Non-parametric synthesis of volumetric textures from a 2D sample

Radu Dragos Urs

N° d'ordre : 4773

THÈSE

PRÉSENTÉE A

L'UNIVERSITÉ BORDEAUX 1

ÉCOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

Par **Radu Dragoş URS**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : AUTOMATIQUE ET PRODUCTIQUE, SIGNAL ET IMAGE, INGÉNIERIE COGNITIQUE

NON-PARAMETRIC SYNTHESIS OF VOLUMETRIC TEXTURES FROM A 2D SAMPLE

Directeur de recherche : M. Christian GERMAIN (Professeur des Universités)
Co-directeur de recherche : M. Jean-Pierre DA COSTA (Maître de conférences)

Soutenue le : 29 Mars 2013

Devant la commission d'examen formée de :

M. ALATA Olivier	Professeur, Université Jean Monnet de Saint-Etienne	Rapporteur
M. CARRÉ Philippe	Professeur, Université de Poitiers	Rapporteur
M. VIGNOLES Gérard	Professeur, Université de Bordeaux I	Président
M. GERMAIN Christian	Professeur, Université Bordeaux Sciences Agro	
M. DA COSTA Jean-Pierre	Maître de conférences, Université Bordeaux Sciences Agro	
Mme. REVOL-MULLER Chantal	Maître de conférences, Université de Lyon I	

Acknowledgments

It is with high pleasure and consideration that I thank the people who made this thesis possible, turning this PhD into an unforgettable and charming experience.

I express my most sincere thanks and gratitude to my two thesis advisers, Prof. Christian Germain and Assoc. Prof. Jean-Pierre Da Costa for the invaluable guidance, comments and feedback they provided during the preparation of this thesis. I thank them for their availability, appreciating equally their human virtues as well their tremendous scientific qualities. I could not have imagined having better advisors and mentors for my PhD study, mentioning the friendly environment they have created, conducive to the research activity. The work inside the Signal and Image research group within the IMS laboratory would not have been possible without this high-quality supervision.

My appreciation and thanks also go to the rest of my thesis committee, being honoured by the presence of Prof. Olivier Alata, Prof. Philippe Carré, Prof. Gérard Vignoles and the Assoc. Prof. Chantal Revol-Muller for their encouragement, insightful comments and fruitful questions, expressing my gratitude to the thesis referees for their positive feedback. I extend my most sincere thanks to the Thermostructural Composites Laboratory (LCTS) for the rich exchanges that we have undertaken. In particular, for his composite materials expertise, I would like to thank Mr. Vignoles, who also gave me the privilege to preside my thesis jury.

A warm thought goes to my colleagues and professors with the research group that I have been part of during my thesis, to all those that I have not quoted but which ultimately contributed to the pleasant ambience in the lab during these last three years. I show a special appreciation to my professors from the Technical University of Cluj-Napoca in Romania, namely Prof. Monica Borda and Assoc. Prof. Romulus Terebeş, who opened me the doors to the image processing research activity and paved the way toward this thesis.

Finally, I would like to conclude these acknowledgments by expressing the profound gratitude from my deep heart to my beloved family – my wife, my parents, my brother and my family-in-law, for their love, their continuous support and for believing in me. Above all, I would like to take the opportunity to thank my wife Alexandra for her love, her personal support and for always being by my side; this thesis triumph is not just mine, but a result of our accomplishments. My mere expression of thanks does not suffice, that’s why I dedicate the thesis to my beloved family.

MÉTHODES NON-PARAMÉTRIQUES POUR LA SYNTHÈSE DE TEXTURES VOLUMIQUES A PARTIR D'UN EXEMPLE 2D

- résumé étendu -

1. Contexte

L'efficacité des techniques de synthèse d'images pour modéliser et reproduire des textures naturelles (bois, roche, grès, marbre, tissu etc.) n'est plus à démontrer. Le domaine de la synthèse de texture est particulièrement dynamique avec des applications importantes dans l'extrapolation, la compression d'images, l'inpaiting ou le mapping mais aussi dans d'autres domaines comme la fusion et le montage vidéo ou la description de la géométrie d'une surface.

De nombreuses techniques de synthèse de textures ont été proposées. Ces techniques, souvent 2D, demandent à être adaptées en vue de la modelisation de structures volumiques telles que celles obtenues par des techniques d'imagerie 3D, comme l'imagerie médicale, l'imagerie sismique pour l'exploration du sous-sol ou la tomographie en sciences des matériaux. Dans ce dernier domaine, les textures 3D présentent un intérêt tout particulier pour l'étude de matériaux à structure interne tridimensionnelle.

Dans certains cas toutefois, pour des raisons de coût, de praticité, ou simplement de résolution, l'utilisation de techniques d'imagerie 3D n'est pas envisageable. Les structures tridimensionnelles constitutives des matériaux sont alors imagées en 2D. Dans ces conditions, toute méthode stéréologique permettant l'extrapolation en 3D d'une information 2D, est susceptible d'améliorer la compréhension de la réalité physique du matériau. C'est à ce titre que nous nous intéressons dans ce mémoire aux méthodes de synthèse de textures volumiques à partir d'images 2D.

2. Objectif

L'objectif de cette thèse est de développer des algorithmes dédiés à la synthèse de textures volumiques anisotropes à partir d'un échantillon 2D. Outre les difficultés liées a leur complexité calculatoire, de telles approches posent des problèmes stéréologiques d'inférence 2D/3D et ne sont envisageables que pour des textures isotropes ou, dans le cas de textures anisotropes – lamellaires ou filaires – sous certaines hypothèses bien définies.

Les approches dites *par patch* relatives à la synthèse de texture à partir d'un exemple, parmi les plus populaires et déjà très étudiées, sont privilégiées. Il s'agit de les étendre au cas de textures volumiques, en particulier de textures structurées et anisotropes.

Les algorithmes développés sont appliqués à la modélisation de la structure nanométrique de matériaux carbonés. La synthèse se fonde sur des images de microscopie électronique. Les données 3D produites ont vocation à être utilisées pour la simulation réaliste de la structure moléculaire du matériau.

Au-delà de l'évaluation visuelle des textures de synthèse, une analyse quantitative est proposée, qui consiste à comparer les caractéristiques de l'image d'entrée avec celles du bloc de sortie. Cette étude permet d'une part d'identifier les stratégies les plus pertinentes pour la

Thèse préparée à CNRS UMR 5218 – Laboratoire IMS, Groupe Signal et Image
351 Cours de la Libération, 33405 Talence cedex, FRANCE

synthèse et d'autre part de les comparer de façon objective à certains algorithmes de la littérature, qu'ils reposent sur des approches paramétriques ou non paramétriques.

3. Les algorithmes mis en œuvre

Différents algorithmes sont capables de synthétiser des textures parmi lesquels certains sont connus pour leur efficacité et leur facilité d'utilisation. C'est le cas des approches non paramétriques basées sur la recherche de voisinages fixes [*Wei03*] [*Kop07*] [*Che10*] ou sur la vraisemblance de patches [*Pag98*]. Dans les approches 3D que nous explorons, la synthèse est effectuée voxel par voxel, le processus s'appuyant sur l'échantillonnage d'un seul modèle 2D d'entrée en garantissant la cohérence selon plusieurs vues de la texture 3D.

3.1 Extensions 2D/3D basées sur la recherche de voisinage fixe

Nous présentons ici trois variantes algorithmiques, inspirées d'approches existantes, que nous avons adaptées, formalisées et implémentées selon un schéma algorithmique commun.

Le premier algorithme est une extension de l'algorithme introduit par Wei et Levoy [*Wei00*]. Cet algorithme, dont nous reprendrons par la suite la structure, servira de *squelette* aux algorithmes suivants. Il s'agit en réalité d'une adaptation multi-2D de l'algorithme de synthèse à partir de sources multiples [*Wei03*] mais en utilisant strictement une seule texture 2D en tant que source de synthèse.

Basée sur l'hypothèse d'un champ de Markov, la méthode repose sur les propriétés de localité et de stationnarité de la texture. Cet algorithme requiert une texture échantillon en entrée et un bloc 3D en sortie initialisé par un bruit aléatoire. La synthèse est réalisée automatiquement sans aucune supervision. Le bloc est peu à peu modifié pour se rapprocher statistiquement de la texture d'entrée.

La procédure de synthèse traite le bloc de sortie voxel par voxel, en examinant les voisinages 2D du voxel courant sur plusieurs vues orthogonales du bloc 3D (*Figure 1*). Cette phase implique la recherche de la meilleure correspondance pour chacun de ces voisinages dans la même image d'entrée. La similitude entre les voisinages est mesurée en utilisant la distance euclidienne. Ainsi, pour chaque voxel de sortie, plusieurs solutions sont possibles, une pour chaque vue orthogonale.

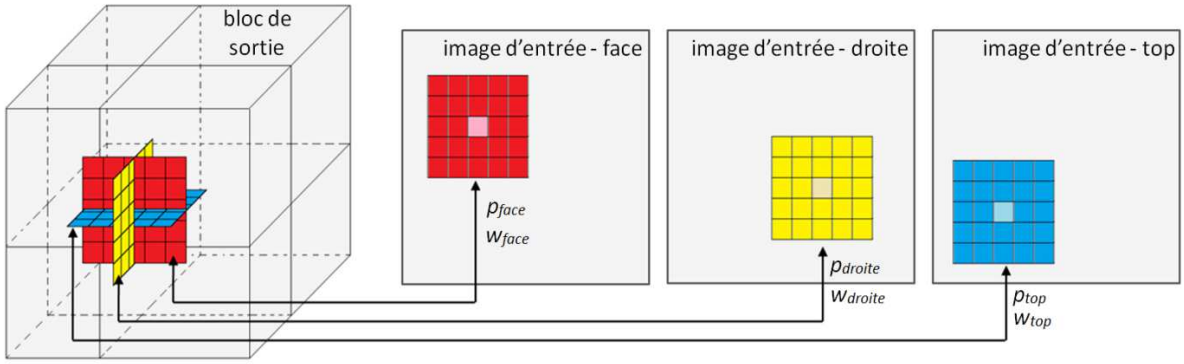


Figure 1 – *Le principe de synthèse non-paramétrique: extraire un voisinage du voxel sur chaque vue orthogonale (face, droite, top), rechercher dans la même image d'entrée les voisinages les plus ressemblants, prendre les valeurs de leurs pixels centraux (p_{face}, p_{droite}, p_{top}), et les utiliser pour modifier le voxel v de sortie: $v=w_{face}\times p_{face}+w_{droite}\times p_{droite}+w_{top}\times p_{top}$. En fonction de la méthode utilisée, le poids associé à chaque pixel peut changer.*

La première solution proposée est d'utiliser comme valeur de mise-à-jour pour le voxel de sortie la moyenne des pixels trouvés pour chaque vue, et de réitérer jusqu'à atteindre les mêmes voisinages après deux itérations consécutives [*Wei03*]. Mais le fait de moyenner conduit à une différence de dynamique entre l'original et la texture synthétisée qui peut compromettre l'ensemble du processus de synthèse.

Une stratégie alternative consiste à optimiser la fonction d'énergie qui mesure la similitude entre la texture volumique et la texture d'entrée. Cette optimisation repose sur une pondération différenciée des solutions trouvées pour chaque vue [*Kwa05*]. La valeur du pixel de sortie se voit ainsi affecter une moyenne pondérée de ces solutions. La procédure d'optimisation est complétée par un schéma de repondération, dans le but de préserver les statistiques globales de la texture d'entrée et de façon à ce que la synthèse ne porte pas seulement sur des décisions locales. Ceci est réalisé en intégrant dans le procédé de synthèse un mécanisme d'ajustement d'histogramme de niveaux de gris [*Kop07*]. Toutefois, les résultats obtenus sont toujours plus ou moins affectés par le flou. Par ailleurs, comme dans l'approche précédente, ce procédé tend à répéter certaines configurations texturales et ne parvient pas à reproduire la diversité observée sur la texture échantillon [*Urs11*].

Afin de rendre la texture synthétisée la plus fidèle possible à la texture d'entrée, une variante des algorithmes précédents intègre deux nouveaux mécanismes d'ajustement basés respectivement sur l'histogramme d'indices et l'histogramme des positions [*Che10*]. Le premier a pour objectif d'augmenter le nombre de configurations parmi lesquelles sera choisie la solution finale; le second est introduit pour faire en sorte que les configurations reproduites dans l'image de sortie soient réparties équitablement sur toute la surface de la texture d'entrée. Cet algorithme nécessite une phase d'apprentissage préalable dont le principe est d'identifier, pour chaque pixel de l'image d'entrée, un ensemble de pixels de configurations locales similaires. Un des intérêts de ce schéma est qu'il réalise indirectement l'ajustement d'histogramme de niveaux de gris en distribuant équitablement les pixels d'entrée dans le bloc de sortie.

La complexité calculatoire des différentes variantes algorithmiques présentées ci-dessus est liée linéairement à la taille de l'image d'apprentissage. Afin d'accélérer la phase de recherche du meilleur voisinage dans l'image d'entrée, un arbre binaire de recherche est utilisé comme structure de données pour des requêtes de type 'point le plus proche' efficaces. Par ailleurs, une implémentation multi-résolution permet de capturer les motifs à différentes échelles sans alourdir la charge calculatoire.

Enfin, le choix du système de voisinage et, par conséquent, le choix du sens de parcours de la texture de sortie sont remis en question. Le remplacement du parcours lexicographique par un parcours aléatoire permet de synthétiser un pixel en s'affranchissant de son simple passé et donc de multiplier les configurations possibles. Toutefois, le temps de convergence peut dans ce cas augmenter sensiblement. L'adoption de parcours tridimensionnels alternatifs tels que les parcours fractals (e.g. courbe de Hilbert) s'avère un compromis pertinent.

3.2 Algorithme 2D/3D basé sur la vraisemblance de patches

Nous traitons ici d'une approche, de type probabiliste, qui réalise de façon explicite ce que les méthodes basées sur la recherche de voisinage fixe tentent d'accomplir indirectement. Elle étend l'approche markovienne initialement proposée par Paget et Longstaff [*Pag98*]. S'appuyant sur les propriétés de stationnarité et de localité de la texture, cette dernière consiste à générer une texture pixel par pixel, en maximisant la vraisemblance de chaque pixel

au sens de la densité de probabilité conditionnelle locale (DPCL). En d'autres termes, on affecte à chaque pixel le niveau de gris le plus probable connaissant son voisinage. Ce procédé requiert la connaissance de la DPCL, fonction qui modélise de façon unique les interactions entre pixels voisins. Cette dernière est estimée à la volée sur l'image d'entrée, de façon non paramétrique par la technique de fenêtrage de Parzen.

L'approche 2D/3D que nous proposons repose sur le même principe. Elle consiste à synthétiser la texture volumique voxel par voxel suivant un parcours prédéfini. La valeur d'un voxel est mise à jour de façon à maximiser ici aussi sa vraisemblance. Toutefois, l'image modèle étant bidimensionnelle, la densité de probabilité conditionnelle locale 3D, ne peut pas être estimée. La synthèse 3D est alors envisagée selon une approche multi-2D à l'instar des approches présentées précédemment. Ainsi la valeur de voxel choisie doit maximiser la vraisemblance exprimée sur chacune des vues orthogonales auxquelles appartient le voxel. Selon la nature de la texture à synthétiser, deux ou trois vues peuvent être considérées. Il reste toutefois à trouver une stratégie permettant la maximisation des vraisemblances exprimées selon les différentes vues. Plusieurs heuristiques sont proposées, dont celle consistant à maximiser le produit des vraisemblances. Une dernière voie est proposée considérant un système de voisinage 3D composé de plans orthogonaux et exprimant la densité de probabilité conditionnelle DPCL 3D à l'aide de DPCL 2D, estimables sur l'image d'entrée.

Les différentes solutions proposées reposent toutes sur un algorithme de relaxation multi-échelle analogue à celui proposé par Paget et Longstaff [*Pag98*]. Le principe de la synthèse est de débuter, à l'échelle la plus haute, par un bloc 3D aléatoire, de densité de probabilité marginale identique à celle de l'image d'entrée à la même échelle. Chaque voxel se voit attribuer une température non nulle. Sa valeur est modifiée au sens de la DPCL dont le calcul est réalisé en donnant d'autant plus d'importance à un voxel voisin que sa température est basse. L'optimisation est gérée voxel par voxel, par un algorithme déterministe de type ICM. Les schémas de décroissance des températures des voxels sont gérés individuellement. Une fois l'échelle la plus haute traitée, elle sert d'initialisation à l'échelle inférieure. Les voxels y sont actualisés de manière analogue. Nous proposons plusieurs stratégies pour l'initialisation du bloc de sortie, pour la gestion des températures et pour le changement d'échelle, incluant un traitement spécifique des voxels hérités de l'échelle supérieure.

4. Expérimentation et application

La prise en compte des méthodes présentées dans les paragraphes précédents a conduit à l'implantation de différentes variantes d'algorithmes qui permettent d'obtenir une texture volumique a partir d'une seule texture 2D.

Ces différentes méthodes ont été mises en œuvre pour la synthèse d'un jeu de textures variées. Quelques résultats sont présentés dans la *Figure 2*, montrant la légitimité de ces algorithmes. Une analyse comparée et une étude de sensibilité ont été menées, mettant en évidence les atouts et les faiblesses des différentes approches. Les textures volumiques produites sont d'une qualité visuelle convaincante, notamment en termes de dynamique. Les propriétés de structure et de périodicité des images sont respectées mais il semble toutefois, d'un point de vue structural, que certaines approches produisent parfois des textures trop ordonnées ou, au contraire, aient des difficultés à capturer la structure de l'échantillon 2D. Par rapport aux méthodes basées sur la recherche de voisinage fixe, les approches fondées sur la maximisation de vraisemblance disposent de paramètres supplémentaires parfois difficiles à contrôler. Lorsqu'ils sont correctement choisis, des résultats très satisfaisants peuvent être obtenus, au prix toutefois d'un coût calculatoire élevé.

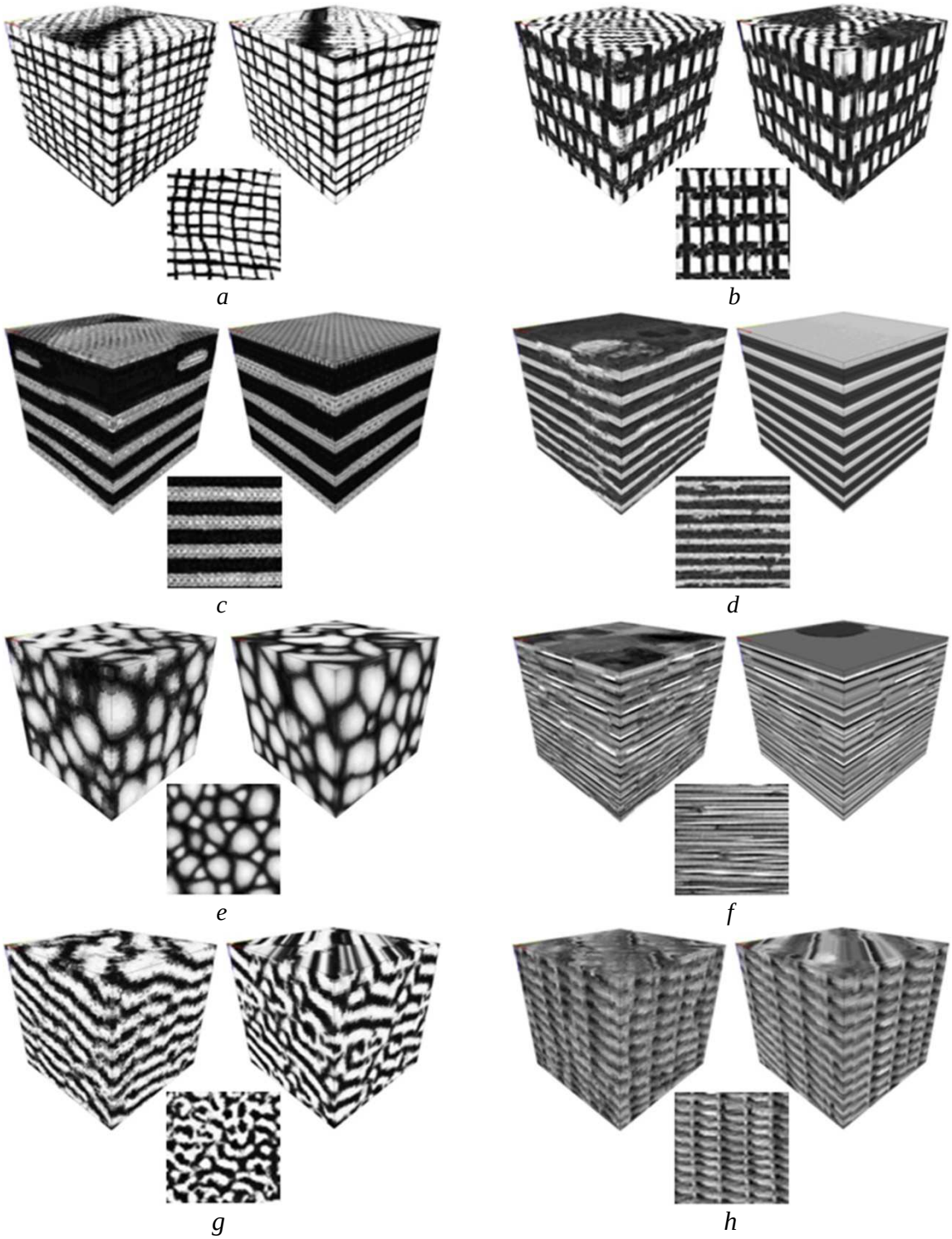


Figure 2 – *Exemples de textures volumiques. Chaque triplet comprend, au-dessous l'échantillon 2D, à gauche le résultat obtenu avec la méthode de Kopf et al. [Kop07] et à droite le résultat obtenu avec notre proposition consistant à maximiser le produit des vraisemblances.*

Les différentes variantes algorithmiques proposées sont également appliquées dans le contexte spécifique de la synthèse de textures volumiques de matériaux carbonés. La texture exemple (comme celles en *Figure 3a* ou *3b*) est une observation 2D unique obtenue par MET (Microscopie Electronique en Transmission) par la technique des franges de réseau. Quelques exemples de textures volumiques synthétisées sont fournies en *Figure 3*.

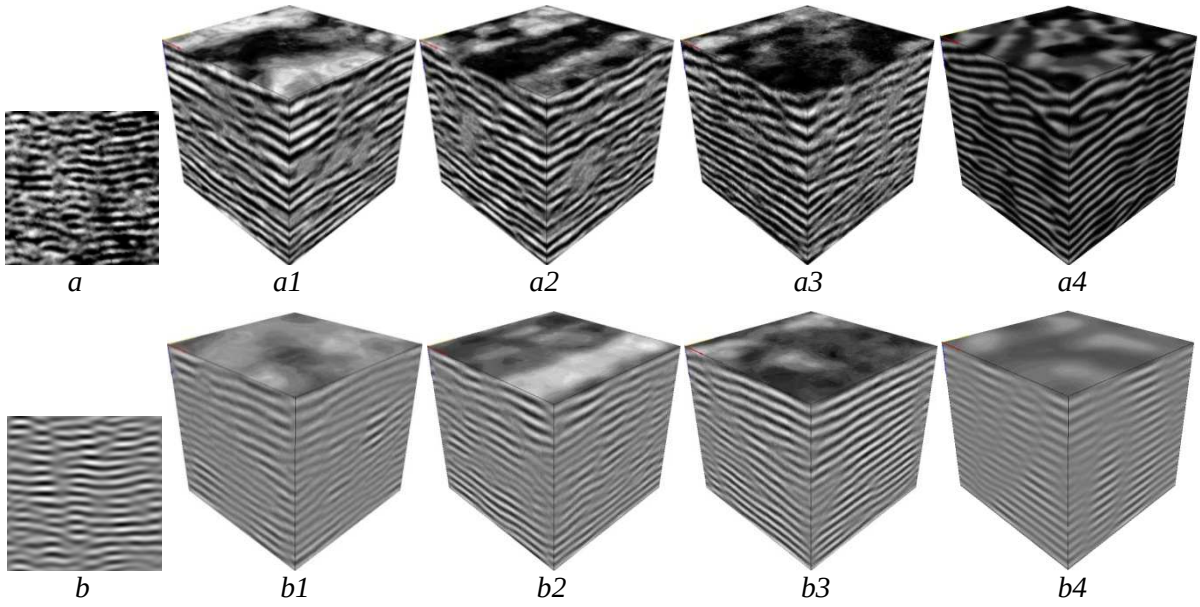


Figure 3 – *Résultats visuels: (a) (b) image type de matériaux carbonés; et les blocs obtenus utilisant les méthodes de (.1) Wei et Levoy [Wei03], (.2) Kopf et al. [Kop07], (.3) Chen et Wang [Che10], (.4) notre proposition (ici avec 32 niveaux de gris seulement).*

En l'état actuel des développements, l'expérimentation a souligné les limites de la méthode basée sur le maximum de vraisemblance, qui tend à produire des textures simplifiées en termes de structure et dynamique par rapport à l'échantillon 2D. Les résultats relatifs aux autres approches sont évalués de façon objective au travers d'une étude quantitative qui permet de comparer les statistiques de niveaux de gris (statistiques d'ordre 1) entre l'échantillon et le bloc de sortie et également de comparer la morphologie des motifs texturaux (orientation locale, longueur et tortuosité de franges) [*Urs12*].

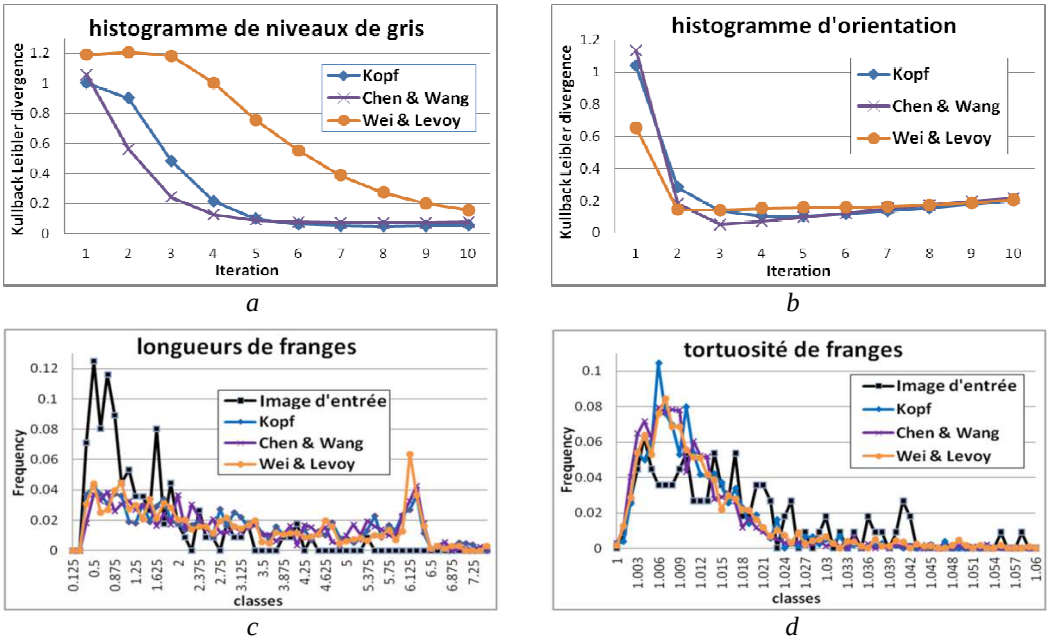


Figure 4 – *Courbes d'évaluation quantitative entre l'image d'entrée et les blocs 3D, contenant des indicateurs pour la comparaison objective: (a) l'histogramme de niveaux de gri, (b) l'histogramme d'orientation locale, (c) longueur et (d) tortuosité de franges.*

Concernant les statistiques d'ordre 1, la comparaison nous montre premièrement l'intérêt de contraindre les histogrammes de niveau de gris [*Kop07*], d'indices et de positions [*Che10*] (cf. *Figure 4a*). Ces derniers permettent en effet une convergence plus rapide de la plupart des statistiques du bloc de sortie vers celles de l'image d'entrée.

Concernant la morphologie, même si les orientations locales sont relativement similaires (*Figure 4b*), la détection de franges (détection de motifs allongés utilisant un algorithme de suivi de courbes de niveau) nous montre que les textures volumiques produites par les approches de type 'recherche de voisinage fixe' contiennent des franges plus longues et plus régulières que l'image d'entrée (*Figure 4c-d*).

L'étude comparative a été élargie à une approche paramétrique de type analyse-synthèse [*Lay09*] [*Dac10*] spécifique aux textures lamellaires et préalablement appliquée aux images de matériaux carbonés. Cette approche s'appuie sur l'inférence 2D/3D des statistiques d'ordre 1 et 2 par le biais d'une décomposition en pyramides 3D orientables. Malgré ses limitations calculatoires et sensitivité à certaines oscillations à haute fréquence, s'avère pour l'instant fournir des résultats plus satisfaisants.

5. Conclusions

Plusieurs algorithmes de synthèse de textures volumiques à partir d'un exemple 2D unique ont été proposés.

Les méthodes de synthèse non paramétriques – dites par patch – ont tout d'abord été privilégiées, en particulier un ensemble d'algorithmes de recherche de voisinages fixes. S'appuyant sur un noyau algorithmique multi-échelle commun, plusieurs approches ont été étudiées, relatives à la manière de combiner les informations provenant des vues orthogonales. Afin d'optimiser l'ordre de visite des voxels lors de la synthèse, des parcours tridimensionnels originaux ont également été proposées.

Dans un second temps, nous avons consacré notre réflexion sur l'élaboration d'un algorithme probabiliste, basé sur l'hypothèse d'un champ de Markov. S'appuyant sur un paradigme 2D, nous avons proposé une extension 2D/3D inédite visant à générer des textures volumiques voxel par voxel. L'approche consiste à maximiser la vraisemblance de chaque voxel au sens de la densité de probabilité conditionnelle locale, au moyen d'un algorithme déterministe de type ICM. Différentes variantes sont proposées, relatives aux stratégies de gestion simultanée des tranches orthogonales contenant le voxel et aux opérations requises par le passage d'une échelle à la suivante.

Les méthodes ont été appliquées à un jeu de textures structurées, de régularité et d'anisotropie variées. Des résultats satisfaisants ont été obtenus en termes de qualité visuelle. Toutefois les algorithmes manifestent une certaine sensibilité à la taille de voisinage et montrent parfois des difficultés à capturer la structure de l'échantillon 2D. Dans le cas des méthodes basées sur la maximisation de la vraisemblance, une simplification de la structure et une réduction de la dynamique sont parfois observées. Toutefois, la qualité remarquable de certains résultats laisse entrevoir des perspectives d'améliorations. Une attention particulière devra dans ce cas être portée à la complexité calculatoire.

Enfin, les approches sont appliquées au cas particulier de textures anisotropes de matériaux carbonés pour lesquelles une procédure expérimentale a été proposée qui vise à une évaluation quantitative et objective des algorithmes de synthèse. Cette étude a porté sur les approches basées sur la recherche de voisinage fixe, et sur une méthode paramétrique d'analyse-synthèse. Les résultats obtenus, en dépit de certaines limites, montrent des résultats convaincants et prometteurs qu'il s'agira d'améliorer à l'avenir, en termes de préservation de la structure et de réduction du temps de calcul.

Abstract

This thesis deals with the synthesis of anisotropic volumetric textures from a single 2D observation. We present variants of non parametric and multi-scale algorithms. Their main specificity lies in the fact that the 3D synthesis process relies on the sampling of a single 2D input sample, ensuring consistency in the different views of the 3D texture. Two types of approaches are investigated, both multi-scale and based on markovian hypothesis.

The first category brings together a set of algorithms based on fixed-neighbourhood search, adapted from existing algorithms of texture synthesis from multiple 2D sources. The principle is that, starting from a random initialisation, the 3D texture is modified, voxel by voxel, in a deterministic manner, ensuring that the grey level local configurations on orthogonal slices containing the voxel are similar to configurations of the input image.

The second category points out an original probabilistic approach which aims at reproducing in the textured volume the interactions between pixels learned in the input image. The learning is done by non-parametric Parzen windowing. Optimization is handled voxel by voxel by a deterministic ICM type algorithm. Several variants are proposed regarding the strategies used for the simultaneous handling of the orthogonal slices containing the voxel.

These synthesis methods are first implemented on a set of structured textures of varied regularity and anisotropy. A comparative study and a sensitivity analysis are carried out, highlighting the strengths and the weaknesses of the different algorithms. Finally, they are applied to the simulation of volumetric textures of carbon composite materials, on nanometric scale snapshots obtained by transmission electron microscopy. The proposed experimental benchmark allows to evaluate quantitatively and objectively the performances of the different methods.

Keywords: *synthesis, solid texture, image processing, multi-scale, causal neighbourhood, Markov Random Field, conditional probability, composite materials*

Résumé

Ce mémoire traite de synthèse de textures volumiques anisotropes à partir d'une observation 2D unique. Nous présentons différentes variantes d'algorithmes non paramétriques et multi-échelles. Leur principale particularité réside dans le fait que le processus de synthèse 3D s'appuie sur l'échantillonnage d'une seule image 2D d'entrée, en garantissant la cohérence selon les différentes vues de la texture 3D. Deux catégories d'approches sont abordées, toutes deux multi-échelles et basées sur une hypothèse markovienne.

La première catégorie regroupe un ensemble d'algorithmes dits de recherche de voisinages fixes, adaptés d'algorithmes existants de synthèses de textures volumiques à partir de sources 2D multiples. Le principe consiste, à partir d'une initialisation aléatoire, à modifier les voxels un par un, de façon déterministe, en s'assurant que les configurations locales de niveaux de gris sur des tranches orthogonales contenant le voxel sont semblables à des configurations présentes sur l'image d'entrée.

La deuxième catégorie relève d'une approche probabiliste originale dont l'objectif est de reproduire, sur le volume texturé, les interactions entre pixels estimées sur l'image d'entrée. L'estimation est réalisée de façon non paramétrique par fenêtrage de Parzen. L'optimisation est gérée voxel par voxel, par un algorithme déterministe de type ICM. Différentes variantes sont proposées, relatives aux stratégies de gestion simultanée des tranches orthogonales contenant le voxel.

Ces différentes méthodes sont d'abord mises en œuvre pour la synthèse d'un jeu de textures structurées, de régularité et d'anisotropie variées. Une analyse comparée et une étude de sensibilité sont menées, mettant en évidence les atouts et faiblesses des différentes approches. Enfin, elles sont appliquées à la simulation de textures volumiques de matériaux composites carbonés, à partir de clichés obtenus à l'échelle nanométrique par microscopie électronique à transmission. Le schéma expérimental proposé permet d'évaluer quantitativement et de façon objective les performances des différentes méthodes.

Mots clés: *synthèse, texture volumique, analyse d'image, multi-échelle, voisinage causal, champ markovien aléatoire, probabilité conditionnelle, matériaux composites*

Table of contents

1. Introduction 1

1.1 Context 3
1.2 Objectives 3
1.3 Thesis outline 4

2. Texture synthesis 5

2.1 Fundamentals 7
2.1.1 What is a texture ? 7
2.1.2 Texture typology 7
2.1.3 Textures applications 9

2.2 2D texture synthesis from a 2D sample: principles and algorithms 10
2.2.1 Texture synthesis definition 10
2.2.2 Texture synthesis applications 11
2.2.3 Texture synthesis typology 13
2.2.3.1 Procedural texture synthesis 13
2.2.3.2 Image-based texture synthesis 14
2.2.3.3 Model-based texture synthesis 17

2.3 Volumetric texture synthesis 19
2.3.1 Procedural approaches 20
2.3.2 Internal solid texturing from 2D cross sections 21
2.3.3 Exemplar-based 3D texture synthesis 21

2.4 Conclusion 24

3. 2D/3D extension based on neighbourhood search 25

3.1 Introduction 27

3.2 Non-parametric synthesis: Wei and Levoy's algorithm 28
3.2.1 Basic principle of the 2D synthesis 28
3.2.2 The 3D extension 30

3.2.3 Multi-resolution implementation 31
3.2.4 Edge handling 33
3.2.5 Computational cost and acceleration 34
3.2.5.1 Tree Structured Vector Quantization 35
3.2.5.2 Nearest-neighbour search using the binary tree 36
3.2.5.3 A concomitant acceleration option 36
3.2.6 Accelerated multi-resolution 2D\3D algorithm 37
3.2.6.1 Iterative search for the best solution 37
3.3 Robust texture optimization 39
3.4 Histogram matching 40
3.4.1 Grey level histograms 40
3.4.2 Index histogram and position histogram 41
3.4.3 Optimized texture synthesis algorithm 44
3.5 Neighbourhood systems and scan type 44
3.6 Implemented algorithms 46
3.6.1 Usage and common implementation details 46
3.7 Conclusion 47

4. 2D/3D extension based on the likelihood of neighbourhoods 49

4.1 Introduction 51
4.2 Non-parametric Markov Random Field model 51
4.2.1 General Markov Random Field model: concept and theory 51
4.2.2 Local Conditional Probability Density Function 53
4.2.3 Texture synthesis via relaxation 55
4.2.4 Multi-scale implementation 55
4.2.5 Pixel temperature function 56
4.2.6 2D texture synthesis algorithm 57
4.3 3D extension of the NP-MRF synthesis algorithm 59
4.3.1 General considerations 59
4.3.2 2D/3D generic algorithm 59
4.3.3 Updating the voxel value: why a full-3D process is not relevant ? 60
4.3.4 Heuristic approaches 61
4.3.4.1 Breaking down the 3D neighbourhood 61

4.3.4.2 Heuristic 1: *NP_ML_H1* 61
4.3.4.3 Heuristic 2: *NP_ML_H2* 62
4.3.5 Implementation details 63
4.3.5.1 Initialisation of the output block 63
4.3.5.2 Scales handling 64
4.4 Conclusion 66

5. Volumetric texture synthesis results 67

5.1 Objective and methodology 69
5.2 Fixed-neighbourhood search based synthesis results 71
5.2.1 Comparing the approaches 71
5.2.2 Parameters influence 74
5.3 Maximum-likelihood based synthesis results 78
5.3.1 Comparing the heuristics 78
5.3.2 Temperature decreasing schema 82
5.3.2 Common parameters influence 84
5.4 Fixed-neighbourhood search vs. likelihood maximisation 87
5.5 About 2D/3D inference 91
5.6 Conclusion 93

6. Application to anisotropic textures of carbonaceous materials 95

6.1 Motivation 97
6.2 Composite materials 98
6.2.1 Material description 98
6.2.2 Pyrocarbons 98
6.2.2.1 From carbon to pyrocarbons 98
6.2.2.2 Pyrolitic carbon 100
6.2.2.3 Structural and textural parameters of turbostratic carbon 102
6.2.3 HRTEM images of interest for synthesis 102
6.2.4 The need for 3D texture synthesis 104
6.3 Volumetric HRTEM texture synthesis 105
6.3.1 Orthotropic properties 105

6.3.2 Results evaluation methodology 106
6.3.3 Fixed-neighbourhood search based synthesis results 109
6.3.4 Maximum-likelihood based synthesis results 116
6.3.5 Parametric vs. non-parametric 3D texture synthesis methods 118
6.3.5.1 A parametric texture synthesis approach 119
6.3.5.2 Results comparison 121
6.4 Conclusion 123

7. Conclusions and perspectives 125

7.1 Synopsis of the work 127
7.2 Future prospects 128

Appendix

A. TSVQ numeric exemplification 131
B. Space filling curves 133
B1. Introduction 133
B2. Lexicographical scanning and the random walk 133
B3. Z-curve 134
B4. Hilbert curve 135
C. MRFs theory 137
C1. Markov Random Fields 137
C2. Gibbs Random Fields 137
C3. Markov-Gibbs Equivalence 138
D. Stochastic relaxation algorithms 139
E. TEM imaging 141
F. Some large snapshots of HRTEM samples 143

Bibliography 147

List of contributed work 159

Chapter 1

Introduction

Contents

1.1 Context 3

1.2 Objectives 3

1.3 Thesis outline 4

1.1 Context

The effectiveness of image synthesis techniques to model and reproduce natural textures (wood, rock, sand, marble, fabric etc.) does not have to be anymore demonstrated. The field of texture synthesis has been particularly dynamic with notable applications in the field of image extrapolation, image editing, image compression or image mapping but has also been extended to other areas such as video completion/merging and animations, or description of the geometry of a surface.

The texture synthesis research field has led to the development of numerous synthesis techniques. These techniques, mainly two-dimensional, need to be adapted for modelling volumetric structures like the ones obtained by 3D imaging techniques, such as Magnetic Resonance Imaging for medical purpose, seismic imaging for underground exploration, or tomography in materials science. In this last case, 3D textures are of particular interest for the study of the three-dimensional internal structure of materials.

However, in some cases, for reasons of cost, practicability, or simply resolution issues, using 3D imaging techniques is not feasible. The three-dimensional structures of the constituent materials are then imaged in 2D. Under these conditions, any stereological method capable of extrapolating 2D information into 3D is likely to improve the understanding of the physical reality of the material. It is in fact in this context that this thesis considers the synthesis of volumetric textures based on 2D images.

1.2 Objectives

The aim of the thesis is to develop algorithms dedicated to the synthesis of volumetric anisotropic textures from a 2D sample. Apart from the difficulties relative to their computational complexity, such approaches pose a 2D/3D inference problem. They are indeed achievable for isotropic textures or, in the case of anisotropic textures – lamellar or wired – only under certain clearly defined hypothesis.

Approaches based on Markovian assumptions, among the most popular and already well-studied, are preferred. It is about extending them to the case of volumetric textures, especially anisotropic ones and focused on grey level textures. The developed algorithms are applied to the modelling of nanometric structures of carbonaceous materials. The synthesis is based on grey-scaled images obtained by electron microscopy. The produced 3D data is intended to be used for realistic simulations of the molecular structure of the material.

Beyond the visual evaluation of the synthesized textures, a quantitative benchmark for the analysis of the synthesized textures is proposed, which consists in comparing input image characteristics with the ones of the output solid texture. This study allows on one hand to identify the most relevant strategies – either parametric or non-parametric – for the synthesis and secondly to compare them objectively, keeping in mind that they should be able to reproduce as faithfully as possible the visual aspect, statistics and morphology of the example texture.

1.3 Thesis outline

This thesis is arranged as follows:

The first part is preparatory, introducing to the reader the notion of texture and the texture typology while acquainting him with the synthesis techniques. It deals with a thorough task of classifying as clearly as possible the synthesis algorithms highlighting the 3D approaches.

Correlated with the previous chapter, the third and the fourth chapters are devoted to the implementation of two types of non-parametric algorithms. Both types are based on the assumption that textures are realisations of Markov Random Fields. The first one, presented in chapter three, consists in synthesis algorithms based on fixed neighbourhood search, building the output texture by direct sampling in the exemplar. Basic principles are presented combined with proposed improvements to increase the synthesis quality. Chapter four considers the synthesis based on the likelihood of patches modelling the dependencies between neighbouring pixels in the exemplar. Every pixel is allocated the most probable grey-level considering only its neighbours. This implies an original 3D extension – and its variants – of an existing algorithm, formerly proposed for 2D texture synthesis.

The fifth chapter is dedicated to the qualitative evaluation of the synthesis results. This is the traditional way to evaluate synthetic textures. A large number of simulated textures are analysed, performed under different frameworks specific to the different versions of the algorithms presented in chapters three and four.

The sixth chapter shows the applicability of the implemented algorithms on a particular set of textured images – the lattice fringe images emerging from the microscopic observation of carbon composite materials. The algorithms were successfully applied to the synthesis of volumetric textures of carbonaceous materials starting from only a single 2D observation obtained by HRTEM (High Resolution Transmission Electron Microscopy). In order to evaluate the ability of the algorithm to reproduce a 3D texture respecting the statistical properties of the input sample, a quantitative study of the performance of synthesis is conducted. This study focuses not only on the dynamics of synthesized images (first order statistics) but also on their morphological properties (lengths and tortuosity of fringes or local orientations).

A research study can never be fully completed, so conclusions are drawn and future perspectives are declared at the end of this dissertation.

Chapter 2

Texture synthesis

Contents

2.1 Fundamentals 7

2.1.1 What is a texture ? 7

2.1.2 Texture typology 7

2.1.3 Textures applications 9

2.2 2D texture synthesis from a 2D sample: principles and algorithms 10

2.2.1 Texture synthesis definition 10

2.2.2 Texture synthesis applications 11

2.2.3 Texture synthesis typology 13

2.2.3.1 Procedural texture synthesis 13

2.2.3.2 Image-based texture synthesis 14

2.2.3.3 Model-based texture synthesis 17

2.3 Volumetric texture synthesis 19

2.3.1 Procedural approaches 20

2.3.2 Internal solid texturing from 2D cross sections 21

2.3.3 Exemplar-based 3D texture synthesis 21

2.4 Conclusion 24

2.1 Fundamentals

2.1.1 What is a texture?

The term *texture* first appeared in the 14th - 15th century, Latin derived, meaning '*to weave*'. If we take into account the definition from the *Merriam-Webster's Collegiate Dictionary* [*Merr*], a texture is defined as:

- something composed of closely interwoven elements
- the structure formed by the threads of a fabric
- the disposition or manner of union of the particles of a body or substance
- the visual or tactile surface characteristics and appearance of something

Ignoring all the metaphoric meanings, in the common speech, 'texture' is used as a synonym for 'surface texture'.

In image processing, it is hard to give a precise definition because texture is alike an *image containing repeated patterns* [*Har92*] with a certain amount of randomness and in the same time can be catalogued as an *image containing no explicit objects* [*Wei09*].

Another good interpretation states that a texture can be defined as a function of the *spatial variation in pixel intensities* (grey values) [*Con80*]. Texture is one of the three fundamental types of features used by humans to distinguish regions in greyscale images; the remaining two are tone and context [*Har73*]. Texture can be regarded as a phenomenon in two levels [*Har73*] [*Gag83*], the first concerns the description of primitives of which the image is composed and which are its characteristic properties; the second involves the spatial dependences between these primitives. Thus, all the spatial properties, periodic or not, of a phenomenon unfolding in the plan of the image can define the texture.

In 3D computer graphics, a texture usually refers to a digital image applied to the surface of a three-dimensional model by *texture mapping* [*Hec86*], to give the model a more realistic appearance.

No satisfactory and universal definition of texture has yet been given, because everyone tries to understand this concept in terms of its center of interest.

Often, textures refer to photographs of real textured materials or surfaces. A variety of such textures are to be found in the well-known Brodatz texture database [*Bro66*] for black and white, and VisTex database for coloured textures [*Visi*]. Five examples of Brodatz textures are shown in the *Fig. 2.1*.

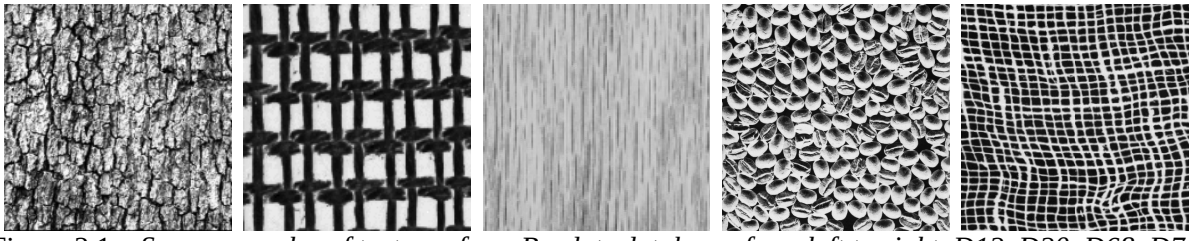


Figure 2.1 – *Some examples of textures from Brodatz database: from left to right, D12, D20, D68, D74 and D103 images.*

2.1.2 Texture typology

Texture is a key component of the human visual perception and if anyone basically can recognise a texture, it is more difficult to classify the textures. A texture is pretty diverse, and can exhibit numerous properties. From a perception point of view, texture can be

described by six different properties, namely, *coarseness*, *contrast*, *directionality*, *line-likeness*, *regularity* and *roughness*, known as the Tamura's texture features [*Tam78*]. If we want finer texture discrimination, these six properties are not as much as necessary [Rao90] [*Cas02*].

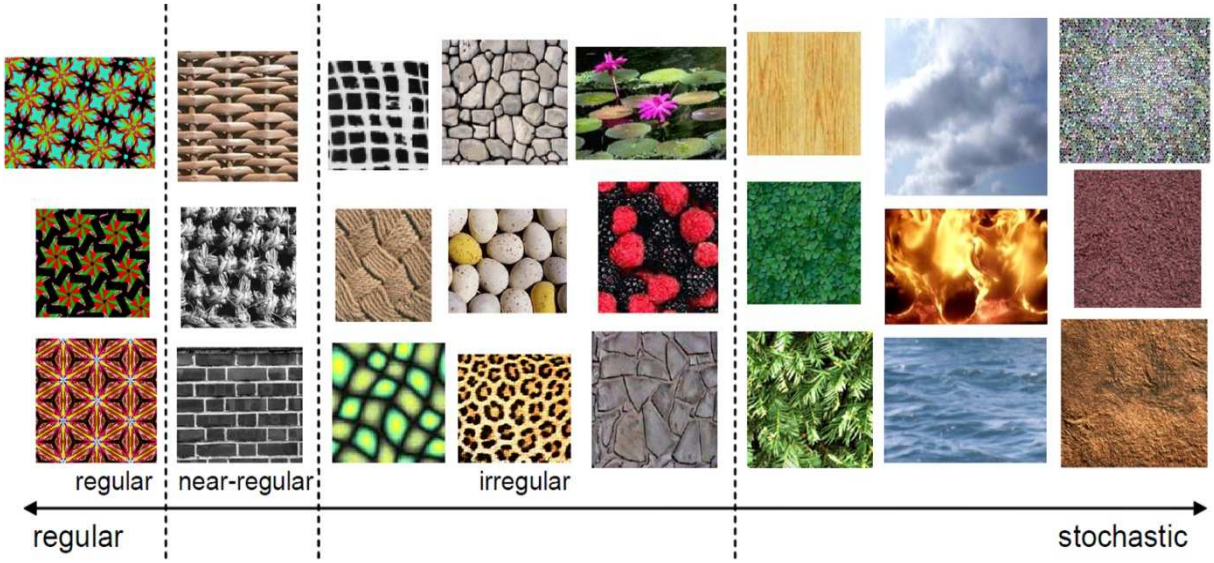


Figure 2.2 – *A texture spectrum arranged by texture regularity [Lin04]*.

However, the most common way to catalogue textures is by their degree of regularity [*Lin04*], along a spectrum going from regular to stochastic as in the *Fig. 2.2*. Most of the real world textures are in-between these two extremes.

Regular textures look like somewhat regular/structured patterns. Their distinctive feature is that the shape and the colour/intensity of all texture elements contained in the texture are repeating in equal intervals. An example of regular textures is wallpaper. Near-regular textures can be viewed as statistical departures of regular textures along different dimensions [*Lin04*]. In the real-world, however, few textures are precisely regular. Most of the time, the textures we see in the real-world are near-regular, such as cloth, windows, brick walls, carpets etc.

Stochastic textures look like noise, like randomly scattered colour dots over the entire image, sometimes specified by minimum and maximum brightness and average colour. Many natural scenes contain a huge number of visual patterns generated by a variety of stochastic and structural processes.

Figure 2.3 – *From micro-texture to macro-texture: from left to right, three different scales of observations, going from low resolution sand texture image to individual pebbles observed at high resolution.*

Many textures look like stochastic textures when viewed from a distance, fact that takes as to distinguish two other types of textures [*Gal11*] as exemplified in *Fig. 2.3*. First, the *micro-textures* are mainly stochastic textures, typical examples of which are images of sand, clouds or a water surface. The second class concerns the *macro-textures*, texture images composed of several small, individual objects and not like a coherent group. The classification into micro or macro-textures does not only depend on the nature of the observed objects but also on the viewing distance.

Unlike macro-textures, the micro-textures contain generally a pattern of which all local spatial parts reveal the same behaviour conferring the texture a homogenous aspect. Some homogeneous textures are presented in *Fig.2.4*.

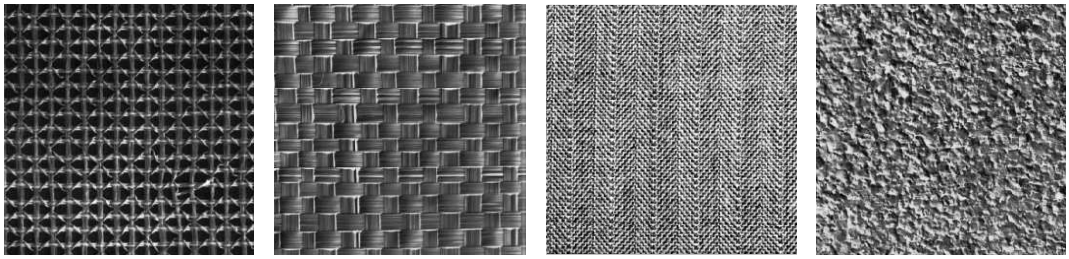


Figure 2.4 – *Some examples of stationary Brodatz textures going from structured to stochastic.*

Intuitively, a texture (or an image in general) is homogeneous if by observing it through different windows, the perceived region properties are similar. The homogeneity is translated into statistical terms by the concept of stationarity. Seeing the texture as a realisation of an underlying random process, its stationarity corresponds to the translation invariance of all the statistical properties of that process. Stationarity can be defined on a broader sense, only at a certain statistical order. For example, a random stationary process of second order has its average, variance and autocovariance invariant by translation. Because the stationarity allows to define, to some extent, simple structural measurements of the random process, recent developments focused on providing methods to evaluate the stationarity of a texture [*Bla07*].

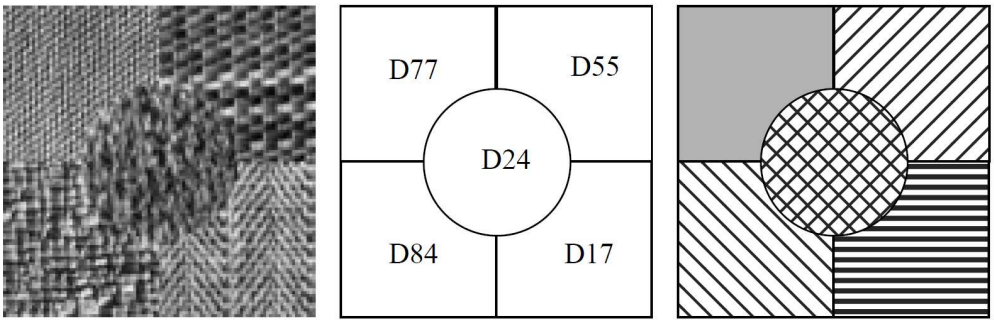
2.1.3 Textures applications

The use of textures is very valuable in many applications. Quite a lot of computer vision tasks use textures, such as recognition, classification or segmentation as illustrated in *Fig. 2.5*.

Texture has been proved to be one of the significant characteristics used in identifying objects of interest or regions in an image [*Skl78*]. Using texture features, one can produce a classification map of the input image where each homogeneous textured region is identified with the texture class it belongs to [*Con80*]. A typical application consists in retrieving similar regions in remote sensing images [*Wik01*].

Texture has been studied in the context of image understanding and analysis. Not surprisingly, texture finds applications in problems related to image/video editing, merging, and completion, setting up a new concept – video texture [*Sch00*], enjoying properties found somewhere between a photo and a video.

Textures improve our lives indirectly by contributing actively in the medical research field, namely in investigating medical images [*Dun00*], an important tool for diagnosis and pathology follow-ups.



a *b* *c*

Figure 2.5 – *(a) An image consisting of five different Brodatz textured regions [Che98]: cotton canvas (D77), straw matting (D55), raffia (D84),herringbone weave (D17) and pressed calf leather (D24). (b) The goal of texture classification is to label each textured region with the proper category label. (c) The goal of texture segmentation is to separate the regions in the image which have different textures and identify the boundaries between them.*

Reproducing the visual realism of the real world is a key objective for computer graphics. For acquiring synthetic images, textures are frequently used and above all obtained by texture synthesis. Synthetic textures are an alternative to the hand-drawn or scanned photographic textures, having the advantage that textures can be made of any size and that repetitions or visual seams can be avoided.

In 2D but lately mainly in 3D graphics, a large interest is in modelling surface details. Because explicit modelling with polygons or geometric primitives becomes less practical for finer and more complicated details, an alternative solution is to map an image, either synthetic or digitized, onto the object surface, technique called texture mapping [*Hec86*].

2.2 2D texture synthesis from a 2D sample: principles and algorithms

Texture synthesis has been an active research topic in computer vision both as a way to verify texture analysis methods, as well as in its own right. It is often the case when a given sample is too small for the surface of an object. Then the sample needs to be extended in some way - and that is where texture synthesis comes in. Simple replication of the sample would cause a tiled appearance, but texture synthesis will create a new texture big enough that will still look like the original one.

2.2.1 Texture synthesis definition

More specifically, the goal of texture synthesis is to reproduce a new texture from a sample, obtaining a texture that looks different and is pixel-wise different from the original sample.

The synthesized texture appears as if it has been created from the same underlying process as the original one, both measured by the standards of human perception. The output texture should ideally be perceived as another part of the same large portion of a homogeneous material the input texture is taken from, as it is illustrated in *Fig. 2.6*.

The size of the output texture is typically selected by the user, the synthesis result should be perceptually similar to the input texture but also containing sufficient variations, so that it should not have any visible artefact (seams, blocks, misfitting edges) and it should contain no repetition.

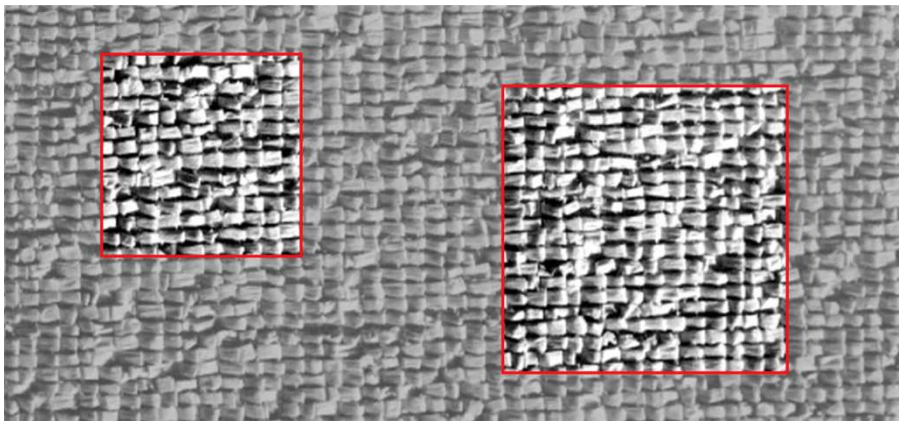


Figure 2.6 – *Example of an ideally texture synthesis process on Brodatz D84 image: given an input texture (left snapshot), a texture synthesis algorithm should ideally provide an output texture (right snapshot) perceived as being a different part of the same large piece of material.*

A very good description of texture synthesis was formulated by [*DeB97*]: *"Mathematically, the goal of texture synthesis is to develop a function F, which takes a texture image I_{input}, to a new texture sample I_{synth}, such that the difference between I_{input} and I_{synth} is above some measure of visual difference from the original, yet is texturally similar. Formally,*

$$F(I_{input}) = I_{synth} \tag{1.1}$$

subject to the constraints that

$$D^*(I_{input}, I_{synth}) < T_{max\_disc} \tag{1.2}$$

and

$$V^*(I_{input}, I_{synth}) > T_{min\_diff} \tag{1.3}$$

where D^* is a perceptual measure of the perceived difference of textural characteristics, and V^* a measure of the perceived visual difference between the input and synthesized images. To be acceptable, the perceived difference in textural characteristics must fall below a maximum texture discriminability threshold T_{max_disc}, and the perceived visual difference must be above a minimum visual difference threshold, T_{min_diff}.

The success of a synthesis technique is measured by its ability to minimize T_{max_disc} while maximizing T_{min_diff}".

2.2.2 Texture synthesis applications

In computer graphics, texture synthesis is a common technique to create generally large textures from small texture samples, for the use of *texture mapping* in surface or *scene rendering* applications, handling boundary conditions and avoiding verbatim repetitions. In rendering, textures can imitate the surface details of real objects, ranging from varying the surface's colour, to actually deforming the surface geometry.

In computer vision, texture synthesis is of interest in *segmentation*, *recognition* or *classification* as in *Fig. 2.5*. These tasks can benefit from a texture model, which could be derived from a successful texture synthesis algorithm. Texture synthesis is useful also because it could provide an empirical way to test *texture analysis*, if the analysis results in some characteristic features that can be implemented into the synthesis algorithm. Because a synthesis algorithm is usually based on texture analysis, the result justifies effectiveness of the underlying models. Compared to texture classification and segmentation, synthesis poses a bigger challenge on texture analysis because it requires a more detailed texture description and also reproducing it is generally more difficult than discriminating them [*Zho06a*].

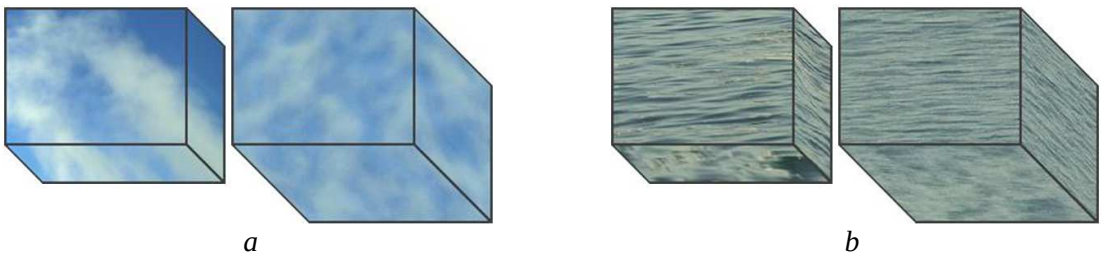
Other applications of texture synthesis comprise:

➢ *Image editing/restoration/completion* [*Ige97*] [*Efr99*] [*Wei00*] [*Bro02*] [*Dro03*] – Over time or due to improper handling or maintaining, photographs or films suffer from scratches or scrambled regions; more often, in pictures or movie frames there are undesirable items (like wires, poles, different objects, persons, animals etc.). All these kind of flaws, often contained within a textured region, can be replaced by texture synthesis. Texture synthesis can also be used as an inpaiting process for reconstructing lost or deteriorated parts or filling the holes in images and videos. Some examples in this direction are in *Fig. 2.7*.

a *b* *c* *d*

Figure 2.7 – *(a)(b) Examples of texture-replacement algorithm [Ige97]: top row - the original image, the bottom row – image with replacement patch by texture synthesis. (c)(d) Examples of image contraction (c) and image expansion (d) using self-similarity based texture warping [Bro02].*

➢ *Animation and video synthesis* [*Szu96*] [*Sch00*] [*Wei00*] [*Kwa03*] [*Dub11*] – Computer generated animations contain cyclic background movements such as ocean waves, waterfall, steam, clouds, smoke or fire. These motions can be regarded as temporal textures with motion having indeterminate extent both in space and time as exemplified in *Fig. 2.8*. Video textures provide a continuous infinitely varying stream of images; they can be used in place of digital photos to infuse a static image with dynamic qualities and explicit action. Applications of video textures and their extensions include the display of dynamic scenes on web pages, the creation of dynamic backdrops for special effects and games, and the interactive control of video-based animation.



a *b*

Figure 2.8 – *Temporal texture synthesis [Wei00]: (a) smoke and (b) ocean waves; in each pair, the spatial-temporal volume of the original motion sequence is shown on the left, and the corresponding synthesis result is shown on the right.*

➢ *Compression* [*Bee96*] [*Sun10*] – Images showing natural scenes often contain large textured regions, such as a grass land, a forest, or a sand beach. Alternative to common image compression formats like JPEG or PNG, textures can be compressed as well, using a different

format. By segmenting out and analysing the textured regions in a pre-processing step, they might be compressed and afterwards re-synthesized by a texture synthesis technique. In addition to image compression, texture synthesis can also be employed for synthetic scenes containing large amounts of textures.

2.2.3 Texture synthesis typology

The research field of texture synthesis has lead to the development of many synthesis techniques over the past two decades. Obviously, the space prevents a complete listing of all the relevant models, but whatever the difficulties, one can roughly separate the existing texture synthesis methods in three major families: *procedural texture synthesis, image-based texture synthesis* and *model-based texture synthesis*.

2.2.3.1 Procedural texture synthesis

Procedural texture synthesis focuses on trying to produce textures using functions or computer algorithms which can be evaluated at any point at a fixed computational cost, and thus they are ideal tools for texturing objects in virtual environments - animation movies or video games [*Ebe02*]. Different authors tried to reproduce the real-world phenomenon like water flow, corrosion or particles distribution in aggregate materials, using physical simulations based on meticulous mathematics.

Such procedures were studied since the early days of computer graphics. Usually they are based on transforming some predefined signal into a desired texture. It's easier to set up functions for highly structured textures, but many of them are also useful for generating some types of non structured textures like noise functions (e.g. the Perlin noise [*Per85*] or Worley noise [*Wor96*]). Perlin noise is still very popular today. Over the last few years several improved procedural noise functions have been proposed: wavelet noise [*Coo05*], anisotropic noise [*Gol08*] or Gabor noise [*Lag09*].

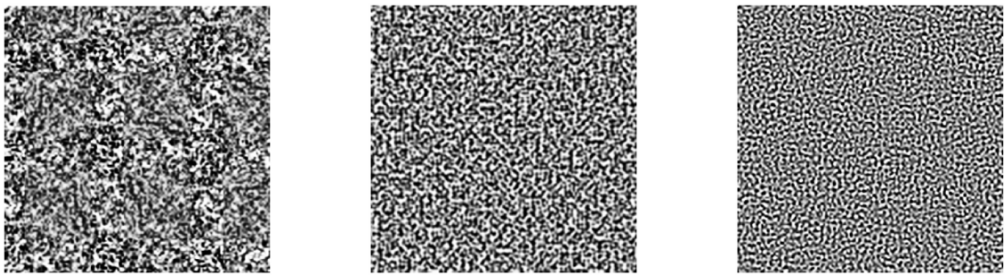


Figure 2.9 – *Some examples of noises representations: from left to right, white noise, Perlin noise [Per85] and wavelet noise [Coo05].*

These approaches can be very fast, they can produce high-quality and continuous memory efficient textures [*Ups89*] not storing explicitly the synthesized textures. Unfortunately, they are specialized for a reduced number of specific textures, such as wood, water, marble, sea shell or animal skin; for every new texture it is required a new algorithm to be written and programmed. Hence, an important practical problem is to derive texture procedural models from sample textures [*Dis97*]. If the early results were limited to textures in 2D, most of the authors tried to extend their algorithms for synthesizing solid textures [*Lew89*] or for solid texturing on surfaces [*Pea85*] [*Tur91*]. Another direction of procedural methods includes the use of reaction-diffusion [*Tur91*] to create striped or spotted patterns.

2.2.3.2 Image-based texture synthesis

This category of approaches contains most of the synthesis techniques. Image based texture synthesis generates a new texture that resembles as much as possible the sample image.

Image-based texture synthesis approaches involve three sub-classes as following:

- texture synthesis by analysis
- pixel-based texture synthesis
- patch-based texture synthesis

Image-based approaches can synthesize a wider variety of textures, as long as sample textures are provided. Most of them synthesize textures by directly copying input image pixels or patches and stitching them together in the synthesized image. They have the advantage of preserving image details by keeping the pixel neighbourhood intact in the synthesized textures. These are local approaches in nature, with no special consideration given to the texture's global structures.

Texture synthesis by analysis is based on texture modelling by statistical constraints. A new texture is synthesized so that a set of statistical constraints estimated from the input image is imposed on the output textures, idea illustrated in *Fig. 2.10*.

The first notable approach constrained the synthesis by using the histograms of filter responses at different scales and orientations [*Hee95*]. This was one of the first coloured textures synthesis methods, and involved an iterative approach of matching the histograms and expanding and reducing the pyramids. Image construction was made by using a sort of Laplacian and Steerable pyramids. The next in line, overcame the iterative process by modifying the input in a coarse-to-fine fashion, restricting the conditional distribution of filter output over multiple resolutions [*DeB97*].

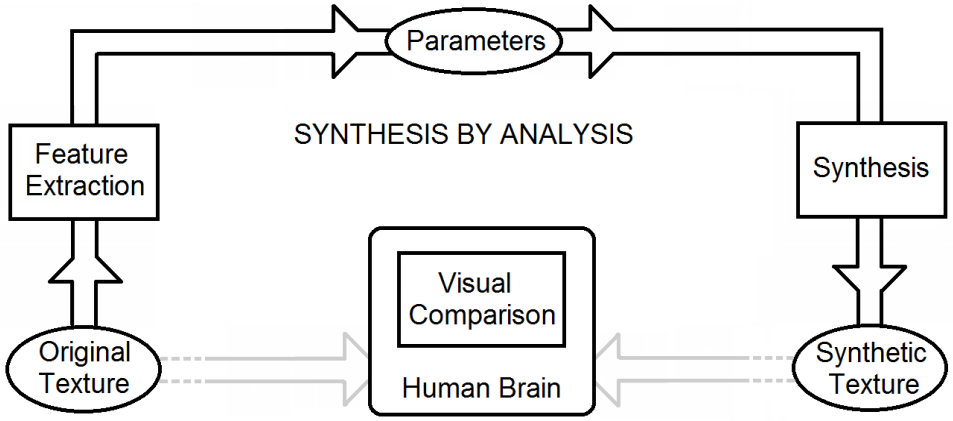


Figure 2.10 – *Block diagram of the texture synthesis by analysis method and its goal: to achieve the same visual appearance in both the original and the synthetic textures.*

A substantial improvement was accomplished by extending the first order statistical modelling to the second order one and replacing the complete filter response update, made by histogram equalization, with a scheme respecting the correlations [*Por00*]. The optimization procedure becomes more complicated, but it allows a fairly sufficient description of texture and makes the synthesis of new textures possible.

Several enhancements of this method have been proposed [*Pey10*] [*Rab10*], involving higher-order statistics between coefficients and more sophisticated decompositions of images. However, these types of approaches lead to some complex formulations that are difficult to optimize.

Pixel-based texture synthesis methods generate the texture one pixel at a time maintaining the coherence of the local texture with its vicinity. They are generally based on the theory of Markov Random Fields (details are to be found in *Chapter 4*). These are among the most successful techniques mainly because of their simplicity and applicability.

To assure a reasonable computational complexity, most of the authors propose efficient searching algorithms combined with a multi-scale implementation capable of capturing patterns at different scales without increasing the computational load.

One of the first methods, classified as a nonparametric Markov chain synthesis algorithm, consisted in ordering the pixels and then synthesizing a new pixel from a nonparametric representation of the conditional probability function. This probability function was derived from samples of the input texture [*Pop93*], using a Gaussian mixture model. The only problems were the limited size of the neighbourhood order and the causality of the synthesis approach. Improvements were brought by using the non-parametric Parzen-window density estimator and a local annealing algorithm capturing the visual characteristics of a texture [*Pag98*]. This model was capable of synthesizing complex textures ranging from the stochastic to the well structured ones.

Following the above work, the texture synthesis by non-parametric sampling [*Efr99*] doesn't require any probability density estimator, but instead simply using the nearest neighbour look up scheme to sample the texture, principle sketched in *Fig. 2.11*.

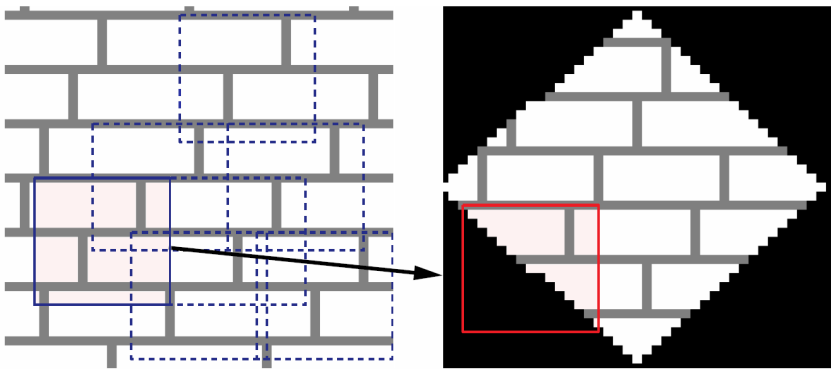


Figure 2.11 – *Overview of the texture synthesis by a non-parametric algorithm [Efr99]: Given a sample texture image (left), a new image is being synthesized one pixel at a time (right). To synthesize a pixel, the algorithm first finds all neighbourhoods in the sample image (boxes on the left) that are similar to the pixel's neighbourhood (box on the right) and then randomly chooses one neighbourhood and takes its centre to be the newly synthesized pixel.*

Even if Efros's algorithm was causal and non multi-scale, this was the precursor of one of the most versatile methods and in the same time, one of the fastest. This method, proposed by Wei and Levoy [*Wei00*], relies on texture locality (every pixel is predictable from the few pixels in its neighbourhood) and texture stationarity (spatial statistics invariance by translation). Starting from a random initialization and following a certain scanning type, the texture is synthesized pixel by pixel, iteratively. To do that, pixels from the exemplar are copied into the output texture, making sure that the output pixel neighbourhood is similar to the neighbourhood of the chosen input pixel. To accelerate the searching of the best neighbourhood in the input image, the authors proposed acceleration techniques.

Because the exhaustive nearest neighbour searching is time consuming, an efficient solution is to reduce the number of causal neighbourhoods per pixel to be compared [*Ash01*]. These neighbourhoods are defined as the neighbourhoods in the sample texture that have been previously chosen for the former synthesis of pixels neighbouring the pixel of interest in the output texture. This algorithm works best on natural textures, such as textures of flower fields, pebbles, forest undergrowth, bushes and tree branches. However, it is not suited for textures containing a high degree of structure, when phase discontinuities occur, making the texture look broken.

Combining the ideas from [*Wei00*] with [*Ash01*] and extending them to work on corresponding pairs of images rather than on single textures has lead to a new type of algorithm, capable of handling texture synthesis and texture transfer using analogies [*Her01*], efficiently combining the Euclidian distance with the nearest neighbour searching.

Other approaches proposed to accelerate texture synthesis by using a jump map [*Zel02*], avoiding nearest neighbour comparisons as in *Fig. 2.12*. Each pixel in the jump map contains a list of pre-calculated references and probabilities for matching pixels, as an input texture analysis stage. A texture is synthesized in real time by copying a matching pixel, referred in the jump map, from the sample texture.

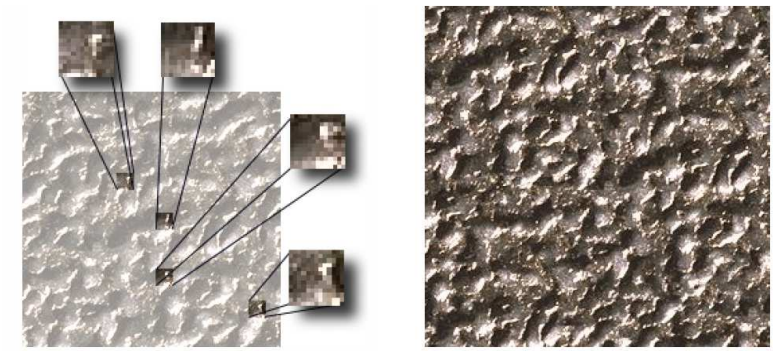


Figure 2.12 – *Basic principle of texture synthesis using the jump map [Zel02]: the jump map records the links between closely-matched neighbourhoods from the input texture (faded for clarity).On the right, the output image is synthesized in a scan-line order, by random walk through the jump map.*

Following an entirely different principle, [*Ton02*] introduced the k-coherence-based method capable of synthesizing bidirectional texture functions (functions that describe the appearance of a real-world surface as a texture function of lighting and viewing directions) as an application to texture synthesis. K-coherence supplies for each pixel a set of *k* nearest causal matches. The source pixels in the causal neighbourhood are used to define a candidate set from which the best matching pixel is copied. The value of *k* depends on the type of texture: for natural textures, where high frequency components are desired, a low *k* should be suitable; for the rest, a high *k* should be used. If *k* equals 1, the method is comparable to [*Ash01*].

Patch-based texture synthesis appeared as an improvement to the pixel-based approaches, the latter methods suffering from excess computations when dealing with the structure of the texture. At each step, a patch of the input image is chosen among the ones which have their neighbourhood similar to the corresponding neighbourhood in the output texture.

One of the first approaches of this kind consists in selecting patches from the input image that agree, according to some measure, with surrounding patches in the output image and then reducing the edge artefacts by using minimum error cuts between the patches. This can be simply defined as image quilting [*Efr01*] and the concept is illustrated in *Fig. 2.13*.

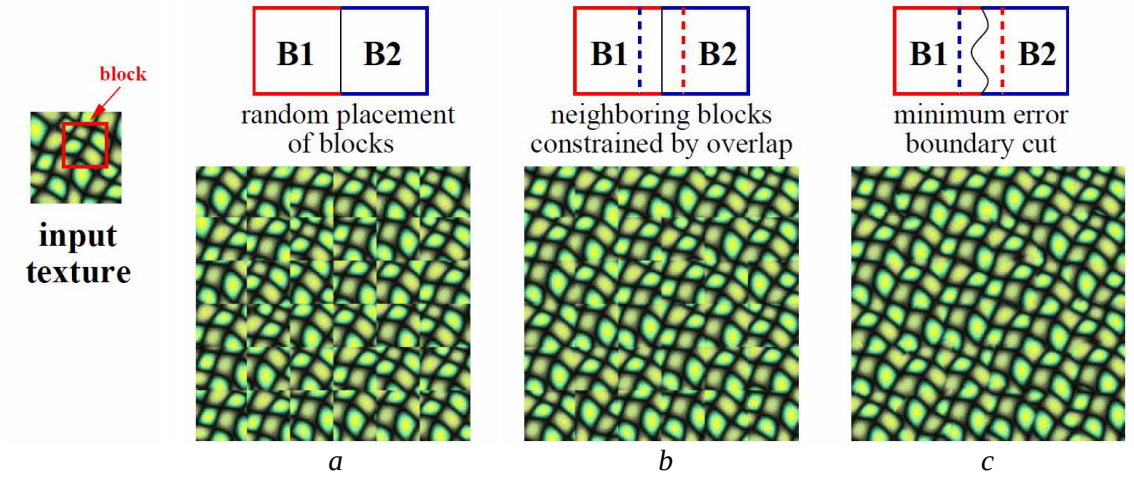


Figure 2.13 – *Example of texture synthesis using the image quilting method [Efr01], i.e. square blocks from the input texture patched together to synthesize a new texture sample: (a) blocks are chosen randomly (similar to [XuG00] [Pra00]), (b) the blocks overlap and each new block is chosen so as to "agree" with its neighbours in the region of overlap, (c) to reduce blockiness the boundary between blocks is computed as a minimum cost path through the error surface at the overlap.*

Patch-based texture synthesis was also performed under the form of chaos mosaics [*XuG00*] - randomly distributing patches from the input texture over the output and smoothing the edges between overlapping patches with simple cross-edge filtering. The drawback is that the synthesis produces large regions of verbatim copying.

To reduce the stability problems of which suffered the previous schemes, a fast neighbour search algorithm based on a quad-tree pyramid structure of the input texture was introduced. It assures an optimal sequential fitting technique, by laying down one patch after the other [*Lia01*].

An improved version of image quilting was done by a graph cut technique called min-cut or max-flow [*Kwa03*] allowing re-evaluations of old cuts relative to new ones. It involves two stages: first, the search of the best patch by comparing the current patch with the information from the sample texture and secondly, copying an optimal portion of the patch determined using a graph-cup algorithm. The results have a very good visual quality even when applied to video texture synthesis.

Work was done to reduce and to guarantee the non-repeating texture patterns while synthesis, by improving texture lapping [*Pra00*] or by introducing a new concept – Wang tiles [*Coh03*]. Wang tiles generate non-repeating tiling of a limited number of tiles by assigning a colour to each edge of the tile and matching edges with similar colour.

Combining the patch-based and the pixel-based techniques led to an interesting hybrid algorithm highlighting the advantages of each method and restraining the drawback of both [*Nea03*]. It consists in using patches as large as possible and stitching the patches of different sizes using a pixel-based method.

For more information about the example-based texture synthesis algorithms the reader is invited to consult [*Kwa07*].

2.2.3.3 Model-based texture synthesis

Model-based texture synthesis is a complementary topic of texture analysis, attempting to generate synthetic textures in harmony with a certain texture model. Model based texture analysis methods consist in building an image model that can be used to describe the texture but also to synthesize it. The parameters of the model have to confine the

important visual characteristics of the texture. Many model based methods have been employed in texture analysis, including the auto-regressive models [*Kas84*] [*Che85*] [*Jai89*], Markov and Gibbs random fields [*Cro83*] [*Gem84*] [*Kho87*] [*Siv99*], Wold models [*Fra93*], fractal models [*Che93*], Gabor and wavelet models [*Tur86*] [*Cla87*] [*Mal89*] etc. In these methods, a texture is shaped as a probability model or as a linear combination of a set of basic functions. Difficulties are on choosing the appropriate model for the targeted texture and on estimating its parameters.

The autoregressive (AR) model assumes a local interaction between texture pixels where pixel intensity is a weighted sum of neighbouring pixel intensities. Derived from 1D time series models that have a past and a present [*Kas84*], these 2D models have to impose similar ordering on the 2D discrete lattice. Other AR models, such as the moving average model (MA) or the autoregressive moving average model (ARMA) are discussed in [*Jai89*]. Although texture synthesis is feasible using for example the non-causal AR model [*Che85*], these model-based methods show difficulties in generating natural textures [*Hai91*].

Markov random fields (MRFs) are popular methods for modelling images [*Bes74*] [*Cro83*] [*Gem84*] [*Pag89*]. A MRF is a probabilistic process in which all interactions are local, but the global effects can still occur as propagation effects. The intensity of each texture pixel is entirely determined by its neighbouring intensities, deemed as a conditional probability. Because a conditional probability density function is not accurately estimated by the MRF, an equivalent maximum a posterior estimator on a Gibbs random field is used [*Cli90*] [*Gem91*]. Consequently, modelling the texture globally by specifying the total energy of the texture lattice is equivalent to model it locally by evaluating the local interactions between pixels in terms of conditional probabilities.

Wold-like model is a more recent method used to model texture images. The model allows the texture to be decomposed into three mutually orthogonal components [*Fra93*] [*Sto98*]: a random component (w.r.t. granularity), a harmonic component (w.r.t. repetitiveness) and a generalized evanescent component (w.r.t. directionality) [*Ram00*]. A 3D Wold decomposition was proposed recently by [*Sti07*], modelling a 3D homogeneous texture field as a unique sum of four mutually orthogonal components: a purely indeterministic part, a deterministic part and two evanescent components. This kind of approaches allows texture synthesis [*Fra93*] [*Zha08*] but with difficulties in estimating the harmonic and evanescent components.

Fractal models are accepted in computer graphics for creating realistic textured images. This is feasible because the fractal distance is relatively insensitive to image scaling but more important it shows strong correlation with the human judgement of surface roughness. Fractals are used to model the roughness property of a surface/texture in image analysis, being able to describe objects having high degree of irregularity. A number of methods for texture synthesis based on fractals have been proposed, based on midpoint displacement and Fourier filtering [*Pei88*]. The fractal method is able to model some natural textures, but because it lacks orientation selectivity, is not suitable for describing local image structures.

There are also other families of classes in which texture synthesis techniques can be broadly categorized, for example *local region-growing methods* [*Pag98*] [*Efr99*] [*Wei00*] vs. *global optimization based methods* [*Per85*] [*Pea85*] [*Tur91*] [*Hee95*] [*Por00*] or discerning the same referenced paper in *parametric methods* [*Per85*] [*Pea85*] [*Tur91*] [*Hee95*] [*Por00*] vs. *nonparametric methods* [*Pag98*] [*Efr99*] [*Wei00*].

2.3 Volumetric texture synthesis

Many of the above techniques were implemented for 2D synthesis, but their extension to the 3D environment remains unstable proving itself as a very complex and computational issue. 3D textures are mainly used for texturing volumetric objects trying to increase the realism of the 3D scenario, but they can also be observed in 3D vision when exploring for instance material structure or seismic data. For such applications, a volumetric synthesis approach by 2D/3D inference is justifiable, being sometimes the only possibility to access the three dimensional structure [*DaC10*].

Volumetric texture synthesis has proved very useful for texture mapping. Mapping represents an important instrument in modelling surface details for rendering photorealistic graphic scenes without explicit modelling of the surface geometry or of the material properties. Nevertheless, defining a distortion-free and discontinuity-free mapping is challenging and sometimes even impossible for some complex objects. In contrast, when applying a solid texture into an object, it allows carving the object out of the texture volumetric data block, avoiding the problems of distortion and discontinuity [*Dis01*]. Furthermore, once a solid texture is available, it can be used to texture arbitrary objects not only on object surface, but also throughout the entire object volume, the solid texture carrying information in the entire volume. One should make the distinction between the 3D solid texture synthesis and the on-surface 2D texture synthesis. Many works were developed in this direction by synthesizing a 2D texture pattern over a mesh in 3D [*Wei01*] [*Yin01*] [*Tur01*] [*Ton02*] [*Gor03*] [*Zho06b*] with the objective of avoiding seams and minimizing pattern distortion.

For many objects, mapping 2D images (i.e. texture maps) to the surface of synthetic objects is satisfactory, '*tricking*' the human perception by not incorporating quantitatively the third dimension of depth. However, for objects formed from materials with inherent 3D structure, it can be difficult to find a 2D image mapping that adequately represents the material variation. Here gets involved the 3D synthesis. Full 3D texture synthesis versions can be easily derived for most of the existing 2D approaches and can be used in practice, provided that a 3D sample is available as an example. On the contrary, the synthesis of volumetric textures based on 2D exemplar(s) is more awkward.

Despite the fact that there have been numerous studies successfully performing texture synthesis, in terms of quality and efficiency, most of the works concentrate on 2D texture synthesis, while solid texture synthesis receives relatively less attention. The scarcity of related papers is mainly attributed to the much higher complexity involved in solid texture synthesis.

Unlike the 2D image samples, which can be captured with any camera, 3D solid texture image samples are more delicate and elaborated. In material science and engineering, exploring the 3D structure of materials is essential to understand and predict their physical properties and behaviour. There are different 3D imaging techniques (based on X-ray, ultrasounds or Magnetic Resonance Imaging) allowing to obtain 3D views of a certain material at different resolutions, but their use is not always appropriate because of the high costs it involves and for various technical reasons (e.g. the sample preparation, intricate for high resolutions). Alternative ways to obtain 3D images of a material are the confocal microscopy [*Ebe01*] [*Lee01*] (reconstructing 3D views from multiple optical sections of the sample) or the dissector [*Ste84*] [*Dav97*] (observing thin, parallel, contiguous slices of the sample). However, 3D microscopy techniques are expensive and imply rigorous sample preparation, while getting thin enough, perfectly parallel slices is quite laborious in practice, making them sometimes unusable.

3D image synthesis techniques may appear as interesting alternatives in such cases. Specific techniques are to be considered, using at least one 2D image of the 3D material as input. 3D information can be obtained from one 2D image by modelling the targeted solid block to match the 2D sample characteristics. As well it is possible to relate measurements of a single 2D cross-section of the volume with the reality of that material under stereological considerations.

One can distinguish three main directions for creating volumetric textures:

- procedural approaches
- run-time 2D texture synthesis on cross sections
- exemplar-based 3D texture synthesis

2.3.1 Procedural approaches

These are the first solid synthesis techniques ever attempted. In fact, the notion of solid texture was first employed in 1984 [*Gar84*], but the term solid texture was officially pioneered in 1985 by [*Per85*], when the reproduction of realistic textures was done by perturbing material-specific mathematical functions in order to create realistic, pseudo-random patterns, using noise functions [*Lew89*]. Noise functions have been largely used in computer graphics to reproduce solid textures containing patterns of marble, crystal, sand, clouds, fire or water. Lately, new attempts were made to improve the intuitive description of patterns [*Lag09*].

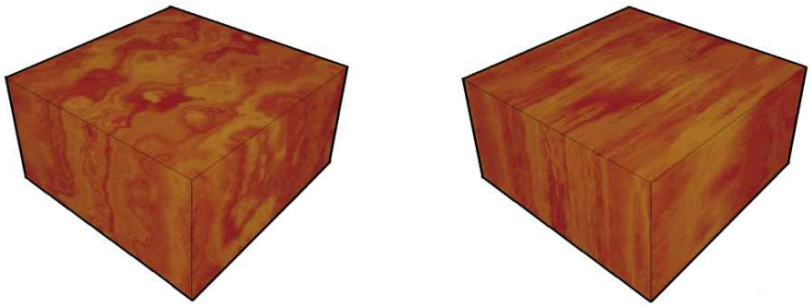


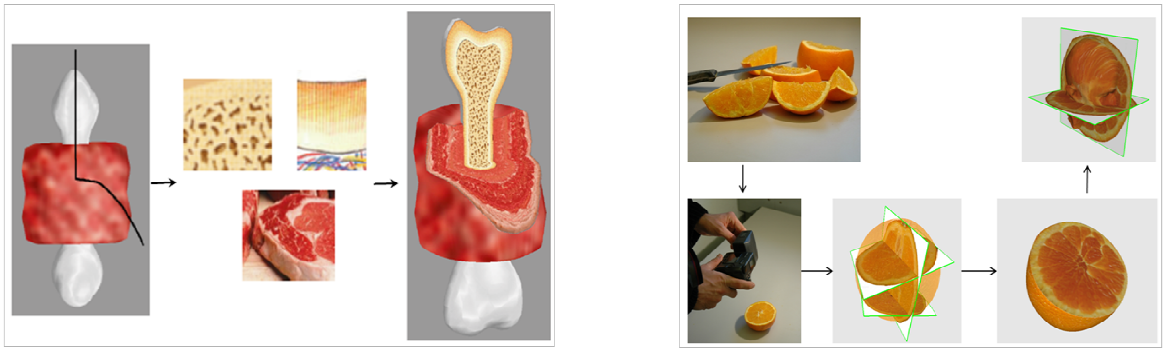
Figure 2.14 – *Solid wood texture synthesis [Dis01]: on the left, the result obtained by perturbing a wood-grain basis function directly with the turbulence function of [Per85]; on the right, the result obtained by using a perturbation function corresponding to a filtered noise which preserves the visual aspect of the wood-grain.*

Another type of procedural approach consisted in using a scripting language to define the internal structure of a layered volume model by specifying depth and material information [*Cut02*]. This approach is capable of producing a variety of complex models as a result of simulation operations applied to the layered solid models. Based on procedural methods, [*Che04a*] presented an efficient 3D texture synthesis algorithm simulating the growth of a material from a single 2D growable texture pattern. Texture evolution is based on the general principle of texture growing (i.e. 3D cube growth simulation following a random path) and texture turbulence (i.e. texture warping by texture perturbation).

Unfortunately, it remains quite difficult to analytically describe texture appearance by procedural methods leading to algorithms hard to control, program and optimize.

2.3.2 Internal solid texturing from 2D cross sections

It consists in synthesizing the internal structure of a 3D model from 2D snapshots of a few cross-sections of a real object, using 2D synthesis techniques, instead of sampling from a complete 3D volumetric representation directly [*Owa04*] [*Pie07*].



a *b*

Figure 2.15 – *(a) Example of a subdivided domain [Owa04] based on user's control, to choose where to cut a bone-and-meat model (on the left) and to add interesting reference textures (on the middle) to the surface meshes. (b) Illustration of the internal structure of an object using a few photos of internal surfaces of a real object taken by an interactive editor and place them in the local reference frame of the 3D model [Pie07].*

These are real-time 2D texture synthesis methods, allowing the user to interact with the designing system. While [*Owa04*] asks the user to provide a direction to orient textures inside the volume, [*Pie07*] requires the user to place the cross sections onto the 3D model and then morphing performs the cross-sections synthesis. The general principle of these two interactive methods is exemplified in *Fig. 2.15*. The 3D interactivity of the method is very appealing even for a non-expert user, but these methods have some limitations like discrepancy among different cross-sections, computational cost, morphing limitations, texture types applicability (morphing works well with isotropic, layered and oriented textures, but fails on textures containing discontinuous elements).

Similar work in this direction was accomplished by [*Tak08*] who presents a method for filling a solid object with spatially varying oriented textures with overlapping solid textures. The underlying concept is to extend the 2D texture patch-pasting approach of lapped textures [*Pra00*] to 3D solids by replacing the 2D texture and triangular mesh with a tetrahedral mesh and 3D texture patches.

2.3.3 Exemplar-based 3D texture synthesis methods

These are the most frequently employed and usually based on 2D exemplar(s). The objective is to fill up a volume with the patterns seen in the exemplar, by inventing 3D information from 2D input data.

The intense work of 3D texture synthesis based on exemplar started in the middle of the 90's with the first parametric methods attempted. It involved multi-scale statistical feature matching operations aiming to reproduce the global statistics of the 2D exemplar in the volume. 3D textures are successfully generated by matching the histogram of the volumetric data with that of the input sample at different levels of resolution [*Hee95*].

Work in this direction was extended following the idea that an image can be relevantly decomposed using a bank of spatial filters, into a set of sub-bands, each sub-band revealing

information about the presence of primitives of specific orientation and scale in the exemplar [*Por00*] [*DaC10*]. The global schema of this type of approach is presented in *Fig 2.16*. The statistical modelling of these sub-bands was extended to the 2nd order. These methods work well on homogenous and stochastic textures, but the quality of the synthetic results degrades in general for structured textures.

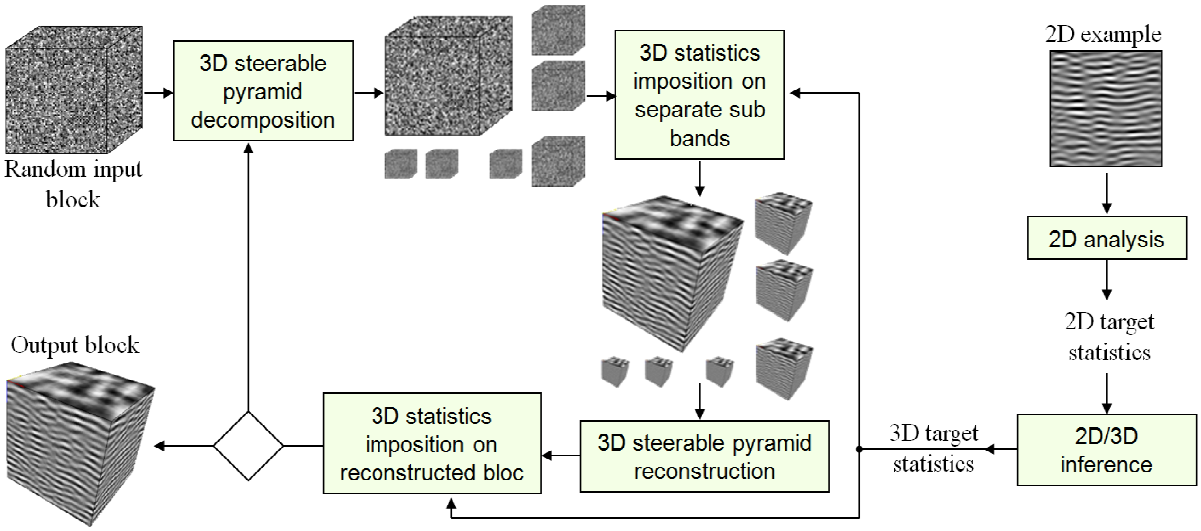


Figure 2.16 – *General schema of the 2D/3D synthesis/analysis extension of the parametric approach of [Por00], developed by [DaC10].*

The pioneering attempts to synthesize solid textures are completed with the attempts made in order to match the spectral characteristics of the 2D exemplar [*Gha95*] [*Gha96*]. The principle of the spectral analysis methods was based on using a spectral analysis (Fast Fourier Transform) of the model to obtain a basis and a noise function and then use a procedural algorithm to obtain the solid texture as in [*Per85*].

Later work combined spectrum with histogram matching and used orthogonal 2D views in order to synthesize anisotropic solid textures [*Dis98*] as it is sketched in *Fig. 2.17*. This method is designed to handle textures whose appearance are well captured by global parameters and even if only limited range of textures can be handled, it is the first approach capable of generating structural solid textures such as wood and marble.

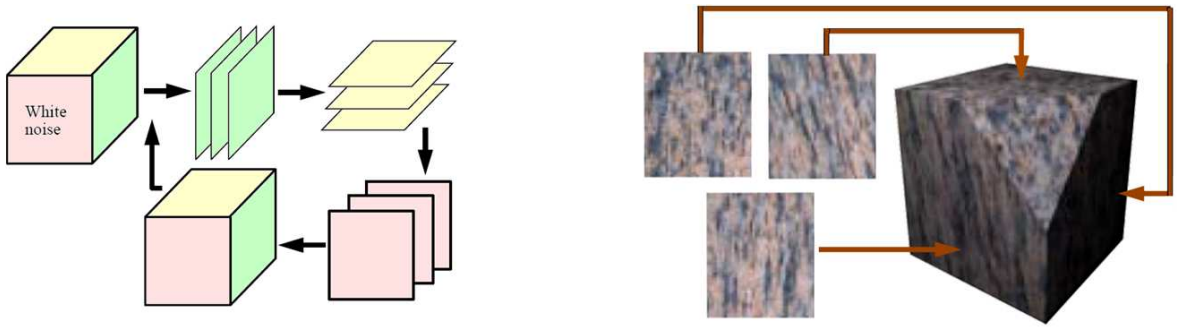


Figure 2.17 – *Solid texture generation using [Dis98]: on the left, the general principles consists in starting with a white noise and then process all the slices successively and reiterate until reaching satisfactory visual resemblance with respect to the models; on the right, an example of solid texture obtained using multiple example textures.*

To overcome the restricted applicability of textures, the non-parametric methods were later used [*Wei03*] [*Kop07*] [*Qin07*] [*Che10*]. These are essentially based upon the assumption

that textures are Markov Random Fields. The solid texture is generated systematically one voxel/patch at a time maintaining the coherence of the local texture with its vicinity. The global schema consists in copying pixels from the exemplar into the output texture, making sure that the output voxel 2D neighbourhoods – taken on up to three orthogonal 2D slices containing the output voxel – are similar to the neighbourhoods of selected input pixels [*Wei03*]. Improvements for producing realistic solid textures from 2D exemplars were brought by [*Kop07*] who combined global texture optimization [*Kwa05*] with colour histogram matching as [*Hee95*]. To upgrade even more the quality of the results, [*Che10*] proposed a high quality texture synthesis by integrating in the texture optimization two new kinds of matching histograms (position and index histograms). These approaches assume the search of the best matching neighbourhood independently in two or three directions corresponding to the orthogonal views of the solid texture. Consequently, a large number of iterations are required before the algorithm stabilizes into a satisfactory result.

In order to take into account long term dependences observed in real textures while preserving a reasonable computational complexity, most authors propose multi-scale implementation schemes, combined with specific acceleration tricks based on efficient searching algorithms: tree structured vector quantization [*Wei00*], k-coherence tree [*Ton02*], discrete solver [*Han06*]) or dimensionality reduction [*Lef06*] (the input exemplar is transformed from a 3D colour space into a high-dimensional appearance space in which each pixel contains much more information than only colour).

The non-parametric pixel by pixel synthesis methods are the most used, finding applications even in modelling real human body systems (for example for creating realistic 3D organic tissues starting from 2D textured samples [*Pri12*]).

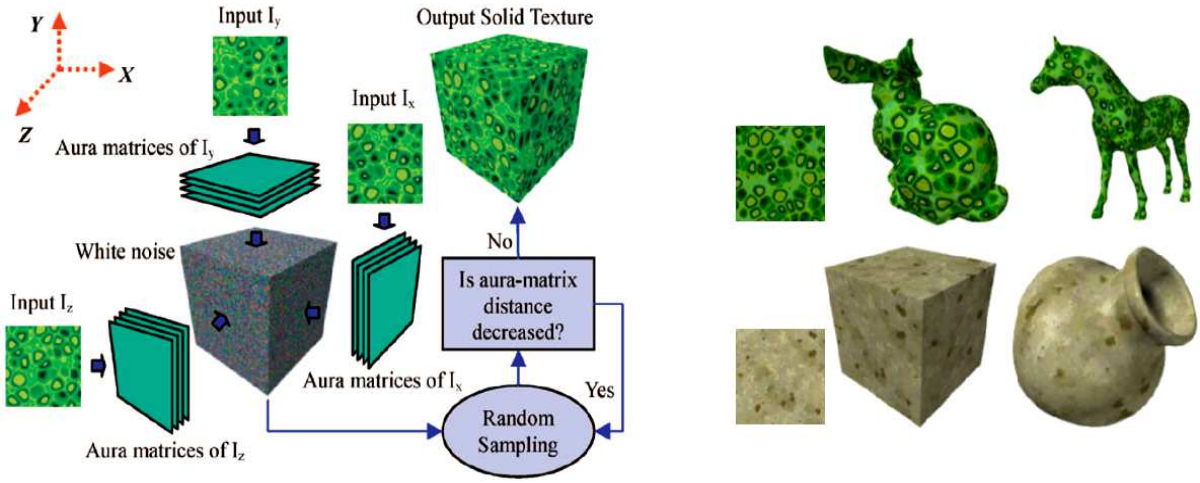


Figure 2.18 – *Aura 3D synthesis [Qin07]: on the left, the general pipeline of aura 3D sampling consisting in computing the aura matrices for each input sample and modify the initial white noise block so that the sampled final volume will have similar texture to the corresponding input sample when a cross section perpendicular to that view direction is cut from the volume.; on the right, some successful examples of synthesized textures.*

[*Qin07*] presented a new mathematical framework, for generating solid textures by sampling the Grey Level Aura Matrices of the input samples constrained in multiple view directions. It allows characterizing the co-occurrence probability distributions of grey levels at all possible displacement configurations. The algorithm and some examples are presented in *Fig 2.18*. The results are quite impressive, but when handling with colours, colour channels must be uncorrelated. However, as stated by [*Kop07*], in most textures the colour channels are strongly correlated, and independently synthesizing the uncorrelated channels leads to visual artefacts.

Ultimately, solutions based on *stereology techniques* (the study of 3D properties of a material based on 2D observations) were attempted by modelling the shape and the spatial distribution of the particles observed on 2D images of binding materials [*Dis99*] [*Jag04*].

To be able to synthesize solid texture of particles, a first proposition was to operate in the spatial domain, consisting in three steps: first – detect and extract the particle structures from the texture model, second – the extracted shape is modelled using the idea of generalized cylinders geometric model and third – to distribute the 3D reconstructed shapes inside a volume [*Dis99*].

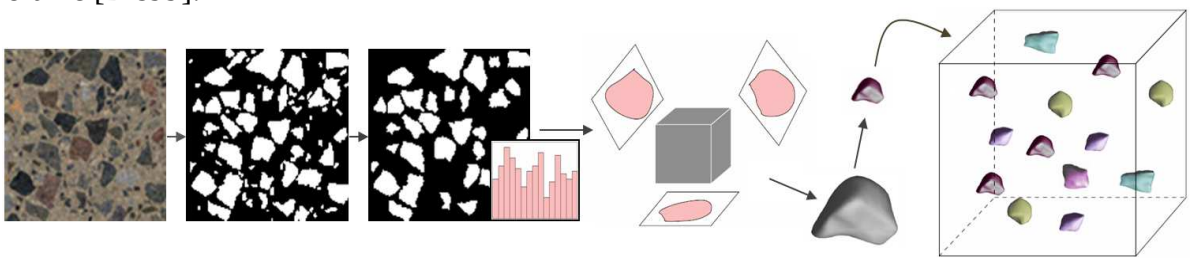


Figure 2.19 – *Overview of the solid texture synthesis of discrete particles from [Chi06]: from left to right, the input 2D image, the second one is the segmented image, the 3rd contains the filtered image to get rid of the very small particles according to the particle size histogram, the 4th depicts the synthesis of a single particle using a visual hull algorithm and, the final, scaling, colouring and spatial arranging of the particles in an iterative way until reaching a specific overall crowdedness.*

The next notable approach addressed the challenges of creating 3D solid representations of aggregate materials; it showed how to perform an accurate estimation of 3D particle distribution from a 2D image, so that later embedded particles of different shapes, sizes and colour details can be generated and arranged accordingly. This approach requires having models for the different particle shapes that may be present in the solid. A more recent method operating in the spatial domain, stereologically independent, permits to build particles solely from the given exemplar [*Chi06*] as exemplified by *Fig. 2.19*.

2.4 Conclusion

This chapter was intended to compose a comprehensive survey covering most of the existing 2D and 3D synthesis techniques. The bibliographic work led in this field shows that the state-of-art of the synthesis methods is a particularly dynamic ground, given a strong idea about the interest of the scientific community in image synthesis. The panoply of methods and means of classification can be seen as a proof of the interest of the scientific community for developing new texture synthesis techniques (lately especially towards 3D environment) and in the same time for improving their controllability and interactivity.

Example-based solid texture synthesis has many advantages over the other 3D texturing/visualisation approaches because it can generate consistent and detailed textures from examples. The drawback, however, is the cost in both computation and memory, as it explicitly computes and stores a dense 3D array of voxels covering the entire target model. Pixel-by-pixel methods allow a finer control during synthesis handling one pixel at a time.

Following this understanding, the next two chapters deal with the non-parametric pixel-per-pixel synthesis based on Markov Random Field hypothesis - fixed neighbourhood search [*Wei03*] [*Kop07*] [*Che10*] and probabilistic modelling [*Pag98*], using a single 2D textured image as input data. The drawbacks of these methods are pointed out and original solutions are proposed accordingly, regarding implementation complexity or memory cost.

Chapter 3

2D/3D extension based on neighbourhood search

Contents

3.1 Introduction 27

3.2 Non-parametric synthesis: Wei and Levoy's algorithm 28

3.2.1 Basic principle of the 2D synthesis 28

3.2.2 The 3D extension 30

3.2.3 Multi-resolution implementation 31

3.2.4 Edge handling 33

3.2.5 Computational cost and acceleration 34

3.2.5.1 Tree Structured Vector Quantization 35

3.2.5.2 Nearest-neighbour search using the binary tree 36

3.2.5.3 A concomitant acceleration option 36

3.2.6 Accelerated multi-resolution 2D\3D algorithm 37

3.2.6.1 Iterative search for the best solution 37

3.3 Robust texture optimization 39

3.4 Histogram matching 40

3.4.1 Grey level histograms 40

3.4.2 Index histogram and position histogram 41

3.4.3 Optimized texture synthesis algorithm 44

3.5 Neighbourhood systems and scan type 44

3.6 Implemented algorithms 46

3.6.1 Usage and common implementation details 46

3.7 Conclusion 47

3.1 Introduction

As presented in the previous chapter, various algorithms are capable of synthesizing textures and among them one that is easy-to-use, efficient and delivers convincing results with moderate computational cost is the algorithm of Wei and Levoy [*Wei00*]. It belongs to a class of texture synthesis algorithms characterized by the fact that they are non-parametric, build the output block in a pixel-by-pixel fashion and are based on fixed neighbourhood search.

The first algorithm studied in this chapter is a variant of the algorithm proposed by [*Wei03*] being itself a 3D extension of the original algorithm [*Wei00*], serving as the backbone method. This algorithm needs a texture sample and a noise input and all further computations are accomplished automatically.

However, to ensure higher efficiency and better quality for all possible textures, some amendments on the reference algorithm have to be made. Improvements are brought firstly by using a global optimization procedure [*Kwa05*] [*Kop07*] involving a grey-level histogram matching technique and by using an enhanced solving strategy.

To be able to ensure a high quality texture synthesis, the synthesis procedure is integrated with a preliminary analysis step that consists in creating for each input pixel a set of similar input pixels in term of neighbourhood metric, but more significant, it is integrated with two new kinds of matching histograms [*Che10*] and an improved solver. This is done with the intention of better reproducing in the output texture the diversity learned in the input sample.

As the new texture is synthesized pixel-by-pixel with the value of each new pixel determined by its local neighbourhood, a challenging topic remains the choice of the neighbourhood system and consequently the texture scan order. Contributions in this direction are brought by proposing more proficient three-dimensional scan types [*Urs12*].

As well, to assure a reasonable computational complexity, the implemented algorithms involve an efficient searching algorithm combined with a multi-resolution implementation capable of capturing textural patterns at different scales without increasing the computational load.

This chapter reassess in a unified methodical form variants of the fixed-neighbourhood search based non-parametric synthesis algorithms, implemented under a common algorithmic benchmark, intended for the qualitative and quantitative evaluation of Chapters 5 and 6.

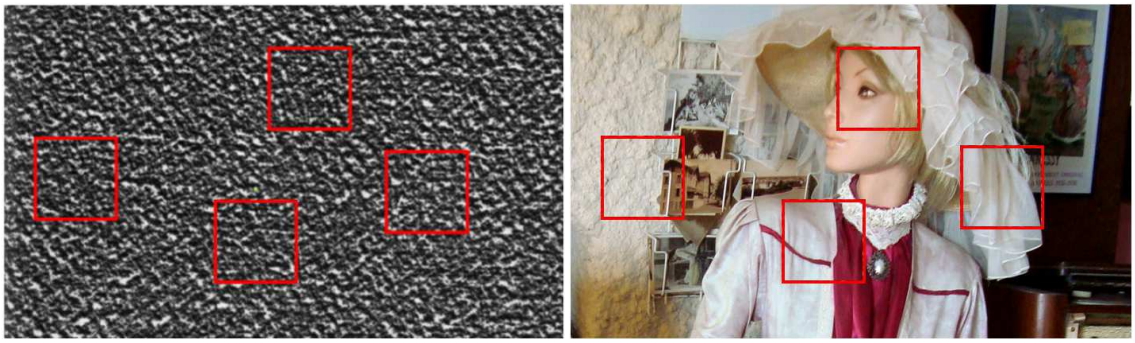
3.2 Non-parametric synthesis: Wei and Levoy's algorithm

This section addresses the 2D/3D synthesis problem by a non-parametric approach inspired by the algorithm of Wei and Levoy [*Wei00*]. The 3D extension is in fact a multi-2D adaptation of the algorithm of texture synthesis from multiple sources [*Wei03*] using only a single 2D image as source of synthesis.

In order to address the 3D problematic, the basic principles of the 2D algorithm have to be assimilated.

3.2.1 Basic principle of the 2D synthesis

The 2D synthesis algorithm proposed in [*Wei00*] relies on the assumptions of a Markov Random Field (MRF) (see *Appendix C*), modelling the texture as a realization of a local and stationary random process. As showed in *Fig 3.1* these assumptions are not suitable for the general case. This means that each pixel of the texture is considered to be characterized by a small set of neighbouring pixels, and this interpretation is the same for all the pixels within the texture.



a *b*

Figure 3.1 – *Markov Random Field assumptions illustrated on (a) the D57 Brodatz texture: each pixel is related only to a small set of pixels inside its neighbourhood (i.e. locality); different regions of proper size are always perceived to be similar (i.e. stationarity). (b) These remarks are not valid in the case of a general image, where the observed regions are very different, despite the fact that inside the image one can find separated textured areas (like the wall-stone or the cloth fabric).*

The algorithm starts from a sample texture and an image containing a uniformly distributed additive noise over the frequency domain. The synthesis itself consists in modifying the output noise to look like the input sample. The new texture is generated in a scan-line order, for each output pixel is being searched a pixel from the input texture that matches the best. The choice of the best match is based on the neighbourhood of the current pixel.

More explicitly, the algorithm captures the neighbourhood of the current pixel and searches in the input texture the most similar neighbourhood. When the matching neighbourhood is found, its corresponding pixel is copied at the current position in the output texture.

The resemblance of two neighbourhoods is based on the Euclidian distance; that is the sum of the squared differences of all the pixels in the neighbourhood:

$$dist_{euclid}\{N(p), N(q)\} = \sum_i [N(p_i) - N(q_i)]^2 \qquad (3.1)$$

where i runs through the neighbourhoods, visiting analogously every pixel p_i from the neighbourhood $N(p)$ of the pixel p and every pixel q_i from the neighbourhood of the pixel q. All the above principles are schematically illustrated in *Fig. 3.2*.

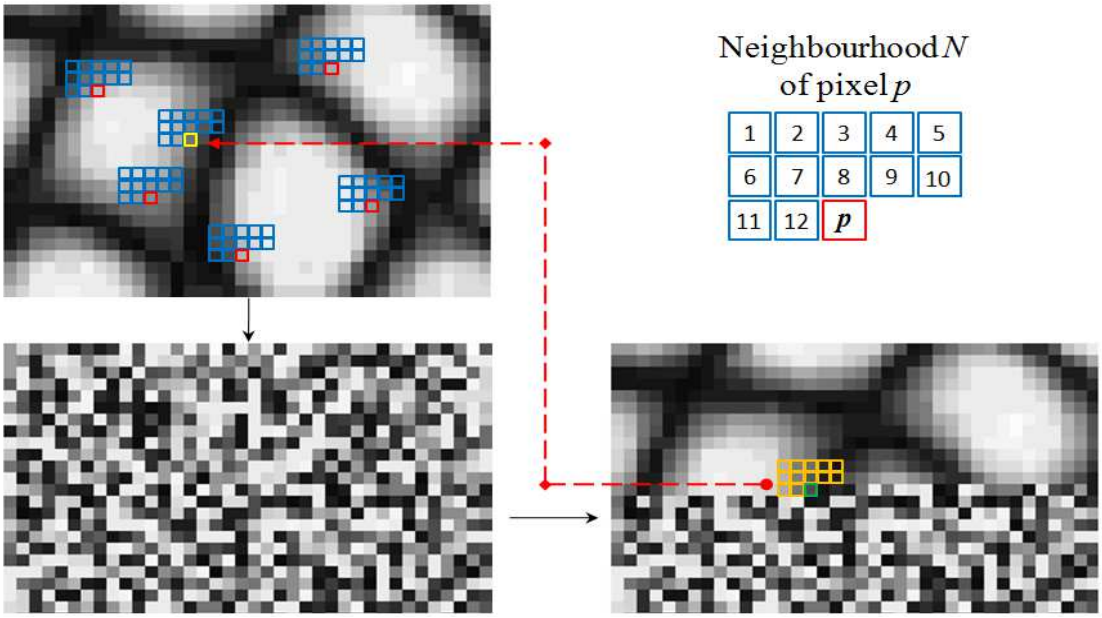


Figure 3.2 – *Schematic representation of the 2D non-parametric algorithm: starting from the input image (top-left image), an output image is initialized with white noise (bottom-left image) and every output pixel, in scan-line order, is replaced with an input pixel that has the closest neighbourhood to the current pixel neighbourhood.*

The 2D algorithm carries out the synthesis in a scan-line order consisted with a semi-causal (half-square) neighbourhood in order to use for the synthesis of a current pixel only already synthesized pixels. This allows the synthesis to take place in one step. Using a non-causal neighbourhood involves several re-iterations to reach a satisfying solution. More implementation details are to be found in the original paper of the 2D method [*Wei00*], or in this thesis, in the section dedicated to the 2D/3D extension.

The implementation of the basic 2D texture synthesis is straightforward, following the steps indicated in *Fig 3.3*.

```
Begin
    Create the output image and initialize it with noise
    For every output pixel
        Extract the neighbourhood
        Search in the input image the most similar neighbourhood
        Replace the output pixel with the pixel of the matched neighbourhood
    End for
End
```

Figure 3.3 – *The basic non-parametric 2D texture synthesis algorithm.*

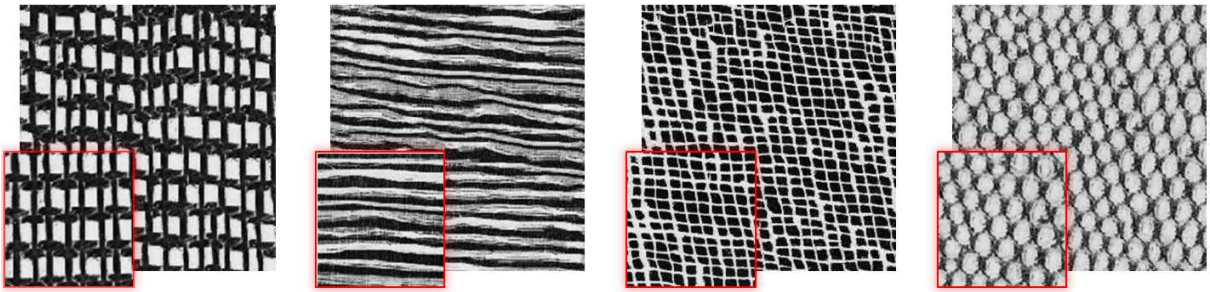


Figure 3.4 – *Example of the basic 2D texture synthesis algorithm from Fig. 3.3: for each of the four cases the synthesis framework consisted in an input image (enclosed in red) of size 100×100 pixels and using a neighbourhood of size 7 to obtain in a scan-line order the 200×200pixels textures.*

Some cases of textures obtained by using this scheme are presented for simple algorithm exemplification in *Fig. 3.4*, reminding that more details about this 2D sketched implementation can be found in the original paper [*Wei00*].

3.2.2 The 3D extension

The solid texture synthesis algorithm, like the 2D algorithm, is based on the MRF hypothesis, meaning that it relies on:

- *texture locality* - every pixel is predictable from the few pixels in its neighbourhood
- *texture stationarity* - spatial statistics invariance by translation

These assumptions are illustrated in *Fig. 3.2*, but a more complete description of MRF is to be found in *Chapter 4*. Unlike most MRF based algorithms, the following 2D/3D extension of the Wei and Levoy's algorithm is completely deterministic but no explicit probability distribution is constructed.

The 2D/3D extension is a multi-2D adaptation of the algorithm of texture synthesis from multiple sources [*Wei03*] but by using only a single 2D image as source of synthesis [*Urs10*].

The goal of the volumetric synthesis algorithm is to generate a new texture so that each local region within each orthogonal section is similar to another region from the input image. For generating a texture the process takes an example texture patch and a random noise as inputs. The noise is modified to look like the given example texture assuring a certain initiating randomness, but in the same time chosen in order to conserve the same grey level distributions as in the exemplar.

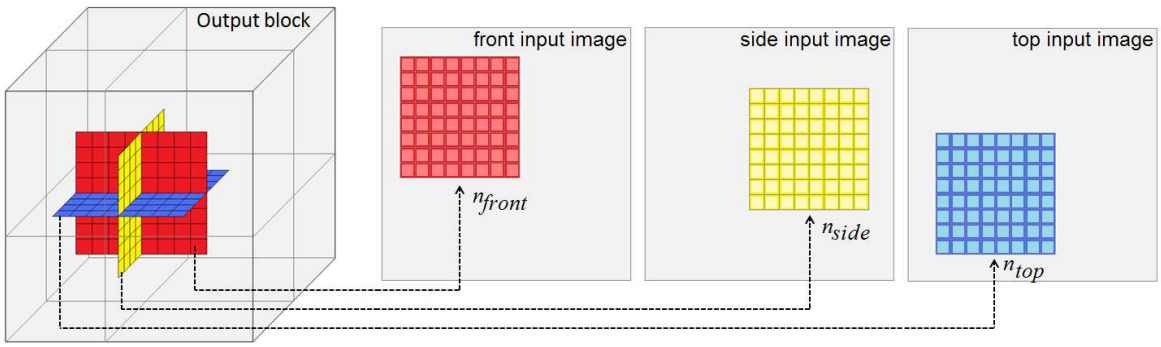


Figure 3.5 – *Principle of non-parametric synthesis: extract three neighbourhoods in the output block (front view, side view and top view), search the best three neighbourhoods in the input image, and combine their corresponding input pixel values to modify the output voxel.*

The synthesis proceeds voxel by voxel by examining the 2D neighbourhoods of the current voxel from multiple orthogonal views of the 3D block (two or three views are good enough, depending on the exemplar texture) as shown in *Fig. 3.5*.

An evident problem is that for an output voxel there are several possible solution candidates. Indeed, for each view (front, side or top view), a pixel solution p_i, i inside {*front*, *side*, *top*} is obtained by comparing the neighbourhood N_i of the current voxel in the view, with all the neighbourhoods in the input image. The value v_i of the input pixel p_i with the most similar neighbourhood is retained. All three candidates v_i, i in {*front*, *side*, *top*}, have then to be combined to get a new value v which is finally assigned to the voxel. Several combination strategies are proposed in the following sections of this chapter.

The similarity between the neighbourhoods is measured using the Euclidian distance. The same process is repeated for each output voxel until all the values are determined. Neighbourhoods crossing the output image borders are handled toroidally (see *section 3.2.4*). The synthesized texture can be of arbitrary size and tileable.

This type of 2D/3D texture synthesis extension raises different enquiries, as it is constrained by the *neighbourhood system* – in particular its size and shape, by the synthesis *scan type* and by the way of *combining information* from the orthogonal views. An important phase in the synthesis process entails an *accelerated search* of the best match for each orthogonal neighbourhood in the input image. By including a *multi-resolution scheme* to capture better the texture motives at different scales, it proves to be a well adaptable and largely applicable synthesis method.

3.2.3 Multi-resolution implementation

To assure a good texture synthesis, the size of the planar neighbourhood has to be adequately chosen so that it should be able to preserve texture structures. So, a texture containing large scale structures requires a large neighbourhood, but unfortunately the computational cost grows with the neighbourhood size.

In order to solve this problem, a multi-resolution image pyramid is used to capture the structures only by a few pixels in lower resolution pyramid levels. The number of pyramid levels has as much influence as the neighbourhood size.

There are many ways to generate and use the image pyramids (Gaussian pyramids [*Pop93*], Laplacian pyramids [*Bur83*] [*Hee95*], steerable pyramids [*Sim92*] [*Hee95*], feature-based pyramids [*DeB97*] or the simple multi-scale images obtained by decimation). We have chosen to use Gaussian pyramids. This consists essentially in two steps: first applying a low-pass filtering using convolution with a Gaussian filter kernel; and secondly, the filtering is followed by a down-sampling with a 2:1 factor.

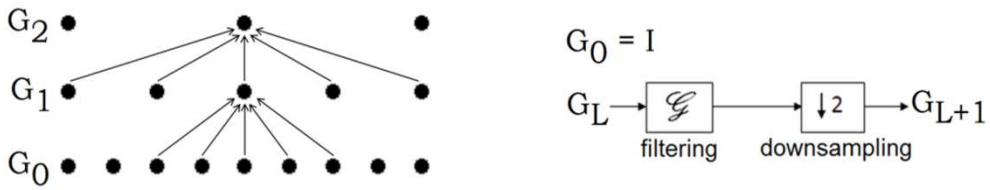


Figure 3.6 – *The principle of Gaussian pyramids of exemplar I: on the left, a simplified exemplification; on the right, the steps it consists in - low-pass filtering and under-sampling operation.*

Hence, to pass from one level to a superior one, the multi resolution algorithm filters and narrows the number of pixels to be used by the superior levels. The multi resolution is applied evenly to the input texture and to the output texture.

Synthesis takes place at every level of the output pyramid based on the pyramid of the corresponding input pyramid. The synthesis starts from the highest level, meaning the lowest resolution and finishes with the bottom of the pyramid that is in fact the searched output texture result. This kind of representation lets each higher resolution level to be constructed from the already synthesized lower resolution levels.

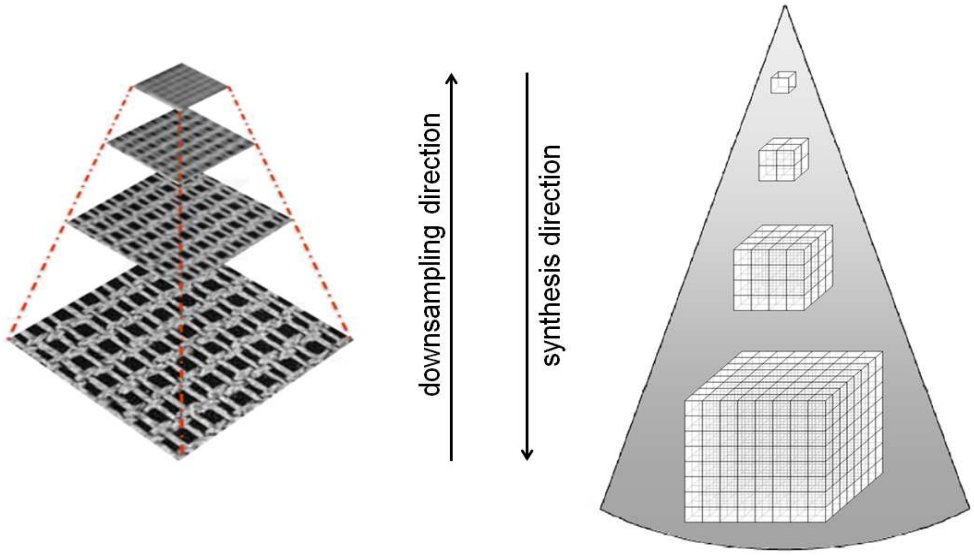


Figure 3.7 – *Multi-resolution illustration: on the left, four pyramids of a 2D textures, on the right, how the multi resolution principle looks in 3D.*

To pass the information from one treated level to a next unprocessed one, the neighbourhood used for generating a pixel contains two ingredients: the neighbourhood of the pixel from the current level and the neighbourhood of the corresponding pixel position from the previous lower resolution pyramid level. This assures that the added high frequency details will be consistent with the already synthesized low frequency structures. An example of the multi resolution neighbourhood is given just below:

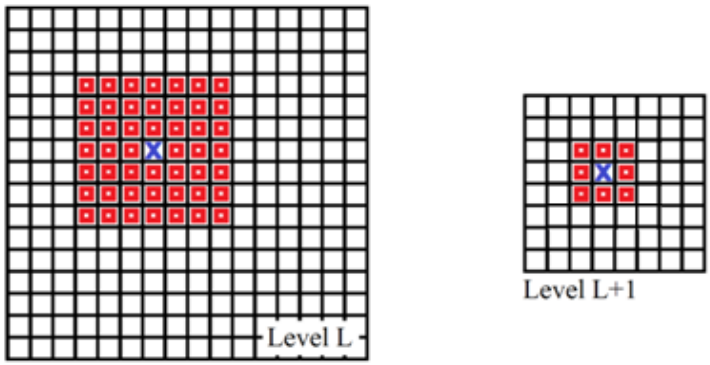


Figure 3.8 – *Example of multi-resolution non-causal neighbourhood: a neighbourhood at level L consists of a* 7×7 *neighbourhood from level L and a* 3×3 *neighbourhood from level L+1.*

To illustrate how multi resolution neighbourhood integrates in the synthesis neighbourhood search process, we address the simplified case in *Fig. 3.8* representing the last two pyramidal levels, meaning that level *L+1* is the top level. Once the top level *L+1* is synthesized using only its current level neighbourhoods the synthesis starts on level *L*. To synthesize a pixel from level *L*, having the coordinates $(x, y)^L$, the synthesis considers that the multi-resolution neighbourhood of the current pixel contains its current level neighbourhood plus the neighbourhood of its corresponding pixel $(x/2, y/2)^{L+1}$ at the previous level.

3.2.4 Edge handling

Pixels on the outer edge of the texture need a special treatment, because their neighbourhood exceeds the borders of the texture. There are two solutions: to discard the edges crossing the boundaries and perform the synthesis only inside the output texture, or to process properly the edges.

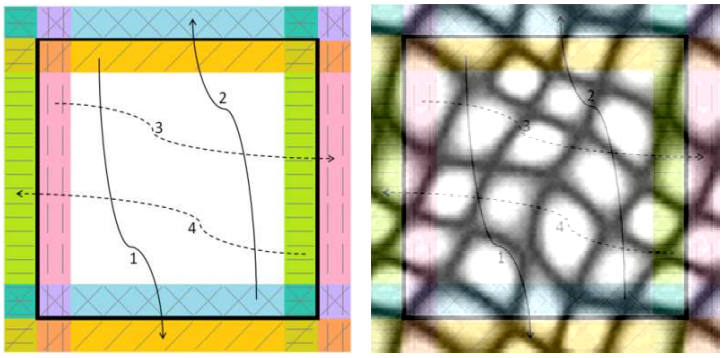


Figure 3.9 – *Example of 2D edge handling: a toroidal image is obtained by consecutively copying regions from the black bordered 2D exemplar.*

Handling the edges leads us in two directions: use a partial, incomplete neighbourhood that is available in the vicinity of the edges or, more likely, to treat the edges toroidally so that the neighbourhoods reach the opposite side of the texture delivering a seamlessly tileable output texture as exemplified in a while in *Fig. 3.11*.

The solution adopted here for synthesis is to consider the textures periodically. This involves working with a relative bigger image than the one targeted at first and in the end carving out the desired output block. This is obtained by copying certain regions from the preliminary texture in order to obtain a continuous representation. The neighbourhood used for synthesizing the upper edge pixels has to contain pixels from the bottom of the texture; the neighbourhood of left edge pixels has to contain right edge pixels and so on.

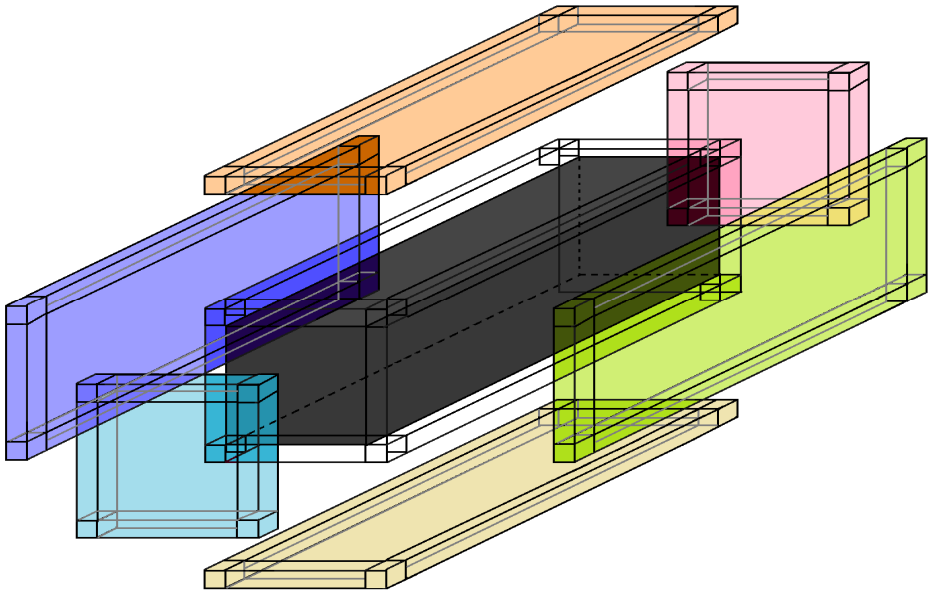


Figure 3.10 – *The principle of 3D edge handling: as in 2D, the same principle is applied for a 3D toroidal block obtained by wrapping it by all sides with regions each other taken from the opposite side of the desired output block.*

These operations are illustrated in *Fig. 3.9* and *Fig. 3.10*, showing how a toroidal texture is obtained in 2D and in 3D, assuring at any time that a pixel at position *(x,y, z)* is the same with the pixel at position *(x mod width, y mod height, z mod depth)*. In 3D, the same copying principle as in 2D is used, with the remarks that the regions are now three-dimensional covering up the output block.

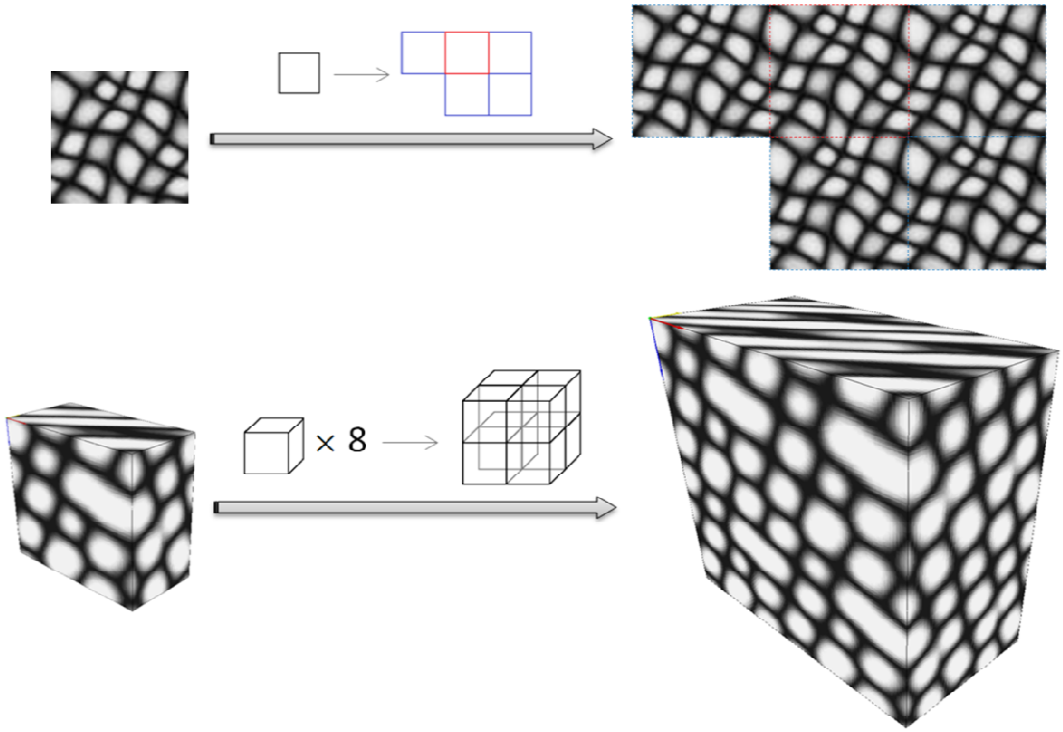


Figure 3.11 – *Examples of synthetic tileable textures in 2D and in 3D; on the left the toroidal texture obtained by synthesis and on the right the corresponding tileability of the structures in the left.*

Easier to understand the edge handling principle is by looking at the examples of tileable textures use in *Fig. 3.11*. Interesting configurations can be obtained by using tileable 2D or 3D textures, and mainly in computer graphics they are very useful in term of computational cost and efficiency when one can synthesize only a small texture and then build bigger textures only by assembling the smaller ones.

3.2.5 Computational cost and acceleration

The algorithm based on the neighbourhood search is pretty slow, performing an exhaustive search when comparing the neighbourhood of an output pixel going through synthesis with all the possible neighbourhoods in the input image. To overcome this problem, acceleration is possible. The neighbourhoods are considered to lay in a multidimensional space. Neighbourhood matching can then be seen as a nearest neighbour search problem in this multidimensional space. This speeding up is based on the study made by [*Pop93*], characterizing the neighbourhoods from a texture with a clustering probability model.

The point is to re-arrange the input neighbourhoods in a hierarchical way that eases the nearest neighbour search. The re-arrangement is done by the Tree Structure Vector Quantization (TSVQ) technique. The nearest neighbour search uses the binary search tree provided by TSVQ.

3.2.5.1 Tree Structure Vector Quantization

The set of neighbourhoods are considered as a large set of points that are divided, finally retrieving the data represented as a binary tree. This repartition diagram is based on combining two techniques – *tree-structured vector quantization* (TSVQ) and *K-means* clustering for efficient nearest point searching (Loyd's algorithm or Voronoi relaxation).

Loyd's algorithm is a particular *k*-means algorithm with *k* being 2. It is a clustering method that aims at partitioning a set of observations into 2 clusters with minimum inner inertia. The observations are seen as points in an *n*-dimensional space, where *n* represents the number of pixels in the neighbourhood. Loyd's algorithm is computed as follows:

- Initialize the two centroids responsible with the partitioning;
- Divide the space into two clusters: the first cluster contains the points that are the closest to the first centroid, and the second one the points that are the closest to the second centroid (based on Euclidian distance);
- Replace the initial centroids with the centroids of the two resulting groups;
- Re-divide the space repartition in accordance with the new centroids;
- Repeat until the two groups are stabilized.

TSVQ principle is to divide the input texture neighbourhoods into hierarchical classes that are as homogeneous and compact as possible, in which the neighbourhoods are very similar. This way we can find a value for the output pixel more rapidly by making our way through hierarchical classes towards a final input pixel having a neighbourhood similar to the output pixel one. The recursive TSVQ implementation consists in the following steps:

- Transform the input texture into the required form, meaning a vector containing several *n*-dimensional points corresponding to the neighbourhood of each input pixel;
- Compute the global centroid (the average value of all available points in the *n*-dimensional space). Use it as the root node of a binary tree;
- Initialize the root node of the binary tree; the information contained in the node is the value of the global centroid;
- Call Loyd's algorithm using as initial centroids the global centroid and a perturbed centroid (the perturbation should be small compared to the global inertia). This allows to divide the space in two homogeneous clusters;
- Initialize the left child and respectively right child of the binary tree using the two former clusters;
- Re-apply the TSVQ algorithm on each of the two child nodes.

Loyd's algorithm is called as long as there is more than one point in the input data vector and as long as the standard deviation exceeds a certain threshold. The quality of the texture increases with the lowering value of the threshold which is also an *n*-dimensional point. A value of (0, 0 ... 0) of the threshold means that a homogenous repartition of pixels in the group is envisaged. The root of the tree corresponds to the global centroid of all *n*-dimensional points in the image, and the rest of the nodes are centroids of groups of points resulted from Lloyd partition. Besides that, each node contains the average of the pixels in the corresponding group. A numeric exemplification can be found in *Appendix A*.

3.2.5.2 Nearest-neighbour search using the binary tree

The main operation involved by the proposed non-parametric synthesis algorithm consists in searching for the most similar input neighbourhood for every in progress output pixel. By rearranging all the input neighbourhoods into a TSVQ representation, the required search phase is done faster, reducing the complexity from *N* to a $log_2 N$ order as represented in *Fig 3.12*.

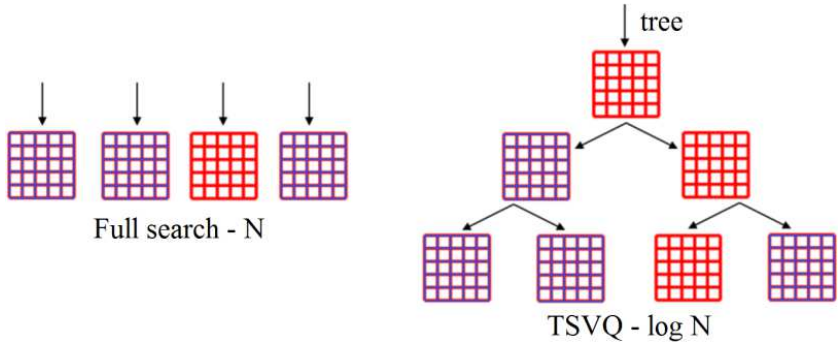


Figure 3.12 – *Tree Structured Vector Quantization (TSVQ): using the binary tree structure for efficient 'nearest point' queries reduces the time complexity N to a logarithmic order* $log_2 N$.

During the search inside the binary tree, one can stop at any moment, but the information in a non-terminal node contains only centroid information, and not exact values from the original image.

Despite the significant acceleration, the disadvantage of the TSVQ is that it is sub-optimal, the path followed in the tree susceptible of backing away from the best solution. This is because at each level in the tree, a set of vectors is ignored, set that contains perhaps the best matching neighbourhood.

The compromise made for producing faster the texture is not at all damaging, because the output textures obtained with the TSVQ and with the full search are very similar.

3.2.5.3 A concomitant acceleration option

Additionally to the above tricks, the acceleration can be made by reducing the dimensionality of the data, using a projection of neighbourhood information in a different space. This is achieved by using *principle component analysis* (PCA) [*Jol86*]. This speeding up is done by applying a PCA projection to the neighbourhood vectors from the exemplar.

PCA acts by finding the eigenvalues and the eigenvectors of the covariance matrix of all the n-dimensional points (pixels neighbourhoods) while the eigenvectors of the largest eigenvalues correspond to a subspace containing the main variations of the neighbourhoods distribution. The compression is obtained by keeping only the number of coefficients sufficient to preserve a certain ratio of the variance. For 90% of the variance, a 7x7 full square neighbourhood, meaning a vector of 49 pixels, it is reduced in the subspace to about only 10-15 dimensions, depending on the size and the richness of the exemplar, and in the same time by the correlation degree of the neighbourhoods.

However, this solution was not adopted here, because the time gained in the research phase is counterbalanced by the time required for treating the projections.

3.2.6 Accelerated multi-resolution 2D/3D algorithm

The improvements brought by the multi-resolution approach and the TSVQ based acceleration are integrated into the multi-2D extension of the basic 2D algorithm (previously illustrated in *Fig. 3.3*). This leads to an iterative synthesis algorithm, that follows the steps presented in *Fig. 3.13*.

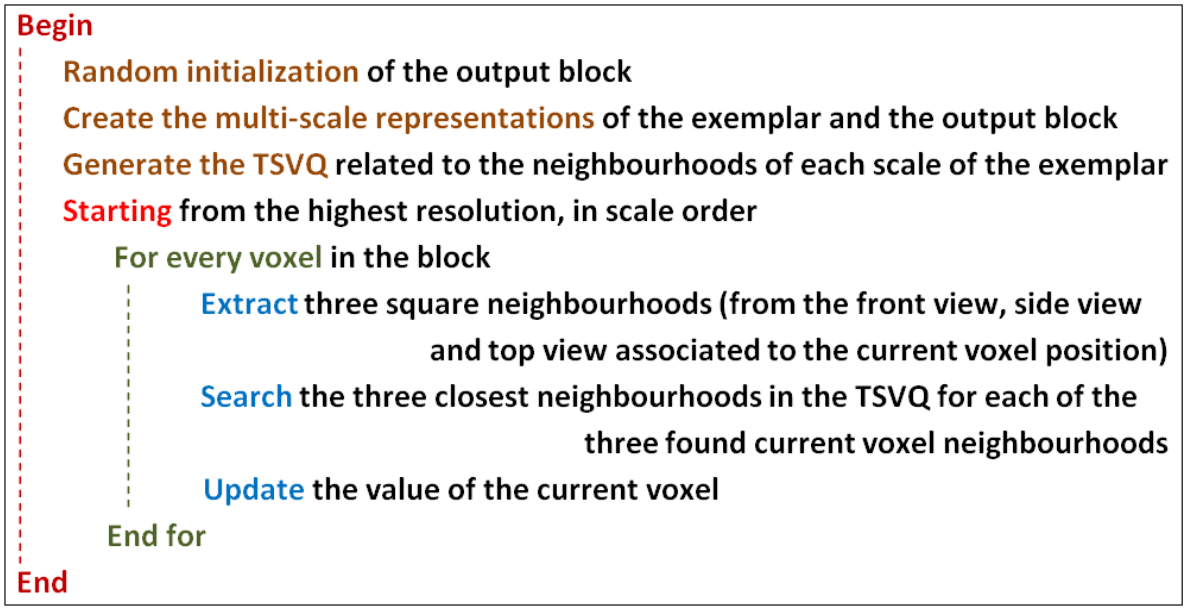


Figure 3.13 – *Schema of the 2D/3D synthesis algorithm enhanced with the multi-resolution approach and the TSVQ acceleration.*

Analyzing the above schema, the problematic consists mainly in the way of combining the several solutions according to each orthogonal view taken into consideration. In this direction, the following sections treat the way of combining the available solutions in order to obtain a better quality of the synthesis results.

3.2.6.1 Iterative search for the best solution

The first solution, proposed by [*Wei03*], consists in minimizing the energy function that measures the similitude between the volumetric texture and the input exemplar. This means that for each output voxel *v*, we have to find an input pixel p_i so that the right term of the error function equation is minimized:

$$E(v,\{p_i\}) = \sum_i \left(\|v - p_i\|^2 + \|N(v) - N(p_i)\|^2 \right) \tag{3.2}$$

where *i* runs through the input exemplar, and *N()* is the 2D square neighbourhood corresponding to the input matching pixel or the output voxel. The energy of a voxel neighbourhood is the Euclidian distance to the closest neighbourhood in the input exemplar.

The neighbourhood of the output voxel is treated as three separate entities, three 2D orthogonal voxel neighbourhoods.

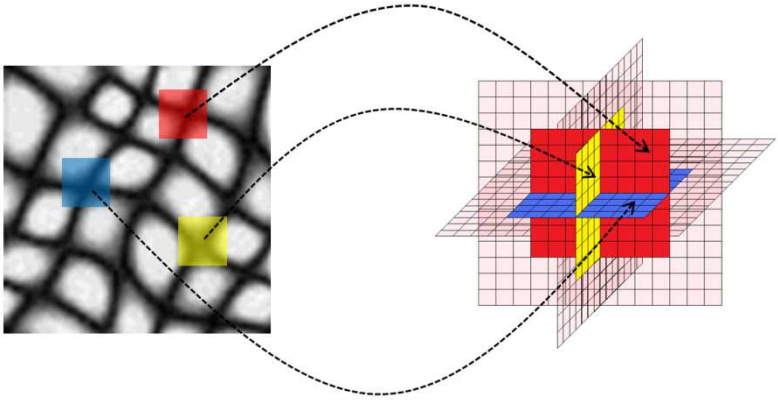


Figure 3.14 – *The 3D neighbourhood seen as three separated 2D neighbourhoods and the three closest neighbourhoods from the input image found for each view.*

Algorithmically, the direct minimization procedure of *eq. 3.2* is done iteratively, by first choosing the input pixel p_i so that the Euclidian distance between the neighbourhoods $\|N(v) - N(p_i)\|^2$ is minimal. Three input values are found consequent for each of the three orthogonal views taken into consideration (the front view p_{front}, the side view p_{side} and the top view p_{top}). The output voxel v is updated with the average of the three found values:

$$v = (p_{front} + p_{side} + p_{top})/3 \tag{3.3}$$

Reiteration continues with the current values and because a value v of the output voxel was computed, the novel term to be minimized is going to be $\|v - p_i\|^2 + \|N(v) - N(p_i)\|^2$. The iterative process stops when the energy remains the same after two consecutive iterations, denoting that the same neighbourhoods are reached after two consecutive iterations. Experiments showed that this inner process takes about four iterations until it stabilizes itself.

The above iterative method over-increases the computation and in addition, the average combination used in *eq. 3.3* is not the best way out for the resulting texture grey levels and indirectly fails to conserve the global structure of the texture [*Urs11*]. Simple averaging leads to a loss of dynamics between the original and the synthesized texture. The winning strategy is to minimize properly the energy function. This involves an optimization procedure that replaces the average with a better combination [*Kwa05*] [*Kop07*] and, in the same time, adding a colour histogram matching mechanism [*Hee95*] [*Kop07*] [*Che10*] in the texture optimization procedure as explained in the following sections.

3.3 Robust texture optimization

A direct thought for upgrading the scheme consists in giving weights to each of the three found voxels and to mix them properly, to be able to update the output voxel in only one move, dropping so the early recurrent process. It follows the idea of minimizing the global texture energy, but now using a robust energy function [*Kop07*].

This is done by replacing the squared term $\|N(v)-N(p_i)\|^2$ with $\|N(v)-N(p_i)\|^r$, where the value of r is carefully chosen so that $r < 2$. A value of *0.8* as indicated by [*Kwa05*] for the exponent r allows the optimization to be more robust against outliers. Iteratively reweighted least squares (IRLS) [*Col80*] is used to minimize the energy function by an iterative method in which a weighed least squares problem is solved at each step. To be able to apply IRLS, the energy function term is rewritten as following:

$$\|N(v)-N(p)\|^r = \|N(v)-N(p)\|^{r-2} \cdot \|N(v)-N(p)\|^2 = w \cdot \|N(v)-N(p)\|^2 \qquad (3.4)$$

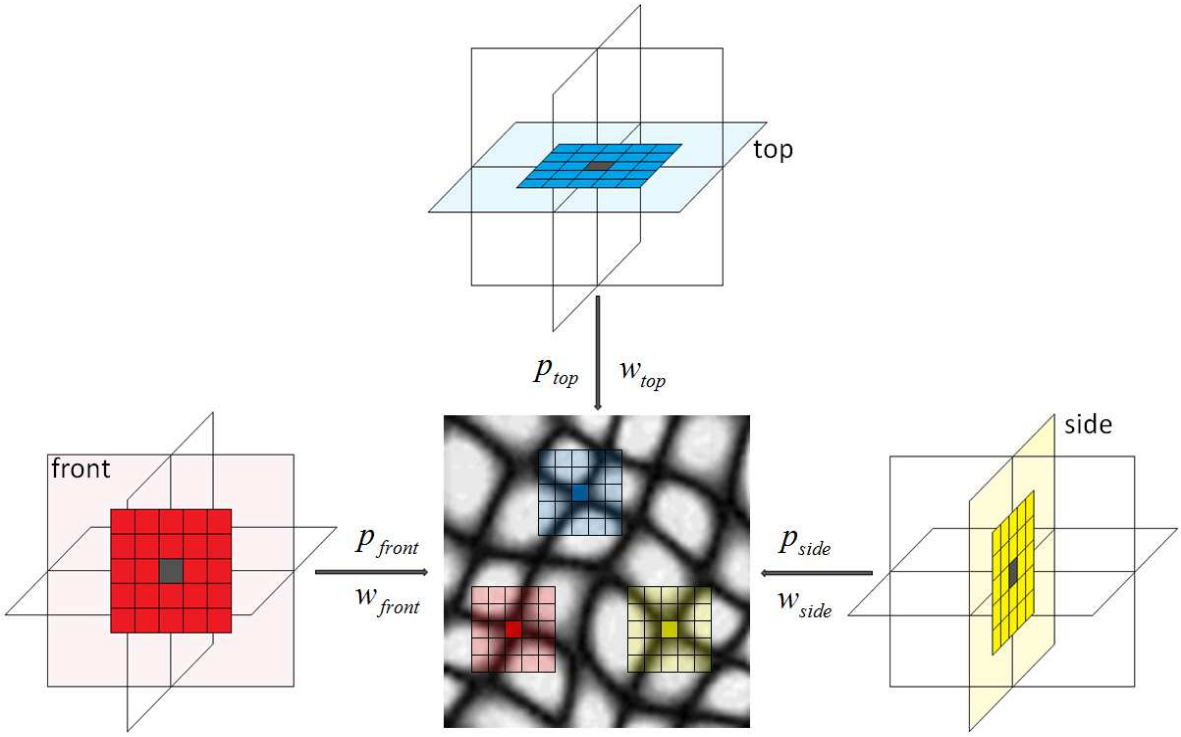


Figure 3.15 – *The solutions found for each orthogonal neighbourhood for a current voxel, and each one's corresponding weight used in the equations.*

Minimizing the quadratic energy function by deriving it with respect to *N(v)* gives the following robust optimization solution for the output voxel [*Kop07*]:

$$v = \frac{w_{front} \cdot p_{front} + w_{side} \cdot p_{side} + w_{top} \cdot p_{top}}{w_{front} + w_{side} + w_{top}} \qquad (3.5)$$

As illustrated by *Fig. 3.15*, the output voxel is interpreted as being the weighted average of the solutions found from the neighbourhoods of the orthogonal views, each solution having in the result its proper influence in the form of a weight that for $r = 0.8$ is computed as:

$$\frac{1}{w_p} = \left\{\min(dist\,[N(p), N(v)])\right\}^{1.2} \tag{3.6}$$

In the synthesis progression this optimization is retrieved as a two step process:

- *the search phase*: extract the neighbourhood of the current voxel for each orthogonal view taken into consideration; for each view, search the matching pixel in the exemplar texture and compute the weight of each solution as a distance function between the current voxel neighbourhood and the neighbourhood of the matching input pixel

- *the optimization phase*: having the orthogonal views solutions and its weights, it remains to calculate the weighted average in order to obtain the update value for the output voxel according to *eq. 3.5*

3.4 Histogram matching

Using no more than the optimization procedure presented in the previous section, texture synthesis can lead to strong locally dependent results because the synthesis relies only on local neighbourhoods when handling the energy function.

To ease the synthesis dependence on local decisions, a re-weighted scheme is required with the purpose of preserving texture global statistics. This is done by incorporating in the synthesis process a global histogram matching technique.

3.4.1 Grey level histograms

The *grey-level histogram matching* [*Kop07*] works by adjusting the weight of each voxel engaged in the weighted average that could bring differences between the grey-level histogram of the 3D result and the grey-level histogram of the 2D exemplar. Mathematically, this is expressed as following:

$$w^* = \frac{w}{1 + \max[0, H_{slice}(v) - H_{input}(p)]} \tag{3.7}$$

The re-weighting scheme can be seen as a correctional procedure by avoiding bad pixels contribution to the synthesized value: if the histogram value $H_{slice}(v)$ in the current slice of a certain grey-level v is very far from the histogram value $H_{input}(p)$ of the corresponding grey-level p in the exemplar, the weight of the voxel is reduced in order to decrease the difference between the two histograms; if the difference between the histograms is very small, the weights remain roughly the same, the weighted average influence is of no consequence leaving the synthesis as before.

Histogram adjusting is done automatically, but in order to be up to-date with the growing output texture, histogram must be updated at the same time with the voxel revising, assuring the convergence of the synthesis algorithm.

The quality of the results delivered with this technique exceeds that of many other methods. Despite this, computing the results by using the weighted average may produce blurry results if the variance of the exemplar voxels is too large [*Kop07*] and like most of the

non-parametric methods, the results are affected more or less by blurring, missing textural patterns or mismatched input/output histograms.

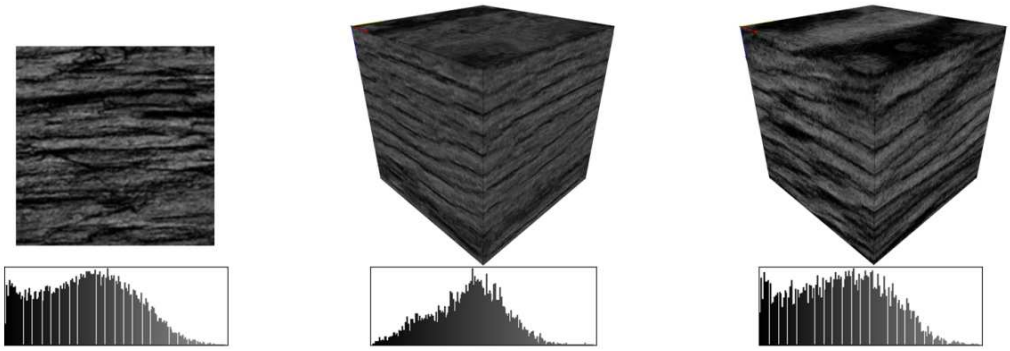


Figure 3.16 – *The influence of the histogram adjustment mechanism: from left to right, an input image of sedimentary rock with the corresponding grey-level histogram, a synthetic block obtained by using the iterative search for the best solution with its corresponding histogram, and on the right a block obtained by using the synthesis enhanced with histogram matching showing a grey levels distribution similar to the one of input texture.*

3.4.2 Index histogram and position histogram

To improve the just above mentioned drawbacks, a new *modus operandi* for texture optimization approaches is proposed allowing a high quality texture synthesis [*Che10*]. It consists in using two new kinds of histograms – *position histogram* and *index histograms* and in updating the output voxel using the *discrete solver* as [*Han06*].

This approach requires a prior synthesis part, when for every pixel in the exemplar a number of *k* best matches are retrieved in the exemplar, idea inspired from the natural texture synthesis algorithm [*Ash01*]. The discrimination is made by using the neighbourhood search based on the Euclidian distance. The outcome is a set of candidates, a *k*-coherence similarity-cluster for each input pixel. Once these items being constructed, the synthesis process can start.

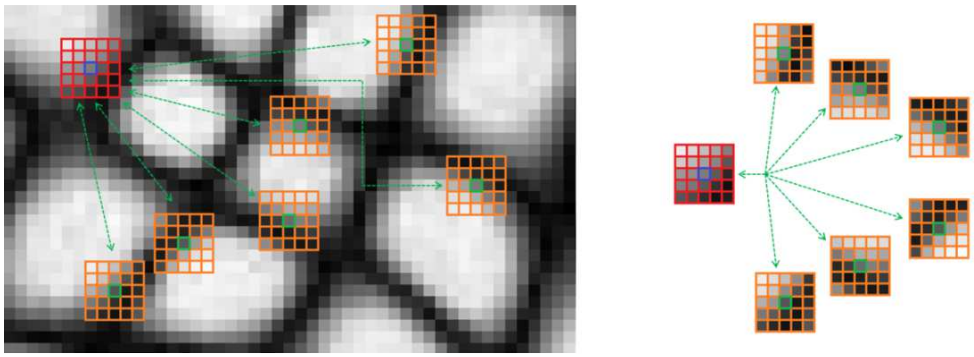


Figure 3.17 – *Example of a set of seven candidates; in a pre-processing step, for every input pixel the candidates set is built, the closest k neighbourhoods in term of grey-level L2 norm; each pixel is considered to belong to his own set.*

Unlike the causal k-coherence search in the context of synthesizing a texture using pixels already processed for synthesis [*Ash01*] [*Ton02*], a simplified version is used with the purpose of having a larger number of possible candidates for a voxel being synthesized. So to assure a large diversity, means to use a sufficient large number of candidates. The

inconvenience is that with the growth of *k*, the computation cost increases as well. An example of a set of seven possible candidates is shown in *Fig. 3.17*.

However, once the set of candidates is build, the synthesis proceeds as in the previous section. The solid texture is supposed to be similar to the exemplar on arbitrary slices through the volume, so for a voxel inside the volume, the algorithm provides a solution for each slice, for each orthogonal view. When a solution is found, instead of taking it directly in the weighted average, it is confronted with its set of candidates and implicitly with the number of times it was chosen during synthesis. Here intervenes the ***index histogram***. Index histogram counts the frequency of the voxels candidates in the volume texture. During the search phase, the choice of pixels is modulated according to their index histogram frequency. The influence of an index histogram bin is reflected as a weight factor that will assure that only the best candidate from a set is used further on.

More specifically, when searching the matching solution of the current voxel *v* in the TSVQ tree of the input, a pixel *p* from the exemplar is retrieved. In the pre-processing phase the pixel *p* has established a set of *k* candidates. The best solution, in term of uniform distribution, to be used further on, is taken out from the set of candidates being the one that has the most similar neighbourhood to the neighbourhood of current voxel *v* in term of the weighted Euclidian distance:

$$dist_{index} = w_{index,i} \cdot dist_{euclid}\,(N(p_i), N(v)) \tag{3.8}$$

$$w_{index,i} = 1 + \max[0, H_{index}(p_i) - \phi], \quad i = 0 \ldots k \tag{3.9}$$

where ϕ is the histogram value when all the indices completely equiprobably distribute in the exemplar. Having all the candidates uniformly distributed means having a well preserved texture. Once a candidate was chosen, its bin is increased with one in the index histogram.

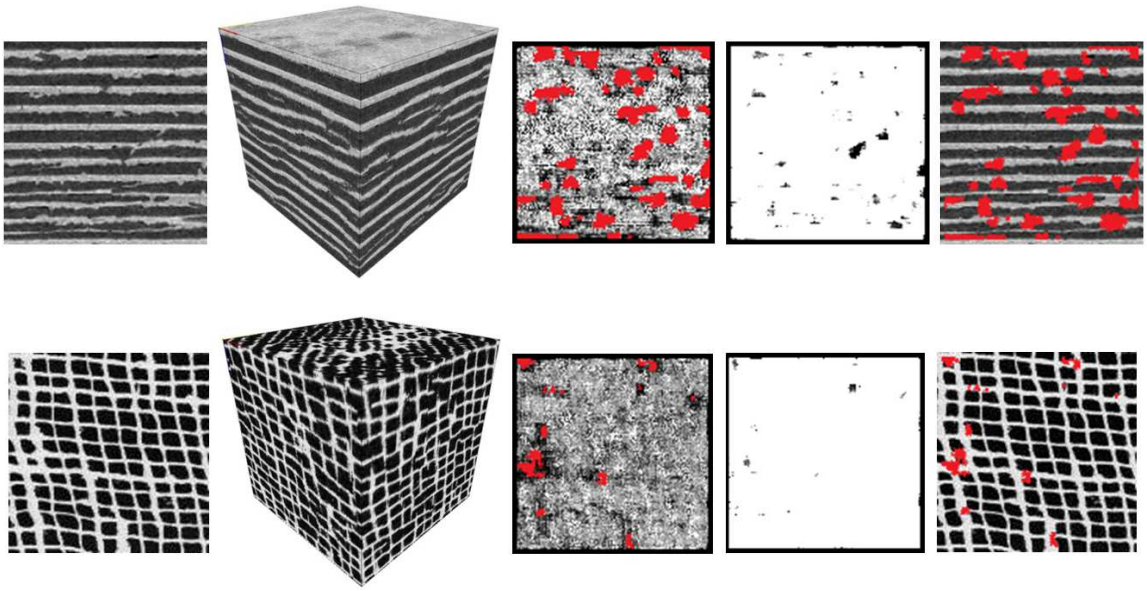


Figure 3.18 – *Two examples of index and position histogram adjustment usage: each of the two lines contains, from left to right, first the input image and the output block, and next the position histogram map, the index histogram map and the input image affected by the position histogram usage. The areas in red represent the input image pixels that are not retrieved in the output block (the pixels that are never used in the synthesis process).*

In the optimization phase, instead of colour histogram adjustment, the ***position histogram*** is used to constrain the weights computed as [*Kop07*]. The position histogram records the number of occurrences in the solid texture of the corresponding pixel in the 2D exemplar. When a solution for each orthogonal representation is found, for each one a weight is computed and then adjusted by using the position histogram:

$$w_{position} = \frac{w}{1 + \max[0, H_{position}(v) - \theta]} \tag{3.10}$$

where θ is the histogram value when all the pixels from the exemplar completely evenly appear in the result. The weights are adjusted to get a histogram as uniform as possible. Once the position histogram matching is achieved, the colour histogram matching is also achieved.

Until this point, for a voxel *v*, the solutions corresponding to each slice, filtered from the set of candidates (p_{front}, p_{side} and p_{top}) and their corresponding position histogram corrected weights ($w_{position\_front}$, $w_{position\_side}$ and $w_{position\_top}$) are known. Next, a prospective update value v^* is computed as a weighted average:

$$v^* = \frac{w_{position\_front} \cdot p_{front} + w_{position\_side} \cdot p_{side} + w_{position\_top} \cdot p_{top}}{w_{position\_front} + w_{position\_side} + w_{position\_top}} \tag{3.11}$$

The value used for updating the current voxel *v* is chosen according to the *discrete solver*, by selecting the pixel the most similar to the prospective value:

$$v = \min(\left| p_{front} - v^* \right|, \left| p_{side} - v^* \right|, \left| p_{top} - v^* \right|) \tag{3.12}$$

When the update value is found, its bin is increased with one in the position histogram. The blurring problem is avoided, since for each voxel, the grey level value comes directly from the input exemplar. More results and comments are to be found in the chapter dedicated to the experimental results.

3.4.3 Optimized texture synthesis algorithm

The global schema of the texture synthesis integrating the index histogram and the position histogram alongside the discrete solver takes the following description:

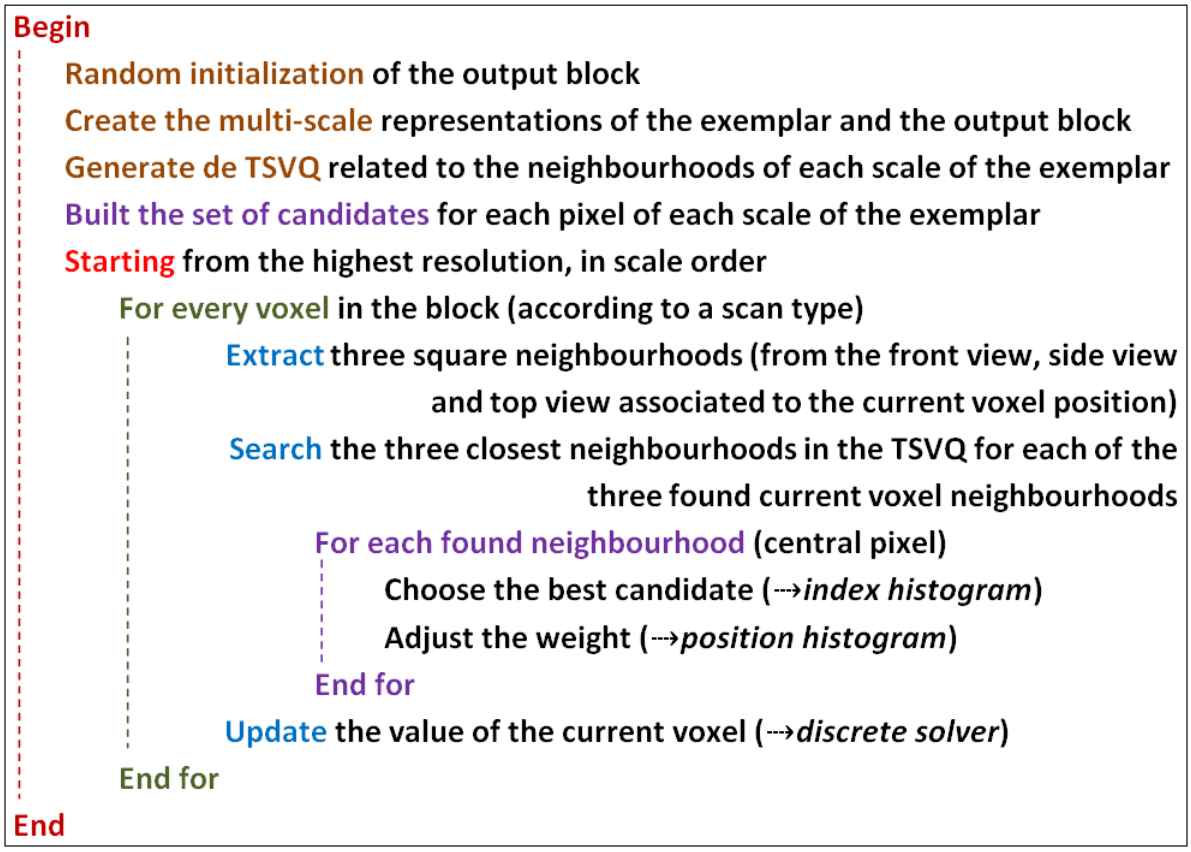


Figure 3.19 – *Schema of the 2D/3D synthesis algorithm integrated with the set of candidates, the two new kinds of histogram (index and position) and the discrete solver.*

3.5 Neighbourhood systems and scan type

Whatever the synthesis method, a problematic issue is the choice of the neighbourhood system. The size and the shape of the neighbourhoods are very important because the set of local neighbourhoods is used as the primary model for textures, directly influencing the quality of the synthesized results.

The size of the neighbourhood is a factor that specifies how stochastic the user perceives this texture [*Efr99*]: if the texture is presumed to be mainly regular at high spatial frequencies and mainly stochastic at low spatial frequencies, the neighbourhood size should be on the scale of the biggest regular texture pattern, otherwise the structure may be lost.

The choice of a causal neighbourhood makes the synthesis of a pixel totally dependent on previous pixels and leads in the case of regular/lamellar/structured textures to a higher degree of regularity of the synthesized results [*Urs12*]. The choice of a non-causal neighbourhood can partly overcome this determinism. A direct corollary is the choice of an adequate scan type.

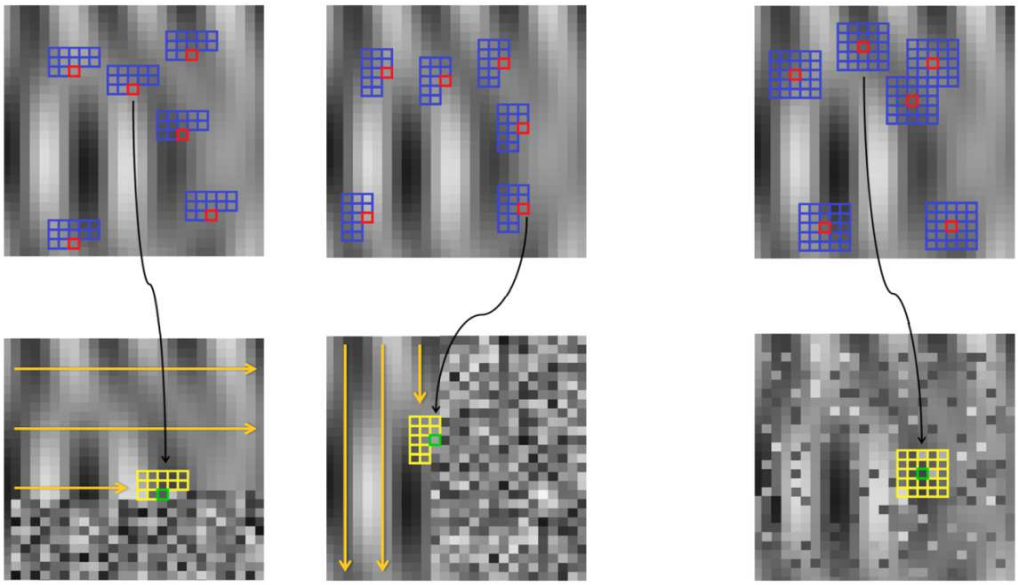
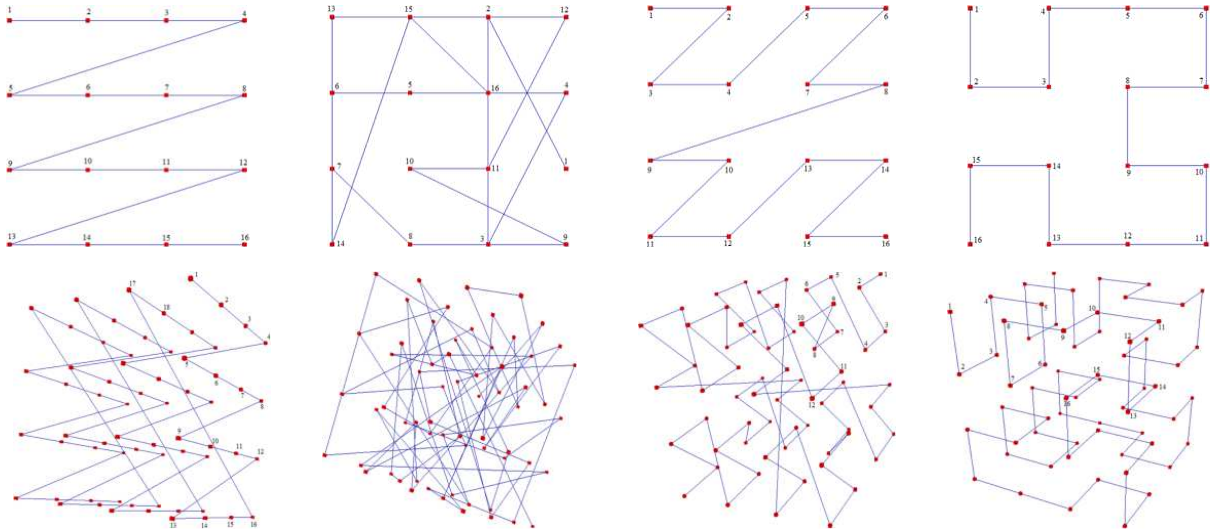


Figure 3.20 – *Neighbourhood system: on the left, two examples of L-shaped causal neighbourhood adapted to the lexicographical path; on the right, one example of square non-causal neighbourhood used with a random walk. The straight yellow arrows indicate the output pixels visiting order during synthesis and the curved black arrows indicate the neighbourhood search in the input image and the choice of the best candidate used to update the voxel in the output texture.*

Replacing the lexicographical scan type with a completely random walk allows the synthesis of a pixel by freeing itself from its past and so multiplying the possible configurations. However, the convergence time can become prohibitive in this case.

Contribution in this direction consists in reconciling the deterministic part and the randomness character by proposing an alternative scan type, namely the use of space filling curves [*Sag94*] [*Che04*] extended to three dimensions, like the Morton code (a.k.a. Z-curve or Lebesgue curve) and the fractal curves (e.g. Hilbert curve).

More information about the space filling curves is to be found in *Appendix B*.



a *b* *c* *d*

Figure 3.21 – *Illustration of different scanning types for 4x4 2D images and 4x4x4 3D blocks, in 2D (the top column) and their corresponding extensions to 3D (the bottom column): from left to right, a – lexicographic path, b - random walk, c - Z-curve and d - Hilbert curve.*

3.6 Implemented algorithms

The above considerations show the multitude of possibilities, still letting space for other reflections. There are different factors to be taken into consideration when dealing with this type of approaches. So, a general benchmark has to be defined in order to set the limits of all the implementations, based on the illustrated algorithmic facts.

3.6.1 Usage and common implementation details

Before passing to the evaluation, the algorithms designation has to be specified. It consists in stipulating the algorithms usage and the key factors specific for every employed algorithm.

The first developed algorithm follows the format from *Fig. 3.13*. To be exact it corresponds to the non-parametric 2D/3D synthesis algorithm of Wei and Levoy [*Wei03*] treated as a multi-resolution implementation and accelerating the neighbourhood search by clustering the input pixels neighbourhoods in a TSVQ form. The main specificity of this approach is that the value of an output voxel is obtained by combining the three solutions from each orthogonal view as in *eq. 3.3*, that is the average operation of the available solutions and repeating the same procedure until reaching a stable result. This approach is going to represent the reference method which is going to be subsequently enhanced. This is going to be named *NP_WL*.

The second algorithm employed, uses the backbone of the *NP_WL* method, but modifications emerge in the way of combining the solutions from the orthogonal views. A robust optimization solution for the output voxel is employed, where each pixel participating in the weighted average of *eq. 3.5* is chosen in order to assure a global grey-level histogram matching between the exemplar and the targeted output block. Because it is a mixture of the earlier *NP_WL* algorithm and the optimization proposed by Kopf et *al.* [*Kop07*], the notation conferred to this non-parametric version is *NP_K*.

The third handled tag, *NP_CW*, corresponds to the same *NP_WL* multi-resolution, TSVQ accelerated backbone algorithm, enhanced with two new kinds of matching histograms (index and position histograms) as in *section 3.4.2* in order to obtain a higher quality of the synthesis results as intended by Chen and Wang [*Che10*]. These particular histogram adjustments manage to assure on the output the same grey-levels as the exemplar, and seek to control the uniform distribution of the input pixels in the synthesis result. This closing non-parametric algorithm respects the synthesis steps suggested in *Fig. 3.19*.

The common influential factors for all versions are the size of the input exemplar, the size of the output block, the size and the shape of the pixels neighbourhood, the number of scales and the pixels visiting order, while for *NP_CW* the size of the set of candidates is also taken into consideration.

3.7 Conclusion

This chapter aimed at providing a comprehensive deployment of the fixed-neighbourhood non-parametric implemented algorithms. Contributory work was lead in creating a common algorithmic benchmark easing the experimental study pursued in *chapters 5* and *6*.

It points out the specificities and drawbacks of each of the above versions proposing solutions to improve the 2D/3D extension of the texture synthesis algorithm.

It starts with introducing the synthesis concepts in the 2D environment and then showing possible 3D extensions. Multi-scale and acceleration tricks are proposed together with our original diagram for visiting the pixels during synthesis.

The main issue, the way of combining the available orthogonal solutions, is sorted out by implementing mainly three strategies (average, weighted average, discrete solver) to improve the quality of the results. Following this plan, notations are given to the developed algorithms - NP_WL, NP_K and NP_CW.

An inventory of the implemented algorithms which are going through our study are listed in the table below, showing next to each variant the way of combining the information from the orthogonal views in order to obtain the update value for the output voxel.

Synthesis method	Strategy for output voxel
NP_WL	$v = (p_{front} + p_{side} + p_{top})/3$
NP_K	$v = \frac{w_{front} \cdot p_{front} + w_{side} \cdot p_{side} + w_{top} \cdot p_{top}}{w_{front} + w_{side} + w_{top}}$
NP_CW	$v = \min(\lvert p_{front} - v^* \rvert, \lvert p_{side} - v^* \rvert, \lvert p_{top} - v^* \rvert)$ $v^* = \frac{w_{position\_front} \cdot p_{front} + w_{position\_side} \cdot p_{side} + w_{position\_top} \cdot p_{top}}{w_{position\_front} + w_{position\_side} + w_{position\_top}}$

Table 3.1 – *Listing the neighbourhood-search based methods and the way of combining the information from the orthogonal views to get the output value.*

However, the possibilities don't close down here, numerous intersections between the approaches remaining feasible (i.e. the *NP_WL* algorithm integrated with a grey-level histogram matching and a discrete solver, or *NP_CW* without the discrete solver or *NP_K* with an input set of candidates etc.).

Chapter 4

2D/3D extension based on the likelihood of neighbourhoods

Contents

4.1 Introduction 51

4.2 Non-parametric Markov Random Field model 51

4.2.1 General Markov Random Field model: concept and theory 51

4.2.2 Local Conditional Probability Density Function 53

4.2.3 Texture synthesis via relaxation 55

4.2.4 Multi-scale implementation 55

4.2.5 Pixel temperature function 56

4.2.6 2D texture synthesis algorithm 57

4.3 3D extension of the NP-MRF synthesis algorithm 59

4.3.1 General considerations 59

4.3.2 2D/3D generic algorithm 59

4.3.3 Updating the voxel value: why a full-3D process is not relevant ?..... 60

4.3.4 Heuristic approaches 61

4.3.4.1 Breaking down the 3D neighbourhood 61

4.3.4.2 Heuristic 1: *NP_ML_H1* 61

4.3.4.3 Heuristic 2: *NP_ML_H2* 62

4.3.5 Implementation details 63

4.3.5.1 Initialisation of the output block 63

4.3.5.2 Scales handling..... 64

4.4 Conclusion 66

4.1 Introduction

This chapter deals with a new type of approach, doing explicitly what the methods from *Chapter 3* tried to accomplish indirectly. More precisely, the previous methods (NP_WL, NP_K and NP_CW) copy pixels from the input exemplar directly/indirectly in the output block, trying to reach a structural similarity. They are based on strict neighbourhood search and intensity histogram matching; additionally, extra matching histograms are proposed in order to distribute uniformly the input pixels in the target texture.

The next method sticks to the idea of having the input/output configurations identically distributed, the statistical decisions being based on the conditioned neighbourhood concept. It is based on a more advanced non-parametric Markov Random Field synthesis principle (NP-MRF) than the previously developed methods. To reach a high likelihood quality, the synthesis should be able to distribute the input pixels and neighbourhood configurations in the output texture while preserving the characteristics of the input texture.

A technique that enables 2D synthesis of textures visually indistinguishable from their models, is the multi-scale NP-MRF texture modelling method proposed in [*Pag98*]. This method mathematically captures the visual characteristics of a texture into a unique statistical model of that texture [*Bes86*] that describes the interactions between pixel values [*Hai91*]. Such an assertion has been proved by using this model to synthesize the texture and by judging the similarity between the synthetic texture and the input texture. At each position in the output texture, the algorithm provides the most probable grey level to be in that position considering the grey levels in neighbouring positions.

The following sections deal first with some indispensable MRF theory and terminology used to describe the non-parametric synthesis process, and continue with the relevant three-dimensional operating method extending the 2D approach method used by [*Pag98*]. The synthesis uses a multi-scale relaxation algorithm, integrating the concept of pixel temperature that serves as degree of confidence relative to a pixel value.

4.2 Non-parametric Markov Random Field model

Markov Random Fields (MRFs) have been studied extensively for solving image analysis problems at all levels. Their foundations have been well established from the 80's mentioning the remarkable works of [*Bes74*] [*Bes86*] [*Gem84*] [*Cli90*] [*Gem91*] [*LiS01*]. Some of the most important applications include image restoration and segmentation, edge detection, texture analysis and also texture synthesis. All these are possible thanks to the MRF ability to model the local dependencies of image pixels.

4.2.1 General Markov Random Field model: concept and theory

The MRF principle can be seen as a sites-and-labels concept [*LiS01*], meaning that the solution to a problem is a set of labels assigned to a set of sites. A label is an event that may happen to a site and the labelling operation is to assign a label to each of the sites.

Translating this into MRF image modelling, one retrieves the concept that an image with discrete grey levels, of size $w \times h$ corresponds to a lattice $S = \{s_1, s_2, ..., s_{w \times h}\}$ (i.e. set of sites), so that each pixel in the image is a site *s* on the related lattice and its grey level is a

value x_s (i.e. label) contained in the finite state space $\Lambda = \{0, 1, 2,\ldots, L\text{-}1\}$, where L is the number of grey levels in the image.

The labelling, also called a configuration in the random fields terminology, corresponds in fact to the image itself, being seen as a mapping function from S to Λ:

$$f : S \to \Lambda, \quad f(s) = x \tag{4.1}$$

To develop the MRF theory, the spatial dependencies between sites (image pixels) have to be defined. This is done by choosing the neighbourhood system. The neighbourhood system is the set of all neighbourhoods $N = \{N_s \subset S, s = (i, j) \in S\}$ where N_s represents the neighbourhood of *s*; for example, the neighbouring sites can be defined as those sites *r*, $r \in N_s \subset S$:

$$N_s = \{r = (x, y) \in S : 0 < (x-i)^2 + (y-j)^2 \le ord\} \tag{4.2}$$

where *ord* is the order of the neighbourhood. Some examples of different order neighbourhood systems as used by [*Gem84*] [*Pag98*] are presented in *Fig. 4.1*. The eight order neighbourhood is the five full-square non-causal neighbourhood used in *chapter 3*.

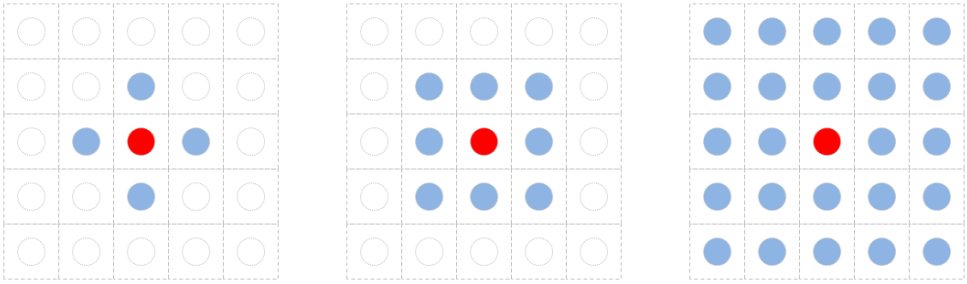


Figure 4.1 – *Examples of different order neighbourhood systems: from left to right, the first (ord=1), second order and the eight order neighbourhoods. The red dots are the site s=(i,j) while the blue dots are the neighbouring sites r=(x,y)* $\in N_s \subset S$.

The neighbourhood system has mainly two properties:

- a site is not neighbouring itself : $s \notin N_s, \forall s \in S$
- the neighbouring relation is symmetrical : $s \in N_r \Leftrightarrow r \in N_s$

Having introduced the notations, the main property of a MRF can straight away be stated: *the random variable* X_s *describing the intensity value at any site s on a lattice* $S = \{s = (i,j) : 0 \le i < w , 0 \le j < h\}$ *can take any value* x_s *from* Λ*, but the probability that* $X_s = x_s$ *depends only on the values* x_r *at sites r neighbouring s.*

The locality and the stationarity properties of a texture, as they were presented in C*hapter 3 - Fig. 3.1*, are nothing but direct consequences of the *markovianity* condition, assumption stating that a pixel value is conditioned entirely only by its neighbouring pixels [*Bes74*] [*Gem84*]:

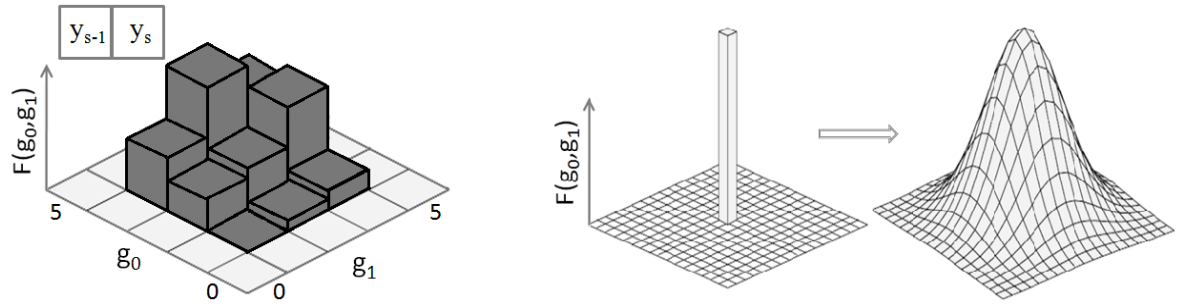
$$LCPDF(s) = P(X_s = x_s \,|\, X_r = x_r, r \ne s) = P(x_s \,|\, x_r, r \in N_s) \quad s \in S, x_s \in \Lambda_s \tag{4.3}$$

This conditional probability is termed as *local conditional probability density function* (LCPDF). MRFs can be specified in terms of their conditional probabilities $P(x_s|x_r, r \in N_s)$ and, inversely, MRFs can be of help in deducing the joint probability distribution Π from the associated local conditional probabilities. The joint distribution defines the probability for a particular labelling realisation. A valid joint distribution is in fact uniquely defined by its LCPDF, by applying the "Hammersley-Clifford theorem" [*Cli90*] or "Markov-Gibbs equivalence theorem" [*Gem84*] establishing the equivalence between the local property (markovianity) and the global property (Gibbs distribution of a Gibbs Random Field). More details about this are to be found in *Appendix C*.

Consequently the LCPDF is capable to capture the visual characteristics of an image, so that it is able to uniquely describe that image.

4.2.2 Local Conditional Probability Density Function

The non-parametric MRF model is based on estimating the LCPDF from a multi-dimensional histogram of the neighbourhood over a homogeneous – i.e. stationary – textured image [*Pag98*]. Each dimension of the histogram represents a site from the neighbourhood system considered in LCPDF definition, the statistical order of the model being equal to the number of dimensions. A complete description of how the multi-dimensional histogram is computed from a homogenous texture disposing of a neighbourhood system is found in the original article [*Pag98*].



a *b*

Figure 4.2 – *Multidimensional histogram and Parzen-window estimator [Sil86]: (a) a one neighbourhood system and its 2-dimensional histogram; (b) illustration of how a histogram data point is spread into the shape of a Gaussian surface.*

Yet, to illustrate the procedure, a simple case of a 2-dimensional histogram obtained for a one neighbourhood system $N = \{N_s = \{s-1\}\}$ is presented in *Fig. 4.2a*. The frequency $F(g_0,g_1)$ counts the number of occurrences of the set of grey levels $\{g_0, g_1\}$ in the image y. The label g_0 represents the value of pixel y_s, while label g_1 is the value of the neighbouring pixel y_{s-1}. To build the histogram, the value of $F()$ is incremented for each site s by scan-line visiting the image Y. In the above minimal case, the frequency is reduced mathematically to compute $F(g_0, g_1) = \sum_{s \in S} \delta(y_s - g_0) \cdot \delta(y_{s-1} - g_1)$, where δ is the Kronecker delta function.

The LCPDF is obtained by performing density estimation on the multi-dimensional histogram. A common non-parametric estimator, that spreads each sample data into a smooth multidimensional histogram over a larger area, is the Parzen-window density estimator [*Sil86*]. Its principle is to associate each point in the histogram with, for example, a standard multi-dimensional Gaussian density as described by *Fig 4.2b*.

Let $z_p = Col[y_p, y_q, q \in N_p]$ be a column vector of size *d*, containing the pixel value y_p and all the values within neighbourhood N_p in the *input sample* *Y* while $z = Col[x_s, x_r, r \in N_s]$ refers to a given configuration in the *output texture X* on which the Parzen-window estimator is to be applied. As demonstrated by [*Pag98*], the nonparametric MRF model gets to be uniquely defined by the nonparametric estimation of the LCPDF:

$$\widehat{LCPDF}(s) = \hat{P}(x_s | x_r, r \in N_s) = \frac{\hat{f}(z = Col[x_s, x_r, r \in N_s])}{\sum_{\lambda_s \in \Lambda} \hat{f}(z = Col[\lambda_s, x_r, r \in N_s])} \tag{4.4}$$

where $\hat{f}$ corresponds to the Parzen-window density estimated frequency:

$$\hat{f}(z) = \frac{1}{n \cdot h^d} \sum_{p \in S_y, N_p \subset S_y} K\left\{\frac{1}{h}(z - z_p)\right\} \tag{4.5}$$

n is the number of possible neighbourhoods N_p in *Y*, i.e. the number of sites $p \in S_y$ for which $N_p \subset S_y$.

The shape of the smoothing is defined by the kernel function *K*, chosen to be the standard multi-dimensional Gaussian density function:

$$K(z) = \frac{1}{(2\pi)^{d/2}} \exp(-\frac{1}{2} z^T \cdot z) \tag{4.6}$$

where $(.)^T$ indicates the matrix transpose.

Finally, the size of *K* is modified by the window parameter *h* in equation 4.5. The value of *h* has to be properly chosen in order to obtain a good estimation. If *h* is too small, the smoothing character of the Parzen-Window is too small, not general enough to represent texture details, so the results will be noisy. If *h* is too large, the estimator is over-smoothed and the results are out-of-focus or blurred. Silverman [*Sil86*] provides an optimal value for the window parameter assuring that the estimation is close to the true density function:

$$h_{opt} = \sigma\left\{\frac{4}{n(2d+1)}\right\}^{1/(d+4)} \tag{4.7}$$

where σ^2 is the marginal variance of the sample texture.

Exploiting the LCPDF estimator in *eq.4.4*, one can say that a value x_s of a pixel at site *s* depends only on the values x_r at sites neighbouring *s*. Mathematically speaking:

$$\hat{P}(x_s | x_r, r \in N_s) = \frac{\sum_{p \in S_y, N_p \subset S_y} \exp\left[-\frac{1}{2h^2}(z_{x_s} - z_p)^T (z_{x_s} - z_p)\right]}{\sum_{\lambda_s \in \Lambda} \sum_{p \in S_y, N_p \subset S_y} \exp\left[-\frac{1}{2h^2}(z_{\lambda_s} - z_p)^T (z_{\lambda_s} - z_p)\right]} \tag{4.8}$$

4.2.3 Texture synthesis via relaxation

Texture synthesis happens by benefiting of the NP-MRF texture model via the LCPDF estimation.

The synthesis of a texture from a MRF model is based upon a stochastic or a deterministic relaxation algorithm [*Gem84*] guided by the LCPDF.

More often used and fairly more known are the stochastic relaxation algorithms like the Metropolis algorithm [*Met53*] and the Gibbs sampler [*Gem84*] (these algorithms are presented in the *Appendix D*). The principle behind this relaxation is that the synthesis starts with an image and iteratively updates pixels in the image with respect to the LCPDF, obtaining a sequence of images $\{x(0), x(1), ..., x(n)\}$ in order to reach the joint distribution of $x(n)$, so that $\lim_{n\to\infty} P(x(n)|x(0)) = \Pi(x(n))$ [*Pag98*].

[*Bes86*] introduced a deterministic relaxation algorithm, called the Iterated Conditional Modes, suggesting that it returns the mode of the LCPDF. The ICM algorithm is presented in *Fig. 4.3*. The ICM algorithm generates a sequence of images, in order to find an equilibrium state at a local maximum, usually somewhere close to the initial image. The synthetic image will suffer from no changes after reaching the equilibrium at a local maximum of Π when all the pixels are affected with the mode of their corresponding LCPDFs.

→ **set up a random visiting order of the sites** $\{a_k \in S, k = 1,2,...\}$
so that $s \in S, a_k = s$ **infinitely often**

→ **choose randomly** $x(0)$

→ **for** k **= 1 to** ∞ **do**

→ **get** $\lambda^*_{a_k} = \arg\max_{\lambda_{a_k} \in \Lambda} P(\lambda_{a_k} | x_r(k-1), r \in N_{a_k})$

→ **image** $x(k)$ **is obtained from the previous image** $x(k-1)$ **such that**

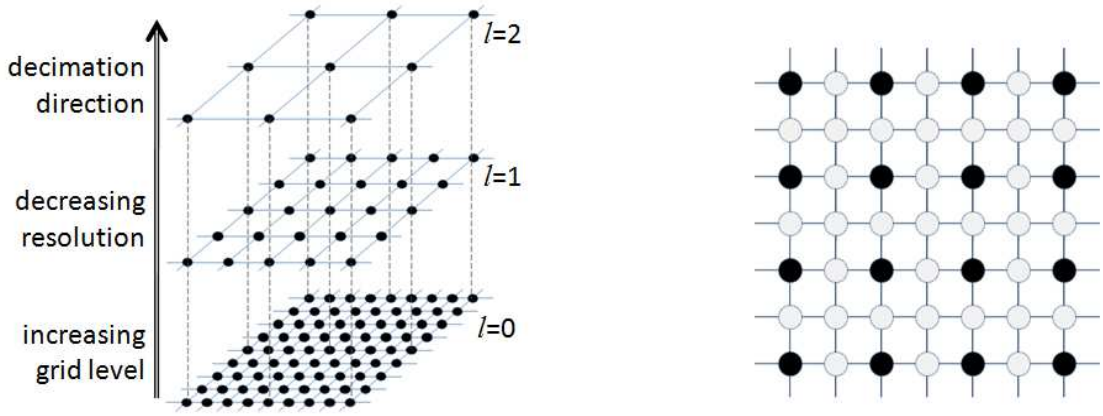
$$x_s(k) = \begin{cases} \lambda^*_{a_k} & \text{if } s = a_k \\ x_s(k-1) & \text{otherwise} \end{cases}$$

Figure 4.3 – *The steps of the Iterated Conditional Modes algorithm.*

In practice, as explained in the two next paragraphs, texture synthesis is performed using a multi-scale synthesis algorithm which incorporates a pixel temperature function used for the first time in [*Pag98*].

4.2.4 Multi-scale implementation

The multi-scale implementation aims both at capturing the textural patterns at different scales (i.e. the local interactions between pixel values) and at speeding up the relaxation process. It indeed helps the ICM algorithm converge to an image closer to the global maximum of the joint distribution [*Bou91*]. Having an image at a local maximum of the LCPDF is enough to obtain a satisfactory synthetic texture, similar to the sample texture.



a *b*

Figure 4.4 – *(a) example of a 3-scale grid; (b) illustration of an intermediary scale showing the 2:1 decimation and the pixels coming from the upper scale (black- filled circles).*

The same multi-scale grid representation as used by [*Bes86*] is employed. It consists in obtaining the lower resolutions, or the higher grid levels $l > 0$, from the image at level $l=0$ by using pixel decimation, as illustrated in *Fig. 4.4*. High frequency texture artefacts are important at high resolutions, while lower resolutions synthesis behaves better on low frequency features. This corresponds actually to the image pyramids in the multi-resolution schema used for the non-parametric algorithms in *Chapter 3*, the difference being that the Gaussian pyramids were used, while here only simple 2:1 decimation is used.

The synthesis (i.e. relaxation) starts at the highest level $l=L$ and continues from one level to another until $l=0$; once a level l was synthesized (i.e. relaxed or reached a maximum), to pass to the next level $l+1$, all the pixels from the smaller level l are copied in their corresponding positions at the higher level $l+1$ and then the synthesis carries on at the upgraded level.

4.2.5 Pixel temperature function

The original 2D synthesis process [*Pag98*] incorporated the pixel temperature function to serve as a degree of confidence relative to the pixel value. The pixel confidence is associated with the probability that grey-level x_s represents the correct pixel value for the site *s*. Each pixel is given a certain temperature t_s, between 0 and 1, where 0 means complete confidence. The synthesis starts at a high global temperature (i.e. no confidence) and the process is considered to be finished when equilibrium is reached – temperature 0 for all pixels. The iterative cooling down schema is controlled by the pixel temperature function. The pixel temperature is used more as an indicator, not having the same role as in a classic annealing schedule [*Gem84*].

Mathematically, full pixel confidence happens when x_s is sampled from a LCPDF at equilibrium, or when the LCPDF is completely conditional on its neighbouring pixel values. Once a pixel at a site *s* was relaxed on one occasion, the confidence associated to its value x_s is updated by relying only on the temperature of the neighbouring pixels t_r, $r \in N_s$ as follows:

$$t_s = \max\left\{0, \frac{\xi + \sum_{r \in N_s} t_r}{|N_s|}\right\}, \ \xi < 0 \qquad (4.9)$$

ξ is used to control the rate of the cooling schedule, that is the rate at which a pixel is cooled based on his neighbours temperatures. At high temperatures the synthesized textures are not spatially correlated, but as the temperatures decrease, the correlations induced by the LCPDF become stronger and the cooling rate should be slowed down to capture correctly the image characteristics (i.e. spatial correlations).

In the MRF model, the pixel temperature function is used to condition the LCPDF: if $t_s = 0$ the LCPDF should be strong, while for $t_s = 1$ it should be weaker. To ensure this, the handling of pixel temperature is integrated in the form of $(z - z_p)$ while estimating the LCPDF as follows:

$$(z - z_p) = Col[x_s - y_p, (x_r - y_r) \cdot (1 - t_r), r \in N_s] \qquad (4.10)$$

Pixel temperature management is a major component of the multi-scale process, first by determining the length of the relaxation process and secondly by reducing the number of pixels to be relaxed. This means that before the relaxation algorithm starts at level *l*, those pixels which were relaxed at the previous level, *l+1*, are copied on level *l*, according to the grid representation, and are given the temperature $t_s = 0$ (i.e. complete confidence) being considered as already relaxed sites having correct values. The other pixels have their pixel temperatures initialized to $t_s = 1$, i.e. no confidence; after what a pixel is relaxed, it has its temperature reduced according to *eq.4.9* until gaining complete confidence.

4.2.6 2D texture synthesis algorithm

After exposing all the concepts and the synthesis components, *Fig.4.5* draws the basic steps to follow in order to compose the 2D texture synthesis. As the previous non-parametric methods presented in *Chapter 3*, this one is based on a 2D example also.

At the beginning of synthesis, the input texture and the desired output texture (initialised with noise) have to be defined as two sets of sites on a lattice. The neighbourhood system and the number of scales have also to be chosen, decimating the lattices in a multi-grid form as represented in *Fig. 4.4*. Every site corresponds to a pixel from the texture and every site is initialized with a temperature of 1 (no confidence) while the global temperature (i.e. sum of all temperatures) of each scale is computed. A certain scale is considered to be synthesized when the global temperature reaches 0, meaning that all the pixels gained full confidence.

The synthesis starts at the highest grid level and proceeds by randomly choosing sites from the lattice. If the temperature of a site is positive the best realization of that site is questioned. The decision is made by finding the value, within the available configuration, that maximizes the LCPDF of that site given by *eq. 4.8*, and using ICM relaxation. After a site is treated, its temperature is updated as indicated in *eq. 4.9*. If the temperature of a site is null, it means that the corresponding pixel has a correct value and no LCPDF estimation is computed, the synthesis process is simply ignoring it. Once the lattice is fully scanned, the global temperature is recomputed.

If the global temperature at a given scale reaches zero, the next lower scale is synthesized (if the highest resolution isn't yet reached). Passing the information from one scale to another one consists in copying the pixels from the poorest scale onto the corresponding sites on the richest scale as illustrated in *Fig 4.4b*. The updated sites have full confidence. At the beginning of a brute scale, at each four pixels displayed in a square shape,

three are noised unreliable pixels that will be later on processed; one has full confidence and will influence its neighbours by means of their LCPDF.

Input

$Y \leftrightarrow$ 2D sample texture $\quad w_Y \times h_Y \leftrightarrow$ size of sample texture Y

$X \leftrightarrow$ 2D output texture $\quad w_X \times h_X \leftrightarrow$ size of targeted texture X

n_size, level $\leftrightarrow$ neighbourhood system and number of scales

Begin

$Y \leftrightarrow$ multi-scale representation $Y^{level} \leftrightarrow$ set of sites s^{level} on the lattice S^{level}

$X \leftrightarrow$ initialized with noise $\leftrightarrow$ multi-scale representation X^{level}

initialize all pixels temperatures $t_s = 1 \quad \forall s \in S^{level}$

for every scale (starting from the highest grid)

if global temperature $\sum_{s \in S^{level}} t_s > 0$

for every site s in the output lattice

if temperature $t_s \neq 0$

find $\lambda_s^* = \arg\max_{\lambda_s \in \Lambda} P(\lambda_s | x_r, r \in N_s)$

$x_s = \lambda_s^*$

update t_s

end if

end for

update global temperature

end if

update upper scale

end for

End

Figure 4.5 – *2D texture synthesis algorithm based on the NP-MRF model.*

Some examples of 2D textures synthesized with the NP-MRF model via LCPDF estimated with Parzen-windowing, using the ICM relaxation, pixel temperature and multi-scale representation are presented in *Fig. 4.6*.

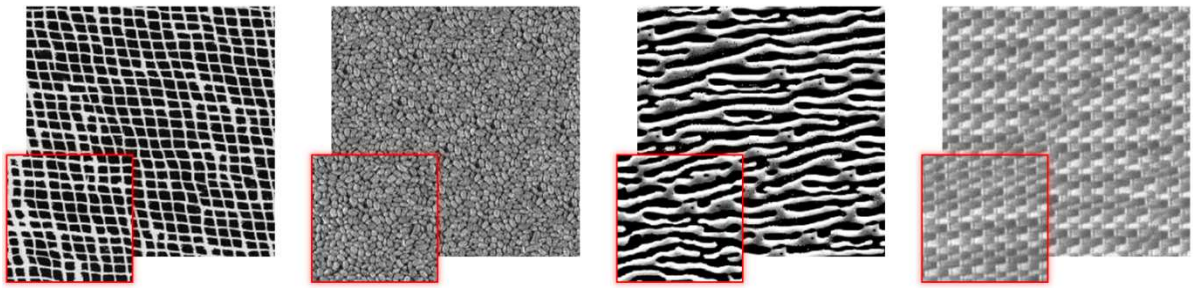


Figure 4.6 – *Illustration of the NP-MRF based 2D texture synthesis algorithm from Fig. 4.5: for each of the four cases the synthesis framework consisted in an input image (enclosed in red) of size 100×100 pixels and using 3 scales and a 7×7 full-square neighbourhood to obtain the 200×200 pixels textures. To reduce the computation time, the input textures were reduced to only 32 grey-levels.*

4.3 3D extension of the NP-MRF synthesis algorithm

4.3.1 General considerations

As already mentioned, the 3D algorithm is a single-image-based synthesis algorithm, so to start the synthesis the process requires a texture image defined as a lattice S, where every pixel is a site on the lattice.

To be able to perform a visually correct synthesis, the input image is required to have a relative good resolution, so that if we decrease its resolution, the important textural patterns should be retrieved at the different scales. The decreasing of the resolution grid is done by simple decimation as in 2D.

The process starts with choosing the number of grid levels and the size of the full-square neighbourhood. The output block dimensions are chosen and it is filled up with noise – actually random pixels from the exemplar, to preserve the dynamics and help the synthesis reach a maximum faster. Next, the sample and the output 3D block are decomposed at different scales and defined as lattice systems. 3D decimation is described by *Fig. 4.7*. From a set of eight voxels (in a 2×2×2 3D configuration) only one is being projected on the next higher level grid.

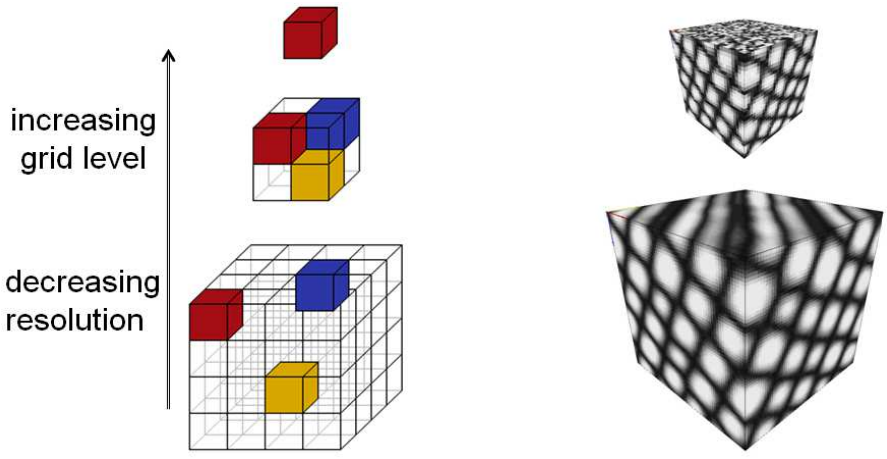


Figure 4.7 – *Example of multi-scale decimation in 3D: (a) the concept of conserving only one voxel from each set of eight; (2) two snapshots, taken inside the synthesis process, showing two adjacent 3D scales (a 32^3 block and a 64^3 block).*

The multi-scale relaxation algorithm is employed similarly to the 2D case. Idem for the pixels temperature used to control the synthesis as a kind of local annealing process. Sites are now in 3D configurations and they are scanned in a random order, searching for each site to maximize its LCPDF using an ICM relaxation.

4.3.2 2D/3D generic algorithm

With the help of the above considerations and the 2D credentials, the synthesis process for a 3D texture can be essentially described as follows:

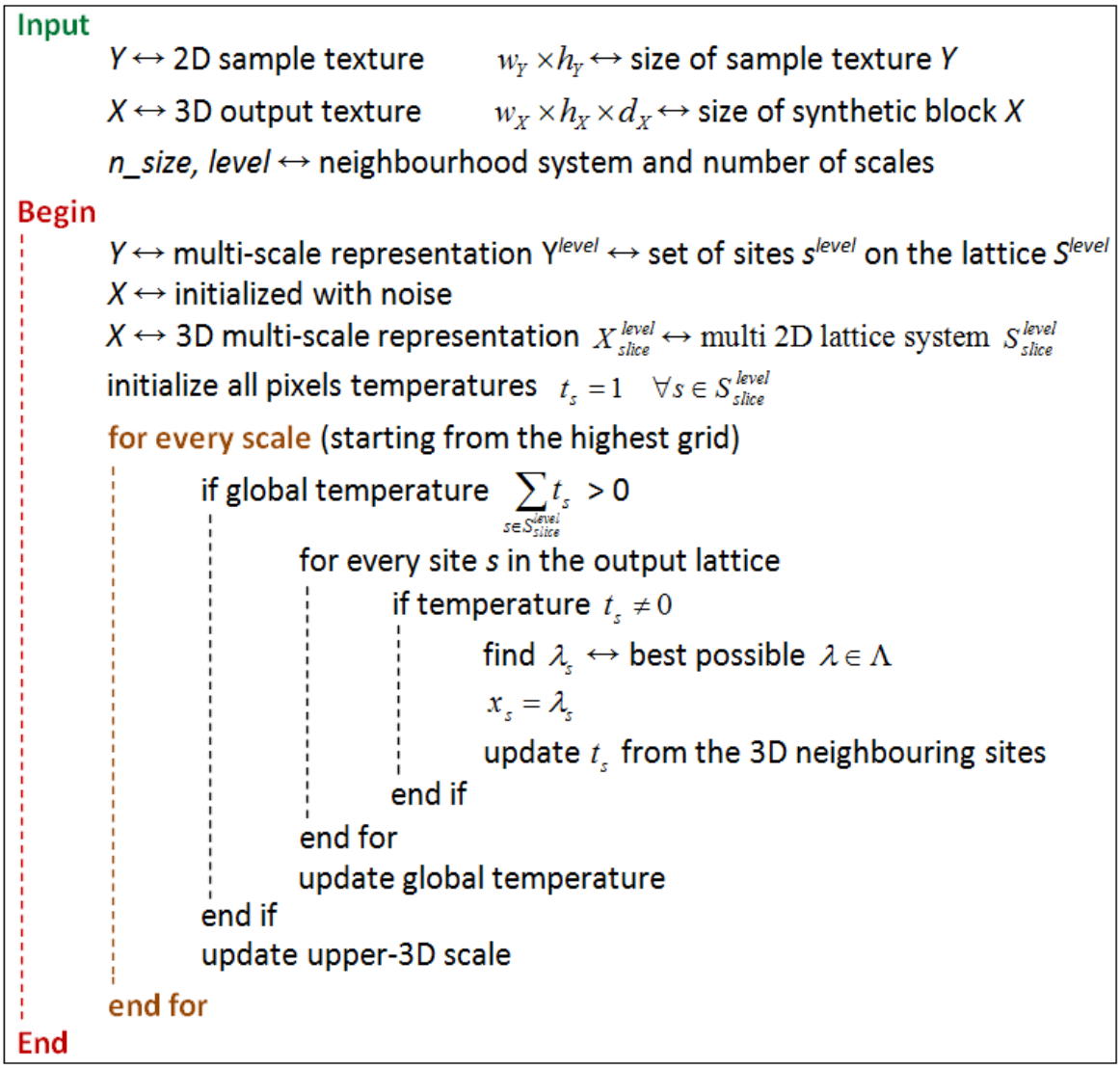


Figure 4.8 – *Algorithm: volumetric texture synthesis using MRF model showing the required data for the input and the steps to follow.*

4.3.3 Updating the voxel value: why a full-3D process is not relevant ?

As in the case of the non-parametric extension to three-dimensions addressed in *Chapter 3*, the difficulties encountered at this point put forward the same question: how to infer 3D information when only 2D information is available?

The problematic issue is to find a good strategy capable of working out the LCPDF of a site on the 3D grid by means of information available on the 2D lattice of the input sample.

The ideal solution would be to figure out the full-3D LCPDF associated to a site s in 3D, and subsequently to retrieve its associated grey level λ_s^* so that the LCPDF is completely conditional on its full-3D neighbouring pixel values.

$$\lambda_s^* = \underset{\lambda_s \in \Lambda}{\arg\max}\{LCPDF_{3D}(\lambda_s)\} = \underset{\lambda_s \in \Lambda}{\arg\max}\left\{\hat{P}(\lambda_s | x_r, r \in N_s^{3D})\right\} \tag{4.11}$$

where N_s^{3D} defines a 3D neighbourhood of the current site *s*. An example could be the second order neighbourhood presented in *Fig. 4.9a* and defined by:

$$N_{s=(i,j,k)}^{3D} = \left\{ r = (x,y,z) \in S : 0 < (x-i)^2 + (y-j)^2 + (z-k)^2 \leq 2 \right\} \qquad (4.12)$$

However, except in trivial cases, it is not possible to estimate the conditional probability of such configurations to assess the 3D interactions inside the 3D neighbourhood. This observation led us to propose heuristic approaches in order to overcome this problem.

4.3.4 Heuristic approaches

The 2D/3D extension infers the output block to be treated in a multi-2D form being seen as an arrangement of manifold 2D sheets. It implies a multi-2D/2D synthesis that comes down to crumbling the 3D neighbourhood into orthogonal 2D neighbourhoods.

4.3.4.1 Breaking down the 3D neighbourhood

The outcome of breaking down the 3D neighbourhood into orthogonal 2D neighbourhoods is that the unknown full-3D LCPDF estimation is reduced to a compound of computable 2D estimations.

The simplification of the 3D lattice neighbourhood (N_s^{3D}) of a site into three orthogonal 2D neighbourhoods – front neighbourhood (N_{Fs}), right neighbourhood (N_{Rs}) and top neighbourhood (N_{Ts}) related to the central site, is shown in *Fig. 4.9*.

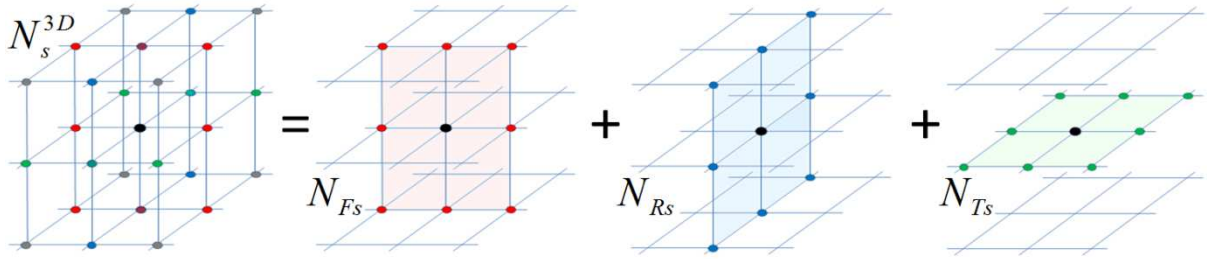


Figure 4.9 – *Breaking down a second order 3D neighbourhood - N_s^{3D} in (a), into three 2D neighbourhoods corresponding to the orthogonal views: (b) front N_{Fs}, (c) right N_{Rs} and (d) top N_{Ts}.*

This kind of conceptualization shows the need for the heuristics to attain for voxel v_s the maximum likelihood (ML) grey level λ_s. It gives the opportunity to formulate several heuristics as proposed in the following two sections.

4.3.4.2 Heuristic 1: *NP_ML_H1*

According to the 3D neighbourhood breaking scheme in *Fig. 4.9*, one can find a suitable grey level for a voxel v_s at site *s* as a combination of the most probable grey levels found separately for each of its orthogonal views, as a 2D/2D search problem.

The result consists in several grey levels and several LCPDFs, one pair for each view – $(\lambda_{front}, \hat{P}_{front})$, $(\lambda_{right}, \hat{P}_{right})$ and $(\lambda_{top}, \hat{P}_{top})$, retrieved through separate ICMs:

$$\lambda^*_{view} = \arg\max_{\lambda_{view} \in \Lambda} \left\{ \hat{P}_{view}(\lambda_{view} | x_r, r \in N_{view\_s}) \right\} \quad (4.13)$$

This solution reminds us of the C*hapter 3*, where the update value for a voxel was computed as a combination of the solutions found for each orthogonal view. Similarly, the grey level chosen as update value can be found by combining the grey levels proposed by each orthogonal view using as strategy:

- choosing the one with the lowest/highest associated LCPDF - *NP_ML_H1a/b*,
- the average as Wei and Levoy [*Wei03*] - *NP_ML_H1c*,
- the weighted mean as Kopf et *al.* [*Kop07*],

while the nearest neighbour search is replaced by the maximum likelihood search.

Algorithmically this can be presented as in *Fig. 4.10*:

for every site *s*

find $\lambda_{front} = \arg\max_{\lambda_s \in \Lambda} \hat{P}(\lambda_s | x_r, r \in N_{Fs})$

find $\lambda_{right} = \arg\max_{\lambda_s \in \Lambda} \hat{P}(\lambda_s | x_r, r \in N_{Rs})$

find $\lambda_{top} = \arg\max_{\lambda_s \in \Lambda} \hat{P}(\lambda_s | x_r, r \in N_{Ts})$

select combination strategy using weights $\rightarrow w_{\lambda_{front}}, w_{\lambda_{right}}, w_{\lambda_{top}}$

update value $x_s = \lambda_{front} \times w_{\lambda_{front}} + \lambda_{right} \times w_{\lambda_{right}} + \lambda_{top} \times w_{\lambda_{top}}$

end for

Figure 4.10 – *Principle of NP_ML_H1 heuristic.*

4.3.4.3 Heuristic 2: *NP_ML_H2*

In order to avoid performing several ICMs and to avoid dealing with several solutions for each orthogonal view, a second heuristic is proposed. The plan consists in finding a unique grey level λ^*_s which maximizes a function of the 2D estimations associated to the orthogonal views:

$$\lambda^*_s = \arg\max_{\lambda_s \in \Lambda} \left\{ f(LCPDF_{front}(\lambda_s), LCPDF_{right}(\lambda_s), LCPDF_{top}(\lambda_s)) \right\} =$$
$$= \arg\max_{\lambda_s \in \Lambda} \left\{ f(\hat{P}_{front}(\lambda_{front} | x_r, r \in N_{Fs}), \hat{P}_{right}(\lambda_{right} | x_r, r \in N_{Rs}), \hat{P}_{top}(\lambda_{top} | x_r, r \in N_{Ts})) \right\} \quad (4.14)$$

where the function *f* can be defined as:

$$f(LCPDF_{front}, LCPDF_{right}, LCPDF_{top}) = \min(LCPDF_{front}, LCPDF_{right}, LCPDF_{top}) \quad (4.15)$$

or

$$f(LCPDF_{front}, LCPDF_{right}, LCPDF_{top}) = \max(LCPDF_{front}, LCPDF_{right}, LCPDF_{top}) \quad (4.16)$$

or

$$f(LCPDF_{front}, LCPDF_{right}, LCPDF_{top}) = product(LCPDF_{front}, LCPDF_{right}, LCPDF_{top}) \quad (4.17)$$

Depending on the function f used for ICM relaxation as sketched in *Fig. 4.11*, these heuristics are tagged as follows:

- *NP_ ML_H2a* for the expression of f from *eq. 4.14* that maximizes the smallest of the three likelihoods in order to maximize all of them
- *NP_ ML_H2b* for f as in *eq. 4.15* that maximizes the biggest likelihood, favouring one of the three views if it shows a good likelihood
- *NP_ ML_H2c* for an f defined as in *eq. 4.16* to maximize the product searching for a compromise for all the three views

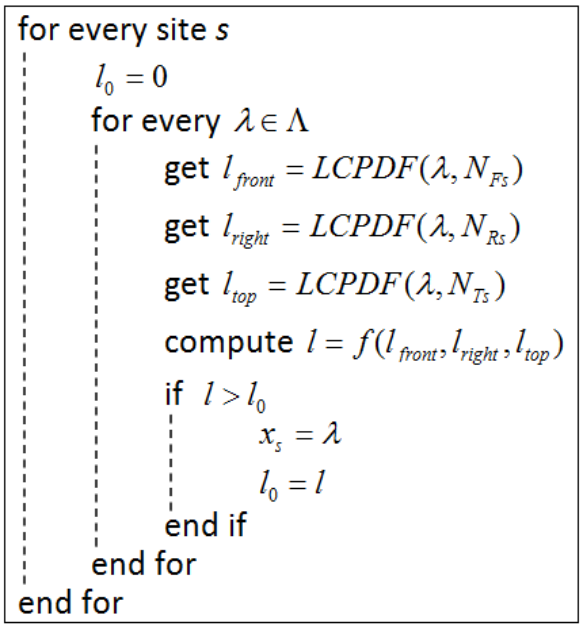


Figure 4.11 – *Principle of NP_ML_H2 heuristic.*

4.3.5 Implementation details

The synthesis plan follows the steps from *Fig 4.8* and uses a heuristic from the ones proposed above to perform the ML estimation, but some annotations have to be mentioned.

4.3.5.1 Initialisation of the output block

The initialisation strategy for the output block consists in using pixels picked randomly (white noise) from the input exemplar in order to keep the same global distribution of grey levels.

But to accelerate the synthesis process by speeding up the relaxation and help it maintain the global structure of the exemplar the output block is initialised with chunks from the input image copied as patches on the orthogonal 2D slices of the block into a random or deterministic way.

An example showing the simple insertions of the input image at different depths of the 3D block is in *Fig. 4.12*.

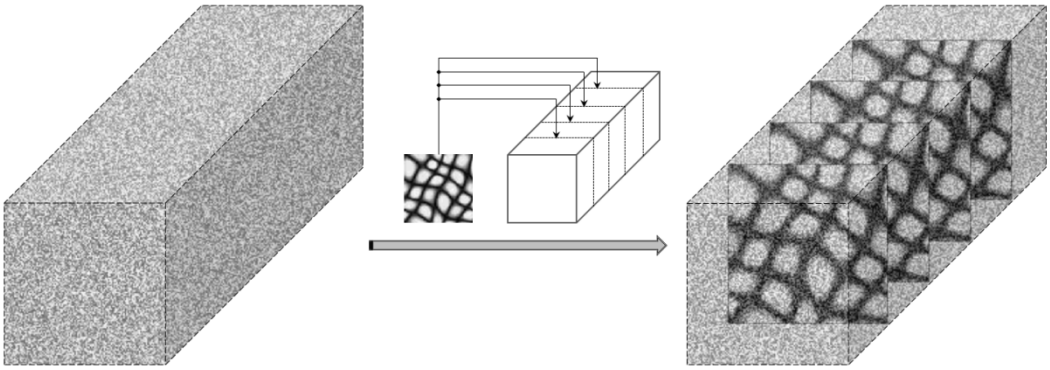


Figure 4.12 – *Output block initialization following a simple strategy: the targeted block is initialized with noise and at certain depths the same image used as synthesis sample is inserted as a slice within the block; this will be the initial state of the output block that will be used further on in the synthesis process.*

4.3.5.2 Scales handling

Concerning the voxels grey levels, the operation consisting in going from a low scale toward higher scale, is done by decimation as in *Fig. 4.7a*. The contrary operation, top to down scale direction, is done systematically after each synthesized scale either copying each high scale pixel value into one of eight lower scale pixels (1/8 up-sampling scheme) either by copying each high scale pixel value into eight lower scale pixels (8/8 up-sampling scheme). This is illustrated in *Fig. 4.13*.

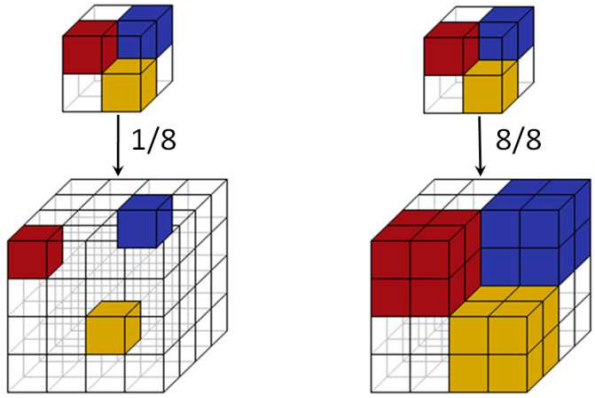


Figure 4.13 – *Going down to a lower scale: the top scale values are used to initialize the lower scale by updating only one voxel in eight (1/8), or by updating the same voxel over all the eight (8/8).*

There are two ways to deal with the temperature of the voxels obtained from a higher scale:

The temperatures of the voxels from a high scale can be copied to the next lower scale as a 1/8 update only after that the high scale was synthesized. This means that the high scale voxels have full confidence (i.e. $t_s = 0$) and they are propagated onto the next lower scale keeping themselves the same temperatures. These voxels are used as fixed voxels for every level. This allows a faster synthesis, but not necessary a better one. If an erroneous voxel is to take place at a higher level (at low resolution) the error will propagate until the final level, in the synthesized texture.

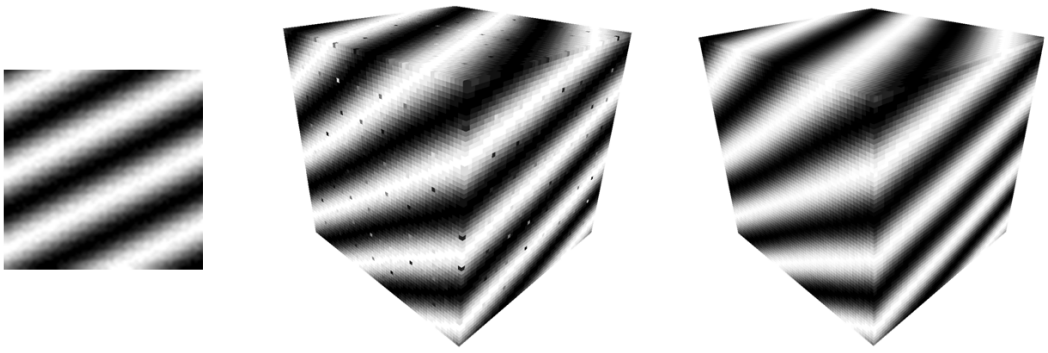
The second and preferred solution consists in using the pixels from a higher scale to initialise the next lower scale, but to reconsider them during the synthesis. It denotes the fact that the grey value of a voxel from a previous level is changeable but its temperature remains fixed.

Making a zoom in the algorithm at the position where the sites are enquired, gives us the extended form in *Fig. 4.14*.

```
...
for every site s at the position (xs,ys,zs) in the output lattice
    if temperature t_s ≠ 0
        find λ*_s = arg max_{λ_s ∈ Λ} {LCPDF_3D(x_s, λ_s)}
        x_s = λ*_s
        update t_s from the 3D neighbouring sites
    end if
    update global temperature
    if (xs,ys,zs) % 2 = 0 (even voxels inherited from the higher scale)
        find λ*_s = arg max_{λ_s ∈ Λ} {LCPDF_3D(x_s, λ_s)}
        x_s = λ*_s
    end if
end for
...
```

Figure 4.14 – *Texture synthesis algorithm enhanced with the synthesis of voxels from the sites having even coordinates (i.e. the voxels inherited from the higher scale).*

This re-synthesizing phase for the voxels having even coordinates increases the computational cost but it improves the results as shown in *Fig 4.15*. The effect of this phase is not visible on the 2D synthesized results as much as it is on the 3D results, because the update from one scale to another in 2D means to update only one pixel in four, while for the 3D case it involves copying one voxels in eight. Even if the inverse-decimation operation involves copying the same pixel over all the four or the voxel over the eight, it doesn't resolve the error propagation issue.



a *b* *c*

Figure 4.15 – *Example of two volumetric synthetic blocks obtained from (a) - a 2D image, by applying the multi-scale relaxation MRF synthesis method using the heuristic NP_ML_H2c: the block in (c) is obtained by re-synthesizing the even voxels updated from the upper scale as suggested in Fig. 4.14, visually improving the result from (b) that was obtained without reconsidering the even voxels.*

4.4 Conclusion

This chapter offers a brief survey of the MRF theory introducing the necessary concepts to describe the LCPDF and its non-parametric estimation. Next, based on the 2D algorithm [*Pag98*], an original 3D extension is proposed, relying mainly on giving a new description to the LCPDF of a site in 3D and using more adequately the voxel temperature function.

Therefore, using our heuristic propositions from *section 4.3* several versions of the 2D/3D synthesis algorithm can be examined. A list of the relevant algorithms that are going to be evaluated in the next chapter is presented in the table below:

Heuristic tag	Heuristic decision to find the best λ_s
NP_ML_H1a	$\min\left(\arg\max(LCPDF_{front}), \arg\max(LCPDF_{right}), \arg\max(LCPDF_{top})\right)$
NP_ML_H1b	$\max\left(\arg\max(LCPDF_{front}), \arg\max(LCPDF_{right}), \arg\max(LCPDF_{top})\right)$
NP_ML_H1c	$mean\left(\arg\max(LCPDF_{front}), \arg\max(LCPDF_{right}), \arg\max(LCPDF_{top})\right)$
NP_ML_H2a	$\arg\max\left\{\min(LCPDF_{front}, LCPDF_{right}, LCPDF_{top})\right\}$
NP_ML_H2b	$\arg\max\left\{\max(LCPDF_{front}, LCPDF_{right}, LCPDF_{top})\right\}$
NP_ML_H2c	$\arg\max\left\{product\,(LCPDF_{front}, LCPDF_{right}, LCPDF_{top})\right\}$

Table 4.1 – *Proposed heuristics for finding the grey level for the output voxel.*

Besides the heuristic itself, the way of initialising the output block (noise or input image patches), the scales update strategy (1/8 up-sampling or 8/8 up-sampling) and the handling of the even pixels (reiterating them or not), there are other factors that contribute to the quality of the results. These are, as in the case of the non-parametric algorithm based on the neighbourhood search, the neighbourhood system (size, shape), the scan type and the number of scales.

Unlike the methods in *Chapter 3*, for which the user decides the number of iterations, here the iterative process is in the form of the cooling schedule, the temperature cooling rate being determined by the factor ξ in *eq. 4.9*. However, the synthesis process can be stopped after a given number of iterations or it can be imposed to stop when the global temperature doesn't change significantly.

From a computational point of view, letting the algorithm to follow its course through all the cooling steps proves to have poor performances. The solution for accelerating the algorithm in this situation consists in parallelising the algorithm; it means to simultaneously relax a set of i.i.d. (i.e. independent and identically distributed) sites.

Chapter 5

Volumetric texture synthesis results

Contents

5.1 Objective and methodology 69

5.2 Fixed-neighbourhood search based synthesis results 71

5.2.1 Comparing the approaches 71

5.2.2 Parameters influence 74

5.3 Maximum-likelihood based synthesis results 78

5.3.1 Comparing the heuristics 78

5.3.2 Temperature decreasing schema 82

5.3.2 Common parameters influence 84

5.4 Fixed-neighbourhood search vs. likelihood maximisation 87

5.5 About 2D/3D inference 91

5.6 Conclusion 93

5.1 Objective and methodology

This chapter deals with putting into practice the non-parametric synthesis algorithms presented in the previous two chapters. The fixed-neighbourhood search based approaches (NP_WL, NP_K and NP_CW) and the maximum-likelihood based synthesis methods (NP_ML_H1a/b/c and NP_ML_H2a/b/c) are applied to the volumetric texture synthesis starting from a single 2D texture, focusing mainly on various Brodatz textures [*Bro66*].

The goal is to analyze the capacity of the non-parametric algorithms to synthesize different types of textures, to show the potential of these methods in various domains. More or less, the synthesized results should resemble the source textures conserving the dynamics and the structure.

In terms of synthesis quality, a critical point consists in taking into account the size of the texture patterns in determining the number of synthesis levels and the neighbourhood size. The scanning order and the output texture initialization strategy are also key factors that help the algorithms to converge towards a reasonable solution. Other more specific factors are for NP_CW the number of candidates, or for maximum-likelihood approaches, the temperature decreasing scheme, the synthesis ending criteria, the up-sampling procedure or the way of handling pixels inherited from the previous scale. By testing the influence of these factors on the synthesis results, one can identify the pertinent strategy for obtaining a good quality of the results.

In addition, comparing the results gives the opportunity of finding the representative approach for the fixed-neighbourhood search based synthesis algorithms, and the representative of the maximum-likelihood based methods. For practicability, 'electing' the representative in each category is performed on a restrained number of textures to show the advantages and disadvantages of the synthesis algorithms. In the end, the approaches for the two cases are compared on a larger base of textures.

The textures of interest are the grey-level images presented in *Fig. 5.1*. As can be seen from these examples, we focus on structured textures – irregular to regular– with properties of anisotropy and periodicity in one or two directions. Care is taken to ensure that the extrapolation of the texture in 3D makes sense in terms of anisotropy. In other words, the existence of a real 3D structure, giving rise to a 2D section similar to the example, has to be possible. As a corollary, the algorithms have to be parameterized so that the synthesis of such 3D structures is possible. For example, an anisotropic 2D texture like the one in *Fig. 5.1.r* cannot be observed on three orthogonal 2D views of a same volumetric texture. However, it could be observed on two orthogonal views, front view and side view for instance. The same is true for the other lamellar textures *5.1q, s* or *t*, but also – which is less obvious – for most of the remaining bi-directional textures. A direct consequence is that the synthesis of meaningful 3D textures from a single sample has to be performed for many of these examples by constraining only two orthogonal views of the solid output block. This is how the algorithms have been parameterized.

The case of isotropic textures is also considered in this evaluation, though to a lesser extent. Examples are textures *5.1k* or *5.1n*. Meaningful 3D textures can be synthesized from such 2D samples by constraining either two or three orthogonal 2D views. Even if some experiments have been conducted by constraining three views, no thorough analysis has been

carried out so far. The results showed for isotropic textures will thus refer to textures synthesized by constraining two views only.

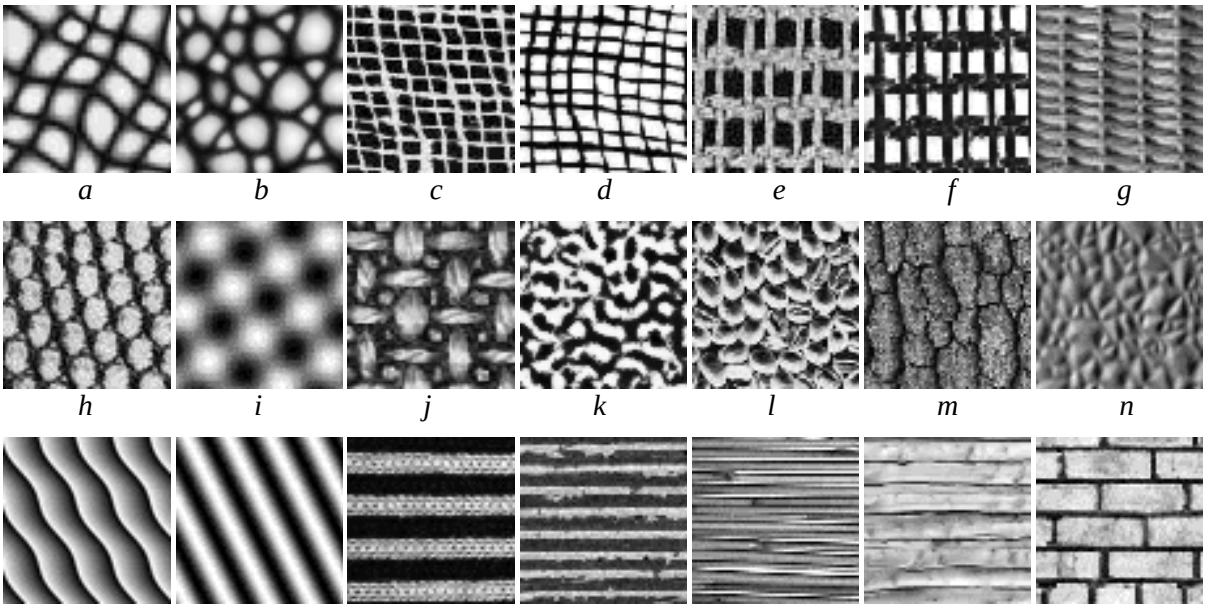


Figure 5.1 – *2D textures used as synthesis examples for the non-parametric algorithms.*

5.2 Fixed-neighbourhood search based synthesis results

This section deals with the comparison of the results obtained by applying the non-parametric synthesis algorithms based on fixed-neighbourhood search described in *Chapter 3*. These algorithms (NP_WL, NP_K and NP_CW) are applied on the 2D sample textures from *Fig. 5.1*.

To begin with, some results are showed using the common framework parameters for which satisfying results are obtained allowing an honest visual comparison, with the aim of finding the approach that provides the best results. By varying the different algorithmic parameters, their influence on the synthetic solid textures is analyzed.

5.2.1 Comparing the approaches

This study focuses on showing in parallel the synthetic blocks using the synthesis algorithms differentiated principally by the way of combining the information from the orthogonal views in order to obtain the update value for the output voxel, as listed below:

Synthesis method	Strategy for output voxel
NP_WL	$repeat \;\; v = (p_{front} + p_{side})/2$
NP_K	$v = \frac{w_{front} \cdot p_{front} + w_{side} \cdot p_{side}}{w_{front} + w_{side}}$
NP_CW	$v = \min(\lvert p_{front} - v^* \rvert, \lvert p_{side} - v^* \rvert)$ $v^* = \frac{w_{position\_front} \cdot p_{front} + w_{position\_side} \cdot p_{side}}{w_{position\_front} + w_{position\_side}}$

Table 5.1 – *Recap of the strategies involved by the non-parametric synthesis algorithms based on fixed-neighbourhood search, used for combining the information from two orthogonal views in order to obtain the update value for the output voxel.*

Each of these strategies (specific for each method) is studied under a common generic support sufficient to produce satisfying results but – more important – to be able to compare the synthesis results for each strategy. This framework consists in using a neighbourhood system composed of a full-square neighbourhood of size 7×7, computed on three Gaussian pyramids, synthesizing the voxels following a random path, output block initialized with white noise and parts from the input sample. In addition, particular only to NP_CW, a set of 15 candidates is used for each pixel from the 2D sample texture. The printed results are the ones obtained after the 10^{th} iteration of the synthesis process.

The input images used as source of synthesis are the 64×64 textures from *Fig. 5.1*, while the output blocks are of size 64×64×64 pixels. The 2D samples used for synthesis are reduced to 32 grey-levels. This relative small size for the output blocks and the reduced grey-levels were chosen in order to be able to compare the results with the ones obtained with the maximum-likelihood based synthesis algorithms (to be seen further on in this chapter), which are proved to be more time consuming, and to assure a large number of synthetic textures for comparison.

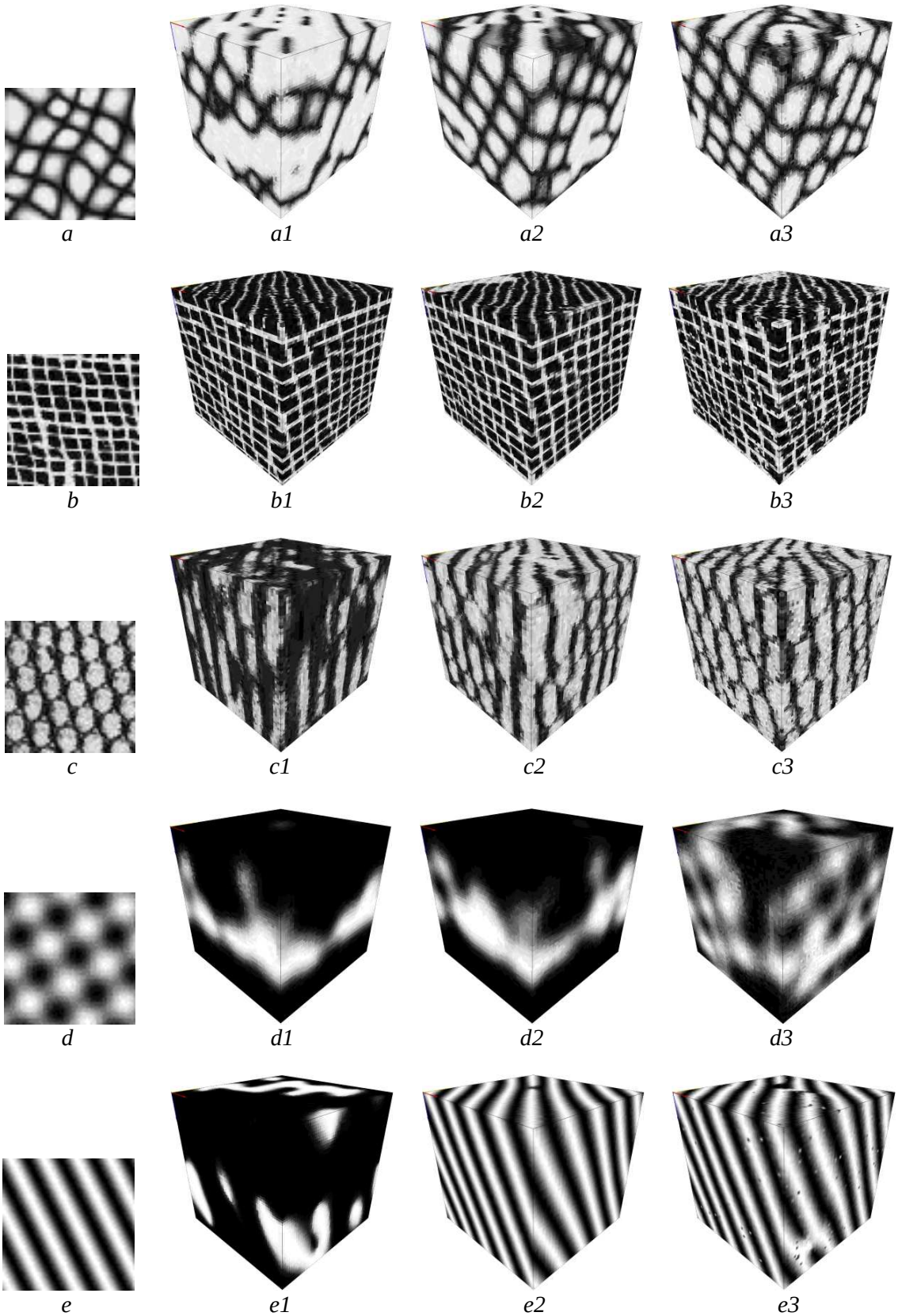


Figure 5.2 – *Volumetric results using the non-parametric algorithms based on a full-square 7×7 neighbourhood search: each row shows from left to right the results obtained by synthesizing the 1st column texture, using NP_WL (2nd column), NP_K (3rd column) and NP_CW (4th column).*

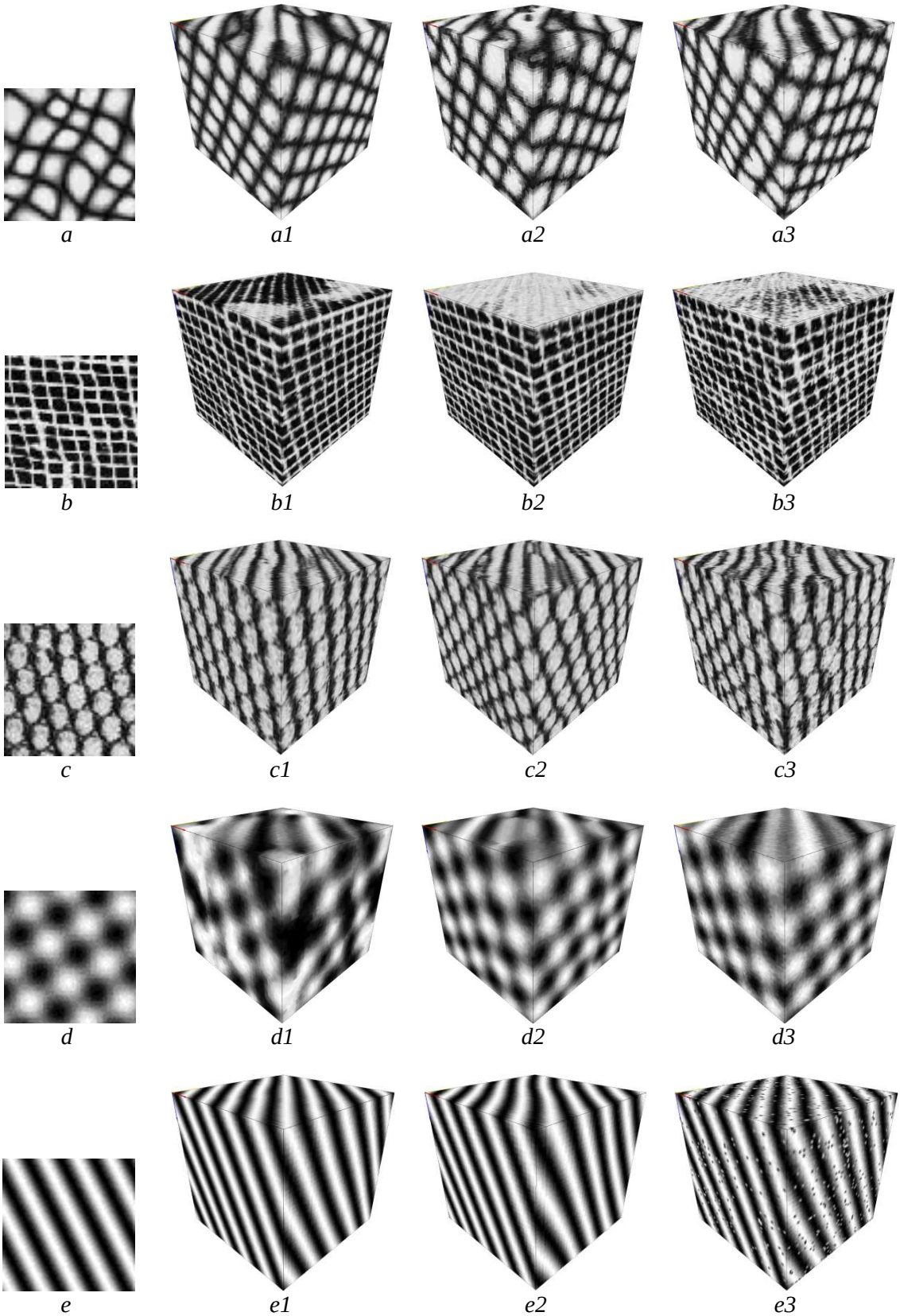


Figure 5.3 – *Volumetric results using the non-parametric algorithms based on a full-square 11×11 neighbourhood search: each row shows from left to right the results obtained by synthesizing the 1st column texture, using NP_WL (2nd column), NP_K (3rd column) and NP_CW (4th column).*

Fig. 5.2 shows some results using the common framework on a 7×7 neighbourhood. Even at this early step of the comparison study, one can make a distinction between the quality of the results for NP_WL (which is not capable of capturing the structure of the sample texture) and the other ones. Concerning NP_K and NP_CW even with a relative small neighbourhood, they can provide promising results.

Then the neighbourhood size is extended to 11×11, results being showed in *Fig. 5.3*. Under these circumstances even the NP_WL acts well presenting relative good results if we ignore the solid texture in *Fig. 5.3d1*.This one is explained by the incapacity of the NP_WL algorithm to resemble the input sample because the average operation involved by the iterative searching process for the best neighbourhood, and for being based only on local decisions.

As for the methods based on histogram adjustment techniques trying to incorporate the global information, the results are similar, only with a comment on the result using NP_CW in *Fig.5.3e3*. The result in cause is unwanted because of the erroneous pixels that appear in the final result. This is caused by the algorithm attempt to distribute completely and uniformly the input pixels in the output block. It means that the algorithm considers a pixel to be a good solution for an output voxel based on the uniformity criterion even if it is not exactly the best one from a neighbourhood distance point of view. But for a simple case as the regular texture in *Fig. 5.3e*, the algorithm fails. This algorithm is indicated for more complex textures and preferably less structured ones, when more diversity is wanted in the output texture.

The NP_K approach satisfies most of the textures in term of synthesis quality, capturing well the input sample structure. The weighted average and the grey-level histogram matching technique work well together. The results don't show lots of defects and per ensemble it provides the most satisfying results. It works also in the cases for which the other two approaches failed. Consequently, only NP_K approach is going to be used for further considerations, to analyse the influence of the typical parameters on the quality of the synthesis results.

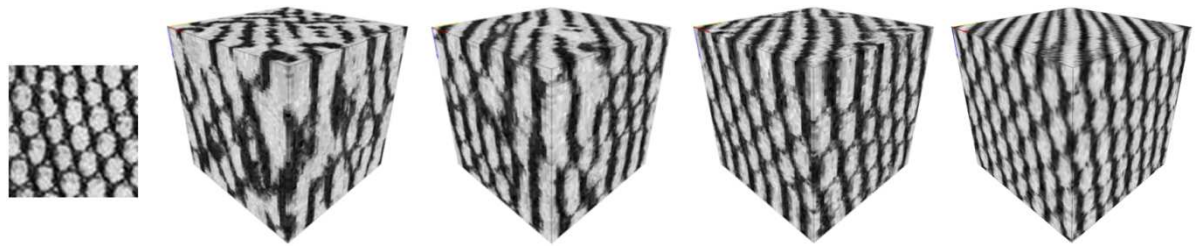
5.2.2 Parameters influence

Just by analyzing the framework of the results in *Fig.5.2* and *Fig. 5.3* it's obvious that the algorithms are very sensitive to the **neighbourhood system**, when determining the number of synthesis levels and the neighbourhood size correlated to the scanning type.

The **neighbourhood size** plays an important role in the synthesis process being a central factor in capturing the input image structure. The neighbourhood has to be bigger than the largest pattern of the sample to be able to capture its structure.

To capture better the textural patterns, **multi-resolution** is used. A multi-resolution image pyramid confines the structure at different resolutions, by fewer pixels in the high levels (i.e. low resolution). The influence of the image pyramids is associated to the neighbourhood size. The results obtained by using image pyramids and a small neighbourhood are similar to using no pyramid but a bigger neighbourhood.

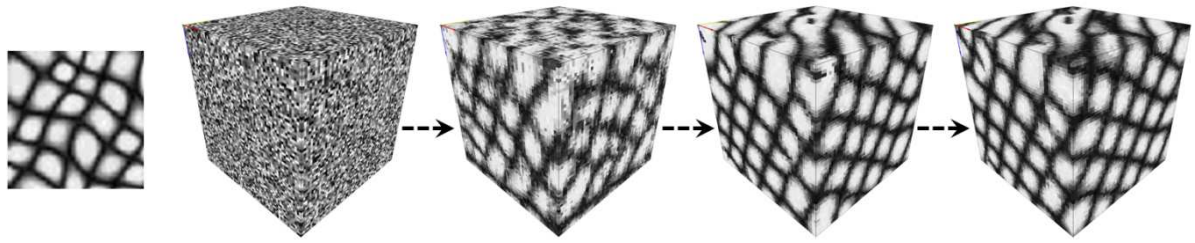
Fig. 5.4 shows the influence of the neighbourhood system on the synthesis results by varying the neighbourhood size. Higher the neighbourhood order, better are the results.



a *b* *c* *d* *e*

Figure 5.4 – *Neighbourhood size influence on the volumetric results using NP_K: from left to right, (a) the input image and next the solid textures obtained for a full-square neighbourhood of size (b) 5×5, (c) 7×7, (d) 9×9 and (e) 11×11 pixels.*

The result blocks showed in *Fig. 5.2* and *5.3* represent the 10th iteration of the synthesis process that starts from an initial random noise state. This noise is modified to look like the sample texture. The synthesis is stopped automatically after ten iterations, although after approximately six iterations the produced results are satisfactory. An example showing the evolution of the output texture during the **iterative process** is in *Fig. 5.5*. The output texture succeeds after several iterations to be alike the input texture.



a *b* *c* *d* *e*

Figure 5.5 – *Texture evolution during synthesis iterations: the output block, starting from (b) random noise, is iteratively modified to resemble (a) the 2D sample; (c) is the block obtained after the 2nd iteration, (d) shows the 6th iterations while (e) contains the 10th iteration.*

A known characteristic of these algorithms based on best-neighbourhood-match searching is that they are sub-optimal, being deterministic algorithms and leading to synthesis results that suffer from repetitive patterns and from more ordered features than the sample [*Che10*] [*Urs12*]. Hence this makes the algorithms be very vulnerable to the neighbourhood system and the number of iterations. Iterating much more tends to produce repetitive, over regular structures, whereas using very large neighbourhoods has the same orderliness effect. A compromise has to be made in order to use a sufficiently large neighbourhood to capture the essential features but also to iterate enough to assure the convergence.

A similar question is raised for the **scanning type**, which is the visiting order of the voxels during synthesis. Replacing the lexicographical scan type (deterministic scan-line order) with a completely random walk allows the synthesis of a pixel by freeing itself from its past and so multiplying the possible configurations. However, the convergence time can become prohibitive in this case. Attempts were made to reconcile the deterministic part and the randomness character by using alternative scan types – namely the use of space filling curves (Z-curve and Hilbert curve) (see *Appendix B*) and by using a mixed initialisation for the output block – white noise and patches from the input image. *Fig. 5.6* shows some results obtained by using different scanning types, namely the ones suggested in *Fig. 3.21*.

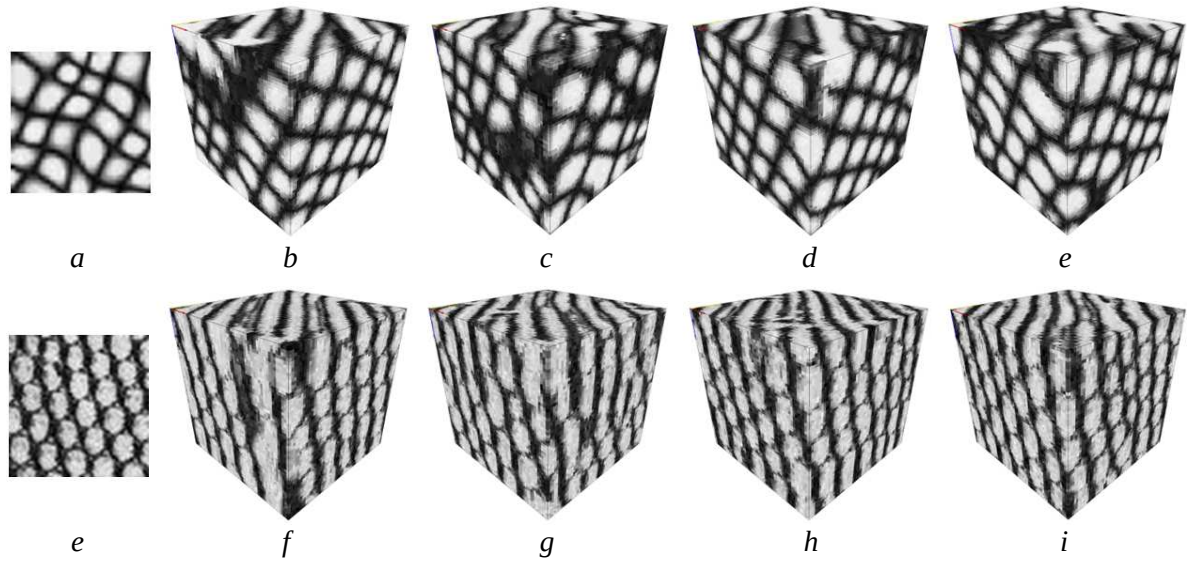


Figure 5.6 – *Texture synthesis by varying the scanning type: each row shows the 2D sample in (a) and the 10th iteration of the synthetic solid textures obtained with NP_K and the common framework using (b) lexicographical scan, (c) random walk, (d) 3D extended Z-curve and (e) three-dimensional Hilbert curve.*

The results in *Fig. 5.6* are relatively little sensitive to the scanning type. However some remarks have to be made.

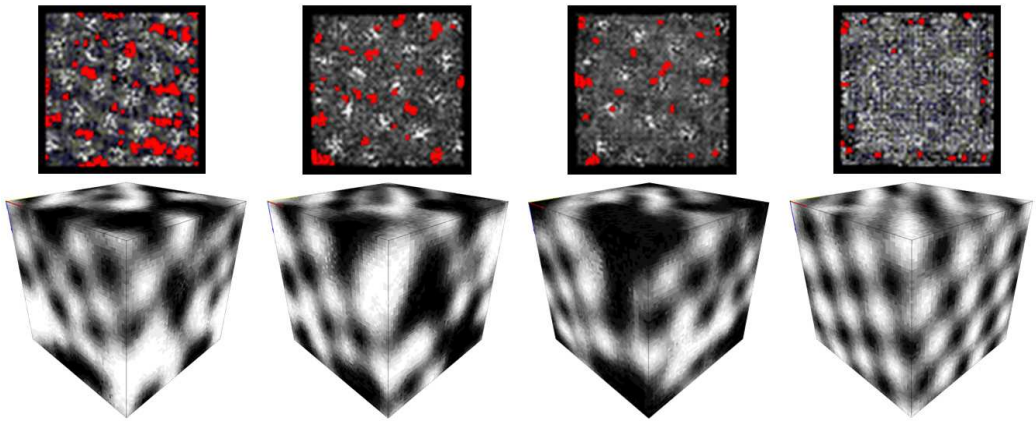
The lexicographical scan based result (*Fig.5.6b*) shows some undesired features because if an error pixel is to take place, it is likely to propagate from one neighbourhood to another retrieving it in the final result. It adds up the repetition of patches whether if they are good or bad. Additionally the lexicographical scan tendency is to regularize the texture. These aspects are better retrieved on the first row images in *Fig. 5.6* for which the sample is less regular. The left view of the 3D texture block in *Fig.5.6b* shows the failure of the algorithm to preserve the structure while its right view is quasi-periodic. As for the other results, they are less regular (as the 2D example).

The random scan assures diversity, makes the repetitive structures disappear but shows difficulties in converging.

The 3D space filling curves used as synthesis path produce satisfying results, not regularizing them too much and in the same time not randomizing them too much, in terms of structure. However the result obtained with the Z-curve path in *Fig. 5.6c* seems to be too regular also. The Hilbert-curve based result is less regular, more alike the initial texture, even if certain patterns don't correspond with the sample.

Apart from these conventional characteristics related to the neighbourhood systems, which have been more or less noted by the original paper authors, a parameter specific only to NP_CW has to be at least referred to – the **set of candidates**. The use of the set of candidates in the context of the position and index histogram matching and the discrete solver described in *section 3.4.2*, seeks to make sure that all the pixels are copied equiprobably from the 2D sample.

Fig. 5.7 illustrates how the synthesis results are affected by the size of the set of candidates, showing alongside the corresponding histogram position map. Larger the set, as better are the results and as uniform is the histogram position map. However these are accomplished with an increased computational cost.



a *b* *c* *d*

Figure 5.7 – *Influence of the set of candidates on the synthesis results: from left to right, the blocks are obtained by using NP_CW with 1, 5, 15 and 25 candidates. On top of each block is represented the position histogram map. In red are the areas containing input pixels that are not retrieved in the result. For the others, the whiter the pixels, the more they are used in the output texture.*

The synthetic results obtained with the neighbourhood search based non-parametric approaches have a satisfactory visual quality, showing the algorithms capacity to produce volumetric textures similar to the 2D samples in terms of dynamics and structure. Good results are obtained by choosing properly the synthesis process parameters.

Being emblematic for the non-parametric synthesis based on fixed-neighbourhood search, NP_K is going to be used as comparative method to the maximum-likelihood based approach in *section 5.4*.

5.3 Maximum-likelihood based synthesis results

This section is dedicated to the results obtained by applying the synthesis algorithms based on the maximum-likelihood estimation (NP_ML_H1a/b/c and NP_ML_H2a/b/c) described in *Chapter 4*, on 2D sample textures.

Firstly, some results are showed using the parameters for which eye-friendly results are obtained, and next the choice of these parameters is motivated by analyzing their influence on the synthetic solid textures.

5.3.1 Comparing the heuristics

The several heuristics proposed in *Chapter 4* are analyzed in the context of synthesizing the textures from *Fig. 5.1*.

To remind these heuristics and to have them at hand in this study, here is their brief description:

Heuristic tag	Heuristic decision to find the best λ_s
NP_ML_H1a	$\min\left(\arg\max(LCPDF_{front}), \arg\max(LCPDF_{right})\right)$
NP_ML_H1b	$\max\left(\arg\max(LCPDF_{front}), \arg\max(LCPDF_{right})\right)$
NP_ML_H1c	$\frac{\arg\max(LCPDF_{front}) + \arg\max(LCPDF_{right})}{2}$
NP_ML_H2a	$\arg\max\left\{\min(LCPDF_{front}, LCPDF_{right})\right\}$
NP_ML_H2b	$\arg\max\left\{\max(LCPDF_{front}, LCPDF_{right})\right\}$
NP_ML_H2c	$\arg\max\left\{LCPDF_{front} * LCPDF_{right}\right\}$

Table 5.2 – *Recap the heuristics involved by the non-parametric synthesis algorithms based on maximum-likelihood criterion, used for finding the most probable grey level for the output voxel by analyzing only two orthogonal views.*

To show the ability capacity of the proposed algorithms to synthesize various textures, the first results are obtained by using a generic framework that experimentally proved itself to produce satisfying results: a neighbourhood system composed of a full-square neighbourhood of size 7×7, computed on 3 scales, synthesizing the voxels in a random way, 8/8 up-sampling strategy, output block initialized with white noise and random parts bits from the input sample and re-synthesizing the even pixels inherited from a higher scale (as suggested in *section 4.3.5.2*).

To assure a reasonable computational time and to maintain a satisfying quality of the results, the temperatures of pixels is decreased with a factor -3 (i.e. ξ in $eq.4.9$) and stopping the synthesis process after reaching a 95% decrease of the global temperature. The results are showed analogously for each employed heuristic. The input images are of size 64×64 pixels and the synthesized blocks 64×64×64 pixels.

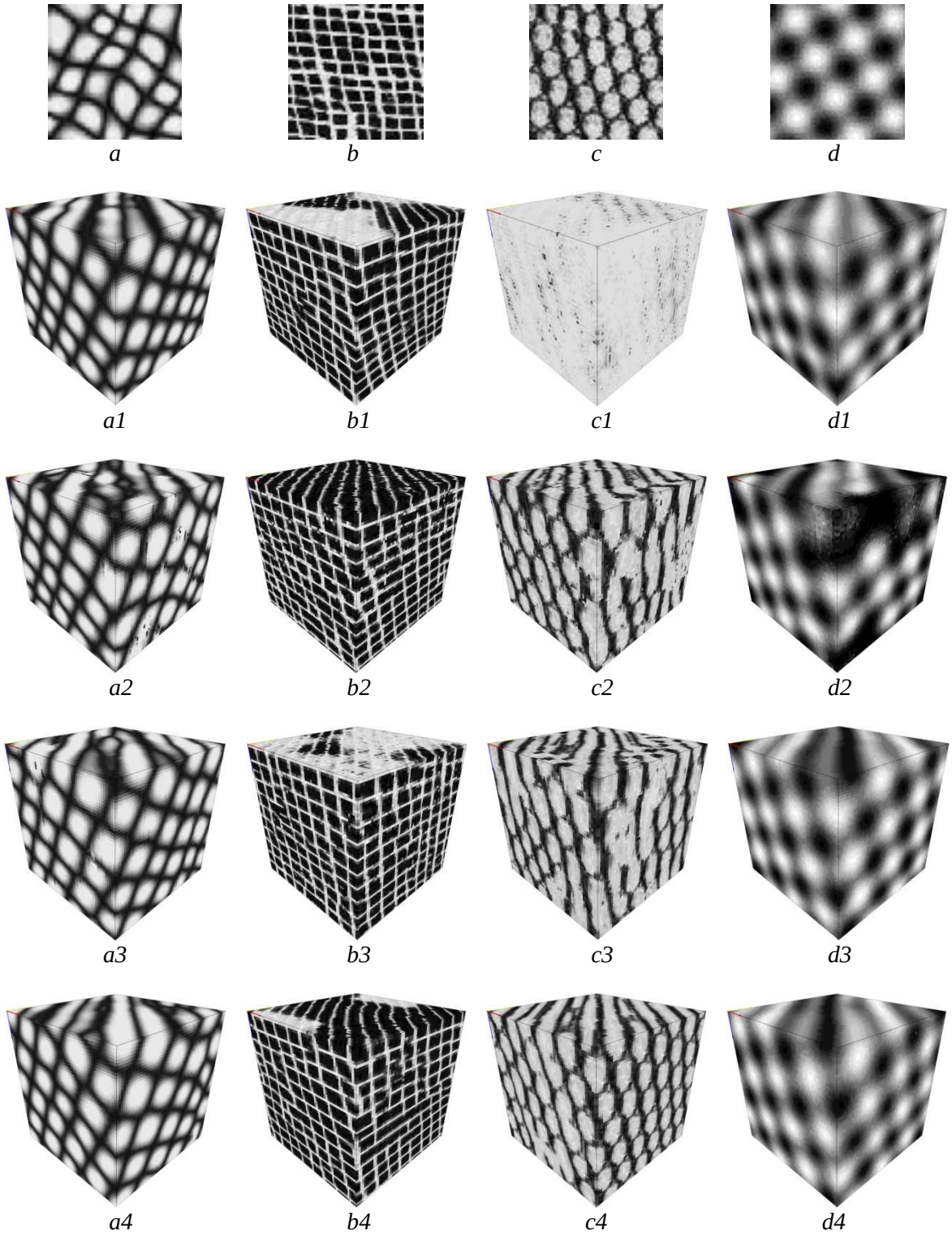


Figure 5.8 – *Volumetric results using the proposed heuristics: each column shows from top to bottom the results obtained by synthesizing the 1st row texture, using NP_ML_H1a heuristic (the 2nd row), using NP_ML_H1c heuristic (the 3rd row), using NP_ML_H2a heuristic (the 4th row) and NP_ML_H2c heuristic (the 5th row).*

Fig. 5.8 shows some results obtained by using the maximum-likelihood heuristics proposed in *Chapter 4* on the basis of a 7×7 full-square neighbourhood.

More precisely, the second row (*Fig 5.8a1-d1*) presents the solid textures obtained by applying the NP_ML_H1a heuristic, meaning that the update value for an output voxel is found by choosing the grey level that corresponds to the smallest from the two LCPDF computed separately for the two orthogonal views taken into consideration (front view and right view). The idea is to choose the smallest LCPDF in order to be valid for each of the two views. It works well except the result in *Fig. 5.8c1* for which the algorithm doesn't captures at all the structure of the texture from *Fig. 5.8c*. This heuristic performs well, but not strong enough to capture the interactions between the pixels, a possible solution being to increase the neighbourhood size.

Using the NP_ML_H1c heuristic (3rd row in *Fig. 5.8*), the update voxel value is computed as the average of the two grey levels, each one of them maximizing the LCPDF of his corresponding view. As in the case of the neighbourhood-search based approaches the averaging operation is not the winning strategy: the voxels are faced with grey-levels that don't exist in the 2D sample, disrupting the image-based synthesis process. Undesired artefacts are seen on almost all the results, providing broken textures (as is the case of textures in *Fig. 5.8a2–d2*).

The last two rows of *Fig. 5.8* contain the results obtained by using for the update voxel value the grey-level that maximizes a function of the 2D estimations of the two orthogonal views. The heuristic used in *Fig. 5.8a3–d3* is NP_ML_H2a maximizing the smallest of the two likelihoods in order to maximize both of them. The results are promising in term of structure conservation, but undesired artefacts still come into sight (visible on the solid textures in *Fig. 5.8a3* and *c3*). The function of the LCPDF estimation is not able to capture the pixels correlations under the employed framework, unsuitable for characterizing all the orthogonal views.

It looks like the NP_ML_H2c heuristics (the last row in *Fig. 5.8*) runs better, being able to capture the visual characteristics of the input texture by maximizing the product function of the LCPDFs of each view. This product based heuristic is stronger (in terms of capturing pixel interactions) than all the other ones even at a relative small neighbourhood size. Using this heuristic as a decisional measure for the output voxel grey value, the synthesis provides more than satisfying results. The synthetic solid textures (*Fig. 5.8a4-d4*) are smooth almost artefact–free textures (except some small structure defects), looking very similar to the sample textures (*Fig. 5.8a-d*) giving the impression that they were produced by the same process.

Separately from the above analysis, are treated the other two remaining heuristics. NP_ML_H1b is the one based on choosing the update value as the one that corresponds to the biggest LCPDF computed individually for each of the two orthogonal views. Results using this heuristic are showed in *Fig. 5.9*. NP_ML_H2b maximizes the biggest likelihood function, favouring only one of two views; results based on this decisional heuristic are in *Fig. 5.10*. The reason why they are compared in this place is that their corresponding results are similar, but not in a good way. Using the common framework with 7×7 neighbourhood proved to be insufficient for them providing catastrophic results (second row of *Fig 5.9* and *Fig. 5.10*). So a larger neighbourhood, an 11×11 one, was used. In this context the structure of the sample texture is captured but only for an orthogonal view (third row of *Fig 5.9* and *Fig. 5.10*). These heuristics are incapable of convincing all the orthogonal views to follow the sample characteristics.

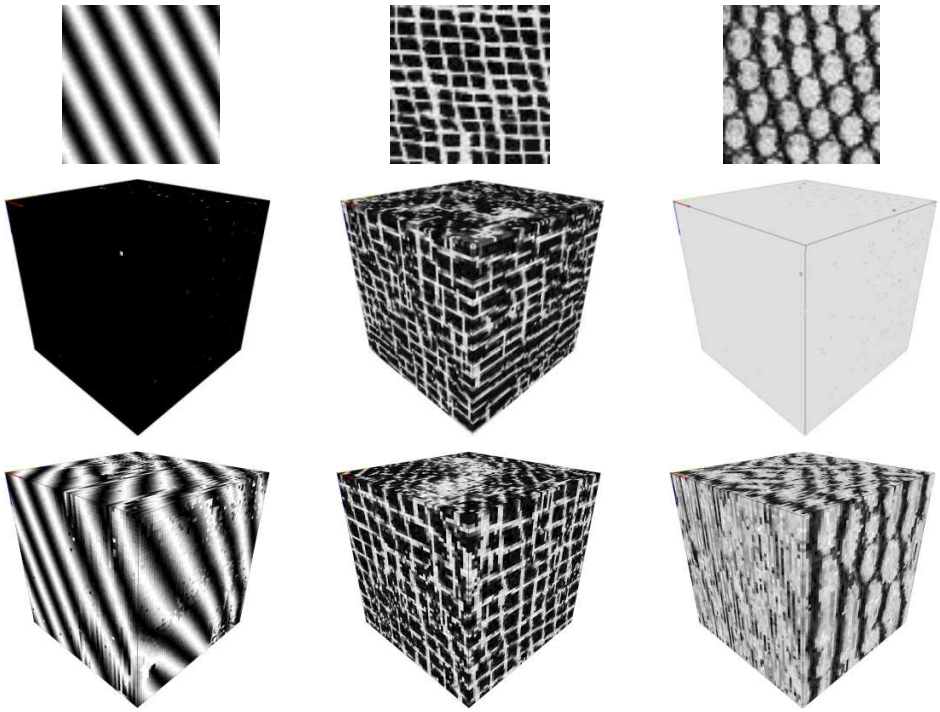


Figure 5.9 – *Volumetric results corresponding to the NP_ML_H1b heuristic: from left to right, each column shows from top to bottom, the sample texture (1st row), the output block computed with a 7×7 neighbourhood (2nd row) and the output block computed with an 11×11 neighbourhood (3rd row).*

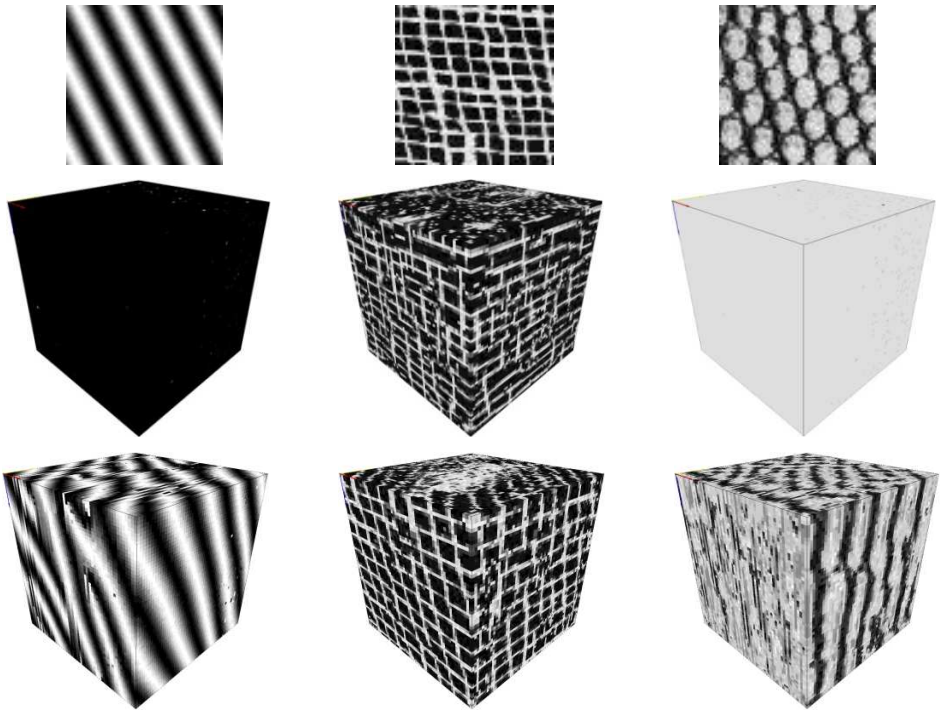


Figure 5.10 – *Volumetric results corresponding to the NP_ML_H2b heuristic: from left to right, each column shows from top to bottom, the sample texture (1st row), the output block computed with a 7×7 neighbourhood (2nd row) and the output block computed with an 11×11 neighbourhood (3rd row).*

To conclude this section, it seems that the results based on NP_ML_H2c heuristic are the most convincing ones. The above remark justifies the choice of using NP_ML_H2c for the next operations as being representative for the maximum-likelihood based approaches.

5.3.2 Temperature decreasing schema

The developed algorithms based on the maximum-likelihood estimation integrate, in the synthesis/relaxation process, a pixel temperature function as a degree of confidence for a pixel grey-value. It is the main factor that influences the duration of the relaxation process, determining the 'annealing', that is the control of the cooling rate. The process starts at high temperatures, and iteratively it is relaxed with the decreasing of the temperature. Two aspects drew our attention – the speed of the **temperature dropping-off** and the **relaxation duration**.

The pace of the cooling schedule is determined by the parameter ξ as described in *eq. 4.9*. A small value assures a good convergence but implies long relaxation sequences while a higher value allows a faster process but not necessarily with satisfying results. A pixel temperature is cooled-down based on its neighbours' temperatures. A fast decrease of temperature is not capable of capturing the spatial correlations between neighbours. The cooling rate influences directly the number of iterations by reducing faster the global temperature.

Strongly related to the cooling rate is the time taken by the relaxation process. The initial scheme consisted in starting from a high global temperature and decreasing it, successively after handling each voxel, until reaching zero when all pixels achieve complete confidence. But an adequate amount of confidence attained without reaching the zero level can produce satisfying results, suggesting that a very small, but positive value of a pixel temperature could be enough. This implies that the relaxation process can be stopped at an earlier time, after accomplishing a certain percentage from the initial global temperature.

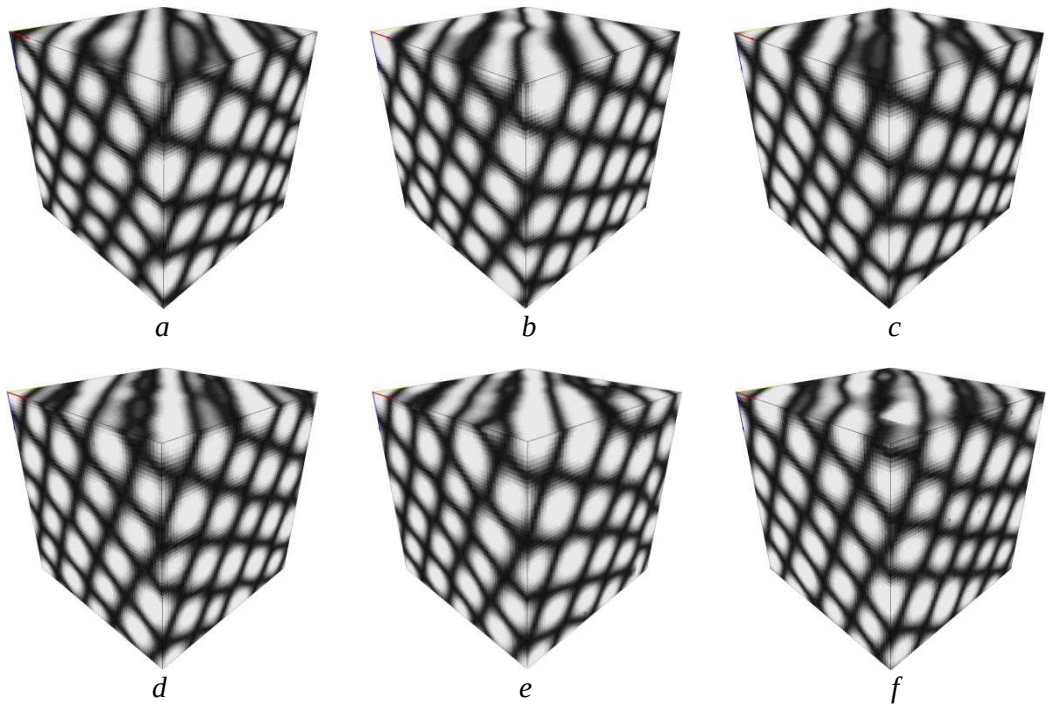
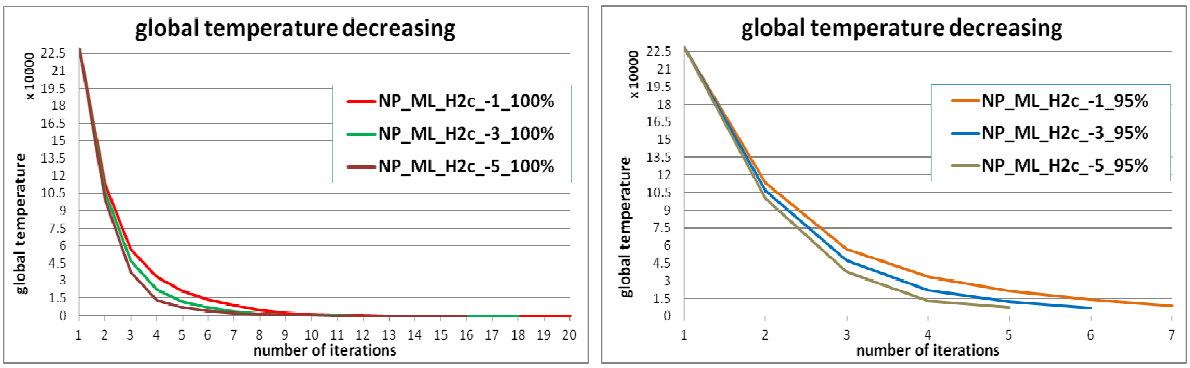


Figure 5.11 – *Results obtained for different temperature decreasing strategies: the first row shows the results obtained by accomplishing 100% of the relaxation process, but by decreasing the pixel temperature with (a) ξ=-1, (b) ξ=-3 and (c) ξ=-5; the second row shows in the same order for the values of ξ the results obtained by stopping the synthesis process after reaching 95% from the initial global temperature.*

In order to obtain satisfactory results with a reasonable computational cost, a compromise has to be made regarding the rate of the cooling schedule and the relaxation time. Some results obtained with different values for ξ while leaving the relaxation process follow its course or by stopping it after reaching 95% from the initial global temperature at each scale, are presented in *Fig. 5.11*. The results are visually very similar, only the block in *Fig. 5.11f* shows a few undesired error pixels, because of the too fast decreasing schema (ξ*=-5*) and of the too early stopped relaxation process (95%).



a *b*

Figure 5.12 – *Plots showing how the global temperature is dropping off with the parameter ξ at -1, -3 and -5 when allowing the algorithm follow its course until reaching zero temperature (plot in a) and when fulfilling only 95% of the initial global temperature (plot in b).*

The influence of these factors is analyzed in the plots from *Fig. 5.12*. For the above presented solid results, the impact of ξ is reflected in a more accelerated decrease of the global temperature; a more important ξ (a smaller value) allows reaching a same global temperature level in fewer iterations than when using a bigger ξ. Stopping the synthesis process at 95% from the initial global temperature, reduces the number of iterations, from 20 to 7 in the case of ξ=-1. For ξ=-3 the number of iterations is reduced from 18 to 6, while for ξ=-5 from 16 to 5. *Fig. 5.13* shows the evolution of a solid texture from an iteration to another until reaching the final state presented in *Fig. 5.11b*. It shows the iterations involved by the relaxation process on the final pyramidal level, until reaching a stable state (deemed at 95% of the global temperature).

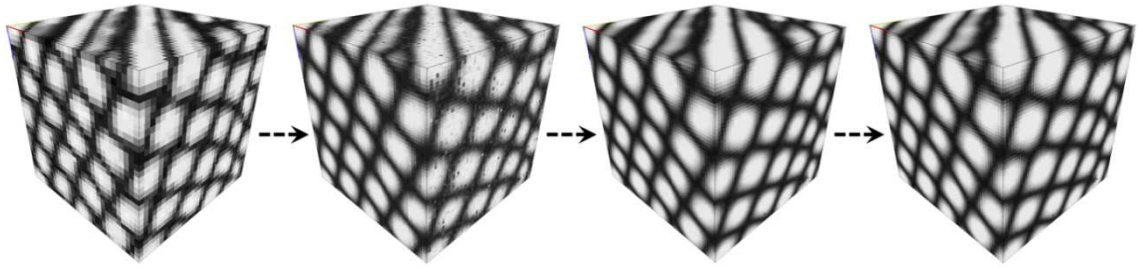


Figure 5.13 – *Time evolution of the solid texture at the last pyramidal level: first four iterations showing the temperature decrease involved by the relaxation process plotted in Fig. 5.12b in the case of NP_ML_H2c_-3_95%.*

The above section analyzed the effect of the two key factors in pixel temperature decreasing schema, on the 3D synthesis results. It shows that reducing the number of iterations and accelerating the decrease of temperature can still produce satisfactory results with the pair of parameters ξ=-3 and 95% of the global temperature stop criterion. To be able

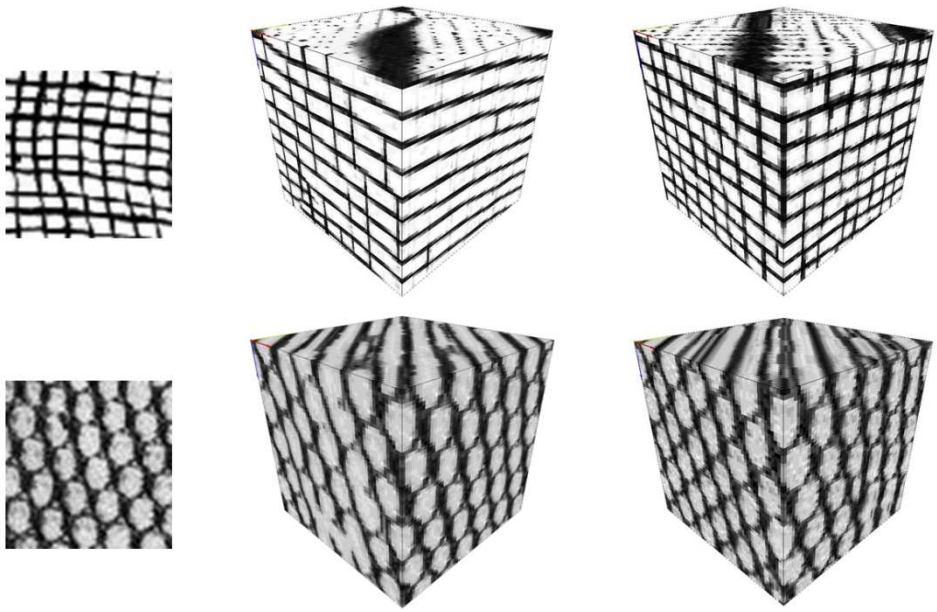
to produce a large number of results, this will be the synthesis strategy used further on in this chapter under the emblematic heuristic – NP_ML_H2c.

5.3.3 Common parameters influence

The maximum-likelihood based synthesis process is sensitive to the **neighbourhood size** as well as the neighbourhood-search based approaches. The neighbourhood size has to be indeed large enough to capture the biggest pattern of the input texture, in order to assure a result similar in structure to the sample capturing the spatial interactions between neighbouring pixels.

This is correlated to the resolution of the sample texture - a higher resolution (meaning more well defined inner patterns) of an image requires a higher neighbourhood size. The question of the **number of pyramidal levels** is also related to this issue, each level being in fact representations of the same image at different resolutions.

Some examples illustrating how the results can be improved by using a bigger neighbourhood are showed in *Fig. 5.14*. A larger neighbourhood captures better the input structure giving a higher quality to the solid textures making it more similar to the sample.

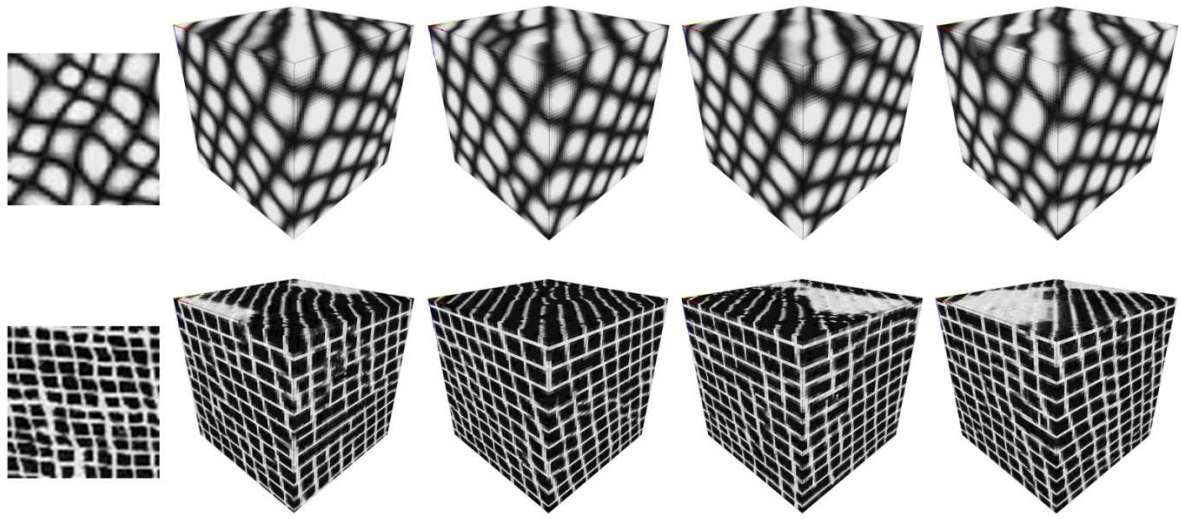


a *b* *c*

Figure 5.14 – *The influence of the neighbourhood size: each of the two rows shows, from left to right, in (a) the sample texture, in (b) the output block obtained with a 7×7 neighbourhood and (c) contains the improved results obtained by using a 9×9 neighbourhood. The other parameters were the same:3 pyramidal levels, 8/8 up-sampling strategy, ξ=-3 and stop at 95% of the process, reiterating the even pixels and initializing the output block with input image patches.*

Other two parameters to be taken into considerations are the **up-sampling strategy** and the **initialisation strategy**. The pass of information from one synthesized level to the next untreated one consists in copying each treated level pixel into one or eight of the eight next level pixels (named 1/8 or 8/8 upsampling strategy in *Chapter 4.3.5.2*). The purpose is to assure faster and better convergence by using for synthesis only already treated pixels. With the same ambitions for the convergence and additionnaly to help the synthesis capture better the input image patterns, an alternative strategy to the random initialisation of the output block is proposed. It consists in integrating in the orthogonal slices of the output block, filled

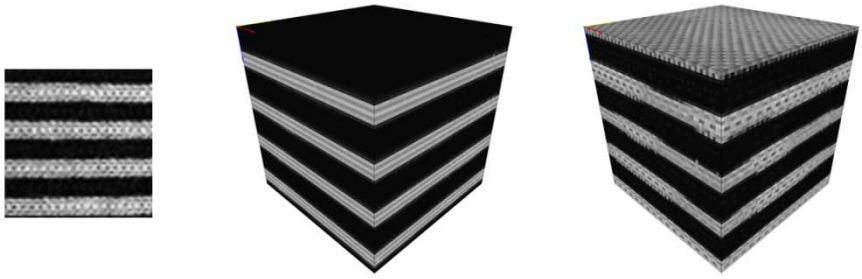
initialy with white noise, patches from the input image. For simplicity, the variant that consists in initializing with input image patches is named as *+in*, while the simple random noise initialization is named as *–in*.



a *b* *c* *d* *e*

Figure 5.15 – *The trace of the up-sampling (1/8 or 8/8) and the initialisation strategy (+in, -in) on the synthesis results: each of the two rows shows, from left to right, (a) the input sample, (b) the results obtained with 8/8 up-sampling and +in, (c) 8/8 up-sampling and –in, (d) 1/8 up-sampling and +in and the last column (e)1/8 up-sampling and –in; all the other parameters were fixed.*

Some examples affected by the variation of the upscaling and the initialization strategy are shown in *Fig. 5.15*. The deduction is that apparently the synthethic solid textures show no major differences when using a certain up-sampling schema and a certain initialisation strategy.

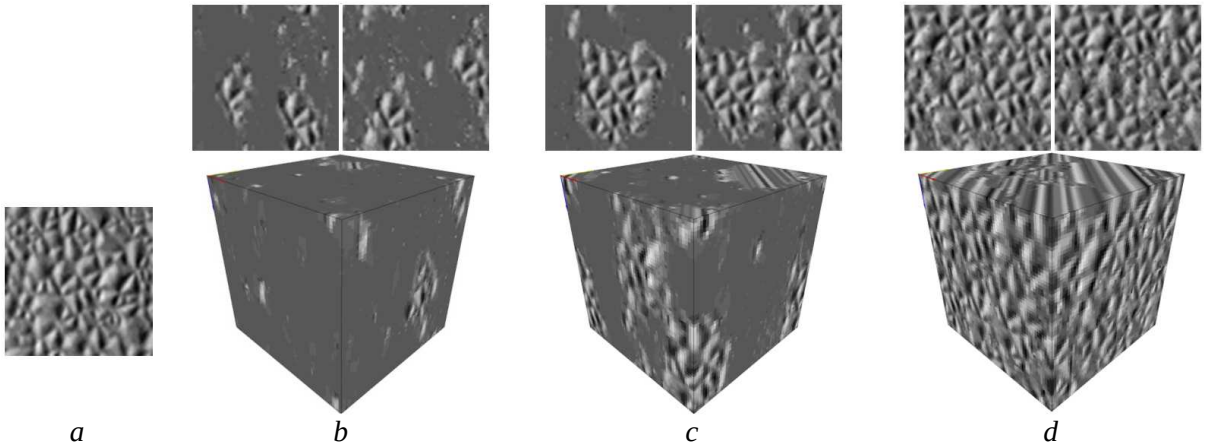


a *b* *c*

Figure 5.16 – *The trace of the up-sampling strategy on a synthesis result: the block in (b) is obtained by using 8/8 up-sampling, while the volumetric result in (c) is obtained with 1/8 up-sampling strategy capable of capturing the structure and the pattern of the sample texture in (a); all the other parameters were fixed.*

But the assumption relative to the triviality of the upsampling schema is rapidly demolished by the result presented in *Fig. 5.16*. In some case the 8/8 up-sampling schema is not capable of assuring a required degree of diversity, reproducing from one level to another the same grey levels obtained at the lowest resolution level. If the lowest resolution level is inappropiate (i.e. structure misplaced, uniform grey-levels) the synthesis process captures the mode of distribution and propagates it through all the scales. To assure disparities while passing from one scale to another, a 1/8 up-sampling strategy is used, paying attention to the information from the already synthesized scale and to a certain randomness induced by the random pixels issued from the initialisation strategy. *Fig. 5.17*, as *Fig. 5.16*, shows how the

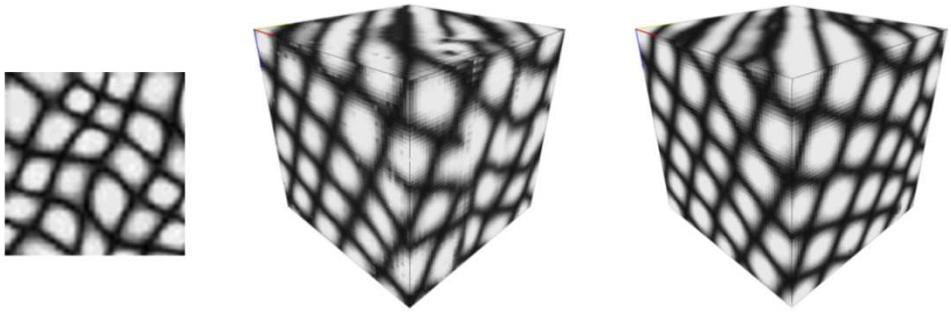
1/8 up-sampling improves the quality of the results but in addition it proves again the vulnerability of the maximum-likelihood based process faced to the neighbourhood system, that needs a bigger neighbourhood to capture the sample structure and to reproduce it in the synthetic results.



a *b* *c* *d*

Figure 5.17 – *The mutual influence of the neighbourhood size and the up-sampling strategy on synthesizing the 2D sample in (a): the result in (b) is obtained using a 9×9 neighbourhood and a 8/8 up-sampling, the result in (c) is obtained by using a 9×9 neighbourhood size but with a 1/8 up-sampling and the result in (d) is obtained with a 11× 11 neighbourhood corroborated with 1/8 up-sampling strategy; above each block are represented two 2D cuts from the 3D block illustrating the synthesis success rate.*

Of particular interest is questioning if the temperature of a pixel from an already synthesized scale has to be or not processed in order to initialize the next scale (according to the upsampling strategy). As illustrated in *Fig. 4.14* from *Chapter 4*, the solution adopted here is to re-synthesize the voxels having even coordinates, i.e. the voxels inherited from the higher scale. Using a 8/8 upsampling and not resynthesizing the even pixels it is not capable of eliminating the errorneous pixels, while with re-synthesis the results are smoother. This even pixels revisiting operation is highly important in producing high quality results. Its effects are evidently positive, results being shown in *Fig 4.15* and in *Fig. 5.18.* Without synthesizing the inherited pixels (as proposed in [*Pag98*]), errorneous pixels appear at different scales and propagate untill the final level.



a *b* *c*

Figure 5.18 – *Example of two volumetric synthetic blocks obtained from (a) - a 2D image, by applying the multi-scale relaxation MRF synthesis method using the heuristic NP_ML_H2c : the block in (c) is obtained by re-synthesizing the even voxels updated from the upper scale, visually improving the result from (b) that was obtained without reconsidering the even voxels.*

5.4 Fixed-neighbourhood search vs. likelihood maximisation

This section is aimed to be a qualitative comparative operation showing in parallel the results obtained with the neighbourhood-search based synthesis approaches and results of the maximum-likelihood based methods.

To ease the comparative study only two representative approaches, one for each class, are used. These are NP_K and NP_ML_H2c, as it was reasoned in the previous sections of this chapter.

Before starting the effective study, a first remark has to be noted on the already shown results – it appears that NP_ML_H2c provides better results than NP_K at a same relatively small neighbourhood size. NP_ML_H2c is capable of providing satisfying results using a 7×7 neighbourhood as illustrated in *Fig. 5.8*. With the same neighbourhood size and using the same 2D samples as source of synthesis, NP_K is not capable to produce solid textures similar in structure to the 2D samples, as illustrated in *Fig. 5.3*. But these were particular examples, so a larger experimentation basis is required to be able to draw conclusions.

To assure a common framework a suitable 9×9 neighbourhood size is used, three pyramidal levels, random scanning, *+in* initialisation. For NP_ML_H2c an 1/8 up-sampling and even pixels revisiting strategy are adopted. Because the deterministic relaxation algorithm involved by the maximum-likelihood based synthesis is computationally very expensive, performing LCPDF estimations for every grey-level available in the input image, the 2D sample textures are of 32 grey-levels.

Synthesis results using the previous framework are shown in *Fig. 5.19* and *Fig. 5.20* placing jointly the NP_K 3D texture and the NP_ML_H2c 3D texture.

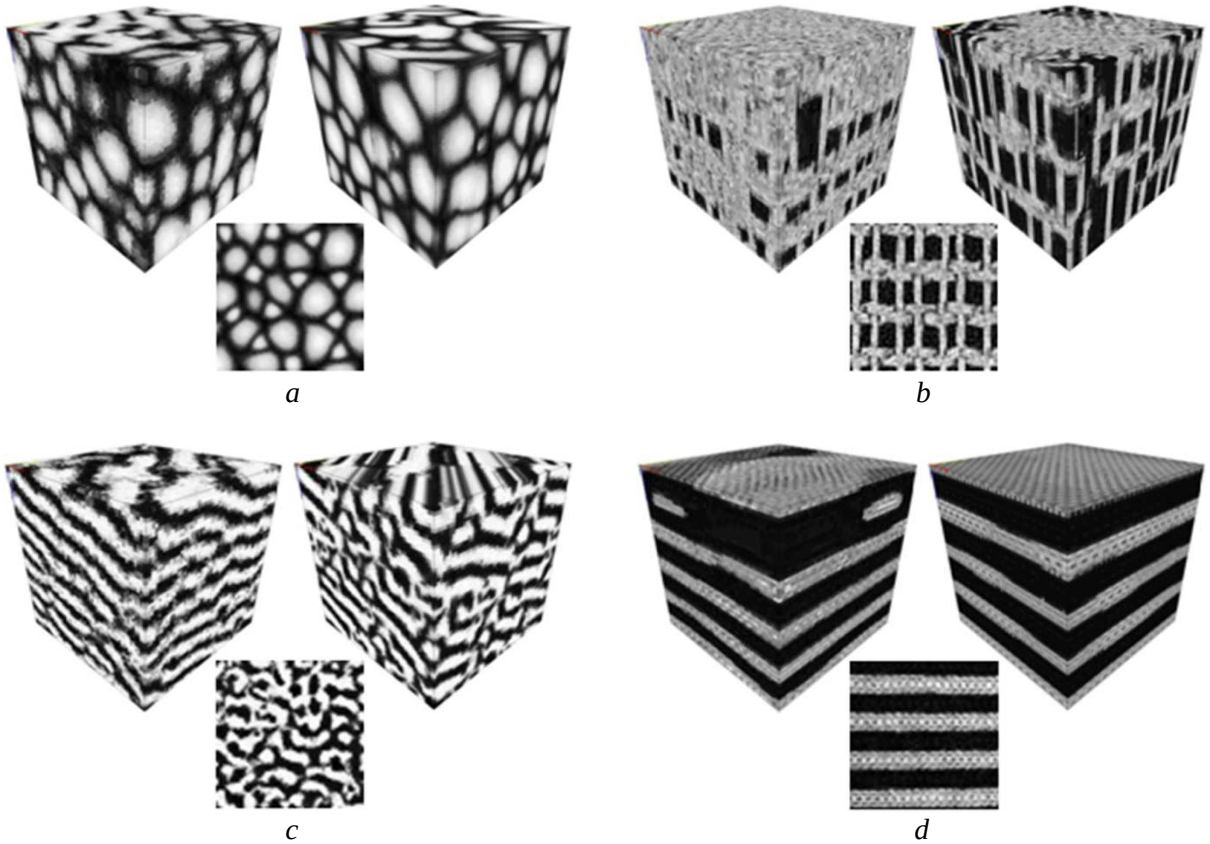


Figure 5.19 – *Volumetric results: each of the four triplets contains in the middle the 2D sample, on the left the 3D texture obtained using NP_K approach on a 9×9 neighbourhood and on the right the 3D block obtained using NP_ML_H2c heuristic using the same neighbourhood size.*

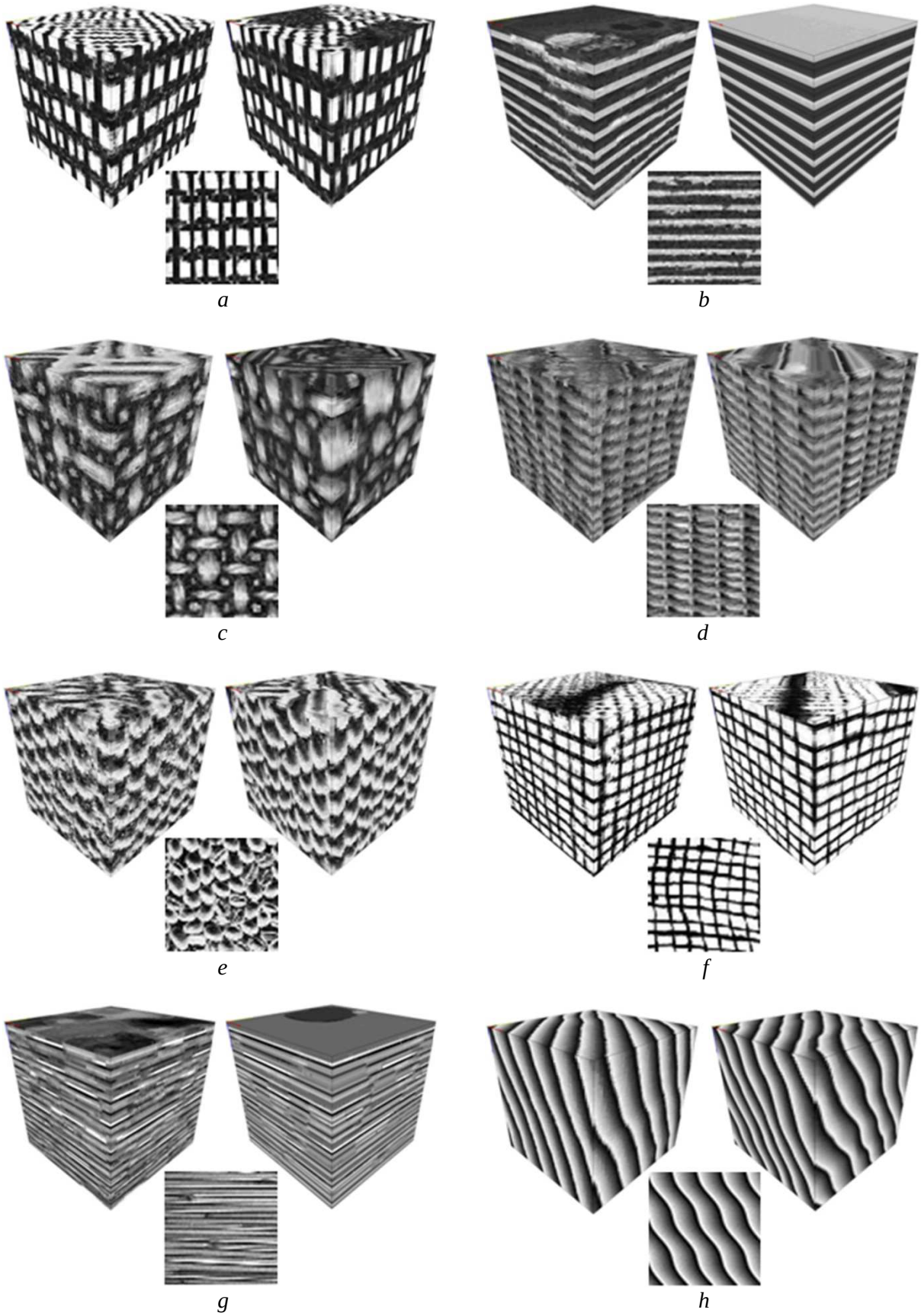


Figure 5.20 – *Volumetric results: each of the eight triplets contains in the middle the 2D sample, on the left the 3D texture obtained using NP_K on a 9×9 neighbourhood and on the right the 3D block obtained using NP_ML_H2c heuristic using the same neighbourhood size.*

For the textures shown in *Fig. 5.19*, the NP_ML_H2c results seem to be better than the ones obtained with the NP_K approach mostly in terms of structure preservation. The textures in *Fig. 5.19a* and *5.19b* obtained with NP_K capture deficiently the structure of the sample while the NP_K block in *5.19c* is more regular than the sample texture. The last case, the texture in *Fig. 5.19d*, shows the NP_K difficulties in sustaining the structure. *Fig. 5.20* carries on the results illustration testifying overall the promising quality of the results both for NP_K and NP_ML_H2c. It seems that NP_ML_H2c produces a better result than NP_K for the case in *Fig. 5.20e* but this is annulled by the results in *Fig. 5.20b* where NP_ML_H2c is inferior, making simpler the 3D texture.

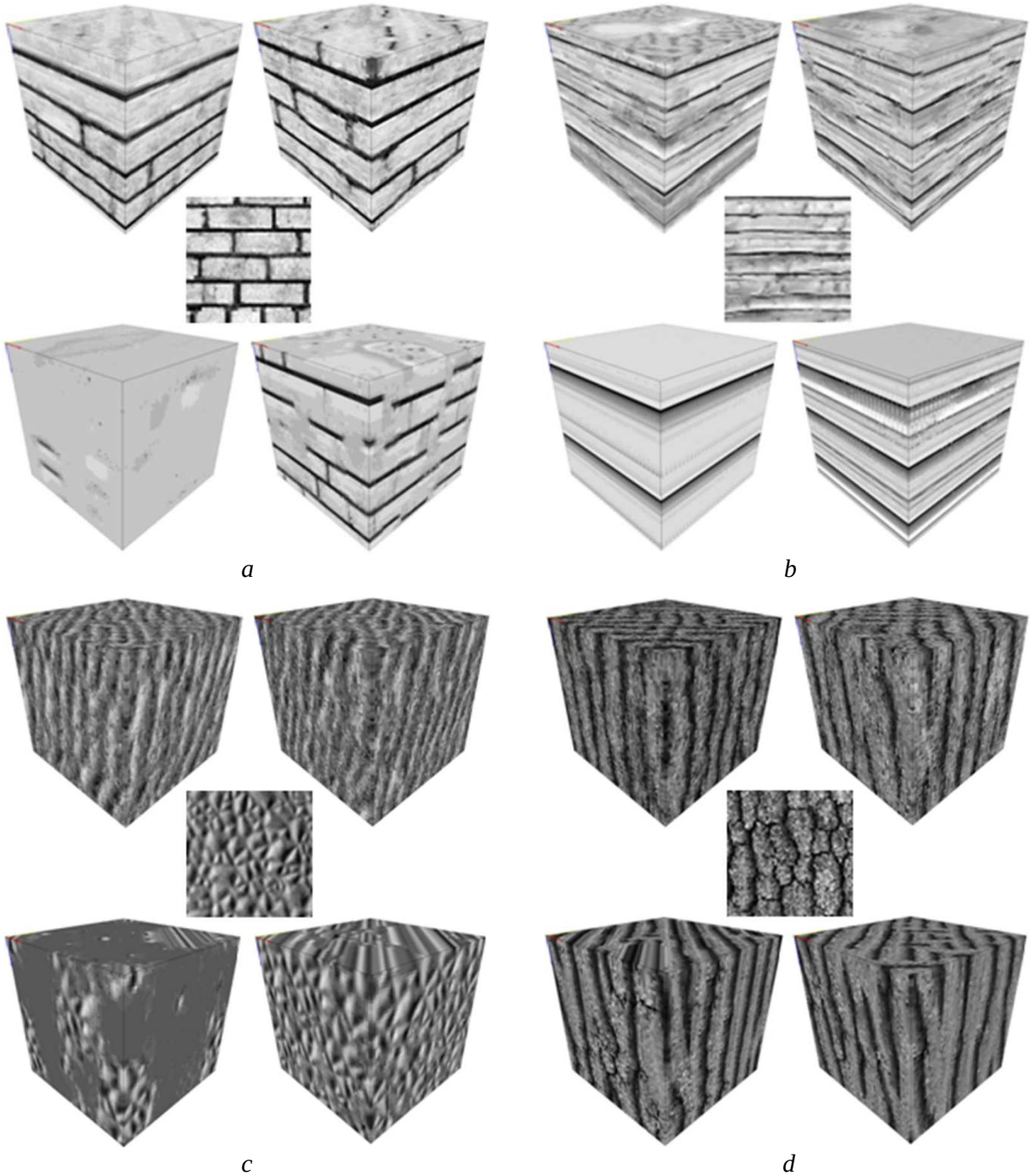
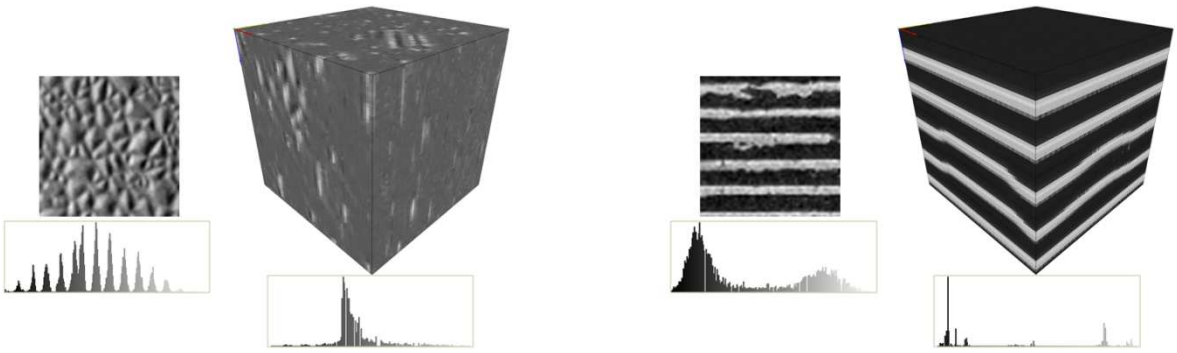


Figure 5.21 – *Volumetric results: each of the four quintuples contains in the centre the 2D sample, above the sample two NP_K results obtained, from left to right, with an 9×9 and an 11× 11 neighbourhood and below the 2D sample the two NP_ML_H2c results obtained by using, from left to right, an 9×9 and an 11× 11 neighbourhood size.*

It's not always that NP_ML_H2c behaves better. The results in *Fig. 5.21a* and *5.21b* prove exactly the opposite; even if NP_K needs a bigger neighbourhood to provide satisfactory results, it behaves better than NP_ML_H2c. A bigger neighbourhood is required likewise by NP_ML_H2c for the synthesis in *Fig. 5.21c*, being the only one capable of capturing the sample pattern. The results in *Fig. 5.21d* are terrible for both of the approaches, incapable of preserving in the synthetic result the sample structure.

In a similar way, the number of grey-levels is questioned. Its influence on two synthetic textures is shown in *Fig. 5.22*. In this case the assumptions according to which the synthesis simplifies the textures, reducing the number of grey levels is once again confirmed, even if the synthesis takes place by using numerous grey-levels and not only the reduced 32 levels. Comparing the result in *Fig. 5.22a* with the one in *Fig. 5.21c* and the result in *Fig. 5.22b* with the one in *Fig.5.20b*, one can conclude that no improvement is brought in term of texture structure by using a larger diversity of grey-levels in the synthesis process. As well, a direct remark pops out by comparing the sample texture grey-level histogram with the synthetic block histogram – the synthesis process reduces the number of grey-levels maintaining only the most present (i.e. the most probable) grey-levels of the 2D sample, as a reminiscent effect of the deterministic ICM relaxation algorithm.



a *b*

Figure 5.22 – *Volumetric results obtained by applying the synthesis algorithm on sample textures having all their original grey-levels.*

The comparative study underlines the idea that the employed synthesis approaches can hardly be differentiated being unable to conclude precisely which one is the best, while showing that satisfactory results can be achieved by tuning properly the synthesis parameters. In fact, one approach is better than the other in function of the viewer/user/application centre of interest. The maximum likelihood based approaches comes up with extra parameters emerged from the pixel temperature function that sometimes prove to be difficult to control but, when properly tuned, can provide outstanding results.

From a computational cost point of view, the NP_ML_H2c approach needs long relaxation steps, searching for every output voxel the most probable grey level, each time (i.e. for each grey-level) a conditional probability being estimated. With the increase of the neighbourhood size and the number of available grey-levels, the deterministic relaxation duration increases as well, becoming an overwhelming process. If one searches for a fast way to generate textures, the fixed-neighbourhood search based approaches are of interest. For example, using a machine operating at 2.7 GHz to generate a 64×64×64 block from a 64×64 texture containing 32 grey-levels, using a 9×9 neighbourhood, 3 scales, pixel temperature decreasing with a -3 factor, stopping the synthesis process at 95% of the global temperature and re-considering for synthesis the inherited pixels, the time required by the NP_ML_H2c relaxation has an order of almost 24 hours, while 10 iterations of the NP_K based results are

obtained in only a few minutes. A future perspective development involves the algorithm acceleration, attainable by algorithm parallelisation.

5.5 About 2D/3D inference

Beside the algorithmic investigation and the results interpretation, our reflexion questions the truthfulness of the generated 3D texture given a 2D sample. The volumetric texture should indeed exhibit a plausible structure in accordance with the reality and each of its 2D slices should be consistent with the 2D sample. This can be simply stated as the 2D-3D inference objective.

The human visual system, based on prior experience, has nearly no difficulty in making possible a robust inference from insufficient data. In practice, people can figure out the most probable 3D structure among numerous possible structures given only a single 2D image. By contrast, the 3D inferring is quite challenging for computer vision systems.

Concerning the texture synthesis algorithms, generating a 3D volume from a 2D image is an under-constrained problem, with multiple solutions, not all of them equally possible. Consequently, the algorithms must be adapted to the input image structure to be able to 'invent' (i.e. infer) detailed 3D structure from a single 2D image and to create qualitatively accurate and visually pleasing results.

Depending on the 2D texture configuration (e.g. asymmetry, homogeneity, granularity etc.), the synthesis has to be constrained by two or three orthogonal views of the solid block. Ideally, one should dispose of samples for each view but, as it is not always the case, the synthesis becomes more awkward. Each slice of the 3D block corresponds to one 2D image that should resemble the sample and in the same time it should allow a certain 3D block's visual plausibility.

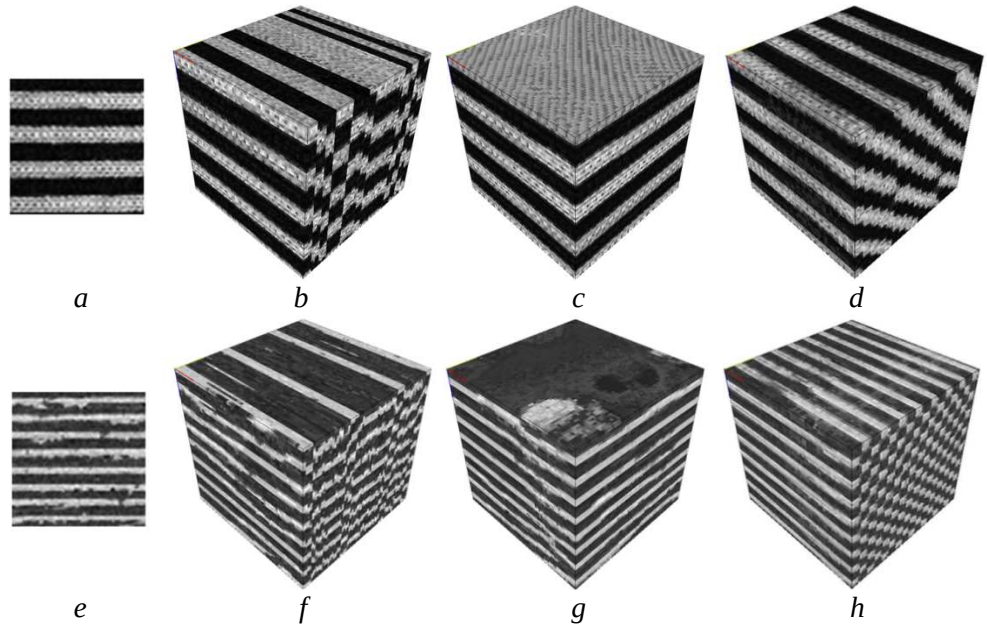


Figure 5.23 – *Results obtained with NP_K approach using a 11×11 neighbourhood: for each row going from left to right, the input image (a,e), the result obtained by constraining only the front view (b,f), the block obtained by constraining two orthogonal views – front and right (c,g) and the one obtained by constraining three orthogonal views (d,h). The results in b and f contain independent slices insufficient for creating a coherent 3D lamellar texture, while constraining three views we cannot reproduce the same texture on the three views. A plausible 3D structure is provided by constraining two orthogonal views with the same input sample.*

Constraining only a single view of the block, the slices become independent one from each other, leading to discrepancies between slices while the 3D block looks like a coarse, discontinuous representation. Disposing generally of only a single sample as source of synthesis, the same image is used to constrain the orthogonal views of the block, coercing the front view, side view or top view to match the sample.

The number of constrained orthogonal views depends on the sample texture structure. For example, for the anisotropic textures in *Fig. 5.1p-t* showing a lamellar organization, it is not possible to constrain three orthogonal views in the synthesis process by a unique 2D sample, as it does not provide enough accurate information to reproduce a plausible volume. To illustrate this point, we can try to synthesize such a texture constraining three views, but the results have no reasonable representation, the algorithm 'guessing' a wrong 3D structure, incompatible with the sample in terms of anisotropy. The results in *Fig. 5.23* show that even if for two directions of view (front view and top view) the texture seems convincing slice by slice, the algorithm fails in the other direction. The textures observed on the 3rd direction (side view) don't resemble at all the 2D sample, being in fact a compromise for the missing sample for that view. To be in accordance with the other two directions, the 3rd direction slices take a diagonal or checkerboard irrelevant arrangement. For these textures it makes sense to constrain only two orthogonal views as already shown in this chapter, knowing that their underlying process consisted in layers stacked one above the other.

Even when not disposing of the texture for the 3rd view, and using a same texture as constraint for two views, the synthesis process does a good job, indirectly inferring the 2D textural information at the 3D level. The textures in *Fig.5.1a-n* are susceptible to constraining two and three views, obtaining in both cases satisfying volumetric results in terms of resemblance between slices and the sample, while preserving a coherent 3D context. For the result in *Fig. 5.24* obtained by constraining two orthogonal views, the perceived patterns have an elongated tubular, cylinder shape. Instead, when using the same sample as constraint for three views, the patterns are bubble or cube shaped. Slicing the block in frontal, lateral and from atop, the obtained 2D textures look similar to the sample, the same remark being valid when slicing frontally and laterally the block obtained by constraining only 2 views.

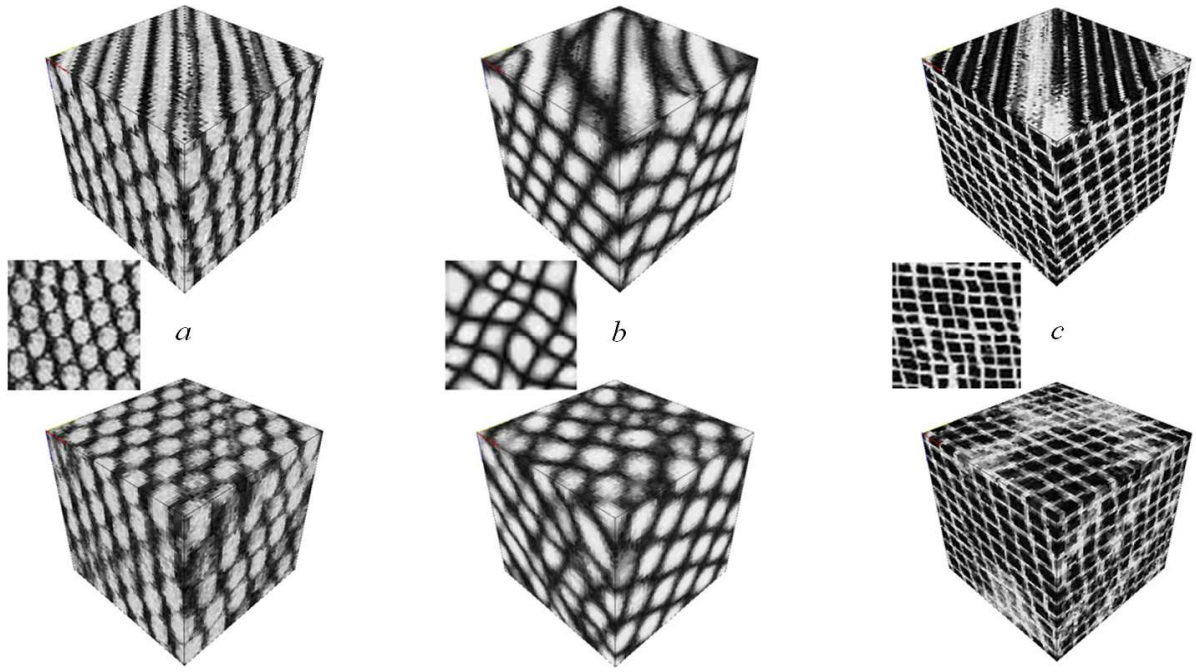


Figure 5.24 – *3D results obtained with NP_K by constraining 2 views (the block above the centre sample) and by constraining 3 views (the block below the sample).*

For textures containing macro-structures inside, the inference problem requires a more intricate analysis based on stereological observations [*Jag04*] [*Chi06*]. For this kind of particle based textures, it is advisable to use another type of approach. Precisely, it requires the identification and reconstruction of 3D particles, the distribution of patterns inside the 3D block and finally the placement and packing refinement (as sketched in the ending of *Chapter 2*). But the inference is once again questionable, because the processes are based on 2D texture, and the neighbour information usually employed by these approaches is insufficient to reproduce the sample arrangement in the volume [*Chi06*].

Depending on the texture consistency and the user's demand, 3D information can be inferred from 2D observations, constraining two, three or even more views (if appropriate samples are available) leaving the inference discussion open for future considerations.

5.6 Conclusion

This chapter was meant to make a comparison of the texture synthesis techniques described in the previous two chapters.

On one hand it consisted in comparing the fixed-neighbourhood search based approaches described in *Chapter 3* and showing the influence of different parameters on the quality of the synthesis results using the representative approach of this class – NP_K. The results obtained using the NP_K approach are convincing for most of the textures, capturing relatively well the sample structure.

On the other hand, the visual comparison is conducted on the maximum-likelihood based approaches, consisting mainly in analyzing the decisional heuristic on the synthesis results. Similarly the emblematic heuristic is chosen – namely NP_ML_H2c – and the influence of the algorithmic parameters is studied. Choosing properly the algorithmic parameters, satisfying results are achieved but with an obvious disadvantage in terms of computational cost, compared to NP_K.

In the end, the two representatives are subject to a concurrent process that puts face to face the synthetic solid textures obtained with the representative approaches, showing the synthesis potential on various textures. The algorithms provide satisfying results in most of the cases, but there are situations when mixed results appear, varying the satisfying results with the undesired ones. So a clear discrimination of the methods in terms of synthesis quality does not seem to be possible. The choice of a relevant synthesis strategy can hardly be made without considering both the computational requirements and the texture of interest.

Chapter 6

Application to anisotropic textures of carbonaceous materials

Contents

6.1 Motivation 97

6.2 Composite materials 98

6.2.1 Material description 98

6.2.2 Pyrocarbons 98

6.2.2.1 From carbon to pyrocarbons 98

6.2.2.2 Pyrolitic carbon 100

6.2.2.3 Structural and textural parameters of turbostratic carbon 102

6.2.3 HRTEM images of interest for synthesis 102

6.2.4 The need for 3D texture synthesis 104

6.3 Volumetric HRTEM texture synthesis 104

6.3.1 Orthotropic properties 104

6.3.2 Results evaluation methodology 106

6.3.3 Fixed-neighbourhood search based synthesis results 109

6.3.4 Maximum-likelihood based synthesis results 116

6.3.5 Parametric vs. non-parametric 3D texture synthesis methods 118

6.3.5.1 A parametric texture synthesis approach 119

6.3.5.2 Results comparison 121

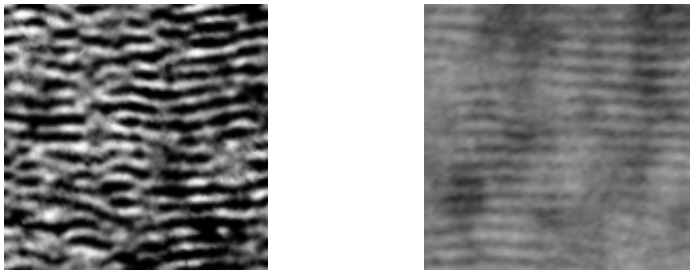
6.4 Conclusion 123

6.1 Motivation

If texture has been the object of so many studies, it is certainly due to its presence in many areas. It can play an important role in sectors as varied as artificial vision for quality control, medical imaging, satellite imagery, multimedia indexing, seismic data exploration but also *material structures understanding*.

In this section, we focus on snapshots of textures that appear on the images of pyrocarbon material obtained by using the lattice fringe technique in High Resolution Transmission Electron Microscopy (*HRTEM*.).

A specific attention is paid to the synthesis of anisotropic textures from a single 2D exemplar, especially laminar textures, i.e. textures made of anisotropic sheets stacked along a given direction. Two image samples of such pyrocarbon textures are presented in *Fig. 6.1*.



a *b*

Figure 6.1 – *Samples of lattice fringe images of pyrocarbons, of size 128×128 pixels. Resolution is about 0.05 nm/pixel.*

Though already studied by 2D HRTEM imaging [*Ger97*] [*Ala98*] [*Shi00*] [*Obe02*] [*Rou02*] [*Ger03*] [*DaC04*] [*Ley09*], such carbon structures cannot be investigated in 3D for many technical reasons related to sample preparation and HRTEM technology. Indeed, the 3D imaging techniques enumerated in *section 2.3* are not suitable for the carbonaceous materials. A direct 3D observation of the atomic structure at nanometric scale is just impossible. The observation is done therefore in 2D, by HRTEM, by the lattice fringe technique. However, considering the assumed structure of such carbonaceous material, more specifically its volumetric homogeneity and anisotropy, it is possible to relate observations made in 2D by HRTEM imaging with the tri-dimensional reality of the material. As well, such properties allow to consider 2D/3D texture synthesis algorithms as a possible way to investigate these 3D structures virtually [*Ley09*] [*DaC10*].

This chapter introduces the concepts of composite materials to better understand their textural and structural characteristics, starting from the atomic scale representation towards the laminar arrangements of plane layers.

Synthetic volumetric textures are produced by using the algorithms described in *Chapter 3* and *Chapter 4*, and are compared quantitatively to the original 2D textures in terms of dynamics and morphological properties. This comparative study allows, on one hand, to identify the most relevant strategies for the synthesis of this kind of HRTEM textures and, on the other hand, to compare them objectively to a previously proposed parametric approach [*Ley09*] [*DaC10*].

6.2 Composite materials

6.2.1 Material description

A composite material, by definition, is made up from several elements that confer special properties to the whole. We distinguish the reinforcement that supports the charges (generally a fiber one) and the matrix that ensures their distribution, the cohesion of the material and the protection of the reinforcement against the environment.

Our interest is based on carbon-matrix composites commonly referred to as carbon-carbon composites (C/C), obtained by graphitising and densifying the original carbon reinforcement. The C/C composites are distinguishable thanks to their lightness, high resistance to ablation, absence of deformation and thermal shocks, while the mechanical resistance and the friction coefficient grow with the increase of temperature. All of these made the C/C materials the perfect candidates for high performance breaking systems and rocket nozzles. These are homo-composite materials, meaning that the matrix and the reinforcement have the same composition – Carbon. The reinforcement contains carbon fibers, while recovered by a pyrocarbon matrix. Carbon fibers can be assembled in pre-forms which are habitually densified by chemical vapour infiltration (CVI) to obtain a composite. The properties of the fibers and the matrix are modulated in correlation to the aimed application.

6.2.2 Pyrocarbons

Carbon exists under different allotropic forms, and most known being amorphous carbon, graphite, diamond, carbon nanotubes and carbon fullerenes as illustrated in *Fig. 6.2*. At the origin of the composite carbon polymorphism stays the capability of the electronic structure of the carbon to establish primarily three types of hybridizations (sp^1, sp^2 and sp^3). Carbon can be found in various molecular configurations, but an intermediary structure between amorphous carbon and graphite shows interesting variable organization rate. Here can be found the turbostratic carbon, the constituent of the C/C fiber and matrix.

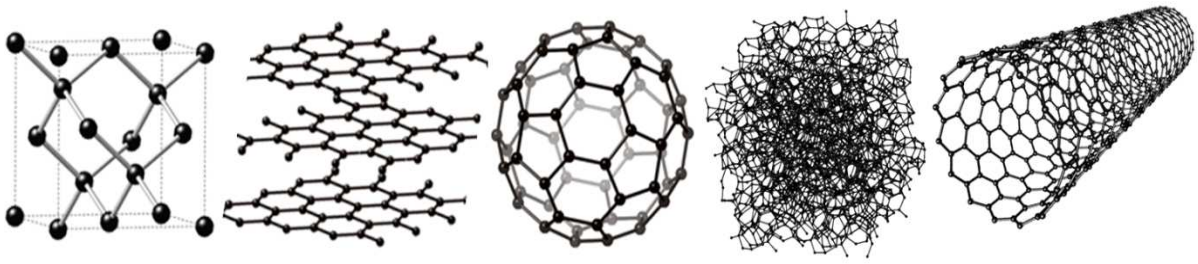


Figure 6.2 – *Five relatively well-known allotropes of carbon: from left to right, diamond, graphite, fullerenes, amorphous carbon, and carbon nanotubes.*

6.2.2.1 From carbon to pyrocarbons

The two most familiar crystalline forms of carbon are graphite and diamond. The two structures differ in the state of hybridization of carbon atoms (sp^2 for graphite and sp^3 for diamond).

A graphene structure is a monolayer of sp^2-bonded carbon atoms that are densely packed in a 2D honeycomb lattice and is a basic building block for graphitic materials of all other dimensionalities and fairly represented in *Fig. 6.3*. These atomic layers are also called aromatic layers, constantly offering new inroads into low-dimensional physics that has never ceased to surprise and continues to provide a fertile ground for applications [*Gei07*].

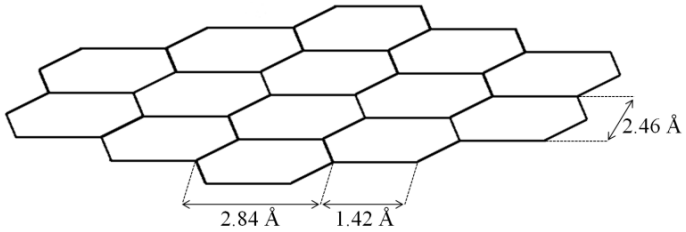


Figure 6.3 – *Illustration of one graphitic layer and its aromatic cycles.*

Graphite is a stack of graphene planes 3.35 Å apart, themselves compounds of carbon atoms assembled in hexagon configuration. Each carbon atom is bound to three of its neighbours in the same plane. The inter-atomic distance is 1.42 Å. Bonds between carbon atoms in the same plane are strong, while those between plans (like the Van Der Waals bonds) are weaker. These lamellar structures are responsible of the anisotropy of the electrical, thermal and mechanical properties of the graphite. This allows in the same time various degree of organization.

Turbostratic carbon is composed of '*elementary bricks*' (Basic Structural Units - BSU), i.e. organized areas formed by graphene plans which exhibit faults, deficiencies or dislocations [*Pie93*]. The BSU, also known as the coherent domain, is a representation model of the smallest carbon entity used to describe, build and explain the aromatic carbon texture evolution. The layers within the turbostratic carbon are graphitic layers or paralleled piled graphenes disordered by rotation [*Bou93*] [*Pie93*].

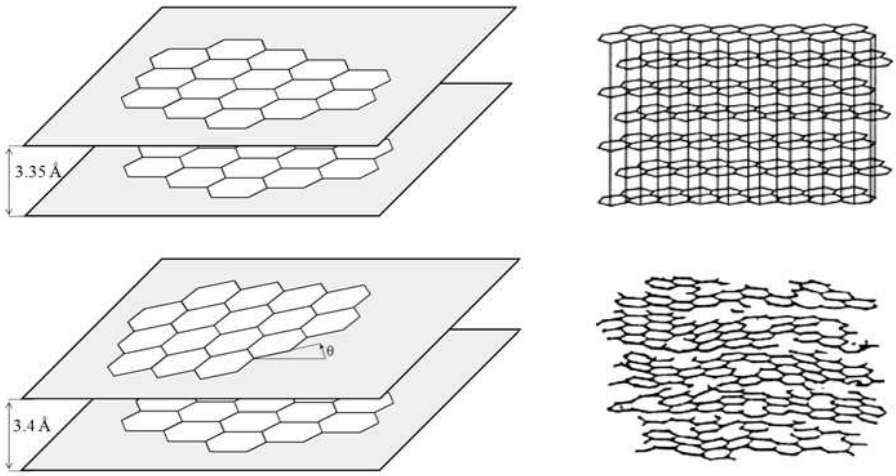


Figure 6.4 – *Turbostratic carbon structure (half-bottom part) vs. graphite structure (half-top part): on left, the simplified graphene planes representation with the reticular distance (3.35Å for graphite, 3.4Å for turbostratic carbon); on the right – the corresponding structural representation of a graphite and a pyrocarbon.*

As shown in *Fig. 6.5*, it is possible to evolve the structure of a turbostratic carbon towards that of graphite, completely or partially, if the orientation of crystallites allows it [*Obe89*]. This process is called graphitization and it is obtained by treating thermally the carbon. The degree of graphitization is directly related to the distance between the planes of

graphene; the higher the degree of graphitization, the closer is the reticular distance to its theoretical value [*Mar89*].

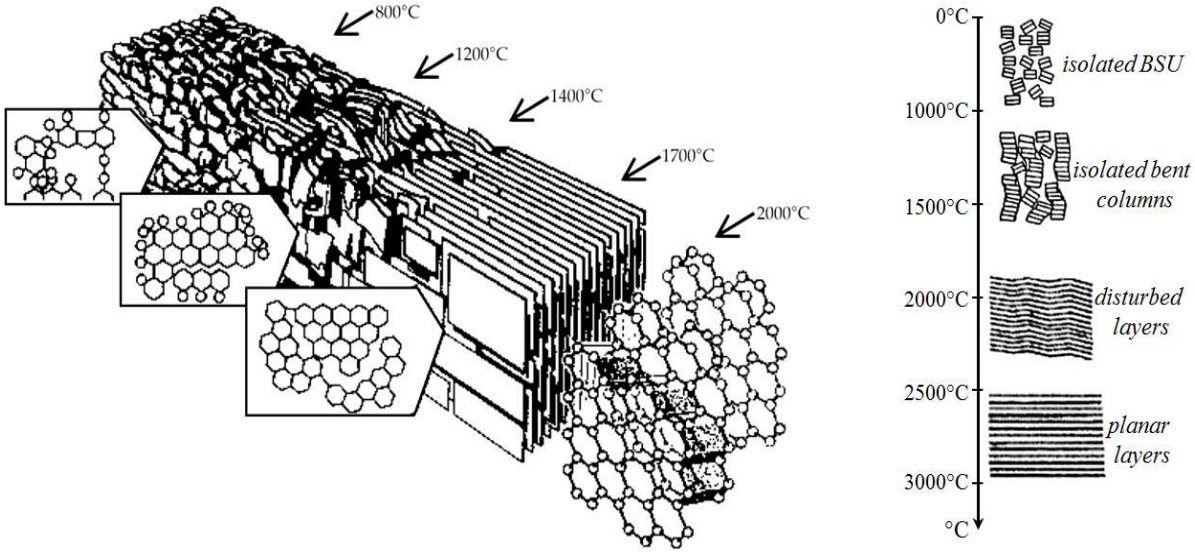


Figure 6.5 – *Illustrating the carbon structure evolution with the temperature (on the left) and the crystallites alignment during graphitization stages (on the right) [Obe89].*

The transformation of carbon structures as presented in *Fig. 6.5* takes place in several stages:

- the 1st stage corresponds to the raw carbon (amorphous predisposition) in the form of isolated coherent domains;
- the 2nd stage consists in the growth of the coherent domains in columns (between 800ºC and 1500ºC); at this stage the turbostratic carbon can be retrieved in the form of deposited pyrocarbon;
- the 3rd stage carries on the graphitisation, being the phase when the graphene layers interconnect forming crystallites (between 1500ºC and 2000ºC); starting from this stage the thermally treated pyrocarbons can be found;
- the 4th stage consists in the proper graphitisation; beginning from around 2000ºC-2200ºC a graphite structure is formed; this is the final stage, because above 2800ºC-3000ºC no supplementary structural modification is observed.

6.2.3 Pyrolitic carbon

Pyrocarbon (PyC) or pyrolitic carbon is a solid form of carbon formed by thermal cracking of a liquid or gaseous precursor having the structure of a turbostratic carbon [*Sav93*]. Similar to the graphite, it is made up of a stack of graphene plans where carbon atoms have sp^2 hybridization. What differentiates pyrocarbon from graphite is that the different graphene plans are stacked in an isotropic way.

The pyrocarbon matrix of the C/C composites can be obtained by liquid or gaseous way. The gaseous way or chemical vapor infiltration (CVI) allows obtaining superior final properties and better reproducibility. Even if it is longer and costlier, it is the privileged approach. We talk about the form of the carbon which deposits on hot surfaces above 900°C by cracking of hydrocarbon [*Bou06*].

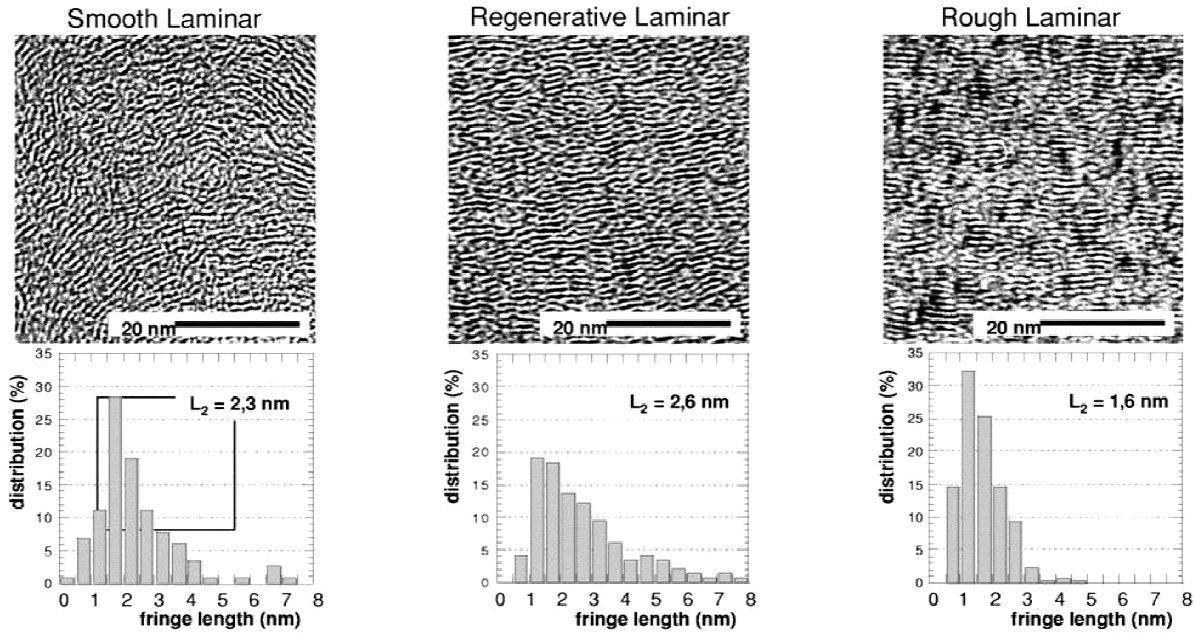


Figure 6.6 – *High resolution TEM mode of the different pyrocarbons [Bou02]: (a) smooth laminar, poor anisotropy and wide distribution of layer diameters (L_2 is the mean value as presented in Fig. 6.8); (b) regenerative laminar, good anisotropy and wide layer diameter distribution; rough laminar, high anisotropy and small layer diameter.*

Pyrocarbon and other turbostratic carbons differ in that the layers are disordered, resulting in wrinkles or distortions within layers. This gives pyrocarbon improved durability compared to graphite.

According to the pyrocarbon degree of anisotropy, one can distinguish several types of classes:

- isotropic laminar (PyC ISO) – which shows random orientation of graphene plans
- smooth laminar (PyC LL) – the organization of the graphene plans still show defects
- rough laminar (PyC LR) – the graphene planes are perfectly parallel to the fibers surface

Between these classes, transition stages appear, leading to other pyrocarbon forms, the regenerated laminar (Pyc LRe) – very anisotropic, the granular pyrocarbon (PyC G) etc.

Figure 6.7 – *Examples of Pyrocarbon applications: from left to right, Messier-Buggati-Dowty breaking system [Mess], Snecma Vinci motor [Snec], graphitic heart valve and inter-phalangeal joint prostheses [Bou06].*

Pyrocarbon applications (see *Fig. 6.7*) are of great interest in mainly three directions:

- the use of C/C composite materials for aircrafts breaking discs, rocket engines nozzle or atmospheric re-entry corps;

- in the medicine field – used in fabricating coatings heart valves or small bone prostheses thanks to the pyrocarbon biocompatible character;
- for the next generation of nuclear reactors, pyrocarbon acting as a confinement barrier.

6.2.3.1 Structural and textural parameters of turbostratic carbon

The turbostratic carbon exhibits locally in his structure ordered 3D entities, consisting of stacked graphene layers, parallel and equidistant to each other, of finite dimensions and disoriented with respect to each other. These entities are characterized by structural parameters (short range order) and textural parameters (long range order) [*Bou93*]:

- d_{002} : the average distance between two successive layers of graphene, also named as the *reticular distance*
- L_a , the coherence width: the average diameter of the stacked graphene layers (of the coherent domain)
- L_c , the coherence length: the average height of the stacked graphene layers
- L_1: the measured diameter of the flat portion of one graphene layer
- L_2: the measured diameter of one graphene layer
- N: the number of stacked graphene layers
- β: the torsion angle along the layers

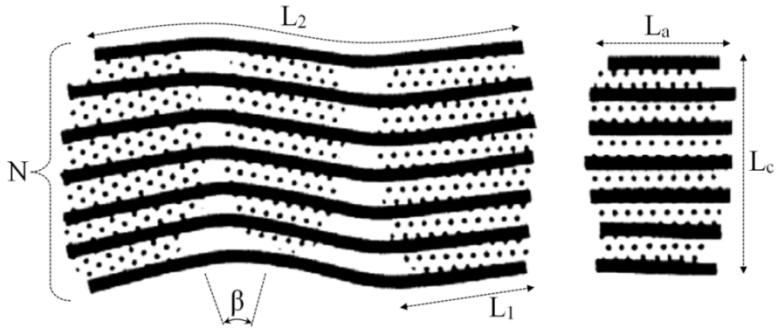


Figure 6.8 – *Schema of stacked layers, showing the structural and textural parameters.*

6.2.4 HRTEM images of interest for synthesis

The images, of which textures are used, are snapshots resulting from the observation in High Resolution Transmission Electron Microscopy (HRTEM) of pyrolytic deposits obtained during phases of densification and heat treatment of C/C composites.

Nowadays HRTEM combines recent instrumental developments with innovative strategies of imaging and image processing. It has become a truly ‘quantitative’ technique that enables one to retrieve the atomistic structure of materials with high and well-known reliability. Some considerations about the principle of image simulation using TEM are shown in *appendix E*.

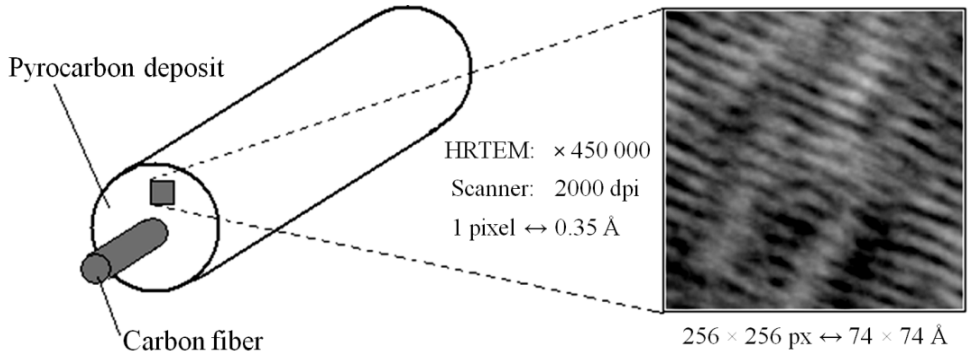


Figure 6.9 – *Example of carbon fiber and pyrolitic deposit and the textural HRTEM snapshot (image reproduced from [DaC01]).*

HRTEM technology provides representations of 10^5 to 10^6 magnification order relative to the initial sample. Digitizing those, leads to the texture images used as synthesis exemplars. *Fig. 6.9* shows an image of 256×256 pixels, obtained by a 450000 order of magnification and a 2000 dpi digitizing, reaching to a final resolution of 0.35 Å per pixel.

Other samples of lattice fringe images of dense pre-graphitic carbons are presented in *Fig. 6.10*, being cuts of 128×128 pixels from the bigger samples available in *appendix F*. These are anisotropic textures composed of elongated structural elements showing low undulation.

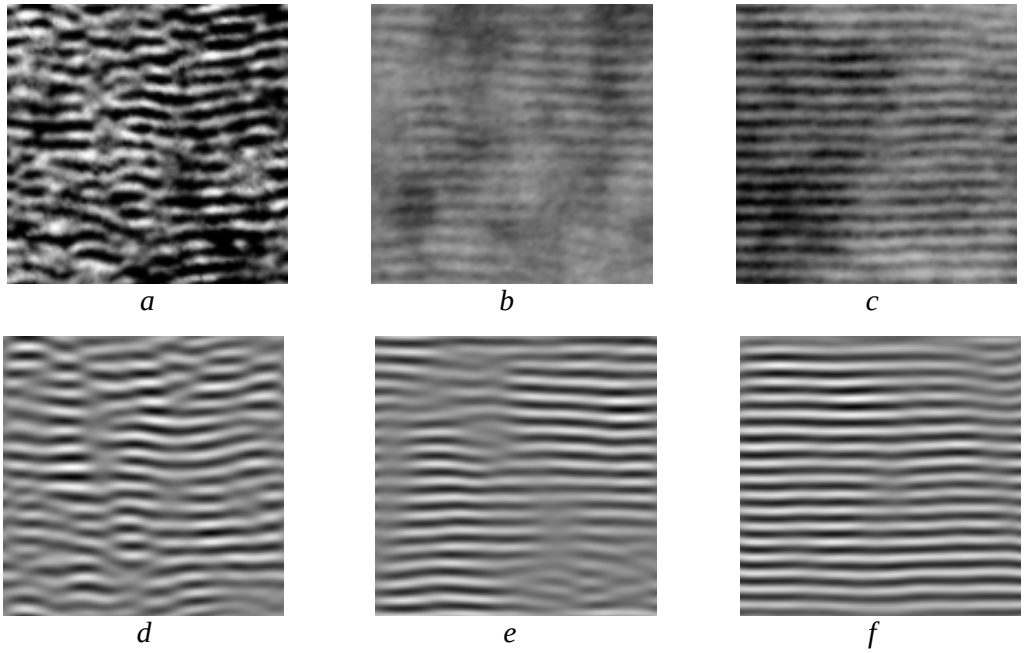


Figure 6.10 – *Samples of lattice fringe images of dense pre-graphitic carbons: raw HRTEM images (a, b, c) and their filtered versions (d, e, f). Sample sizes are 128×128 pixels.*

The raw samples in *Fig. 6.10a-c* show locally periodic patterns together with high and low frequency artefacts. Neither high frequency nor low frequency components provide useful information about the atomic structure of the observed material. A good compromise for analyzing them consists in using their filtered versions (in *Fig. 6.10d-f*) by applying a radial and directional band-pass filter in the frequency domain [*DaC04*].

6.2.5 The need for 3D texture synthesis

The above images obtained by TEM observation of pyrocarbon deposits show interference fringes which reproduce in projection the atomic layers (the aromatic layers). The degree of organization of these layers reflects the material graphitisation stage.

The study of the spatial organization of these fringes plays an important role in characterizing the properties of the composite materials. Despite the fact that they have been investigated in 2D [*Shi00*] [*Obe02*] [*Rou02*] [*DaC04*] [*Ley09*], no three-dimensional imaging technique appears to be appropriate at such high resolutions.

Nevertheless, the three-dimensional structure can be accessed by inferring it from the 2D sample using 2D/3D image synthesis [*Ley09*] [*DaC10*]. Further on, the synthetic 3D images serve as virtual material that can be used for atomistic reconstruction. Atomistic reconstruction methods are entrenched tools for linking experimental characterization data to the atomic scale structure of matter [*Ley09*]. Most of the existing methods are not suitable for anisotropic systems, as it is the case of the nanotextures of dense graphene based carbons in *Fig. 6.10*, guessing the initial atomic structure.

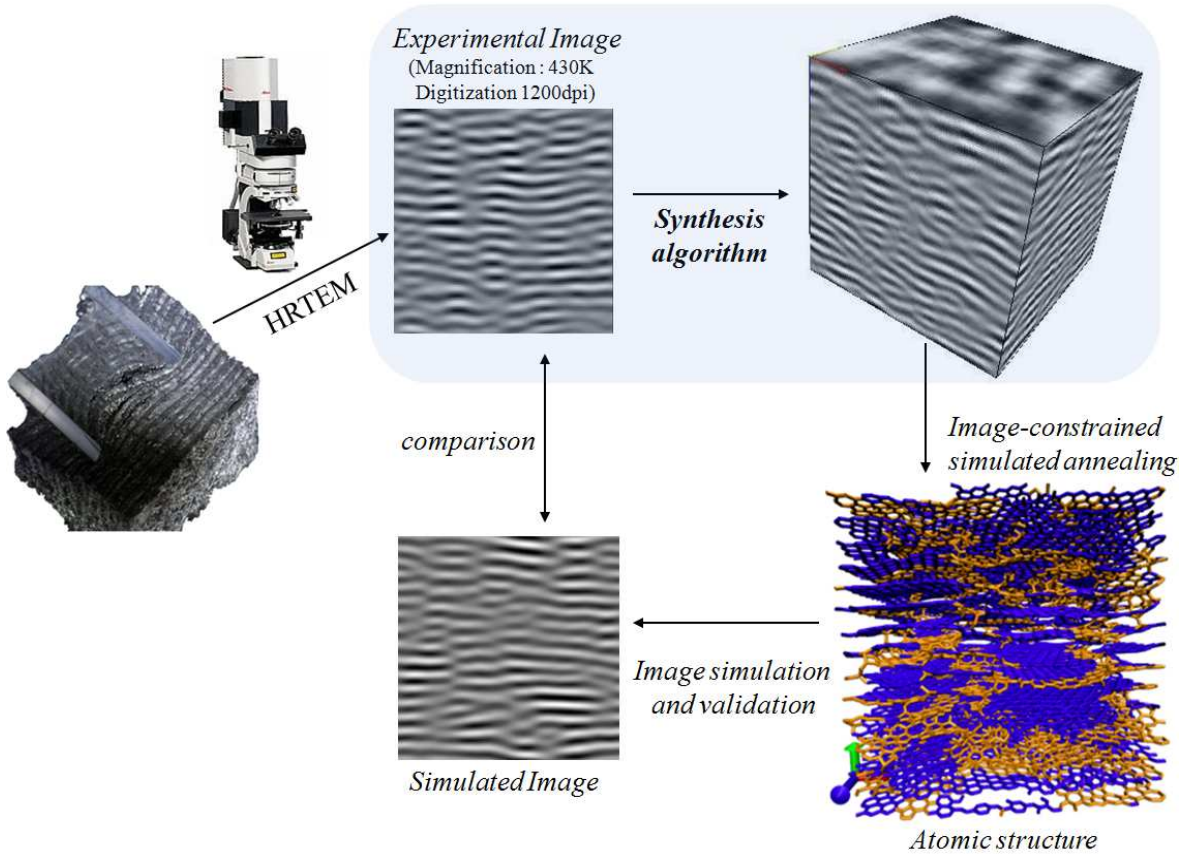


Figure 6.11 – *The 2D/3D texture synthesis step inside the IGAR method.*

A recent method, called Image-Guided Atomistic Reconstruction (IGAR), capable of generating the atomistic models for anisotropic nanotextures was developed by [*Ley09*] compelled by a 3D synthetic texture. Using an image-constrained simulated annealing, the 3D atomistic structure is simulated by exploiting the external potential of the 3D textures to bring the atoms to settle preferentially at specific positions (i.e. the black areas inside the 3D texture).

Supplementary, a simulated HRTEM texture can be obtained from the atomistic structure, and it can be compared with the experimental texture in order to validate the

atomistic model [*Far12*]. A schematic illustration of this process is showed in *Fig. 6.11* pointing out the texture synthesis step it involves.

The published works concerning the IGAR method are based on a parametric synthesis algorithm showing the synthesis method potential and also its limits [*DaC10*]. One objective of this chapter (and implicitly of this thesis) is to evaluate other types of approaches, and this time non-parametric ones.

6.3 Volumetric HRTEM texture synthesis

Texture synthesis is a major step in the atomistic reconstruction of HRTEM data, but the volumetric texture synthesis procedure starting from a 2D HRTEM sample is however delicate. Therefore some statements have to be made before presenting the algorithms implementation and the benchmark used for comparing the 3D results.

6.3.1 Orthotropic properties

The nearly periodic directional structure observed in HRTEM images is typical of 3D laminar arrangements. The dense graphene based carbons show plane mono-atomic layers of graphene stacked together along the direction of orthotropy i.e. the vertical direction of the textures from *Fig. 6.10*. These materials have an orthotropic structure, meaning that the properties of the material are the same on all the directions orthogonal to the direction of orthotropy.

The observed 2D HRTEM images of such a material have the same aspect whatever the viewing angle as long as it is orthogonal to the orthotropy direction as exemplified in *Fig. 6.12*. Thanks to these properties of symmetry that are verified by the dense graphene based carbons, its 3D statistical properties can be deduced from the ones observed in the 2D HRTEM samples [*DaC10*].

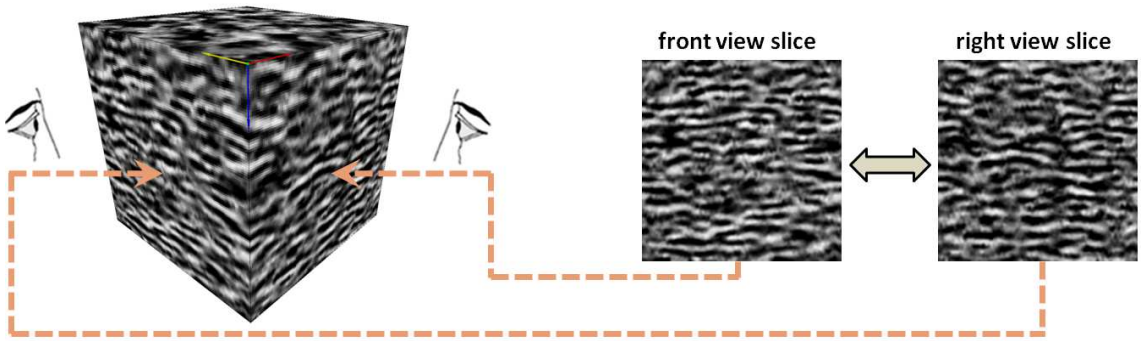


Figure 6.12 – *Properties of symmetry: two orthogonal slices of the same block (for the front view and the right view) show the same statistical properties.*

So it is possible to relate measurements made on a single 2D cross-section of the volume with the 3D properties of the material using stereological considerations (i.e. orthotropy) [*Urs10*] and accessing the three-dimensional structure by inferring it from 2D using 2D/3D texture synthesis.

Consequently, in the followings of this chapter, the synthesis is made by constraining only two orthogonal views of the volume (the front and the side view, since no sample is

available for the 3rd view) with a single 2D HRTEM sample as the ones in *Fig. 6.10*. The principle behind the texture synthesis process is illustrated in *Fig. 6.13*.

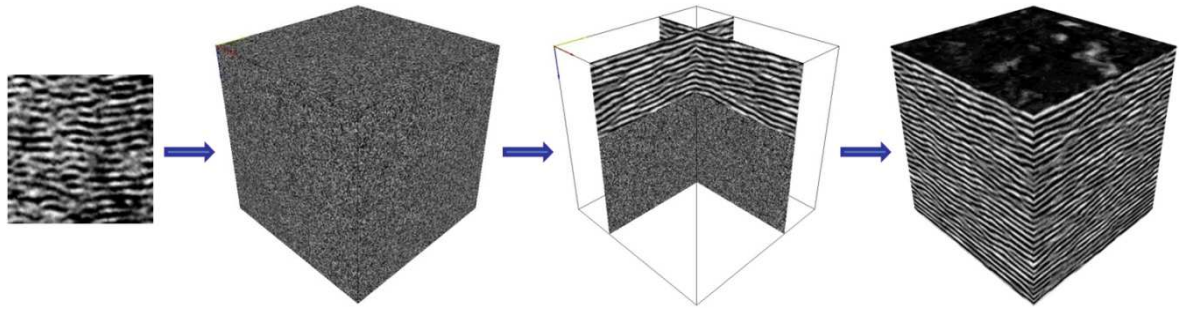


Figure 6.13 – *The texture synthesis process: starting from a random initialization, the synthesis is performed voxel per voxel by examining the 2D voxel neighbourhoods extracted from two orthogonal views of the 3D block (front and side); the value of the output voxel is updated with the best candidate from the input image after a fixed-neighbourhood search, or with the most probable grey-value from the input image to be in that position conditioned by its neighbours (in function of the chosen methodology).*

6.3.2 Results evaluation methodology

In the particular case of HRTEM textures, beyond the visual interpretation (as in the case of textures used in computer graphics), a quantitative study matters a lot, finding itself very useful in describing and understanding the internal arrangement of the observed layers. For this reason an experimental evaluation is carried on, that allows to identify the relevant strategy for the volumetric synthesis and also to compare objectively various algorithms from the literature.

This quantitative study evaluates the results taking into account, on one hand, the grey level dynamics (first order statistics) and, on the other hand, the morphological properties like the lengths, the order of tortuosity and the local orientations of the elongated patterns (fringes) present in the texture.

Properly stated, it aims at evaluating the algorithms capacity to reproduce the orthotropic layered structure of pyrolytic carbons, respecting the HRTEM observations.

Grey level dynamics study means at the outset to compute the 1st order statistics* as the mean, the average deviation, the standard deviation, the variance, the skewness or the kurtosis of the grey level pixels, in the exemplar and in the output block.

There's no need to mention additional information about these statistics, perhaps only some brief remarks on the skewness (the 3rd standardized moment) and the kurtosis (the 4th moment). These two are descriptors of the shape of a probability distribution: the skewness measures the degree of asymmetry while the kurtosis is a measure of peakedness or degree of flattening.

Next, the Kullback-Leibler divergence is computed as a dissymmetry measure of the difference between the probability distribution of the exemplar grey level and the output 3D block grey level.

It is not a 'distance' in the mathematical sense of the term, but in the literature one can find commonly the symmetrical version of the Kullback-Leibler divergence between two

* The 1st order statistics estimate properties of individual pixel values, ignoring the spatial interaction between image pixels, whereas 2nd and higher order statistics estimate properties of two or more pixel values occurring at specific locations relative to each other [*Sri08*].

distributions, that can be interpreted as a distance, and that we have used in the quantitative analysis in the following form:

$$D_{KL\_sym}(P\|Q)=\sum_i p(i)\log\frac{p(i)}{q(i)}+\sum_i q(i)\log\frac{q(i)}{p(i)}, \tag{6.1}$$

where *P* and *Q* correspond to the probability distribution of the bins in the grey levels histogram of the input exemplar and the output block while *i* runs through all the bins in the histogram, mentioning that the output block computation is seen as a multi-2D analysis.

Although at the end of the synthesis process, the user can freely intervene to change certain statistics in order to match them exactly to the exemplar (like the mean or the variance) I choose not to interfere, leaving the algorithm to follow its natural course, with its advantages and disadvantages to draw attention to. That's why, in addition to the grey level dynamics comparison, a morphological study is conducted, study that's immune to the above potential grey level basic corrections.

Concerning the **structural properties** of the 3D blocks, the study is oriented towards the comparison of the local orientations of the volumetric block (analysed as a multi-2D block representation) relative to the input sample, based on a structure tensor estimated on a 7×7 neighbourhood inspired from [*DiZ86*] [*Big91*]. The analysis of the shape of the local tensor heads to a local orientation.

For an image I, the structure of the tensor is given by:

$$T=\begin{pmatrix}\overline{I_xI_x} & \overline{I_xI_y}\\ \overline{I_yI_x} & \overline{I_yI_y}\end{pmatrix}=\begin{pmatrix}\overline{I_x^2} & \overline{I_{xy}}\\ \overline{I_{xy}} & \overline{I_y^2}\end{pmatrix} \tag{6.2}$$

where the subscripts indicates spatial derivatives and the bar $\overline{\ }$ indicates the convolution with a Gaussian filter. In practice, the tensor calculation is based on the image gradient, the gradient magnitude of the image intensity at each point being approximated by convolving the image with the Sobel convolution kernel.

Eigenvalue analysis of the tensor leads to two eigenvalues defined by:

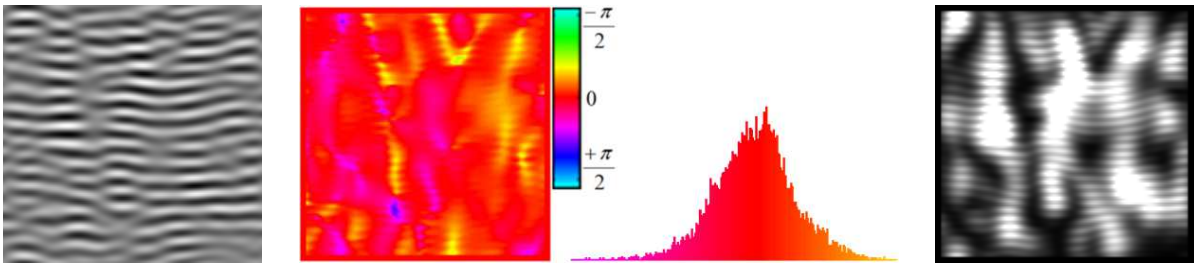
$$\lambda_1=\frac{1}{2}\cdot\left(\overline{I_x^2}+\overline{I_y^2}+\sqrt{\left(\overline{I_x^2}-\overline{I_y^2}\right)^2+4\cdot\overline{I_{xy}}^2}\right) \tag{6.3}$$

$$\lambda_2=\frac{1}{2}\cdot\left(\overline{I_x^2}+\overline{I_y^2}-\sqrt{\left(\overline{I_x^2}-\overline{I_y^2}\right)^2+4\cdot\overline{I_{xy}}^2}\right) \tag{6.4}$$

The direction of the eigenvectors related to λ_1 indicates the prominent local orientation:

$$\theta=\frac{1}{2}\cdot\arctan\left(\frac{2\cdot\overline{I_{xy}}}{\overline{I_x^2}-\overline{I_y^2}}\right) \tag{6.5}$$

The corresponding confidence, computed as $(\lambda_1-\lambda_2)/(\lambda_1+\lambda_2)$, associated to the local orientation describes the total local derivative energy. *Fig. 6.14* illustrates the image orientation field alongside the image confidence and its associated orientation histogram.

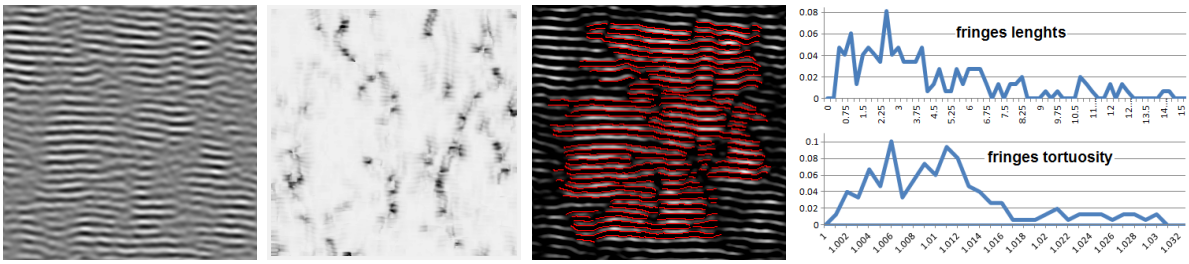


a *b* *c* *d*

Figure 6.14 – *Texture orientation: (a) the HRTEM sample from Fig.6.11d; (b) the orientation field map and the colour bar giving the correspondence between colours and orientations; (c) the orientation histogram computed from the orientation field and (d) the corresponding confidence map.*

The comparison of the **morphological properties** consists in tracking the texture level curves and describing the elongated patterns contained within the texture. We address the orientation and pattern length measurement techniques as described by [*DaC00*] and [*Ray10*] to extract textural features (fringes) and compute their lengths and tortuosity. It allows a relevant description of the shape of the patterns within the texture.

The procedure consists first in filtering the raw HRTEM samples as regards spatial frequency and direction [*DaC04*] in order to obtain the lattice fringe images upon which one can perceive promptly the graphene layers organization. Pattern analysis proceeds with tracking the level curves, by successively finding the sets of seeds (i.e. sets of linked interpolated pixels) that have matching grey values.



a *b* *c* *d*

Figure 6.15 – *Illustration of the level-curve tracking algorithm applied on a HRTEM pyrocarbon sample: (a) a lattice fringe image, (b) the associated confidence map, (c) level curves extraction and (d) shows the distribution of the L_2 and τ parameter values in the HRTEM image.*

Once the fringes are extracted, they can be classified as a function of their lengths (that is the L_2 parameter of the graphene layers) by counting the frequency of appearance of the fringe lengths, and in function of their degree of tortuosity (that is the τ parameter of the graphene layers).

Consequently, tortuosity classes are built in order to compare the distribution of the tortuosity values. An example showing the level-curve tracking algorithm outcomes is presented in *Fig. 6.15*.

Curves that ‘continue’ outside the image are neglected. This should not be a problem if the image is large enough to track a sufficient number of fringes. For smaller synthetic texture samples, because the algorithm already provides tileable textures, borderline fringes can be detected by considering the circular image as it is illustrated in *Fig. 6.16*.

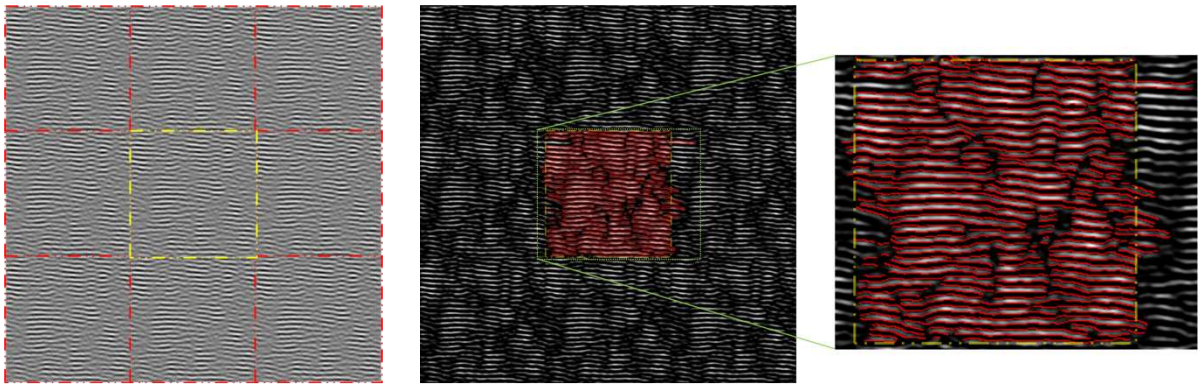


Figure 6.16 – *Example of the level-curve tracking algorithm employ, in the case of a circular lattice fringe image: from left to right, the seamless tileable texture obtained by using the sample in Fig 6.21a and the level curve extraction with a proper zoom on the area of interest*

To recapitulate, the measured characteristics involved by the evaluation methodology are as following:

- for grey-level dynamics: grey-level histograms and 1st order statistics (standard deviation, skewness and kurtosis);
- for the structural properties: local orientations and the fringes lengths and tortuosity.

6.3.3 Fixed-neighbourhood search based synthesis results

Adapting the algorithms described in *Chapter 3* to using only two orthogonal views, synthetic volumetric textures are produced. The results are shown in *Fig.6.17* for the raw HRTEM images and in *Fig. 6.18* for the filtered HRTEM samples.

Fig. 6.17 and *Fig. 6.18* contain some volumetric results obtained from a single HRTEM texture using non-parametric approaches. Figures *6.17d-f* and *6.18d-f* show the volumetric textures using NP_WL algorithm, while figures *6.17g-i* and *6.18g-i* show the results obtained using NP_K, i.e. the backbone of NP_WL but optimized by using the weighted average and the colour histogram matching technique. On the 4th row in figures *6.17* and *6.18* are represented the 3D blocks obtained by using NP_CW, i.e. using the discrete solver plus the index histogram and the position histogram matching technique.

The experimentation framework for the mentioned results consists in using 5 pyramidal levels, full-square non-causal neighbourhoods of size 7×7 pixels, Hilbert curve scan type and, for NP_CW, a set of 15 candidates for each pixel. The input exemplar is of size 128×128 pixels, while the results are of size 128×128×128 pixels.

The provided results represent the 3D block obtained after 10 iterations. We have observed that, generally, iterating much more does not yield better results since a high number of iterations creates repetitions of patterns.

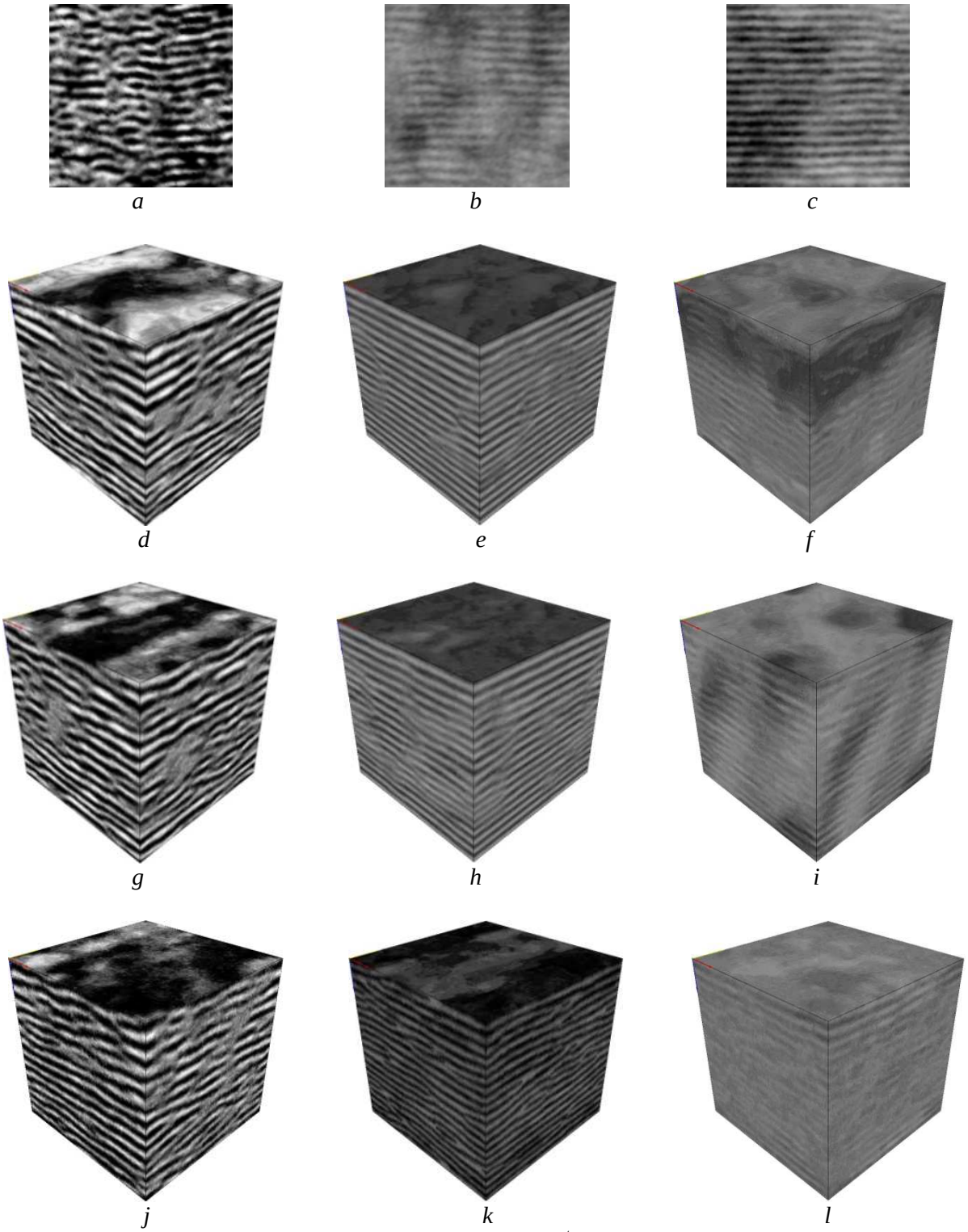


Figure 6.17 – *Volumetric results: from top to bottom, the 1st row (a, b, c) shows the 2D raw HRTEM textures; the 2nd row (d, e, f) represents the 3D views of the solid textures obtained after 10 iterations from the samples in the 1st row using NP_WL approach; the 3rd row (g, h, i) contains the 3D textures obtained by applying the NP_K method and finally the 4th row (j, k, l) corresponds to the results obtained with the NP_CW proposed method in Chapter 3. This illustration allows a direct visual comparison of the fixed-neighbourhood search based methods for each raw sample, by scanning each column from top to bottom (input sample, NP_WL result, NP_K result and NP_CW result).*

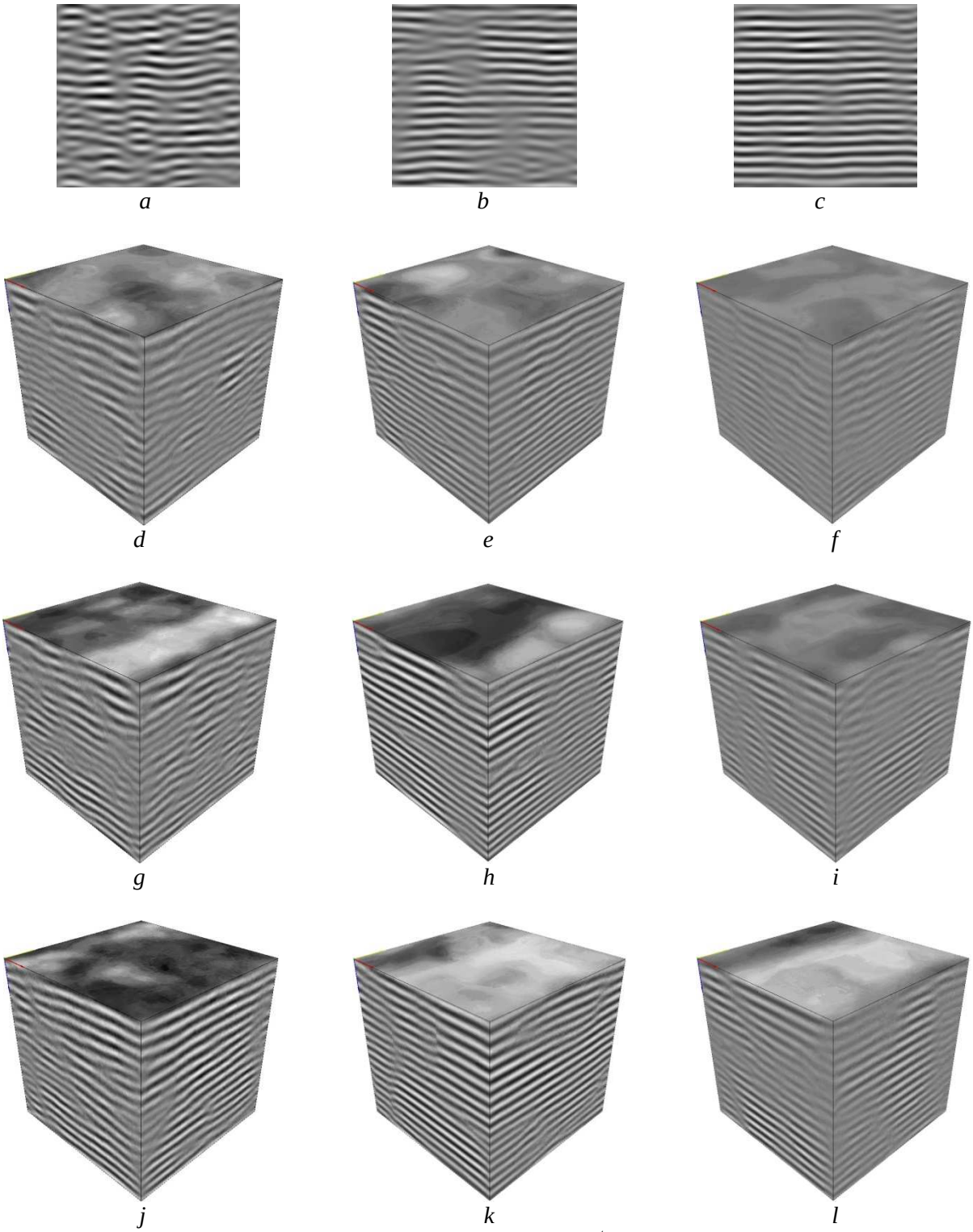


Figure 6.18 – *Volumetric results: from top to bottom, the 1st row (a, b, c) shows the 2D filtered HRTEM textures; the 2nd row (d, e, f) represents the 3D views of the solid textures obtained after 10 iterations from the samples in the 1st row using NP_WL approach; the 3rd row (g, h, i) contains the 3D textures obtained by applying the NP_K method and finally the 4th row (j, k, l) corresponds to the results obtained with the NP_CW proposed method in Chapter 3. This illustration allows a direct visual comparison of the fixed-neighbourhood search based methods for each filtered sample, by scanning each column from top to bottom (input sample, NP_WL result, NP_K result, NP_CW result).*

With a reasonable computational cost (a few minutes per synthesized block) the non-parametric approaches provide results of a fairly satisfactory visual quality. For the volumetric textures from *Fig. 6.17f* and *Fig. 6.18f* the version of NP_WL's algorithm is not able to preserve contrast (because of the averaging operation used when combining the two orthogonal views). This is not the case for NP_K and NP_CW thanks to its weight optimization and its histogram adjustment process. However, some results exhibit several visual disadvantages such as blurring, missing or repeating input textural patterns (like in *Fig. 6.17f,i,l* and *Fig. 6.18f*), while in a few cases NP_CW fails in preserving the dynamics (*Fig. 6.17k-l*).

Globally, after comparing visually the results, one can say that the synthesis behaves better on the filtered images than on the raw HRTEM samples.

But the visual interpretation of the synthesis results remains subjective, distinguishing more or less a bad or a good result. Here intervenes the *quantitative comparison* as described in *section 6.3.2*. Different indicators of this objective study are shown in *Fig. 6.19*.

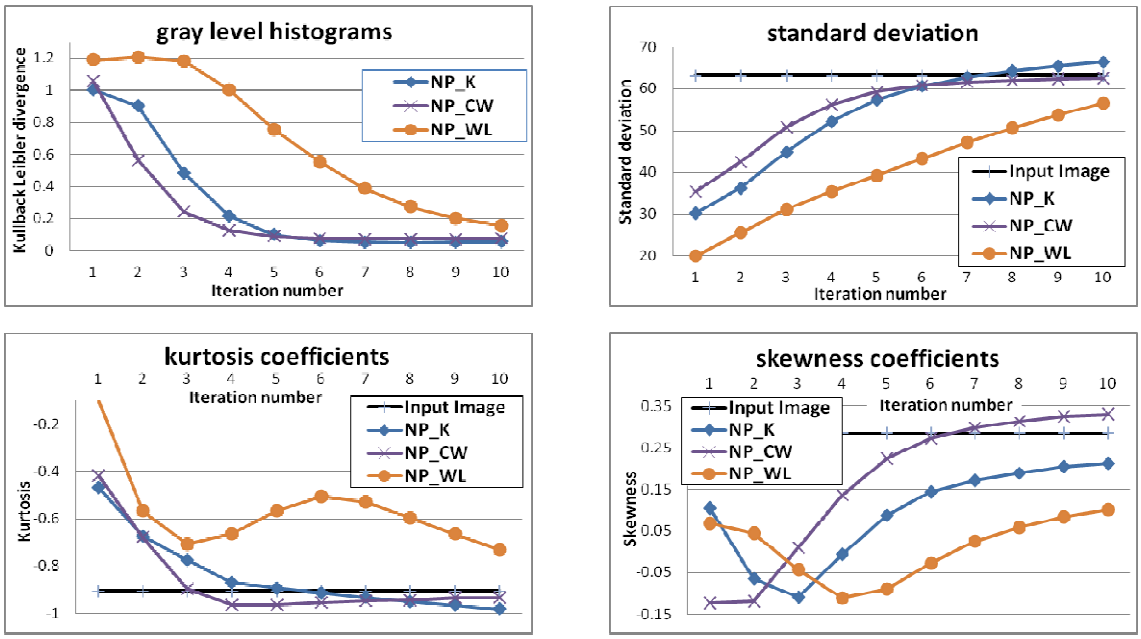
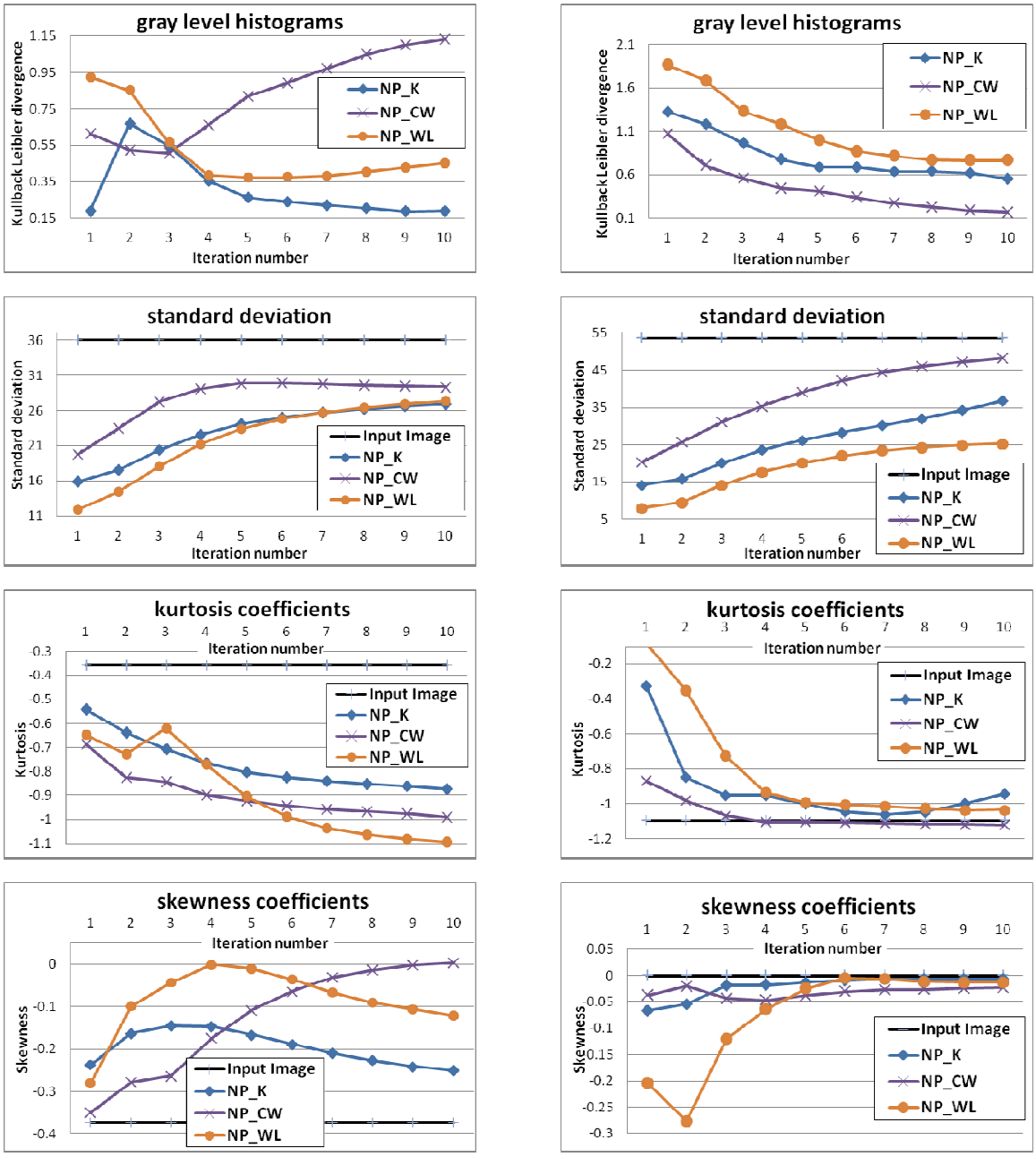


Figure 6.19 – *Different indicators of the objective comparison of the 3D textures obtained by synthesizing the HRTEM texture in Fig. 6.10a: grey level histograms and 1st order statistics - standard deviation, kurtosis and skewness. The procedure consist in comparing input image statistics with the ones obtained after a multi-2D solid block analysis and by computing Kullback-Leibler divergence between the histograms of the exemplar and the output block all along 10 iterations.*

From the first hint of the eye on the *dynamics* of the results, it's obvious that the histogram adjustments bring significant improvements compared to the backbone method, as demonstrated quantitatively by the grey level comparison in *Fig. 6.19* and from *Fig. 6.20b*. However, in the case of dynamics comparison in *Fig 6.20a*, the non-parametric algorithm improved with the index and position histogram is not capable of preserving the grey levels, by contrary, it diverts the statistics from the target.

And as expected, the visual conclusions are reinforced quantitatively by the grey level histogram comparison, generally strengthening the visual assumptions relative to the significant improvements brought by the histogram matching techniques. The quantitative indicators reveal mostly the 3D textures tendency towards the same statistics observed on the HRTEM exemplars.



a *b*

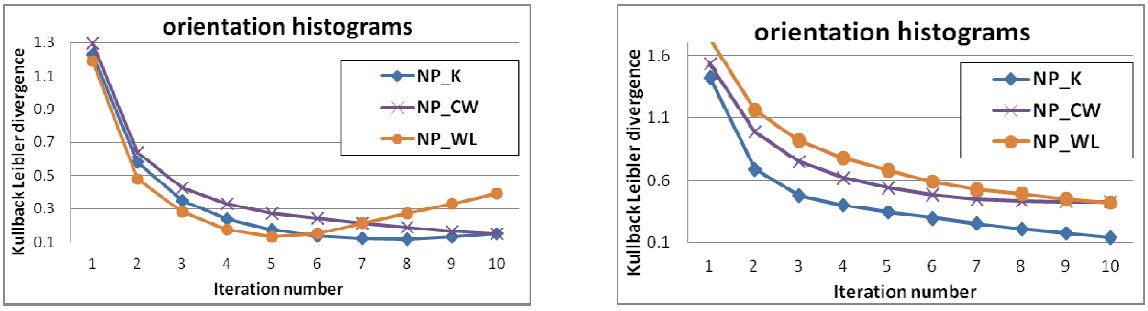
Figure 6.20 – *Different indicators of the objective comparison of the 3D textures obtained by synthesizing the raw HRTEM texture in Fig. 6.10c (a - the 1st column plots) and its equivalent filtered sample in Fig. 6.10f (b - the 2nd column plots): from top to bottom, grey level histograms and 1st order statistics - standard deviation, kurtosis and skewness. The procedure consists in comparing input image statistics with the ones obtained after a multi-2D solid blocks analysis or by computing Kullback-Leibler divergence between the histograms of the exemplar and the output block.*

Another important remark is that in general the approaches take more or less time to reach the statistics of the input image. If the approaches involving histogram adjustments take less than NP_WL, approximately five iterations, the basic method takes almost double in number of iterations to reach the same statistics, above all because of the averaging operation when combining the two orthogonal views.

Experimentally, we have observed that usually, iterating much more does not yield better results, since a high number of iterations creates repetitions of patterns and produces more regular textures than those of the input image.

As presented in *Fig. 6.10*, we dispose of the raw HRTEM images, and a filtered version of these images. Filtered images were used by [*DaC10*] to ease image synthesis, because their parametric synthesis algorithm showed difficulties in handling raw images. The filtering consisted in using two filters, a radial and a directional band-pass filter in the frequency domain [*DaC04*]. As shown in the *Fig. 6.20*, the non-parametric methods behave relatively well with both images, though in some cases (for example the NP_CW approach) the filtered image results are better than the results obtained with the raw image.

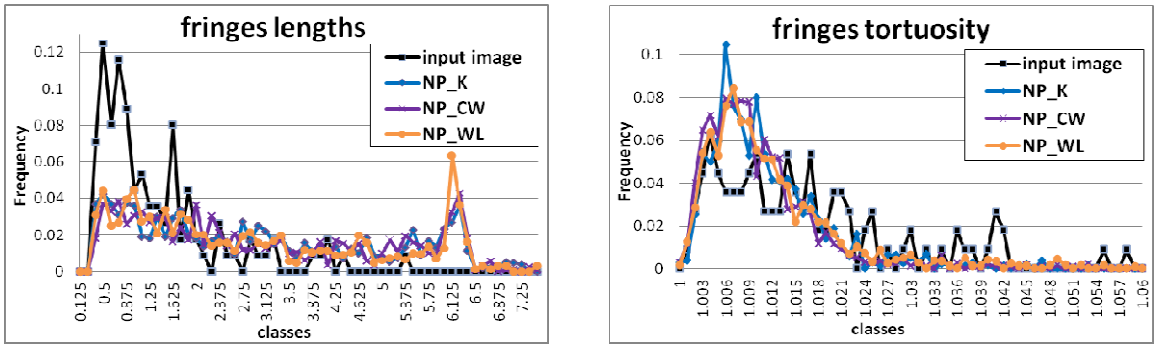
Algorithms comparison based on the *local orientation* histograms as shown in *Fig. 6.21* confirms the previously stated convergence. However, in some cases, the impression is that the proposed algorithms tend to produce textures more regular than that of the exemplar. Iterating more doesn't resolve the reproduction of patterns. It's mostly the result of the insubstantial average of NP_WL algorithm and its local deterministic character.



a *b*

Figure 6.21 – *Spatial structure variation indicators: the Kullback - Leibler divergence between the orientation histograms of the exemplar and the output blocks obtained by applying the algorithms on the sample in Fig. 6.10c (plot a) and on the sample in Fig. 6.10f (plot b).*

The *morphological properties* comparison was done by comparing the L_2 parameter and the τ parameter of the input lattice fringe image and the synthesized texture block obtained after ten iterations, being analysed as frontal multi-2D lattice fringe images. The results from *Fig. 6.22* show that the produced textures contain overall longer fringes than the input image (*plot 6.22a*) and in the same time they are more regular (*plot 6.22b*). This phenomenon may be related to the deterministic character of the algorithms, based on local decisions, which tends to produce these very regular or repetitive textures. Using larger images on input and synthesizing bigger blocks on output should impose a sufficient number of fringes capable of describing better the morphology.



a *b*

Figure 6.22 – *Plots of the morphological structure indicators: (a) the distribution of fringe lengths in the input HRTEM image from Fig. 6.10a and in the multi-2D synthesized volumetric textures from Fig. 6.16d,g,j; (b) the distribution of the tortuosity values corresponding to the same input image and output blocks used in a. Fringes were retrieved by managing front and side slices in the 3D textures.*

Fig. 6.23 shows the synthetic texture evolution in time, iterationally speaking, starting from an initial noise status and modifying it during synthesis until reaching a satisfactory status. Moreover, even if during the optimisation phase, the synthesis is constrained only by the grey level histogram, after a while, besides the dynamics, the inner structures show orientations similar to the preliminary 2D texture.

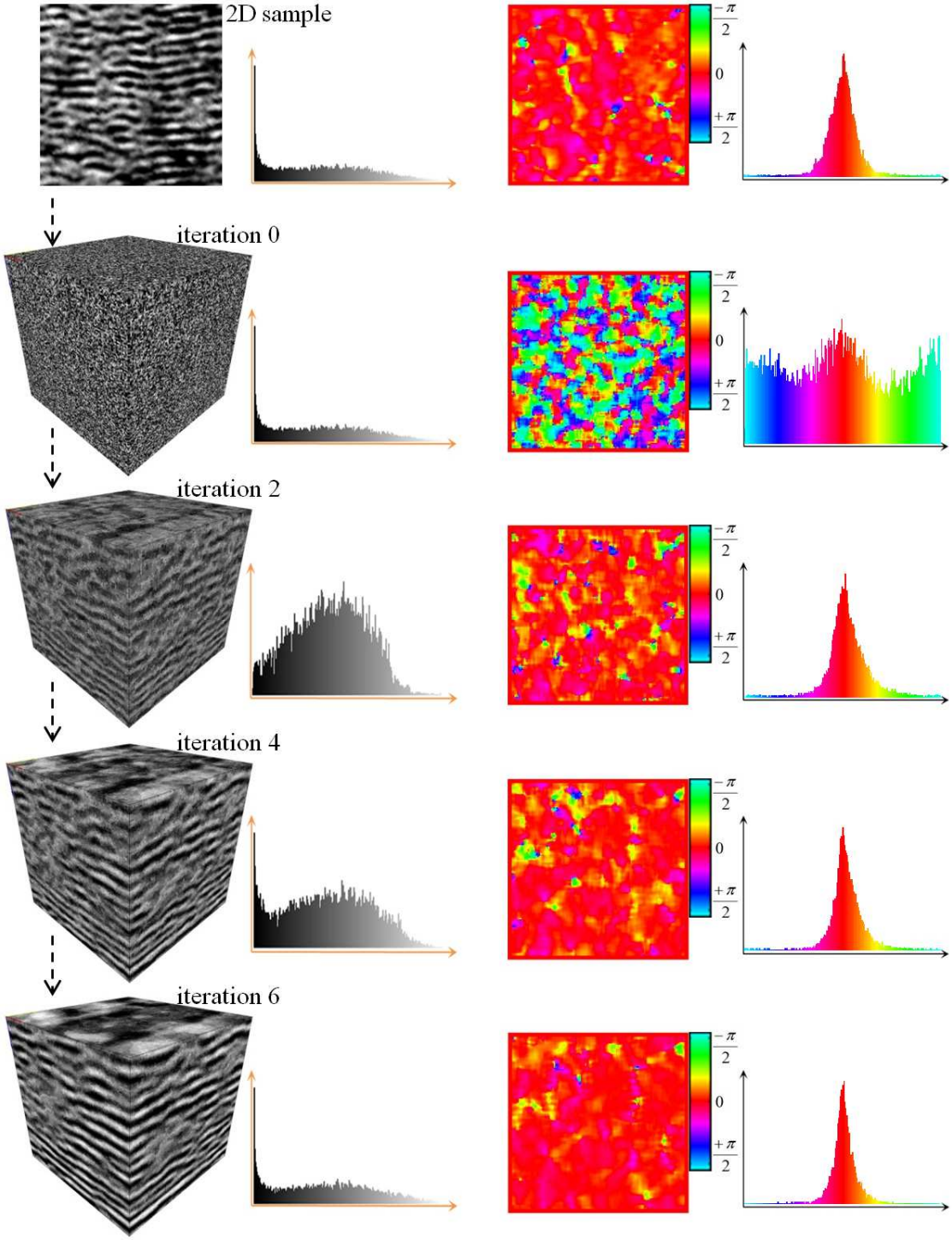


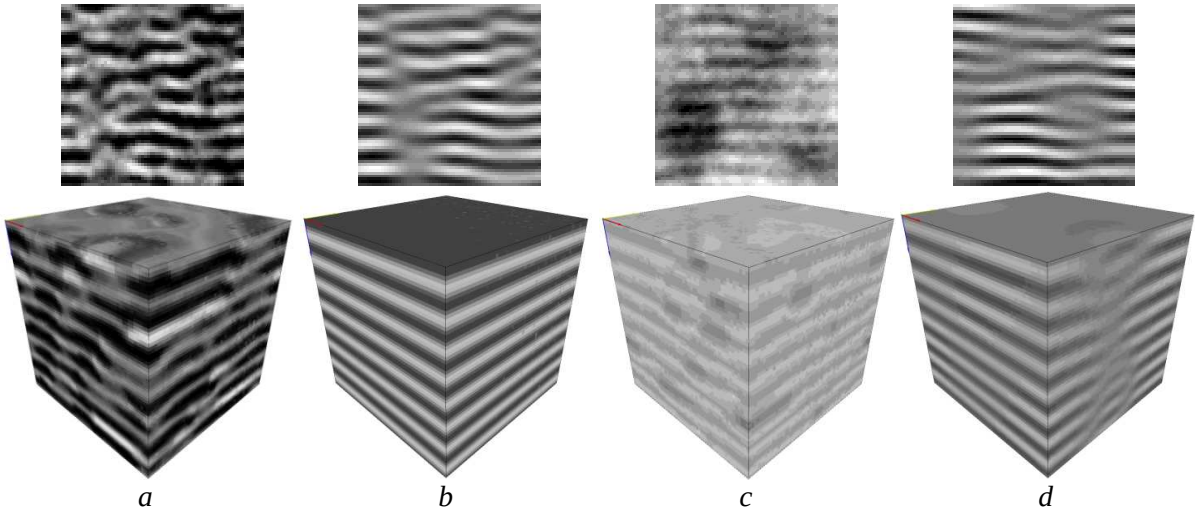
Figure 6.23 – *Texture evolution in time, using the backbone approach and only the grey level histogram matching: from top to bottom, the exemplar and the evolution of the initial noise, modified during the synthesis until reaching in a few iterations a satisfactory result from the point of view of the grey levels and the orientations. From left to right, the texture representation, the corresponding grey level histogram, the local orientations map and the orientation histogram*

Analyzing the comparative anisotropic texture results obtained by using the neighbourhood-search based algorithms, one can deduce that the most satisfactory method is NP_K, as it was already concluded in *Chapter 5* for a large variety of textures. For any further comparaisons in this chapter NP_K is going to be used as the representant of the non-parametric pyrocarbon texture synthesis methods based on fixed-neighbourhood search.

6.3.4 Maximum-likelihood based synthesis results

Going over the study from *section 5.3* corroborated with the orthotropic considerations, the algorithms described in *Chapter 4* are adapted to exploit two orthogonal views and implemented under the framework that proved to bring in the most satisfying results. The framework used to synthesize HRTEM samples consists in: a neighbourhood system composed of a 9×9 neighbourhood, three decimation scales, 1/8 up-sampling strategy, visiting the voxels in a random way and re-synthesizing the even pixels inherited from a lower resolution scale. The heuristic used in making the decision for an output voxel is NP_ML_H2c.

The inconvenient of the likelihood maximisation based synthesis approach is that the pixel temperature function incorporated with the deterministic ICM relaxation algorithm is difficult to forecast and, above all, leads to a time overwhelming process. Next, although the computational time was reduced by decreasing the temperature of a pixel with a -3 factor and stopping the synthesis after reaching 95% from the initial global temperature, the synthesis remains cumbersome for synthesizing big textures. Hence the input HRTEM samples used as source for synthesis are the 64×64 textures reduced to 32 grey-levels while the targeted blocks are of 64×64×64 pixels size, as it illustrated in *Fig. 6.24*.

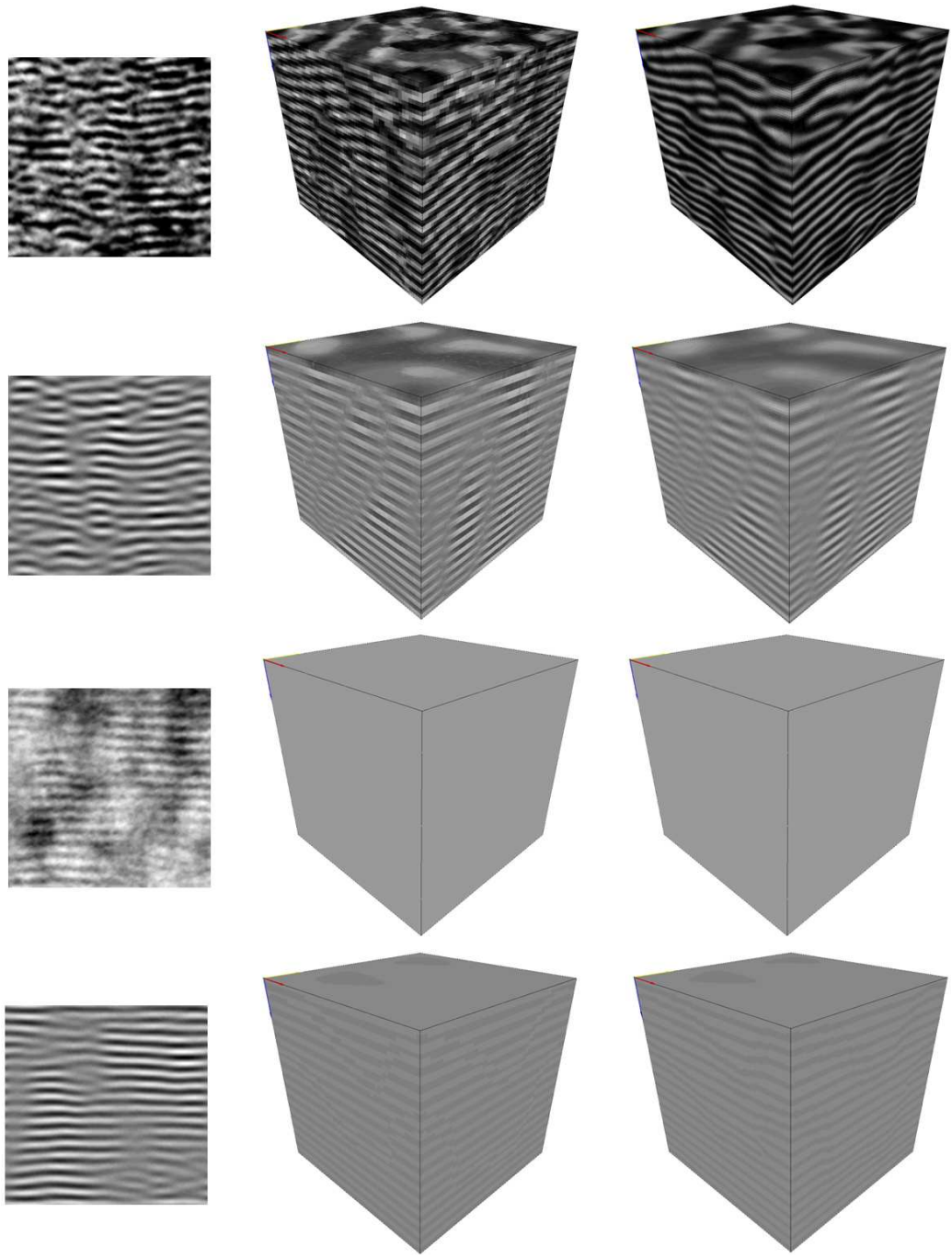


a *b* *c* *d*

Figure 6.24 – *Volumetric results obtained by using NP_ML_H2c approach on the lattice fringe samples of pre-graphitic carbons represented on top of each 3D synthetic texture. The HRTEM samples are cut out from the images in Fig. 6.10. Sample sizes are 64×64 pixels and the grey-level distribution contains 32 level.*

The results produced so far by the NP_ML_H2c algorithm using the above framework parameters are unsuccessful, only the result in *Fig. 6.24a* gives a brief impression of resemblance between the sample and the synthesis results. The results in *Fig. 5.24b-d* are not capable of replicating the sample structure. They seem to reproduce in the synthetic block only the most representative grey-levels of the 2D sample, giving the impression that the

synthesis behaves as a filtering operation, simplifying the texture. This opinion is sustained by the results presented in *Fig. 6.25*, showing the volumetric textures at the first and the last iteration of the final scale, obtained using the same framework but on bigger textures. The samples are the textures in *Fig. 6.10a,b,d,e* reduced to 32 grey-levels, and the 3D textures are results of the NP_ML_H2c algorithm of 128×128×128 size. The synthesis results are either too simplified in terms of structure and dynamics, either containing only the representative modes (i.e. grey-values), estimated during ICM relaxation.



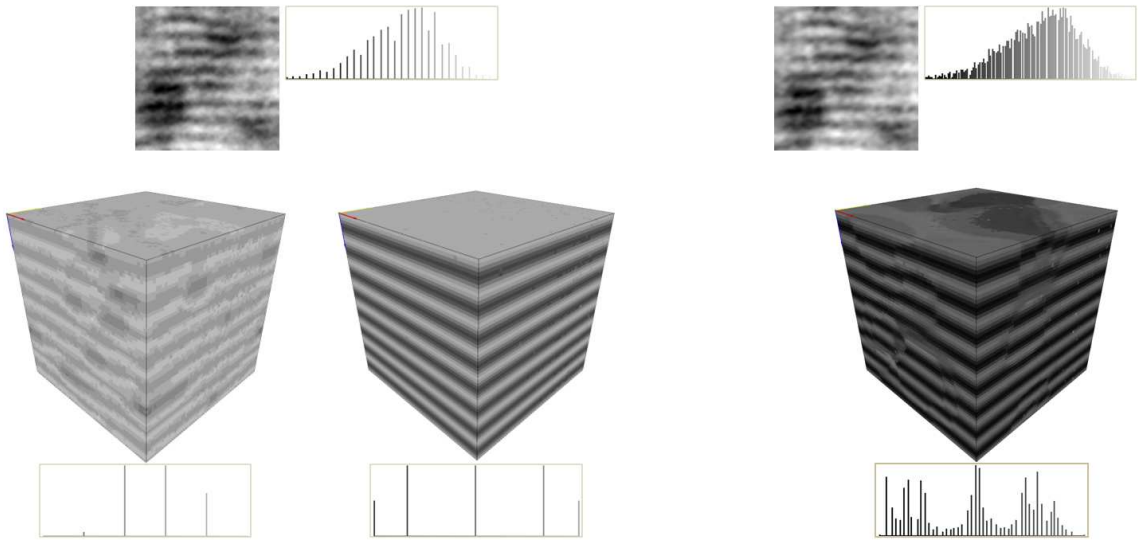
a *b* *c*

Figure 6.25 – *Volumetric results using as source for NP_ML_H2c synthesis the 128×128 samples from column (a), containing 32 grey-levels: the two columns show, from left to right, the 3D 128×128×128 texture obtained at the first iteration (b) and at the last iteration(c) of the final scale (experimentally, using a decreasing cooling rate of factor -3 and accomplishing 95% of the global temperature, the end result is achieved after four iterations).*

This inconsistency of the results may be explained by the scarcity of grey-levels, more accentuated at the smallest scale and the presence of relatively homogenous large areas of a same grey-value in the sample, that are not capable to provide enough diversity to be captured by the neighbourhood.

But using a bigger neighbourhood will increase even more the computation cost and as suggested by the results in *Fig. 5.21* and *Fig. 6.26a,b* it wouldn't guaranty the structure preservation.

Correspondingly, using numerous grey-levels (i.e. all the grey-levels available in the 2D sample) still seems to be inadequate to help the synthesis capture the pattern, as it was shown in *Fig. 5.22* and *Fig. 6.26c*, showing that it is not necessarily the number of grey-values that poses problems.



a *b* *c*

Figure 6.26 – *The influence of the neighbourhood size and the number of grey-levels on the synthesis results: from left to right, the first two result correspond to the synthesis of the HRTEM sample (64×64 and 32 grey-levels) from above the two blocks, in (a) using a 9×9 neighbourhood and in (b) a 11×11 neighbourhood; the result in (c) is obtained by using the full-square 9×9 neighbourhood but applied on the original HRTEM sample containing all the grey-levels; all the other parameters were fixed. Next to each texture, the corresponding grey-level histogram is shown.*

The deterministic pixel-temperature based process requires more precision in tuning its parameters, or to dispose of a sample texture of which structure is well represented on all the required resolutions.

Despite the fact that the maximum-likelihood based synthesis approaches showed their potential, but also its limits, at this current stage of implementation the volumetric textures corresponding to the HRTEM samples are un-exploitable in terms of the comparison study, relative to the fixed-neighbourhood search based techniques. However, suitable results are expected to be obtained by accelerating the algorithm (providing a parallelised synthesis process) and by ameliorating the relaxation scheme (by better using the pixel temperature function or by using a stochastic relaxation algorithm).

6.3.5 Parametric vs. non-parametric 3D texture synthesis methods

A real advantage of the comparison benchmark is that it is suitable to comparing different algorithms from the literature. In the case of the HRTEM textures used as synthesis

samples, it's highly interesting to dispose of several synthesis methods from which to choose the best one and integrated it inside the IGAR method.

This thesis deals with a class of non-parametric approaches, so a high interest is to be able to compare them with a different kind of synthesis approaches. Here the opponent approach is the parametric one, based on a synthesis-by-analysis procedure, presented in [*DaC10*] and used as a first step of the IGAR method for atomistic simulations [*Ley09*].

As shown in *section 6.3.3* the non-parametric approaches based on fixed-neighbourhood search provide satisfactory results, with notable performances setting apart the NP_K approach.

The remarks in *section 6.3.4* together with the experimentation in *Chapter 5* underline the limits of the non-parametric approaches based on likelihood maximisation, yielding the results inappropriate for the evaluation methodology proposed in *section 6.3.2*.

In consequence the further study involves only the NP_K non-parametric approach and the following parametric approach.

6.3.5.1 A parametric texture synthesis approach

The parametric approach taken into consideration is the 2D/3D synthesis/analysis extension of the parametric method proposed in 2D by Portilla and Simoncelli [*Por00*]. The 2D/3D extension of the pyramidal scheme of [*Por00*] was developed by [*DaC10*] and it is schematically presented in *Chapter 2* in *Fig. 2.16.* It is based on the idea that an image can be relevantly analysed through a bank of spatial filters, with specific orientations and scales.

The 2D/3D synthesis algorithm proceeds in three steps:

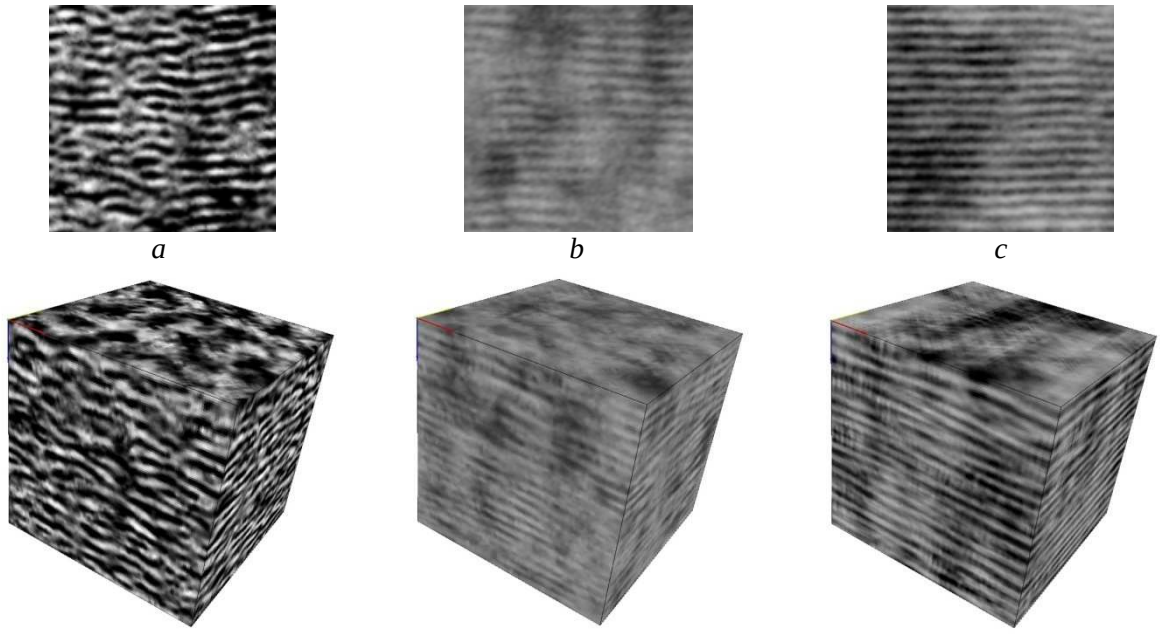
- *2D analysis step*: it consists in decomposing the HRTEM image sample into a set of multi-resolution sub-bands using steerable pyramid decomposition; a set of reference 2D spatial statistics is produced;

- *2D-3D statistical inference*: 3D reference statistics are inferred from the 2D ones using specific anisotropy assumptions;

- *3D image synthesis*: the 3D target statistics are imposed on the sub-bands of a random 3D data block.

More details about this approach and its implementation can be found in the original paper [*DaC10*].

The 3D synthesis method is applied on the raw and filtered samples from *Fig. 6.10*. The synthesis framework consists in decomposing the steerable pyramids with 3 levels and 4 orientations and the chosen target statistics being the mean, the variance, the skewness, the kurtosis and the autocorrelation coefficients (computed on 7×7 neighbourhoods). The 128×128×128 results are provided in *Fig. 6.27* for the raw samples and in *Fig. 6.28* for the filtered versions. For simplicity and to be in accordance with the other notations, this parametric approach will be tagged in as P_D&G.

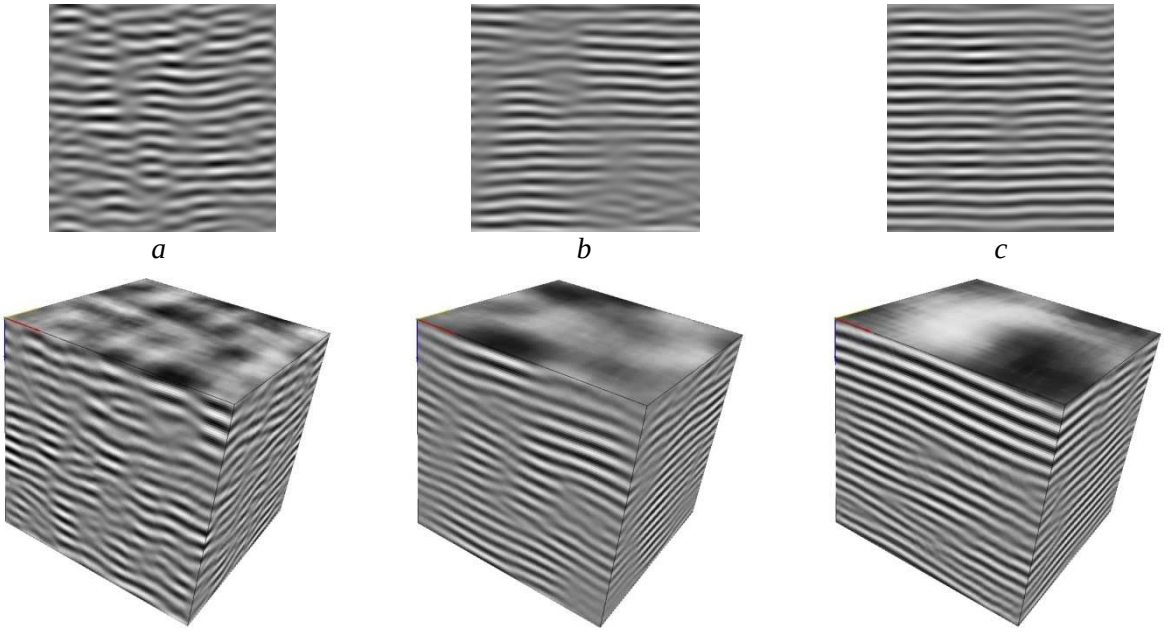


a *b* *c*

d *e* *f*

Figure 6.27 – *Parametric 3D synthesis results: the second row represents the 3D views of the solid textures synthesized from the raw 2D samples in the first row.*

The solid results obtained by synthesizing the raw HRTEM textures provide relatively convincing results (*Fig. 6.27d*), but some artefacts are found (the blocks from *6.27e* and *6.27f*) due to the way the autocorrelation coefficients are estimated on the sample images, assuming periodicity of the 3D block. As for the results obtained with the filtered textures, they seem to be free from circularity artefacts. The solid textures in *Fig. 6.28d* and *Fig. 6.28e* are alike the samples in the sense of straight/distorted patterns. These results are susceptible to some high frequency oscillations [*DaC10*] making for example the results in *Fig. 6.28f* to look as disordered as *6.28e*.



a *b* *c*

d *e* *f*

Figure 6.28 – *Parametric 3D synthesis results: the second row represents the 3D views of the solid textures synthesized from the filtered 2D samples in the first row.*

6.3.5.2 Results comparison

The comparison study is carried out between the results obtained with NP_K from *Fig. 6.17* and *6.18* and the results obtained with P_D&G from *Fig. 6.27* and *6.28* using as source of synthesis the HRTEM samples in *Fig. 6.10a, b, d* and *e*. More precisely the comparative pairs are the following:

- *6.17g* with *6.27d* for the raw HRTEM sample in *6.10a* (couple tagged as *C1*)
- *6.17h* with *6.27e* for the raw HRTEM sample in *6.10b* (couple tagged as *C2*)
- *6.18g* with *6.28d* for the filtered sample in *6.10d* (couple tagged as *C3*)
- *6.18h* with *6.28e* for the filtered sample in *6.10e* (couple tagged as *C4*)

The comparison is performed using the same methodology described in *section 6.3.2* with the mention that the 10th iteration of the synthesis process is used, mostly because NP_K needs more iterations to provide a satisfactory result. For P_D&G it is not necessarily the case, because it shows stable results early on (even from the 2nd iteration) and not varying significantly during eventual extra iterations.

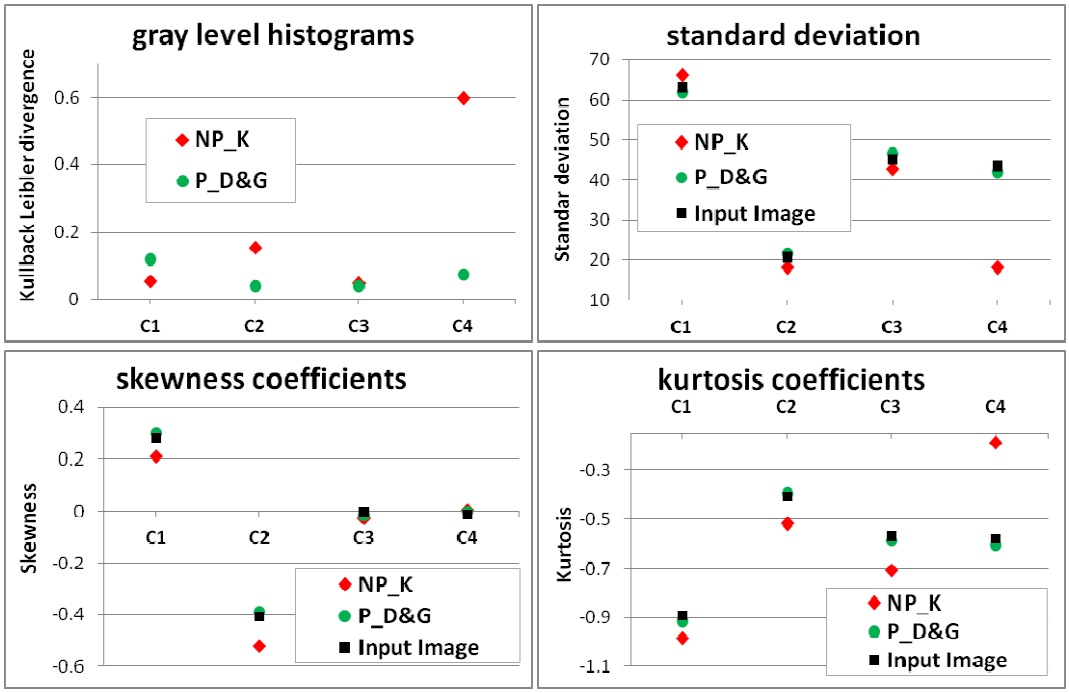


Figure 6.29 – *Indicators of the dynamics comparative study that concerns the couples C1 to C4, the colour markers labelling the synthesis methods.*

Fig. 6.29 compares the results in terms of grey-level dynamics, comparing punctually first order statistics of the NP_K results with the ones of the results obtained with P_D&G. All along these results, the parametric method provides 3D synthetic textures closer to the 2D samples in terms of grey-level histogram, and statistics very similar to the ones of the corresponding 2D samples. An exception occurs for the case of the C1 evaluation, showing that for using the raw HRTEM sample in *Fig. 6.10a*, NP_K behaves better than P_D&G being closer to the sample, in terms of grey-level histogram. NP_K results tend towards the 2D sample characteristics but even after 10 iterations there still remains a relative gap between them. However, this gap is not very significant, results being qualitatively similar, generally the grey levels distribution seeming to be evenly for the input sample, NP_K volumetric results and P_D&G solid texture.

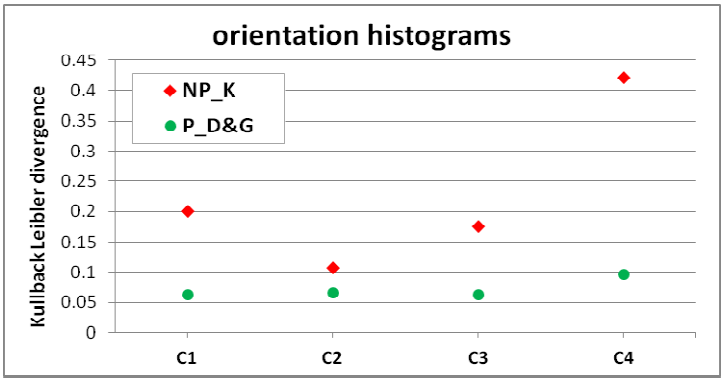
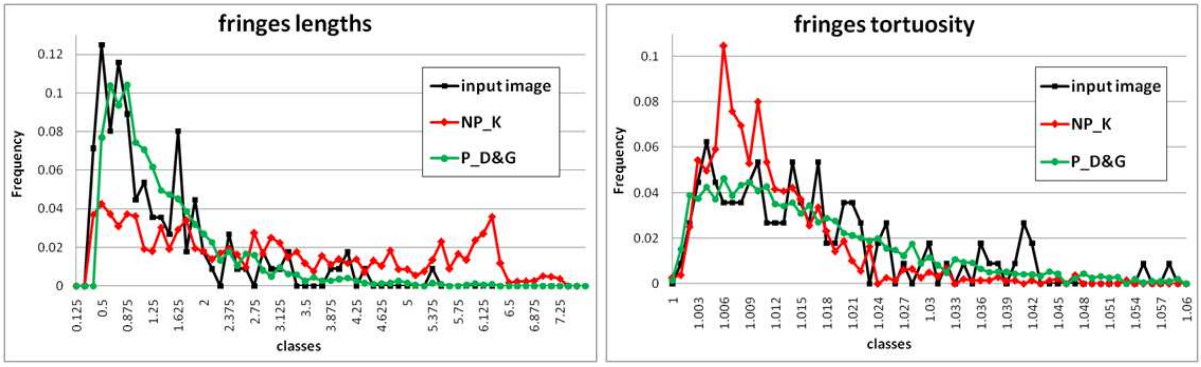


Figure 6.30 - *Spatial structure variation indicators: the Kullback - Leibler divergence between the orientation histograms of the exemplar and the output blocks obtained by applying the algorithms on the comparative pairs described as C1-C4.*

The comparative study is further on performed in order to check if the structural properties of the sample are retrieved in the volumetric textures. First, the local orientation maps are computed and orientation histograms are generated. A particular measure (i.e. Kullback-Leibler divergence) is employed between the orientation histogram of the HRTEM exemplar and the orientation histogram of the parametric synthesis result, and respectively the non-parametric synthesis texture. This is illustrated by the plots in *Fig. 6.30*, showing that the volumetric textures obtained with the parametric approach preserve better the local orientations of the exemplar patterns.



a *b*

Figure 6.31 – *Morphological structure indicators: the distribution of the fringe lengths and the tortuosity values in the input HRTEM image from Fig. 6.11a and in the multi-2D synthesized volumetric textures of the couple C1 (Fig. 6.17g for NP_K and Fig. 6.27d for P_D&G); fringes were retrieved by managing front and side slices in the 3D textures.*

The morphological properties comparison is interested in describing the elongated patterns contained within the texture evaluating the L_2 parameter and the τ parameter of the input lattice fringe image and the texture synthesized with the NP_K algorithm and the P_D&G algorithm. *Fig. 6.31* confirms that the parametric algorithm produces textures that contain similar fringes with the input image both in terms of length and degree of tortuosity. The P_D&G result patterns are shorter and less regular (as the 2D sample) than the patterns present in the NP_K result.

Drawing the line of the comparison study between the non-parametric algorithm based on fixed-neighbourhood search and the parametric synthesis-by-analysis algorithm, the latter one is more capable to produce volumetric textures with anisotropy properties similar to the 2D sample texture.

6.4 Conclusion

This applicative chapter introduced some necessary concepts of the composite materials of which HRTEM images were used for 2D/3D synthesis in a complex atomistic reconstruction scenario.

Afterward the algorithms described in the previous chapters are evaluated regarding to their capacity to reproduce 3D textures respecting the structures observed on the HRTEM images. A comparative study was developed in order to identify the most relevant non-parametric synthesis approaches for this particular HRTEM textures and to compare a representative non-parametric approach with a parametric algorithm from the literature. The non-parametric algorithms based on likelihood maximization are not suitable, in their present form, for this particular type of textures. However, this work leaves the perspective for a future improvement in terms of quality and computational requirements.

The quality of the results obtained with the non-parametric neighbourhood search based algorithm and predominantly with the parametric one is highly encouraging, the solid textures showing both statistical properties (anisotropy, contrast, etc.) and structural features (pattern lengths and disorientations) roughly similar to those observed in the original samples highlighted by the quantitative comparison between synthetic data slices and reference HRTEM samples. But in the current state of development, the parametric method, in spite of its drawbacks, seems to provide the most satisfying results.

Chapter 7

Conclusions

Contents

7.1 Synopsis of the work ……127

7.2 Future prospects ……128

7.1 Synopsis of the work

This section aims at summing up the results presented in this manuscript and at proposing overall conclusions derived from this thesis in a clear-cut form avoiding to restate the closing remarks of each chapter.

The main objective of this work was to investigate different strategies for volumetric texture synthesis starting from a 2D texture. The non-parametric synthesis methods have been privileged, setting off with the fixed-neighbourhood search based algorithms. The proposed approaches are assessed under a common multi-scale accelerated algorithmic benchmark, differentiating the way of combining the orthogonal views information. To make use of the already synthesized data and to assure certain randomness, we have proposed an original diagram for visiting the voxels during synthesis. Apart from its theoretical reasoning, only minor contributions in synthesis quality are noticed.

Next we have focused much of our attention on developing a probabilistic non-parametric algorithm based on a Markov Random Field conceptualisation. We have proposed a genuine 2D/3D extension intending to generate volumetric textures voxel by voxel by maximizing the likelihood of every voxel estimated in terms of a *local conditional probability density function* (LCPDF). Based on the 2D paradigm, our original 3D extension relies on giving a new description to the LCPDF of a site in 3D. Full-3D LCPDF estimation is almost unattainable leaving the occasion to formulate heuristics based on 2D concepts, thus deciding for each output voxel the most probable grey-level. Notable remarks are made on how to use better the voxel temperature function involved by the synthesis deterministic relaxation process. As well, we propose different strategies for the output block initialisation and for the scales handling policy – advising a specific treatment for the voxels inherited from a higher scale and two up-sampling strategies.

All these procedures have been materialized on a set of significantly different textures, identifying the strengths and the weaknesses of the synthesis algorithms. Moreover we have analyzed for two foremost methods, a fixed-neighbourhood search based one and a likelihood-maximization one, their sensitivity to different algorithmic parameters, tracing the best strategies for texture synthesis. The synthesis results are convincing in terms of visual quality, manifesting occasionally difficulties in capturing the sample structure. The maximum likelihood based approaches that comes up with extra parameters, concerning the pixel temperature function, sometimes proves to be difficult to control but is also capable of producing outstanding results when tuned properly.

Finally, we have applied the non-parametric algorithms to the synthesis of laminar anisotropic HRTEM images of dense carbons, the volumetric synthesis being the only real solution to access the 3D structure of such carbonaceous materials at nanometric scale. Results interpretation was accompanied by an original experimental procedure marking a quantitative and objective evaluation, investigating grey-level dynamics and morphological properties. The experimentation benchmark stressed the limits of our current implementation of the maximum-likelihood based approach that simplifies in terms of structure and dynamics the synthetic results relative to the 2D samples. The quantitative comparative study was broadened with a 2D/3D extension of a parametric synthesis-by-analysis approach, which proved to provide the most satisfying results.

7.1 Future prospects

The obtained results are convincing and looking very promising to be improved in the future. The future prospective targets are of three kinds: on first hand, enhancements in terms of preserving the sample structure and dynamics, on the second hand, to reduce the computational cost, and thirdly, to extend the synthesis algorithms to all kinds of textures, especially isotropic textures.

For the non-parametric fixed-neighbourhood search based algorithms immediate developments can be made regarding the investigation of other measures than the L2 norm (Euclidian distance); it is of interest to be able to inject during the synthesis process operations allowing to adjust better the dynamics and the statistical properties, remaining non-parametric. An interesting track to follow consists in inferring in the synthesis process not only the grey-level data but also morphological information such as local orientations for instance. To find for an output voxel the best grey-value, the judging criterion can be based on the closest neighbourhood search but taking account of the local orientation. This should assure a better preservation of the sample structure in the volumetric texture and would be appropriate to address the case of non-stationary isotropic textures (textures that show changes in orientation at large scale).

Additionally, it is always possible to propose different strategies to combine the information from the orthogonal views, hence to quantify the quality of the synthesized images.

The non-parametric maximum-likelihood based approach offers more opportunities. First of all, handling the pixel temperature function involved in the relaxation algorithm reveals to be tricky for the three-dimensional synthesis process. Adjusting more accurately the algorithmic parameters can lead to better results. In any case the cooling schedule and the deterministic algorithm need a more intricate analysis. Stochastic relaxation schemes such as the Metropolis algorithm, the Gibbs sampler or the *simulated annealing* algorithm could also be investigated.

In the same idea, a full-3D estimation of the local conditional probability density function could be of help; if this is not possible, other heuristics are as well welcomed. The goal is that by using the Markov Random Field probabilistic model, the algorithm can capture the visual characteristics of an image, providing a statistical model capable to describe the visual interactions between adjacent voxels.

The computational complexity is not to be neglected, being known that the non-parametric voxel-by-voxel methods are time and memory consuming, involving exhaustive operations. For the neighbourhood search based algorithm, acceleration is attained by using the binary tree clustering, but the maximum-likelihood based approach is computationally very expensive. Accelerating it can be achieved by multithreading/multitasking programming and by relaxing simultaneously a set of i.i.d. (i.e. independent and identically distributed) sites from the targeted block lattice grid.

Concerning the applicative part, the means to generate extensively HRTEM textures are strongly correlated to the previous remarks, while constantly providing volumetric textures for the atomistic simulations. To deliver textures of more important sizes, the time calculation is crucial. In addition, some materials showing locally anisotropic structure but non-stationary could be examined by means of the non-parametric synthesis methods.

Appendix A

TSVQ numeric exemplification

The acceleration tricks used by the synthesis techniques proposed in *Chapter 3* are based on re-arranging the input neighbourhoods into a binary tree to ease the nearest neighbour search. This is done by using the Tree Structure Vector Quantization (TSVQ) technique described in *section 3.2.5*. The root of the tree contains the global centroid of all *n*-dimensional points and the rest of the nodes are centroids of portioned groups of points.

The next three figures show some numeric examples of the TSVQ technique applied on three images of size 6×6 pixels and using a 3×3 neighbourhood. According to the synthesis algorithm, the pixels used in the binary tree are only the pixels that have a valid neighbourhood. The area in red of size 4×4 is the only one taken into consideration by the TSVQ. For simplicity in showing the binary tree, the nodes contain only the pixel value (i.e. the average of the pixels in the corresponding group) and not all the neighbourhood data information. The first example in *Fig. A.1* shows the binary tree obtained on an image containing only two grey-levels, presenting only two distinctive neighbourhoods. *Fig. A.2* contains an image of 16 grey-levels, one for each valid pixel.

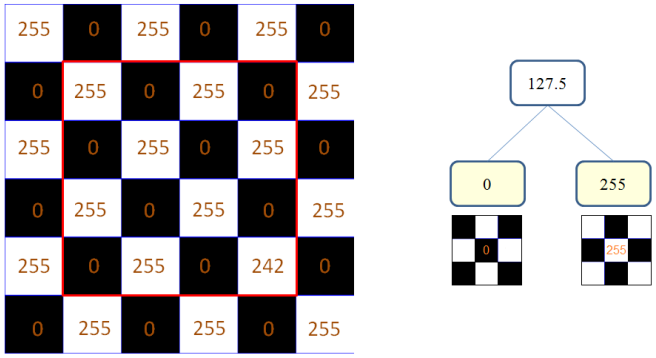


Figure A.1 – *TSVQ exemplification on a 6x6 binary checkerboard image showing the final two nodes and their corresponding neighbourhoods.*

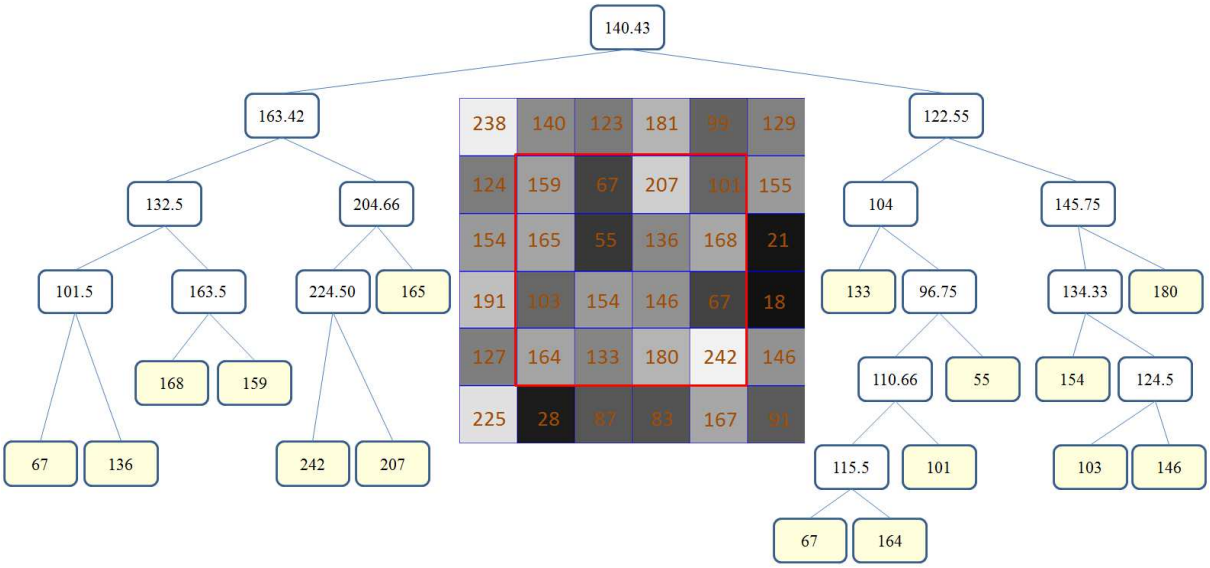


Figure A.2 – *TSVQ exemplification on a 6x6 image containing 16 grey-levels.*

Appendix B

Space filling curves

B1. Introduction

The interest in space filling curves started early from 1890 with the work of Giuseppe Peano [*Pea90*] and David Hilbert [*Hil91*]. A space-filling curve is used to map discrete positions from a multi-dimensional grid into a one-dimensional grid, by visiting points only once and without crossing the path [*Pea90*]. The first implementation consisted in a geometric representation capable of traversing a $2^n \times 2^n$ space [*Hil91*].

Since then, many developments were made, so that space filling curve representation found applications in image pixel allocations, VLSI component layouts, magnetic resonance imaging (MRI) etc. In the same time many mathematical formulations were made. So, one can retrieve the Peano curve, the Hilbert curve, the Z-curve (Lebesgue curve) or the fractal curves (Cantor, von Koch, Sierpinski or Mandelbrot) and its many different variants [*Sag94*].

B2. Lexicographical scanning and the random walk

In this work I was interested in proposing alternative ways to the traditional lexicographic scanning (scan-line order) and the random-walk in the purpose of 3D texture synthesis. These two kinds of points visiting order are presented just below:

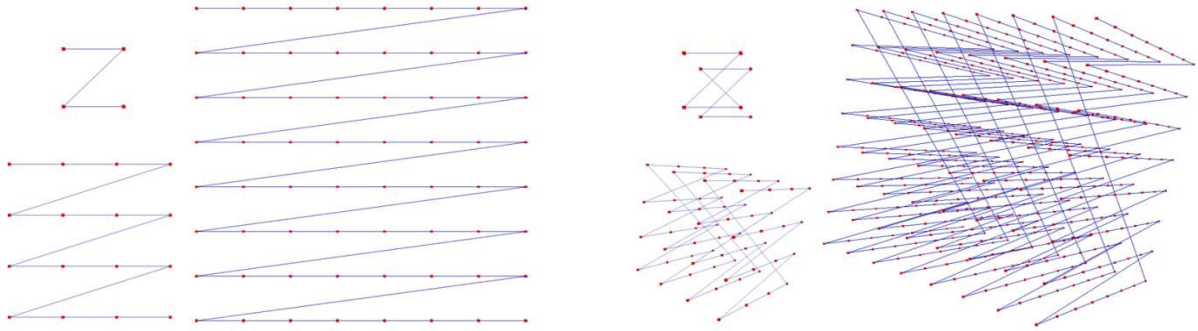


Figure B.1 – *Illustration of the scan-line order: the group on the left shows the 2D case on three square configurations containing 2×2 points, 4×4 points and 8×8 points; the group on the right shows the scan-line path on three cubes of different sizes containing 2^3, 4^3 and 8^3 points.*

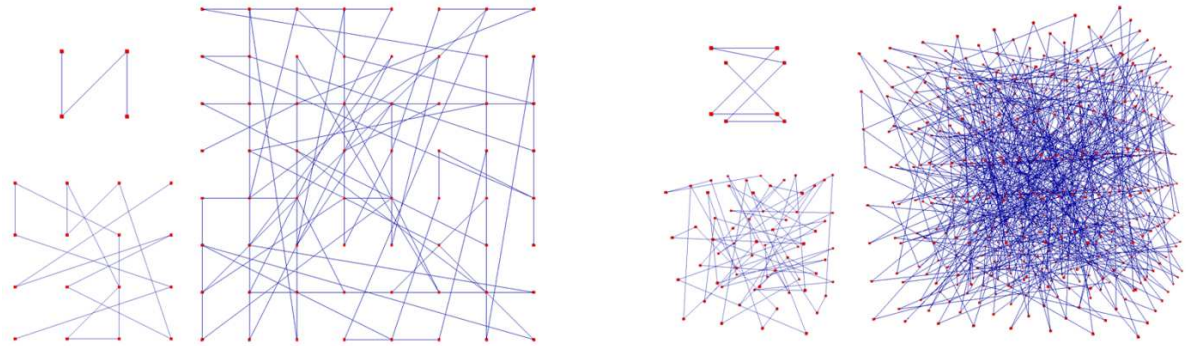


Figure B.2 – *Illustration of the random-walk: the group on the left shows realizations of random paths in the 2D case on three square configurations containing 2×2 points, 4×4 points and 8×8 points; the group on the right shows realizations of random paths on three cubes of different sizes containing 2^3, 4^3 and 8^3 points.*

These are the simplest to implement, but not exactly the most satisfying ones in the interest of 3D texture synthesis. The lexicographic path tends to produce textures more regular than the synthesis exemplar, while the random walk can lead to prohibitive convergence costs, in the context of using non-causal full-square neighbourhoods.

Intermediary solutions consist in maintaining the best out of the two – keep a certain randomness in order to assure diversity and in the same time use partially the information from a previously synthesized voxel. To acquire these two types of points visiting orders are proposed, in order to traverse once the points and without crossing the path – the Z-curve and the Hilbert curve.

B3. Z-curve

The Z-curve or Lebesgue curve is one of the simplest filling curve algorithms, sorting the points in a *Z*-shape order. A sequence of Z-curves is obtained by starting with a basic Z-shape and the rest of the curves are generated sequentially one from another using the same principle.

This is better illustrated in *Fig B.3*, where three iterations of the Z-curve are presented in 2D and 3D. It's easier to understand the principle in 2D: first the basic shape containing four points is constructed; next make four copies of the same shape and connect the last point of one shape to the first point of the next one and so on; the new obtained bigger realization is recopied, the process is repeated, generating in the end a non-self-intersecting plane-filling curve. Off course, one can use another way to connect the initial four points and to obtain other configurations.

The three-dimensional Z-curve is obtained similarly, starting with eight points connected as two parallel 2D basic z-shape configurations.

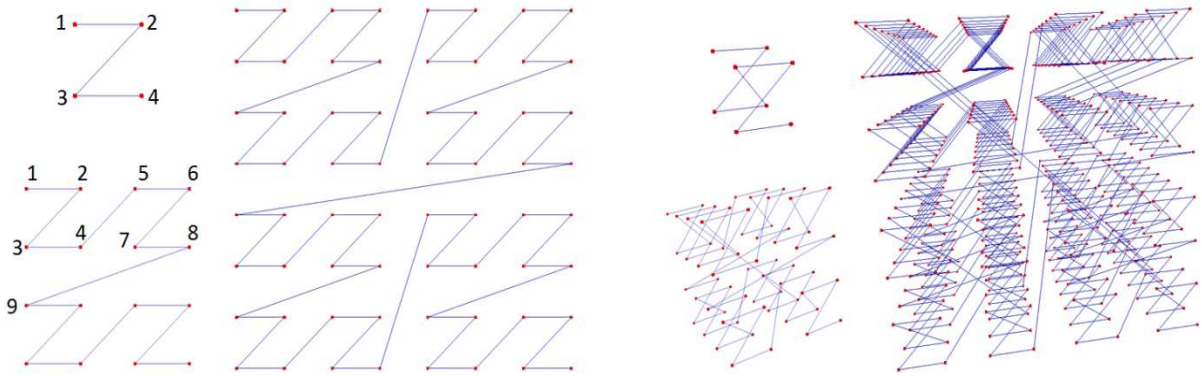


Figure B.3 – *Illustration of the Z space filling curve: the group on the left shows the 2D space case on three square configurations containing 2×2 points, 4×4 points and 8×8 points; it shows the basic four points z-shape and the basic shape repetitions in order to obtain o bigger Z-shape; similar to the 2D case, the group on the right shows the Z-order path on three cubes of different sizes containing 8, 64 and 512 points.*

The Z-value of a point in multi-dimensions is obtained by simply interleaving its binary coordinates, the resulting Z-values being connected recursively in their numerical order. A representation of these binary operations is showed in *Fig. B.4*.

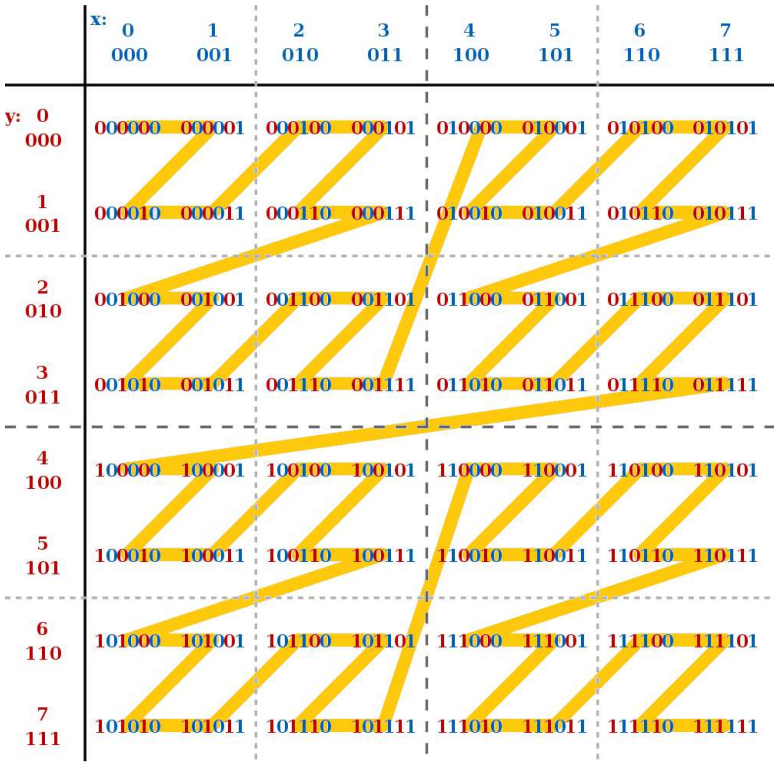


Figure B.4 – *The binary coordinates interleaving schema for obtaining the points of a two-dimensional Z-curve representation.*

B4. Hilbert curve

The Hilbert curve resembles in principle with the Z-curve. The differences appear in the fact that the way of connecting the 4 initial pixels in the 2D case respects another arrangement, while the update from a phase to another is not done by direct copying but by copying and rotating the basic 4 points-shape. The same connection between a last point and a next shape initial point is done.

The 2×2 (*H1*), the 4×4 (*H2*) and the 8×8 (*H3*) 2D Hilbert space-filling curves are shown in *Fig. B.5*. *H2* is a curve connecting four copies of *H1* in different orientations.

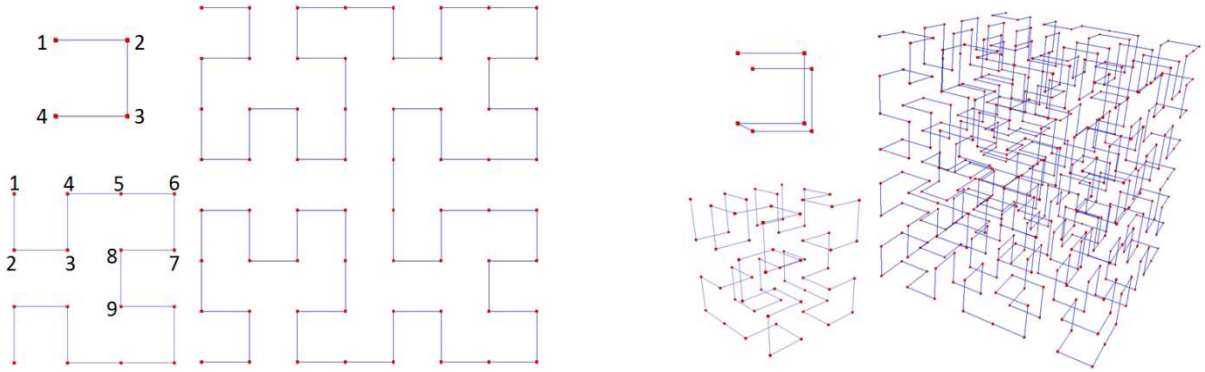


Figure B.5 – *Illustration of the Hilbert space filling curve: the group on the left shows the 2D case on three square configurations containing 2×2 points, 4×4 points and 8×8 points; the connexions are as in the case of the 2D Z-curve but the initial basic shape is different; the group on the right shows the Hilbert curve in a three-dimensional space exemplified on three cubes of different sizes containing 2^3, 4^3 and 8^3 points.*

However, the analytical form of the Hilbert curve is more intricate. The 3D extension follows the 2D principle but increases the complexity. Several algorithms have been proposed to build a Hilbert-curve, the best implementations being based on using bitwise operations or by using recursive procedures of rewriting systems (like the L-systems). The latter one can be explained by considering a plotter whose pen can move in several directions (up, down, right, left and even other intricate guidelines for the 3D case).

For my case, I choose to implement the Hilbert space filling curve using the algebraic operations proposed by [*Che04*]. These operations include tensor matrix product (for building a bigger matrix from two small matrices), direct sum and three permutations – stride permutation, vector reversal and Gray permutation. The objective is to draw a curve on a $2^n \times 2^n \times 2^n$ cube and to visit all the points without crossing the curve so that any two connected points have the Hamming distance equal to 1. Typically, a $2^n \times 2^n \times 2^n$ Hilbert space-filling curve is recursively constructed from eight congruent $2^{n-1} \times 2^{n-1} \times 2^{n-1}$ sub-cubes connected in different orientations. Five orientations obtained by coordinate transformations, indexed from I to V, and the corresponding sub-cubes are presented below in *Fig. B.6*. The sub-cubes are connected in the order of the $2 \times 2 \times 2$ Gray permutation: it maps the index sequence $(0, 1, 2, 3, 4, 5, 6, 7)$ to $(0, 1, 3, 2, 6, 7, 5, 4)$.

Type	Sub-cube #	Coordinate transformation
I	0	$(I, J, K) \Rightarrow (K, I, J)$
II	1, 2	$(I, J, K) \Rightarrow (J, K, I)$
III	3, 4	$(I, J, K) \Rightarrow (I, -J, -K)$
IV	5, 6	$(I, J, K) \Rightarrow (-J, K, -I)$
V	7	$(I, J, K) \Rightarrow (-K, -I, J)$

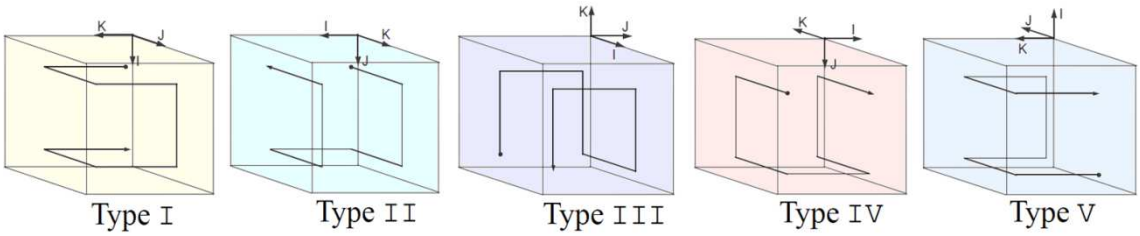


Figure B.6 – *Sub-cube types of the 3-D Hilbert space-filling curve and their coordinate transformations as suggested by [Che04].*

To summarize, building a 3-D Hilbert space filling curves is done by following the steps: block allocation, gray permutation, coordinate transformation and recursive construction. The reader can retrieve the iterative and the recursive algebraic operations involved by these steps in the original paper of [*Che04b*].

Appendix C

MRFs theory

The following concise theoretical form deals with the concept of Markov Random Fields and its equivalence with Gibbs Random Fields complementing the notional aspects in *Chapter 4*. It represents a condensed adaptation from [*LiS01*].

C1. Markov Random Fields

A family $F=\{F_1, F_2 \dots F_m\}$ is a random field if F is a family of random variables defined on the set of sites *S*, a particular case being when each variable F_i takes a value from a finite set of labels *L*. The sites-and-labels concept is applicable. $F_i = f_i$ means the event for which the random variable F_i takes the value f_i and the set of labels $f=\{f_1, f_2 \dots f_m\}$ corresponds to the configuration of F such that $\{F_1 = f_1, F_2 = f_1 \dots F_m = f_1\}$ or simply $F = f$ is a joint event. $P(F_i = f_i)$ or simply $P(f_i)$ represents the probability that a random variable F_i takes a value f_i. The joint distribution stands for $P(F = f)$.

A random field F is considered to be a Markov Random Field (MRF) on the set of sites *S* with respect to the neighbouring system *N* (i.e. the set of all neighbourhoods N_s) if and only if it satisfies the conditions of *positivity* - $P(f) > 0$ and *markovianity* - $P(f_i | f_j, j \neq i) = P(f_i | f_{N_i})$, where f_{N_i} represents the set of labels at the sites neighbouring *i*. The markovianity property describes the local characteristics of the random field, describing the local interactions between labels (i.e. LCPDF - *local conditional probability density function*).

C2. Gibbs Random Fields

A random field F is considered to be a Gibbs Random Field (GRF) on the set of sites *S* with respect to the neighbouring system *N* if and only if its configurations respect a Gibbs distribution:

$$P(f) = \frac{1}{Z} e^{-\frac{1}{T}U(f)} \qquad (c.1)$$

where

$$Z = \sum_f e^{-\frac{1}{T}U(f)} \qquad (c.2)$$

is a normalizing constant, T is a constant called the temperature and $U(f)$ is the energy function [*Gem84*].

To complete the definition, the notion of clique *c* intervenes [*Gem84*] [*Pag98*], defining a complete subgraph of (*S*, *N*) or more exactly a subset of sites in *S* that are all neighbours to one other.

Every site is a single-site clique; a pair of neighbouring sites defines a pair-site clique; and so on. The collection of all cliques is denoted $\mathcal{C}$.

The energy can be formulated as the sum of all clique potentials $U_c(f)$ over all possible cliques *c*:

$$U(f) = \sum_{c \in \mathcal{C}} U_c(f) \qquad (c.3)$$

The value of $U_c(f)$ depends on the local configuration on the clique *c* so that $P(f)$ is capable of describing the probability of reaching a specific configuration depending on the global measurements of local potentials.

C3. Markov-Gibbs Equivalence

An MRF is characterized by its local property (markovianity), whereas a GRF is characterized by its global property (Gibbs distribution).

The Hammersley-Clifford theorem [*Cli90*] [*Gem91*] establishes the equivalence of these two types of properties. The theorem states that for a given neighbourhood system *N*, a random field *F* is an MRF on *S* with respect to *N* if and only if *F* is a GRF on *S* with respect to *N*.

The MRF-Gibbs distribution equivalence theorem gives form to the joint probability of a MRF by expressing it in terms of clique potential functions:

$$P\left(f_i \middle| f_j, j \in N_i\right) = \frac{1}{Z_i} e^{\sum_{c \in \mathcal{C}_i} U_c(f)} \qquad (c.4)$$

where Z_i is a local normalizing constant $Z_i = \sum_{k=1}^{m} P\left(\lambda_k \middle| f_j, j \in N_i\right)$ over the finite state space $L = \{\lambda_1, \ldots, \lambda_m\}$ and the exponential sum is over the set of the local clique $\mathcal{C}_i = \{c \in \mathcal{C}, i \in c\}$.

Complete demonstrations are found in the MRF dedicated papers, indicated here as bibliography [*Bes74*] [*Gem84*] [*Cli90*] [*Gem91*].

Appendix D

Stochastic relaxation algorithms

The non-parametric synthesis algorithm based on the maximum-likelihood estimation described in *Chapter 4* uses as relaxation algorithm the ICM deterministic one proposed by [*GEM84*]. But more known and globally more utilized are the stochastic algorithms. It is the case of the Metropolis algorithm described in *Fig. D.1* and the case of the Gibbs sampler algorithm presented in *Fig. D.2*.

→ set up a random visiting order of the sites $\{a_k \in S, k = 1,2,...\}$
so that $s \in S, a_k = s$ infinitely often

→ choose randomly $x(0)$

→ for k = 1 to ∞ do

→ choose randomly $\lambda_{a_k} \in \Lambda$

→ get $p = \min\left\{1, \dfrac{P(\lambda_{a_k} | x_r(k-1), r \in N_{a_k})}{P(\lambda_{a_k}(k-1) | x_r(k-1), r \in N_{a_k})}\right\}$

→ choose randomly $q \in [0-1)$

→ image $x(k)$ is obtained from the previous image $x(k-1)$ such that

$$x_s(k) = \begin{cases} \lambda_{a_k} & \text{if } s = a_k \quad \text{and} \quad p > q \\ x_s(k-1) & \text{otherwise} \end{cases}$$

→ end for

Figure D.1 – *The steps involved by the Metropolis algorithm.*

→ set up a random visiting order of the sites $\{a_k \in S, k = 1,2,...\}$
so that $s \in S, a_k = s$ infinitely often

→ choose randomly $x(0)$

→ for k = 1 to ∞ do

→ choose $\lambda_{a_k} \in \Lambda$ from the distribution $P(\lambda_{a_k} | x_r(k-1), r \in N_{a_k})$

→ image $x(k)$ is obtained from the previous image $x(k-1)$ such that

$$x_s(k) = \begin{cases} \lambda_{a_k} & \text{if } s = a_k \\ x_s(k-1) & \text{otherwise} \end{cases}$$

→ end for

Figure D.2 – *The steps involved by the Gibbs Sampler.*

Appendix E

TEM imaging

The principle of the transmission electron microscope is an optical analogue to the conventional light microscope (LM) as illustrated in *Fig. E.1*.

An illumination source in the form of an electric beam is focused on the specimen (the pyrocarbon sample) by the condenser electromagnetic lens. The resulting image of the specimen is modified through the objective and the projector lens. The electron image is transformed into a visible representation on a fluorescent screen, and saved into the computer as a black-and-white image.

Areas that scatter few electrons (electron-lucent areas) appear as bright areas in the image whilst areas that scatter more electrons or absorb electrons (electron-dense areas) appear as dark areas. For TEM, the specimen has to be transparent for the electron beam, i.e. thin enough to get an image due to the transmission and the scattering of the electron beam through it.

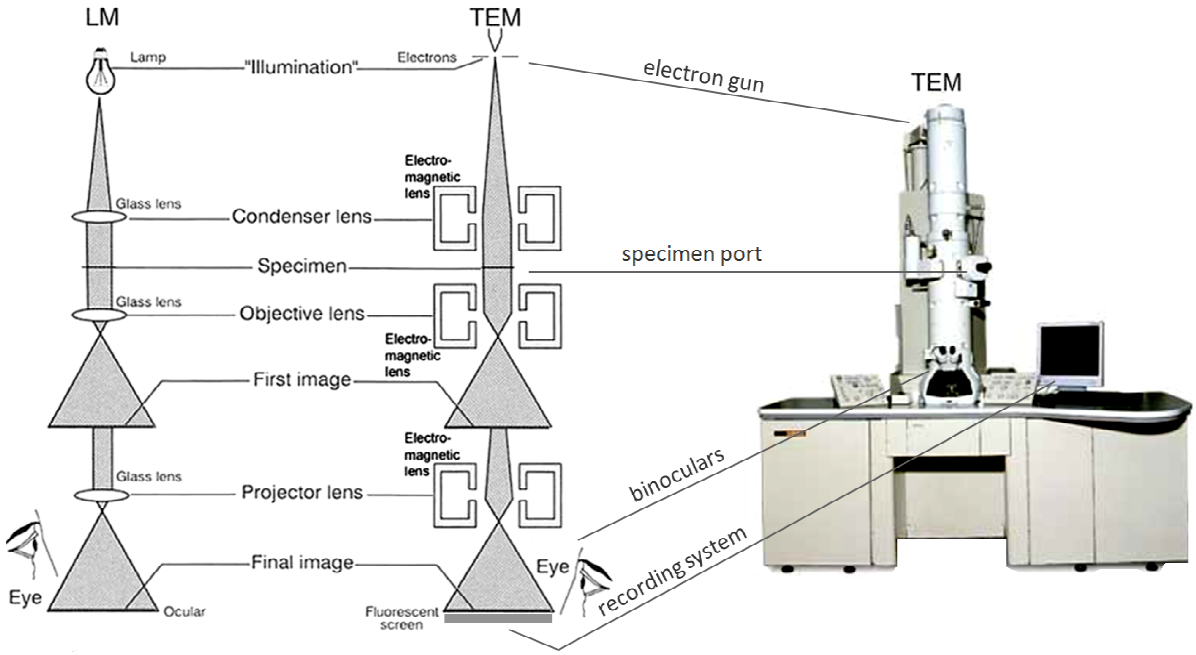


Figure E.1 – *Schematic representation of transmission electron microscope and its analogy to the conventional light microscope.*

Mathematically, the diffraction of electrons by a solid corresponds to the Fourier Transform of the specimen atomic configuration, automatically accomplished by the electromagnetic lens. The resulted electrons give to the image the atomic arrangement of the specimen and by using the network fringe technique the greatest intimacy (scale of angstroms) of the specimen concerning the orientation, the spacing and the length of the reticular layers is achieved.

A detailed description of the TEM techniques exceeds the purpose of this work, but the interested reader is kindly encouraged to go over [*Des93*] and [*Ncem*].

Appendix F

Some large snapshots of HRTEM samples

This appendix contains 3 pairs of large HRTEM images, from which smaller patches are being taken and used as source of synthesis in *Chapter 6*. Each of the following three pages, starting with the present one, contains two images: on top the raw version and on bottom the filtered version. The following images are of size 1024×1024 pixels.

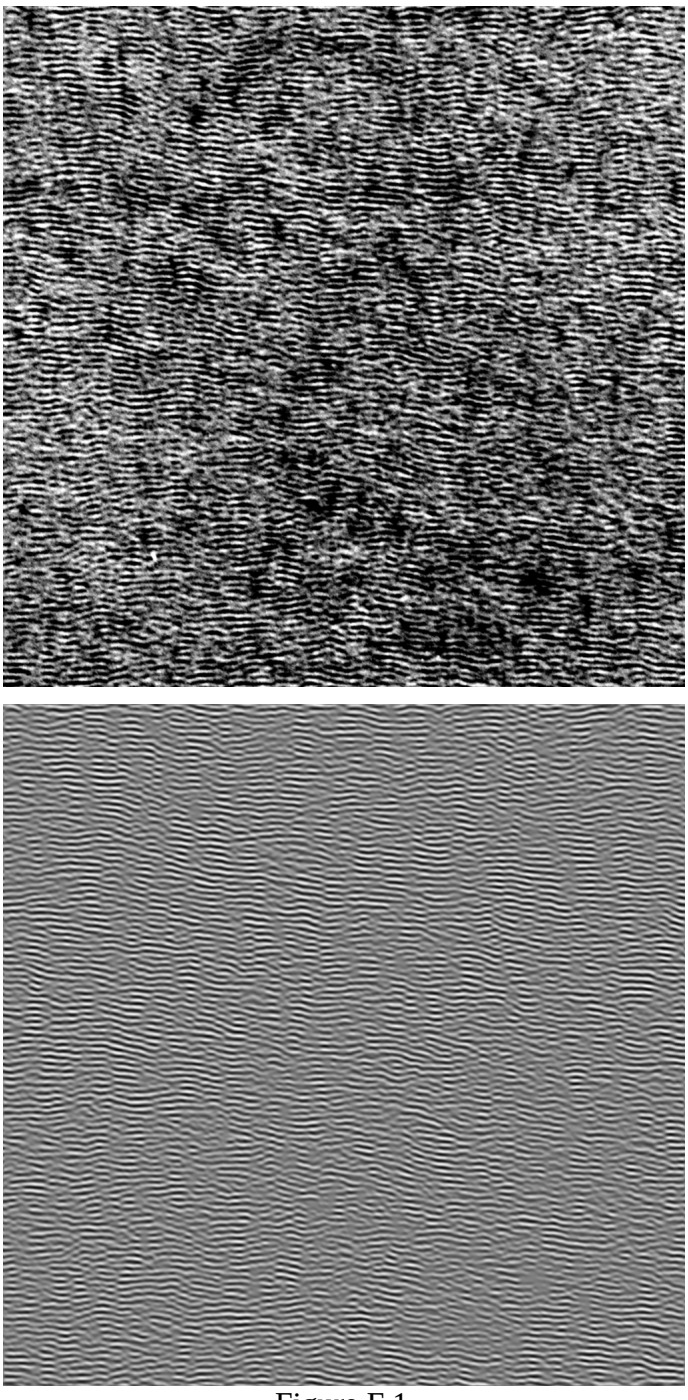


Figure F.1

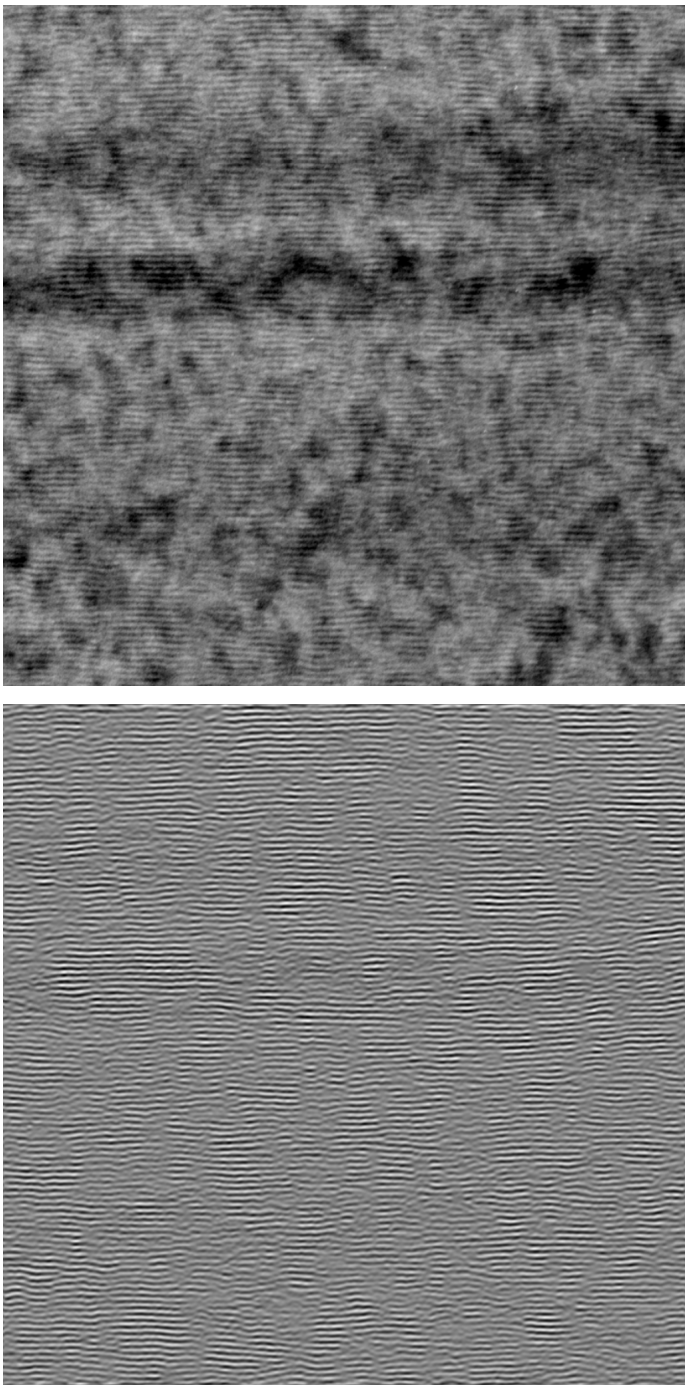


Figure F.2

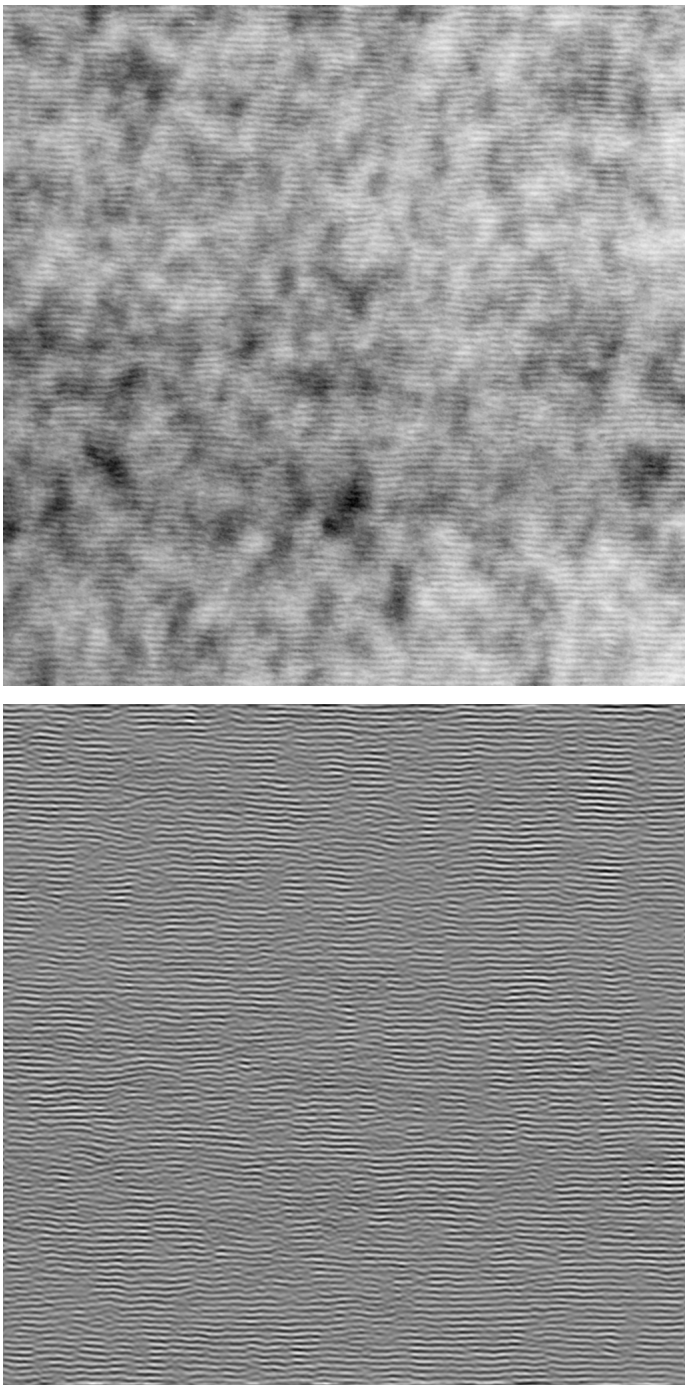


Figure F.3

Bibliography

[Ala98] O. Alata. *Caractérisation de textures par coefficients de réflexion 2D. Application en classification et segmentation*. Thèse N°1823, Université Bordeaux 1, 1998.

[Ash01] M. Ashikhmin. *Synthesizing natural textures*. In SI3D: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 217-226, New York, NY, USA, ACM Press, 2001.

[Bee96] A. C. Beers, M. Agrawala and N. Chaddha. *Rendering from compressed textures*. Proceedings of SIGGRAPH, pp. 373–378, August 1996.

[Bes74] J. E. Besag. *Spatial interaction and the statistical analysis of lattice systems*. In Journal of the Royal Statistical Society, series B, vol. 36, pp. 192-326, 1974.

[Bes86] J. E. Besag. *On the statistical analysis of dirty pictures.* Journal of The Royal Statistical Society, vol. B-48, pp. 259-302, 1986.

[Big91] J. Bigun, G. Granlund and J. Wiklund. *Multidimensional orientation estimation with applications to texture analysis and optical flow*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(8): 775-789, 1991.

[Bla07] R. Blanc. *Apport des statistiques spatiales à l'élaboration de critères d'homogénéité et à l'inférence en analyse de textures. Application à la caractérisation de matériaux.* Thèse N°3377, Université Bordeaux 1, 2007.

[Bou91] C. Bouman and B Liu. *Multiple resolution segmentation of textured images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13 -2, pp. 99-113, 1991.

[Bou93] X. Bourrat, A. Oberlin and R. Bachelard. Carbon, vol 31, issue 2, pp. 287-302, 1993.

[Bou02] X. Bourrat, A. Fillion , R. Naslain, G. Chollon and M. Brendle. *Regenerative laminar pyrocarbon*, Carbon, vol 40, issue 15, pp 2931-2945, 2002.

[Bou06] X. Bourrat, J. -M. Vallerot, F. Langlais and G. Vignoles. *La croissance des pyrocarbones*. Composites thermo-structuraux, l'actualité chimique n° 295-296, mars-avril 2006.

[Bro66] P. Brodatz. *A Photographic Album for Artists and Designers*. Dover, New York, 1966.

[Bro02] S. Brooks and N. Dodgson. *Self-similarity based texture editing*. Proceedings of ACM SIGGRAPH, Volume 21(3), pp. 653-656, July 2002.

[Bur83] P. J. Burt and E. H. Adelson. *A Multiresolution Spline With Application to Image Mosaics*. ACM Transactions on Graphics, 2(4), pp.217-236, 1983.

[Cas02] V. Castelli and L. D. Bergman. *Image Databases: Search and Retrieval of Digital Imagery*. Wiley, New York, 2002.

[Che85] R. Chellappa and R. L. Kashyap. *Texture synthesis using 2-D noncausal autoregressive models*. IEEE Transactions on Acoustics, Speech, and SignalProcessing, vol. ASSP 33, no. 1, pp. 194-203, 1985.

[Che93] R. Chellappa, R. Kashyap and B. Manjunath. *Model based texture segmentation and classification.* Handbook of Pattern Recognition and Computer Vision, World Scientifc Publishing, pp. 277–310, 1993.

[Che98] H. Chen, L. F. Pau and P.S.P Wang. *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, chapter 2.1, pp. 207-248, World Scientific Publishing Co., 1998.

[Che04a] Y. Chen and H.-S. IP. *Texture evolution: 3D texture synthesis from single 2D growable texture pattern*, the Visual Computer, Vol. 20, No. 10, pp. 650-664, Dec. 2004.

[Che04b] C. -S. Chen, S. -Y. Lin, and C. -H. Huang. *Algebraic formulation and program generation of three-dimensional Hilbert space-filling curves*. In Proceedings of the International Conference on Imaging Science, Systems, and Technology, pp. 254–260, 2004.

[Che10] J. Chen and B. Wang. *High quality solid texture synthesis using position and index histogram matching*. The Visual Computer, vol.26, pp. 253-262, 2010.

[Chi06] J. W. Chiou and C. K. Yang. *A New Algorithm for Solid Texture Synthesis*. Lecture Notes in Computer Science 4292/2006 (Advances in Visual Computing), pp. 780-789, 2006.

[Cla87] M. Clark, A. C. Bovik and W. S. Geisler. *Texture segmentation using Gabor modulation/demodulation*. Pattern Recognition Letters, vol. 6, no. 4, pp. 261-267, 1987.

[Cli90] P. Clifford. *Markov Random Fields in Statistics.* in Disorder in Physical Systems: A Volume in *Honour of John M. Hammersley*, eds. G. Grimmett and D. Welsh, Oxford University Press, pp.19–32, UK, 1990.

[Coh03] M. F. Cohen, J. Shade, S. Hiller and O. Deussen. *Wang tiles for image and texture generation*. ACM Transactions on Graphics 22, 3, pp.287–294, 2003.

[Col80] D. Coleman, P. Holland, N. Kaden, V. Klema and S. C. Peters. *A system of subroutines for iteratively reweighted least squares computations*. ACM Trans. Math. Software, 6, 3, pp.327–336, 1980.

[Con80] R. W. Conners and C. Harlow. *A Theoretical Comparison of Texture Algorithms*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2(3), pp. 204-222, 1980.

[Coo05] R. L. Cook and T. DeRose. *Wavelet noise*. In *SIGGRAPH*, pp. 803– 811, New York, NY, USA, 2005.

[Cro83] G. Cross and A. Jain. *Markov Random Field Texture Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5, pp. 25-39, 1983.

[Cut02] B. Cutler, J. Dorsey, L. McMillan, M. Muller and R. Jagnow. *A procedural approach to authoring solid models. ACM Trans. Graph. 21*, 3, pp. 302– 311, 2002.

[DaC01] J. -P. Da Costa. *Analyse statistique de textures directionnelles. Application a la caracterisation de materiaux composites*. Thèse N°2463, Univ. Bordeaux 1, 2001.

[DaC04] J. -P. Da Costa, C. Germain, P. Baylou and M. Cataldi. *An image analysis approach for the structural characterization of pyrocarbons.* Proceedings of Composite Testing, University Press, Bristol, UK, 2004.

[DaC10] J. -P. Da Costa and C. Germain. *Synthesis of solid textures based on a 2D example: application to the synthesis of 3D carbon structures observed by transmission electronic microscopy*. Proc. of SPIE, vol. 7538, Image Processing: Machine Vision Applications III, pp.10, 2010.

[Dav97] N. Davidson, A. Clarke and G. Archetypal. *Large-area, high-resolution image analysis of composite materials*. Journal of Microscopy 185, 233-242, 1997.

[Deb97] J. S. DeBonet. *Multiresolution sampling procedure for analysis and synthesis of texture images*. In *SIGGRAPH:* Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., pp. 361– 368, 1997.

[Des93] J. -F. Despres. *Les interphases de carbone pyrolytique dans les composites carbone/carbure de silicium*, thèse n°180, Université de Pau et de Pays de l'Adour, 9 septembre 1993.

[Dis97] J. Dischler and D. Ghazanfarpour. *A procedural description of geometric textures by spectral and spatial analysis of profiles*. Computer Graphics Forum, 16:129– 139, 1997.

[Dis98] J. Dischler, D. Ghazanfarpour and R. Freydier. *Anisotropic solid texture synthesis using orthogonal 2D views*. Computer Graphics Forum 17, 3 (Proc. Eurographics), pp. 87–96, 1998.

[Dis99] J. Dischler and D. Ghazanfarpour. *Interactive Image - Based Modeling of Macrostructured Textures*. In Computers & Graphics 19**,** pp. 66–74, 1999.

[Dis01] J. Dischler and D. Ghazanfarpour. *A survey of 3D texturing*. In Computers & Graphics, Vol. 25, issue 1, pp.135-151, 2001.

[DiZ86] S. Di Zenzo. *A note on the gradient of a multi-image*, Computer Vision, Graphics, and Image Processing, 33(1), 116, 1986.

[Dro03] I. Drori, D. Cohen-Or and H. Yeshurun. *Fragment-based image completion*. In *ACM Transactions on Graphics, SIGGRAPH*, volume 22, i.3, pp. 303-312, 2003.

[Dub11] S. Dubois, R. Peteri and M. Menard. *Textures dynamiques: etat de l'art, modelisation, applications*. 23ème colloque GRETSI, Bordeaux, France, 2011.

[Dun00] J. S. Duncan and N. Ayache. *Medical image analysis: Progress over two decades and the challenges ahead*. IEEE Trans on PAMI. Vol 22, pp.85–106, 2000.

[Ebe01] C. Eberhardt and A. Clarke. *Fiber-orientation measurements in short-glass-fibre composites. part 1. automated, high-angular-resolution measurement by confocal microscopy*. Composites Science and Technology 61, 1389-1400, 2001.

[Ebe02] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin and S. Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[Efr99] A. Efros and T. Leung. *Texture synthesis by non-parametric sampling*. In the International Conference on Computer Vision, volume 2, pp. 1033–1038, Sep 1999.

[Efr01] A. Efros and W. T. Freeman. *Image quilting for texture synthesis and transfer*. In SIGGRAPH: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 341-346, New York, NY, USA, ACM Press, 2001.

[Far12] B. Farbos, J. -M. Leyssale, J. -P. Da Costa, G. Vignoles. *Atomistic models of pyrolytic carbons obtained with the IGAR method*. *Carbon 2012 Conference*, Cracow, Poland, S. Blazewicz & E. Frackowiak eds., ref. 382, 2012.

[Fra93] J. M. Francos, A. Z. Meiri and B. Porat. *A Unified Texture Model Based on a 2-D Wold-Like Decomposition*. IEEE Trans. Signal Processing, vol. 41, pp. 2665-2678, 1993.

[Gag83] A. Gagalowicz. *Vers un modele de textures.* These, Universite Paris 6, 1983.

[Gal11] B. Galerne, Y. Gousseau and J.-M. Morel. *Random Phase Textures: Theory and Synthesis.* IEEE Trans. on Image Processing, vol. 20, pp: 257 - 267, 2011.

[Gar84] G. Gardner. *Simulation of Natural Scene Using Textured Quadric Surfaces*. In: SIGGRAPH, 11–20, 1984.

[Gei07] A. K. Geim and K.S. Novoselov. *The rise of graphene.* In Nature materials, Nature publishing group, vol 6, march 2007.

[Gem84] S. Geman and D. Geman. *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, no. 6, pp. 721-741, 1984.

[Gem91] D. Geman. *Random fields and inverse problems in imaging*. In Lecture Notes in Mathematics, vol. 1427, pp. 113-193. Springer-Verlag, 1991.

[Ger92] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[Ger97] C. Germain. *Contribution à la Caractérisation Multi-échelle de l'Anisotropie des Images Texturées*. Thèse N°1808, Université Bordeaux I, 1997.

[Ger03] C. Germain, J. -P. Da Costa, O. Lavialle and P. Baylou. *Multiscale estimation of vector field anisotropy. Application to texture characterization.* In Signal Processing, vol. 83, pp. 1487-1503, 2003.

[Gha95] D. Ghazanfarpour and J.-M. Dischler. *Spectral analysis for automatic 3-D texture generation*. Computers and Graphics 19, 3, pp. 413–422, 1995.

[Gha96] D. Ghazanfarpour and J.-M. Dischler. *Generation of 3D texture using multiple 2D models analysis*. Computer Graphics Forum 15, 3 (Proc. Eurographics), pp. 311–323, 1996.

[Gol08] A. Goldberg, M. Zwicker and F. Durand. *Anisotropic noise*. *SIGGRAPH*, pp. 1–8, New York, NY, USA, 2008.

[Gor03] G. Gorla, V. Interrante, and G. Sapiro. *Texture Synthesis for 3D Shape Representation.* IEEE Transactions on Visualization and Computer Graphics Volume 9 Issue 4, pp.512-524, October 2003.

[Hai91] M. Haindl. *Texture synthesis.* CWI Quarterly, vol.4, pp. 305-331, 1991.

[Han06] J. Han, K. Zhou, L. -Y. Wei, M. Gong, H. Bao, X. Zhang and B. Guo. *Fast example - based surface texture synthesis via discrete optimization*. Visual Computer 22(9), pp.918–925, 2006.

[Har73] R. M. Haralick, K. Shanmugam, and I. H. Dinstein. *Textural Features for Image Classification*. IEEE Transactions on Systems, Man and Cybernetics, vol. 3, pp. 610-621, 1973.

[Har92] R. M. Haralick and L.G. Shapiro. *Computer and Robot Vision.* Vol I, Addison Wesley, 1992.

[Hec86] P. Heckbert, *Survey of Texture Mapping*, Pixar, IEEE Computer Graphics and Applications, Nov. 1986.

[Hee95] D. J. Heeger and J. R. Bergen. *Pyramid-based texture analysis/synthesis*. In SIGGRAPH , pp. 229–238, New York, NY, USA, 1995.

[Her01] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless and D.H. Salesin. *Image analogies*. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 327-340, New York, NY, USA, ACM Press, 2001.

[Hil91] D. Hilbert. *Über die stetige abbildung einer linie auf Flächenstück*. In *Mathematische Annalen*, 38:459–460, 1891.

[Ige97] H. Igehy and L. Pereira. *Image replacement through texture synthesis*. In International Conference on Image Processing, vol 3, pp. 186–189, Oct 1997.

[Jag04] R. Jagnow, J. Dorsey and H. Rushmeier. *Stereological Techniques for Solid Textures*. In: SIGGRAPH '2004, pp.329–335, 2004.

[Jol86] I. T. Jollife. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[Kho87] A. Khotanzad and R. Kashyap. *Feature Selection for Texture Recognition Based on Image Synthesis*. IEEE Transactions on Systems, Man, and Cybernetics, 17, pp. 1087-1095, 1987.

[Kop07] J. Kopf, C.W. Fu, D. Cohen-Or, O. Deussenn, D. Lischinski and T.T. Wong. *Solid Texture Synthesis from 2D Exemplars*. ACM SIGGRAPH, TOG, vol. 26, issue 3, 2007.

[Kwa03] V. Kwatra, A. Schödl, I. A. Essa, G. Turk and A. F. Bobick. *Graphcut textures: Image and video synthesis using graph cuts*. In *ACM Transactions on Graphics, SIGGRAPH*, volume 22(3), pp. 277-286, July 2003.

[Kwa05] V. Kwatra, I. Essa, A. Bobick and N. Kwatra. *Texture optimization for example-based synthesis*. ACM Transactions on Graphics 24, 3 (Proc. SIGGRAPH), pp.795–802, 2005.

[Kwa07] V. Kwatra, S. Lefebvre, G. Turk and L.-Y.Wei. *Example-Based Texture Synthesis*. In Course Notes for SIGGRAPH 2007.

[Lag09] A. Lagae, S. Lefebvre, G. Drettakis and P. Dutre. *Procedural noise using sparse Gabor convolution*. In *SIGGRAPH*, 28(3), August 2009.

[Lee01] K. S. Lee, S. W. Lee, J. Youn, T. Kang and K. Chung. *Confocal microscopy measurement of the fiber orientation in short fiber reinforced plastics*. Fibers and Polymers 2, 41-50, 2001.

[Lef06] S. Lefebvre and H. Hoppe. *Appearance - space texture synthesis*. In ACM Transactions on Graphics, 25(3), pp.541-548, 2006.

[Lew89] J. -P. Lewis. *Algorithms for Solid Noise Synthesis*. In Computer Graphics (SIGGRAPH Proceedings), volume 23, pp. 263–270, July 1989.

[Ley09] J. -M. Leyssale, J. -P. Da Costa, C. Germain, P. Weisbecker and G. Vignoles. *An image-guided atomistic reconstruction of pyrolytic carbons*. Applied Physics Letters, vol. 95, 2009.

[Lia01] L. Liang, C. Liu, Y. Q. Xu, B. Guo and H. Shum. *Real-time texture synthesis by patch-based sampling*. ACM Transactions on Graphics 20, 3, pp.127– 150, 2001.

[Lin04] W. -C. Lin, J. H. Hays, C. Wu, V. Kwatra and Y. Liu. *A comparison study of four texture synthesis algorithms on regular and near-regular textures*. Technical report, Carnegie Mellon University, 2004.

[LiS01] S. Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer Verlag, Tokyo, second edition, 2001.

[Mal89] S. Mallat. *Multifrequency Channel Decomposition of Images and Wavelet Models.* IEEE Trans. Acoustic, Speech and Signal Processing, 37,12, pp. 2091-2110, 1989.

[Mar89] H. Marsh, *Introduction to carbon science.* Butterworths, pp 321, 1989.

[Merr] Merriam-Webster's Collegiate Online Dictionary - *www.merriam-webster.com*

[Mess] Messier – Buggati – Dowty breaking system from *http://materiaux-energetiques. com/activites/wheels-and-brakes/carbon-brake/*

[Met53] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller. *Equations of state calculations by fast computing machines*. Journal of Chemical Physics, vol. 21, pp. 1087-1091, 1953.

[Ncem] The National Center for Electron Microscopy, NCEM HRTEM Image Simulation Software, *http://ncem.lbl.gov/frames/source/software.htm*

[Nea03] A. Nealen and M. Alexa. *Hybrid texture synthesis*. In EGRW: Proceedings of the 14th Eurographics workshop on Rendering, pp. 97-105, Airela-Ville, Switzerland, Eurographics Association, 2003.

[Nen97] S. Nene and S. Nayar. *A simple algorithm for nearest neighbor search in high dimensions*.IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:989–1003, 1997.

[Obe89] A. Oberlin. *High-resolution TEM studies of carbonization and graphitization*, Chemistry and Physics of Carbon, Ed. Peter A. Thrower. Thrower & M. Dekker, New York, 22, pp. 1-143, 1989.

[Obe02] A. Oberlin. *Pyrocarbons*. Carbon, 40, pp:7-24, 2002.

[Owa04] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi. *Volumetric illustration : designing 3d models with internal textures*. ACM Trans. Graph. 23, 3, pp.322–328, 2004.

[Pag98] R. Paget and I. Longstaff. *Texture synthesis via a noncausal nonparametric multiscale Markov random field*. In IEEE Transactions on Image Processing 7, 6, pp. 925–931, 1998.

[Pea90] G. Peano. *Sur une courbe qui remplit toute une aire plane*. In Mathematische Annalen, 36:157-160, 1890.

[Pea85] D. R. Peachey. *Solid Texturing of Complex Surfaces*. Computer Graphics (SIGGRAPH Proceedings), 19(3), July 1985.

[Pei88] H. Peitgen and D. Saupe. *The Science of Fractal Images.* Springer-Verlag, New York, 1988.

[Per85] K. Perlin. *An Image Synthesizer*. Computer Graphics (SIGGRAPH Proceedings), volume 19, pp. 287–296, July 1985.

[Pey10] G. Peyré. *Texture synthesis with grouplets*. Transactions on Pattern Analysis and Machine Intelligence, 4(32), pp.733–746, 2010.

[Pie93] H. Pierson. *Handbook of carbon, graphite, diamond and fullerenes,* p.48, Noyes Publications, 1993.

[Pie07] N. Pietroni, M. Otaduy, B. Bickel, F. Ganovelli and M. Gross. *Texturing internal surfaces from a few cross sections*. Computer Graphics Forum 26, 3, pp.637–644, 2007.

[Pop93] K. Popat and R. W. Picard. *Novel cluster-based probability model for texture synthesis, classification, and compression*. In Proceedings SPIE visual Communications and Image Processing, Boston, 1993.

[Por96] J. Portilla, R. Navarro and O. Nestares. Texture synthesis - by - analysis method based on a multiscale early-vision model. Opt. Eng. 35(8) pp. 2403–2417, August 1996.

[Por00] J. Portilla and P. Simoncelli. *A parametric texture model based on joint statistics of complex wavelet coefficients*. International Journal of Computer Vision *40*, 1, pp. 49–70, 2000.

[Pra00] E. Praun, A. Finkelstein and H. Hoppe. *Lapped textures*. In SIGGRAPH : Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., pp. 465–470, 2000.

[Pri12] J. C. Prieto, C. Revol-Muller, F. Peyrin, P. Camelitti and C. Odet, *3D Texture synthesis for modeling realistic organic tissues*, VISAPP 2012. International Conference on Computer Vision Theory and Applications, 24-26, pp.60-65, Rome, Italy, February 2012.

[Qin07] X. Qin and Y. -H. Yang. *Aura 3D textures*. IEEE Transactions on Visualization and Computer Graphics, Volume 13 Issue 2, pp. 379-389, 2007.

[Rab10] J. Rabin, G. Peyre, J. Delon and M. Bernot. *Wasserstein barycenter and its application to texture mixing*. Technical report, 2010.

[Ram00] C. Ramananjarasoa, O. Alata and M. Najim. *2-D Wold Decomposition: New Parameter Estimation Approach to Evanescent Field Spectral Supports*. Signal Processing X EUSIPCO, Vol. 2, pp.913–916, Tampere, Finlande, 2000.

[Rao90] A. R. Rao. *A taxonomy for texture description and identification*. Springer-Verlag, NY, USA, 1990, ISBN: 0-387-97302-8.

[Ray10] P. I. Raynal, M. Monthioux, J. -P. Da Costa and O. Dugne. *Multi-scale quantitative analysis of carbon structure and texture: III. Lattice fringe imaging analysis*. *Carbon 2010 Conference*, Clemson, SC, M. Thies ed. ref. 627, 11-16 July 2010.

[Rou02] J. -N. Rouzaud and C. Clinard. *Quantitative high resolution transmission electron microscopy: a promising tool for carbon materials characterization*. Fuel Processing Technology, 77-78, pp. 229-235, 2002.

[Sag94] H. Sagan. *Space Filling Curves*, Springer-Verlag, New York ,1994.

[Sav93] G. Savage. *Carbon-carbon composites*. Eds. Chapman & Hall, London, 1993.

[Sch00] A. Schodl, R. Szeliski, D. H. Salesin and I. Essa. *Video textures*. In *Computer Graphics*, ACM SIGGRAPH Proceedings, pp. 489-498, New Orleans, July 2000.

[Shi00] H. Shim, R. Hurt and N. Yang. *A methodology for analysis of 002 lf fringe images and its application to combustion-derived carbons*. Carbon 38, pp29-45, 2000.

[Sil86] B.W. Silverman. *Density estimation for statistics and data analysis.* Chapman and Hall, London, 1986.

[Sim92] P. Simoncelli, W. T. Freeman, E. Adelson and D. Heeger. *Shiftable Multi-Scale Transforms*. IEEE Transactions on Information Theory, Special Issue on Wavelets 38, pp.587–607, 1992.

[Siv99] K. Sivakumar. *Morphologically constrained GRFs:application to texture synthesis and analysis*. IEEE Trans. PAMI 21-2, pp. 148–153, 1999.

[Skl78] J. Sklansky. *Image Segmentation and Feature Extraction*, IEEE Transactions on Systems, Man, and Cybernetics, SMC-8, pp. 237-247, 1978.

[Snec] Snecma Vinci motor from *http://www.safran-group.com/site-safran/aerospatial/propulsion-aeronautique-et/moteurs-spatiaux/vinci-r-231/*

[Ste84] D. Sterio. *The unbiased estimation of number and sizes of arbitrary particles using the dissector*. Journal of Microscopy 134, 127-136, 1984.

[Sri08] G. N. Srinivasan and G. Shobha. *Statistical Texture Analysis*. Proceedings of World Academy of Science, Engineering and Technology, vol. 36, 2008.

[Sti07] Y. Stitou, F. Turcu, Y. Berthoumieu and M. Najim. *Three-Dimensional Textured Image Blocks Model Based on Wold Decomposition*. IEEE Transactions on Signal Processing, vol. 55, issue 7, pp. 3247-3261, 2007.

[Sto98] R. Stoica, J. Zerubia and J. Francos. *The Two-Dimensional Wold- Decomposition for Segmentation and Indexing in Image Libraries*. Proceedings of ICASSP, Seattle, 1998.

[Sun10] W. Sun, Y. Lu, F. Wu, S. Li and J. Tardif. *High-Dynamic-Range Texture Compression for Rendering Systems of Different Capacities*. IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 1, pp. 57-69, Jan.-Feb. 2010.

[Szu96] M. Szummer and R.W. Picard. *Temporal texture modeling*. In International Conference on Image Processing, volume 3, pp. 823–826, Sep 1996.

[Tak08] K. Takayama, M. Okabe, T. Ijiri and T. Igarashi. *Lapped solid textures: filling a model with anisotropic textures*. Proceedings of ACM SIGGRAPH, Volume 27, Issue 3, 2008.

[Tam78] H. Tamural, S Mori and T Yamawaki. *Textural features corresponding to visual perception*. IEEE Transations on Systems, Man and Cybernetics, Vol. 8, No. 6, pp. 460-473, 1978.

[Ton02] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo and H. Shum. *Synthesis of bi-directional texture functions on arbitrary surfaces*. In SIGGRAPH: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. ACM Press, pp.665–672, 2002.

[Tur86] M. R. Turner. *Texture discrimination by Gabor functions*. Biological Cybernetics, vol. 55, no. 2-3, pp. 71-82, 1986.

[Tur91] G. Turk. *Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion*. In Computer Graphics (SIGGRAPH Proceedings), vol. 25, pp. 289–298, July 1991.

[Tur01] G. Turk, *Texture Synthesis on Surfaces*, Proc. ACM SIGGRAPH 2001, pp. 347-354, 2001.

[Ups89] S. Upstill. *The RenderMan Companion*. Addison Westley, 1989.

[Urs10] R. Urs, J. -P. Da Costa, J. -M. Leyssale, G. Vignoles and C. Germain. *A non-parametric approach for 3D texture synthesis.* 3D-IMS 2010 – 2nd Conference on 3D-Imaging of Materials and Systems 2010, Carcans-Maubuisson, France, D. Bernard ed., Sept 2010.

[Urs11] R. Urs, J. -P. Da Costa, J. -M. Leyssale, G. Vignoles et C. Germain. *Algorithmes non paramétriques pour la synthèse de textures volumiques à partir d'un exemple 2D.* XXIIIe Colloque GRETSI – Traitement du Signal et des Images, Bordeaux, Septembre 2011.

[Urs12] R. Urs, J. -P. Da Costa, J. -M. Leyssale, G. Vignoles and C. Germain. *Non-parametric synthesis of laminar volumetric textures from a 2D sample.* In Proceedings of British Machine Vision Conference, pp.54.1-54.11, 2012.

[Visi] Vision Texture online database *http://vismod.media.mit.edu/vismod/imagery/VisionTexture*

[Wei00] L. -Y. Wei and M. Levoy. *Fast texture synthesis using tree-structured vector quantization*. In SIGGRAPH, 27th International Conference on Computer Graphics and Interactive Techniques, pp. 479-488, 2000.

[Wei01] L. -Y. Wei and M. Levoy. *Texture Synthesis over Arbitrary Manifold Surfaces*, Proceedings of ACM SIGGRAPH, pp. 355-360, 2001.

[Wei03] L. -Y. Wei and M. Levoy. *Texture synthesis from multiple sources*. Proceedings of ACM SIGGRAPH Sketches & Applications, New York, 2003.

[Wei09] L. -Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. *State of the art in example based texture synthesis*. In Eurographics, EGSTAR. Eurographics Association, 2009.

[Wor96] S. Worley: *A cellular texture basis function*. In SIGGRAPH : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM Press, pp. 291–294, 1996.

[Wik01] K. Wikantika, A. Harto and R. Tateishi. *The use of spectral and textural features from Landsat TM image for land cover classification in mountainous area.* Proceedings of the IECL Japan workshop, Tokyo, 2001.

[XuG00] Y. Q. Xu, B. Guo and H. Shum. *Chaos mosaic: Fast and memory efficient texture synthesis*. In Tech. Rep. MSRTR-2000-32, Microsoft Research, 2000.

[Yin01] L. Ying, A. Hertzmann, H. Biermann and D. Zorin. *Texture and Shape Synthesis on surfaces*, Proc. 12th Eurographics Workshop Rendering, June 2001.

[Zha08] Y. Zhang, Z. Sun and W. Li. *Texture synthesis based on Directional Empirical Mode Decomposition*. Computers & Graphics, Vol. 32 – 2, pp. 175–186, 2008.

[Zho06a] D. Zhou. *Texture Analysis and Synthesis Using a Generic Markov-Gibbs Image Model*. Computer Science - University of Auckland, Master's thesis, 2006.

[Zho06b] K**.** Zhou, X. Huang, X. Wang, Y. Tong, M. Desbrun, B. Guo and H.-Y. Shum. *Mesh quilting for geometric texture synthesis*. Proceedings of ACM Transactions on Graphics SIGGRAPH, Volume 25 Issue 3, pp.690 – 697, July 2006.

[Zel02] S. Zelinka and M. Garland. *Towards real - time texture synthesis with the jump map*. In EGRW: Proceedings of the 13th Eurographics workshop on Rendering, Eurographics Association, pp.99–104, 2002.

List of contributed work

List of publications

[Urs10] **R. Urs**, J.-P. Da Costa, J. -M. Leyssale, G. Vignoles and C. Germain. *A non-parametric approach for 3D texture synthesis.* 3D-IMS 2010 – 2nd Conference on 3D-Imaging of Materials and Systems 2010, Carcans-Maubuisson, France, D. Bernard ed., Sept 2010.

[Urs11] **R. Urs**, J. -P. Da Costa, J. -M. Leyssale, G. Vignoles et C. Germain. *Algorithmes non paramétriques pour la synthèse de textures volumiques à partir d'un exemple 2D.* XXIIIe Colloque GRETSI – Traitement du Signal et des Images, Bordeaux, Septembre 2011.

[Urs12] **R. Urs**, J. -P. Da Costa, J. -M. Leyssale, G. Vignoles and C. Germain. *Non-parametric synthesis of laminar volumetric textures from a 2D sample*. In Proceedings of British Machine Vision Conference, pp.54.1-54.11, 2012, *http://dx.doi.org/10.5244/C.26.54*.

List of contributed talks

R. Urs. *Synthèse de textures volumiques par inférence 2D/3D*. Contributed talk for *Journée sur la texture 3D* at *Maison des Sciences de l'Ingénieur (MSI)*, Rochelle University, July 2010.

R. Urs. *2D/3D synthesis of HRTEM-like snapshots of dense carbons*. Contributed talk on Imaging & Image Analysis at Carbon 2012 satellite workshop "Order/Disorder in Bulk Carbon Materials: structure/texture, quantification, imaging, modelling, structure-property relationships" in June 15-16, 2012, at Cracow AGH (Poland), under the PyroMaN project - *http://www.pyroman.cnrs.fr/pyroman/fr/*.