Procedural Descriptions of Anisotropic Noisy Textures by Example

Guillaume Gilet[1], Jean-Michel Dischler[1] and Luc Soler[2]

[1]LSIIT UMR CNRS-UDS 7005, Université de Strasbourg, France
[2]IRCAD, France

Abstract

This short paper introduces a new approach to automate the creation of procedural anisotropic "noisy" textures by using an example. As for past approaches that allow one to obtain procedural descriptions of stochastic textures, it uses a sum of multi-scale noise functions and a combined spectral / histogram-based approach. The improvement, here, consists in better controlling the spectral domain by using Gabor noise functions. This allows us to extend the range of textures that can be addressed, while bringing a number of advantages compared to classical example-based texture synthesis: extreme compactness, continuous definition over infinite space, easy extension to solid (even animated solid) textures and straight texture value computation in the fragment shader.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction, related works and motivation

Classical example-based synthesis, as initiated by [HB95], often relies on the pre-generation of texture images (see survey of [WLKT09]), thus requiring an explicit storage in data arrays for rendering. This can become a problem for many interactive applications, since texture memory is limited on the GPU. Procedural textures [EMP*98] avoid this limitation. But, unfortunately, creating procedural textures by using an example image is extremely difficult. There are thus currently only few published methods that tackle this problem. In [DG97], for example the authors focus on procedural displacement maps using example 1D profiles. The parameters of a sum of noise functions (amplitude and scale) are progressively adjusted with a combined spectral and histogram analysis. Beyond displacement textures, isotropic color textures were also processed by analyzing just 1D lines extracted from 2D pictures. In [LVLD09], a more complete spectral and histogram-based technique is proposed to determine the parameters of the sum. Unfortunately, it is likewise limited to isotropic and fully stochastic textures. Bour et al. [BD04] propose to browse the parameter spaces of a complete database of shaders. While addressing a larger range of textures, this technique requires, however, a database of pre-existing shader programs.

Our motivation is to extend the works of [DG97] and [LVLD09], to be able to process more complex noisy textures, including anisotropic ones. Although combined spectral and histogram-based approaches were abandoned for years by the example-based texture synthesis community, we show that it remains the better solution for stochastic textures, especially if targeting rendering applications. This is because of the many advantages that a procedural description provides compared to explicit data arrays. Compared to [BD04] no pre-existing database of shaders is required in our case. Stated shortly, our main contribution consists in proposing a new original way of decomposing the 2D spectral domain into ellipses, by exploiting some spectral properties of the recently introduced Gabor noise [LLDD09].

2. Multiscale random decomposition

Following up [DG97], our approach consists in defining a texture function $T(X)$, $X=(x,y)$ in the form of a sum of N noise functions, combined with a histogram function H. To account for anisotropy, we further introduce a rotation $R_\alpha(X)=(x\cos(\alpha)-y\sin(\alpha), x\sin(\alpha)+y\cos(\alpha))$:

$$T(X)=H[\sum_{i=0}^{N} w_k G_n^k(R_{\alpha_k}(s_k X))] \quad (1)$$

with w_k weights, $s_k = (s_x^k, s_y^k)$ scales and α_k rotation angles. A suitable random basis function for G_n^k is Gabor noise as it unifies a spatial (Gaussian) and spectral (cosine) component.

3. Spectral decomposition using ellipses

Our technique, concisely summarized in what follows, is based on some simple mathematical properties: it is established that for a given 2D Gabor noise $G_n(X)$ with cosine frequency f, stripe orientation θ, and isotropic standard deviation σ the spectral domain matches a disk, with its center position c_u, c_v depending on f and θ, e.g. $c_u = f\cos(\theta)$,$c_v = f\sin(\theta)$ and with radius r related to σ (in fact, Gabor noise allows one to define rings centered on the origin, but we have chosen to use disks, because these can be shifted on any location in spectral domain, which allows use to consider even the most complex spectral energy distributions). See figure 1, top row, left two pictures, showing a Gabor noise and its corresponding spectral domain (there are two visible disks because of symmetry). Calling $h(X)$ a 2D function and $F(h) = \widehat{h}(\xi)$ its Fourier Transform, we have:

$$F(h(aX)) = \frac{1}{|a|}\widehat{h}(\frac{\xi}{a}) \quad (2)$$

$$F(h(R_\alpha(X)) = \widehat{h}(R_\alpha(\xi)) \quad (3)$$

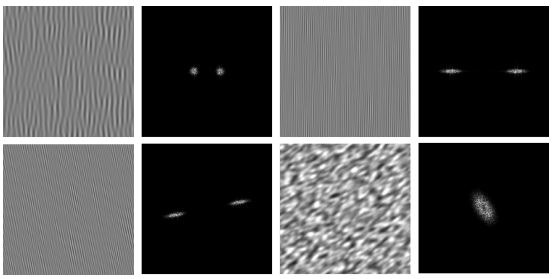

Figure 1: *Gabor noise spectral properties that we exploit. The spectral disk becomes an ellipse by scaling. It can be further rotated by spatial rotation. The ellipse is centered if the Gabor noise frequency component is null (bottom, right).*

with $a = (a_x, a_y)$ non-zero scaling factors and $\xi = (\xi_u, \xi_v)$ frequency coordinates. The first property means that if we apply a scaling a to a 2D Gabor noise $G_n(X)$ with fixed value σ, the corresponding disk in spectral domain becomes an axis aligned ellipse E with center $(E_u = c_u/a_x, E_v = c_v/a_y)$ and radii $r_u = r/a_x, r_v = r/a_y$ (see figure 1 first row, two pictures on the right). The second property means that if we further apply a rotation α to the scaled G_n, the ellipse will be also rotated by angle α, thus modifying its position and main radius direction (see figure 1 second row, two pictures on the left). We note that when the frequency component f of G_n is null, the ellipse becomes centered on the origin (see second row, two pictures on the right). Let us now reverse the problem. Given an ellipse E in spectral domain, defined by its center position $(E_u, E_v) \neq (0,0)$, its main direction E_α, and its two radii (r_u, r_v), we are able to associate a scaled and rotated Gabor noise, since the parameters are linked by a simple equation system. By considering the scaled and rotated Gabor noise parameters f, θ, s and α as unknowns (σ is assumed to be fixed, thus r is known and also fixed), this system can be straightforwardly solved to obtain:$\alpha = -E_\alpha$, $s = (r/r_u, r/r_v)$, $\theta = \arctan((E_v\cos\alpha - E_u\sin\alpha)/(E_u\cos\alpha + E_v\sin\alpha))$ and $f = \sqrt{E_u^2 + E_v^2}$. The particular case of an ellipse centered on origin is even more easy.

Since we are able to retrieve Gabor noise parameters from any ellipse in spectral domain, our proposal consists in decomposing the spectral domain of an example image into a set of N ellipses, each being associated with one G_n^k, $k \in [1,N]$. To approximate the spectral domain by a set of ellipses, we propose to apply quantization. That is, the energy spectrum is quantized into q values, q chosen by the user, see figure 2. On the first row, the left shows the example, the middle its spectral domain and the right quantization using $q = 3$.

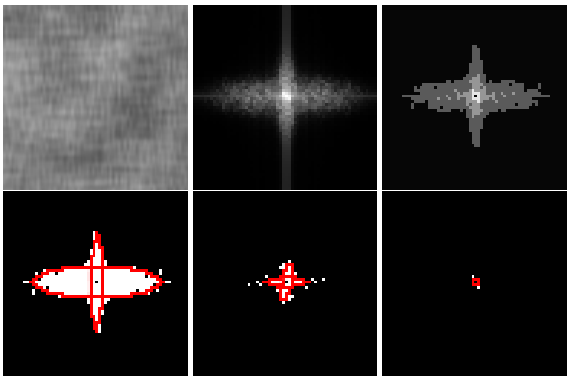

Figure 2: *Analyzing an input texture in frequency domain by quantization and by computing matching ellipses to derive Gabor noise parameters.*

The second row shows the three binary areas resulting from quantization. A set of ellipses is determined for each area by a minimization procedure. That is, the ellipses are computed such that they "best" overlap the area. For one ellipse E, this is numerically expressed by following minimization: $min_{E_u,E_v,r_u,r_v,E_\alpha} |\{E(i,j) \oplus A(i,j)\}|$, where $\oplus$ represents the binary "exclusive or" operator and $||$ the cardinal number of a set of pixels and $A(i,j)$ the binary area. We note that more than one ellipse might be necessary to cover one area. In practice, we start the procedure with one ellipse, computed iteratively using the previous minimization formula, and then add a second ellipse. The latter is kept only if the area overlap is improved by a minimum factor. For computational efficiency, we never exceed two ellipses per area. The resulting ellipses are highlighted on figure 2. Five

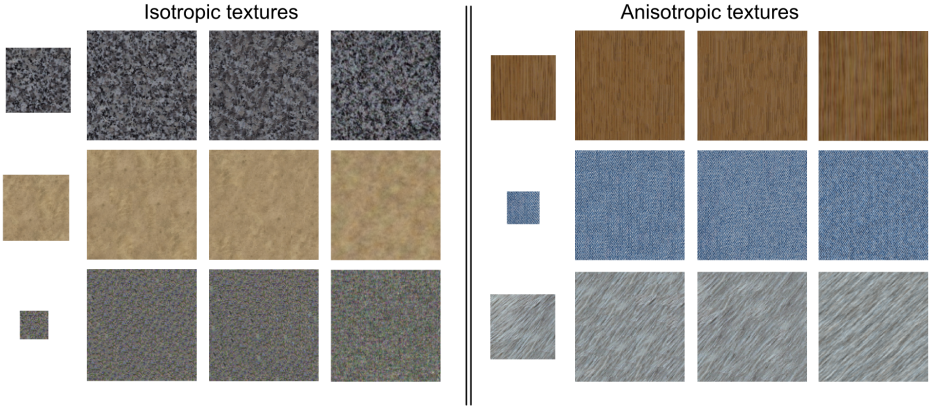

Figure 3: *Results for noisy isotropic (left side) and anisotropic (right side) textures. From left to right: the input example, quilting [EF01], tsvq [WL00] and our procedural texture.*

ellipses are visible, thus resulting in $N = 5$ Gabor noises. Whenever the radius of an ellipse is very small, we replace the Gabor noise by a cosine wave. Strongly focused high amplitude components in spectral domain, usually correspond to periodic components in spatial domain.

The only remaining unknown noise parameter in formula 1 is w_i. As value, we use the amount of energy in spectral domain covered by the corresponding ellipse. But these values need to be further refined since ellipses overlap. In addition, the histogram function H has an influence on the final energy distribution of T in frequency domain. To progressively adjust the weights, we use the iterative relaxation technique described in [DG97]. The histogram function H is computed as in [DGF98] to make the histogram of the example match the histogram of T. We use a table of 128 bins that are stored in texture memory.

4. Results

All results in this section have been obtained with a PC with Intel Core 2 Quad Q9300 CPU (4Gb RAM), and a NVidia GeForce GTX 280, with 1Gb RAM. To process color, we use, as for early pyramidal and spectral approaches, a decomposition into principal components in order to decorrelate the three color channels. These are then processed separately. This approach is known to have drawbacks (e.g. sometimes the separate color processing does not well preserve the initial color spectrum), but in our case it has a major advantage: the transformation is linear. So, it is possible to get back to the usual RGB color space by using a simple 3×3 matrix product, which can be performed directly on the GPU. Figure 3, left part, shows three examples of noisy isotropic textures (granite, animal skin and carpet) and, right part, three examples of anisotropic textures (wood, denim and fur). Note that denim contains some regular aspects. For each example, we show from left to right: the input example,

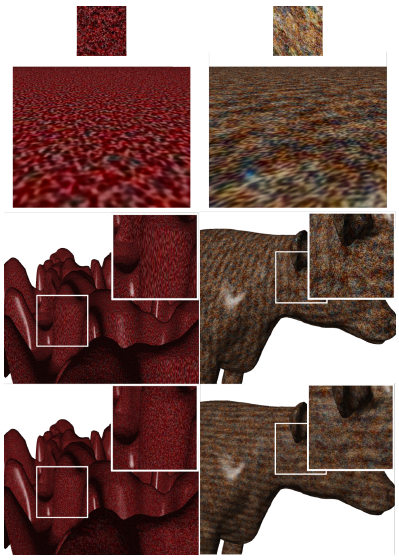

Figure 4: *Two examples of procedural textures applied to 3D surfaces: (top row is the example). Second row: rendering on a infinite planar surface. Third row: (left) distortions, (right) discontinuities due to parameterization. Fourth row: extension to 3D to avoid these issues.*

result from quilting [EF01]), from tsvq [WL00] and finally our result. In fact, we expected classical pixel- and patch-based methods to perform much better than our technique, since these are aiming at good quality. This is true for some textures, especially with quilting, which often provides the better qualitative result. Yet, our approach remains highly competitive. In some cases, it even provides better results as it avoids repetition impression, especially when the input example is small (see carpet). Our textures, however, have a

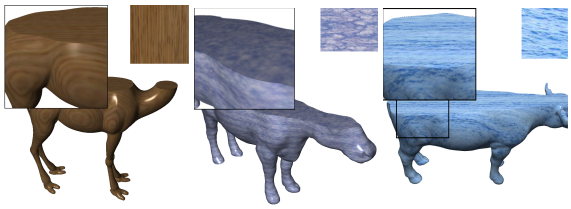

Figure 5: *Left and middle: extension to solid (3D) texture. Right: extension to 4D (animated 3D) texture.*

major advantage that compensates a lot limitations: they are all fully procedural. Two examples of resulting advantages are shown on figure 4 (top row are the examples: red stone and rust). Since our textures are defined on a per-pixel shader basis, they can be straightforwardly applied to an infinite planar surface without any repetition (second row). The user can select in real-time the apparent resolution of textures, which can be infinitely high. Another advantage is shown on third and fourth rows. The third row illustrates texturing based on a 2D surface parameterization using 2D noises. But surface parameterization might cause texture distortions (see left) or discontinuities (see right). Our procedural textures can be easily extended to 3D by using a 3D noise instead of a 2D one, thus avoiding these problems (see last row). Beyond 3D extension to fill the interior of objects (see figure 5, left and middle), even 4D extensions are possible for time variation (figure 5, right is one frame out of an animation shown in the accompanying video). The parameters of the third and fourth dimensions can be taken from one of the other two dimensions. Extension to arbitrary dimensions is a major advantage, since even in full 3D or 4D, our textures still require less than 0.5 Kb memory (for storing H). To our knowledge, 4D textures have not been addressed so far, because of timing and memory issues. All textures presented here are defined by less than 10 noises, plus some cosine waves in some cases (maximum 18 waves for the denim). The parameters of all noises have been computed quite fast. In the worst case, it took 3 minutes for the wood texture that uses 7 noises and 6 cosine waves. Antialiasing is also possible with our approach, because of the underlying multi-scale description of the procedural texture (using the filtering scheme proposed in [LLDD09]). We note that we did not compare our approach to [BD04], that also produces procedural textures. The reason is that the result depends a lot on the quality of the underlying shader database. A fair and objective comparison, without previously analyzing and comparing the contents of the shader databases, is thus difficult. One could criticize the fact that only unstructured textures can be addressed. But as shown by figure 6 noisy patterns are actually quite common. They can be present inside structured textures, in the form of small scale "sub-patterns". Since our method works well even for very small texture pieces, we believe that a subdivision of input textures into

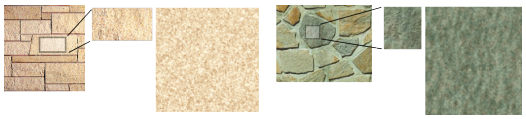

Figure 6: *Examples of subtexture synthesis.*

sets of small scale patterns could be a good starting point for obtaining procedural descriptions of structured textures. On the performance side, all of the textured objects are rendered at more than 60fps with a on-the-fly reconstruction of the texture in the pixel shader.

5. Conclusions

In this paper we have presented a new approach for noisy procedural texture synthesis based on examples. It extends previous work as it allows us to process a wider range of stochastic textures (anisotropic, like wood, as well as semi-regular, like denim). We have based our approach on a multiscale formulation using Gabor noise functions summed with different weights. We believe our approach can be a good starting point for a more complete analysis technique that would automatically produce procedural descriptions of structured textures.

References

[BD04] BOURQUE E., DUDEK G.: Procedural texture matching and transformation. *Computer Graphics Forum 23*, 3 (2004), 461–468. 1, 4

[DG97] DISCHLER J., GHAZANFARPOUR D.: A procedural description of geometric textures by spectral and spatial analysis of profiles. *Computer Graphics Forum 16*, 3 (1997), 129–139. 1, 3

[DGF98] DISCHLER J. M., GHAZANFARPOUR D., FREYDIER R.: Anisotropic solid texture synthesis using orthogonal 2d views. *Computer Graphics Forum 17*, 3 (1998), 87–96. 3

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *SIGGRAPH 2001* (2001), ACM Press, pp. 341–346. 3

[EMP*98] EBERT D., MUSGRAVE K., PEACHEY P., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*. 1998. 1

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *SIGGRAPH* (1995), pp. 229–238. 1

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse gabor convolution. In *SIGGRAPH 20909* (2009), pp. 1–10. 1, 4

[LVLD09] LAGAE A., VANGORP P., LENAERTS T., DUTRÉ P.: *Isotropic stochastic procedural textures by example*. Tech. rep., May 2009. 1

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 2000* (2000), pp. 479–488. 3

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR* (2009), Eurographics Association. 1