



Pós-Graduação em Ciência da Computação

"BIDIMENSIONAL AND TRIDIMENSIONAL
SAMPLE BASED SYNTHESIS OF VECTORIAL
ELEMENTS DISTRIBUTION PATTERNS"

Por

VLADIMIR ALVES DOS PASSOS

Dissertação de Mestrado



Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, 07/2010

Universidade Federal de Pernambuco
Centro de Informática

Vladimir Alves dos Passos

Bidimensional and Tridimensional Sample Based Synthesis of Vectorial Elements Distribution Patterns

*Trabalho apresentado ao Curso de Pós-Graduação em
Ciência da Computação do Centro de Informática da Uni-
versidade Federal de Pernambuco como requisito parcial
para obtenção do grau de Mestre em Ciência da Com-
putação.*

Orientador: *Marcelo Walter*

Recife
16 de julho de 2010

Passos, Vladimir Alves dos

**Bidimensional and tridimensional sample based synthesis
of vectorial elements distribution patterns / Vladimir Alves dos
Passos. - Recife: O Autor, 2010.**

ix, 53 folhas : il., fig.

**Dissertação (mestrado) Universidade Federal de
Pernambuco. Cln. Ciência da Computação, 2010.**

Inclui bibliografia.

1. Computação gráfica. 2. Síntese de texturas. I. Título.

006.6

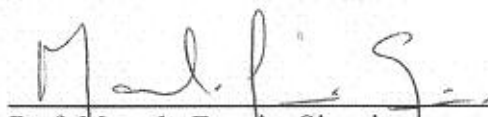
CDD (22. ed.)

MEI2010 – 050

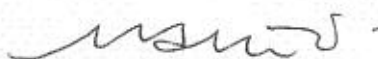
Dissertação de Mestrado apresentada por **Vladimir Alves dos Passos** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Bidimensional and Tridimensional Sample Based Synthesis of Vectorial Elements Distribution Patterns**", orientada pelo **Prof. Marcelo Walter** e aprovada pela Banca Examinadora formada pelos professores:



Prof. Tsang Ing Ren
Centro de Informática / UFPE

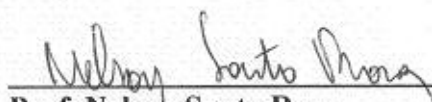


Prof. Marcelo Ferreira Siqueira
Departamento de Informática e Matemática Aplicada / UFRN



Prof. Marcelo Walter
Instituto de Informática / UFRGS

Visto e permitida a impressão.
Recife, 15 de julho de 2010.



Prof. Nelson Souto Rosa
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Resumo

Nós apresentamos um método eficiente para síntese de padrões definidos como uma coleção de elementos vetoriais bidimensionais, a partir de uma amostra do padrão. Soluções recentes para este problema fazem uso de triangulação da entrada ou de medidas estatísticas da amostra para controlar o estágio de síntese. Nós propomos um método aplicável a texturas coloridas, desde regular até estocásticas, e que provê controle local sobre a densidade dos elementos. A amostra é segmentada em grupos de elementos similares e definimos uma nova métrica, que não ignora elementos isolados, para cálculo de distância entre vizinhanças de elementos, para comparar vizinhanças diferentes e incompletas. O laço principal de síntese consiste em um crescimento procedural, onde sementes são substituídas por referências a elementos da amostra, gerando novas sementes até que o espaço de síntese seja preenchido. Os resultados mostram a mesma qualidade visual de trabalhos anteriores, e resolvem padrões não abordados em trabalhos anteriores. Nós também mostramos que este método pode ser estendido para sintetizar padrões vetoriais sobre malhas poligonais.

Palavras-chave: Síntese de texturas; Padrões de Distribuição; Texturas Vetoriais.

Abstract

We present an efficient method for the synthesis of patterns defined as 2D collections of vector elements. Current solutions to this problem rely on the triangulation of the input space or on statistical measures of the sample to drive the synthesis step. We propose a method applicable to colored textures from regular to stochastic, which provides control over the local density of elements. Also, our results show the same visual quality as previous works. The sample is segmented into groups of similar elements and we use a novel local neighborhood distance metric to compare distinct and incomplete neighborhoods. This metric does not ignore existing unpaired elements. The main synthesis loop consists of a procedural growth, where seeds are replaced by a reference to an element from the sample, generating new seeds until the target space is filled. We also show that this method can be adapted to synthesize textures over surfaces.

Keywords: Texture Synthesis; Distribution Patterns; Vectorial Textures.

Contents

1	Introduction	1
1.1	Problem Characterization	1
1.2	Contributions	3
1.3	Organization	3
2	Related Work	4
2.1	Fundamentals	4
2.1.1	Textures	4
2.1.2	Sample Based Texture Synthesis	5
2.2	Methods for Raster Texture Synthesis	5
2.2.1	Pixel Based Synthesis	6
2.2.2	Patch Based Synthesis	8
2.3	Texture Synthesis on Surfaces	9
2.3.1	Scale and Orientation	9
2.3.2	Dense Point Distribution	10
2.3.3	Synthesis Over Texture Atlas	11
2.3.4	Triangle Patches	12
2.4	Vectorial Pattern Distribution Synthesis	13
2.4.1	Stroke Synthesis	14
2.4.2	Stroke Pattern Analysis and Synthesis	15
2.4.3	An Example-based Procedural System for Element Arrangement	18
2.5	Appearance-guided Synthesis of Element Arrangements by Example	19
2.6	Discussion	22
3	Proposed Method	23
3.1	Overview	23
3.2	Input classification	24
3.2.1	Characteristics of Elements	24
3.2.2	Principal Component Analysis	25
3.2.3	Classification Description	26
3.2.4	Spatial features of the pattern	27
3.3	Synthesis	27
3.3.1	Covering The Target Space	28
3.3.2	Seeding and Growth	29
3.3.3	Distance Metric For Neighborhoods	30

3.3.4	Choosing The Best Matching	32
3.3.5	Patches	33
3.4	Tridimensional Synthesis	33
3.4.1	Input Information and Parameters	34
3.4.2	Covering The Target Space	34
3.4.3	Obtaining The Neighborhood	35
3.4.4	Generating New Seeds	36
3.4.5	Rendering	36
4	Experiments and Results	37
4.1	2D Synthesis	37
4.2	3D Synthesis	38
5	Conclusions	48
5.1	Limitations	48
5.2	Future Work	49

List of Figures

1.1	Left: Raster Texture; Right: Pattern Distribution texture; Below: The basic element for each texture.	2
1.2	Samples and results of the method for the four different types of texture.	2
1.3	Results of tridimensional synthesis and respective input sample in the up left corner.	3
2.1	Spectrum of textures used in synthesis, according to the degree of randomness. Image adapted from [LHW ⁺ 04]	4
2.2	MRF experiment on a general image (a) and on a texture (b). On the general image, when looking at two different locations (a1,a2), under a limited visibility range, each location is clearly distinct. However, in the image, the two locations (b1,b2) are similar. Also, the central pixel from the windows in (b) is determined only by its visible neighbors. Those characteristics are the stationarity and locality properties, respectively.	6
2.3	Scan-line ordering synthesis [Wei07]. Figure (a) is the input sample and the output starts as noise. Only the last few rows and columns are used, as shown in (b). The unused noisy pixels are presented as black. Neighborhoods crossing the output boundaries are handled toroidally. From (b) to (e), different pixels being synthesized are shown.	7
2.4	Synthesis results with window size of (a) 1x1, (b) 5x5, (c) 7x7, (d) 9x9 and (e) 30x30. As the size increases, the result quality tends to increase too, as well the computational cost. [Wei07]	7
2.5	Comparison of pixel and patch-based algorithms. The gray region indicates synthesized portion.[Wei07]	8
2.6	Methods for handling adjacent patches during synthesis. (a) two patches shown in different colors. (b) the overlapped region is simply blended from the two patches. (c) an optimal path is computed from the overlapped region.	8
2.7	User constraints for a vector field (left) and the resulting vector field (right) (from [Tur01]).	10
2.8	Various orientation fields are used to create textures on a sphere from [WL01].	10
2.9	Results from the texture synthesis approach of Wei and Levoy [WL01].	11
2.10	Synthesis results from the unfolding method of Ying et al. [YHBZ01].	12
2.11	Synthesis results from the method of Lefebvre and Hoppe [LH06]. The upper row shows the texture atlases for the models on the bottom.	12

2.12	Texture synthesis results using the Jump Maps method of Zelinka and Garland [ZG03].	13
2.13	Synthesis results using the face clustering method of Soler et al. [SCA02].	14
2.14	Results from (a) [DHvOS00], (b) [SABS94], (c) [Ost99], (d) [Her98], (e) [KMM ⁺ 02], (f) [JEGPO02].	15
2.15	Grouping and triangulation of the analysis step of [BBT ⁺ 06]	16
2.16	Synthesis by Barla [BBT ⁺ 06], without displacements (a) and with displacements (b).	18
2.17	Synthesized arrangements and input sample patterns (from [IMIM08]).	20
2.18	Synthesis results attesting the variety of distributions that the model of [HLT ⁺ 09] can handle.	21
3.1	Features evaluated for an illustrative example. Here, the elements were grouped into two distinct groups, curve and circle. In this case, three features are evaluated. The smaller distance between distinct circles (A), distinct curves (B), and circles and curves (C).	23
3.2	Illustration of the local texture growth. Each seed is represented by a number, and the highlighted seed is the first on the list.	24
3.3	Example of the characteristics evaluated for one element and used to classify it, before normalization.	25
3.4	The classification pipeline. Figure (1) is the initial sample; Figure (2) shows the result of histogram segmentation. The upper histogram is the classification by area. The green column, with 9 elements, becomes a group itself and the red column, with 61 elements, is further refined. The white empty columns are ignored. The lower histogram shows the refinement, where each of the remaining characteristics are reduced to 1 dimension by PCA. Figure (3) is the resulting classified sample in this case 6 groups were computed	26
3.5	Illustrative example of feature evaluation. In this illustration, there is a texture with 6 elements, so the minimal distances $d(x, A)$ for each element (surrounding images) need to be evaluated, and also the minimal distances between groups $d(B, A)$ (central image). B and A correspond to circle and curve groups. All of the minimal distances are represented by black line segments. In image I, it is shown the minimal distances found for the curve element (1). The upper line represents the minimal distance between (1) and any other curve. The lower line represents the minimal distance between (1) and any circle. Similarly, there are the features for each curve (Figures II and VI) and for each circle (Figures III, IV, V). On the central image, the long vertical line represents the minimal distance between circles, the short vertical line is the minimal distance between curves, and the remaining line is the minimal distance between curves and circles.	28

3.6	Seeding process. On the left, the current arrangement being synthesized. (A) marks the position of the seed extracted from the head of the list. The best matching element is searched throughout the sample texture. In this case, assume the seed reerences the group with curves. The best matching element was found to be the highlighted curve. The neighboring elements (I) are extracted and four new seeds are generated from them (II). This set is pasted over the seed (A), and on the right arrangement, only seed 1 will be accepted.	30
3.7	Varying density along the horizontal axis. Sample and result.	31
3.8	Distance metric for neighborhoods, shown in an artificial example. We have the comparison between two pairs of neighborhoods, AB and CD. Neighborhoods A and C are exactly the same, while B and D differ only on the position of one curve. Neighborhoods are illustrated together in the same frame, with the smallest distances evaluated for each element (before normalization). Only the displacement of one of the elements, bringing it closer to another element from the other neighborhood, makes both neighborhoods more similar, and this effect is easily detected by the final value of the metric. A lower value denotes a more similar pair of neighborhoods.	32
3.9	Illustration of the influence of the modified dissimilarity factor. Figure (a) is the sample. In figure (b) the original factor is used. At the upper right part of the figure, the repetition of the big circles is perceived. The left portion also was not correctly synthesized. On (c) there is a new result, using the modified factor, providing a result closer to the sample.	33
3.10	Given a sample (a) and varying the patch size 0%, 50% and 100%, (b, c, d), respectively. Notice the same quality in the visual result.	34
3.11	Process to obtain the neighborhood of an element on the 3D space. In A) triangles from the mesh, with some elements over them. The circle represents the visibility sphere centered at the element whose neighborhood we are interested on. Elements 6 and 7 lies outside the sphere. On illustration B) it is shown the orthographic projection of elements 1 to 5 onto the plane of the triangle that contains the central element. The projection follows the normal direction N. The direction Y is obtained from the vector field defined for the surface, and the direction X is the cross product of Y and N. On illustration C), the final result after evaluating the 2D coordinates of each projected element.	35
3.12	Result with popup issues. When the scale of the element is small compared with the local curvature of the surface this problem is minimized, but at the borders this artifact can still be cleary seen.	36
4.1	Synthesized colored arrangements.	38
4.2	Texture synthesis results. From left to right: sample (smaller square), result by Hurtut [HLT ⁺ 09] and ours.	40
4.3	Texture synthesis results. From left to right: sample (smaller square), results by Hurtut [HLT ⁺ 09] and results of the proposed method.	41

4.4	Results from the analysis step. Figure (a) shows the result of the implementation of the method used on previous work. Figure (b) shows the results of the simplified method. In Figure (a) can be seen elements with highly distinct orientation classified on the same group (purple), and yet some small circles classified on the same group of the big elements (light blue, yellow). In Figure (b), such mistakes does not occur.	42
4.5	Texture synthesis results for illustrative geological patterns.	42
4.6	Comparison among the results from [HLT ⁺ 09] and the proposed method	43
4.7	Comparison among the results from [HLT ⁺ 09] and the proposed method	44
4.8	Comparison among the results from [HLT ⁺ 09] and the proposed method	45
4.9	Influency of parameters on the 3d synthesis. Figures (a), (b) and (c) show the variation of density, from high to low. Figures (d), (e) and (f) show the variation of path size from none, half and full visibility range, respectively. Figures (g), (h) and (i) show the variation of the scale, achieved by changing the ratio between the surface area and the sample area.	46
4.10	Tridimensional texture synthesis results using four different textures. On the left, the model is rendered with Gouraud shading model to illustrate the illumination over the triangles; on the right, no illumination model was applied. The concentration of elements by itself produces the shadowing effect.	47
5.1	Inter-relations between elements not well captured by the method for near-regular textures.	49

Introduction

In computer graphics and image processing, the main goal of texture synthesis is to generate images of arbitrary resolution that comprises some perceptive pattern. Such images are called textures. The complexity of textures varies widely and the suitability and applicability of a synthesis system can be measured by the range of textures types that the system can generate. For instance, there are regular textures such as a chessboard or pseudo regular textures, such as a compact distribution of cells with slightly varying shapes.

Texture synthesis has been explored for more than one decade, and the solutions for processing raster textures are well established in the community, not only for static textures, but also for video synthesis. Only recently there has been results for vectorial textures, which are essentially distribution patterns, and this topic is still an area of active research. Due to the nature of the vectorial data used in these works, the results are easily interpreted as non photorealistic. Synthesizing distribution patterns is not limited to the non photorealistic field, it depends entirely on the quality and complexity of the texture being processed. This work captures concepts of existing approaches for vectorial and raster texture synthesis, to present a new technique that generates results in 2D space, and show that the technique can be naturally amendable to support the synthesis of patterns in mesh surfaces without parametrization.

1.1 Problem Characterization

In sample based synthesis, the input texture is a small sample of the texture used to synthesize, that contains enough information to characterize this particular texture. In order to generate a new image of arbitrary resolution that resembles the original sample, the system must be able to perceive and correctly reproduce the pattern presented on it.

Traditionally, textures are treated as raster images, where each pixel is defined by its color. Vector textures, or pattern distributions, has a distribution of objects defined by geometric primitives with any number of desirable descriptors, such as color. This type of texture is explored in this work.

in the raster texture scenario, there is the process of determining the correct color of each pixel in the target texture, while in the distribution pattern scenario, which lacks the implicit grid, the synthesis process is adapted to generate a new distribution of elements that resembles the original distribution contained in the sample. Some degree of flexibility is desirable such that the final result does not become a simple tiling of the sample texture and small variations on the distribution can be obtained without losing the pattern characteristics initially presented in the sample.

The difference between raster textures and distribution patterns is illustrated in Figure 1.1, that shows a raster texture on the left and a pattern distribution as presented in [HLT⁺09] on the right. Below each illustration, there is the basic elements. The pattern distributions are conveniently described by the Scalable Vectorial Graphics (SVG) format [Fe01].

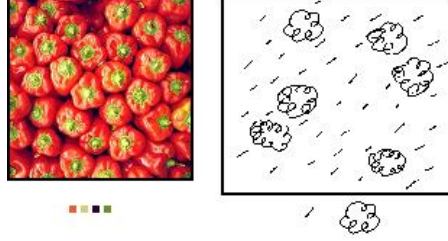


Figure 1.1 Left: Raster Texture; Right: Pattern Distribution texture; Below: The basic element for each texture.

Correctly reproducing the characteristics of the texture implies that the final image is free of vicinity errors. Those errors are qualitatively defined as misplaced elements which visually affect the overall result, making the observer able to identify patterns that are not presented on the input sample.

In order to qualitatively evaluate the robustness of the method, the procedure should be able to generate pattern distributions for all types of textures, such as regular, near-regular, irregular and stochastic. Figure 1.2 shows some results obtained by the proposed method. Similar to the raster texture synthesis, near-regular images are the most difficult to process. Although some good results can be obtained, it is not a problem entirely solved. Another desired feature is the possibility to adapt the system to different applications. A system that synthesizes the pattern over tridimensional surfaces and directly texturize tridimensional objects was implemented, using a similar approach as in 2D synthesis, but with few adaptations, to obtain results such as shown in the Figure 1.3.

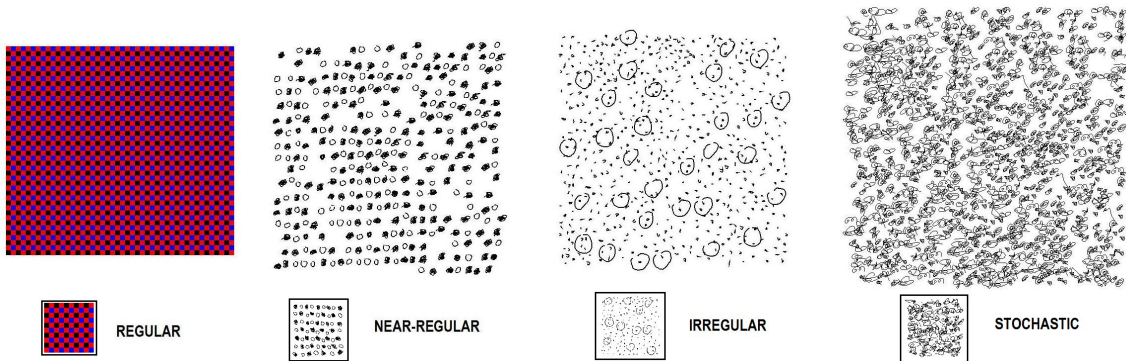


Figure 1.2 Samples and results of the method for the four different types of texture.

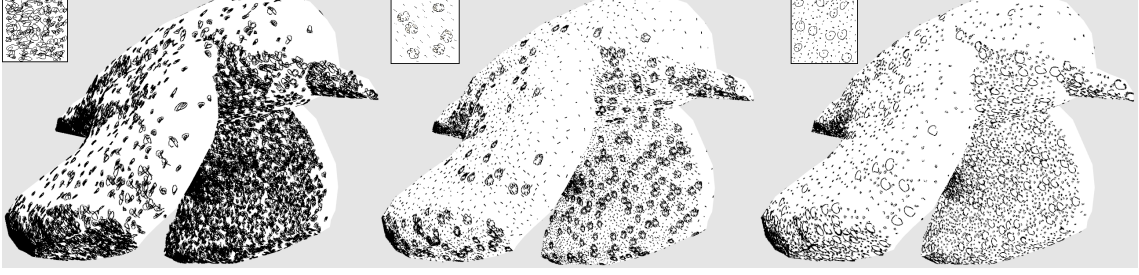


Figure 1.3 Results of tridimensional synthesis and respective input sample in the up left corner.

1.2 Contributions

The main contribution of this work is a simple and improved method for synthesizing distribution patterns based on an input sample of a vectorial image. This method was mainly inspired by the works of [BBT⁺06], [IMIM08] and [HLT⁺09], and the ideas of pixel-based and patch-based raster texture synthesis. The basic structure of the method can be used for both 2D synthesis and synthesis on surfaces of a general topology. This method is controllable by the density parameter, which have influence over the average distance between the elements. The density is also used in the tridimensional synthesis to produce the illumination effect. Also, unlike previous work that deals basically with black and white images, we introduced the RGB color system as a discriminant characteristic of the vectorial elements, in addition to previously used characteristics such as area, elongation, and number of internal crossings. This increases the variety of textures that can be processed by the method. Another contribution is the dissimilarity function used to numerically compare the difference between two neighborhoods. The function used in this work is sensitive to small differences between incomplete neighborhoods, providing a better response than previously used functions. The results obtained showed that the method can generate qualitatively better images for all the four mentioned types of textures and it can also be easily implemented and adapted for different applications, such as tridimensional texturing.

1.3 Organization

This document is divided into five chapters. The second chapter presents an overview of the existing bibliography on raster texture synthesis, discussing the evolution from simple pixel-based synthesis to patch-based, and tridimensional texturing. Also, we present the main approaches in the field of distribution pattern synthesis that inspired most of the ideas contained in this work. The third chapter presents a detailed description of the proposed solution, from the analysis of the sample input to the decision criterion used to identify a newly created element as correctly placed in the target. The fourth chapter presents results obtained by the proposed method, and discuss the limitations and improvements achieved. The last chapter presents the conclusions and directions for future work.

CHAPTER 2

Related Work

The synthesis of vectorial textures is a growing field. Recently, many works have been published and presented in conferences of great visibility in Computer Graphics ([BBT⁺06], [IMIM08], [HLT⁺09]). In this chapter, we discuss in details the approaches and the fundamentals of texture synthesis for raster images, presenting the background to understand the solution of this work.

2.1 Fundamentals

2.1.1 Textures

The definition of texture is essential for the development of a synthesis system. A texture can be roughly defined as *an image that contains some noticeable infinitely repeating pattern and some degree of randomness*. We can think of the obvious examples such as fabric or tiled bricks, however more complex examples exist, such as honeycomb textures composed of hexagonal cells with slight variation in size and shape on each cell, or interference patterns of waves on the surface of water. The amount of randomness will vary among textures and it is a fundamental concept for the classification of textures. We show in Figure 2.1 a qualitative division of several raster textures that illustrates the idea of randomness as classification for a texture. This idea can be easily extended to vectorial textures, as illustrated before in Figure 1.2.

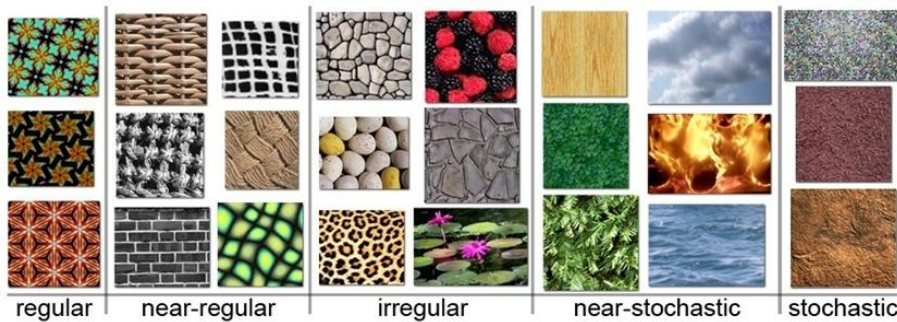


Figure 2.1 Spectrum of textures used in synthesis, according to the degree of randomness. Image adapted from [LHW⁺04]

Inside each category there is an infinite number of widely distinct textures, due to the differences of color, or elements contained within the patterns, but in the light of methods for texture synthesis, randomness is a valuable characteristic. Methods usually are successful to synthesize

regular or random patterns, but have difficulties to work in the middle term. The ideal method should be able to perceive and reproduce the pattern present on any type of texture.

The synthesis of distribution patterns is the main goal of this work. As it will be seen later, a vectorial texture will be reduced to a distribution pattern. Vectorial textures are essentially the same as raster textures, an image with repeating patterns and some randomness, but differ on how the data is defined.

A raster texture is a collection of pixels in a regular grid, described by color components. It is not, a priori, easy to extract high level information of the texture, such as the structure contained on the image. Vectorial textures, however, are a collection of strokes described by vector data, that presents some degree of regularity and some degree of randomness. In this sense, vectorial textures are a subset of raster textures. Each element in the texture is described by splines. There is no underlying discrete grid that imposes a limitation on the coordinates of the control points of the splines, and consequently, the points of the curves that define each element are located in the 2D Euclidean space.

2.1.2 Sample Based Texture Synthesis

Textures can be obtained in many ways, from scanned images to hand-drawn pictures. Alternatively, sample based texture synthesis automatically generates textures from small samples given as input.

When perceived by a human, the generated texture must appear to share the same visual features of the original sample texture. Thus, there is the challenge of capturing those characteristics from a given finite example. The synthesis process should be able to capture both the regular and the stochastic aspects of the sample. Therefore, a sampling procedure that synthesizes a new texture that maintains those features must be developed.

On raster texture synthesis, a pixel is the basic building element. From one pixel, only the color at a particular location can be known, but the visual system operates at a higher level and entire regions of pixels are perceived. Seams discontinuities are easily created and perceived by the user.

With distribution patterns, entire objects are copied on the target image at once, instead of pixel-by-pixel. Consequently, no discontinuities can be generated. This is an advantage of vectorial images synthesis over raster textures synthesis, that will be discussed later.

2.2 Methods for Raster Texture Synthesis

In this section, the main methods for synthesizing raster textures are presented. But first, it should be stated that the most successful texture model is based on Markov Random Field (MRF). MRF models a texture as a realization of a local and stationary random process. Each pixel in the image is characterized by the set of its spatially neighboring pixels, and this characterization is the same for all pixels. This scheme is illustrated in Figure 2.2.

Based on the MRF model, the goal of texture synthesis is stated as: given a sample input, synthesize an output texture so that for each pixel, its spatial neighborhood is similar to at least one neighborhood in the input. The size of the neighborhood, or the visibility of the

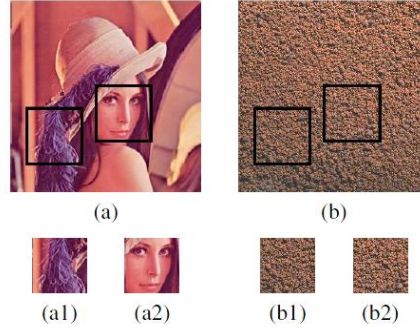


Figure 2.2 MRF experiment on a general image (a) and on a texture (b). On the general image, when looking at two different locations (a1,a2), under a limited visibility range, each location is clearly distinct. However, in the image, the two locations (b1,b2) are similar. Also, the central pixel from the windows in (b) is determined only by its visible neighbors. Those characteristics are the stationarity and locality properties, respectively.

pixel, is a user specifiable parameter. The similarity between neighborhoods in the target and sample textures will guarantee the perceptual quality. Also, the synthesis must be efficient and controllable.

2.2.1 Pixel Based Synthesis

The methods for pixel-based synthesis basically synthesize each pixel individually following the MRF model, in which the color is determined by the neighborhood. Here it is explained the seminal idea presented by Efros in 1999 [EL99], which initiated the whole field of texture synthesis from samples.

First, the target texture is seeded from a small portion of the sample. Starting from the seed, new pixels are generated outwards in a spiral path. For each generated pixel, a fixed-sized window centered at the pixel is intersected with already synthesized pixels. This collection of intersected pixels is compared throughout similar shaped collections in the sample image, to find the most similar N candidates. A random candidate is chosen, and the central pixel of this candidate is drawn into the target pixel. This process is repeated for each non synthesized pixel, until all of the target space has been covered. The only parameter defined by the user is the window size, or visibility. Intuitively, this size should be as large as the features of the sample texture. The use of exhaustive search for neighborhoods causes the algorithm to run fairly slow.

The later works of [WL00] and [Wei02] proposed a similar algorithm, but instead of a spiral ordering and a neighborhood based on the intersection of the window and synthesized pixels, they introduced a scan-line ordering and a fixed shape and size of neighborhood, inspired by [PP93]. The algorithm is illustrated in Figure 2.3 and works as follows: First, the output is initialized as random noise, having each pixels is drawn with a random color value between the minimum and the maximum possible values. The output is visited in a scan-line ordering, each pixel individually. A spatial neighborhood around the pixel is searched through the input sample, for the best match, and the corresponding pixels is copied. The important differences between this algorithm and [EL99] is that it is completely deterministic, because the best can-

didate is used instead of a random sampling. This implies that the output is invariant for a given initial condition. Also, the scan-line ordering enables the use of a fixed shape neighborhood.

The used metric to determine the best match of neighborhoods is the $L2$ norm of the RGB channels of the pixels. Given two windows of pixels, the distance between them is the sum of the euclidean distances between paired pixels, using the RGB color as the coordinates.

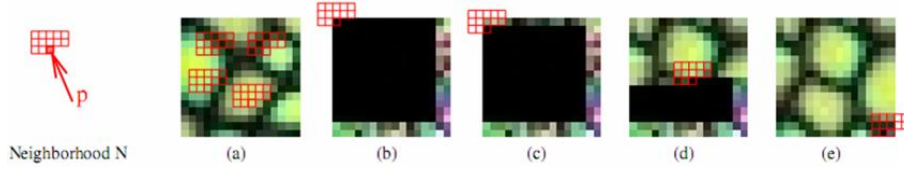


Figure 2.3 Scan-line ordering synthesis [Wei07]. Figure (a) is the input sample and the output starts as noise. Only the last few rows and columns are used, as shown in (b). The unused noisy pixels are presented as black. Neighborhoods crossing the output boundaries are handled toroidally. From (b) to (e), different pixels being synthesized are shown.

It was empirically observed that the quality of the result strongly depends on the size of the window. Intuitively, it should be on the scale of the largest regular structure of the texture, in other words, big enough to capture low frequency structures. Otherwise, the structure will be lost and the result image will look too random, as illustrated in Figure 2.4. Also, the shape of the neighborhood will affect the final quality. It must be causal, i.e., the neighborhood should only contain those pixels preceding the current output pixel in raster scan order. With this, only already synthesized pixels will be included on the neighborhoods, with exception of the last columns and rows, which will be used at the very first steps of the algorithm.

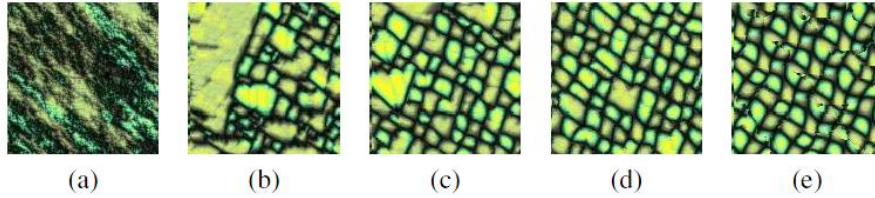


Figure 2.4 Synthesis results with window size of (a) 1x1, (b) 5x5, (c) 7x7, (d) 9x9 and (e) 30x30. As the size increases, the result quality tends to increase too, as well the computational cost. [Wei07]

As further improvement of the basic algorithm, multi-resolution pyramids [HB95] were used for textures with large structures, and coherence [Ash01] for improving speed. Multi-resolution becomes necessary when the large structures represented on the texture are too big to be captured by small window sizes. Increasing the size of the window will cause the rapid growing of computational cost, so multi-resolution pyramids are used. The lower resolution levels will capture larger structures, and this information will be transmitted to the higher resolution levels without increasing the size of the window. The coherence idea comes from the observation that when pixels are copied, it is very unlikely that they will land on random output locations. Instead, pixels that are together in the input sample, have a tendency to be also together on the output.

2.2.2 Patch Based Synthesis

The quality and speed of pixel-based approaches can be improved by synthesizing whole patches rather than pixels. Intuitively, it can be thought that assembling patches rather than synthesizing individual pixels improve the quality, since the pixels within the same copied patch will maintain the visual coherence found on the sample. The figure 2.5 illustrates the idea of patch-based texture synthesis. It can be seen that this is an extension from pixel-based algorithms, with the difference that instead of copying pixels, entire patches are copied. As illustrated, to ensure output quality, patches are selected according to their neighborhood, which is a thin band of pixels around the unity being copied. The major difference between the two approaches lies in how the synthesis unity is copied onto the output. In patch-based algorithms, the issue is more complicated since a patch, being bigger than a pixel, overlaps with the already synthesized portions of the image, thus some decisions have to be made about how to handle the conflicting regions.

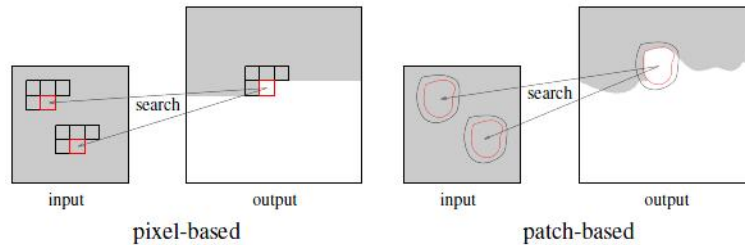


Figure 2.5 Comparison of pixel and patch-based algorithms. The gray region indicates synthesized portion.[Wei07]

In [PFH00], new patches simply overwrite existing regions. By using patches with irregular shapes, this approach took advantage of the texture masking effects of human visual system, and works well for stochastic textures. Lin [LLX⁺01] took a different approach by blending the overlapping regions. This can cause blurry artifacts in some situations. Instead of blending, Efros [EF01] uses dynamic programming to find an optimal path to cut through the overlapped regions, and this idea is further improved by [KSE⁺03] via graph cut. Finally, the other possibility is to warp the patches to ensure pattern continuity across the boundaries ([SCA02],[WY04]). Figure 2.6 illustrates the solutions for the overlapping problem.

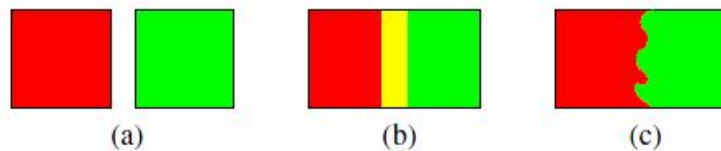


Figure 2.6 Methods for handling adjacent patches during synthesis. (a) two patches shown in different colors. (b) the overlapped region is simply blended from the two patches. (c) an optimal path is computed from the overlapped region.

Cohen et al. [CSHD03] present a tile-based texturing algorithm that shares a similar philosophy to previous patch-based algorithms. They used tiles as the basic units for synthesis.

However, instead of using arbitrarily shaped patches and handle patch overlaps via blending or cutting, they used a special kind of tile called *Wang Tiles* which have no overlap with each other and whose contents are constructed carefully so that the texture patterns are continuous across tile edges that have compatible colors. Once this tile set is constructed, an arbitrarily large texture can be constructed by assembling these tiles so that adjacent tiles share identical edge colors.

2.3 Texture Synthesis on Surfaces

Although a variety of algorithms have been developed for synthesizing textures on regular 2D grids, one of the end goals is to be able to similarly texturize surfaces. In this section, it will be discussed the main three categories of algorithms used to accomplish this task. It is assumed that a 3D polygonal mesh and a texture exemplar have been provided, and we wish to create a texture in the mesh that is based in the sample. In addition to the essential step of synthesizing the texture, two extra tasks need to be accomplished when dealing with surfaces. The first is how to specify the orientation and scale of the texture on the surface, and the second is how to render the texture on the surface, after it has been synthesized.

2.3.1 Scale and Orientation

Some textured objects can exhibit variations in the scale of the texture features at different portions of the surface. For example, the spots on the legs of a giraffe are smaller than on its body. A complete texture synthesis system must allow the user to define and control the scale of the texture over the surface, as a scalar field. Most often, this is done by a sparse interpolation method: the user specifies the scale values at a few selected locations, and the system smoothly interpolates the values over the surface.

Many textures have characteristic features that are oriented in a particular direction. Examples of oriented textures include woven fabric, bricks, animal stripes and wood grain. When synthesizing textures on a 2D grid, it is usually understood that the new texture will have the same orientation as the sample, however this is not true for tridimensional surfaces and it is typically left for the user to specify the orientation field across the surface. Similarly, the system provides the capability for the user to define key vectors in specific points in the surface, and then, smoothly interpolate them to generate a dense vectorial field.

The work of Praun et al. [PFH00] is considered the first texture synthesis approach for surfaces that allows the user to specify a vector field to guide the texture. They used Gaussian radial basis functions to interpolate sparse user constraints across the surface. Their user constraints were tangent vectors at specific user-chosen points. For distance measures over the surface, they made use of Dijkstra algorithm. More recently Zhang et al. [ZMT06] used radial basis functions to extend not just tangent vectors but also singularities such as sources, sinks and saddles.

A different approach is to use a hierarchical blurring procedure to extend tangent vectors over the surface [Tur01], as illustrated in Figure 2.7. For textures with a particular symmetry, a tangent vector per location might also be used, but with a global angle parameter that specifies

the amount of rotational symmetry [WL01]. Figure 2.7 shows various forms of directional guidance for two textures.

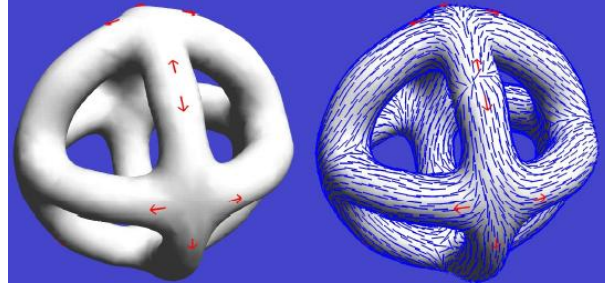


Figure 2.7 User constraints for a vector field (left) and the resulting vector field (right) (from [Tur01]).

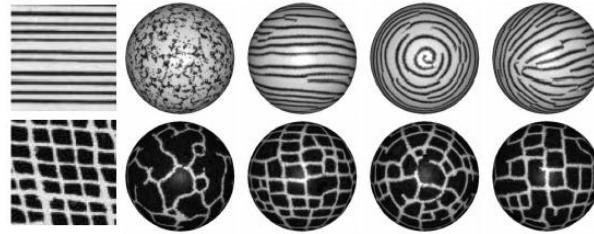


Figure 2.8 Various orientation fields are used to create textures on a sphere from [WL01].

2.3.2 Dense Point Distribution

In this approach to perform texture synthesis on surfaces, the surface is first populated with a dense set of points. After an orientation field has been provided, these points can then be treated almost as pixels in a pixel-based synthesis approach. When a regular distribution of points is desired, the most common approach to create a dense sampling of a surface is to use point repulsion. The idea is to sprinkle many points on the surface at random and then treat them as if they were particles repelling each other. The points slide over the surface according to repulsion forces until they settle down. Care must be taken to assure that the points remain on the surface when they are moved. This approach was used as a first step for creating reaction-diffusion patterns on surfaces [Tur91], for creating clonal patterns on animal coats [WFR98], and the same method was also used for more general texture synthesis [Tur01],[WL01].

Given a dense collection of points and a tangent vector field to orient the texture, a visitation order may be defined for these points. The idea is to mimic the left-to-right, top-down order of raster-based synthesis methods. However, Lefebvre and Hoppe have found that the visitation of pixels does not need to be in scanline order to obtain high-quality results [LH05], so other ordering methods might also be used for surfaces.

Each point on the surface can be initially assigned a random color, and then the usual matching can be performed between the exemplar neighborhoods and the neighborhoods on the surface. Synthesis proceeds by sweeping over the surface and replacing a point's current

color with the color from the exemplar whose neighborhood is the best match for it. The orientation field is used to help define the neighborhood of a given point. To find the color of an adjacent location on the surface, the idea is to crawl over the surface some distance along the vector field and some distance perpendicular to this vector field. The color at this position is determined by interpolation of colors from nearby points. By assembling many such colors, the entire neighborhood of a point is created. Figure 2.9 illustrates good results of this approach.

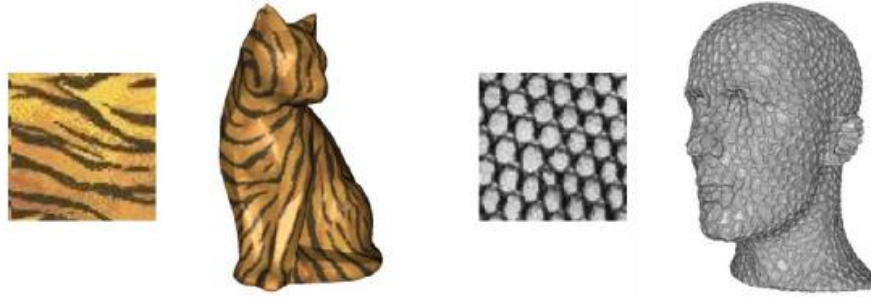


Figure 2.9 Results from the texture synthesis approach of Wei and Levoy [WL01].

2.3.3 Synthesis Over Texture Atlas

Another approach for texturing surfaces is to use the parametrization of the object, that can be previously supplied together with the mesh description or calculated on the fly. In this approach, the texture is synthesized over the 2D space of this parametrization, and the mapping between the surface and the plane must be taken into account to control the neighborhood creation.

Ying et al. [YHBZ01] perform their texture synthesis using such surface-to-plane maps. Their method decomposes the surface into a set of overlapping charts, where each chart is a mapping from a portion of the surface to the plane. The models demonstrated in this work are subdivision surfaces, which makes such chart creation straightforward. As with many methods, their approach is performed in a hierarchical manner using Gaussian pyramids. Each flattened portion of the surface has associated with it an output texture map to be synthesized. These texture maps are visited in breadth-first order, and within each map, the pixels are synthesized in scanline order. The neighborhood for a given pixel is created by marching over the surface outward from the surface point corresponding to the central pixel. Some results of this method can be seen in Figure 2.10.

Lefebvre and Hoppe also used a surface unfolding approach to synthesize textures on surfaces [LH06]. Similar to Ying et al., they use an atlas of charts for a surface to perform synthesis, but they concentrate on polygonal models rather than on subdivision surfaces. Their chart creation method attempts to create large charts composed of many triangles, but with minimal distortion. Through the use of distance fields and PCA data reduction, their synthesis approach only requires 5x5 pixel neighborhoods, which also allows the approach to be implemented using graphics hardware. This approach can also be used to perform texture advection and to create results based on pre-computed radiance transfer as illustrated in Figure 2.11.



Figure 2.10 Synthesis results from the unfolding method of Ying et al. [YHBZ01].

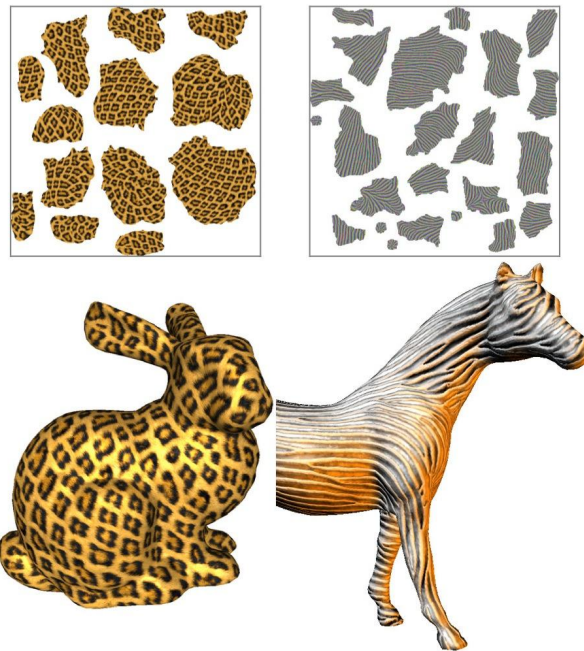


Figure 2.11 Synthesis results from the method of Lefebvre and Hoppe [LH06]. The upper row shows the texture atlases for the models on the bottom.

2.3.4 Triangle Patches

In contrast to pixel-based approaches, there are surface texturing methods that operate at the triangle level. One such method extend to surfaces the jump map approach of Zelinka and Garland [ZG03]. The basic method is to first perform a slow analysis of the texture in order to group the pixels in the sample accordingly to the similarity of neighborhoods. This allows a new texture to be quickly placed on a surface by assigning texture coordinates from the example to each vertex of a triangle mesh. Each vertex of the mesh is visited according to an order that is given by a user-controlled vector field, sweeping from sources to sinks in the field. New vertices are visited according to region growing by selecting the vertex that has the greatest proportion of already synthesized neighbors. Based on the texture coordinates of the neighboring vertices, texture coordinates are calculated for the new vertex. Often this will be

a location that is adjacent to the coordinates for nearby vertices, but sometimes a jump will be taken to a similar region. Synthesis results from this method are shown in Figure 2.12.



Figure 2.12 Texture synthesis results using the Jump Maps method of Zelinka and Garland [ZG03].

Another approach that treats triangles as patches is due to Soler et al. [SCA02]. This approach starts by creating a hierarchy of face clusters, where a cluster is a collection of triangles that are adjacent. Each coarse cluster is further divided into a number of smaller clusters. Then each face cluster is flattened onto the plane in a manner to limit distortion. Each vertex of the cluster will be assigned a texture coordinate after a mask that represents the fixed texture around the current patch is extracted and used to search in the sample for the best mask. The texture coordinates are propagated down the hierarchy and relaxation is used to improve continuity across edges. The comparison of neighborhoods is much like the search step used in 2D patch-based methods [EF01]. In order to accelerate this search, Soler et al. use an FFT to perform a masked sum of squared differences comparison. Figure 2.13 shows results from this approach.

2.4 Vectorial Pattern Distribution Synthesis

In previous sections many approaches to synthesize rasterized textures from a supplied exemplar were presented. In this section, vectorial texture synthesis will be focused, by presenting recently developed works on the field and explaining them in details. First, the definition of vectorial textures will be remembered. A vectorial texture is a collection of strokes, described by vector data, that presents some degree of regularity and some degree of randomness. Since SVG files are used, most of the geometry of the elements contained on the texture is described with splines, but line segments, arcs or any other geometric primitive can be used as well.

In principle, samples of vectorial textures could be converted into raster images and therefore, traditional synthesis algorithms could be applied on them. However, there are a number of reasons why a specific solution for this type of textures is important. The basic approach for texture synthesis has flaws such as the generation of discontinuities between regions. Although minimized by new algorithms, this problem still exists. A vectorial texture, as stated before, is a collection of basic finite elements. On existing algorithms, those elements are essentially reduced to locations, and so, the synthesis is made on the distribution of those locations. There-



Figure 2.13 Synthesis results using the face clustering method of Soler et al. [SCA02].

fore, the texture can also be called pattern distribution. The synthesis of such distributions will generate new distributions, which follow the same pattern of the sample. The elements from the sample are fully copied onto the locations of the new distribution, which minimizes discontinuities issues. For instance, a circle will always be a circle on the result. With raster texture synthesis, broken circles could be generated depending on the complexity of the sample.

Synthesis of distribution patterns will also perform more efficiently than traditional approaches, due to the low number of elements contained in an average pattern. For instance, it could be desired to generate a 1024×1024 texture of circles from a 64×64 sample. The large output resolution implies a high computational cost. With vectorial textures, however, if the same sample has 10 circles, the desired result would have in average 256×10 circles. The total number of elements to process is much smaller, and so is the computational cost.

2.4.1 Stroke Synthesis

Stroke pattern synthesis systems have been studied in the past, to generate stipple drawings [DHvOS00], pen and ink representations [SABS94][WS94], engravings [Ost99], and painterly rendering [Her98]. However, they have relied primarily on generative rules, either chosen by the authors or borrowed from traditional drawing techniques. For the proposed approach, we are interested on mimicking the texture pattern from the supplied sample texture, therefore, a sample-based synthesis technique. This gives the user freedom to choose his own style, instead of being attached to the specifically style supported by the method.

Kalnins et al. [KMM⁺02] described a method for synthesizing stroke deviations from an underlying smooth path, to generate new strokes with a similar appearance to those in a given example set. Hertzmann et al. [HOCS02] and Freeman et al. [FTP03] address a similar problem. Neither method reproduces the interrelation of strokes within a pattern. Jodoin et al. [JEGPO02] focus on synthesizing hatching strokes, which is a relatively simple case in which strokes are arranged in a linear order along a path. Results from these works are illustrated in Figure 2.14.

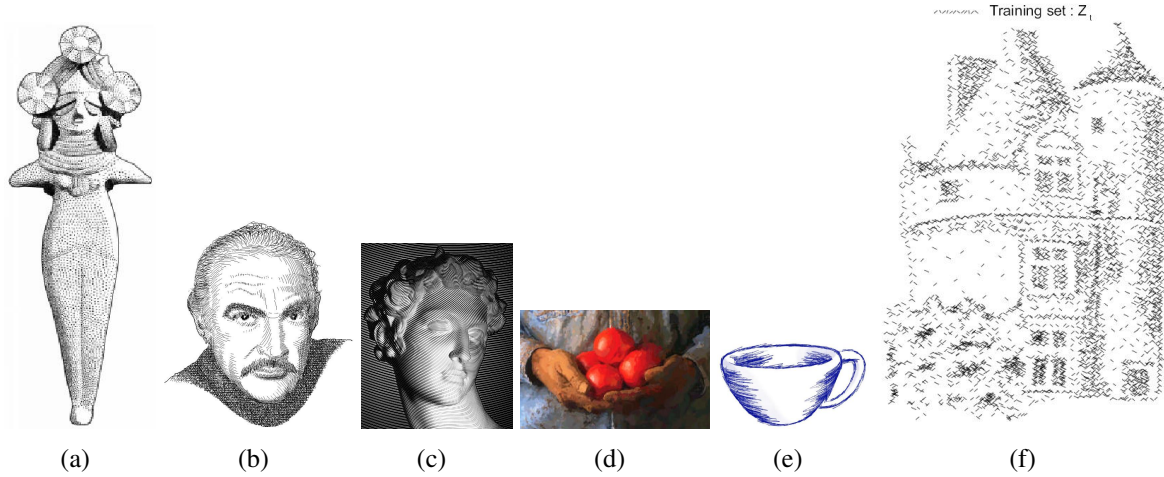


Figure 2.14 Results from (a) [DHvOS00], (b) [SABS94], (c) [Ost99], (d) [Her98], (e) [KMM⁺02], (f) [JEGPO02].

2.4.2 Stroke Pattern Analysis and Synthesis

Barla et al. [BBT⁺06] presented the first method to synthesize bidimensional pattern distributions, based on samples. The work focus on a wider range of patterns and take into account the global organization of the elements by means of neighborhood comparisons. The main task of their work is to synthesize a new vectorial texture from a small exemplar provided by the user. Also, this texture is not limited to a linear ordered string of strokes, but to a 2D distribution of unrelated strokes. The generated texture must follow the same style, or the same rules of construction, that generated the sample. This goal is the same of traditional texture synthesis methods, but they took inspiration on ideas from works of non-photorealism [Her03] where the main goal is to reproduce the style used to draw the original image. Here, it is also considered that all the necessary information about the pattern presented on the texture is self-contained on the provided sample.

The method has two steps: analysis and synthesis. In the analysis step the supplied sample image is processed, the collection of strokes are clustered and simplified into a collection of objects, and the triangulation of the distribution of these objects is evaluated. Here, some simplifications of the input image are made. The first is to consider the distribution pattern of the drawn objects almost regular. In other words, it is not possible to find regions of great density and regions of coarse density, on the same texture. Being so, it can be thought that

every texture has an underlying grid that resembles an almost near regular triangulation of the 2D space. The vertices of this triangulation are the centers of the objects on the texture.

A second simplification is to consider that the objects from the texture do not intersect each other. Since the input sample is only a collection of strokes, the analysis step will group them into objects, accordingly to restrictions of perceptual grouping of line segments [AJG⁺91]. This grouping is controlled by a parameter that will represent the biggest size of one element from the texture, which must be supplied by the user. This parameter avoids the analysis step grouping together two strokes of distinct objects. Each stroke is initially processed and the oriented bounding box is evaluated. A clusterization step will use metrics of proximity to determine if two distinct elements are part of the same group. For those metrics, they used the Hausdorf distance between bounding boxes. The alignment of distinct strokes is also checked to determine if two separated strokes are part of the same drawing element. Initially, each stroke is one distinct group, so that pairs of groups are visited and grouped together if they are perceptually close. The resulting data is a much smaller collection of elements, the objects, represented by bounding boxes. When two objects intersect each other, this algorithm will classify their strokes into the same group.

The last step of the analysis is to evaluate the spatial distribution of the objects. For that, they used the Delaunay triangulation, using the center of the bounding boxes as vertices. Since they assumed the input distribution as almost regular, it is expected that each element is slightly displaced from the barycenter of its neighborhood. This displacement vector is evaluated and stored as information of the style. Figure 2.15 summarizes the analysis step.



Figure 2.15 Grouping and triangulation of the analysis step of [BBT⁺06]

Each object resulting from analysis is described by the oriented bounding box and the color. Barla argues that more characteristics may be attributed to the objects, but in this work, only those two are used. Also, it is not well explained how the color is treated during the analysis, but in every example presented, there is no grouping of strokes with distinct colors. So, we assumed that the final object receives the color of any of the strokes contained on it.

Following the analysis, there is the synthesis step. This step is strongly based on the previously presented algorithms for synthesis of raster textures, where they perform a comparison of neighborhoods for the search of the best matching element. Before synthesis is performed, they must compute what is the underlying grid that will be the structure of the texture. For raster images, the grid is the regular pixel matrix, but for vectorial textures, the elements are freely placed on the 2D space. The initial simplification of a regular distribution has a decisive

influence here. The resulting texture must follow the same constructive rules of the sample texture, and one of those rules is that the distribution of the elements is almost regular. Therefore, an almost regular distribution of points is evaluated according to Lloyd's method [Llo82]. For this distribution, the triangulation is evaluated. The vertices of the triangulation will be non-synthesized elements, which they call a seed.

Once the initial position of seeds are determined, the next step is to replace each one with an appropriate reference to an element from the sample, in such a way that the spatial relationship between distinct objects is well preserved and the style of the drawing is correctly copied. Exemplifying, in a texture of spirals and lines equally distributed on the space, it is incorrect to have small regions with several spirals and no lines among them.

From the triangulation, the neighborhood is evaluated for each object on the sample and each seed. The range of visibility for the neighborhood is a parameter supplied by the user, but since the textures used on this work contain few elements, the neighborhood is always defined as the 1-ring of neighboring vertices. The synthesis proceeds similarly to the raster texture synthesis. The first seed references any object from the sample, and then each non-synthesized seed connected to synthesized seeds are visited. For each seed, the best match of neighbors is searched on the sample, and the characteristics of the chosen candidate is copied to the seed. This process is repeated until all the seed have been visited.

In raster synthesis, the neighborhood of a pixel is compared by taking pairs of neighboring pixels located at the same relative position on both neighborhoods and evaluating the Euclidean distance between their colors. The sum of all the distances gives the difference between both neighborhoods. In [BBT⁺06], the neighbors distribution is not regular, and therefore, the relative position of an element will coincide with the relative position of some other element on the other neighborhood. The approach of Barla is to find pairs of elements whose relative positions are close, and discard the remaining unpaired elements. Non synthesized seeds are also discarded as valid elements. For each pair, a comparison of color, shape and a set of three other measures is done. If the color is distinct, the pair is discarded. The metric to compare the shape of two elements is the Hausdorff distance between the bounding boxes. If the metric results in a similarity above some threshold, the pair is accepted. Otherwise, the three remaining attributes are compared. Those attributes are parallelism, overlapping and superimposition. Parallelism compares the inclination of the bounding boxes, overlapping and superimposition compare the width and height of the bounding boxes. If one of those three measures is outside the accepted range of similarity, the pair is discarded. The error is given by the sum of perceptual measures of each accepted pair. During synthesis, not all the neighboring seeds are already synthesized, the sum of the errors is normalized by the number of accepted pairs. This approach gives the same relevance to neighborhoods that contain only one or many more pairs, but in practice, it was showed to be enough. The threshold values are empirically determined, and the system is robust to small variations of those values.

When the synthesis is complete, each seed is displaced by some distance in the direction of the displacement vector previously evaluated, in the last step of the analysis. This will improve the similarity of the styles, but it also gives the user the possibility of controlling the generated texture with small variations. In Figure 2.16, it is observed the difference among the results with and without this displacement.

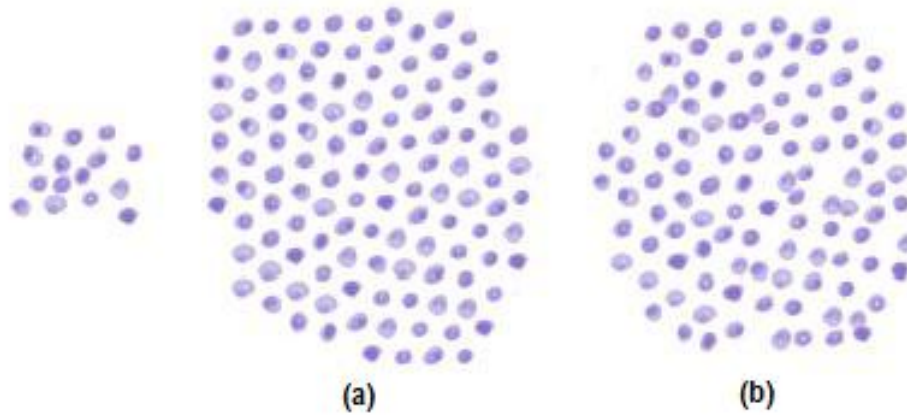


Figure 2.16 Synthesis by Barla [BBT⁺06], without displacements (a) and with displacements (b).

2.4.3 An Example-based Procedural System for Element Arrangement

Inspired in the work presented by Barla [BBT⁺06], Ijiri et al. [IMIM08] developed a similar approach for the example based synthesis for element arrangement. The aim is essentially the same: the generation of a larger arrangement pattern, visually similar to the reference arrangement supplied by the user. The main difference consists on using a procedural growth algorithm to generate the new arrangement. Also, the growth can be controlled through the use of a flow field. They introduced three types of tools for controlling global aspects of the resulting arrangement: an spray tool, the flow field tool and the boundary tool.

The analysis step is similar to the proposed in [BBT⁺06], however in this work, they considered that the collection of strokes of the input sample is already correctly classified and grouped into distinct objects. For the analysis, remains the task of evaluating the Delaunay triangulation to extract connectivity between neighbors. It was noted that this approach usually generates skewed triangles around boundaries, which are removed by discarding edges that are not part of at least one un-skewed triangle. Also, the elements around the boundary are not used because they do not have enough neighborhood elements. When regular or near-regular arrangements are used as input, Delaunay triangulation often generates undesired connections. Since the algorithm is strongly affected by local connectivity, these connections can break the regularity. The solution proposed was to allow a manual correction by flipping an edge by clicking on it.

The synthesis step begins with an element at the center of the pattern and expands outward by placing a new element, one by one, based on neighborhood comparisons on the previously synthesized elements. As stated before, the target space does not have regular structures as a grid of pixels, so the position for placing a new element and its neighborhood is unclear. To tackle this problem, they applied a procedural system that defines a local growth of seeds based on connectivity of elements [PL90]. At each growth step, a seed is replaced with the best fitting element, and new seeds are placed by copying the immediate neighborhood of the chosen reference element.

A seed is a candidate position for a new element that optionally has a symbol id associated

with it. This id represents a collection of objects that share the same id, and therefore, the seed could be replaced by any of them. The id information is already provided with the sample texture. The synthesis process begins by placing a single seed. Seeds are sorted by distance from the initial seed and the nearest is chosen in each growth step. In the spray tool mode, the initial seed is generated at the cursor position, and seeds are grown inside a user-specified distance from the cursor during the painting. Priority of elements is based on the ids, such that seeds of high priority are synthesized first.

The neighborhood comparison for the element arrangement is computed in two steps, similarly to [BBT⁺06]. First, they construct a set of matching patterns for a candidate element. Second, for each matching pattern, they calculate an error function as a weighted sum of the euclidean distances between relative positions of neighbors, the id function that returns 1 when two elements share the same id, and 0 when they had different ids, and a function that returns 1 when the two elements are the same. The best pattern results on the smaller error value.

Three modes are introduced to generate different results for different purposes. In the non-rotation mode, the comparison is made as mentioned earlier. In the rotation mode, the reference elements can be rotated to minimize positional differences. They applied the shape matching problem introduced by [MHTG05]. In the flow field mode, the user specifies a vector field on the plane, and the orientation at the target seed position is obtained from it. The whole reference pattern is rotated.

Once the best element is found, the system replaces the seed with the found reference and introduces new seeds by using neighboring elements. All the elements from the 1-ring neighborhood are overlayed on the target space, and the elements that were paired during the neighborhood comparison are discarded. The remaining elements will form new seeds. Edges are constructed to connect the new seeds. To avoid dense or sparse distributions, or collisions of edges, several heuristics are used. For each new seed, a neighboring area is checked for the presence of any element. If an element is found, the seed is discarded and an edge is connected to this element. The radius of this circular area is chosen as half the length of the shortest edge from the sample. Then, the angle between edges is checked. If the angle is smaller than half of the corresponding angle in the chosen reference ring shape, the new seed and the edge are deleted. Also, any possible collision of the new edge is checked. If the length of the new edge is greater than 1.5 times the original length, an extra seed without id is placed at the edge midpoint.

Finally, a relaxation process controls the accumulating error due to the local growth step. This process tends to displace elements on the target arrangement in such a way to keep the local features of the synthesized pattern as similar as possible to that of the reference pattern. Some results are illustrated in Figure 2.17.

2.5 Appearance-guided Synthesis of Element Arrangements by Example

In a recent work, Hurtut et al. [HLT⁺09] tackle the same problem of synthesizing arrangements of vectorial patterns. Their solution shares similarities and differences with the two works presented earlier. Here, the input sample is a collection of already built elements, but they are not explicitly labeled. The main contribution is that the arrangement analysis and re-synthesis

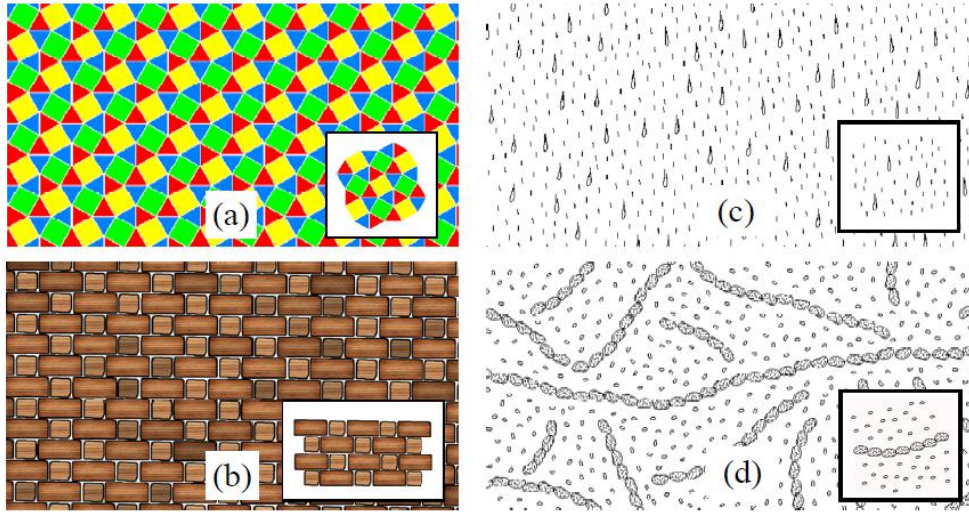


Figure 2.17 Synthesized arrangements and input sample patterns (from [IMIM08]).

is formalized as a statistical learning problem. Another contribution is the use of dominant element appearance traits as soft constraints influencing the synthesized distribution itself.

The first step of the method aims at categorizing the elements from the sample according to their appearance. If some elements exhibit a similar appearance, they should be recognized as belonging to the same category. Elements considered to be unique, will be grouped in an outlier category. This is an automated solution for the manual labeling of [IMIM08]. As a reason for the use of appearance-driven element categorization, they argue that according to the Gestalt law of similarity grouping, the human visual system tends to mentally perform perceptual categorization and build groups from isolated elements. Once those ensembles are established, visual interactions can arise. Inter category interactions are also important to devise a good capture of the visual attributes of the arrangement.

Julesz [Jul86] studied human perceptual discrimination of textures composed of stroke-based elements called *textons*. In his theory, discriminative features include principal orientation, the number of crossings and extremities. Those features are used as descriptors. The area and elongation are included as well, since the textons of Julesz all shared the same size. Crossings and extremities are measured directly from the strokes constituting the element, while the other features are measured from the oriented bounding box.

Grouping together elements that share similar features is done by comparing features that drastically capture different visual characteristics. Before comparison, features are normalized on $[0,1]$. This interval must cover enough visual variation for each characteristic. Orientation is normalized by 2π . Elongation is normalized by 3, a value used as a threshold. Any elongation greater than 3, is normalized to 1. The area is normalized by the limit of 5% of the reference sample area. The number of crossings and extremities are normalized by 10. Other descriptors could be added, at the cost of possibly introducing redundancy and increasing the dimensionality of the descriptor vector.

The categorization is made in two steps. First, the elements are categorized according to

the area of the bounding box. Each of the resulting groups are then categorized independently, according to the dominant feature. For that, it is performed a dimensionality reduction. The motivation behind this is that visual perception argues that size is the first information to be perceived for visual recognition tasks. And secondly, as Julesz observed in his studies, it often happens that not all the features contribute to perceptual categorization steps. Dimensionality reduction by PCA will compact the information of the features, around the most relevant direction, minimizing the effect of redundancy and noisy features. The grouping is established by detecting relevant nodes, according to the *a contrario* method [DMM03].

For the synthesis step, the arrangement of elements is modeled according to a statistical process. Once the parameters of this model are learned on the reference arrangement, the synthesis consists in running realizations of this model at the user-supplied scale, shape and density. For the modeling, the spatial arrangement is captured via a multitype point process, a statistical model dedicated to analysis of interactions between a finite set of typed categories. The spatial organization and underlying correlation between the appearance of elements is grasped by considering pair-wise element distances as interactions between the categories. This model supports a wide range of distributions, from stochastic to near-regular.

The model process is easily simulated using Markov chain Monte Carlo methods. This provides a convenient means to generate new arrangements that apparently obey the same stochastic process as the provided sample. Figure 2.18 illustrate several results obtained with this approach.

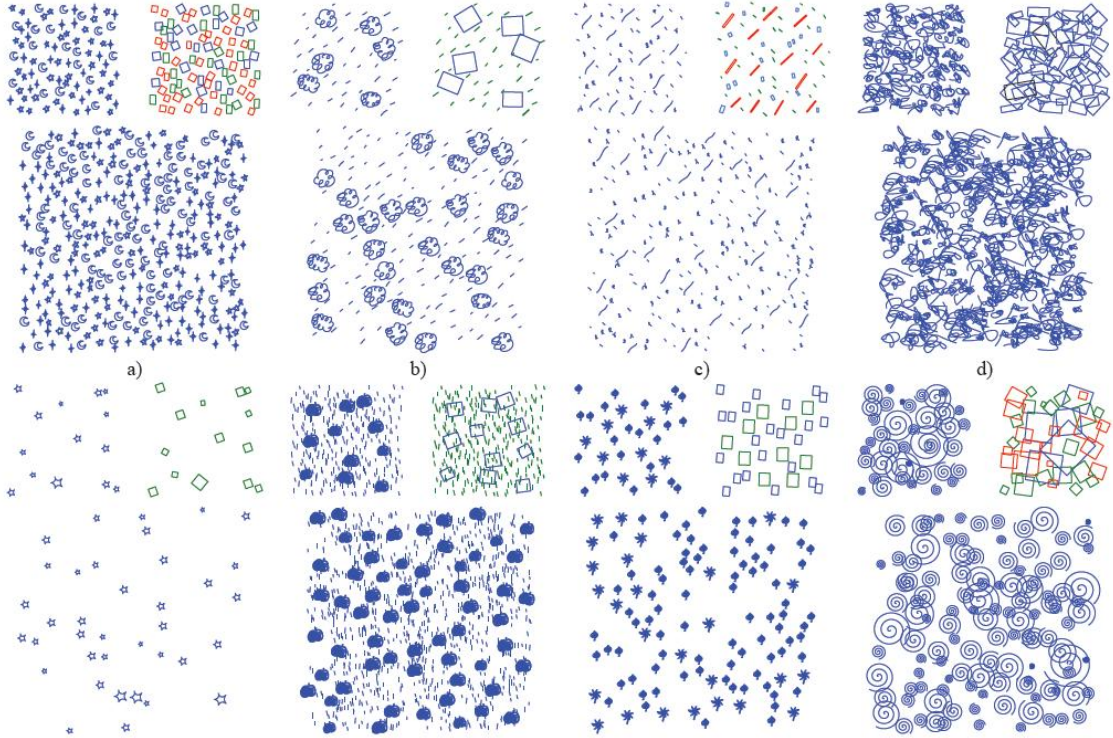


Figure 2.18 Synthesis results attesting the variety of distributions that the model of [HLT⁺09] can handle.

2.6 Discussion

Raster texture synthesis methods are highly developed nowadays, both in quality and efficiency. Although pixel-based methods can generate good results, computational cost can be prohibitively high unless one uses offline computation or specific hardware. Patch-based methods are faster, but less controllable because the interior of a patch is unchangeable. In 3D synthesis, triangle patches methods are fast, but achieving variability of the texture is difficult because the mapping is made in the same space of the sample texture. Synthesis with dense points over the surface has the same problems of pixel-based methods, but is much simpler than synthesis over texture atlas. In vectorial texture synthesis, methods that use statistical models are able to capture well the characteristics of a texture, but procedural methods are much easier to implement and understand.

The proposed approach presented in this work shares the same goal of most recent works, which is to synthesize a distribution pattern based in a given sample. The method can be extended to texturize tridimensional meshes and the element color is considered as a relevant characteristic for the synthesis. The same basic structure of [BBT⁺06] is used, to perform the synthesis in two steps: analysis and sampling. However, the proposed analysis step is a simplified method of the one used in [HLT⁺09] and the proposed sampling step is based in the procedural growth of [IMIM08], but with a distinct neighborhood comparison metric. The proposed method, also, provides local control of density of elements. The next chapter presents the method in details.

Proposed Method

This chapter presents a detailed description of the developed algorithm, explaining the use and motivation behind the choices with several illustrative cases.

3.1 Overview

Following the same structure of previous work on arrangement pattern synthesis, the proposed method has two main parts: classification and synthesis. The first part represents the bridge between the synthesis core and the user's input. Its role is to process the provided sample, extracting relevant information for the next step, redeeming the user from constructing images with complex descriptions.

In this system, the user provides a sample input image and the target area, in terms of size. The sample is defined as a collection of elements, where each element is described by vectorial data such as quadratic or cubic splines. The description of a single element may contain several disconnected splines. The Scalable Vectorial Graphics (*SVG*) format naturally supports this type of image and is used on all of the results.

In the classification part, the elements from the input sample are grouped together into collections according to similarity measures such that, ideally, any two elements of the same collection should be visually similar. From this resulting data, important features used to control the synthesis is evaluated. They correspond to the minimal distance between pairs of distinct elements from two groups, as illustrated in Figure 3.1.

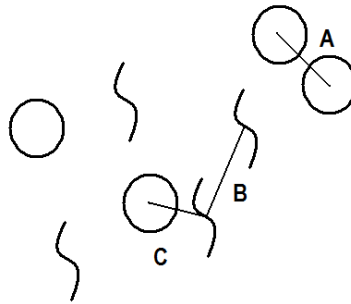


Figure 3.1 Features evaluated for an illustrative example. Here, the elements were grouped into two distinct groups, curve and circle. In this case, three features are evaluated. The smaller distance between distinct circles (A), distinct curves (B), and circles and curves (C).

The synthesis initially divides the target output image into squared regions of appropriate

size. A recursive procedure starts by visiting a random region and placing a seed in a random position on its interior. This seed is replaced by a chosen element from the sample and new seeds are generated accordingly to the relative position of the neighborhood of this element on the sample. The list of seeds grows iteratively until the target space is completely visited, in other words, until there are no unvisited regions left. During synthesis, keep a list of seeds located in the target image space. This list is FIFO, i.e. the first seed from this list is extracted and replaced by a reference to one element from the sample, inserting new seeds at the end of the list. Therefore, the texture grows in a spiral order starting from the initial seed. If eventually the list of seeds becomes empty, a new seed is randomly created into a random non-visited region. The growth is illustrated in Figure 3.2, and shares many similarities with the method proposed by [IMIM08]. The main difference is the absence of underlying triangulation of the arrangement, which will provide the control over the local density as will be explained later.

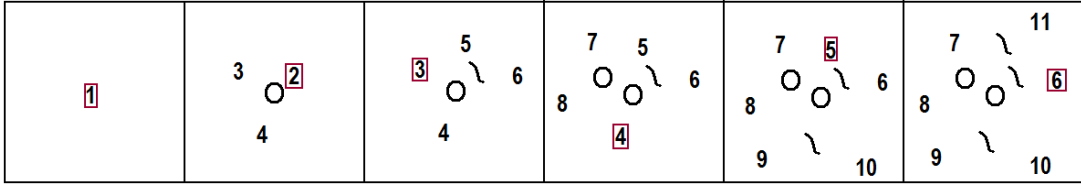


Figure 3.2 Illustration of the local texture growth. Each seed is represented by a number, and the highlighted seed is the first on the list.

3.2 Input classification

Unlike raster texture synthesis, this approach aims the synthesis of arrangement of elements instead of the synthesis of pixel colors. The synthesis follows a texture pattern provided by a 2D sample. Initially, such elements are ungrouped and defined only by their own vector data.

In recent works, in order to identify a texture, the sample is processed and the elements are classified into groups, where each group contains only closely similar elements. Then, geometric features regarding the inter-relation between groups is used as information that can fully describe the texture. These features can be coded on the triangulation of the distribution, or in the parameters of an statistical model. In this work, however, this information is used directly for procedural growth.

3.2.1 Characteristics of Elements

In order to classify the elements, the data must be simplified and evaluate relevant characteristics. Ideally, the number of those characteristics should be the least necessary to fully identify each group on the pattern, much as a human observer would use a limited number of characteristics to naturally identify groups.

First, the data of each element is simplified by evaluating the oriented bounding box and using its center as the position of the element. Then, all the necessary characteristics used to

classify the element can be found. The same approach of [HLT⁺09] is used with minor differences. First, the characteristics are area, elongation, orientation, number of internal crossings, and color. The number of extreme points is not used, since the authors did not provided a detailed description of how they evaluate such information. However, it was noticed that the absence of this characteristic does not affect the results on many textures, since other characteristics are enough to distinguish elements.

The area of the oriented bounding box is simply the length times the height. Elongation is the size of the longest edge divided by the shortest. The orientation is taken as the angle between the principal axis and the horizontal direction. The principal axis is the longest edge. Here, we face the problem of pseudo symmetry, since the orientation of the same element can be equally evaluated to point towards the opposite direction, but the element itself may not be symmetric. Previous works did not addressed this problem, but the samples usually contains symmetric elements or other elements that can be differentiated by others characteristics than orientation. Just choosing to use as orientation the angle closest to zero is good enough. The number of internal crossings is evaluated from the original vectorial data of the element. Count the number of times that a line segment crosses another segment from the same element. For the color, take the RGB value and convert to the grayscale value using [KOF08]. All the characteristics are normalized, as explained in the next section. Figure 3.3 illustrates the characteristics.

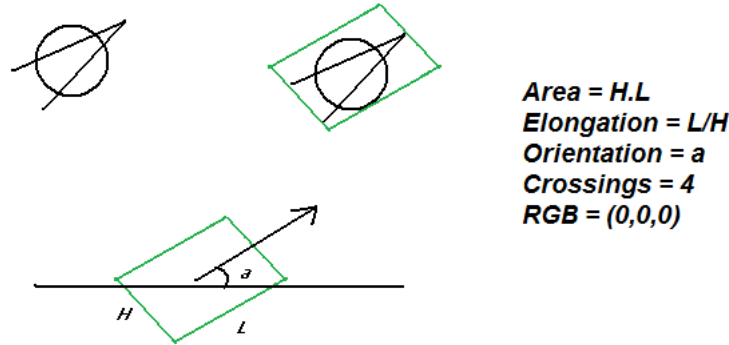


Figure 3.3 Example of the characteristics evaluated for one element and used to classify it, before normalization.

3.2.2 Principal Component Analysis

In the analysis step, Principal Component Analysis (PCA) is used to achieve dimensionality reduction of the vector of features evaluated for each element. For the sake of completeness, it will be briefly explained how the reduction performs.

Suppose there is a data set of points, each one described by N values, or a N -dimensional set. PCA is used to find a new coordinate system that better describes this set. In this case, better means that the inter-relation among the dimensions is captured. For instance, the set of points $P = \{(1, 1); (2, 2); (3, 3)\}$ can be better described if instead of the vectors $(1, 0)$ and $(0, 1)$, the normalized vector in the direction of the line $x = y$ were used. In other words, the

data set P would be described by only one dimension, and the distinction between points would be maintained. If the direction $x = -y$ were chosen, all the points would be projected in the same value.

This better coordinate system is formed by the eigenvectors of the covariance matrix of the data set. The eigenvector associated with the highest eigenvalue is the chosen vector in which the points are projected. Therefore, an N -dimensional set becomes unidimensional.

3.2.3 Classification Description

The classification of elements is performed in two stages. The first stage corresponds to the classification by area, and the second stage corresponds to the reduction of the four remaining characteristics, by PCA to avoid high dimensionality issues, and the classification by this new data. Histograms for those values will be constructed, and from them, groups can be defined. A simplification of the process compared to previous work by [HLT⁺09] is made. Instead of using an *a contrario* method to find relevant modes on the histogram, use a simple 5 bins histogram, where each non-empty bin becomes a group itself. Five bins were chosen as an initial guess due to the visual observation of the histograms generated for all tested samples. In practice, this number was enough to classify all samples with straightforward implementation, although for a few patterns there are over categorizations.

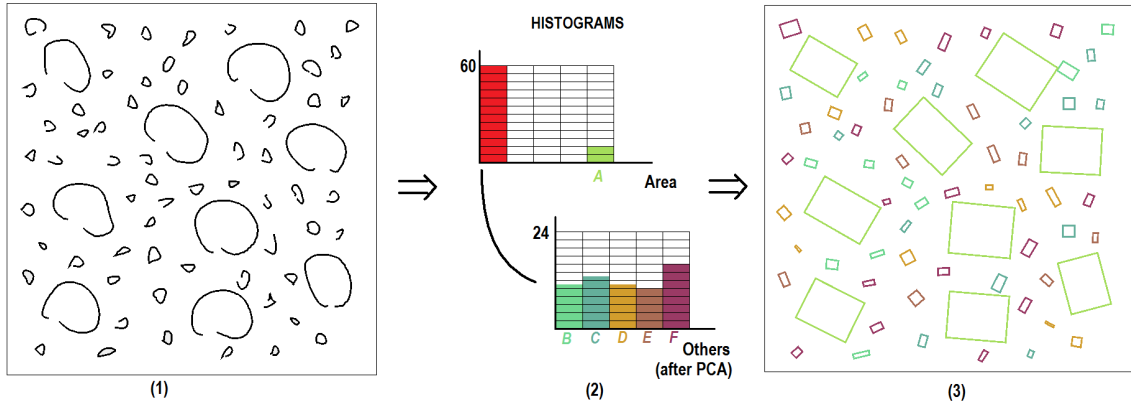


Figure 3.4 The classification pipeline. Figure (1) is the initial sample; Figure (2) shows the result of histogram segmentation. The upper histogram is the classification by area. The green column, with 9 elements, becomes a group itself and the red column, with 61 elements, is further refined. The white empty columns are ignored. The lower histogram shows the refinement, where each of the remaining characteristics are reduced to 1 dimension by PCA. Figure (3) is the resulting classified sample in this case 6 groups were computed

First, each characteristic is normalized by a value. The areas are normalized to 5% of the surface of the sample; orientations are normalized to 2π ; elongations are normalized to 3; number of extremities are normalized to 10; grayscale is already normalized to 1. We construct the 5 bins histogram with the areas of each element. If a given bin is empty, nothing is done. When the bin contains 12 elements or less, it is considered a group by itself. If the bin contains more than 12 elements, classify those elements one more time. In the second classification,

apply dimensionality reduction with PCA on the remaining characteristics, and construct the histogram of the resulting data. Again, the histogram contains 5 bins, and each non-empty bin will become a distinct group. Figure 3.4 shows the steps of classification for one sample. It was verified that the simplified procedure is enough to obtain good results not only with the samples from previous work, but also new complex ones, as will be shown later in the results.

The following table summarizes the parameters. Except for the the number of bins, all of the parameters have the same values used in [HLT⁺09] and seems to only weakly depend on the provided sample. The same values were used for new textures and good results still were obtained.

Number of bins on the histogram	5
Area normalization factor	5% of the sample
Orientation normalization factor	2π
Elongation normalization factor	3
Extremities normalization factor	10
Smallest size of group on the first step of the classification	12

3.2.4 Spatial features of the pattern

As previously mentioned, the classification of elements from the sample is made in order to obtain a unique description of the texture. Having each element labeled with a group, spatial features used to guide the synthesis during the positioning of elements onto the target space, can be evaluated. Those features acts as spatial restrictions that the new pattern must follow in order to maintain the overall appearance of the input sample. The are defined using the following functions:

$$d(x,A) = \min\{d(x,y)|y \in A, y \neq x\}$$

$$d(B,A) = \min\{d(x,A)|x \in B\}$$

where $d(x,y)$ is the euclidean distance between the center of two elements x and y , and A and B are groups. The function $d(B,A)$ is evaluated for each pair of groups and also, all the $d(x,A)$ for each element x , as illustrated in Figure 3.5. Each element x will contain its own table of features to be used later, during synthesis. More features could be added, but this would increase the computational cost, redundancy of information and make the result less controllable. In this work, these features are used to control the local density of the texture, as it will be shown later.

3.3 Synthesis

At this point, all of the necessary information for the synthesis stage was computed from the sample. From now on, we apply a procedure that will make use of this information to build the new arrangement. Also, this is where the algorithm for 2D and 3D synthesis becomes different. Initially inspired on the algorithm presented by [HLT⁺09], a similar, but simpler, procedural growth method was built. The simplicity resides in the fact that no underlying triangulation

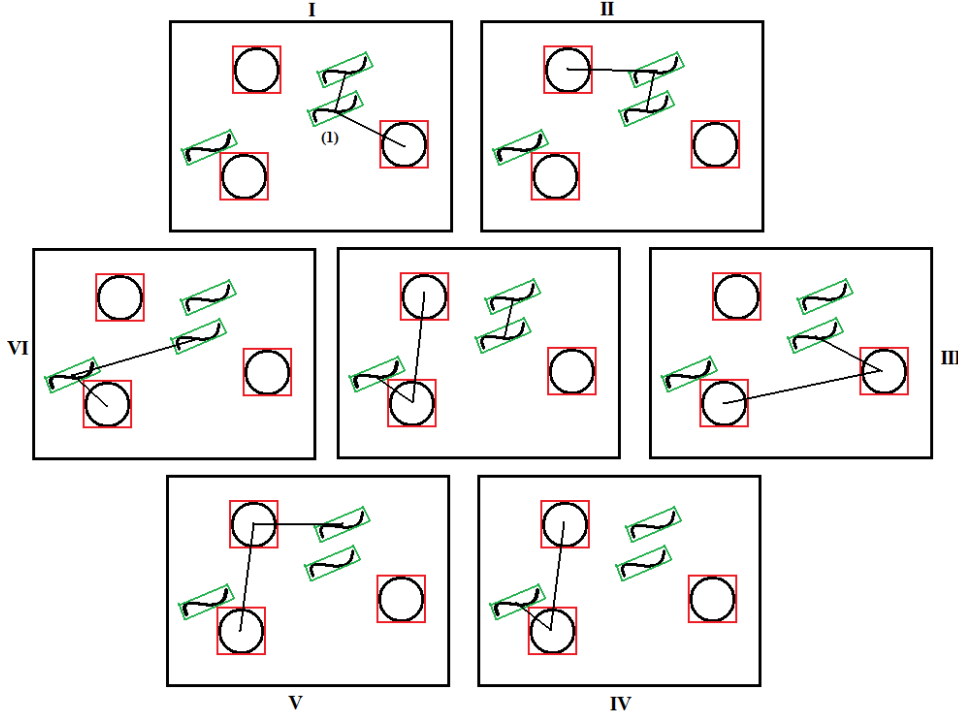


Figure 3.5 Illustrative example of feature evaluation. In this illustration, there is a texture with 6 elements, so the minimal distances $d(x, A)$ for each element (surrounding images) need to be evaluated, and also the minimal distances between groups $d(B, A)$ (central image). B and A correspond to circle and curve groups. All of the minimal distances are represented by black line segments. In image I, it is shown the minimal distances found for the curve element (1). The upper line represents the minimal distance between (1) and any other curve. The lower line represents the minimal distance between (1) and any circle. Similarly, there are the features for each curve (Figures II and VI) and for each circle (Figures III, IV, V). On the central image, the long vertical line represents the minimal distance between circles, the short vertical line is the minimal distance between curves, and the remaining line is the minimal distance between curves and circles.

is used, and therefore, no triangulation must be constantly evaluated for every new element inserted on the texture.

Although the basic structure of the algorithm remains the same for both 2D and 3D synthesis, few ideas must be adapted. First, the entire algorithm for the 2D case is explained, and then, what changes should be made in order to extend the method for the 3D case.

3.3.1 Covering The Target Space

On raster texture synthesis, the target space is defined by the resolution and the synthesis is concluded when all pixels have been visited. Based on this idea, set the target output size as a parameter, but still remains the problem of knowing when the synthesis is complete. An underlying grid is artificially created by dividing the area of the target texture into a matrix of squared regions, initially marked as unvisited, and use them as the ‘pixels’. The length of the

edge of each region is defined as the minimal distance between elements from the sample. This number can be easily found on the table of distances between groups, previously evaluated, since the number of groups is usually small. Using the minimal distance will assure that the result will have none undesired holes. When a new seed is placed inside a region, the respective region is switched to visited. The synthesis is complete when there are no remaining unvisited regions.

Following the same nomenclature from previous work, seeds are objects which will become copies of elements from the texture. In this system, those seeds may contain a reference value to one of the groups found during the classification step. This will force the seed to become an element from one specific group, unless there is no such reference.

3.3.2 Seeding and Growth

The main loop of the synthesis step can be stated as follows:

```
While (number of unvisited regions) > 0 {
    Randomly visit one of the unvisited regions;
    Try to randomly place a valid empty seed at this region;
    While (number of seeds on the list) > 0 {
        synthesizes the first seed of the list;
    }
}
```

Initially, this is what happens: a seed is placed at a random location inside any of the regions of the target space, and the texture is expanded outwards replacing seeds by copies of elements from the sample and placing new seeds around them. The element to be copied is chosen based on neighborhood comparisons, and the location of new seeds is defined from the relative position of the neighborhood of the copied element. All the seeds are stored in a FIFO list, where the head contains the seed that will be replaced at the next iteration, and new seeds are inserted at the tail of the list. The growth process is illustrated in Figure 3.6.

When a seed is being processed, the reference for the group is taken and the best matching element, which represents the element whose neighborhood is the most similar to the vicinity of the seed, is searched in the sample texture. To compare neighborhoods, it is used a metric, as explained later in Section 3.3.3, that returns a value between 0 and 1, representing the dissimilarity between two distinct neighborhoods without restrictions on the number of elements on them. The reference for the group contained on the seed is used to restrict the search into one specific group. Once the best matching is found, the seed is replaced by a copy of this element and create new seeds using the relative position of its neighboring. Each new seed will contain a reference to the group of the element that originated it.

To be accepted, each new seed s must be checked whether it will violate any of the minimal distances previously evaluated from the sample. Each new element n from the target texture contains a reference to one element from the sample. Therefore, each n can be visited and the referenced table of minimal distances evaluated during the analysis step is extracted. From s

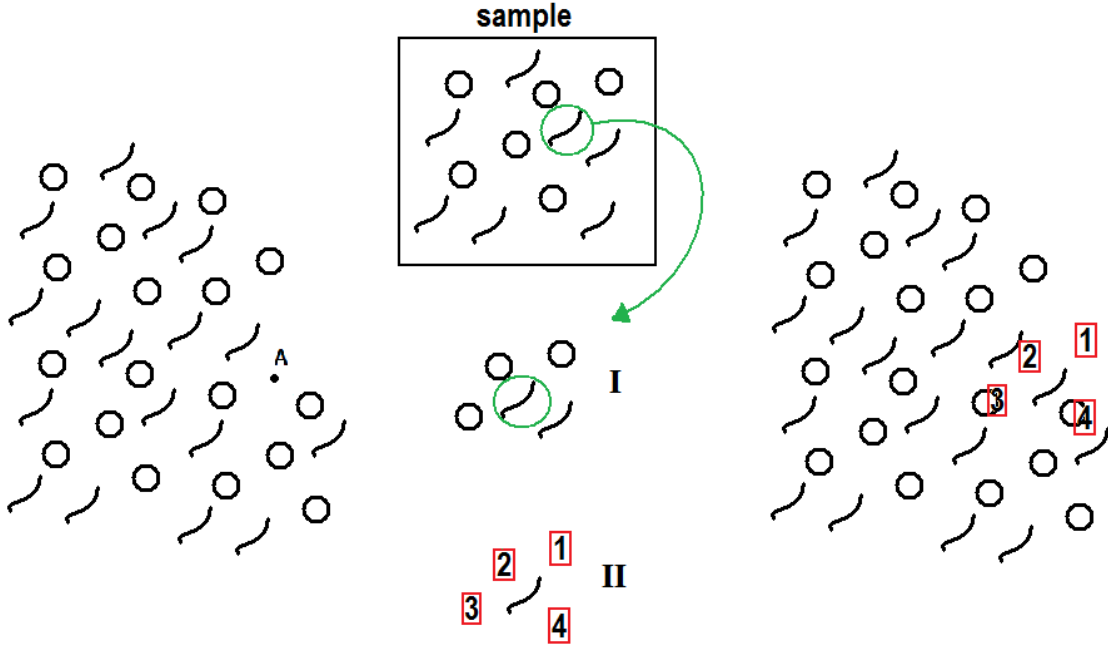


Figure 3.6 Seeding process. On the left, the current arrangement being synthesized. (A) marks the position of the seed extracted from the head of the list. The best matching element is searched throughout the sample texture. In this case, assume the seed reerences the group with curves. The best matching element was found to be the highlighted curve. The neighboring elements (I) are extracted and four new seeds are generated from them (II). This set is pasted over the seed (A), and on the right arrangement, only seed 1 will be accepted.

there is the reference for the group G and the correspondent minimal distance, $\min(G)$, on the table. The seed passes the test if only $d(n, s) > \min(G)$. Similarly, the same condition for all the already accepted seeds must be checked. However, since in this case there is not any referenced elements from the sample, a table of minimal distances between groups is used. Only when the seed passes all tests, it will be accepted. If the seed fails at least one of the tests, it is discarded.

Using this approach it is possible to control the density of elements on the result. Instead of directly using the minimal distances on the validity tests, they are multiplied by a density parameter. When this parameter is less than 1, closer pairs of elements will be accepted on the final distribution, and therefore, a denser texture will be created. When this parameter is greater than 1, a coarser texture will be created. Figure 3.7 shows the effect obtained by varying this parameter.

3.3.3 Distance Metric For Neighborhoods

It was shown how the texture grows, replacing seeds by elements from the sample. In this subsection, it is explained the metric to compare two distinct neighborhoods. This metric will be used to choose the best element from the texture to replace the seed.

In raster texture synthesis usually an L_2 norm is computed for pixels inside a given regular

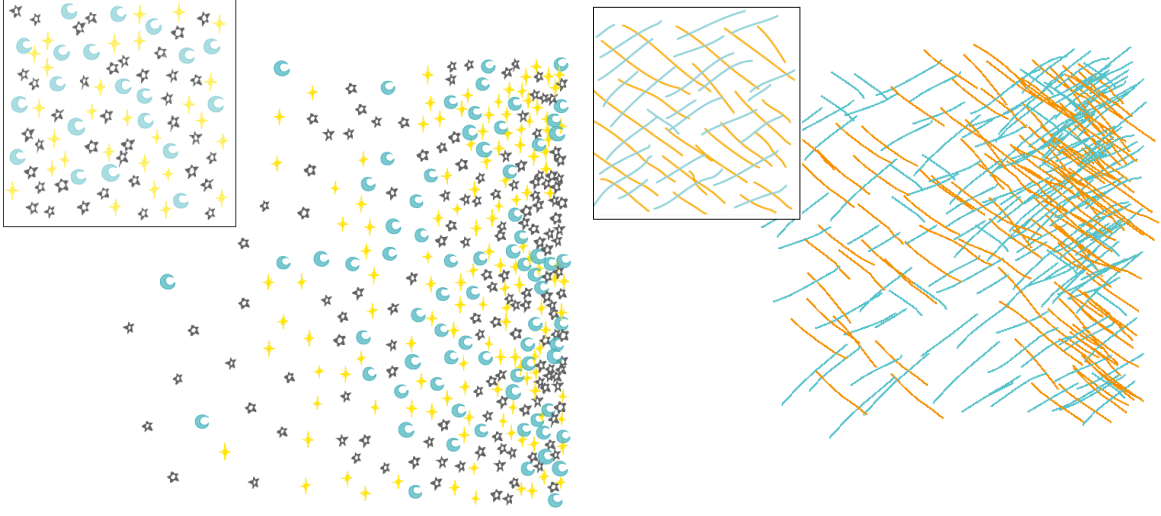


Figure 3.7 Varying density along the horizontal axis. Sample and result.

neighborhood. There is also the notion of neighborhood, but instead of a rectangular one as used in raster textures, a circle around an element is defined with a user-defined visibility radius. A circular neighborhood provides a fairer region for computing the metric. When using triangulations, the neighborhood is limited by the n -ring vicinity around the central vertex as done in previous work by [IMIM08], which can be globally too small or too big depending on the texture. Using a circular neighborhood, free of triangulation, every element has the same visibility. Since the system does not deal with pixels, but elements, some difficulties arise: the metric should be able to compare two neighborhoods with different number of elements and should consider the overall distribution of elements inside the neighborhood.

The range of visibility is a parameter which controls the balance between quality and computational cost. If the parameter is too small, each element will be able to see few of its neighbors. Therefore, the overall characteristics of the textures will not be properly captured during synthesis. In order to improve the quality of results, a larger range must be chosen, but this might rapidly increase the computational cost due to the increase on the number of neighbors that an element can see.

Since the sample input is static, the neighbors of each element can be previously calculated on the analysis step. However, as the target texture grows, new elements are added and the neighborhood of some elements may change. Therefore, the neighbors of each element are dynamically updated as the synthesis progress. The sampling procedure to compute the distance metric works as follows: extracts the first seed from the list and find the neighbors inside the visibility window. From this collection, evaluates the relative position of each neighbor, considering the seed as the origin. For any candidate extracted from the sample, its neighborhood is similarly evaluated.

The proposed metric is normalized to a range $[0, 1]$ and gives the dissimilarity between two neighborhoods I and J as follows. For each relative location i of the elements on I , uses the smaller Euclidean distance between i and j , where j is the relative location of the elements on

J. When the element at *j* is from a different group of the element at *i*, it is ignored. If the end of the list of elements from *J* is reached and no distance was evaluated, then consider the element *i* as too far and set the value to 1. Otherwise, normalize the value found by the diameter of the visibility window. Once all those values are evaluated for the elements of *I*, similarly evaluate the distances for the elements on *J*. The final overall dissimilarity will be the average of all those values. A value of 1 represents a completely different neighborhood, while a value of 0 represents an exactly equal neighborhood. This metric is illustrated in Figure 3.8. Special care is needed for the neighborhoods near the edges that are often incomplete. To avoid the edges, consider only those elements from the sample whose visibility range does not cross the sample boundaries.

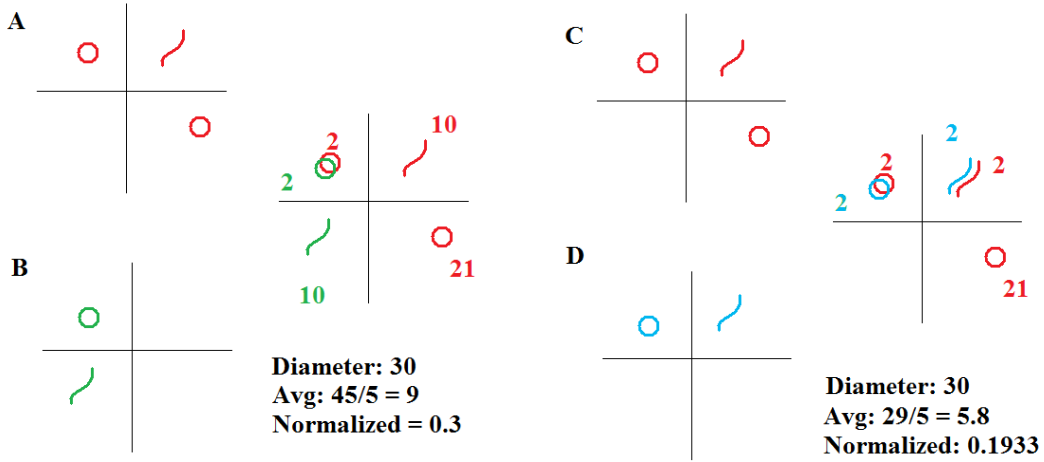


Figure 3.8 Distance metric for neighborhoods, shown in an artificial example. We have the comparison between two pairs of neighborhoods, AB and CD. Neighborhoods A and C are exactly the same, while B and D differ only on the position of one curve. Neighborhoods are illustrated together in the same frame, with the smallest distances evaluated for each element (before normalization). Only the displacement of one of the elements, bringing it closer to another element from the other neighborhood, makes both neighborhoods more similar, and this effect is easily detected by the final value of the metric. A lower value denotes a more similar pair of neighborhoods.

In Barla [BBT⁺06], two neighborhoods are intersected and only the paired elements are considered. If another neighborhood contains the same distribution for the paired elements, but different distribution for unpaired elements or even a different number of them, the metric would return the same value. The main difference from this metric is that none of the neighbors are discarded and the resulting value is sensitive to small variations on the distribution.

3.3.4 Choosing The Best Matching

In order to pick the final best element, the same approach from raster texture synthesis is used, building an ordered list with the best matching elements and using the corresponding dissimilarity value to order the list. Then, the best element is randomly chosen from those whose dissimilarity factor is smaller than 1.1 times the smallest factor obtained. In order to keep the

system from falling on a local minimum, always choose the same sub-group of elements, for each element, count how many times it was already chosen and use this value to modify the dissimilarity factor. The new factor is $\log(1 + n) + d$ where n is the number of times the element was chosen and d is the regular dissimilarity. This new factor will be used to order the list, instead of the unmodified dissimilarity. The \log function is used to avoid the quick growth of the factor, and use the term $(1 + n)$ to avoid singularities when the element was never chosen, in other words, when $n = 0$. The effect of this factor is shown in Figure 3.9.

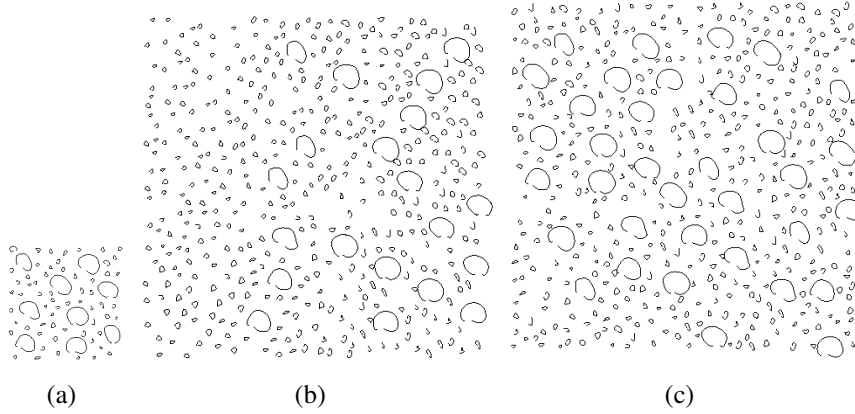


Figure 3.9 Illustration of the influence of the modified dissimilarity factor. Figure (a) is the sample. In figure (b) the original factor is used. At the upper right part of the figure, the repetition of the big circles is perceived. The left portion also was not correctly synthesized. On (c) there is a new result, using the modified factor, providing a result closer to the sample.

3.3.5 Patches

The same idea of patches of texture from raster synthesis was used to improve the efficiency of the proposed method. Instead of copying only one element from the sample, a cluster of elements will be copied. A parameter – patch size – which will split the visibility region into two parts, inner and outer, is defined. The inner part will represent the patch region, where, instead of seeds, the elements will be copied directly onto the target. The same check for violation of features is performed before any of those elements are accepted but they do not generate new seeds. In Figure 3.10 the influence of this parameter varying between 0 and 1 times the range of visibility is illustrated. This possibility greatly improves the computational cost, and allows similar quality in the visual results.

3.4 Tridimensional Synthesis

In this subsection, all the changes applied to the basic structure of the synthesis algorithm already presented, are explained in order to render patterns on arbitrary 3D meshes. The analysis step remains the same, because the main task of the method remains the same: to synthesize a new texture, similar to a supplied sample. The difference is that the synthesis is made over a

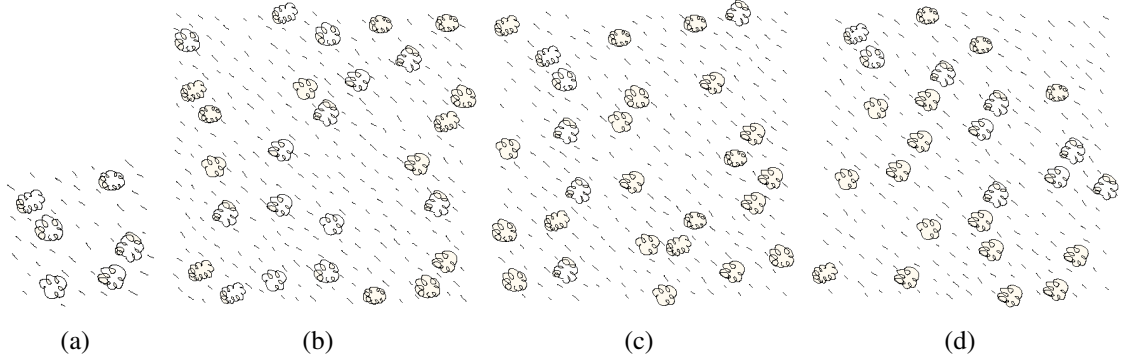


Figure 3.10 Given a sample (a) and varying the patch size 0%, 50% and 100%, (b, c, d), respectively. Notice the same quality in the visual result.

surface defined in the tridimensional space, instead of on the plane, as done so far.

There are basically two possibilities for texturing objects: directly on the surface, or over the 2D parametrization space. In this work, the first class of methods is used. The synthesized elements are placed and oriented directly on the tridimensional space of the surface of the mesh.

3.4.1 Input Information and Parameters

The input is an object defined as a polygonal mesh. In this case, no parameterization is necessary; the synthesis will be performed directly over the surface. Also, an orientation field must be defined in the surface. This orientation field gives the local orientation of the texture in the surface. This field can be obtained by several ways, but in this case, projecting the vertical global direction in the surface is enough. This projected vector, and the normal of the triangle, will define the local coordinate system for each triangle.

Besides the visibility range and the patch size, a scale parameter that defines the proportion of the area of the surface with respect to the area of the sample texture, is needed. This scale parameter will be used every time it is necessary to make geometric transformations between the 3D space of the object and the 2D space of the sample.

3.4.2 Covering The Target Space

The target space is a general and free of parametrization 3D triangulation. A regular grid can not always be constructed for such surfaces, so another way to divide the target space is necessary. For this, the mesh data itself is used. Each triangle will represent a visitable region. In the 2D case, a square as small as necessary is constructed to guarantee that every location will be visited and undesired roles will not be created, as explained in Section 3.3.1. With the mesh, the size of the triangles is already determined. If the triangles of some neighborhood are too small, this neighborhood will be visited an excessive number of times. This will cause a high computational cost, but the visual result will not be affected. When the triangles are too big, they could be divided into smaller triangles to avoid the creation of undesired roles, but it was observed that in practice, this is not a problem. When the texture grows, the space is filled

incrementally.

3.4.3 Obtaining The Neighborhood

During synthesis, neighborhoods are constantly being obtained from the target space. They must be compared with the neighborhoods from the sample texture, which are on the 2D space. Therefore, it was chosen to map the 3D neighborhood of an element to the 2D space, and use the same metric discussed previously to measure dissimilarities. The mapping process is simple. Given the position of the central element and the triangle that contains it, it can be obtained from the triangle the local coordinate system defined for the position. Then, orthographically project any of the neighboring elements that lie inside the visibility range, using the normal direction of the triangle to project. The projection is done over the plane of the triangle. Then, find the (x,y) coordinates of the projection, using the local coordinates and the central element as the origin. These coordinates are multiplied by the scale factor, before measuring the dissimilarity from any neighborhood from the sample. Computing exact distances over the surface would be prohibitively expensive and would not affect the visual result significantly for weakly regular textures. Figure 3.11 illustrates the process.

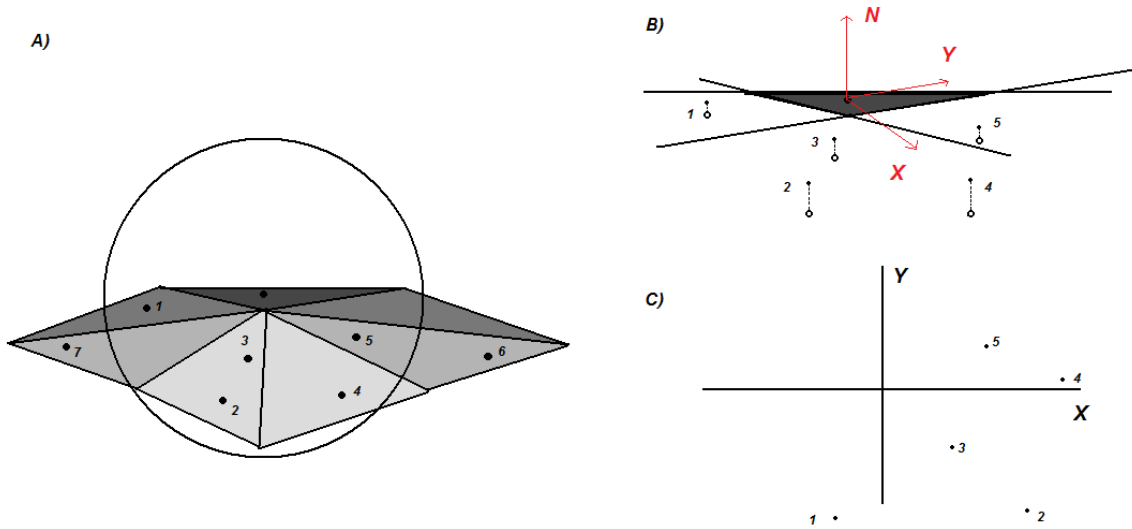


Figure 3.11 Process to obtain the neighborhood of an element on the 3D space. In A) triangles from the mesh, with some elements over them. The circle represents the visibility sphere centered at the element whose neighborhood we are interested on. Elements 6 and 7 lies outside the sphere. On illustration B) it is shown the orthographic projection of elements 1 to 5 onto the plane of the triangle that contains the central element. The projection follows the normal direction N . The direction Y is obtained from the vector field defined for the surface, and the direction X is the cross product of Y and N . On illustration C), the final result after evaluating the 2D coordinates of each projected element.

3.4.4 Generating New Seeds

When the best matching element is found, new seeds are created from its neighborhood. In the 2D synthesis, the orientation and scale of the target space is the same as the sample, so placing elements is straightforward. In the 3D synthesis, however, the relative neighborhood must be appropriately placed over the surface. This is accomplished by doing the reverse procedure explained on the previous subsection. First, multiply the X and Y values by the inverse of the scale factor and then align the axis with the local coordinate frame of the triangle that contains the central element. Then, use the normal direction to orthographically project the points over the surface. The resulting point is the final location of the new seed. A smooth surface is assumed, but when there are highly curved regions and the projection of the points falls off the surface, the new seed is ignored. These new seeds, as in the 2D case, cannot violate spatial restrictions and must be checked against all the already accepted elements. To simplify the method, just use the 3D Euclidean distance between two points, instead of finding the smallest path over the surface. Also, to increase speed, during the projection of points over the surface, consider only those triangles whose center lies inside the visibility range of the central element. Plausible visual results are obtained, despite these simplifications.

3.4.5 Rendering

Since parametrization or texturing of surfaces is not used in this work, the elements are generated as 3D geometric figures that are rendered together with the surface. The geometric data of the elements are control points of Bezier curves, that must be scaled and oriented to align with the local coordinate system. After that, several line segments are constructed, following the curves. The segments that are located outside of the triangle, are orthographically projected again over the surface. Segments that cross edges between two triangles, are splitted. All this process is done to avoid popup issues, such as the illustrated on the Figure 3.12.



Figure 3.12 Result with popup issues. When the scale of the element is small compared with the local curvature of the surface this problem is minimized, but at the borders this artifact can still be clearly seen.

Experiments and Results

In this chapter a full range of results synthesized using the proposed solution and results for texturing 3D surfaces, are presented. The research on synthesis of arrangement patterns has significantly improved the quality of visual results. New methods and approaches were proposed, and the range of possibilities increased with each new method. It is performed a qualitative evaluation of the results by visual comparison with previous results, using the same samples as input.

Subjective visual assessment is still the main method for validation of results. It was not performed any quantitative measurements between methods mainly for two reasons: there is no accepted good metric to compare similar textures on the field of texture synthesis, and each proposed method applies different approaches for solution and therefore it is not possible to compare tables of parameters since the parameters are not the same.

In general, computational cost varies from real time to one minute for the presented results. Since the algorithm performs a near brute-force search for the best matching neighborhood, and also each neighborhood comparison will depend on the number of elements inside each neighborhood, the computational cost strongly depends on the sample. Usually, small and scarce textures are used, allowing a real time synthesis. Considering such textures and that the use of specialized hardware is not required, vectorial texture synthesis is preferable. Also, the computational cost is comparable to previous work.

4.1 2D Synthesis

The first important difference between ours and previous methods is the inclusion of color. Due to the conversion from RGB to grayscale, the color feature could be introduced on the synthesis step without greatly increasing the computational cost of the PCA step. Also, vectorial textures are usually simple, compared with raster textures, and simply introducing the color feature on the second stage of element grouping is enough to provide good results, as it is seen on Figure 4.1.

On the same figure, it is seen a result for a regular texture of chessboard. In this texture, the color is the only feature that differentiates one element from another. The statistical method of [HLT⁺09] cannot produce regular results, and [BBT⁺06] has some issues to produce irregular textures. On these examples, it is seen that the proposed method delivers plausible visual results for both type of textures.

Figures 4.2 and 4.3, compare our results for various samples from the current state-of-the-art in this topic [HLT⁺09]. It can be seen that, apart from minor visual differences such as the

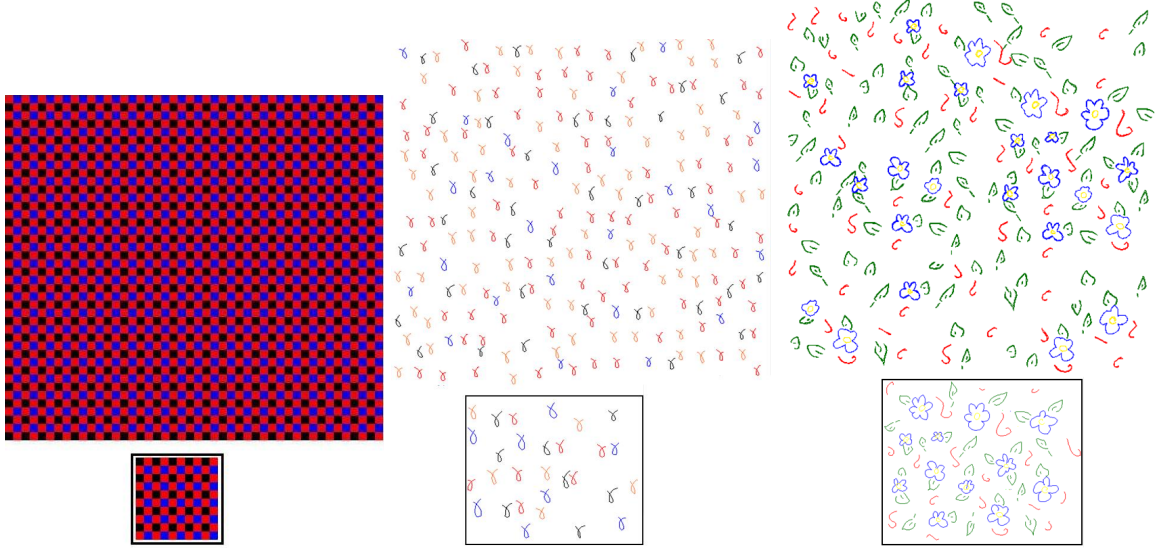


Figure 4.1 Synthesized colored arrangements.

width of lines or clarity of colors (which has no relation with the synthesis), our arrangements are qualitatively similar. Original results were provided by the authors, and our results were visualized directly from the system.

Figure 4.5 shows results obtained by the proposed method for hand-drawn geological patterns. Notice the continuous variation of the orientation value of the elements on the samples. This presents issues due to the use of histogram, since the inclination value changes in a circular fashion, where elements with 2π value have the same inclination of 0 valued elements. Also, the lack of strong similarity between elements on pattern 4.5(a) implies the necessity of a large number of groups. As shows the Figure 4.4, with our implementation of the method used in previous work, such patterns were not well classified, while with the simple 5 bin histogram method, better results were obtained. Over-categorization does not represent a big problem, but under-categorization will affect negatively the results, specially in such patterns with compacted long thin elements and no intersections between them.

The next set of results (Figures 4.6, 4.7, and 4.8) illustrates the flexibility of the proposed system, by comparing the original sample with results generated with the same resolution. Notice that although they are slightly different, they all retain the same overall visual appearance. Since a random factor is used in the algorithm, a new pattern is generated every time.

4.2 3D Synthesis

This section shows in Figure 4.10 some results obtained by the proposed system. For these results, the illumination intensity was used to determine the density parameter, resulting in a shading effect. Bright triangles are associated with a small value of density, and dark triangles with a larger density. Although good results can be obtained, the computational cost of the method can increase rapidly if used for rendering NPR images with the most usual styles, such

as hatching or stipples. Figure 4.9 shows the influence of parameters for the synthesis.

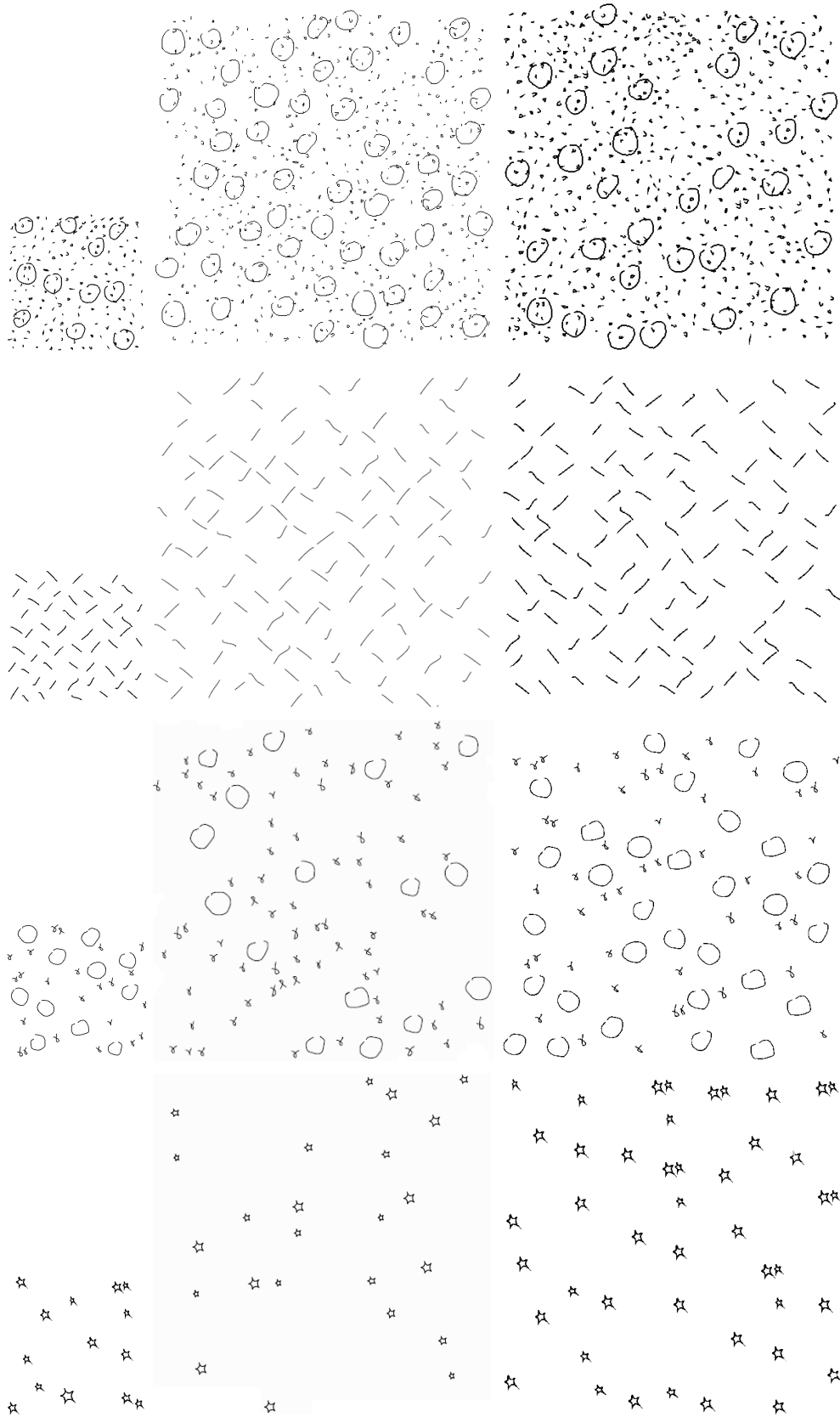


Figure 4.2 Texture synthesis results. From left to right: sample (smaller square), result by Hurtut [HLT⁺09] and ours.

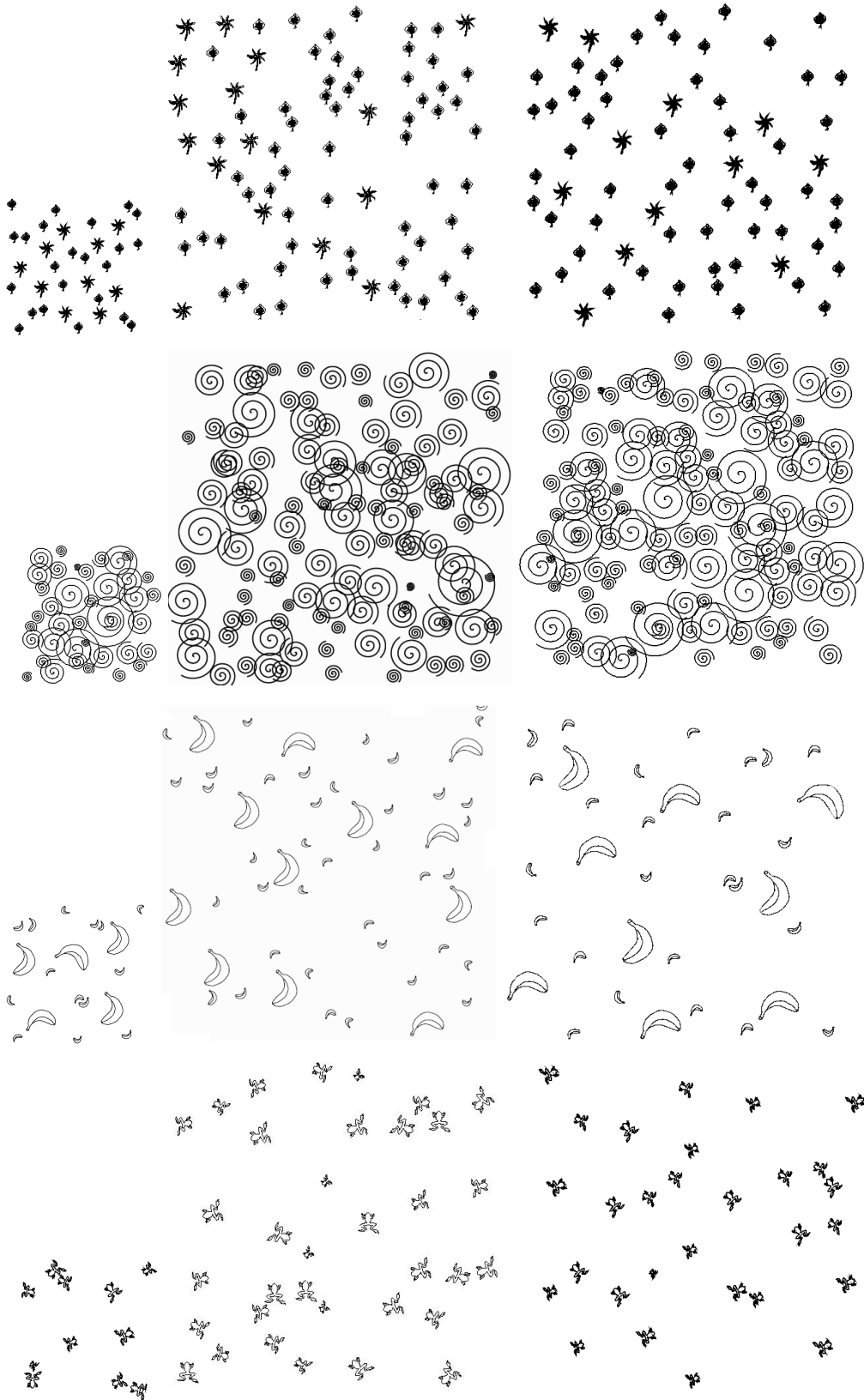


Figure 4.3 Texture synthesis results. From left to right: sample (smaller square), results by Hurtut [HLT⁺09] and results of the proposed method.

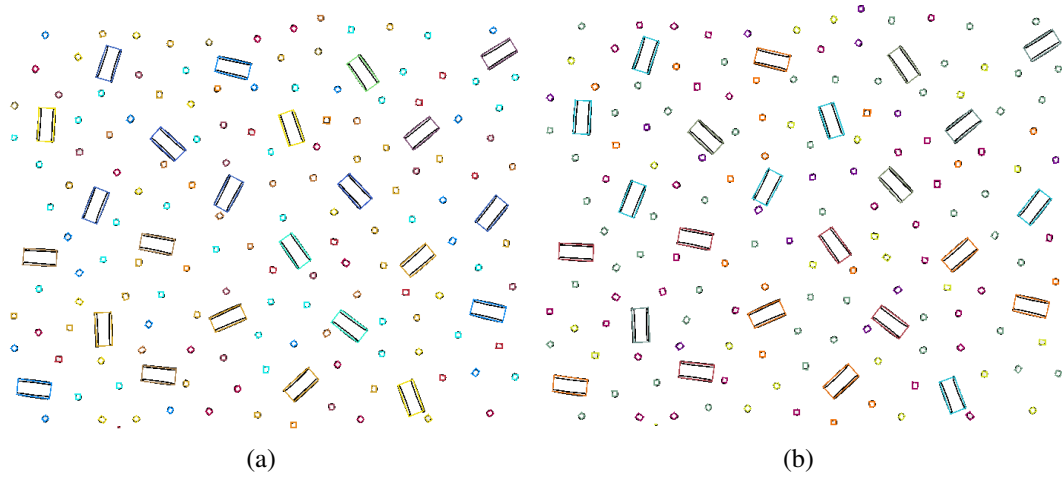


Figure 4.4 Results from the analysis step. Figure (a) shows the result of the implementation of the method used on previous work. Figure (b) shows the results of the simplified method. In Figure (a) can be seen elements with highly distinct orientation classified on the same group (purple), and yet some small circles classified on the same group of the big elements (light blue, yellow). In Figure (b), such mistakes does not occur.

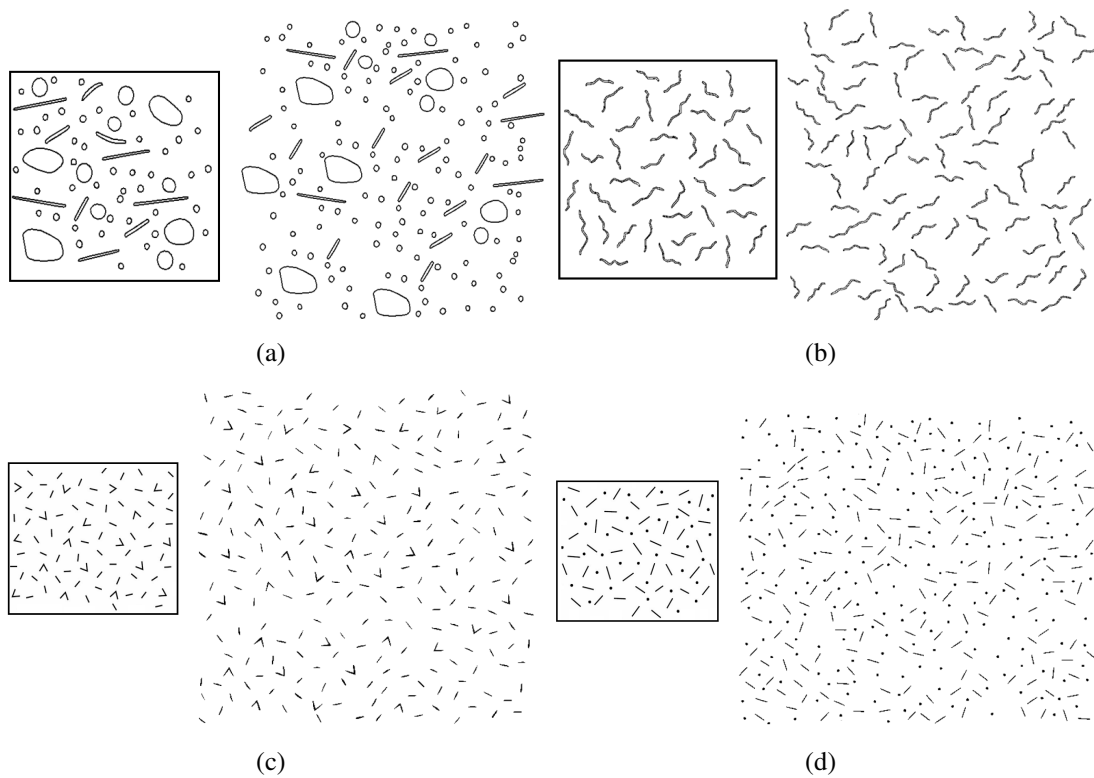


Figure 4.5 Texture synthesis results for illustrative geological patterns.

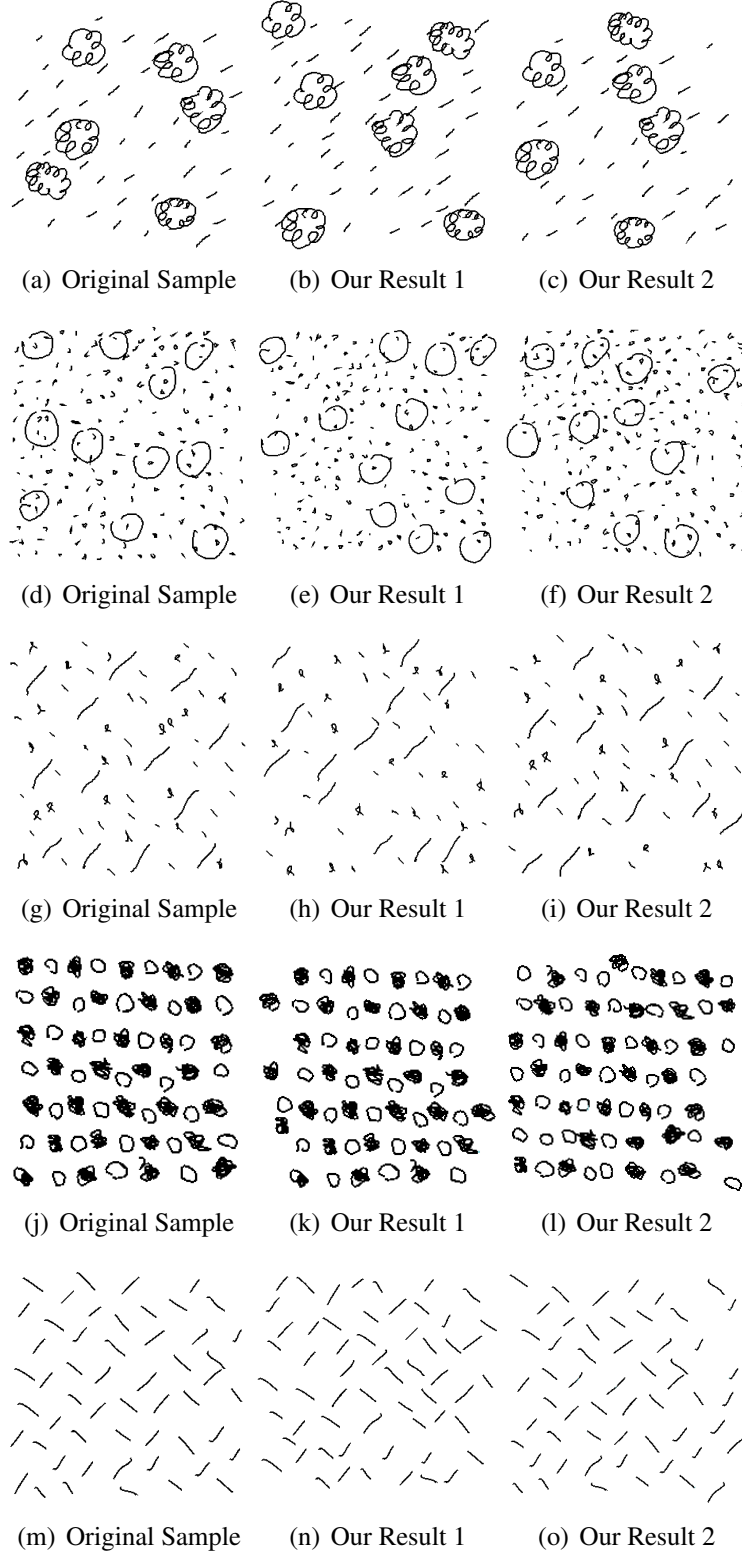


Figure 4.6 Comparison among the results from [HLT⁺09] and the proposed method

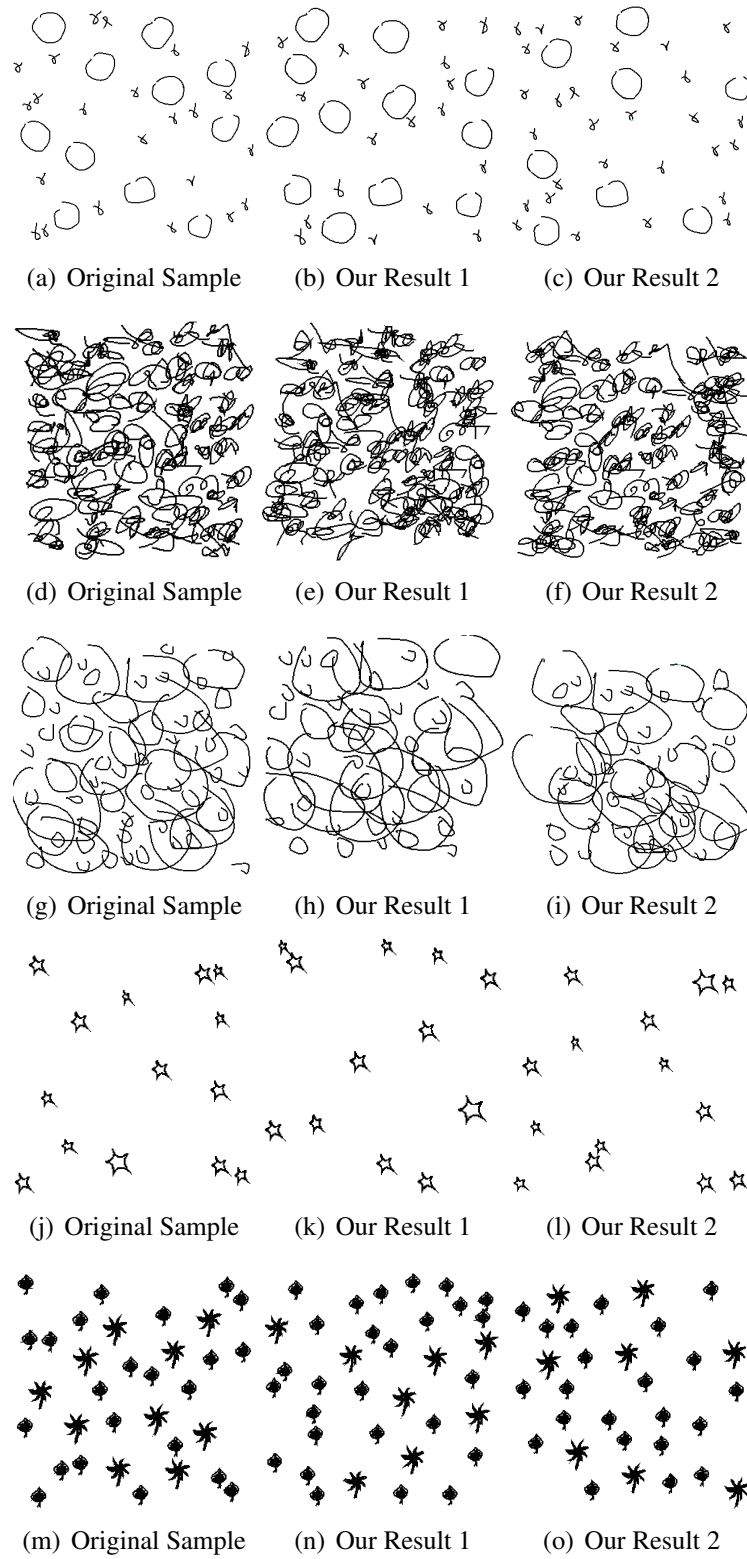


Figure 4.7 Comparison among the results from [HLT⁺09] and the proposed method

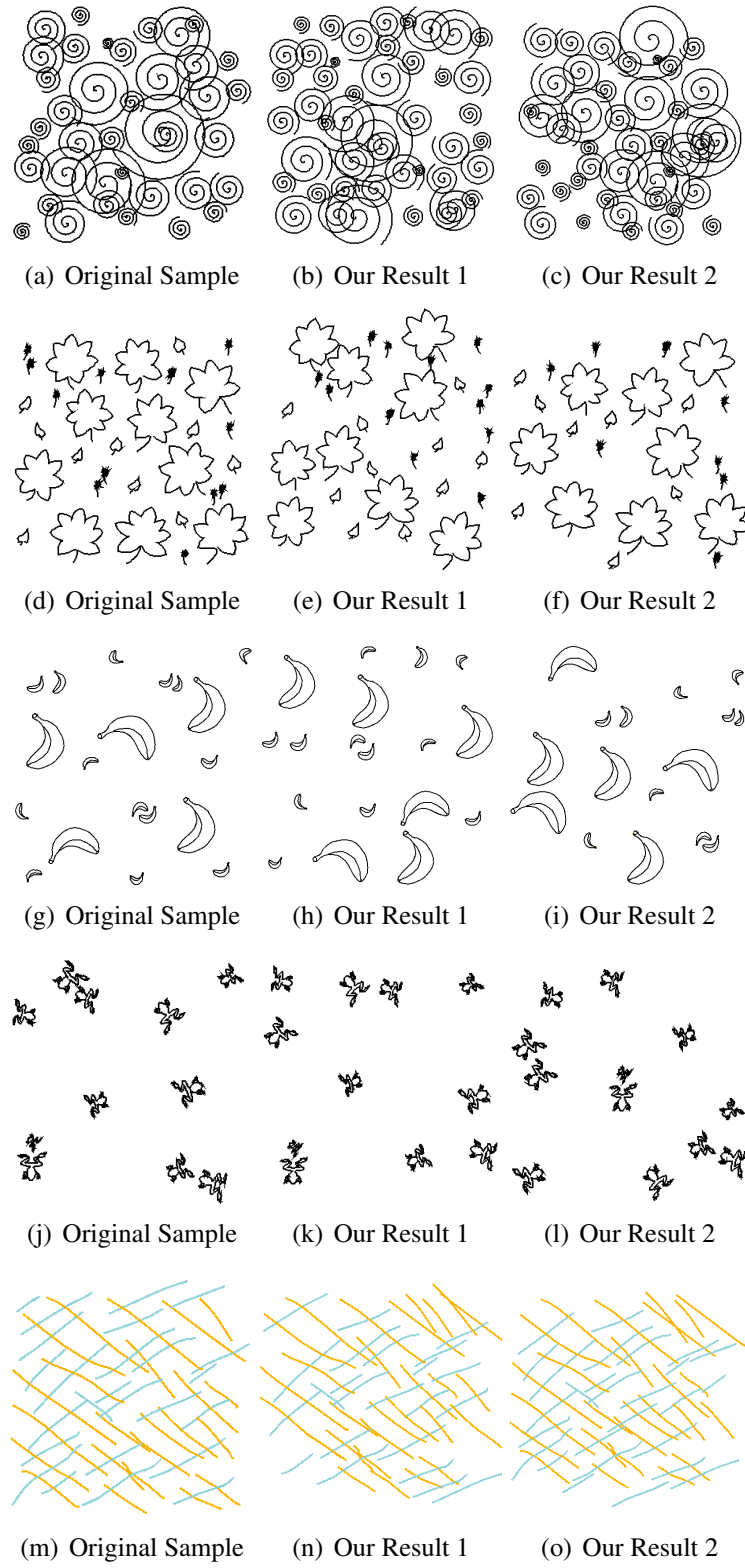


Figure 4.8 Comparison among the results from [HLT⁺09] and the proposed method

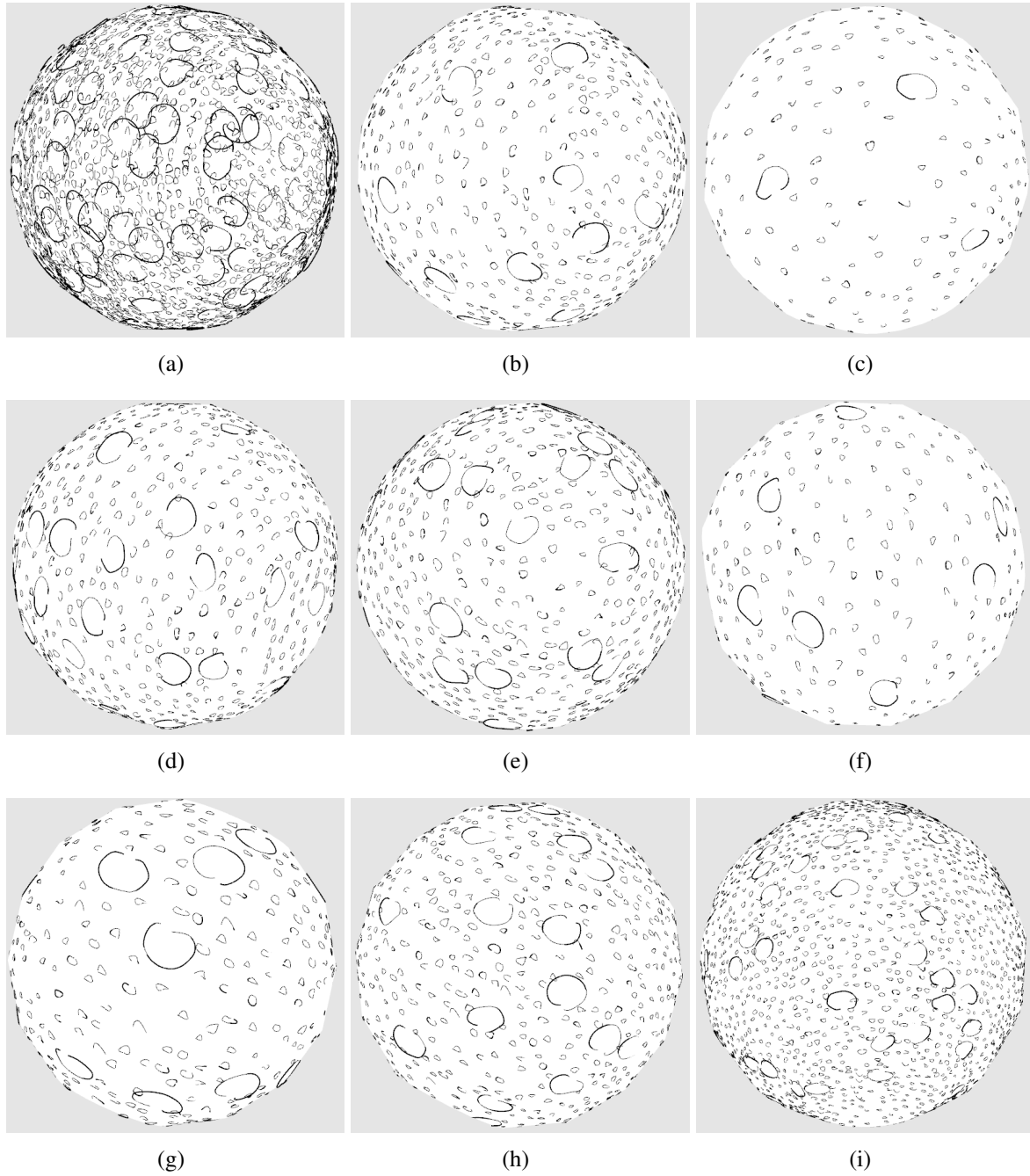


Figure 4.9 Influence of parameters on the 3d synthesis. Figures (a), (b) and (c) show the variation of density, from high to low. Figures (d), (e) and (f) show the variation of path size from none, half and full visibility range, respectively. Figures (g), (h) and (i) show the variation of the scale, achieved by changing the ratio between the surface area and the sample area.

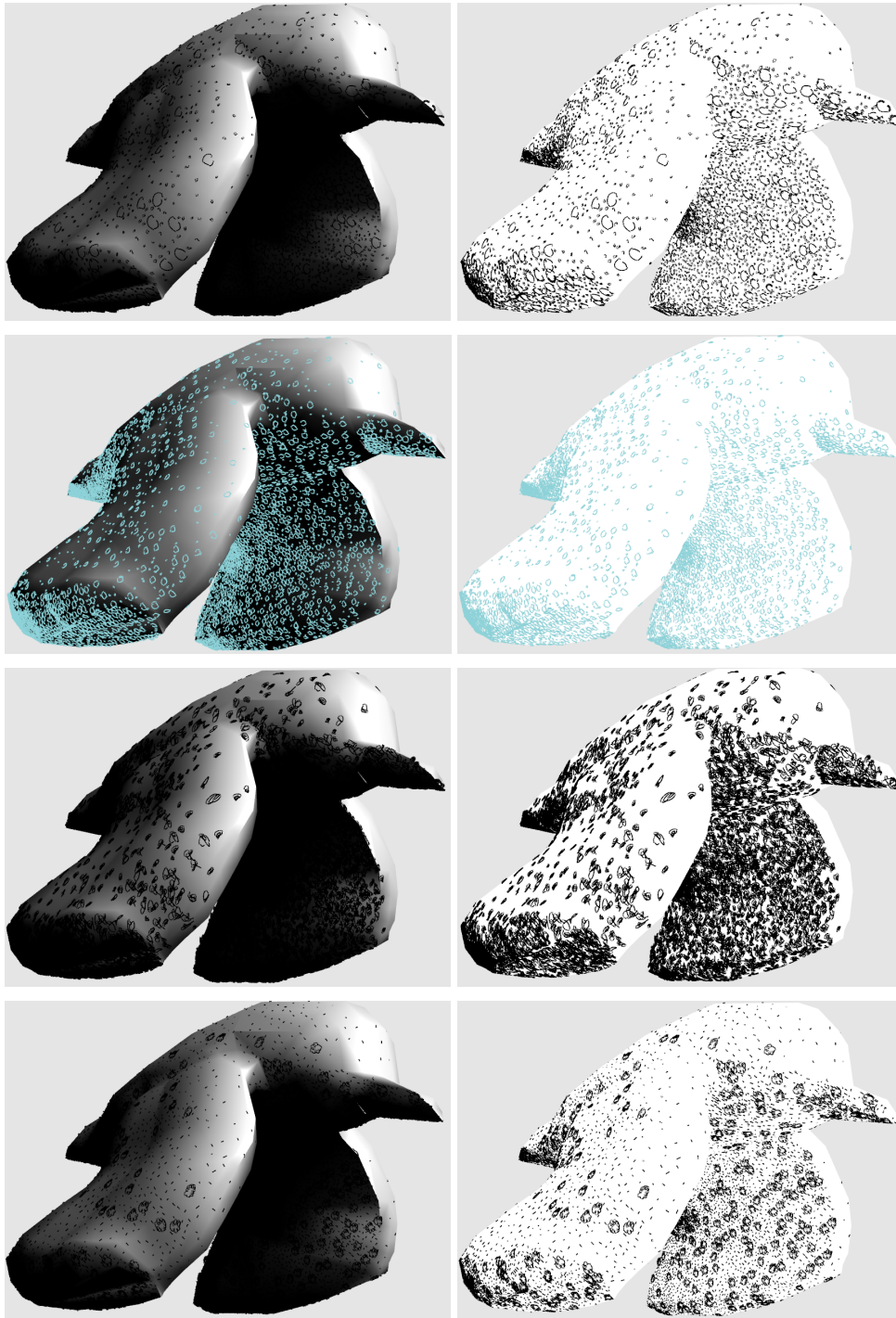


Figure 4.10 Tridimensional texture synthesis results using four different textures. On the left, the model is rendered with Gouraud shading model to illustrate the illumination over the triangles; on the right, no illumination model was applied. The concentration of elements by itself produces the shadowing effect.

Conclusions

It was presented a method to synthesize patterns of arrangement of vectorial elements from a small supplied sample. The method follows previous ideas, [BBT⁺06, IMIM08, HLT⁺09], where the synthesis is made in two steps: analysis and synthesis. Our method improves the current state-of-the-art by introducing color, which is easily implementable and applicable to all types of textures, from regular to stochastic. Also, the algorithm allows the control of density of elements on the distribution, and can be extended to be able to perform synthesis in 3D meshes.

We have noticed that vectorial textures, that are reducible to distribution patterns, are rather simple and visual acknowledgement of failures on synthesized images can be difficult. This is related to how human mentally define a texture. It is also related to what characteristics are evident in a sample and should be preserved when expanding that sample. Therefore, a simpler method, such as proposed in this work, is enough to produce good results. A small subset of constraints, such as minimal distances, is capable of capturing the essential information within a distribution pattern. Such characteristic allows a fast and easy implementation of a purely procedural synthesis system.

5.1 Limitations

Although the proposed method does not require a triangulation of the distribution, and therefore the size of the neighborhoods is constant and defined by one parameter, many tests need to be performed to see if an element lies inside the visibility region of some given neighborhood. Also, during the evaluation of the dissimilarity metric, for each element, the closest elements is searched in other neighborhoods. Usually, the number of elements in samples and the size of the output texture is relatively small, such that all computation is done from real time up to 1 minute for the presented results. However, the computational cost increases with the number of elements in the image and also with the average number of elements inside the neighborhoods. The processing time may easily take longer than one minute. This computational cost may increases even more when performing 3D synthesis due to the constant evaluation of projections. Some of the results where obtained within 5 minutes.

The analysis step presents another limitation. In [BBT⁺06], the elements are constructed from the collection of strokes scattered over the plane. This limited the system to synthesizing only few types of well behaved textures. Subsequent works considered the elements already formed, and focused on the task of grouping them according to similarity features. Those features are characteristics of the element itself and are independent of the neighboring elements.

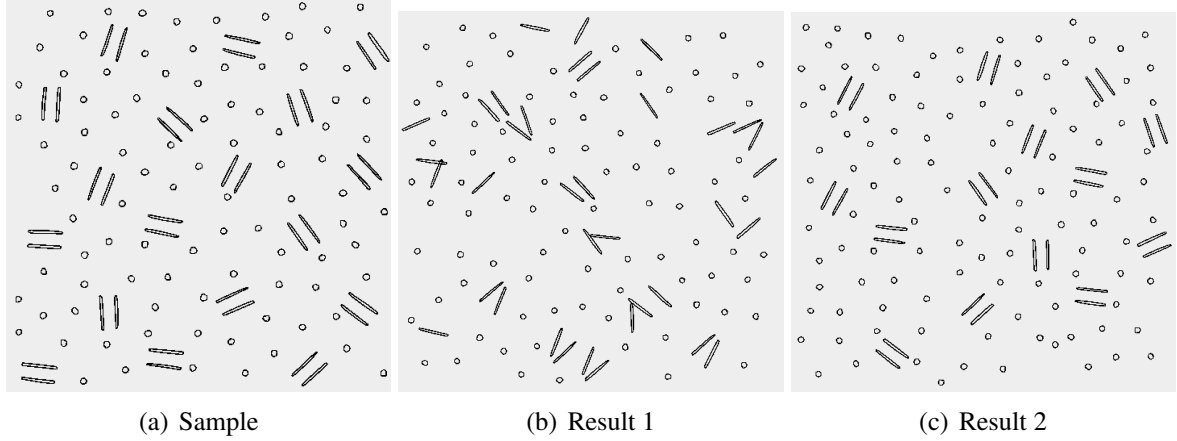


Figure 5.1 Inter-relations between elements not well captured by the method for near-regular textures.

We have noticed that for near-regular textures, the overall characteristic of the pattern must be captured during the grouping stage, but this is not done yet.

In Figure 5.1 we clearly see this limitation. Two textures were synthesized, for two distinct cases of the same sample pattern. First, in Result 1, the sample texture is provided with two types of elements: bars and circles. The synthesis, however, does not capture well the characteristic of the texture, that the bars are always paired and each pair is distinct due to the inclination. It can be seen in the result some unpaired bars. Also, during the grouping stage, the number of groups is not big enough to classify each bar correctly, since they slightly differ in the inclination value. Bars with distinct inclination values are classified in the same group. Also, during synthesis, it may happen that for a pair of bars, one of them is changed by another, resulting in pairs of nonparallel bars. To solve these problems, we modified the sample by considering each pair of bars as one single element, editing manually the file. In Result 2, there is the result after this change. A better analysis algorithm would grasp the regularity of the texture, even if each bar is coded on the sample as a single element.

A limitation in 3D synthesis is due to the nature of the method itself. Since the synthesis is performed directly on the surface, the use of geometric primitives to represent the elements of the texture may not be adequate to render the desired image. Too many line segments will be created if the mesh is detailed with several small triangles. The rendering cost may be too large, compared to texturized objects.

5.2 Future Work

Due to the limitations stated, to perform the synthesis in the parametric space instead of directly over the surface may be a better solution since the method is originally 2D. New metrics of neighborhood comparison could also improve the efficiency since our metric constantly evaluates minimal distances. Quantitative metrics to measure the similarity between vectorial textures with different resolutions is also desirable.

Bibliography

- [AJG⁺91] Etemadi A., Schmidt J.P., Matas G., Illingworth J., and Kittler J. Low-level grouping of straight line segments. In *Proc. Briths Machine Vision Conf.*, pages 119–126. Springer-Verlag, 1991.
- [Ash01] Michael Ashikhmin. Synthesizing natural textures. In *In ACM Symposium on Interactive 3D Graphics*, pages 217–226, 2001.
- [BBT⁺06] Pascal Barla, Simon Breslav, Joëlle Thollot, François X. Sillion, and Lee Markosian. Stroke pattern analysis and synthesis. *EUROGRAPHICS2006*, 25(3):663–671, September 2006.
- [CSDH03] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *ACM Trans. Graph.*, 22(3):287–294, 2003.
- [DHvOS00] Oliver Deussen, Stefa Hiller, Cornelius W. A. M. van Overveld, and Thomas Strothotte. Floating points: A method for computing stipple drawings. *Comput. Graph. Forum*, 19(3), 2000.
- [DMM03] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. A grouping principle and four applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(4):508–513, 2003.
- [EF01] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, New York, NY, USA, 2001. ACM.
- [EL99] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *In International Conference on Computer Vision*, pages 1033–1038, 1999.
- [Fe01] Jon Ferraiolo and ed. Scalable vector graphics (svg) 1.0 specification, 2001.
- [FTP03] William T. Freeman, Joshua B. Tenenbaum, and Egon C. Pasztor. Learning style translation for the lines of a drawing. *ACM Trans. Graph.*, 22(1):33–46, 2003.
- [HB95] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer*

- graphics and interactive techniques*, pages 229–238, New York, NY, USA, 1995. ACM.
- [Her98] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, New York, NY, USA, 1998. ACM.
- [Her03] Aaron Hertzmann. A survey of stroke-based rendering. *IEEE Computer Graphics and Applications*, 23(4):70–81, 2003.
- [HLT⁺09] T. Hurtut, P.-E. Landes, J. Thollot, Y. Gousseau, R. Drouillhet, and J.-F. Coeurjolly. Appearance-guided synthesis of element arrangements by example. In *NPAR '09: Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pages 51–60, New York, NY, USA, 2009. ACM.
- [HOCS02] Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M. Seitz. Curve analogies. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 233–246, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
- [IMIM08] Takashi Ijiri, Radomir Mech, Takeo Igarashi, and Gavin S. P. Miller. An example-based procedural system for element arrangement. *Computer Graphics Forum*, 27(2):429–436, April 2008.
- [JEGPO02] Pierre-Marc Jodoin, Emric Epstein, Martin Granger-Piché, and Victor Ostromoukhov. Hatching by example: a statistical approach. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 29–36, New York, NY, USA, 2002. ACM.
- [Jul86] B Julesz. Texton gradients: The texton theory revisited. *Biological Cybernetics*, 54(4-5):245–251, aug 1986.
- [KMM⁺02] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. Wysiwyg npr: drawing strokes directly on 3d models. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 755–762, New York, NY, USA, 2002. ACM.
- [KOF08] Giovane R. Kuhn, Manuel M. Oliveira, and Leandro A. F. Fernandes. An improved contrast enhancing approach for color-to-grayscale mappings. *Vis. Comput.*, 24(7):505–514, 2008.
- [KSE⁺03] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, July 2003.

- [LH05] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Trans. Graph.*, 24(3):777–786, 2005.
- [LH06] Sylvain Lefebvre and Hugues Hoppe. Appearance-space texture synthesis. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 541–548, New York, NY, USA, 2006. ACM.
- [LHW⁺04] Wen-Chieh Lin, James Hays, Chenyu Wu, Vivek Kwatra, and Yanxi Liu. A comparison study of four texture synthesis algorithms on near-regular textures. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 16, New York, NY, USA, 2004. ACM.
- [Llo82] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, IT-28(2):129–137, mar 1982.
- [LLX⁺01] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001.
- [MHTG05] Matthias Muller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 471–478, New York, NY, USA, 2005. ACM.
- [Ost99] Victor Ostromoukhov. Digital facial engraving. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 417–424, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.
- [PFH00] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *Proceedings of ACM SIGGRAPH 2000*, pages 465–470, July 2000.
- [PL90] P. Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.
- [PP93] Kris Popat and Rosalind W. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression. In *In Visual Communications and Image Processing*, pages 756–768, 1993.
- [SABS94] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 101–108, New York, NY, USA, 1994. ACM.
- [SCA02] Cyril Soler, Marie-Paule Cani, and Alexis Angelidis. Hierarchical pattern mapping. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 673–680, New York, NY, USA, 2002. ACM.

- [Tur91] Greg Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *SIGGRAPH Comput. Graph.*, 25(4):289–298, 1991.
- [Tur01] Greg Turk. Texture synthesis on surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 347–354, New York, NY, USA, 2001. ACM.
- [Wei02] Li-Yi Wei. *Texture synthesis by fixed neighborhood searching*. PhD thesis, Stanford University, Stanford, CA, USA, 2002. Adviser-Marc Levoy.
- [Wei07] Li-Yi Wei. Part i: fundamentals. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 1, New York, NY, USA, 2007. ACM.
- [WFR98] Marcelo Walter, Alain Fournier, and Mark Reimers. Clonal mosaic model for the synthesis of mammalian coat patterns, 1998.
- [WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [WL01] Li-Yi Wei and Marc Levoy. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 355–360, New York, NY, USA, 2001. ACM.
- [WS94] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100, New York, NY, USA, 1994. ACM.
- [WY04] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Trans. Graph.*, 23(3):364–367, 2004.
- [YHBZ01] Lexing Ying, Aaron Hertzmann, Henning Biermann, and Denis Zorin. Texture and shape synthesis on surfaces. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 301–312, London, UK, 2001. Springer-Verlag.
- [ZG03] Steve Zelinka and Michael Garland. Interactive texture synthesis on surfaces using jump maps. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 90–96, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
- [ZMT06] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Vector field design on surfaces. *ACM Trans. Graph.*, 25(4):1294–1326, 2006.