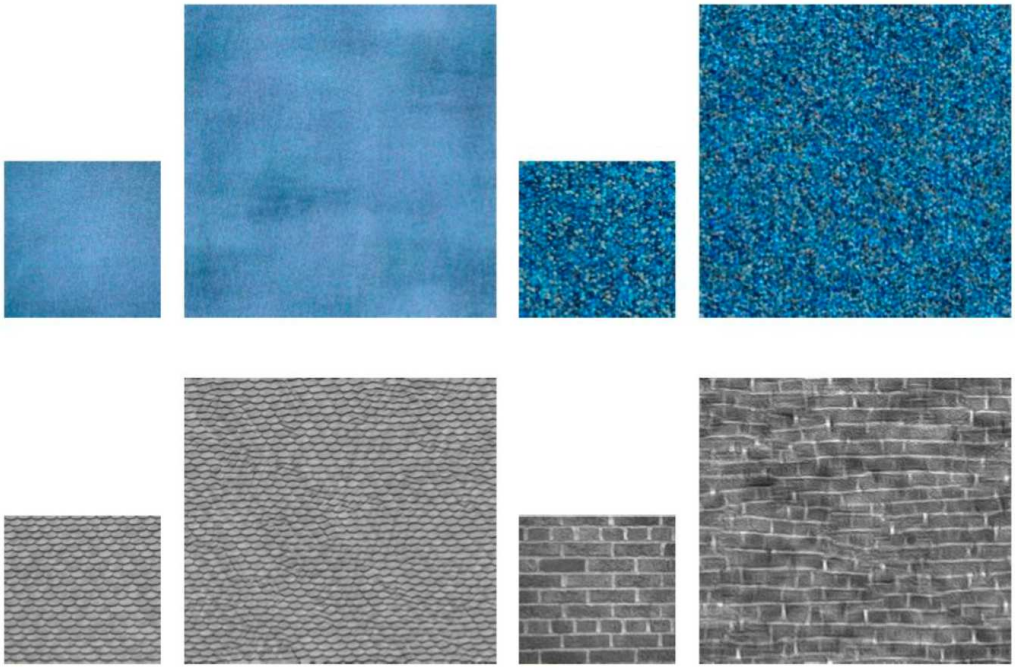


Figure 5.15 – Synthesis results of the Portilla and Simoncelli method [50]. It is satisfactory for many small grain textures (first row) but may miss the global structure (second row).

5.4 Applying the classical methods

5.4.1 Experiment 1: exemplar-based methods

We started by applying the exemplar-based methods described in section 5.3 to small crops of the provided inputs. In Figure 5.21 and Figure 5.22 one can observe these results.

In general when applying RPN and PS to the small crops the results obtained are too blurry. In the case of EF and MSLG, for some inputs, satisfying results are observed. This is true especially when the input is stationary. Otherwise garbage growing and verbatim copies are often a risk. One can also observe that for some examples the result of the MSLG method might be slightly blurry. In Figure **??** larger crops of the inputs are considered. These crops are taken on images of lower resolution regarding the original inputs. For these examples it is evident that applying the PS or RPN method will not work at all. On the other hand for the EF and LG algorithms the verbatim copy and garbage growing is more evident and likely to happen, particularly when the inputs have

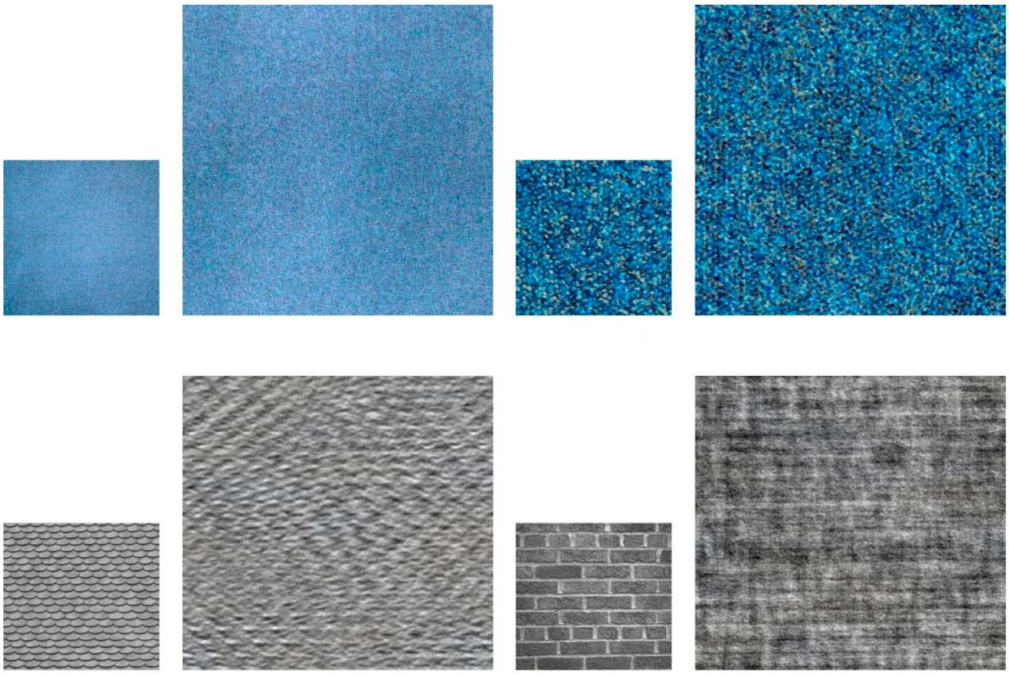


Figure 5.16 – Synthesis results of the RPN method [21]. This method works extremely well for a class of "microtextures" including tissues and granular textures with no geometric structures [20].

strong isolated patterns, as it is the case with wood textures. We follow by showing a combination of these methods on the same examples.

5.4.2 Experiment 2: combining exemplar-based methods

In this section we present some synthesis results when using a combination of different texture synthesis methods. We apply them to the two sized crops of the different input images. The combinations considered are the following:

- The locally Gaussian method combined to the Efros and Freeman method denoted as *LG+EF*
- The locally Gaussian method combined to the Portilla and Simoncelli method denoted as *LG+PS*

LG + EF The combination of the locally Gaussian method with the Efros and Freeman method consists of two steps. The first step synthesizes the given input I with the locally Gaussian method generating a new texture image that we denote I_{lg}. The second step consists in applying the Efros and Freeman algorithm to the given input sample initializing the output image that we denote I_{ef} with the image I_{lg}. The method is basically the same as the one described in section 5.3. The only step of the algorithm that is modified is the *patch selection* step. In the method described in [15] at each iteration the added patch was chosen among those (in the input sample) whose overlap region was similar to the one of the patch under construction. When combining the methods instead of only comparing the overlap areas we compare the entire patches. Initializing the output with a first synthesis I_{lg} enables the method to use the whole patch under construction to find a candidate in the input sample I. The candidate patch taken from I is then quilted in I_{ef} in the

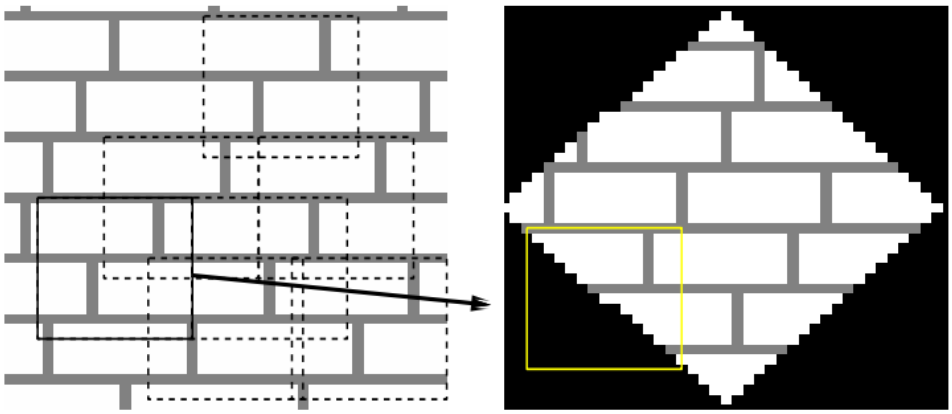


Figure 5.17 – Overview of the Efros and Leung algorithm [14]. Given a texture image (left) a new image (right) is being synthesized a pixel at a time. For a pixel p being synthesized the method finds all neighbourhoods in the left image that match the neighbourhood of p and then chooses randomly one of the neighbourhoods and assigns its central pixel value to p. The image was taken from the work in [14].

corresponding position with the same stitching step as in [15]. This combination allows to recover the lost resolution of the LG synthesis. However it is not capable of masking the garbage growing effects as effectively LG+PS combination does.

LG + PS The combination of the locally Gaussian method with the Portilla and Simoncelli method consists of two steps. A first step where given the input image I a new texture I_{lg} is generated using LG. The second step uses PS where the initialization "noise image" is replaced by I_{lg}. As explained in section 5.3 the statistics to impose are learnt on the input I. What follows is a synthesis step where the output image is projected on the subspaces of constraints. There exist several local solutions to this projection step. When initializing PS with the result of LG the initialization image is generally quite close to the images living in the sub-space of the whole set of constraints. Thus the result obtained is improved compared to PS images starting from a random noise image. Naturally fixing the initialization of the PS algorithm removes the randomness of the generated texture. But this is not the case since the initialization is itself random as it is generated from another random process.

We observed that combining EF to LG sometimes slightly improve the granularity of the result but that if the result of the LG method diverged then applying EF would not mask these problems. However, the combination of PS with LG masks efficiently these drawbacks ans improves significantly the quality of the synthesized image.

In Figures 5.23 and 5.24 one can observe that *so far the combination LG+PS permits to emulate all local textures with no definite global structure.* That is for local synthesis combining these two methods (LG and PS) is a good solution to synthesize locally the decors provided. These small images are *easy* textures. We now move one step farther and attempt a synthesis of larger parts of the original inputs (see Figure 5.25). As expected the results are less satisfying than for local synthesis.

In summary, LG+PS is a sophisticated method, more than any existing, and it works satisfactorily for local textures with no strong global organization. Thus it makes correct and even amazingly *good imitations* of *small* pieces of textures, yet remains *unsatisfactory* for *large structured* pieces of wood. Thus another method must be considered for such images which in fact *are more than a*

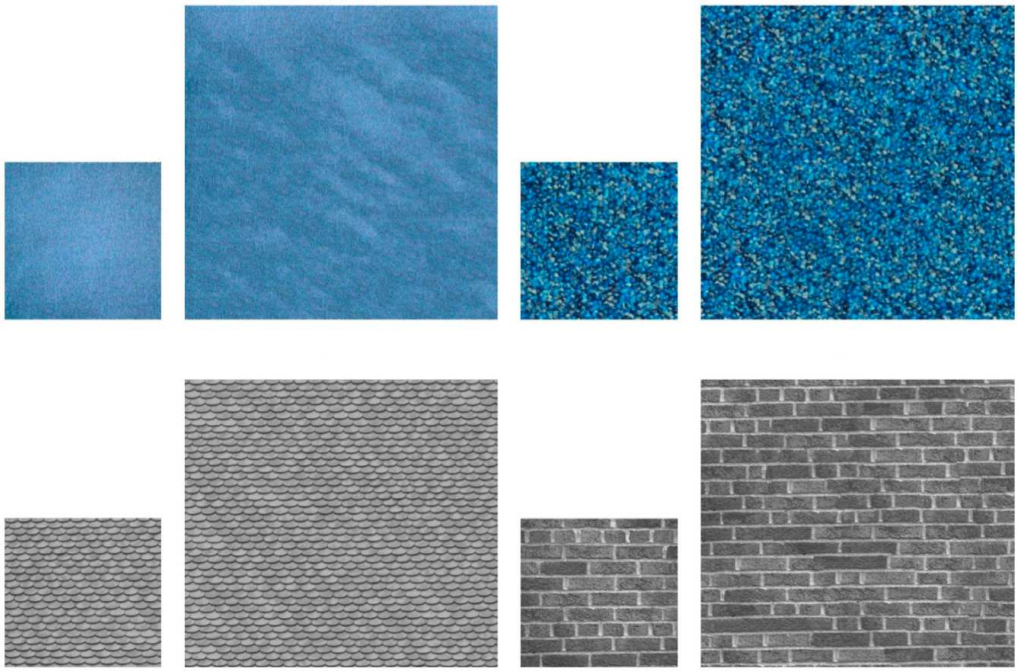


Figure 5.18 – Synthesis results of the Efros and Freeman method [15]. Works for microtextures with a risk of not respecting the global statistics (see the first experiment, top left). Works for macrotextures with the risk of verbatim copy.

texture, as they have this non-stationary global organization with rings and knots, among other special, very recognizable *structures*. In short, wood at a large scale is *NOT a texture* in the usual sense and therefore imitations of it will *either fail or tend to repeat* the input. We therefore tried to expand the method to *capture a structure at low scale and scale up with details* that we know how to do deal with it. As we shall see, this will fail because it will fatally create *repetitions*.

5.4.3 Experiment 3: multiscale on the low resolution inputs

A last experiment was performed with the locally Gaussian algorithm using the multiscale approach (MSLG). The goal was to create a low resolution landscape on which details are progressively added. The multiscale approach is a classic solution to contemplate details at different scales within an image. This approach was applied on a low resolution version of the whole sample images.

The existing methods are able to perform a correct synthesis on some details of the provided examples at several scales. However due to the non stationarity of the provided textures, adaptations of the methods should be considered. We observed that even using a multiscale approach to create a low resolution landscape is not enough for the wood texture where large and special global organizations are visible and difficult to synthesize without creating a verbatim copy effect or garbage growing due to the strong non stationarity of this kind of images. Hence we decided to see if some perturbations of the input might be enough to hide the resemblance applying some geometric deformations on the inputs. This is presented in the following section.

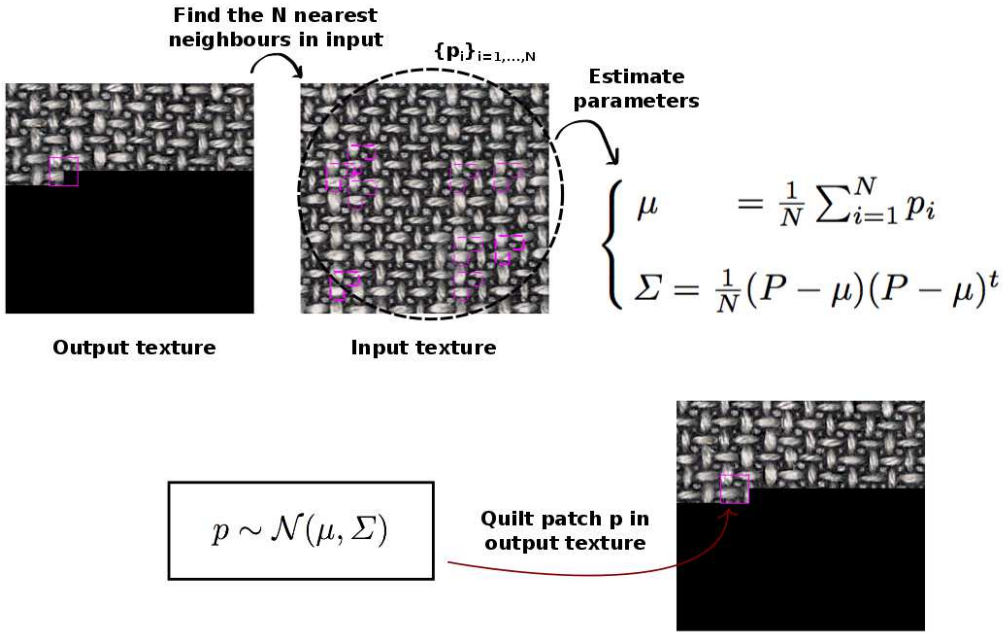


Figure 5.19 – Overview of the multivariate Gaussian distribution of a patch. For the patch q (pink patch in the output image) being synthesized the algorithm finds the m nearest neighbours in the input image that matches q. Then, on this set of m patches a multivariate Gaussian distribution is estimated from which a patch p is sampled and quilted in the output at the corresponding position.

5.5 Masking repetitions with anamorphosis

As we have seen in the previous three experiments, applying and combining different exemplar based texture synthesis methods works for local textures. For more complex images this is not the case. The images are highly non stationary and this makes it very difficult. Even though we tried synthesizing the inputs at lower resolution aiming at recovering a synthesized sample containing the global arrangements to continue by adding progressively the details of the different scales, the results were not satisfactory. The textures still retain a strong global organization at lower scales. If the low resolution synthesized image contains some repetitions, for example, then it is not easy to recover when scaling up in resolution. Thus we decided to change the strategy and aim at hiding these undue repetitions. As a first alternative we considered applying different anamorphoses to the inputs in order to deform the global organizations of the sample in a way that it is unrecognizable from one synthesis to another. Yet, the deformation cannot be too brutal. For example in a wood image, the deformation of a knot can be strong but should maintain it approximately convex. Unfortunately, we observed that after deformation, these global configurations are still recognizable from one image to the other, in particular for wood textures. We tried several different deformation functions that we separate in one dimensional anamorphoses and two dimensional anamorphoses. Basically, the one dimensional anamorphosis will deform the input in one direction and the two dimensional anamorphosis in two directions. For the one dimensional case we chose three simple functions to analyze their behaviours. For the two dimensional case we chose a geometric deformation performed as a simulation of turbulence [44].

Applying several different one dimensional and two dimensional anamorphosis to the wood

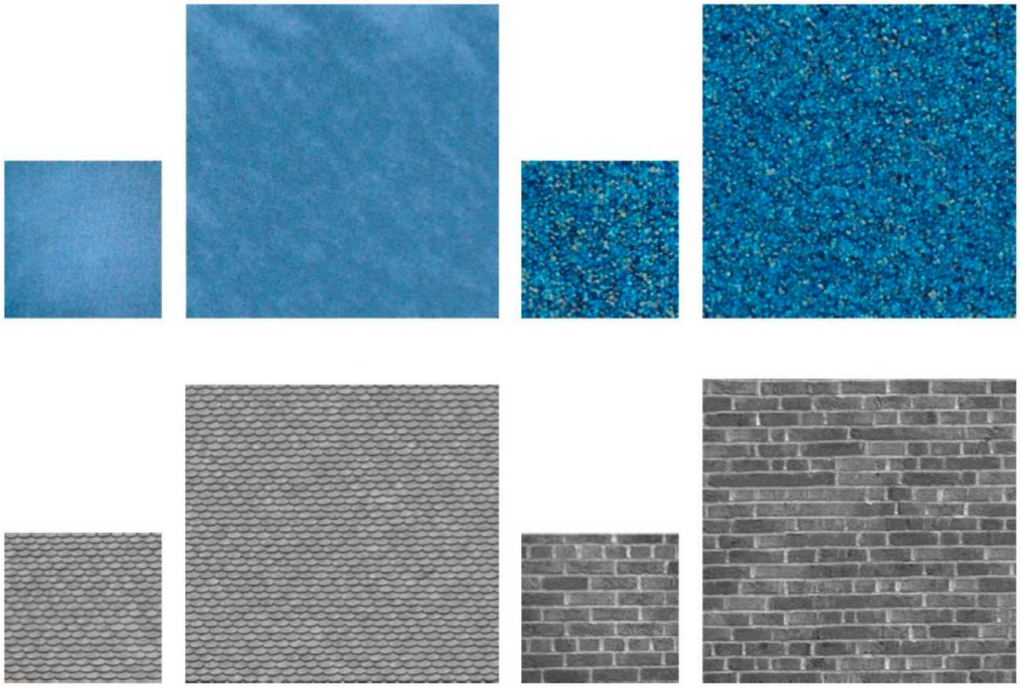


Figure 5.20 – Synthesis results of LG [8]. Works for microtextures with a risk of no respect of the global statistics (see the first experiment, top left). Works for macrotextures with a lower risk of verbatim copy than EF for this kind of textures.

samples is not enough to hide the repetitions among the different generated planks. This is explainable as human perception is invariant to a regular anamorphosis. If one observes the results in Figures 5.31, 5.32, 5.33, 5.34, 5.35 "from farther" the different synthesis results look very much alike. *The one dimensional and two dimensional anamorphoses do not introduce much visual alteration at this scale.* Hence they do not succeed at hiding the repetitions.

Figure 5.21 – 1) RPN loses line structures. PS homogenizes the input. EF verbatim copy. LG satisfying for this input. 2) RPN loses line structures. PS homogenizes the input. EF satisfying for this input. LG garbage growing. 3) RPN loses line structures. PS satisfying for this input. EF satisfying for this input. LG garbage growing. 4) RPN loses line structures. PS homogenizes the input. EF garbage growing. LG garbage growing. 5) RPN loses line structures. PS homogenizes the input and color artifact. EF verbatim copy and garbage growing. LG verbatim copy.

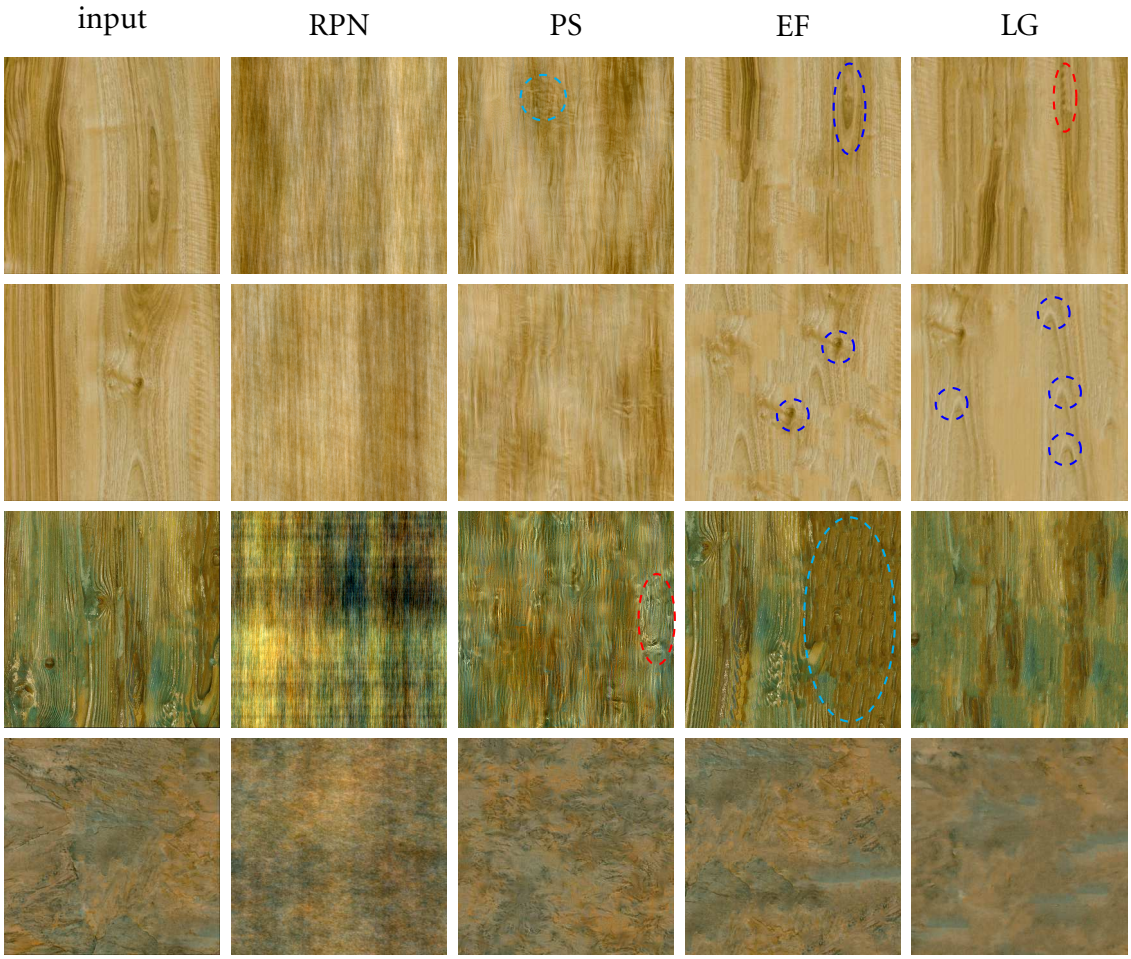


Figure 5.22 – 1) RPN and PS lose all geometric structures. PS creates some blur. EF and LG generate a risk of verbatim copy and garbage growing. 2) RPN and PS lose all geometric structures. EF and LG generate a risk of verbatim copy. 3) RPN loses all geometric structures. PS homogenizes the input and creates some blur. EF generates a risk of garbage growing. LG loses some resolution. 4) RPN loses geometric structures. PS homogenizes the input. EF is satisfying for this input. LG loses resolution.

Figure 5.23 – Both LG+EF and LG+PS present satisfying results. LG+PS is more accurate recovering the original granularity of the input. 2) Both LG+EF and LG+PS present satisfying results. Although LG+PS is more accurate recovering the original granularity of the input. 3) In this case the first LG synthesis presented garbage growing. Combining PS to LG is better in recovering from this drawback compared to LG+EF where the garbage growing remains a risk. 4) The first LG synthesis presented garbage growing (it is only the right part of the example that is emulated). The combination of PS with LG is once again better for recovering these drawbacks compared to LG+EF. 5) You can see in LG+EF that the initial LG synthesis did verbatim copies of the white stripes three times. Combination with EF does not remove this. Combination with PS hides this drawback.

Figure 5.24 – 1) The first LG synthesis presented garbage growing. It is noticeable in the result of LG+EF. The combination of PS with LG is once again better adding the unexplored zones of the input. 2) The first LG synthesis presented garbage growing and low resolution. The combination LG + EF does not recover from these drawbacks while the combination of PS with LG is very satisfying. 3) For this input both results are satisfying. LG+PS has a better recovery of the stripes. 4) The input is a "global texture" with *conspicuous global geometric organization*. We observe that even LG+PS *fails* to reproduce such global structure.

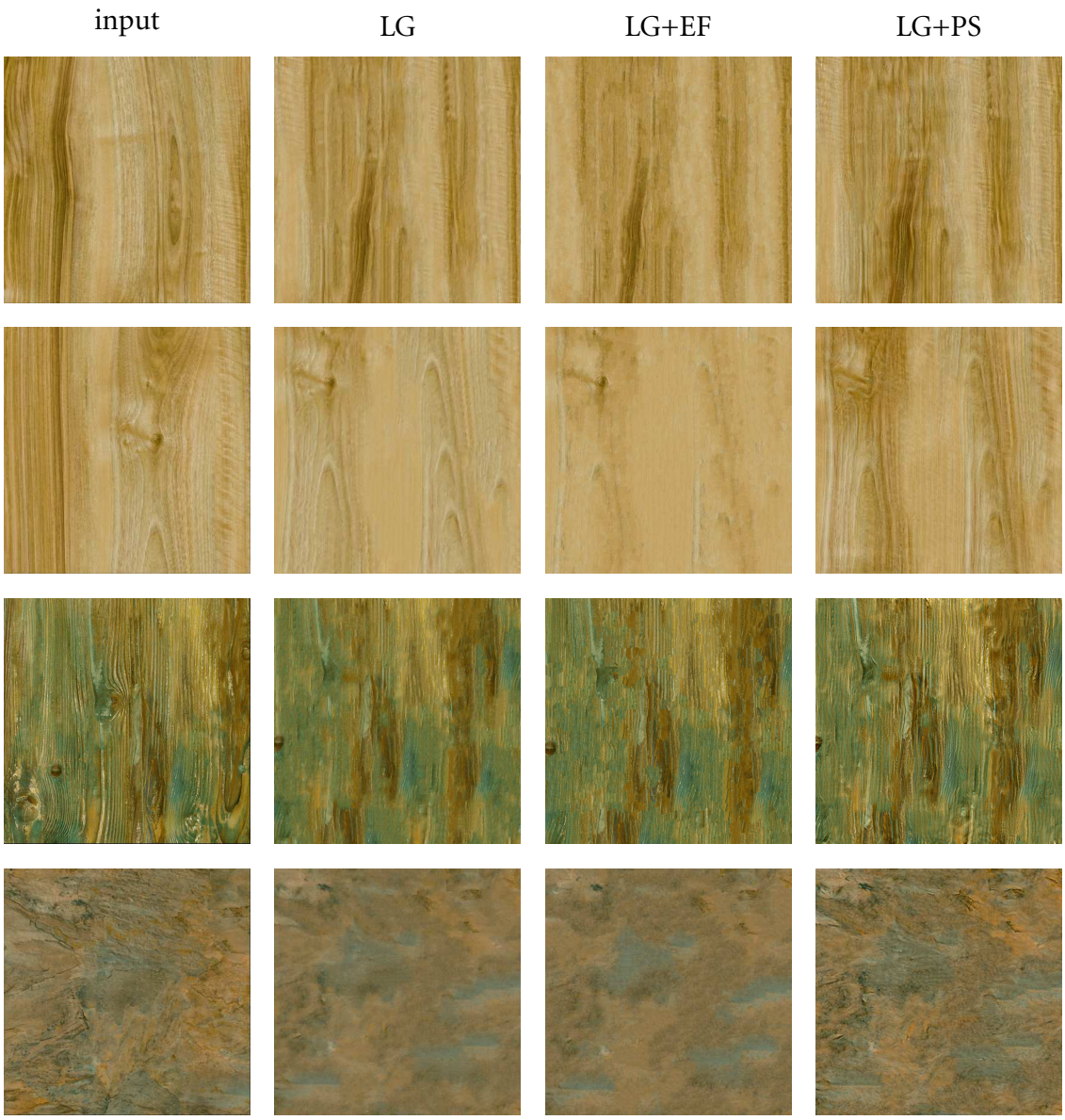


Figure 5.25 – For all fur examples, the inputs are a "global texture" with *conspicuous global geometric organization*. We observe that even LG+PS *fails* to reproduce these global structures.

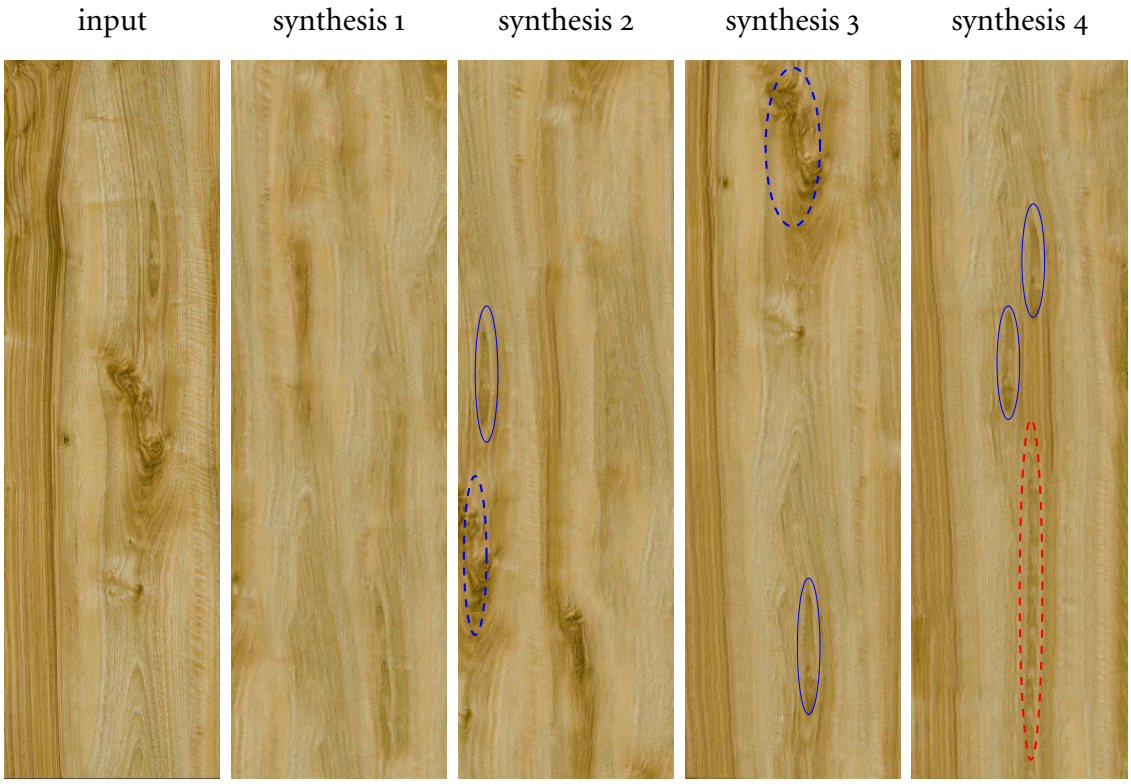


Figure 5.26 – One observes in the different synthesis results the verbatim copies of some particular global organizations from the input and the garbage growing. In some favourable cases the algorithm avoids scanning the regions with these particular organization (synthesis 1).

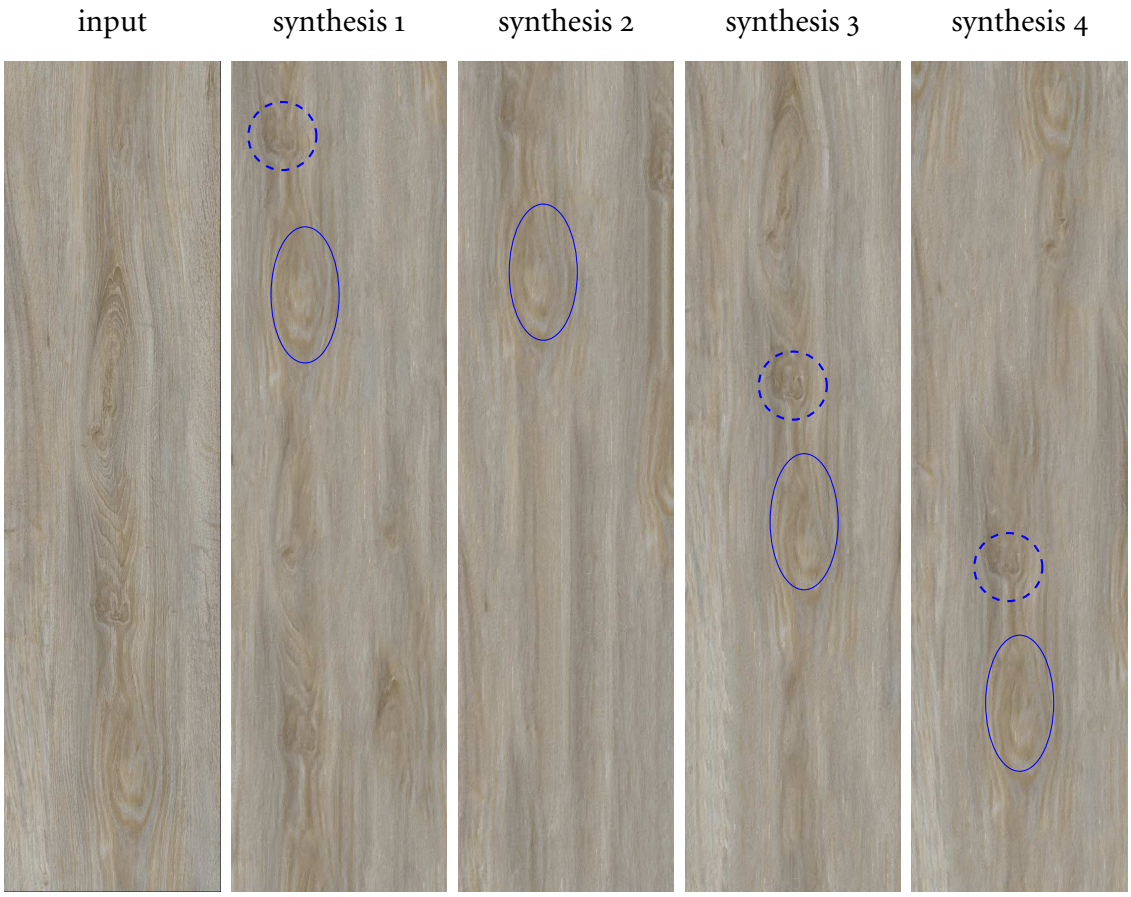


Figure 5.27 – One observes in the different synthesis results the verbatim copies of some particular global organizations from the input and the garbage growing.

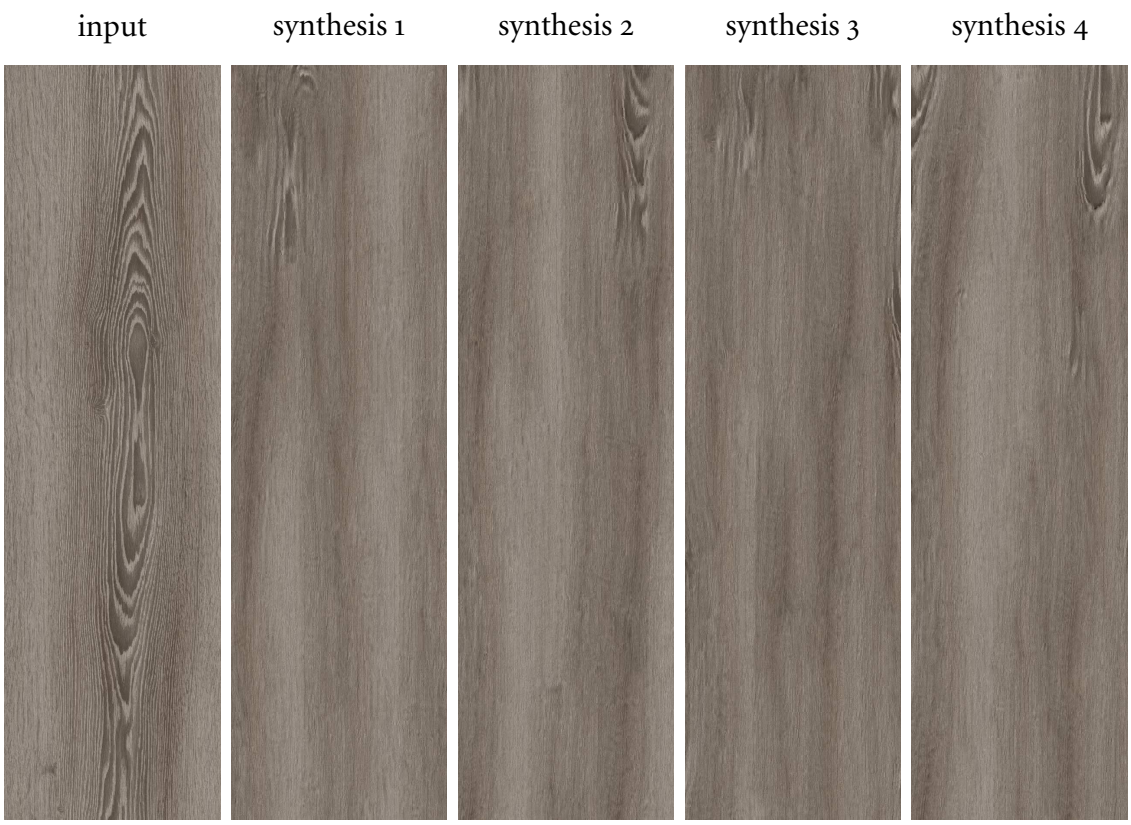


Figure 5.28 – One observes in the different synthesis results that all of them are not able to recover the central organization of the input sample. This is typically due to the garbage growing effect. The algorithm is stuck in some zone of the input sample.

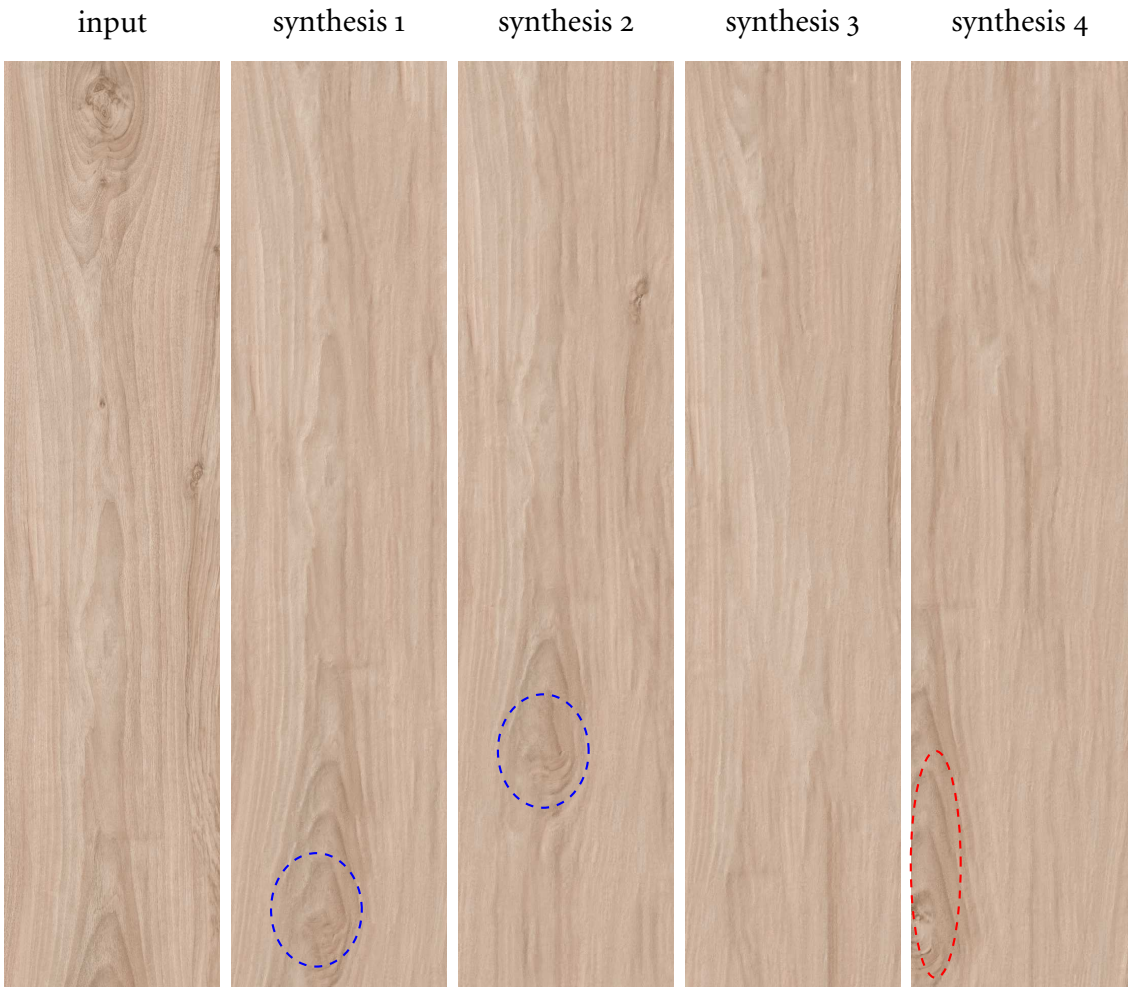


Figure 5.29 – One observes in the different synthesis results the verbatim copies of some particular global organizations from the input and the garbage growing. In some favourable cases the algorithm avoids scanning the regions with these particular organization (synthesis 3).

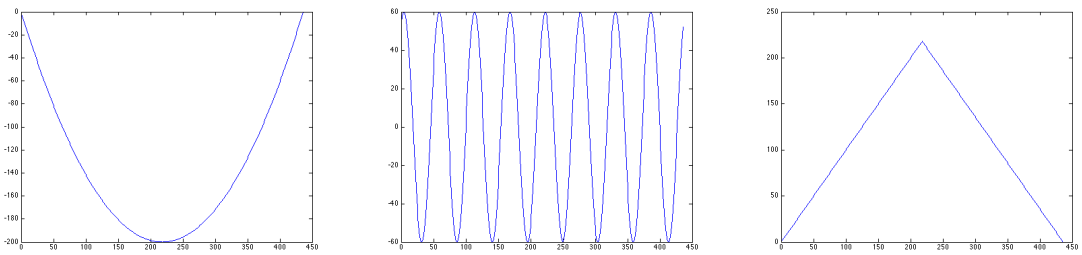


Figure 5.30 – One dimensional anamorphoses functions used to lightly deform the texture images aiming at masking the evident salient structures.

Figure 5.31 – One dimensional anamorphism applied to a wood texture. For the parabola and affine anamorphosis the global organizations of the input sample are slightly modified. For the sinusoid anamorphosis a stronger deformation is noticeable. In general the deformed images remain very similar to the input. *Even for strong anamorphoses the risk of repetitions remains.*

Figure 5.32 – One dimensional anamorphism applied to a wood texture. For the parabola and affine anamorphosis the global organizations of the input sample are slightly modified. For the sinusoid anamorphosis a stronger deformation is noticeable. In general the deformed images remain very similar to the input. *Even for strong anamorphoses the risk of repetitions remains.*

Figure 5.33 – One dimensional anamorphism applied to a wood texture. For the parabola and affine anamorphosis the global organizations of the input sample are slightly modified. For the sinusoid anamorphosis a stronger deformation is noticeable. In general the deformed images remain very similar to the input. *Even for strong anamorphoses the risk of repetitions remains.*

Figure 5.34 – One dimensional anamorphism applied to a wood texture. For the parabola and affine anamorphosis the global organizations of the input sample are slightly modified. For the sinusoid anamorphosis a stronger deformation is noticeable. In general the deformed images remain very similar to the input. *Even for strong anamorphoses the risk of repetitions remains.*

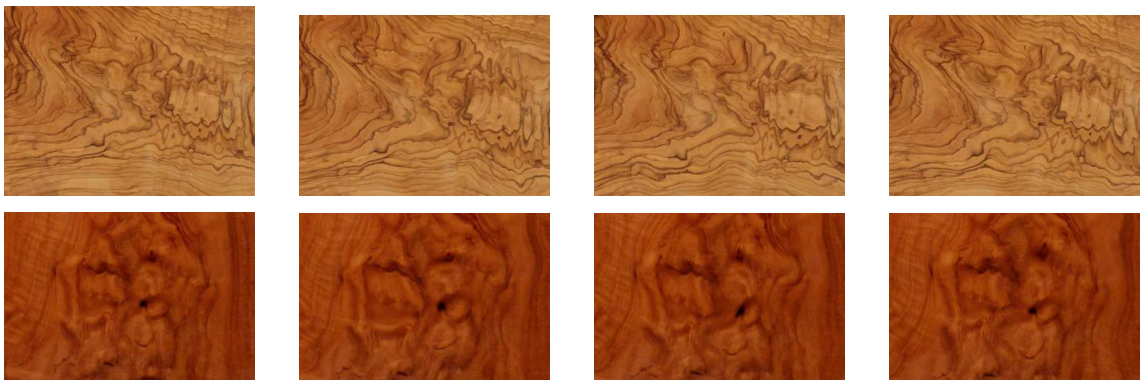


Figure 5.35 – Two dimensional anamorphism applied to two different wood textures. The deformed images remain very similar to the input.

5.6 Random mosaics

Another approach we considered to avoid evident repetitions between synthesis and within one same synthesized image is the random mosaic method. This experiment consists in slicing the input image vertically (or horizontally), randomly permuting these slices and after blending together the pieces using a stitching step to recompose a new texture image. The stitching technique can for example use the approach of image editing in [47]. The advantage of this approach is that no "unnatural" repetition of the global configurations will be observed since each part is used once. We also hope that the slicing succeed to separate the salient structures but not up to the point where the salient and less salient structures would be completely lost. This approach restricts the new sample to have the same size as the input and the number of possible permutations is limited. A possibility to overcome these limitations was to consider more than one input image in order to generate one new texture. There is no reason to limit ourselves to the use of a single input sample as it is done up to now in the exemplar-based methodologies.

The following experiences show three variants of this approach. The first one called the one dimensional random mosaic consists in slicing the image in one direction direction (horizontal or vertical). The second variant, the two dimensional random mosaic, consists in slicing the input in both directions, thus generating patches that are put together after being randomly permuted. This variant is suitable for textures that are isotropic like the stones and metal images that we presented. For the wood textures this variant is not of interest since there is an evident dominant direction and partitioning the image in both directions would completely alter the nature of the input image. The last variant, the multi-random mosaic, consists in using several inputs and then applying the one dimensional or two dimensional random mosaic, randomly permuting all the pieces and constructing a new images by picking some of the pieces.

Applying the random mosaic algorithm yields very good results for the case of the stone and metal textures. The generated images are all very different from one synthesis to another. Controlling the size of the slicing is important in order not to destroy the characteristic structures of these images. However for the wood texture this approach is not suitable. All global organization are lost and some of the salient structures are recognizable from one synthesis to another. For the case where the random mosaic approach works one can as well observe that the generated textures look different at all scales seeing the textures from far away still yields visual results that are different.

5.7 Synthesizing low frequencies

One of the difficulties of the input images given in section 5.2 is to correctly synthesize their low frequencies, i.e. their global organizations. We have seen with the previous experiments that using a multiscale approach is not satisfactory, since repetitions are still visible at lower scales hence propagated to the higher scales. Ideally we aim at generating a low resolution image having the "natural" amount of global configurations on which we gradually add the correct granularity of the different scales.

For this purpose we suggest to separate the input image I into two components: the low and high frequency images. We denote by G_θ a Gaussian kernel and define as the low frequency image $I_{\mathrm{LF}} = I * G_\theta$ and as the high frequency component $I_{\mathrm{HF}} = I - I * G_\theta$. We follow by applying the RPN method [21] to the low frequency component I_{LF}. In other terms, we add a random phase σ to the Fourier transform $\hat{I}_{\mathrm{RPN}} \leftarrow \hat{I}_{\mathrm{LF}} \mathrm{e}^{i\sigma}$ where σ is a white noise image niformly distributed over $[-\pi, \pi]$. This new low frequency image is denoted as $\tilde{I}_{\mathrm{RPN}}$. To recompose the new texture we simply cumulate the low and high frequencies components $\tilde{I} = \tilde{I}_{\mathrm{LF}} + I_{\mathrm{HF}}$.

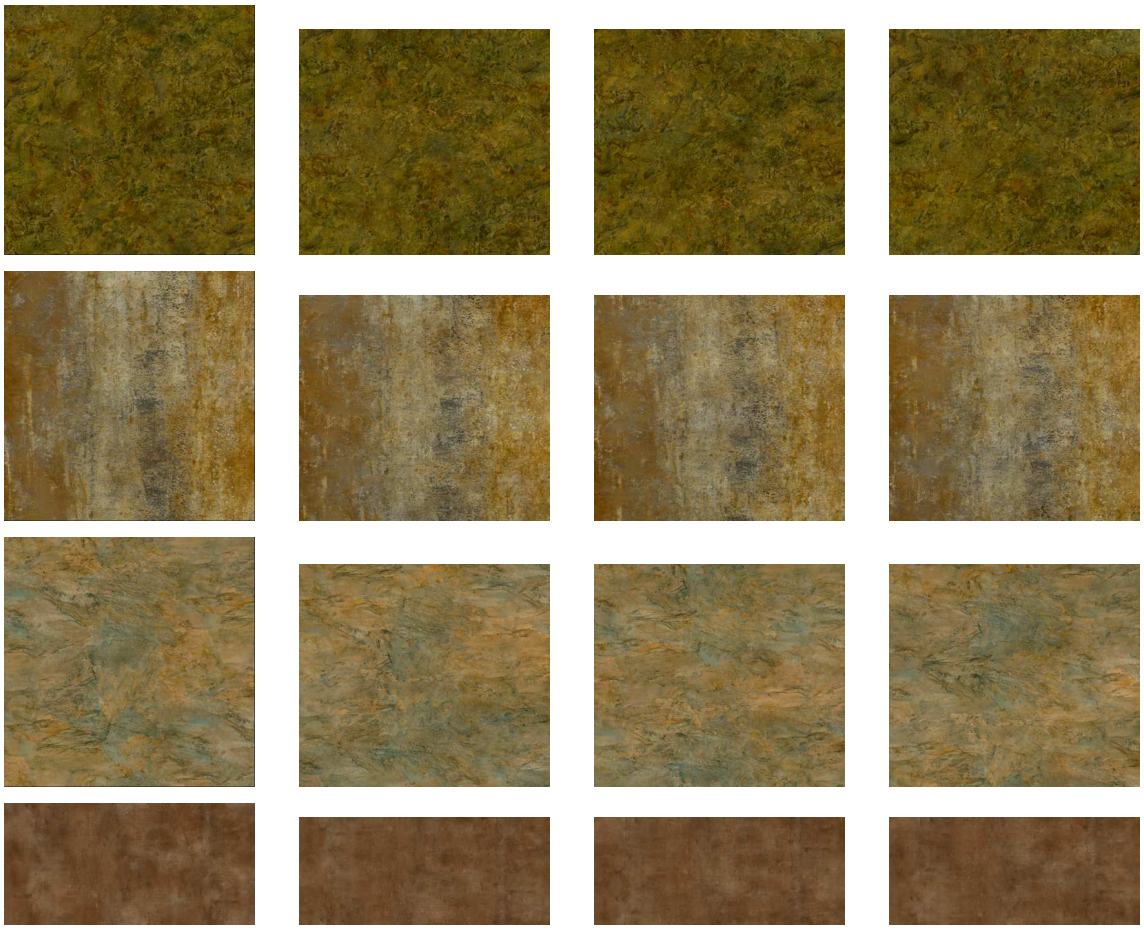


Figure 5.36 – Results applying the one dimensional random mosaic algorithm to stone and metal textures. From left to right: input, synthesis 1, synthesis 2 and synthesis 3.

Figure 5.37 – Results applying the one dimensional random mosaic algorithm to a wood texture. From top to bottom: input, synthesis 1, synthesis 2 and synthesis 3.

Figure 5.38 – Results applying the one dimensional random mosaic algorithm to a wood texture. From top to bottom: input, synthesis 1, synthesis 2 and synthesis 3.

Figure 5.39 – Results applying the one dimensional random mosaic algorithm to a wood texture. From top to bottom: input, synthesis 1, synthesis 2 and synthesis 3.

Figure 5.40 – Results applying the one dimensional random mosaic algorithm to a wood texture. From top to bottom: input, synthesis 1, synthesis 2 and synthesis 3.

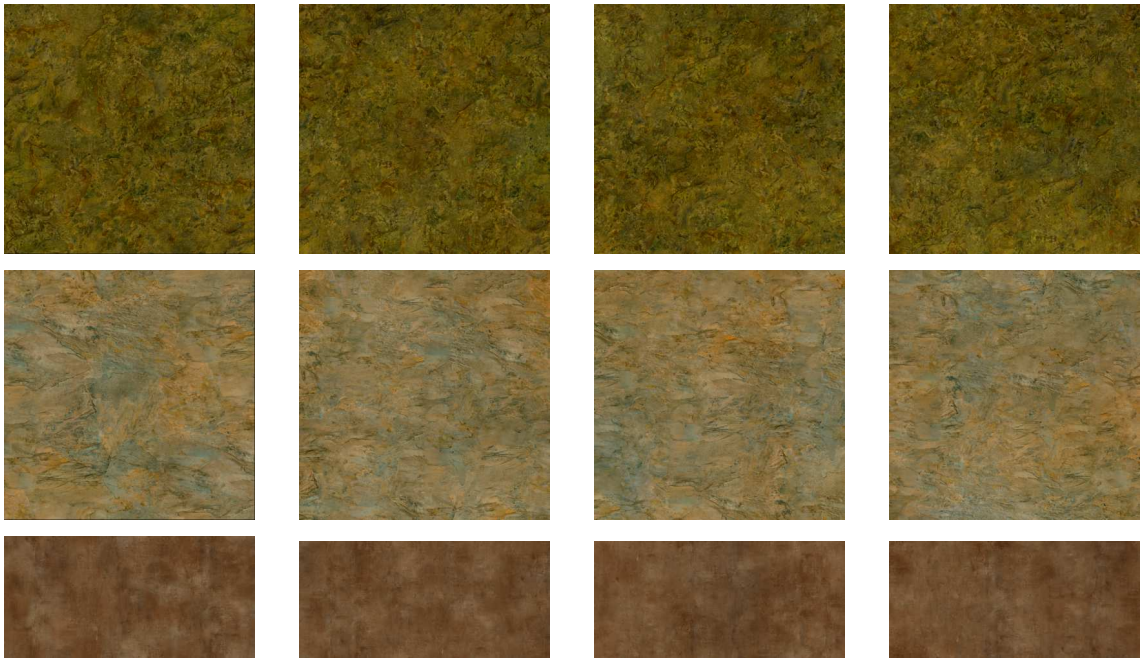


Figure 5.41 – Results applying the two dimensional random mosaic algorithm to stone and metal textures. From left to right: input, synthesis 1, synthesis 2 and synthesis 3.

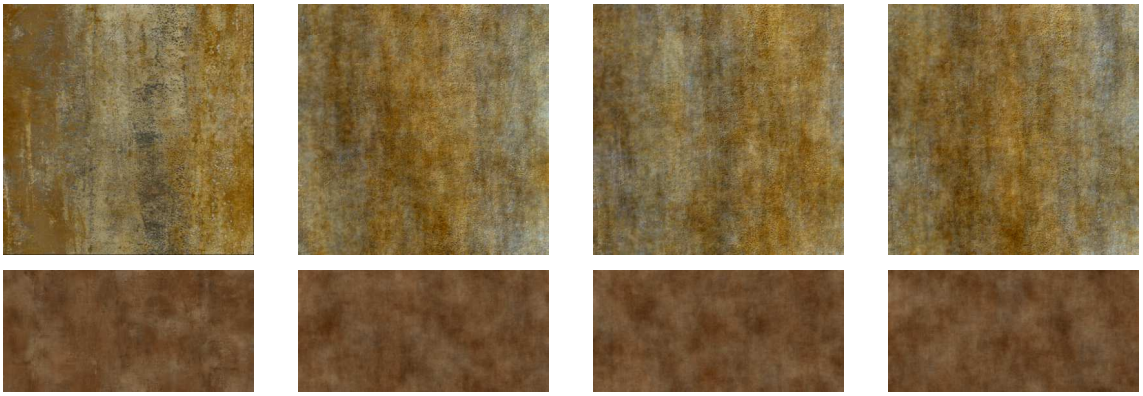


Figure 5.42 – Synthesis results after randomizing the low frequencies of the input sample. From left to right: input, synthesis 1, synthesis 2 and synthesis 3. A strong correlation exists between the low and high frequencies of these input images. Observing these results from far reduces this correlation.

We can conclude from Figure 5.42 that separating the high and low frequencies and randomizing the low frequency component does not yield convincing results given that the low and high frequencies of these images are highly correlated. Nevertheless *this method works on these metal examples at lower resolution inputs* as can be observed in Figure 5.42 when seen from far.

5.8 Mixing texture images

The last experiment that we considered consisted in mixing two texture images. Ideally one of the textures denoted as I_1 should be very sober and the second texture I_2 would have no restriction. Let us consider the case of wood textures. This could allow us to dispose of several sober images of a same essence of wood on which we will add different particular structures taken from a second image. This approach is clearly limited with respect to the number of images one can generate. This approach is considered as a tool which allows us to add some variations on the generated textures. To achieve the texture mixing we used the Poisson editing work described in [47].

The goal of the Poisson editing technique [47] is to seamlessly edit and clone selected regions of an image. The mathematical tool at the heart of the approach is the Poisson partial differential equation with Dirichlet boundary conditions. Given two images I_1 and I_2, the goal is to replace a region of I_1 with one of I_2. The domain to replace in I_1 is denoted by Ω and $\delta\Omega$ its boundary. Then the solution that we denote as I minimizes the following equation

$$\begin{cases} \nabla I(x) = div(\vec{I_2}) & \text{if } x \in \Omega \\ I(x) = I_1(x) & \text{if } x \in \delta\Omega. \end{cases}$$

Another application of the Poisson editing method [47] is to add or remove details to an image mixing their gradients retaining the stronger variation at each pixel of the image. This is what we use to mix two different texture images

$$\nabla I(x) = \begin{cases} \nabla I_1 & \text{if } \nabla I_1 \geq \nabla I_2 \\ \nabla I_1 & \text{if otherwise.} \end{cases} \tag{5.1}$$

In Figures 5.43 and 5.44 texture mixing results are shown. For each example two input textures I_1 ans I_2 are provided where one is much sober than the other. The resulting image I is then

Figure 5.43 – Texture mixing results. Three different examples are shown. Each example corresponds to three images: inputs I_1 and I_2 and I resulting from mixing I_1 and I_2.

obtained by applying (5.1). The results obtained are interesting. However the synthesized images are still quite similar to the input ones.

5.9 Conclusion

In this chapter we have presented an extensive analysis on how to imitate large natural textures and explored the limitations of this endeavour. The texture images considered in this chapter were high resolution photographies of different materials such as wood, metals and stones. The study shows that, in spite of the remarkable progress that have been done in the area of exemplar-based texture synthesis methods, they are posed for a very particular problem that is using stationary and small samples of image textures. The existing methods are still incapable of dealing with such complex textures containing strong global configurations and being non-stationary.

We have tested several methods and combination of methods. In general the results achieved on small crops of the given input images are satisfying. In particular, the combination of the locally Gaussian method with the Portilla and Simoncelli method yields satisfying results for small images. For bigger images presenting strong global organizations all methods fail. They cannot work on highly structured and non-stationary images. Insisting on using such exemplar-based methods leads to repetitions of the conspicuous structures.

We tried other alternatives such as applying random permutations on different parts of the input image or applying one dimensional and two dimensional geometric deformations to mask the conspicuous structures but none of these successfully hide the repetitions to the human eye. This conclusion is final in the case of the wood textures who have very strong salient configurations.

Figure 5.44 – Texture mixing results. Three different examples are shown. Each example corresponds to three images: inputs I_1 and I_2 and I resulting from mixing I_1 and I_2.

The one dimensional and two dimensional random mosaic method returns satisfactory results on some samples.

As a more general conclusion when trying to synthesize samples that are not stationary and very large the exemplar-based methods fail. Patch based methods yield too many repetitions of the salient structures. The risk of growing garbage exists due to the strong non-stationarity. Methods like the Portilla and Simoncelli are limited by the large size of the input. Combining the Portilla and Simoncelli algorithm with the locally Gaussian method works, up to a given resolution. Afterwards the methods tend to homogenize the input; i.e. will start "mixing" the different structures present destroying them. We deduced that using a single wood texture sample, it is almost impossible to generate several different new images from this example. It is very difficult to randomly create knots with just one example without repeating it. This is a big limitation for exemplar based methods. The exemplar based methods usually work because the examples images present a lot of auto similarities that can be used on the one hand to infer a stochastic process defining the texture or on the other hand to estimate the models of a patch-based method.

A Efros and Freeman Image Quilting Algorithm for Texture Synthesis

Exemplar-based texture synthesis is defined as the process of generating, from an input texture sample, new texture images that are perceptually equivalent to the input. Efros and Freeman's method is a non-parametric patch-based method which computes an output texture image by quilting together patches taken from the input sample. The main innovation of their work relies in the stitching technique which significantly reduces the transition effect between patches. In this paper, we propose a detailed analysis and implementation of their work. We provide a complete mathematical description of the linear programing problem used for the quilting step as well as implementation details. Additionally we propose a partially parallel version of the quilting technique.

A.1 Introduction

Texture synthesis is a classical image processing problem that finds its applications in virtual reality rendering (video games, animation movies, ...). Given an input texture image, it consists in producing output texture images that are both visually similar to and pixel-wise different from the input, and having possibly a larger size. One can separate texture synthesis algorithms into two categories, namely statistical constraint approaches and non-parametric patch-based methods, although "hybrid" algorithms have been proposed recently [59] and the algorithm we propose in Chapter 2.

Statistical constraint texture synthesis algorithms model the texture based on statistical and/or perceptual considerations. They generally involve two different steps, one for analysis and one for synthesis. The analysis step consists in estimating a set of relevant statistics from the input texture image. The synthesis step computes an image that satisfies the statistical constraints estimated during the analysis step. Following the seminal paper of Heeger and Bergen [26, 6], several methods are based on statistics of wavelet coefficients or more involved multiscale image representations [50, 48, 53]. Another approach initially proposed by van Wijk [60] consisting in randomizing the Fourier phase of an image has been extended to an exemplar-based synthesis method in [21, 20] suitable for micro-textures (textures that do not present salient geometric patterns and that are not constituted of individual discernible objects).

Non-parametric patch-based algorithms attempt at producing a new texture by arranging local neighborhoods of the input texture in a consistent way. The first methods of this kind are sequential algorithms that create a new texture one pixel at a time [14, 63]. These algorithms represented

a breakthrough in the field since they are able to reproduce macro-textures with specific geometric structures, see e.g. the IPOL paper [1]. Along with the method of Liang *et al* [42] developed at the same period, image quilting [15], the algorithm investigated in this paper, computes the new texture by arranging seamlessly full patches of the input texture. The main innovation of this method is the procedure to stitch a new patch in the sequentially built output texture to avoid discontinuities as much as possible. This is achieved by computing an optimal boundary cut between the patch and the synthesis area thanks to a linear programming optimization (see Section A.2 for details). Let us note that another solution of this "stitching step" is proposed in [**?**] using graphcuts. Many contributions have since improved the results of image quilting, at least regarding the computational cost, and we refer to the state of the art [62] for a more complete survey of this category of texture synthesis algorithms. Let us also mention the recent paper [35], based on [**?**, 10], that discusses and atempts to solve some limitations of these approaches.

The plan of the paper is as follows. Section A.2 describe in detail all the steps of the algorithm and in particular gives a detailed mathematical description of the linear programing problem for the computation of the minimum error boundary cut. Section A.3 gives implementation details and discuss a parallelization of the Efros-Freeman algorithm which, to the best of our knowledge, is a contribution of this paper. Section A.4 presents numerous experiments of our implementation of the Efros-Freeman algorithm. This experimental section shows that this algorithm produces most of the time visually good results. However, a set of failure cases shows that the shortcomings of the Efros-Leung algorithm, that is, garbage growing and verbatim copy [14, 1], are also present in the Efros-Freeman results, but only at a larger scale.

A.2 Algorithm Description

In [15] the authors propose a sequential patch-based algorithm to synthesize textures. I_0 and I_s denote the input sample and the output texture respectively. The output image I_s is constructed patch by patch in a raster scan order. The goal of each iteration is to fill a patch P_{old} of I_s that is only partially defined on a region called overlap region (see Figure A.3). To do so a patch P_{in} of I_0 that matches P_{old} on the overlap region is randomly selected. An optimal boundary cut between P_{old} and P_{in} is then computed within the overlap region. This optimal boundary cut is used to construct the new patch P_{new} by blending P_{old} and P_{in} along the cut.

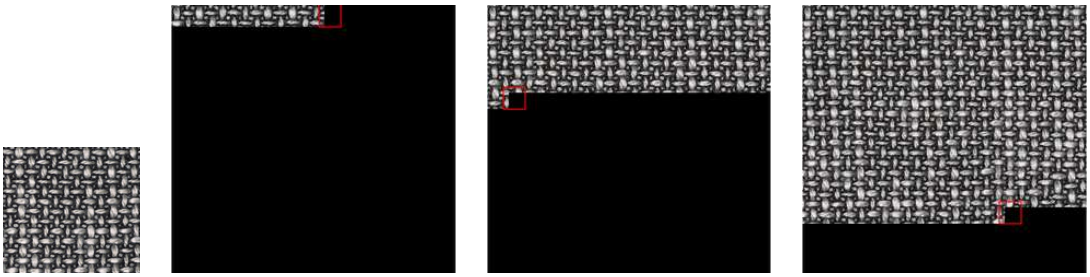
The whole image quilting algorithm is described in Algorithm 12 and the remaining of this section will detail each step of the algorithm, namely the initialization, the patch search procedure to select P_{in}, the computation of the minimum error boundary and the blending procedure along the boundary.

A.2.1 Initialization

The first step of the algorithm is to initialize I_s. For that a random patch P_{in} of size $w_p \times w_p$ is taken from I_0 and placed at the top-left corner of I_s.

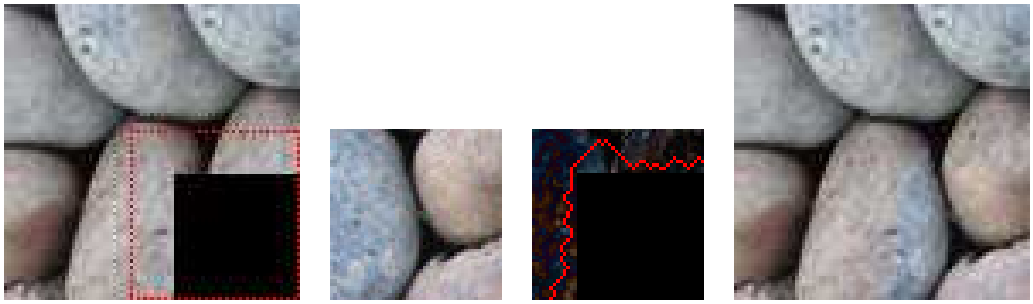
A.2.2 Patch Selection

Once the image has been initialized the algorithm synthesizes the remaining patches of I_s sequentially in raster scan order. At each iteration one has to fill a patch P_{old} of I_s that is only defined on an overlap region of width w_o. Note that there are three possible overlap regions: vertical overlap



input iteration 10 iteration 87 iteration 217

Figure A.1 – Three different iterations of the synthesis process are shown. At each iteration a patch is being synthesized. This patch is represented by the pink square for the three cases.



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

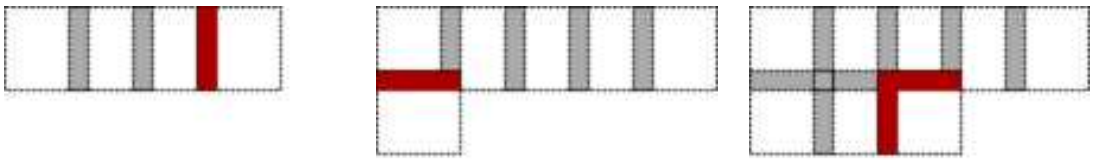
Figure A.2 – Quilting a square patch from the input texture into the synthesized texture. (a) sub-part of the synthesized texture. The red dotted zone shows where the new patch will be quilted. (b) patch to quilt in the red dotted zone shown in (a). (c) error surface between the patch in (b) and the red dotted zone in (a). The red path shows the minimum error boundary. (d) the patch in (b) is quilted along the minimum error boundary in the corresponding zone showed in (a).

Algorithm 12: Image Quilting

1 **Input:** Sample texture I_0, patch size w_p, overlap size w_o, tolerance parameter ε, output/input size ratio r

2 **Output:** Synthesized texture I_s

```
Initialize I_s
for each patch P_old in I_s do
    Select a compatible patch P_in ∈ I_0 using the Patch Selection algorithm (see Algorithm 13)
    Compute minimum error boundary cut between P_old and P_in (see Algorithm 14)
    Construct the patch P_new by blending P_old and P_in along the boundary cut (see Equation (A.4))
    Replace P_old with P_new within I_s
end for
```



(a) Vertical overlap (b) Horizontal overlap (c) L-shaped overlap

Figure A.3 – Three overlap cases arises in the raster scan order: vertical overlap for the first row, horizontal overlap for the first column, and L-shaped overlap everywhere else.

for the first row, horizontal overlap for the first column, and L-shaped overlap everywhere else (see Figure A.3).

To select a patch P_{in} of the input image I_0 one computes the square distance between the overlap region of the patch P_{old} of I_s and the corresponding regions of all the patches of I_0. The minimal distance d_{min} is determined and P_{in} is randomly picked among all patches whose distance to P_{old} is lower than $(1+\varepsilon)d_{\text{min}}$ where ε is the tolerance parameter.

To conclude this section let us give a detailed expression of the distance used to compare patches. A patch of I_0 is represented by the position of its top-left corner $(m,n) \in \{0,\dots,M_0 - w_p\} \times \{0,\dots,N_0 - w_p\}$. The squared distance image D contains at each position (m,n) the distance between P_{old} and the patch from I_0 who's top-left corner is (m,n) according to some binary weight Q that equals one in the overlap region and zero otherwise. More precisely for all $(m,n) \in \{0,\dots,M_0 - w_p\} \times \{0,\dots,N_0 - w_P\}$, one has

$$D(m,n) = \sum_{i,j=0}^{w_p-1} Q(i,j)(P(i,j) - I_0(m+i,n+j))^2. \tag{A.1}$$

The computation of the squared distance image D is discussed in Section A.3.1 and the whole patch selection procedure is summarized in Algorithm 13.

Algorithm 13: Patch Selection

1 **Input:** Sample texture I_0, patch under construction P_{old}, patch size w_p, tolerance $\varepsilon > 0$, binary weight Q defining the overlap region

2 **Output:** Patch position (m, n)

1: Compute the squared distance image D (see Equation (A.1)) containing the distances between the patch P_{old} and all patches of I_0.

2: $d_{\text{min}} \leftarrow \min_{(k,l)} D(k, l)$.

3: Uniformly draw a patch position (m, n) among the set $\{(k, l),\ D(k, l) < (1 + \varepsilon)d_{\text{min}}\}$.

A.2.3 Minimum Error Boundary Cut

This step of the algorithm is the main contribution of the Efros-Freeman algorithm [15].

The Patch Search step (see Section A.2.2) gives a patch P_{new} of I_0 having coordinates (m, n) that is similar to the partially defined current patch P_{old} in their overlap region. We recall that the overlap regions that arise in the raster scan order are of three types: vertical, horizontal, and L-shaped overlap. These three cases are illustrated in Figure A.3. To get the final patch P one must combine the patches P_{old} and P_{new}. Denoting Q the binary weight for the overlap regions as in the previous section, then, for any binary image M such that $0 \leq M \leq Q$, P can be defined as the combination

$$P_{\text{new}} = MP_{\text{old}} + (1 - M)\, P_{\text{in}}.$$

The main idea of Efros and Freeman [15] is looking for a binary shape M where the transition between P_{old} and P_{new} along the boundary of the shape is minimal. For simplicity, and to be able to use linear programming, the authors do not allow for any shape, but only for the ones whose boundaries are simple forward paths from one end to the other of the overlap region. This results in two pieces of image that are sewn together along some general boundary path, hence the algorithm name "quilting". In the remaining of this section, we describe in details the computation of the minimum error boundary cut for the three overlap cases, starting with the vertical and horizontal cases.

Connected Paths and Boundary Error

In the following, we call a *path* of length $\ell \geq 1$ in $\{0, \ldots, w_p - 1\}^2$ any ordered sequence $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_{\ell-1})$ of 8-connected pixels $\gamma_k \in \{0, \ldots, w_p - 1\}^2$, that is, for all $k \in \{0, \ldots, \ell - 2\}$, $\max(|\gamma^1_{k+1} - \gamma^1_k|, |\gamma^2_{k+1} - \gamma^2_k|) = 1$ (for convenience, within this section the first and second coordinates of a pixel γ_k are denoted by γ^1_k and γ^2_k).

Let us recall that we use matrix coordinates for the pixels. Hence a path γ of length ℓ is said to be *vertical* if for all $k \in \{0, \ldots, \ell - 2\}$, $\gamma^1_{k+1} - \gamma^1_k = -1$ (vertical paths are oriented from bottom to top), and *horizontal* if for all $k \in \{0, \ldots, \ell - 2\}$, $\gamma^2_{k+1} - \gamma^2_k = -1$ (horizontal paths are oriented from right to left).

Denoting by $e(i, j) = (P_{\text{new}}(i, j) - P_{\text{old}}(i, j))^2$ the squared difference between the two patches P_{old} and P_{new}, the boundary error of a path γ of length ℓ is defined by

$$\mathcal{E}(\gamma) = \sum_{k=0}^{\ell-1} e(\gamma_k).$$

Vertical Boundary Cuts

In this section we explain how the optimal boundary between P_{old} and P_{new} is defined and computed in the case of vertical overlap, that is when the overlap region is the rectangle $\{0, \ldots, w_p - 1\} \times \{0, \ldots, w_o - 1\}$.

The optimal boundary is defined as the vertical path γ that minimizes the boundary error $\mathcal{E}(\gamma)$ while joining both ends of the overlap regions. More precisely, define the admissible vertical paths as

$$\Gamma_v = \{\gamma = (\gamma_0, \ldots, \gamma_{w_p-1}),\ \forall k,\ \gamma_k^1 = w_p - 1 - k \text{ and } \gamma_k^2 \in \{0, \ldots, w_o - 1\}\}$$

(one notices that paths of $\gamma \in \Gamma_v$ are indeed vertical since $\gamma_{k+1}^1 - \gamma_k^1 = -1$). Then an optimal boundary is defined as any solution of the optimization problem

$$\min_{\gamma \in \Gamma_v} \mathcal{E}(\gamma). \tag{A.2}$$

Problem (A.2) is solved using dynamic programming. This is possible because Problem (A.2) verifies the principle of optimality: if $(\gamma_0, \ldots, \gamma_{w_p-1})$ is an optimal solution of Problem (A.2), then, for all $0 \leq r \leq w_p - 1$, the subpath $(\gamma_0, \ldots, \gamma_{r-1})$ is an optimal vertical path to reach the r-th point γ_{r-1} starting from the bottom of the overlap region.

Let us now discuss in detail the dynamic programming method to solve Problem (A.2). For all points $(i, j) \in \{0, \ldots, w_p - 1\} \times \{0, \ldots, w_o - 1\}$, denote by $\Gamma_v^{(i,j)}$ the set of vertical paths γ that start at the bottom side of the overlap region and end at the point (i, j), that is

$$\Gamma_v^{(i,j)} = \{\gamma = (\gamma_0, \ldots, \gamma_{w_p-1-i}),\ \forall k,\ \gamma_k^1 = w_p - 1 - k,\ \gamma_k^2 \in \{0, \ldots, w_o - 1\}, \text{ and } \gamma_{w_p-1-i} = (i, j)\}.$$

Now, for all points $(i, j) \in \{0, \ldots, w_p - 1\} \times \{0, \ldots, w_o - 1\}$, define $E_v(i, j)$ as the minimal cumulative vertical error to reach (i, j) starting from the bottom side, that is,

$$E_v(i, j) = \min_{\gamma \in \Gamma_v^{(i,j)}} \mathcal{E}(\gamma).$$

Then, since

$$\Gamma_v = \bigcup_{j \in \{0, \ldots, w_o - 1\}} \Gamma_v^{(0,j)},$$

one has

$$\min_{\gamma \in \Gamma_v} \mathcal{E}(\gamma) = \min_{j \in \{0, \ldots, w_o - 1\}} E_v(0, j).$$

Now, remark that the last-but-one point of an optimal vertical path that ends at (i, j) is one of the (at most) three points below (i, j), namely the points $(i+1, j-1)$, $(i+1, j)$, $(i+1, j+1)$ (there can be only two neighboring points if (i, j) is at the border, that is, if $j = 0$ or $j = w_o - 1$). Hence, for all $i \in \{0, \ldots, w_p - 2\}$, one has

$$E_v(i, j) = e(i, j) + \min(E_v(i+1, j-1), E_v(i+1, j), E_v(i+1, j+1)), \tag{A.3}$$

where by convention, if one of the index $j - 1$ or $j + 1$ is not a valid index, the corresponding term $E_v(i+1, j-1)$ or $E_v(i+1, j+1)$ is discarded from the minimum. Besides, for the last line $i = w_p - 1$, one simply has $E_v(w_p - 1, j) = e(w_p - 1, j)$ since the paths are only made of one pixel. Hence the dynamic programming procedure for solving Problem (A.2) is to compute the costs $E_v(i, j)$ line by line from bottom to top using Equation (A.3) recursively, and then search for the minimal value of the first line $j^* = \arg\min_j E_v(0, j)$. The full optimal path γ can then

be traced back starting from this coordinate $(0, j^*) = (\gamma^1_{w_p-1}, \gamma^2_{w_p-1})$. More precisely, if (i, j) are the coordinates of the point γ_{w_p-1-i} of the optimal path γ, then its preceding point $\gamma_{w_p-1-(i+1)}$ is given by

$$\gamma_{w_p-1-(i+1)} = \arg\min(E_v(i+1, j-1), E_v(i+1, j), E_v(i+1, j+1)).$$

In practice one stores the matrix $T_v(i, j) = \arg\min(E_v(i+1, j-1), E_v(i+1, j), E_v(i+1, j+1))$ while computing the vertical cumulative error E_v given by Equation (A.3) in order to trace back the path without additional computation since for $i = w_p - 2$ to 0, $\gamma_i = T_v(\gamma_{i+1})$.

Horizontal Boundary Cuts

The case of horizontal overlap is just the symmetric case of vertical overlap one. Still let us introduce the notation that will be necessary for dealing with the L-shaped overlap case. One defines the set of horizontal paths as

$$\Gamma_h = \{\gamma = (\gamma_0, \ldots, \gamma_{w_p-1}),\ \forall k,\ \gamma^2_k = w_p - 1 - k \text{ and } \gamma^1_k \in \{0, \ldots, w_o - 1\}\},$$

and for all points $(i, j) \in \{0, \ldots, w_o - 1\} \times \{0, \ldots, w_p - 1\}$, one defines

$$\Gamma^{(i,j)}_h = \{\gamma = (\gamma_0, \ldots, \gamma_{w_p-1-j}),\ \forall k,\ \gamma^2_k = w_p - 1 - k,\ \gamma^1_k \in \{0, \ldots, w_o - 1\}, \text{ and } \gamma_{w_p-1-j} = (i, j)\}$$

the set of paths that start at the right side of the overlap region and end at (i, j), as well as

$$E_h(i, j) = \min_{\gamma \in \Gamma^{(i,j)}_h} \mathcal{E}(\gamma)$$

the minimal cumulative horizontal error to reach (i, j) starting from the right side of the overlap region. Then $E_h(i, w_p - 1) = e(i, w_p - 1)$, and for all $j \in \{0, \ldots, w_p - 2\}$, one has the recursive relation

$$E_h(i, j) = \min(E_h(i-1, j+1), E_h(i, j+1), E_h(i+1, j+1))$$

which enables to compute $E_h(i, j)$ for all $(i, j) \in \{0, \ldots, w_o - 1\} \times \{0, \ldots, w_p - 1\}$. Since

$$\min_{\gamma \in \Gamma_h} \mathcal{E}(\gamma) = \min_{i \in \{0, \ldots, w_o - 1\}} E_h(i, 0)$$

the end point of the optimal horizontal path is $(i^*, 0)$ where $i^* = \arg\min_i E_h(i, 0)$ and the path can be traced back thanks to the matrix $T_h(i, j) = \arg\min(E_h(i-1, j+1), E_h(i, j+1), E_h(i+1, j+1))$.

L-shaped Boundary Cuts

The case of L-shaped overlap regions is slightly more complex than the vertical and horizontal cases since the geometry of the L-shape does not allow for a clear ordering of the pixels of the path. The original paper [15] only mentions that, in the L-shaped overlap case, "the minimal paths meet in the middle and the overall minimum is chosen for the cut". We propose below a rigorous interpretation of this sentence.

A separating path will start at the bottom side of the L-shape (that is, the points (i, j) with $i = w_p - 1$ and $j \in \{0, \ldots, w_o - 1\}$) and end at the right side of the L-shaped (that is, the points (i, j) with $i \in \{0, \ldots, w_o - 1\}$ and $j = w_p - 1$). To restrict the geometry of admissible paths (and allowing for dynamic programming), it is further required that an L-shaped path has to meet

at some diagonal point (i,i), $i \in \{0,\dots,w_o-1\}$, that the first part from the bottom side to this diagonal point (i,i) is a vertical path, and that the remaining part from the diagonal point (i,i) to the right side is a reversed horizontal path. More formally, for all indexes $i \in \{0,\dots,w_o-1\}$, one defines

$$\Gamma_L^i = \{\gamma = (\gamma_0,\dots,\gamma_{2(w_p-i)-1}),\ (\gamma_0,\dots,\gamma_{w_p-i-1}) \in \Gamma_v^{(i,i)} \text{ and } (\gamma_{2(w_p-i)-1},\dots,\gamma_{w_p-i-1}) \in \Gamma_h^{(i,i)}\},$$

and for all $\gamma \in \Gamma_L^i$ we denote by γ^v and γ^h its associated vertical and horizontal paths of $\Gamma_v^{(i,i)}$ and $\Gamma_h^{(i,i)}$ respectively. The set of the admissible L-shaped boundaries Γ_L is then defined as the disjoint union of the sets Γ_L^i, that is,

$$\Gamma_L = \bigcup_{i=0}^{w_0-1} \Gamma_L^i.$$

As before, we search for an optimal L-shaped boundary cut $\gamma \in \Gamma_L$ having minimal boundary error

$$\mathcal{E}(\gamma) = \sum_{k=0}^{\ell(\gamma)-1} e(\gamma_k),$$

where $\ell(\gamma)$ is the length of the path γ (which is equal to $2(w_p-i)-1$ if $\gamma \in \Gamma_L^i$). The optimal L-shaped boundary can be found in splitting the above sum into a vertical part and an horizontal part, since for all $\gamma \in \Gamma_L^i$ one has

$$\mathcal{E}(\gamma) = \mathcal{E}(\gamma^v) + \mathcal{E}(\gamma^h) - e(i,i).$$

Hence one has

$$\begin{aligned}
\min_{\gamma\in\Gamma_L} \mathcal{E}(\gamma) &= \min_{i\in\{0,\dots,w_o-1\}} \min_{\gamma\in\Gamma_L^i} \mathcal{E}(\gamma) \\
&= \min_{i\in\{0,\dots,w_o-1\}} \min_{\gamma\in\Gamma_L^i} \mathcal{E}(\gamma^h) + \mathcal{E}(\gamma^v) - e(i,i) \\
&= \min_{i\in\{0,\dots,w_o-1\}} \left[\left(\min_{\gamma^v\in\Gamma_v^{(i,i)}} \mathcal{E}(\gamma^v) \right) + \left(\min_{\gamma^h\in\Gamma_h^{(i,i)}} \mathcal{E}(\gamma^h) \right) - e(i,i) \right] \\
&= \min_{i\in\{0,\dots,w_o-1\}} E_v(i,i) + E_h(i,i) - e(i,i),
\end{aligned}$$

where E_v and E_h are the vertical and horizontal cumulative errors defined in the previous sections. Hence,

$$\min_{\gamma\in\Gamma_L} \mathcal{E}(\gamma) = \min_{i\in\{0,\dots,w_o-1\}} E_v(i,i) + E_h(i,i) - e(i,i).$$

The above equation enables us to determine the optimal position (i^*,i^*) on the diagonal of the optimal path γ once the matrices E_v and E_h have been computed recursively. We can then trace back the vertical part γ^v and the horizontal part γ^h of the optimal path γ using the matrices T_v and T_h. Let us remark that the "overall minimum" evoked in [15] must not be interpreted as the minimum of $E_v(i,i) + E_h(i,i)$ but the minimum of $E_v(i,i) + E_h(i,i) - e(i,i)$.

Algorithm 14: Minimum Error Boundary Cut

1 **Input:** Output texture I_s, patch P_{old} (from I_s), patch P_{in} (from I_0), patch size w_p, overlap size w_o

2 **Output:** Boundary cut γ

1: Compute the squared difference $e(i,j) = (P_{\text{new}}(i,j) - P_{\text{old}}(i,j))^2$
2: **switch** (overlap type)
3: **case** vertical**:**
4: Compute the minimal cumulative vertical error E_v:
5: $E_v(w_p - 1, j) = e(w_p - 1, j), \quad j \in \{0, \ldots, w_o - 1\}$
6: **for** $i = w_p - 2$ to 0 **do**
7: $E_v(i,j) = e(i,j) + \min(E_v(i+1, j-1), E_v(i+1, j), E_v(i+1, j+1)), \quad j \in \{0, \ldots, w_o - 1\}$
8: $T_v(i,j) = \arg\min(E_v(i+1, j-1), E_v(i+1, j), E_v(i+1, j+1)), \quad j \in \{0, \ldots, w_o - 1\}$
9: **end for**
10: Determine $j^* = \arg\min_j E_v(0, j)$
11: Trace back the path γ starting at $\gamma_{w_p - 1} = (0, j^*)$ using T_v: for $i = w_p - 2$ to 0, $\gamma_i = T_v(\gamma_{i+1})$
12: **case** horizontal**:**
13: Compute the minimal cumulative horizontal error E_h:
14: $E_h(i, w_p - 1) = e(i, w_p - 1), \quad i \in \{0, \ldots, w_o - 1\}$
15: **for** $j = w_p - 2$ to 0 **do**
16: $E_h(i,j) = e(i,j) + \min(E_h(i-1, j+1), E_h(i, j+1), E_h(i+1, j+1)), \quad i \in \{0, \ldots, w_o - 1\}$
17: $T_h(i,j) = \arg\min(E_h(i-1, j+1), E_h(i, j+1), E_h(i+1, j+1)), \quad i \in \{0, \ldots, w_o - 1\}$
18: **end for**
19: Determine $i^* = \arg\min_i E_h(i, 0)$
20: Trace back the path γ starting at $\gamma_{w_p - 1} = (i^*, 0)$ using T_h: for $j = w_p - 2$ to 0, $\gamma_j = T_h(\gamma_{j+1})$
21: **case** L-shaped**:**
22: Compute the minimal cumulative vertical error E_v as in the vertical case
23: Compute the minimal cumulative horizontal error E_h as in the horizontal case
24: Determine $i^* = \arg\min_i (E_v(i,i) + E_h(i,i) - e(i,i))$ (with $i \in \{0, \ldots, w_o - 1\}$)
25: Trace the vertical part of γ starting at (i^*, i^*) using T_v: for $i = w_p - i^* - 2$ to 0, $\gamma_i = T_v(\gamma_{i+1})$
26: Trace the horizontal part of γ starting at (i^*, i^*) using T_h: for $j = w_p - i^*$ to $2(w_p - i^*) - 1$, $\gamma_j = T_h(\gamma_{j-1})$
27: **end switch**

A.2.4 Blending along the Cut

This is the last step of an iteration. Its goal is to construct the new patch P_{new} by blending P_{old} and P_{in} using the previously computed boundary cut. The boundary cut defines a binary mask M that

equals one on the left and/or top of the cut and zero otherwise. The patch P_{new} can be defined as

$$P_{\text{new}} = MP_{\text{old}} + (1 - M)\, P_{\text{in}}. \tag{A.4}$$

We noticed that the Matlab implementation of the authors proposes an optional smoothing of the binary mask presumably to avoid noticeable transitions along the cut. We implemented this option but we did not observe noticeable improvements. Since it is not discussed in the original paper we do not use it for the experimental results of this paper.

A.3 Implementation

A.3.1 Computing Patch Distances with FFT

In this section we explicit an algorithm to compute the squared distance between a patch P_{old} and all the patches of the input texture I_0 using Fast Fourier Transform (FFT), that is, to compute the squared distance image D of Equation (A.1) involved in the patch search algorithm (see Algorithm 13). To the best of our knowledge, regarding texture synthesis this acceleration was first discussed by Kwatra *et al.* [**?**]. Recall that we consider a patch P_{old} of size $w_p \times w_p$ and that I_0 is of size $M_0 \times N_0$. Hence the naive summation to compute

$$D(m,n) = \sum_{i,j=0}^{w_p-1} Q(i,j)(P_{\text{old}}(i,j) - I_0(m+i, n+j))^2$$

for all $(m,n) \in \{0,\dots,M_0 - w_p\} \times \{0,\dots,N_0 - w_p\}$ requires around $w_p^2 M_0 N_0$ operations with $w_p = 40$ pixels as a typical value. An important asset of the proposed FFT-based implementation is that the computational cost is limited to two FFT calls for images of size $M_0 \times N_0$ and is thus independent of the patch width w_p.

Let us first recall some notation. A patch of size $w_p \times w_p$ within an image is represented by its top-left corner, hence all admissible patches of I_0 have (m,n)-coordinates in the set $\{0,\dots,M_0 - w_p\} \times \{0,\dots,N_0 - w_p\}$.

Discrete Fourier Transform Given any image $V \in \mathbb{R}^{M_0 \times N_0}$ we denote by $\mathcal{F}(V) = \hat{V}$ the discrete Fourier transform of V and $\mathcal{F}^{-1}(V) = \check{V}$ the inverse discrete Fourier transform of V defined by

$$\hat{V}(k,l) = \frac{1}{M_0 N_0} \sum_{m=0}^{M_0-1} \sum_{n=0}^{N_0-1} V(m,n) e^{-2i\pi\left(\frac{km}{M_0} + \frac{nl}{N_0}\right)} \quad \text{and} \quad \check{V}(k,l) = \sum_{m=0}^{M_0-1} \sum_{n=0}^{N_0-1} V(m,n) e^{2i\pi\left(\frac{km}{M_0} + \frac{nl}{N_0}\right)}.$$

Convolution Product Denote by $V * W$ the convolution product between V and W, that is,

$$V * W(m,n) = \sum_{k=0}^{M_0-1} \sum_{l=0}^{N_0-1} V(k,l) W(m-k, n-l),$$

where the indexes are $(m-k, n-l)$ are understood modulo (M_0, N_0). With these conventions for the DFT, one has $\mathcal{F}(V * W) = M_0 N_0 \hat{V}\hat{W}$, where the multiplication between images is the componentwise product. However, recall that the FFTW library computes the Fourier transform

without normalization. Hence the operators that are computed are respectively $M_0N_0\mathcal{F}$ for the forward transform and $\mathcal{F}^{-1}$ for the backward transform. Hence we have

$$V * W = \mathcal{F}^{-1}(\mathcal{F}(V * W)) = \frac{1}{M_0N_0}\mathcal{F}^{-1}(M_0N_0\mathcal{F}(V)M_0N_0\mathcal{F}(W)).$$

This means that after multiplying the two FFTW forward transforms and performing the backward inverse transform, one has to normalize in dividing by the size of the images.

Cross-correlation Let us denote by $\Gamma(V, W)$ the cross-correlation between two images,

$$\Gamma(V, W)(m, n) = \sum_{k=0}^{M_0-1}\sum_{l=0}^{N_0-1} V(k, l)W(m + k, n + l).$$

Note also $\tilde{V}$ the symmetric of V with respect to the origin, that is, $\tilde{V}(m, n) = V(-m, -n)$. Note that one has $\Gamma(W, V)(m, n) = \Gamma(V, W)(-m, -n)$, that is, $\Gamma(W, V) = \widetilde{\Gamma(V, W)}$ and that the cross-correlation is simply a convolution between the symmetric version of the first image and the second image, that is,

$$\Gamma(V, W)(m, n) = \tilde{V} * W(m, n) = V * \tilde{W}(-m, -n).$$

Hence the computational cost for a cross-correlation image is the same as the one for a convolution product, that is three FFT calls.

We need to compute the squared distance between a patch P_{old} and all the patches of I_0 according to some binary weight Q that represents the overlap region (that is a horizontal, vertical, or L-shaped mask). More precisely for all $(m, n) \in \{0, \dots, M_0 - w_p\} \times \{0, \dots, N_0 - w_p\}$, we want to compute

$$D(m, n) = \sum_{i,j=0}^{w_p-1} Q(i, j)(P_{\text{old}}(i, j) - I_0(m + i, n + j))^2.$$

Let us denote by P_{ext} and Q_{ext} the extensions of P_{old} and Q into images of size $M_0 \times N_0$ by filling the domain with 0-valued pixels, that is

$$P_{\text{ext}}(m, n) = \begin{cases} P_{\text{old}}(m, n) & \text{if } (m, n) \in \{0, \dots, w_p - 1\} \times \{0, \dots, w_p - 1\}, \\ 0 & \text{otherwise,} \end{cases}$$

and similarly for Q_{ext}. Then one has

$$\begin{aligned} D(m, n) = & \\ &= \sum_{i,j=0}^{w_p-1} Q(i, j)(P_{\text{old}}(i, j) - I_0(m + i, n + j))^2 \\ &= \sum_{k=0}^{M_0-1}\sum_{l=0}^{N_0-1} Q_{\text{ext}}(k, l)(P_{\text{ext}}(k, l) - I_0(m + k, n + l))^2 \\ &= \sum_{k=0}^{M_0-1}\sum_{l=0}^{N_0-1} \left(Q_{\text{ext}}(k, l)P_{\text{ext}}(k, l)^2 - 2Q_{\text{ext}}(k, l)P_{\text{ext}}(k, l)I_0(m + k, n + l) + Q_{\text{ext}}(k, l)I_0(m + k, n + l)^2\right) \\ &= \sum_{i,j=0}^{w_p-1} Q(i, j)P_{\text{old}}(i, j)^2 - 2\Gamma(Q_{\text{ext}}P_{\text{ext}}, I_0)(m, n) + \Gamma(Q_{\text{ext}}, I_0^2)(m, n), \end{aligned}$$

where the multiplications $Q_{\text{ext}}P_{\text{ext}}$ and I_0^2 are componentwise.

Within our procedure, we will compare a patch P_{old} that is taken from the output image that is only partially defined, and the binary weight Q representing the overlap region. Hence, if all the undefined pixels of the output patch P_{old} are set to 0 (by initializing I_s to 0), one has

$$\forall (i,j) \in \{0, w_p - 1\}^2, \quad Q(i,j)P_{\text{old}}(i,j) = P_{\text{old}}(i,j).$$

Consequently,

$$\sum_{i,j=0}^{w_p-1} Q(i,j)P_{\text{old}}(i,j)^2 = \sum_{i,j=0}^{w_p-1} P_{\text{old}}(i,j)^2 = \|P_{\text{old}}\|_2^2$$

and $Q_{\text{ext}}P_{\text{ext}} = P_{\text{ext}}$, and thus $\Gamma(Q_{\text{ext}}P_{\text{ext}}, I_0) = \Gamma(P_{\text{ext}}, I_0)$. Hence, the binary mask Q is only influent when computing $\Gamma(Q_{\text{ext}}, I_0^2)$, a computation that is done just once at the beginning of the procedure since it does not depend on P_{old}.

In the end, the image D is simply given by

$$D = \|P_{\text{old}}\|_2^2 - 2\Gamma(P_{\text{ext}}, I_0) + \Gamma(Q_{\text{ext}}, I_0^2) \tag{A.5}$$

where the last image $\Gamma(Q_{\text{ext}}, I_0^2)$ is computed once before running the algorithm and stored in memory. Hence the cost for computing D is only three FFT calls for the computation of the cross-correlation $\Gamma(P_{\text{ext}}, I_0)$, and is even reduced to two FFT calls by storing the DFT of I_0.

Note that the image D is defined for all $(m,n) \in \{0, \ldots, M_0 - 1\} \times \{0, \ldots, N_0 - 1\}$, but the squared distances for (m,n)-coordinates outside $\{0, \ldots, M_0 - w_p\} \times \{0, \ldots, N_0 - w_p\}$ correspond to patches of size $w_p \times w_p$ that are defined by periodic boundary conditions and thus those patches must be discarded when searching for the minimal distance.

Minimum distance Theoretically the results of the distance computation using the sum square differences or the FFT are equal. But in practice, the computation using the FFT is subject to rounding errors. Especially if there is a patch such that $D(m,n) = 0$, when adding positive and negative terms in Equation (A.5) the result might be negative due to numerical errors. This rounding error problem leads to a negative minimal squared distance $d_{\min}$, which leads to errors in the patch search algorithm (Algorithm 13) since then none of patches satisfy the condition $D(m,n) < (1+\varepsilon)d_{\min}$. To avoid this, all squared distances $D(m,n)$ smaller than 1 are set to $D(m,n) = 1$ (the value 1 is smallest distance-value greater than 0).

A.3.2 Parallelization

Although the Efros-Freeman algorithm was presented as a purely sequential algorithm, we found out that the procedure can be partially parallelized, and thus the algorithm significantly accelerated when running on multi-core platforms.

In general, non-parametric patch based methods can not be completely parallelized because the sequential assignment of patches is strongly dependent of the previous synthesis steps. Nevertheless, in the case of the quilting method, a new patch assignment is only correlated to some previous synthesis steps and not all of them. To illustrate this let us consider the first few steps of the algorithm. The first step of the algorithm adds the seed patch and the second step adds a second patch on the right. When adding the third patch, the first patch of the second row can be created at the same time since it only depends on the one that is on top of it and on its top-right. More generally the synthesis of a new row of patches can be started as soon as two patches of the previous row have been synthesized. This procedure is illustrated in Figure A.4.

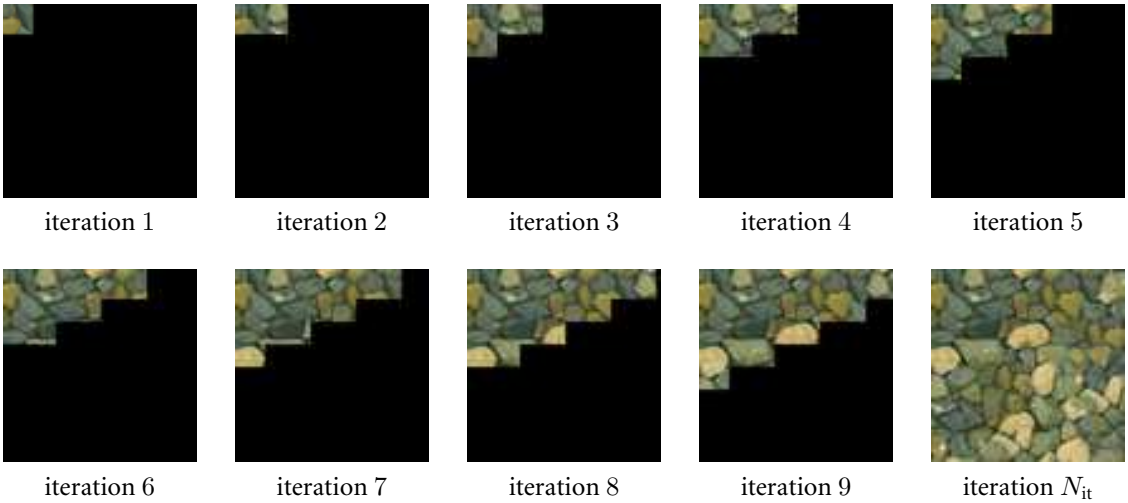


Figure A.4 – Evolution of the synthesized image with the parallelization. New patches from two subsequent iterations are added simultaneously.

Let us discuss the acceleration induced by the parallelization. Suppose we synthesize an image I_s made of $N_r \times N_c$ patches where N_r is the number patch rows and N_c is the number of patch columns. Without the parallelization the number of iterations N_{it} is $N_{\text{it}} = N_r N_c$. When parallelizing the number of iterations N_{it} is reduced to $2(N_r - 1) + N_c$. This parallelization is especially effective if one has as many processors available as patches to synthesize simultaneously in one iteration. The maximal number of patches to synthesize simultaneously is equal to $\min\left(N_r, \left\lfloor \frac{N_c+1}{2} \right\rfloor\right)$, where $\lfloor x \rfloor = \max(n \in \mathbb{N}, n \leq x)$, since within an iteration there is at most one new patch per row and one new patch per pair of column.

The parallel version of the quilting algorithm is summarized in Algorithm 15. Let us notice that for this parallelization to be valid the overlap size must satisfy $w_o \leq w_p/2$ which is always the case in practice.

Algorithm 15: Parallelization

1 **Input:** Sample texture I_0, N_r is the number patch rows, N_c is the number of patch columns
2 **Output:** Synthesized texture I_s

```
N_it ← 2(N_r − 1) + N_c
Initialize I_s
for k = 1 to N_it − 1 do
    for i = max(0, ⌊(k − N_c + 1)/2⌋) to min(N_r − 1, ⌊k/2⌋) do
        I_s ← synthesize a new patch at position (i(w_p − w_o), (k − 2i)(w_p − w_o))
    end for
end for
```

This loop is run in parallel

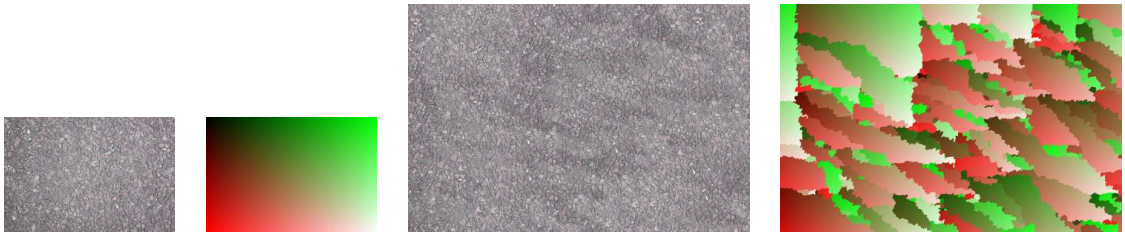


Figure A.5 – *Results representation*. From left to right: texture sample, position map, synthesized image and synthesis map. The synthesis map shows for each synthesized patch its position in the texture sample. It allows then to identify exactly the verbatim copy regions (continuous color areas of the map).

A.4 Experiments

In this section texture synthesis results are shown using the algorithm described previously [15]. Two apects of this method are emphasized. On the one hand the visual quality result and on the other hand the tendency of the method to verbatim copy parts of the input sample. For this we represent each result as shown in Figure A.5: the input sample I_0, the color map, the synthesized texture I_s and the synthesis map. The color map is an image of size equal to the size of I_0 and each of its pixels is assigned a different color. This allows to visualize each position of I_0 with the corresponding color of the color map. The synthesis map allows to identify from where comes every patch copied in I_0 and quilted in I_s. This allows to easily identify verbatim copies. Each pixel $p_i^{'}$ in I_s is assigned a pixel p_j from I_0. The synthesis map at position $p_i^{'}$ is mapped to the value of the color map at position p_j.

Let us discuss the influence of the parameters. There are three of them: the patch size w_p, the overlap size w_o and the tolerance parameter ε. The overlap size is expressed as the proportion respect to the patch size. That is $w_o = 0.25$ implicitly means $w_o = 0.25w_p$.

In Figures A.6 and A.7 successful results are shown for different type of textures. For each example the patch size w_p is adapted taking one of the following values $\{10, 20, 40, 80\}$. The remaining parameters are fixed to $w_o = 0.25$ and $\varepsilon = 0.1$.

In Figure A.8 three failure cases are shown. The first one is due to the patch size. When this one is too small, in particular for macro textures, the algorithm fails to recover the details of the different scales as can be seen in the first row examples' in Figure A.8. The second failure is the verbatim copy effect. For some texture samples the verbatim copy zones are visually noticeable and unnatural. This is illustrated in the second row examples' in Figure A.8. The last failure case is the "growing garbage" drawback. This is when the method is stuck in a region of the input sample repeating it on a large part of the output. This is due to the local aspect of the algorithm and it is more noticeable when the texture examples are not stationary. This is also shown in Figure A.8 in the last row. It is important to notice that the two last failure cases are more noticeable when the output size - input size ratio is higher than two.

Influence of the patch size

First of all the patch size influence is analyzed and these results are shown in Figure A.9. For this $w_o = 0.25$ and $\varepsilon = 0.1$ are fixed and the synthesis results for $w_p \in \{10, 20, 40, 80\}$ are compared. This parameter clearly depends on the input sample. In general macro textures have details at different scales. If the patch size chosen is not able to capture them all then the synthesis fails. A second observation is that the larger the patch size is the larger the verbatim copy zones are. For

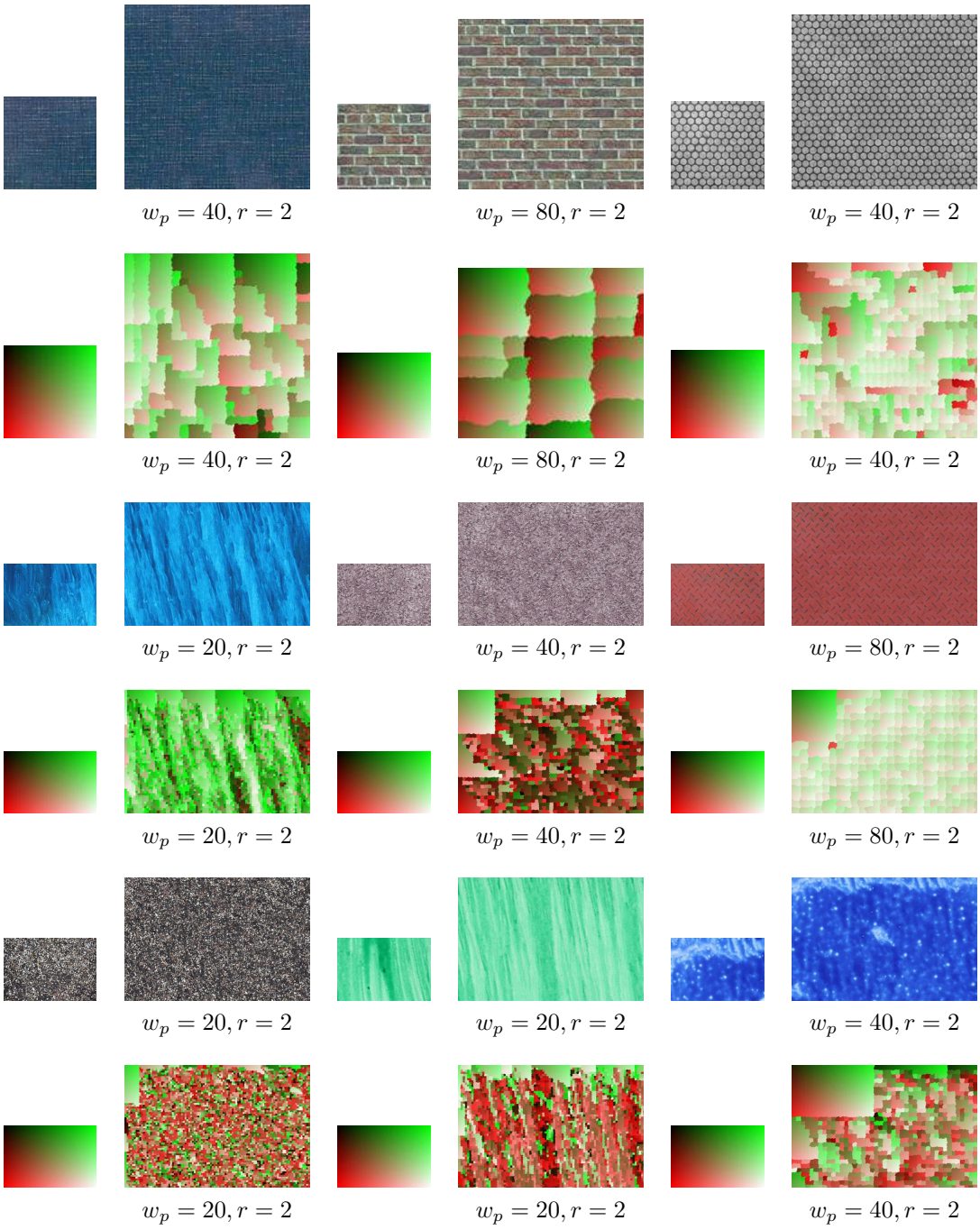


Figure A.6 – *Successful results of Efros and Freeman image quilting algorithms*. The small images represent the example texture and the big ones the corresponding synthesis result. Each row of examples is followed by a row containing the corresponding color and synthesis maps. For all examples the patch size w_p and the ratio r used is indicated. The overlap size is fixed to $w_o = 0.25$ and the tolerance error to $\varepsilon = 0.1$.

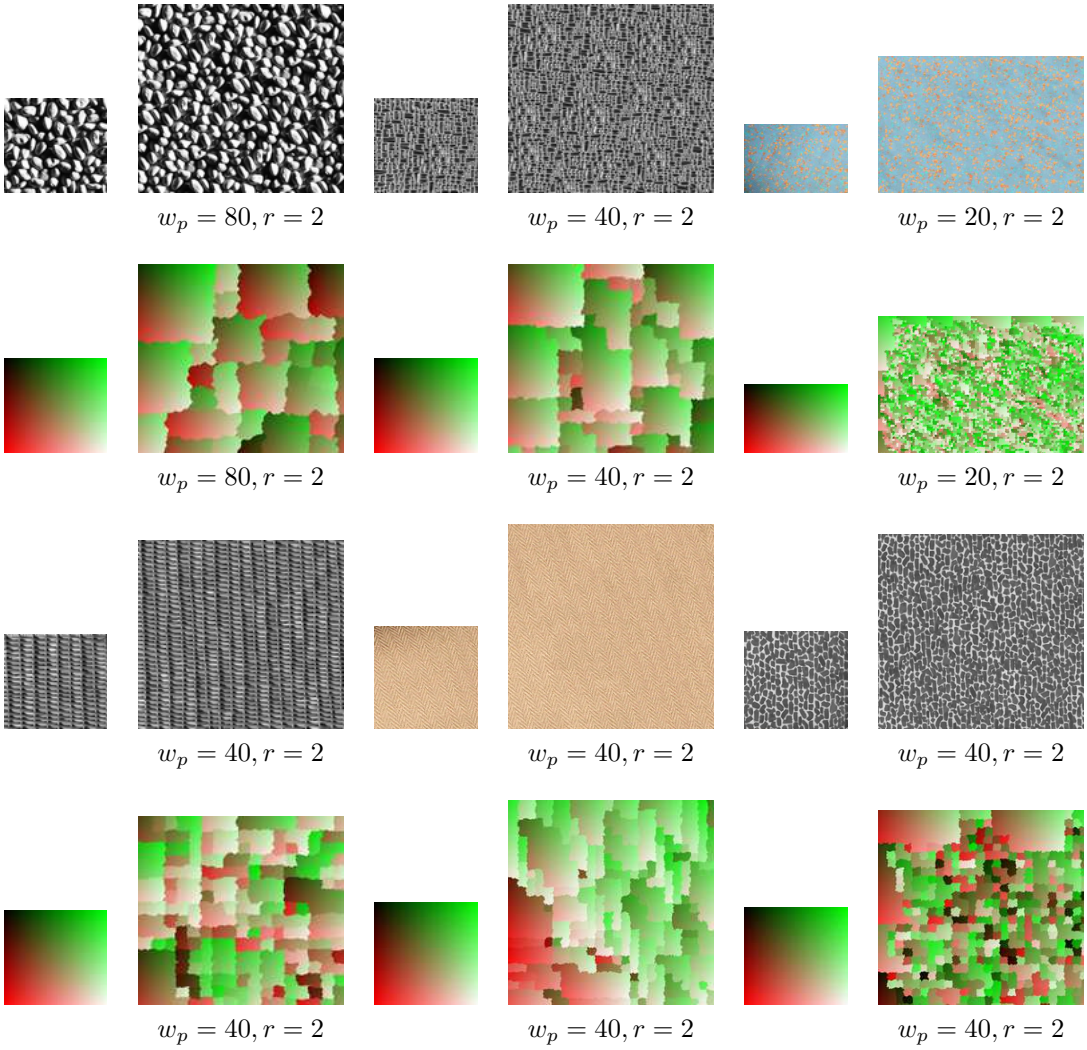


Figure A.7 – *Successful results of Efros and Freeman image quilting algorithms*. The small images represent the example texture and the big ones the corresponding synthesis result. Each row of examples is followed by a row containing the corresponding color and synthesis maps. For all examples the patch size w_p and the ratio r used is indicated. The overlap size is fixed to $w_o = 0.25$ and the tolerance error to $\varepsilon = 0.1$.

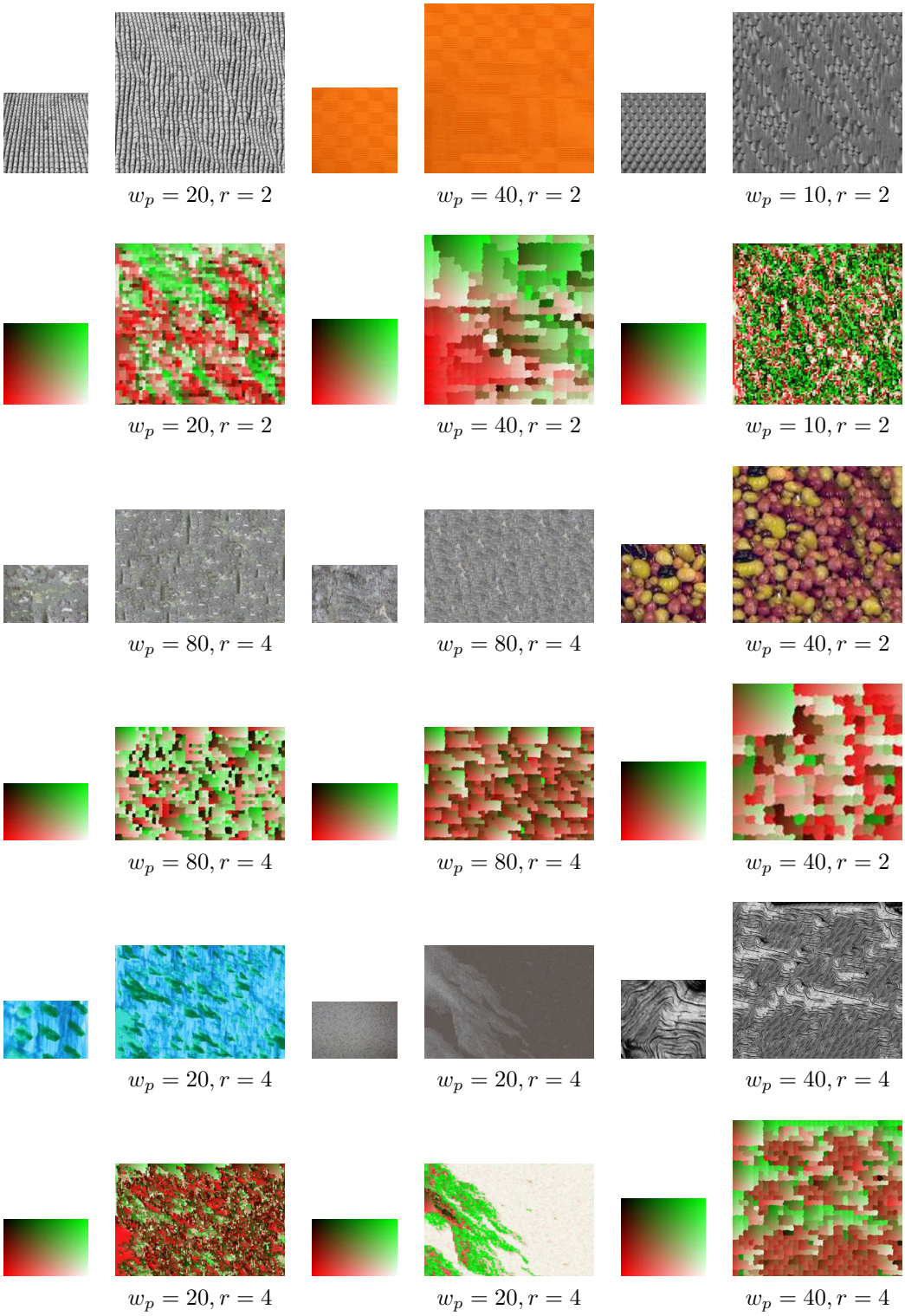


Figure A.8 – *Failures of Efros and Freeman image quilting algorithms*. The small images represent the example texture and the big ones the corresponding synthesis result. First row examples' show failures related to an incorrect patch size. Second row examples' show the case of verbatim copy failures. Third row examples' show the case of growing garbage. For all examples the patch size w_p and the ratio r used is indicated.

smaller patches this effect is reduced, since more "similar" patches are available in patch search step and then the set of candidate patches is larger allowing more variation.

Influence of the overlap size

The overlap size is an important parameter in this method. To analyze its influence $w_p = 40$ and $\varepsilon = 0.1$ are fixed and the results for $w_o \in \{0.10, 0.25, 0.5\}$ are compared. The general conclusion is that for larger overlap sizes the transition from one patch to the other is satisfying on the cost of growing garbage for some texture sample as can be seen in the two first examples in Figure A.10 for $w_o = 0.50$. On the other hand if w_o is low the set of candidate patches is bigger and thus reduces the verbatim copy effect on the cost of decreasing the visual quality of the results as can be seen in the third example in Figure A.10.

Influence of the tolerance parameter

This last parameter is directly related to the verbatim copy effect. For this analysis $w_p = 40$ and $w_o = 0.25$ are fixed and the results for $\varepsilon \in \{0.05, 0.1, 0.3, 0.5, 0.7\}$ are compared. Increasing ε implies having more candidate patches for the synthesis. This allows more variation when choosing a patch in the image and this is directly seen in the synthesis maps of the examples in Figure A.11 where for $\varepsilon \in \{0.5, 0.7\}$ the patches are taken more "randomly" from the input sample thus reducing the size of the verbatim copy regions. For some texture examples the visual quality of the result can decrease. On the other hand, as expected, low values of ε leads to very large verbatim copy areas.

A.5 Conclusion

In this paper we analyzed in detail Efros and Freeman's texture synthesis algorithm [15]. Extensive numerical experiments have been proposed to illustrate the performance of the method as well as the influence of its parameters. We conclude from these experiments that in general, for the correct set of parameters, the visual results are satisfying to the cost of verbatim copying large parts of the input textures that might be visually disturbing. This is a common drawback of Efros and Leung's method [14]. Another issue that arises in [15] is the garbage growing effect. This is especially apparent when the input texture is not stationary. We also noticed that it is related to the raster scan order used to synthesize the image, which propagates the errors. All these drawbacks are in general more apparent when synthesizing an image significantly larger than the input. Hence, due to the raster scan order, the quilting algorithm is not suitable to generate very large texture images since the quality tends to decrease with the distance to the top left corner of the image.

We provide with this analysis a strategy to partially parallelize the method who was initially introduced as essentially sequential. This allows a significant speed up when running with multi-core processors.

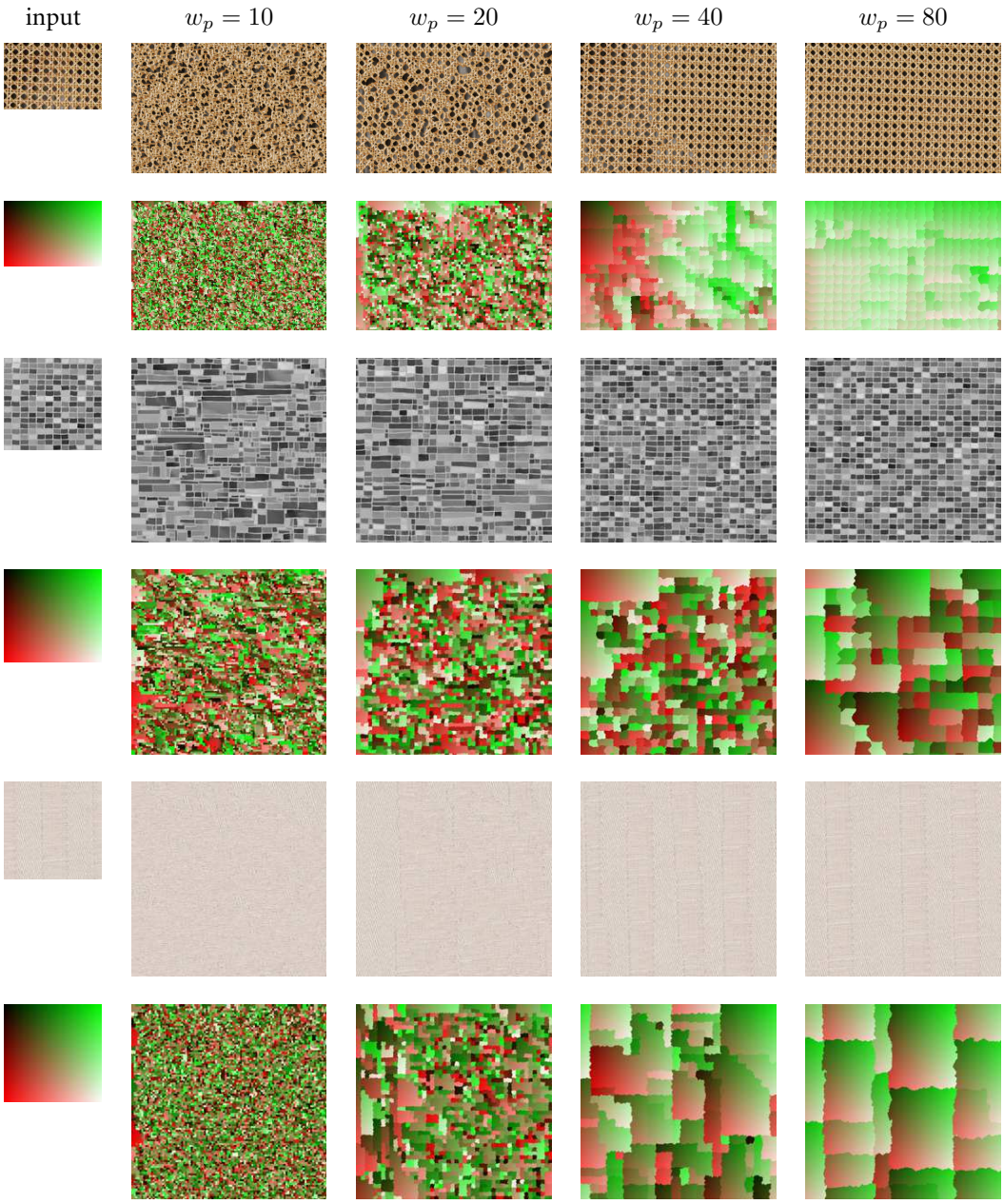


Figure A.9 – *Patch size influence*. Each couple of rows shows from left to right the input, the synthesis results for $w_p = 10, 20, 40, 80$ and the corresponding color and synthesis maps. For all examples $w_o = 0.25$ and $\varepsilon = 0.1$.

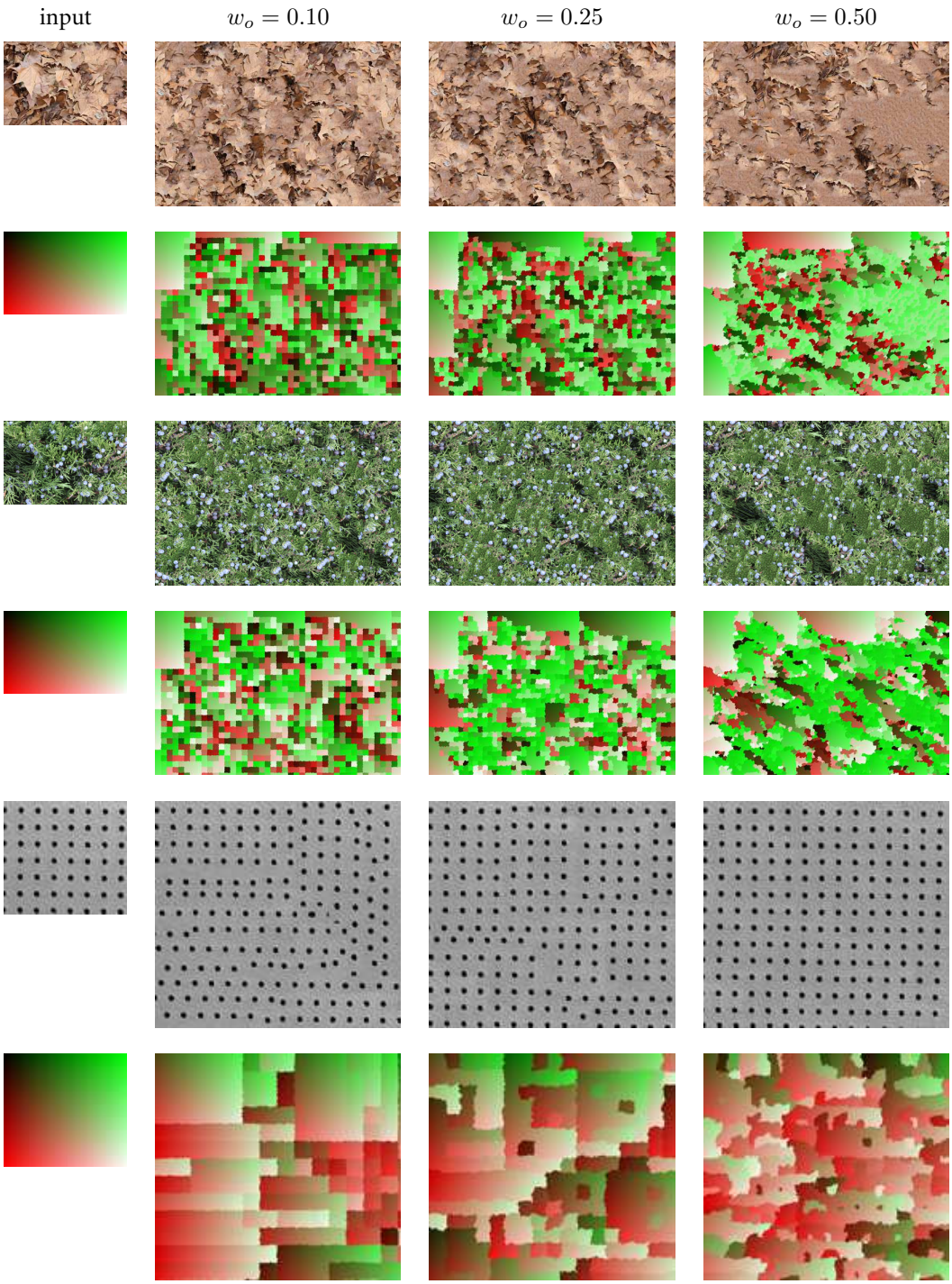


Figure A.10 – *Overlap size influence*. Each couple of rows shows from left to right the input, the synthesis results for $w_o = 0.1, 0.25, 0.5$ and the corresponding color and synthesis maps. For all examples $w_p = 40$ and $\varepsilon = 0.1$.

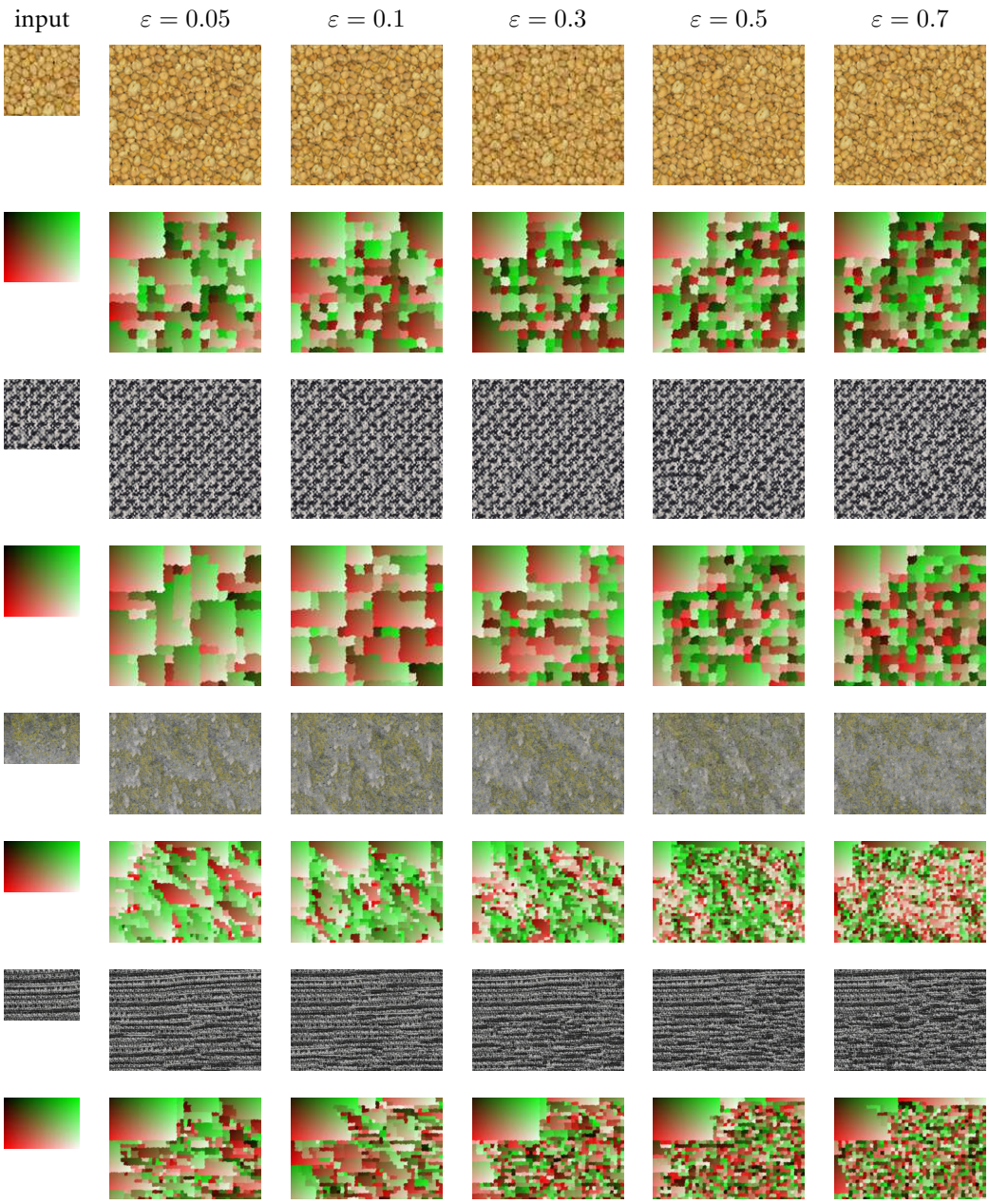


Figure A.11 – *Tolerance error influence*. Each couple of rows shows from left to right the input, the synthesis results for $\varepsilon = 0.05, 0.1, 0.3, 0.5, 0.7$ and the corresponding color and synthesis maps. For all examples $w_p = 40$ and $w_o = 0.25$.

[]

Bibliography

[1] Cecilia Aguerrebere, Yann Gousseau, and Guillaume Tartavel. Exemplar-based texture synthesis: the efros-leung algorithm. *Image Processing On Line*, 2013:213–231, 2013.

[2] M. Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226. ACM, 2001.

[3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[4] Charles Bordenave, Yann Gousseau, and François Roueff. The dead leaves model: a general tessellation modeling occlusion. *Advances in Applied Probability*, pages 31–46, 2006.

[5] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[6] Thibaud Briand, Jonathan Vacher, Bruno Galerne, and Julien Rabin. The heeger & bergen pyramid based texture synthesis algorithm. *Image Processing On Line*, 4:276–299, 2014.

[7] A. Buades, M. Lebrun, and J.-M. Morel. Implementation of the "non-local bayes" image denoising algorithm,". *Image Processing On Line*, 2012.

[8] TM Caelli and B Julesz. Experiments in the visual perception of texture. *Biol. Cybernetics*, 28:167–175, 1978.

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

[10] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image melding: combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.*, 31(4):82–1, 2012.

[11] Jeremy S De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368. ACM Press/Addison-Wesley Publishing Co., 1997.

[12] Julie Delon. Midway image equalization. *Journal of Mathematical Imaging and Vision*, 21(2):119–134, 2004.

[13] Julie Delon. Movie and video scale-time equalization application to flicker reduction. *IEEE Transactions on Image Processing*, 15(1):241–248, 2006.

[14] Alexei Efros, Thomas K Leung, et al. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[15] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[16] Vadim Fedorov, Gabriele Facciolo, and Pablo Arias. Variational Framework for Non-Local Inpainting. *Image Processing On Line*, 5:362–386, 2015.

[17] Sira Ferradans, Nicolas Papadakis, Gabriel Peyré, and Jean-François Aujol. Regularized discrete optimal transport. *SIAM Journal on Imaging Sciences*, 7(3):1853–1882, 2014.

[18] Sira Ferradans, Gui-Song Xia, Gabriel Peyré, and Jean-François Aujol. Static and dynamic texture mixing using optimal transport. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 137–148. Springer, 2013.

[19] Lester R Ford and DR Fulkerson. *Flow in networks*. Princeton University Press, 1962.

[20] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. Micro-texture synthesis by phase randomization. *Image Processing On Line*, 2011, 2011.

[21] Bruno Galerne, Yann Gousseau, and Jean-Michel Morel. Random phase textures: Theory and synthesis. *Image Processing, IEEE Transactions on*, 20(1):257–267, 2011.

[22] Bruno Galerne, Arthur Leclaire, and Lionel Moisan. Microtexture inpainting through gaussian conditional simulation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1204–1208. IEEE, 2016.

[23] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. *arXiv preprint arXiv:1505.07376*, 2015.

[24] Thierry Guillemot and Julie Delon. Implementation of the midway image equalization. *Image Processing On Line*, 6:114–129, 2016.

[25] Baining Guo, Harry Shum, and Ying-Qing Xu. Chaos mosaic: Fast and memory efficient texture synthesis. *Microsoft research paper MSR-TR-2000-32*, 2000.

[26] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.

[27] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[28] Bela Julesz. Visual pattern discrimination. *Information Theory, IRE Transactions on*, 8(2):84–92, 1962.

[29] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[30] Bela Julesz. A theory of preattentive texture discrimination based on first-order statistics of textons. *Biological cybernetics*, 41(2):131–138, 1981.

[31] Bela Julész, EN Gilbert, LA Shepp, and HL Frisch. Inability of humans to discriminate between visual textures that agree in second-order statistics—revisited. *Perception*, 2(4):391–405, 1973.

[32] Bela Julesz, EN Gilbert, and Jonathan D Victor. Visual discrimination of textures with identical third-order statistics. *Biological Cybernetics*, 31(3):137–140, 1978.

[33] Rabin Julien and Gabriel Peyré. Régularisation de wasserstein. application au transfert de couleur. In *GRETSI'11*, 2011.

[34] Rabin Julien and Gabriel Peyré. Synthese de textures par transport optimal. In *GRETSI'11*, 2011.

[35] Alexandre Kaspar, Boris Neubert, Dani Lischinski, Mark Pauly, and Johannes Kopf. Self tuning texture optimization. In *Computer Graphics Forum*, volume 34, pages 349–359. Wiley Online Library, 2015.

[36] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 795–802. ACM, 2005.

[37] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (ToG)*, volume 22, pages 277–286. ACM, 2003.

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

[39] M. Lebrun, A. Buades, and J.-M. Morel. A nonlocal bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013.

[40] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Transactions on Graphics (TOG)*, 24(3):777–786, 2005.

[41] Elizaveta Levina and Peter J Bickel. Texture synthesis and nonparametric resampling of random fields. *The Annals of Statistics*, pages 1751–1773, 2006.

[42] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001.

[43] G Matheron. Schéma booléen séquentiel de partition aléatoire. *Rapport technique N-83, Centre de Morphologie Mathématique, École des Mines de Paris*, 214, 1968.

[44] Enric Meinhardt-Llopis. Turbulence simulation: geometric deformations. `http://dev.ipol.im/~coco/ipol_demo/workshop_simu/`, January 2015.

[45] Jean-Michel Morel and Guoshen Yu. Is sift scale invariant? *Inverse Problems and Imaging*, 5(1):115–136, 2011.

[46] Donald F Morrison. Multivariate statistical methods. 3. *New York, NY. Mc*, 1990.

[47] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318, 2003.

[48] G. Peyré. Texture synthesis with grouplets. *IEEE Trans. Pattern. Anal. Mach. Intell.*, 4(32):733–746, 2010.

[49] Gabriel Peyré. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34(1):17–31, 2009.

[50] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

[51] Julien Rabin, Julie Delon, and Yann Gousseau. Regularization of transportation maps for color and contrast transfer. In *2010 IEEE International Conference on Image Processing*, pages 1933–1936. IEEE, 2010.

[52] Julien Rabin, Julie Delon, and Yann Gousseau. Transportation distances on the circle. *Journal of Mathematical Imaging and Vision*, 41(1-2):147–167, 2011.

[53] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*, pages 435–446. Springer Berlin / Heidelberg, 2012.

[54] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[55] Jean Serra. *Image analysis and mathematical morphology, v. 1.* Academic press, 1982.

[56] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[57] E. P. Simoncelli and B. A. Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001.

[58] Eero P Simoncelli and William T Freeman. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *ICIP (3)*, pages 444–447, 1995.

[59] Guillaume Tartavel, Yann Gousseau, and Gabriel Peyré. Variational texture synthesis with sparsity and spectrum constraints. *Journal of Mathematical Imaging and Vision*, 52(1):124–144, 2014.

[60] J. J. van Wijk. Spot noise texture synthesis for data visualization. In *SIGGRAPH '91*, pages 309–318, New York, NY, USA, 1991. ACM.

[61] Cédric Villani. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.

[62] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117. Eurographics Association, 2009.

[63] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[64] Gui-Song Xia, Sira Ferradans, Gabriel Peyré, and Jean-François Aujol. Synthesizing and mixing stationary gaussian texture models. *SIAM Journal on Imaging Sciences*, 7(1):476–508, 2014.

[65] Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural computation*, 9(8):1627–1660, 1997.

université
PARIS-SACLAY

Titre : Synthèse de texture à partir d'exemples: modèles et applications

Mots clés : synthèse de texture à partir d'exemples, modèle Gaussien, patch, multi échelle, transport optimal

Résumé : Cette thèse s'attaque au problème de la synthèse de texture par l'exemple en utilisant des modèles stochastiques locaux de patchs pour générer de nouvelles images.

La synthèse de texture par l'exemple a pour but de générer à partir d'un échantillon de texture de nouvelles images qui sont perceptuellement équivalentes à celle de départ. Les méthodes peuvent se regrouper en deux catégories: les méthodes paramétriques et les non paramétriques à base de patchs. Le premier groupe a pour but de caractériser une image de texture à partir d'un ensemble de statistiques qui définissent un processus stochastique sous-jacent. Les résultats visuels de ces méthodes sont satisfaisants, mais seulement pour un groupe réduit de types de texture. La synthèse pour des images de textures ayant des structures très contrastées peut échouer. La deuxième catégorie d'algorithme découpe, puis recolle de manière consistente des voisinages locaux de l'image de départ pour générer de nouvelles configurations plausibles de ces voisinages (ou patchs). Les résultats visuels de ces méthodes sont impressionnants. Néanmoins, on observe souvent des répétitions verbatim de grandes parties de l'image d'entrée qui du coup peuvent être reproduites plusieurs fois. De plus, ces algorithmes peuvent diverger, reproduisant de façon itérative une partie de l'image de l'entrée en négligeant le reste.

La première partie de cette thèse présente une approche combinant des idées des deux catégories de méthodes, sous le nom de *synthèse localement Gaussienne*. On préserve dans cette nouvelle méthode les aspects positifs de chaque approche: la capacité d'innover des méthodes paramétriques, et la capacité de générer des textures fortement structurées des méthodes non paramétriques à base de patchs. Pour ce faire, on construit un modèle Gaussien multidimensionnel des auto-similarités d'une image de texture. Ainsi, on obtient des résultats qui sont visuellement supérieurs à ceux obtenus avec les méthodes paramétriques et qui sont comparables à ceux obtenus avec les méthodes non-paramétriques à base de patchs tout en utilisant une paramétrization locale de l'image. La thèse s'attache aussi à résoudre une autre difficulté des méthodes à base de patchs: le choix de la taille du patch. Afin de réduire significativement cette dépendance, on propose une extension multi échelle de la méthode.

Les méthodes à bases de patchs supposent une étape de recollement. En effet, les patchs de l'image synthétisée se superposent entre eux, il faut donc gérer le recollement dans ces zones. La première approche qu'on a considérée consiste à prendre en compte cette contrainte de superposition dans la modélisation des patchs. Les expériences montrent que cela est satisfaisant pour des images de textures périodiques ou pseudo-périodiques et qu'en conséquence l'étape de recollement peut être supprimée pour ces textures. Cependant, pour des images de textures plus complexes ce n'est pas le cas, ce qui nous a menée à suggérer une nouvelle méthode de recollement inspirée du transport optimal.

Cette thèse conclut avec une étude complète de l'état de l'art en génération d'images de textures naturelles. L'étude que nous présentons montre que, malgré les progrès considérables des méthodes de synthèse à base d'exemples proposées dans la vaste littérature, et même en les combinant astucieusement, celles-ci sont encore incapables d'émuler des textures complexes et non stationnaires.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université
PARIS-SACLAY

Title : Exemplar based texture synthesis: models and applications

Keywords : exemplar based texture synthesis, Gaussian model, patch, multi scale, optimal transport

Abstract : This dissertation contributes to the problem of exemplar based texture synthesis by introducing the use of local Gaussian patch models to generate new texture images.

Exemplar based texture synthesis is the process of generating, from an input texture sample, new texture images that are perceptually equivalent to the input. There are roughly two main categories of algorithms: the statistics-based methods and the non-parametric patch-based methods. The statistics-based methods aim at characterizing a given texture sample by estimating a set of statistics which will define an underlying stochastic process. The new images will then be samples of this stochastic process, i.e. they will have the same statistics as the input sample. The question here is what would be the appropriate set of statistics to yield a correct synthesis for the wide variety of texture images? The results of statistical methods are satisfying but only on a small group of textures, and often fail when important structures are visible in the input. The non-parametric patch-based methods reorganize local neighborhoods from the input sample in a consistent way to create new texture images. These methods return impressive visual results. Nevertheless, they often yield verbatim copies of large parts of the input sample. Furthermore, they can diverge, starting to reproduce iteratively one part of the input sample and neglecting the rest of it, thus growing what experts call "garbage".

In this thesis we propose a technique combining ideas from the statistical based methods and from the non-parametric patch-based methods. We call it the *locally Gaussian* method. The method keeps the positive aspects of both categories: the innovation capacity of the parametric methods and the ability to synthesize highly structured textures of the non-parametric methods. To this aim, the self-similarities of a given input texture are modeled with conditional multivariate Gaussian distributions in the patch space. A new image is generated patch-wise, where for each given patch a multivariate Gaussian model is inferred from its nearest neighbors in the patch space of the input sample, and hereafter sampled from this model. The synthesized textures are therefore everywhere different from the original. In general, the results obtained are visually superior to those obtained with statistical based methods, which is explainable as we use a local parametric model instead of a global one. On the other hand, our results are comparable to the visual results obtained with the non-parametric patch-based methods. This dissertation addresses another weakness of patch-based methods. They are strongly dependent on the patch size, which has to be decided manually. It is therefore crucial to fix a correct patch size for each synthesis. Since texture images have, in general, details at different scales, we extend the method to a multiscale approach which reduces the strong dependency of the method on the patch size.

Patch based methods involve a stitching step. Indeed, the patches used for the synthesis process overlap each other. This overlap must be taken into account to avoid any transition artifact from patch to patch. Our first attempt to deal with it was to consider directly the overlap constraints in the local parametric model. The experiments show that for periodic and pseudo-periodic textures, considering these constraints in the parametrization is enough to avoid the stitching step. Nevertheless, for more complex textures it is not sufficient. This leads us to suggest a new stitching technique inspired by optimal transport and midway histogram equalization.

This thesis ends with an extensive analysis of the generation of several natural textures. This study shows that, in spite of remarkable progress for local textures, the methods proposed in the extensive literature of exemplar based texture synthesis are still limited when dealing with complex and non-stationary textures.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France