Geometric Details on Skeleton-based Implicit Surfaces

C. Zanni[1], P. Bares[1], A. Lagae[2], M. Quiblier[1], M.-P. Cani[1]

[1]Laboratoire Jean-Kutzmann, University of Grenoble & INRIA Grenoble - Rhône-Alpes, France, [2]Katholieke Universiteit Leuven, Belgium

Abstract

We present a modeling technique to enhance implicit surfaces with procedural geometric details. The details are based on Gabor noise, which enables us to seamlessly handle anisotropy. The orientation of details can be defined with respect to the orientation of the main shape features, which reduces the amount of user input. The method extends to complex details that are tilted with respect to the normal of the input surface. Lastly, our method allows the blending of the resulting enhanced implicit primitives without causing the details to blur.

1. Introduction

Implicit surfaces, defined as the set of points where $f(x,y,z) = c$, where f is a smooth scalar field and c an iso-value, have been popular for their constructive nature: two implicit shapes can be set to blend by simply summing their field functions. However these surfaces have long been restricted to the modeling of smooth, blobby-like shapes, since most methods for adding geometric details require some local parameterization. As stressed by Sherstyuk [She99], the two main ways to enhance implicit surfaces with geometric details are either to add small implicit primitive near the surface, or to deform the scalar field with volumetric noise such as hypertextures [PH89]. In the first case, in addition to the burden of adding many primitives by hand, small details may blur when blended, requiring the use of costly blending mechanisms instead of the sum [BBCW10]. In the second case, controlling the distribution and orientation of details on the surface would be difficult due to the volumetric nature of hypertextures, and small disconnected components could appear near the main surface.

In this paper we introduce the first extension of constructive implicit modeling to surfaces with procedural geometric details. Our method enables the addition of well-distributed anisotropic details over implicit primitives, provides an intuitive control of their orientation from the skeletons that define the primitives, and extends the blending properties of implicit surfaces without causing the details to blur.

2. Related work

Adding details to implicit surfaces: Although many methods are available to add detail to mesh-based surfaces (e.g., displacement mapping [Coo84, ADBA09] or shell mapping [PBFJ]), methods to add detail (or texture) to implicit surfaces are scarce due to the difficulty to add coherent surface detail when no parameterization is available. A solution for texturing implicit surfaces is to introduce a physical particle system driven by the gradient field to match points of the implicit surface with those of a textured support surface [ZGVdF]. The only method that requires no parametrization is *geometric texture synthesis by example* [BIT04], which extends example-based texture synthesis to volumetric objects defined by iso-surfaces of a discrete scalar field, stored in a voxel grid. Unfortunately, this method is compute-intensive, the complexity of details is limited by the resolution of the grid, and the method does not provide any mechanism enabling geometric texture to be smoothly interpolated when two textured surfaces blend.

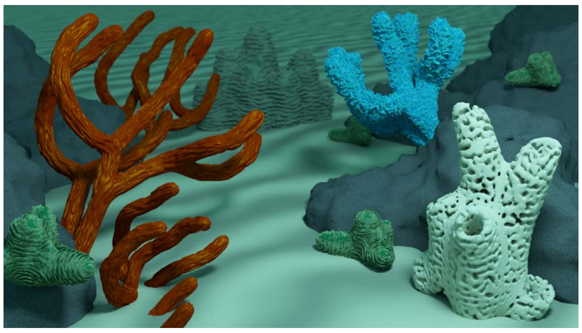


Figure 1: *Implicit modeling of a coral reef. All small scale detail are added with our method.*

Skeleton-based implicit surfaces: Constructive implicit modeling makes use of skeletons (such as points, segments

DOI: 10.2312/conf/EG2012/short/049-052

or triangles) for generating the field function that defines an implicit primitive. Convolution surfaces, introduced in [BS], are defined by integrating a decreasing kernel function $K : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ along the skeleton Sk:

$$f(P) = \int_{s \in Sk} K(d_{Sk}(s,P))\,\mathrm{d}s \; . \tag{1}$$

In the paper, we use this family of primitives, although our method is applicable to standart distance surfaces as well. Note that we use at least C^1 scalar fields, leading to continuously varying normal vectors $\mathbf{N}(P) = -\frac{\nabla f(P)}{\|\nabla f(P)\|}$.

Anisotropic surface Gabor noise: Our method for generating details on implicit surfaces is based on Gabor noise [LLDD09]. The most basic form of Gabor noise, 2D anisotropic Gabor noise, is defined as

$$n(\mathbf{x};K,a,\omega) = \sum_i w_i g(\mathbf{x} - \mathbf{x_i};K,a,\omega)\,, \tag{2}$$

where the noise parameters K, a and ω control the amplitude, bandwidth and frequency of the noise. The random weights $\{w_i\}$ are distributed according to a standard uniform distribution, g is the Gabor kernel, and the random positions $\{\mathbf{x_i}\}$ are distributed according to a Poisson process with mean λ. We use surface Gabor noise, a generalization of 2D Gabor noise to surfaces. Surface Gabor noise is independent of the geometric representation of the surface. Evaluating it only requires reference frames to be defined over the surface to express the direction of anisotropy.

3. Procedural noise on skeleton-based implicit surfaces

Let $\mathcal{S}$ be the implicit surface to be enhanced with details and $f_{\mathcal{S}}$ be the scalar field that defines it. $f_{\mathcal{S}}$ is generated thanks to a set of skeletons $Sk = \{Sk_i\}$ generating individual fields f_i. This section explains how we define a smooth, anisotropic Gabor noise on $\mathcal{S}$ that seamlessly follows its geometry.

Smooth vector field: Generating a surface Gabor noise requires the definition of continuously varying local frames over the surface. To reduce user input, we provide a method for automatically generating them from a meaningful spatial vector field, defined as a weighed sum of the skeleton directions. Since the skeletons control the shape of $\mathcal{S}$, this ensures that the frames will coherently follow the surface.

Let us develop the method when the skeleton $\{Sk\}$ defining $\mathcal{S}$ is a set of segments. Similarly to the way smooth field values are computed from skeletons in convolution surfaces, we generate a smoothly varying vector field by integrating the directions $\mathbf{Sk}(s)$ of the skeleton, weighted by the field contribution of the associated infinitesimal arc length:

$$\mathbf{Wd_{Sk}}(P) = \int_{s \in Sk} K(d_{Sk}(s,P)).\mathbf{Sk}(s)\,\mathrm{d}s. \tag{3}$$

The additivity of the integral leads to an additive property for the resulting vector field:

$$\mathbf{Wd_{Sk}}(P) = \sum_i \mathbf{Wd_{Sk}}_i(P) \tag{4}$$

Moreover, the direction $\mathbf{Sk}_i$ being constant on a single segment, equation (3) combined with (1), yields:

$$\mathbf{Wd_{Sk}}_i(P) = f_i(P).\mathbf{Sk}_i \tag{5}$$

This results into a continuous space vector field $\mathbf{Wd}_{\mathcal{S}}(P)$ that encodes the shape of $\mathcal{S}$, as depicted on Figure 2(a).

We extend the definition of $\mathbf{Wd}$ to all kinds of implicit primitives by using equation (4) and (5), replacing $\mathbf{Sk}$ by $\mathbf{Sk}(P)$, a meaningful direction which may vary in space, defined from the primitive's skeleton. For instance, for triangles we use any vector included in its defining plane; note that the orientation as well as the direction defined for each skeleton affects the resulting vector field.

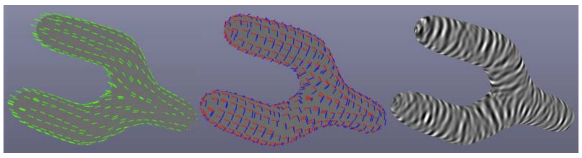

(a) (b) (c)

Figure 2: *(a) weighted skeleton direction, (b) procedural frames, (c) resulting anisotropic Gabor surface noise.*

The implicit primitive is now associated with two smooth vector fields: the field gradient (defining the normal $\mathbf{N}$) and the weighted direction $\mathbf{Wd}_{\mathcal{S}}$ we just defined. We use them for computing smoothly varying local frames $(\mathbf{n},\mathbf{t},\mathbf{b})$ (normal, tangent, bitangent) over the surface by setting:

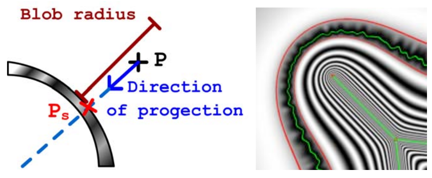
$$\{\mathbf{n} = \mathbf{N}(P),\; \mathbf{b} = normalized\,(\mathbf{n} \wedge \mathbf{Wd}_{\mathcal{S}}(P))\,,\; \mathbf{t} = \mathbf{n} \wedge \mathbf{b}\}$$

The resulting local frames are depicted on Figure 2(b); in the case of tubular shapes, $\mathbf{t}$ tends to be in the same direction as the weighted direction. In practice, we generate the frames efficiently using closed-form formula for convolution field gradients, derived from closed form-formula for field function [HC11]. Note that our frame field is not only coherently defined on the surface, but also in the surrounding space; we will use this for another purpose in section 4.

Anisotropic surface noise: Once the frames are defined, they are used to generate surface Gabor noise as explained in Section 2.2 (equation 2). The parameters defining the noise are the noise magnitude K, the bandwidth a, and the orientation with respect to the local frame ω. For instance, we can define noise that follows the main orientation of the surface, that is orthogonal to it or that progressively swirls around it. Such anisotropic Gabor surface noise is depicted on Figure 2(c).

4. Generating geometric details from surface noise

We now explain how we generate geometric details over an implicit surface from the noise we just defined. We assume that the details to be added are small compared to the geometric features of the initial shape, i.e. that there is enough room to add them in concave regions.



(a) (b)

Figure 3: *Principle of the method: (a) projection on surface to get noise value, (b) in red: iso-surfaces $f_{\mathcal{S}} = iso_d^-$ and $f_{\mathcal{S}} = iso_d^+$ defining the deformation region.*

Basic method: The goal is to use the noise for locally deforming the scalar field that defines the input implicit surface. To save computational time, we only apply this deformation in a local neighborhood of the iso-surface of interest. Let the interval $[iso_d^-; iso_d^+]$ define the region, chosen from the size of details, where the field is to be deformed (figure 3).

As we would have done with displacement mapping, the noise value on the surface will be used to quantify the deformation. Thus, to deform the field around the surface, we must associate to each point in space a point on the surface, in order to get the amount of deformation.

To compute this association in a coherent way, we project the evaluation point P on the surface along stream-lines of the field, defined by following the field gradient; this was also used in [ZGVdF], but here the aim is to obtain a space/surface mapping. This gives us both a point $P_{\mathcal{S}}$ on the surface (with the associated noise value) and a distance to it. The scalar field $f_{\mathcal{S}}(P)$ of the input surface is then modulated as if there was a blob centered at $P_{\mathcal{S}}$. The blob intensity is set from the noise value.

Note that projecting space points along streamlines insures that the resulting details remain coherent in concave areas (neighboring details do not intersect nor blend). Moreover, the method can be used both for adding material and for carving the input surface, depending on the positive vs. negative value of the noise.

Generating free-form and tilted details: Our method provides easy ways to control the shape and orientation of details, since each step of the algorithm above can be adequately parameterized. The algorithm for computing the new field value $f(P)$ is thus:

$$\textit{If } f_{\mathcal{S}}(P) \notin [iso_d^-; iso_d^+] \textit{ then } f(P) = f_{\mathcal{S}}(P).$$

$$\textit{Else:} \quad \left[\begin{array}{l} \textit{let } P_{\mathcal{S}} = proj(P, \mathbf{v}_{\alpha,\theta}), \\ \textit{let } noise = n\left(P_{\mathcal{S}}; k, a, \omega\right), \\ f(P) = f_{\mathcal{S}}(P) + D(noise).K\left(\frac{\|PP_{\mathcal{S}}\|}{R}\right). \end{array} \right.$$

where (k, a, ω) are the noise parameter on the surface, $\mathbf{v}_{\alpha,\theta}$ is the direction of projection, D is a map that gives the amount of deformation from the noise value and R is the desired height of details. We use $K(r) = (1-r)^6$ to define the blob's contribution. $\mathbf{v}_{\alpha,\theta}$ is defined within the frame computed in Section 3. Using arbitrary direction instead projecting along streamlines enables us to tilt geometric details with respect to the normal of the main surface, leading to the generation of details that are not mere high fields (see figure 4).

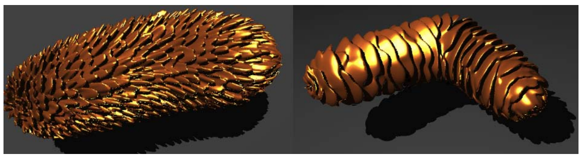


Figure 4: *Effect of varying the direction of projection.*

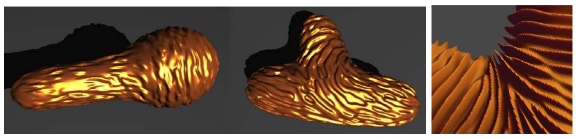
Blending implicit surfaces carrying details: The parameters defining geometric details can be set to vary from an implicit primitive to the next. During blending, these parameters are smoothly interpolated according to the respective field contribution of each primitive at the evaluation point, or at the projected point for noise parameters:

$$param(P) = \frac{\sum_i f_i(P).param_i(P)}{\sum_i f_i(P))}.$$

This leads to a smooth change of detail orientation and shape in blending regions.

5. Results

As our results show, the method generates details that behave nicely in concave regions (Figure 5(c)), instead of self-colliding as when displacement mapping is used. Moreover, when implicit surfaces blend, the noise values are smoothly interpolated, resulting in a natural behavior of details; when details have different orientations on the initial surfaces, their orientation smoothly varies in the blended regions of the resulting shape (Figure 5(a,b)). Note that details variation depends on the range of influence of the fields: for instance, the T-junction in Figure 5(b) affects the orientation of details on the other side of the main cylinder, which would not be the case using sharper field functions.



(a) (b) (c)

Figure 5: *(a,b,c) Smooth changes in details direction.*

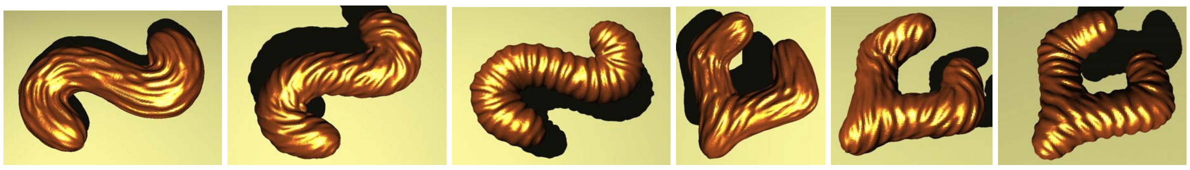


Figure 6: *Example emphasizing the variety of choices that can be made for details orientation, all coherent with input shape.*

	Triangles in Meshes	Computation Times
Coral reef - (average per mesh)	326 920	102.3s
Dragon (figure 7)	226 828	88.4s
Figure 4 - (average per mesh)	285 622	90.8s
Figure 5(a,b) - (average per mesh)	125 174	9.7s

Table 1: *Computation time on a 2.4 GHz Intel Core 2 Duo (only one core used).*

Limitations: On the negative side, the way shape parameters are interpolated still needs to be improved for enabling the blend of implicit surfaces with details of different sizes and tilt. Moreover, the complexity of the detail we can generate is limited by the range of patterns that can be obtained through Gabor noise. Lastly, generating frame field without any singularity is not possible on some surfaces. In our case, singularities arise when $\mathbf{Wd}_{\mathcal{S}}(P)$ and $\nabla f_{\mathcal{S}}(P)$ are collinear. Fortunately, computing the scalar product between these two directions gives us a distance to singular point, enabling us to locally cancel the noise. A solution would be to use isotropic surface noise in this region, which can handle singularities.

Computational efficiency: Timings are given in Table 1; due to the fact that migration along stream-lines can be computationally expensive, we replace it by a projection in the initial gradient direction. From our experiments, this solution is more efficient and still leads to good results. There is room for a lot of improvements in the performances of our method, from an efficient caching system to the derivation of closed-form equations for the gradient of the part of the field that represents details.

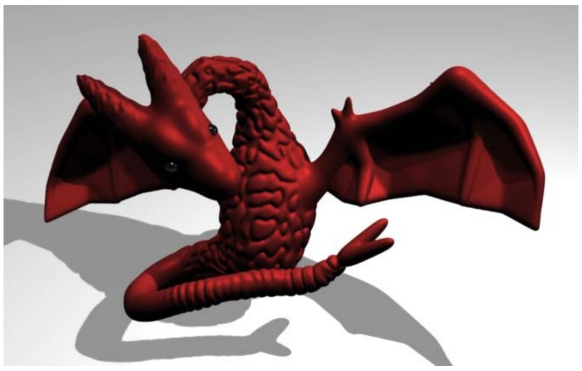


Figure 7: *Dragon model showing the variety of details that can be generated. Computation time was less than 2 minutes.*

6. Conclusion

We have presented one of the first methods for extending constructive implicit modeling to surfaces with geometric details, based on Gabor noise. Not that our method can be adapted to non-skeleton-based primitives, provided that one or several reference axis is set to define their global orientation. In addition to solving the problems listed in the discussion of results, our future work will focus on improving the way details are defined: First, sketching the profile of details should be a fast and intuitive way to set their parameters. Secondly, detail parameters could be stored along the skeleton, which would enable them to vary over individual implicit primitives.

References

[ADBA09] Andersen V., Desbrun M., Bærentzen J. A., Aanæs H.: Height and tilt geometric texture. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I* (2009), ISVC '09, Springer-Verlag, pp. 656–667.

[BBCW10] Bernhardt A., Barthe L., Cani M.-P., Wyvill B.: Implicit blending revisited. *Comput. Graph. Forum 29*, 2 (May 2010), 367–375.

[BIT04] Bhat P., Ingram S., Turk G.: Geometric texture synthesis by example. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), SGP '04, ACM, pp. 41–44.

[BS] Bloomenthal J., Shoemake K.: Convolution surfaces. In *Proceedings SIGGRAPH '91*, ACM, pp. 251–256.

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

[HC11] Hubert E., Cani M.-P.: Convolution surfaces based on polygonal curve skeletons. *Journal of Symbolic Computation* (2011). http://hal.inria.fr/inria-00429358.

[LLDD09] Lagae A., Lefebvre S., Drettakis G., Dutré P.: Procedural noise using sparse Gabor convolution. *ACM Trans. Graphics 28*, 3 (2009), 54:1–54:10.

[PBFJ] Porumbescu S. D., Budge B., Feng L., Joy K. I.: Shell maps. In *ACM SIGGRAPH 2005 Papers*, ACM, pp. 626–633.

[PH89] Perlin K., Hoffert E. M.: Hypertexture. *SIGGRAPH Comput. Graph. 23* (July 1989), 253–262.

[She99] Sherstyuk A.: Interactive shape design with convolution surfaces. In *Proceedings of the International Conference on Shape Modeling and Applications* (1999), IEEE Computer Society, pp. 56–.

[ZGVdF] Zonenschein R., Gomes J., Velho L., de Figueiredo L. H.: Texturing implicit surfaces with particle systems. In *Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '97*, ACM, p. 172.