Eurographics Symposium on Rendering 2010
Jason Lawrence and Marc Stamminger
(Guest Editors)

Volume 29 (2010), Number 4

An Image-Based Approach for Stochastic Volumetric and Procedural Details

G.Gilet & J-M. Dischler

LSIIT UMR CNRS-UDS 7005, Université de Strasbourg, France

Abstract

Noisy volumetric details like clouds, grounds, plaster, bark, roughcast, etc. are frequently encountered in nature and bring an important contribution to the realism of outdoor scenes. We introduce a new interactive approach, easing the creation of procedural representations of "stochastic" volumetric details by using a single example photograph. Instead of attempting to reconstruct an accurate geometric representation from the photograph, we use a stochastic multi-scale approach that fits parameters of a multi-layered noise-based 3D deformation model, using a multi-resolution filter banks error metric. Once computed, visually similar details can be applied to arbitrary objects with a high degree of visual realism, since lighting and parallax effects are naturally taken into account. Our approach is inspired by image-based techniques. In practice, the user supplies a photograph of an object covered by noisy details, provides a corresponding coarse approximation of the shape of this object as well as an estimated lighting condition (generally a light source direction). Our system then determines the corresponding noise-based representation as well as some diffuse, ambient, specular and semi-transparency reflectance parameters. The resulting details are fully procedural and, as such, have the advantage of extreme compactness, while they can be infinitely extended without repetition in order to cover huge surfaces.

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

Using photographs for creating complex photo-realistic objects and textures is currently a topic of great interest in computer graphics and rendering, as it reduces and eases the work of artists and content creators. Moreover it generally leads to improved rendering results with respect to realism. However, there are difficulties gathering textures from photographs, as textures are provided already mapped onto 3D surfaces. Textures with underlying relief (bark, roughcast, etc.) raise even more problems, due to masking, parallax and lighting effects at texture scale. Obviously, for such textures, cropping a part of the photograph and reproducing it with classical 2D texture synthesis without considering underlying relief does not produce satisfactory rendering results: it looks like wallpaper while the small-scale relief impression at texture scale is lost. To be rendered realistically, textures with geometric details require some 3D information along with a specific 3D rendering technique, like for example displacement mapping [Coo84] or volumetric texture mapping [KK89]. In other words, for such textures, a 3D geometric information must be recovered from the photograph.

Recovering some geometric and/or relief information from a single photograph at texture scale is clearly a difficult problem, traditionally solved by texture measurement or vision techniques. Unfortunately, these methods often require complex manipulations involving specific hardware devices or imposing drastic constraints on the surface reflectance and on the lighting and viewing condition. They do not generally work for arbitrary structures, especially when small scale details are involved.

In this paper we propose a solution that avoids an explicit geometric reconstruction at texture scale, by focusing only on noisy (stochastic) small-scale details. Such details are very frequent in natural scenes (as presented in the various real-world examples of figure 1) and can generally be well represented by noise functions [Per85].

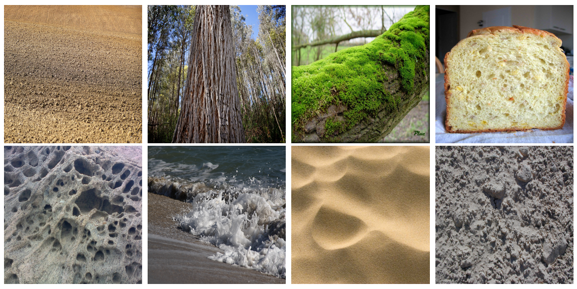
Figure 1: *Photographs of real-world objects covered by various stochastic details.*

Practically, we propose an interactive solution inspired by image-based modeling and rendering (IBMR) techniques. No pre-existing shader database is required, nor do users need any shader programming knowledge. As in previous works, our model uses a weighted sum of noise functions at different scales. To determine the parameters of this sum, we use an iterative error minimizing technique, that bears some similarity with [BD04], but with a different metric based on multi-scale filter responses. Such filters are better suited for processing the non-stationary aspect of 3D details. However, unlike classical image-based techniques that only re-map color textures, the details recovered with our method are real 3D, thus allowing zooming and relighting with masking and parallax effects.

2. Related works

Noise and procedural textures. Noise has been introduced to the computer graphics community by Perlin in [Per85] more than two decades ago. Since then, it has been used to model various natural phenomena and textures [EMP*98]. Recent works (see for instance [Per02, GZD08, LLDD09, LLC*10]) demonstrate that procedural textures remain highly useful for rendering applications. Procedural details have the major advantage that they can be used to visually enhance huge environments without towering memory requirement.

Unfortunately the creation of procedural models can be tedious and often requires programming knowledge. Therefore, some approaches attempt to automatically match parameters to examples, like in [BD04]. However, this method only considers color patterns and does not create new shaders but relies on a pre-existing shader database, thereby limiting the scope of the application. Other methods exploit some spectral properties of recent noise functions [LLDD09] to create procedural textures ([LVLD09], [GDS10]). But again, no photographs are used and only color textures with no geometric details are considered. Such methods are thus ill suited to realistically render complex natural effects, like the examples of figure 1.

Texture acquisition and measurement. Texture synthesis by example generally ignores volumetric effects. Nevertheless, it is possible to create solid textures from 2D color samples. By empirically associating a given transparency to some color bands, volumetric and geometric effects can be rendered, as done for instance in [KFCO*07]. Yet such an approach fails for most volumetric cases based on photographs. An alternative and accurate solution therefore consists in making measurements. Many measurement systems, along with corresponding texture models have been proposed, especially by using view and/or light dependent approaches, like polynomial textures [MGW01] or bidirectional textures [DvGNK99]. Such techniques improve realism and implicitly take into account underlying texture relief. Unfortunately, using sophisticated measurement devices and digitization protocols is far more complex than using a single example photograph. It also involves lots of technical constraints.

Textures from photographs. Extracting textures from photographs is a broad and common technique used in the field of image-based modeling and rendering (IBMR). Users create a coarse approximation of the real-world 3D object and overlap it with its photograph in order to extract one corresponding color texture map or even view-dependant maps [DTM96]. Individual patterns might also be extracted and then reproduced using texture synthesis. In this case, texture distortions due to surface curvature and perspective have to be considered [ELS08]. However, this technique does not consider geometric effects at texture scale, thus hindering the recovery of consistent parallax and lighting effects for textures where underlying geometry and relief prevail. In fact, most image-based techniques just ignore small-scale geometry at texture scale.

To the best of our knowledge, methods that attempt to reconstruct volumetric details from a single image are rare, probably because of the extreme difficulty of this problem. Most methods focus on specific features like trees [TFX*08] or hair [PCK*08, WYZG09, BPvdP*09] and use statistical approaches instead of an accurate 3D reconstruction. The approach of [DG97] can be considered as the closest to our concern: it uses examples to create displacement textures and creates procedural models based on a sum of noises at different scales. However, this method does not use photographs as input examples. It is limited to 1D profiles, e.g. curves. Our approach also uses elements of IBMR, like in [ELS08], since we place a virtual object over the picture. We also use an iterative parameter fitting technique similar to [BD04], but with a new error metric.

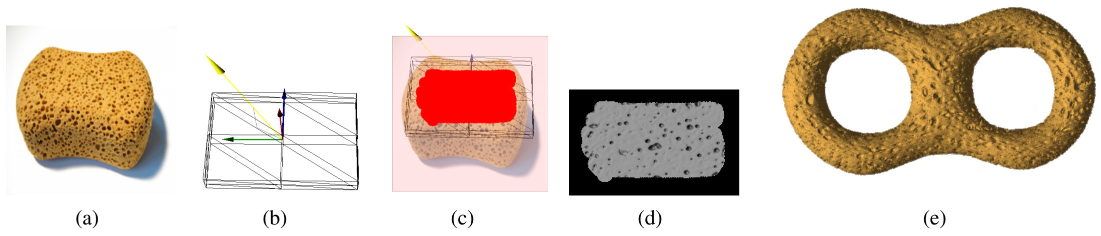

Figure 2: *Workflow of our method. Using an example photograph (a), the user gives a coarse approximation of the geometry (b) and chooses a relevant zone for analysis (c). Results consist of a set of parameters for a procedural representation (d), which can be applied to any shape (e).*

3. Method principles

The basic principle of our approach is summarized by figure 2. The user supplies a photograph I_p of an object covered by noisy 3D details. He then provides a simple 3D shape O (parallelepiped, ellipsoid, etc.) matching some parts of the real object as well as an estimated lighting condition. In our case, the user further delineates a validity mask, since there is generally no exact match between O and the real object. Only pixels of this zone will be considered for computation. The remaining steps are then fully automatic. They consist in computing a set of parameters for a multi-scale noise representation of the details. The user also gets the parameters for a simple ambient, diffuse and Phong specular reflectance model, including, if desired, semi-transparency. Obtained details are 3D and can be transferred to any shape and rendered under new lighting and viewing conditions, including environment map-based lighting.

In the remaining parts of this paper we focus on the automatic part of our procedure, the manual part being a human-computer interaction task out of our scope. We note that we will also decouple two texture aspects: color patterns and geometric details. Many textures unify both aspects, *e.g.* color and relief. Unfortunately, decorrelating both is a difficult problem because of shading/lighting conditions. Indeed, it is hard to evaluate in a photograph whether an intensity variation is due to small-scale relief or to a local change of the color of the object. Color patterns do not represent the core topic of our paper: we will use an existing method [GDS10] to create procedural color patterns in order to further enhance the visual similarity with the photograph.

Our details extraction technique relates to the fact that many natural 3D phenomena are based on random processes acting at different scales on objects (erosion, corrosion, stochastic motion, etc.). We shall assume, in our case, that the 3D details are resulting from a random process that can be expressed as a stochastic "perturbation" function $r(\mathbf{x})$, returning a random value normalized between $[0,1]$ for any point $\mathbf{x} = (x,y,z)$ of space. We further assume that any given random function $R(\mathbf{x})$ can be expressed as a linear combination of simpler "basis" random functions at different scales (we will call this basis function *noise*):

$$R(X) = \sum_{i=0}^{n} w_i F_n(f_i X) \tag{1}$$

where F_n denotes the noise function, w_i weights and f_i scales. Our objective is to express $r(\mathbf{x})$ in the form of $R(\mathbf{x})$. Such a multi-scale expression matches the fractal geometry of nature and has been used for years to model various natural phenomena (terrains, clouds, etc.). Contrary to Fourier or wavelet decompositions, multi-scale coefficients cannot be computed in a deterministic way in the case of stochastic processes. In fact, R does not need to verify: $r(\mathbf{x}) = R(\mathbf{x}) + \varepsilon(\mathbf{x}) \forall \mathbf{x}$. We just want $r(\mathbf{x})$, the original random process, and $R(\mathbf{x})$, the corresponding multi-scale expression, to be "close" from a purely statistical point of view. Let C be a corresponding set of statistical characteristics. In fact, we want $C(r) \approx C(R)$ instead of $r(\mathbf{x}) \approx R(\mathbf{x})$.

For our 3D details synthesis application, we have to select a suitable noise function F_n as well as a set of statistics C. The corresponding parameters of R can then be computed using an optimization technique that iteratively fits the parameters of equation 1 using gradient descent, so as to minimize the error $e = |C(r) - C(R)|$. Unfortunately, in our case, we do not have access to the random function r. So, we cannot compute any statistics C. Instead, we have a photograph of the result of the random process r on an object. We thus have to transfer statistics for random signals to statistics for pictures. Therefore, we must introduce a picture metric $M(I,I')$ that measures the similarity between two texture images I and I'. Based on this metric, we can first compute a synthetic image I_s using the user-defined virtual object O enhanced with stochastic details obtained from function R. This image is then compared to the picture of the photographed object I_p characterized by details resulting from random process r. In the next three sections, we discuss respectively the choice of the noise function, the choice of the texture similarity metric M and the method to compute I_s with O and R before describing our iterative technique used to fit parameters.

4. Random Basis Function

The choice of the noise function F_n influences the range of stochastic phenomena that we will be able to represent. The spatial and spectral (*i.e.* Fourier power spectrum) domains are often used to characterize noise functions. The former matches the probability distribution function (PDF), while the latter matches autocorrelation. An efficient noise function unifying both aspects is a function based on sparse convolution, e.g. based on a set of randomly distributed points convolved with a given kernel function (Linear ramp, Gaussian, Gabor, etc.) using a given distance metric (Euclidian, Manhattan, etc.).

To address a wide range of natural phenomena, we propose to use a formulation unifying cellular noise [Wor96] and Gabor noise [LLDD09] in a single expression. That is, we consider a convolution with all $k-th$ to $k'-th$ ($1 <= k <= k'$) closest points, using a simplified Gabor kernel function. Our unified noise $N(\mathbf{x})$, $\mathbf{x} = (x,y,z)$ is defined as:

$$N_{k,k',\phi,\theta}(\mathbf{x}) = \sum_{i=k}^{k'} e^{\frac{-D(P_i,\mathbf{x})}{2\sigma}} cos(2\pi\phi(xcos(\theta_i)+ysin(\theta_i))) \tag{2}$$

P_i represents the i-th closest point to $\mathbf{x}$ (points are ordered by increasing distance), $D(\mathbf{x},\mathbf{y})$ a distance metric (we use an Euclidian distance), σ the standard deviation of the Gaussian envelope, ϕ the cosine frequency and $\theta_i \in [0,\theta]$ a random cosine stripes orientation. σ controls the radius of influence of the convolution kernel, and thus should be set according to the spatial distribution of the points P_i. This is an implementation dependant parameter that will be selected once and for all, thus removing it from the set of parameters of $N(\mathbf{x})$. To control anisotropy and scaling, we introduce a diagonal 3D matrix S defined by (s_x, s_y, s_z). Orientation is then further given by a 3D orthonormal frame matrix Q. We can now substitute F_n by $N_{k,k',\phi,\theta}(Q(S(\mathbf{x})))$ in equation 1.

To avoid a combinatorial explosion during the parameter fitting step (section 7), we must keep the amount of parameters of R as low as possible. Therefore, we propose to consider only "harmonic" sets of noises. Harmonics represent a common technique for creating multi-scale natural phenomena (such as Perlin's turbulence function). Indeed, for many natural phenomena, the frequency distribution in spectral domain matches a Gaussian shape, which is well accounted for with harmonics. Subsequently, the noise functions of formula 1 do all share the same parameters k, k', θ and ϕ. The coefficients w_i and f_i are replaced by a single factor $m > 0$.

The PDF, which corresponds to a visually prominent part, is an important factor to be taken into account. In order to efficiently control the PDF, we introduce a function $H(t)$, $t \in [-1,1]$, returning a value between 0 and 1 that will shift the values of noise, so as to modify its basic PDF.

The final formulation of R is then:

$$R(\mathbf{x}) = H\left(\frac{1}{\eta}\sum_{i=0}^{n}\frac{1}{m^i}N_{k,k',\phi,\theta}(m^i Q(S(\mathbf{x})))\right) \tag{3}$$

where η is a constant normalization factor. The parameters of R are n, m, k, k', ϕ, θ, Q, (s_x, s_y, s_z), plus parameters defining H. To keep the number of parameters of H as low as possible, we use a piece-wise linear function, depending on a little number of equally spaced control points h_i, $i \in [0, n_H]$. H is defined as: $H(t) = h_{\lfloor t/n_H \rfloor} * (1 - frac(t/n_H)) + h_{\lfloor t/n_H \rfloor + 1} frac(t/n_H)$, where $frac()$ means the fractional part and $\lfloor\ \rfloor$ the integer part.

5. Texture similarity metric

In this section we describe the similarity metric M that we use to compare the photographed details (i.e. the supplied picture I_p) resulting from random process r, with the synthetic details (i.e. the computed picture I_s) resulting from function R (equation 3). There is a large amount of literature concerning the discrimination and classification of textures. In [DG97, BD04], a combined spectral and histogram metric has been used to compare an example texture with a synthetic procedural texture. But such an approach assumes that the texture is stationary. In our case, according to the object's shape, the viewer's position, the lighting condition and the geometry of the details, the visual aspect of the same texture can vary a lot on I_p. Computing a global histogram and Fourier transform of I_p consequently makes little sense.

Instead, we propose to use a metric based on local filter responses, such as Gabor filters and windowed Fourier transforms, which have been widely studied in the past [DH95, RHS99, AWY97]. These filters are interesting for our texture metric, since they unify both spatial and spectral information. Concretely, we use a filter of finite support, based on a convolution mask of size $(2m_s)^2$, applied at multiple image resolutions. Let us call I^l the l-th level of a Gaussian pyramid of an image I, such that $I^0 = I$. The filter response ρ around pixel (i,j) at level l for discrete frequencies (u,v) is defined by the following windowed Fourier transform:

$$\rho_{i,j,l}(u,v) = \sum_{a=-m_s+1}^{m_s}\sum_{b=-m_s+1}^{m_s} I^l(a+i,b+j)\cdot e^{\frac{-(a^2+b^2)}{2\sigma^2}} e^{\frac{-2i\pi((a+m_s-1)u+(b+m_s-1)v)}{2m_s}} \tag{4}$$

The window is an isotropic Gaussian function centered on 0. The filter result ρ is a complex number characterized by its amplitude and phase.

We consider that two images I and I' depict similar stochastic details (textures), if the distribution of amplitude of filter responses applied for different frequencies (u,v) and levels l are similar. This similarity is measured by comparing histograms $h_{u,v,l}(I)$ composed of h_n bins. The histogram value $h^b_{u,v,l}(I)$ associated with a given bin b is defined by the normalized number of values of $\rho_l(u,v)$ that fall within the range $[v_b, v_{b+1}[$ corresponding to b for all (i,j) of image I. The texture comparison metric M between two images I and I' is thus defined by following mean squared error (MSE) :

$$M(I,I') = \sum_{b,u,v,l} \left(h^b_{u,v,l}(I) - h^b_{u,v,l}(I') \right)^2$$

The histograms are depending on frequencies u,v and on level l.

Figure 3 shows the actual set of histograms we used for characterizing textures. Three texture examples are shown, with three sets of histograms for levels 0, 1 and 2. For each level we have a set of 17 histograms (each composed of 32 bins). These histograms correspond to frequencies $(u,v) \in \{(0,0),(1,0),(1,1),(0,1),(1,-1),(2,0),(2,2),(0,2),(2,-2),(2,1),(1,2),(-2,1),(1,-2),(3,0),(0,3),(3,3),(3,-3)\}$. We used a Gaussian window of size $m_s = 4$ (i.e. 8×8 pixels). The two first examples (top and middle row), are extracted from the same image. They correspond to the same texture. Note the similarity of the histograms, which is confirmed by the low MSE of 0.9, 2.3 and 6.4 respectively for levels 0, 1 and 2 (the global MSE is 9.6). The third texture has similar frequencies in spectral domain, but is visually different, which is also confirmed by the histograms. In this case the MSE with the first row is 11.5, 17.2 and 30.1 for the same levels (the global MSE is 58.8). For all examples in this paper, we used the same set of 17 frequencies with three levels of scale.

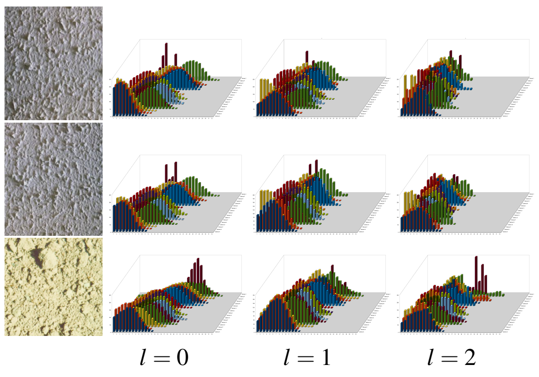

Figure 3: *Histograms resulting from an analysis of visually similar (first two rows) and different (last row) textures.*

6. Computing volumetric details

In this section, we describe how R (equation 3) can be associated to a virtual object O so as to compute an image I_s that will be compared with the photograph I_p during the iterative error minimization procedure. In order to synthesize the details, we propose to use two common approaches. The first is displacement mapping, which consists in shifting surface points along their normal: $P' = P + A_d R(P) N_P$, where P represents the surface point, N_P its normal, A_d an amplitude of displacement and P' the resulting shifted point. R can be limited to a 2D scalar function defined on the object surface in this case. But it can also be a 3D function, to avoid the need for a surface parameterization.

The second is a generalization of displacement mapping to full volumetric objects [PH89]. This technique is called hypertexturing. In our case, points are shifted along the gradient of density: $P' = P + A_d R(P) G(P)$, where P represents the point to be shifted (including object interior points) and G the normalized gradient of density on P. In this case the function R must be defined in full 3D space. To apply such a generalized displacement, we use the hardware accelerated method of [KPH*03]. It consists in applying a direct volume rendering technique to an object stored as 3D density texture on the GPU.

To compute picture I_s we also need a local lighting model, as well as a transparency transfer function th which associates semi-transparency with density values $d(P)$ of O [KPH*03]. As lighting condition, we use the one provided by the user (generally a single light source direction). To simplify the transfer function, we use a power function depending on a single coefficient d_h: $th(x) = pow(d(P), d_h)$. We then apply a local shading model based on ambient, diffuse and specular coefficients, respectively Ka, Kd and Ks.

7. Iterative parameters fitting technique

An iterative optimization technique is used to determine the parameters of the random function R. The optimization consists in determining the global minimum of the image comparison metric $M(I_p, I_s)$, which is a scalar function depending on R. The latter is defined by parameters: n, m, k, k', ϕ, θ, Q, (s_x, s_y, s_z), and the control points h_i of H. All other parameters are set by the user: the object O, the lighting condition and the amplitude A_d of displacement. The user also decides whether or not there is semi-transparency and / or a specular component in the reflectance model.

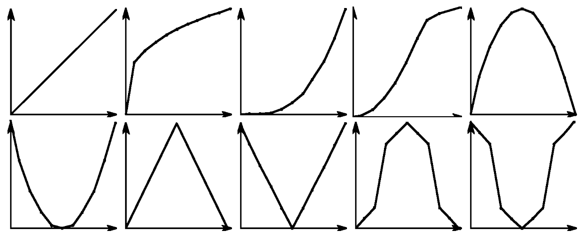

Figure 4: *Initial set of control points used for H.*

A common numerical approach for solving such a multi-dimensional function consists in starting from an initial

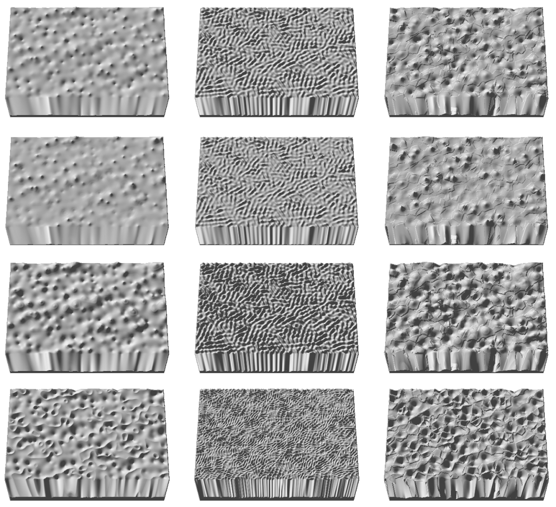

Figure 5: *Examples of noise results for various parameters. Each column corresponds to noise generated with the following parameters : (first) k,k'=$(1,50)$, $\phi = 0$, $\theta = 0$; (second) k,k'=$(1,50)$, $\phi = 2$, $\theta = \pi$; (third) k,k'=$(1,2)$, $\phi = 0$, $\theta = 0$. For each row, examples respectively use the first, third, fourth and seventh function of figure 4.*

guess and then refining this guess by using a gradient descent based on the Jacobian matrix. The key issue is to provide a good initial guess to avoid falling into a local minimum, instead of the desired global one. Unfortunately, the amount of parameters of R is high. This excludes a naive approach that would consist in subdividing the parameter space into a finite set of samples, and then compute M for all possible combinations of these samples. It is therefore important to make use of heuristics.

The first heuristic we have chosen consists in estimating the anisotropy of R, e.g. Q and (s_x, s_y, s_z). The latter is obtained by orienting Q such that the z axis matches the normal of the virtual object on the center of the user painted matching zone. Then, an anisotropy factor f is computed between x and y using the spectral domain around this center (the mean energy distribution ratio between the two main frequency axes). (s_x, s_y, s_z) is thus replaced by a single unknown global scaling factor s such that $(s_x, s_y, s_z) = (s \cdot f, s, s)$. Another heuristic consists in limiting values for n, k and k', which are discrete. Because of the limited size of the example image, we also limited n to a range between 1 and 4. We noticed that for many couples (k,k') the visual difference of the corresponding noises is low. Hence, we limited values to the sets $(1,50)$,$(1,2)$, $(2,2)$ and $(2,3)$. Because of the rapid amplitude decrease of harmonics, the values of m are also initially limited to a small set: 1.5, 2 and 4. Likewise, the frequencies of the cosine stripes of the noise function are limited to following discrete values: 0, 2 and 4.

One important parameter is H. We empirically fixed the amount of control points to 11, and used a starting set of classical mathematical functions (power, smooth step, etc.). This set is shown on figure 4. Examples of initial noise functions along with the resulting effect on a planar surface (using displacement mapping) are shown in figure 5 (we show a single harmonic, i.e. $n = 1$). The rows illustrate the influence of H, while the three columns represent noises obtained respectively with k,k'=$(1,50)$, $\phi = 0$, $\theta = 0$, with k,k'=$(1,50)$, $\phi = 2$, $\theta = \pi$ and with k,k'=$(1,2)$, $\phi = 0$, $\theta = 0$. The remaining parameters are finally all coarsely sampled.

The iterative procedure starts by computing images using a combination of these initial values. The metric is only computed for the highest levels. A finite set of n-closest configurations is then kept (the ones that give the lowest values for M). For each retained configuration, we start a gradient descent-based refinement. At the end the result providing the lowest error M is kept. At each iteration, before computing M, the coefficients of reflectance are adjusted, by comparing the image luminosity between the example and the synthetic image. The luminosity of the darkest pixels is used to adjust Ka, those of the brightest pixels to adjust Ks and the global mean luminosity for adjusting Kd. The shininess coefficient is adjusted by counting and comparing the amount of "bright" pixels.

We note that in spite of the use of these heuristics, the amount of pictures that must be computed for exploring the parameter space remains high during the initial step. Fortunately, the hardware based rendering technique allows us to compute from 5 to 50 fps, depending on the deformation technique used and the size of the matching area. This allows us to compute hundreds of images each minute. Yet, the entire procedure generally requires several hours (depending on the desired precision) for matching parameters. We also note that obtained parameters are not guaranteed to represent the best match because of possible local minima (this is a classical problem of iterative gradient descent techniques).

8. 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. We conducted several tests with various input images, ranging from full 3D effects, to simpler displacement mapping.

Figure 6 shows five different "full 3D" examples :(from top to down) grass, exotic bark, a cloud, dry ground and sand waves. The left-side images show the example photograph, with the used virtual object on the upper right corner. The right shows a synthesis result rendered on an object with "real" 3D effects. Less than a minute was sufficient to place the virtual object and the light direction. The objects with 3D details are rendered using direct volume rendering at respectively 27, 9, 4, 13 and 32 fps, depending on the sampling

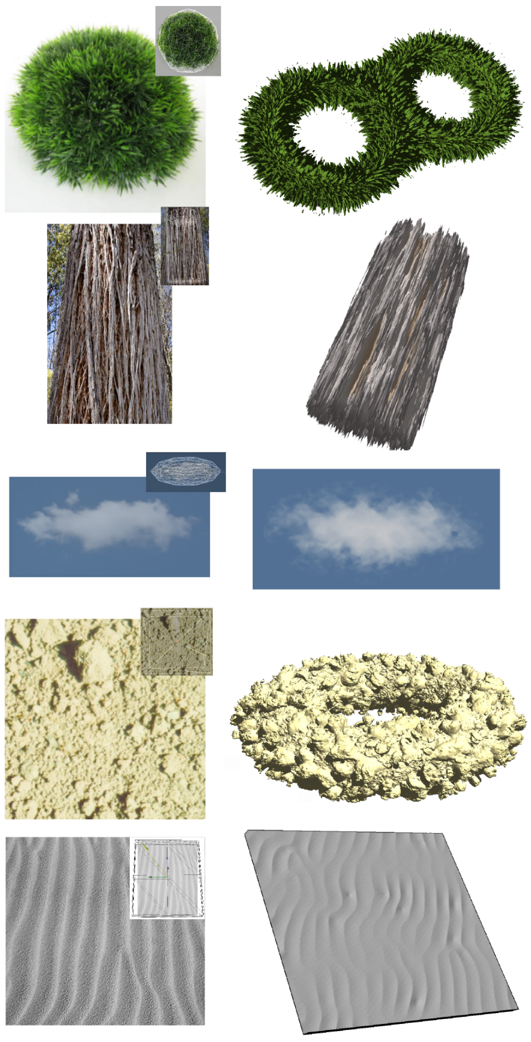

Figure 6: *Five examples of "full 3D" effects obtained with our approach. Left: the example and the virtual object placed by the user. Right: a rendering of obtained 3D details on arbitrary objects with arbitrary lighting.*

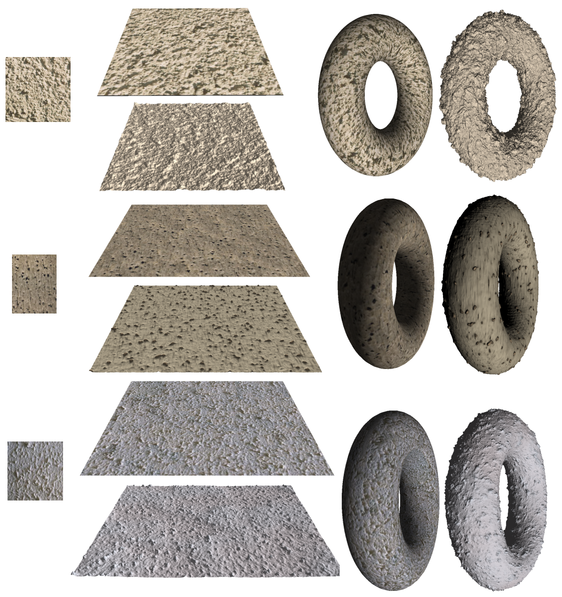

Figure 7: *Three examples of roughcast. Left column: the example. Middle column: texture mapping on a plane (Top: result obtained with graphcut [KSE*03] and classical texture mapping. Bottom: our approach including displacement mapping). Last column: applying the textures to a torus, (left) classical texture mapping, (right) our procedural displacement map.*

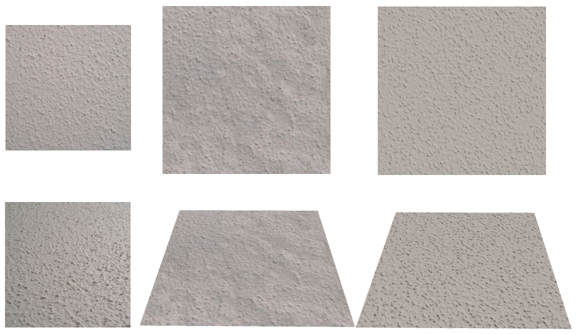

Figure 8: *Evaluating the quality of rendering. Left: two photographs of roughcast for two different views. Middle: classical texture synthesis and texture mapping. Right: our procedural texture including relief. Only the top left image was used as example.*

step of the volume rendering and the amount of noises (semi-transparent objects are rendered at lower framerates because no early ray termination can be applied). All of these textures are fully procedural and use formula 3. We have also tested our approach on simpler displacement map textures, illustrated in Figure 7. In this case, user work was reduced to indicating an estimated light source direction. No virtual object was required. These examples represent different types of roughcast and plaster. The texture parameters computation required 126, 73 and 168 minutes. We have compared our result to a classical texture synthesis technique (in this case the graphcut [KSE*03] and then mapped the result in both cases onto a plane (middle column) and a torus (right column). The difference between both approaches is obvious. A classical texture synthesis technique does not provide convincing results, since underlying texture relief is ignored. We note however that our approach is unable to recover local high frequency color variations. In our case, color is added empirically by correlating it to the height (user work) or by using the color texture synthesis technique of [GDS10] if the latter can be decorrelated from relief. In spite of an approximate processing of color, our approach provides more realistic rendering results in the end, especially when textures are applied to non-planar object surfaces, such as the torus. In addition, our textures are procedurally defined in a continuous space.

In order to evaluate more objectively our approach, we have compared our results with a photograph showing a single roughcast texture for two different viewing directions. This is shown by figure 8. The left shows two photographs of the same wall, one at orthogonal view (top), the other at grazing angle (bottom). The second column shows the result obtained with graphcut and classical texture mapping using approximately the same viewing conditions. Only the top left photograph was used as an example. At grazing angle classical texture mapping fails to capture the change of appearance due to the relief and parallax. The last column shows our result. Again, only the top right example was used. In our case, at grazing view angle, the parallax effect is well recovered, thus globally providing a result that comes closer to the reference (the photograph on the left bottom). For this procedural texture only three noise harmonics were used.

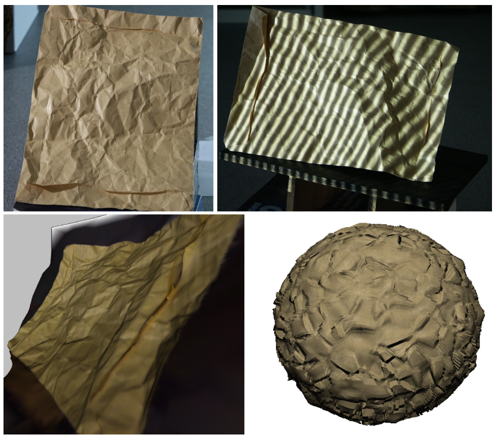
Figure 9: *Comparison between our method (bottom right) and explicit geometry reconstruction with texture mapping (bottom left) using a structured light 3D scanner (top right).*

Figure 9 illustrates limitations and shows a comparison between our method and explicit geometry reconstruction. The first row shows the photograph of the object used as input for our method (left). An explicit acquisition of geometry is used with a structured light 3D scanner (right). While the result using geometry acquisition with texture mapping (last row, left) shows an accurate visualization of the object, our method (right) fails to capture the inherent structure of the texture. The underlying geometry is in this case too structured and hence not well represented by noise.

9. Conclusions

In this paper we have presented a new approach for procedural texture synthesis based on photographs. Unlike classical texture synthesis based on examples, our approach produces procedural descriptions of the textures including relief information. This allows us to improve rendering quality, by increasing realism. It also improves suitability for computer graphics applications by providing extremely compact texture definitions. We have based our approach on a generic multiscale random function. Its parameters are computed by error minimization using a metric based on local filter banks.

One current major limitation is that we can only process noisy details. Although these are frequent in nature, it would be valuable to propose solutions that would be able to reproduce also more structured details, like minerals for instance. We also would like to adapt our approach to some specific structured details such as fur. Fur is more complex to be obtained by photographs, because of additional parameters that need to be taken into account like curliness and surface vector fields.

Acknowledgment

This project has been funded by the French National Agency for Research (ANR) under Nr. ANR-07-MCDO-001-03

References

[AWY97] Azencott R., Wang J.-P., Younes L.: Texture classification using windowed fourier filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19* (1997), 148–153. 4

[BD04] Bourque E., , Dudek G.: Procedural texture matching and transformation. *Computer Graphics Forum 23*, 3 (2004), 461–468. 2, 4

[BPvdP*09] Bonneel N., Paris S., van de Panne M., Durand F., Drettakis G.: Single photo estimation of hair appearance. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* (2009). 2

[Coo84] Cook R.: Shade trees. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 223–231. 1

[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. 2, 4

[DH95] Dunn D., Higgins W.: Optimal gabor filters for texture segmentation. *IEEE Transactions on Image Processing 4*, 7 (Jul 1995), 947–964. 4

[DTM96] Debevec P. E., Taylor C. J., Malik J. .: Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH 96* (1996), pp. 11–20. 2

[DvGNK99] Dana K., van Ginneken B., Nayar S., Koenderink J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (1999), 1–34. 2

[ELS08] Eisenacher C., Lefebvre S., Stamminger M.: Texture synthesis from photographs. *Computer Graphics Forum,* *Proceedings of the Eurographics conference 27*, 2 (2008), 419–428. 2

[EMP*98] EBERT D., MUSGRAVE K., PEACHEY P., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*. 1998. 2

[GDS10] GILET G., DISCHLER J., SOLER L.: Procedural descriptions of anisotropic noisy textures by example. In *Eurographics (short)* (2010). 2, 3, 7

[GZD08] GOLDBERG A., ZWICKER M., DURAND F.: Anisotropic noise. In *SIGGRAPH 2008* (New York, NY, USA, 2008), ACM, pp. 1–8. 2

[KFCO*07] KOPF J., FU C., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.: Solid texture synthesis from 2d exemplars. In *SIGGRAPH 2007* (2007). 2

[KK89] KAJIYA J., KAY T.: Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph. 23*, 3 (1989), 271–280. 1

[KPH*03] KNISS J., PREMOZE S., HANSEN C., SHIRLEY P., MCPHERSON A.: A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (2003), 150–162. 5

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3 (2003), 277–286. 7

[LLC*10] LAGAE A., LEFEBVRE S., COOK R., DEROSE T., DRETTAKIS G., EBERT D., LEWIS J., PERLIN K., ZWICKER M.: State of the art in procedural noise functions. In *EG 2010 - State of the Art Reports* (May 2010), Hauser H., Reinhard E., (Eds.), Eurographics, Eurographics Association. 2

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse gabor convolution. In *SIGGRAPH 2009* (New York, NY, USA, 2009), ACM, pp. 1–10. 2, 4

[LVLD09] LAGAE A., VANGORP P., LENAERTS T., DUTRÉ P.: *Isotropic stochastic procedural textures by example*. Tech. rep., May 2009. 2

[MGW01] MALZBENDER T., GELB D., WOLTERS H.: Polynomial texture maps. In *SIGGRAPH 2001: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 519–528. 2

[PCK*08] PARIS S., CHANG W., KOZHUSHNYAN O. I., JAROSZ W., MATUSIK W., ZWICKER M., DURAND F.: Hair photobooth: geometric and photometric acquisition of real hairstyles. In *SIGGRAPH 2008: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–9. 2

[Per85] PERLIN K.: An image synthesizer. In *Computer Graphics ACM Siggraph annual Conference Series* (1985), pp. 287–296. 1, 2

[Per02] PERLIN K.: Improving noise. *ACM Trans. Graph. 21*, 3 (2002), 681–682. 2

[PH89] PERLIN K., HOFFERT E.: Hypertexture. *ACM SIGGRAPH Computer Graphics 23*, 3 (July 1989), 253–262. 5

[RHS99] RANDEN T., HUSOY J., SCHLUMBERGER S.: Filtering for texture classification: a comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 4 (Apr 1999), 291–310. 4

[TFX*08] TAN P., FANG T., XIAO J., ZHAO P., QUAN L.: Single image tree modeling. In *SIGGRAPH Asia 2008: ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–7. 2

[Wor96] WORLEY S.: A cellular texture basis function. In *SIGGRAPH 96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM Press, pp. 291–294. 4

[WYZG09] WANG L., YU Y., ZHOU K., GUO B.: Example-based hair geometry synthesis. In *SIGGRAPH 2009: ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–9. 2