Geometric Texture Modeling

Gershon Elber
Technion

With the invention of texture mapping in computer graphics, the photorealism level of synthesized geometry has taken a giant leap forward.[1,2] Texture mapping has become an essential tool in any synthetic rendering scheme that aims at photorealism. Texture mapping typically associates any point on the surface of the rendered object with a location in the texture space. The surface point is then assigned rendering attributes, such as color or translucency from the respective location found in the texture space. Texture mapping techniques have also been used in attempts to emulate highly detailed geometry on the surfaces of objects. This article focuses primarily on texturing techniques that relate to shape modeling and surface geometry alteration.

The proposed surface detail synthesis approach closes the loop from modeling to rendering using computer graphics texture mapping techniques, bringing them back into the geometric modeling phase.

The bump-mapping technique was the first successful attempt to emulate 3D geometric surface details by modulating the normals of the rendered objects' vertices.[2,3] At each surface location, you assign a small perturbation to the normal, based on some mapping from that surface location to some normal perturbation texture function. The bump-mapping technique is highly successful in conveying a bumpy shape while the geometry remains smooth. The illusion created by bump mapping is quite convincing. Nevertheless, in the silhouette areas of the object in hand, the surface's outline remains smooth. No casting of shadows of rough silhouette edges due to this bumpiness can occur, since the geometry itself is not modified.

The displacement maps method took a major step in synthesizing real 3D detailed geometry by allowing actual surface modifications.[2] It's common for this mapping scheme to be represented as a height field that modulates the amount that each surface location is elevated in one direction—typically the surface normal direction. Let $S(u, v)$ be the surface to displace and let $\bar{n}(u, v)$ be its unit normal field. Then, given a displacement as a scalar height field, $d(u, v)$, the geometry of the new, modulated surface equals

$$S_d(u,v) = S(u,v) + \bar{n}(u,v)d(u,v) \quad (1)$$

The computation or the approximation of the unit normal field of $S(u, v)$ can be quite involved and is, in general, considered a computationally complex task that hinders the use of such techniques in real time. The partial derivatives of $S(u, v)$ span the tangent plane of S, if regular. Therefore, $\partial S/\partial u$, $\partial S/\partial v$, $\bar{n}$ can serve as a basis for $\mathbb{R}^3$ and be used to prescribe a displacement in an arbitrary 3D direction.

This work extends the notion of displacement maps and relaxes several of their constraints for use in texture modeling. Actual and precise geometry can now be synthesized for arbitrary further processing as opposed to rendering only. I reconstruct the object using a geometry that completely and accurately captures the desired surface details. The approach proposed here for surface detail synthesis closes the loop from modeling to rendering by drawing on the computer graphics texture mapping techniques and bringing them back into the geometric modeling phase. Numerous objects with precise surface details that are difficult to model using traditional geometric modeling schemes and contemporary modeling tools could be easily constructed using the proposed geometric texture modeling scheme. This method supports both the free-form NURBS domain as well as the polygonal mesh surface representation.

The graphics group at Technion (referred to as "we" in this article) employs nonlinear trivariate polynomial functions to derive the texture mapping function and parameterize the space around the object surfaces. While the concepts presented in this work are not constrained to this specific mapping function, this representation was selected since it allows us to fully and precisely capture the original surface geometry as well as guarantees continuous normals. The presented approach employs the same basic tool used in free-form deformations (FFD),[4] and hence the resulting texture undergoes a displacement transformation as well as a deformation that coerces it to follow the underlying surface's precise shape. Consequently, any geometry in $\mathbb{R}^3$ can be employed as supplementary detail texture, making the texture modeling phase as general and as precise as needed. Moreover, because any object can serve as the geometric texture map of itself, a complete closure is formed. The "Background and Related Work" sidebar discusses other research.

Modeling the geometric texture

Consider an object $\mathcal{O}$ and let S be one surface on the boundary of $\mathcal{O}$ with regular parameterization $S(u, v)$, $u, v \in [0, 1]$. Denote $S(u, v)$ as the base surface, the surface on which we are to place the detailed texture geometry. Further, let $\bar{n}(u, v)$ be the unit normal field of $S(u, v)$

 Published by the IEEE Computer Society 0272-1716/05/$20.00 © 2005 IEEE

Background and Related Work

A first attempt to capture the volume around the surface of an object was presented in Kajiya and Kay,[1] who introduced the concept of texels. They wrote, "a texel is a three dimensional array that holds the visual properties of a collection of micro-surfaces." Geared toward ray tracing only, and not actual surface modeling, the geometry is replaced by its visual properties, gathering the scattering and reflectance functions. Neyret also used trilinear texels.[2] Due to the texel's linearity, one surface location is assigned one displacement direction. Texels are continuous, but their normals are not. Similarly, because the base of the texel is bilinear, it's impossible to precisely follow nonlinear polynomial surfaces. One advantage of the approach used by Kajiya and Kay and Neyret stems from its ability to ray trace the scene without the need to duplicate the geometry behind the texture function. Being able to invert the mapping function, the casted ray is remapped back into a canonical texel. This texel approach can also handle texture mapping that is one to many. That is, a single surface location can be mapped to several points above the surface.

Three-dimensional supplementary details can also be added to surfaces using other means. Typically, such texturing processes can be divided into two stages. The first, the texture placement phase, determines the locations at which the texture supplementary elements are to be placed, typically using some notion of surface parameterization. The second, texture modeling phase, locally molds the texture elements to fit their final shape.

Fleischer et al. attempted to create 3D details, such as scales or thorns, over the surface.[3] They also simulated natural cellular development for use in texture generators for the placement phase of the individual elements. The texture modeling phase in this work exploited a geometric modeler that is parametric. Cellular texture generators as a tool for texture placement have received significant attention—for example, brick textures were investigated for architectural models.

The question of texture placement is of major concern in irregular polygonal meshes of arbitrary topology, as is the issue of seamless tiling of a repeatable texture over these domains. These concerns have been the focus of many researchers in recent years. For example, prescribed vector fields over the geometry were used to achieve this proper placement. Others derived a parameterization and local tangents for each vertex of the mesh, attempting to reach the same goal. Simulation of reaction-diffusion is another scheme to distribute texture elements for surfaces with no parameterization. Methods to minimize distortion, in a conformal or isometric sense, were also investigated in recent years—for example by Sheffer and de Sturler.[4]

Trivariate functions were introduced to the graphics community in the much cited work of Sederberg and Parry on warping applications.[5] They employed trivariate functions as mappings for $\mathcal{T}:\mathbb{R}^3 \Rightarrow \mathbb{R}^3$. They represented bending, stretching, and warping operators using trivariate functions that bent, stretched, or warped a subspace of $\mathbb{R}^3$. The work of Sederberg and Parry, also known as free-form deformations (FFD), has many derivations such as the extended FFD, and nontensor product FFD representations.[6] All these derivations share the ability to embed a given object in the parametric domain of the trivariate, and that object undergoes the same nonlinear transformation along with the entire subspace around the object.

While an exceptionally general and powerful object modification operator, the difficult question has always been how to intuitively derive the proper trivariate function to perform a certain warping operation. The proposed approach employs the trivariate functions' powerful capabilities as a warping tool, while automatically defining the trivariate function to follow the 3D domain above the surface boundary of object $\mathcal{O}$.

The presented method assumes that the surface to which you add details has a readily available proper parameterization, by being a polynomial, a rational parametric surface or a polygonal mesh with parameterization. Other works are samples of the state of the art in this area of texture placement, which provide proper distributions of the texture elements on the surface, be they a square textured tile, a thorn, a brick, or hair samples.[1,3,4] To achieve the proposed geometric texture modeling, you can employ any of the proposed texture distribution or parameterization techniques and augment them by providing extended and unifying texture modeling capabilities. Once you have a proper texture placement, the texture modeling function can fully and smoothly encompass the third dimension above the surface. That is, given a surface location along which to place the texture, the location can be mapped to a continuum above and/or below the surface. Moreover, by having a complete and continuous parameterization of the surface domain and above and/or below it, you can fully adapt the displaced geometric texture to the shape of the surface as well as offer an efficient estimation scheme for the normals of the deformed and displaced geometry.

References

1. J.T. Kajiya and T.L. Kay, "Rendering Fur with Three Dimensional Textures," *Proc. ACM Siggraph*, ACM Press, 1989, pp. 271-280.
2. F. Neyret, "Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures," *IEEE Trans. Visualization and Computer Graphics*, vol. 4, no. 1, 1998, pp. 55-70.
3. K. Fleischer et al., "Cellular Texture Generation," *Proc. ACM Siggraph*, ACM Press, 1995, pp. 239-248.
4. A. Sheffer and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening," *Eng. with Computers*, vol. 17, no. 3, Springer, 2001, pp. 326-337.
5. T.W. Sederberg and S.R. Parry, "Free-Form Deformation of Solid Geometric Models," *Proc. ACM Siggraph*, vol. 20, ACM Press, 1986, pp. 151-160.
6. S. Coquillart, "Extended Free-Form Deformations: A Sculpting Tool for 3D Geometric Modeling," *Proc. ACM Siggraph*, ACM Press, 1990, pp. 187-196.

pointing outside of $\mathcal{O}$. Having a continuous normal field constitutes the requirement for S to be C^1. This constraint is commonly satisfied by the polynomial and/or the rational representations that govern contemporary geomet-

Unit Normal Field Approximation

Assume surface $S(u, v)$ is a polynomial or a rational free-form surface. Unfortunately, the unit normal field

$$\bar{n}(u,v) = \frac{\frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}}{\left\| \frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v} \right\|}$$

is not rational due to the normalization that is imposed in the denominator of $\bar{n}$. One simple way of approximating a unit size normal field would be to normalize all control points in the normal field of

$$n(u,v) = \frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v}$$

The result of this approximation affects both the magnitude and the direction of the normal field.

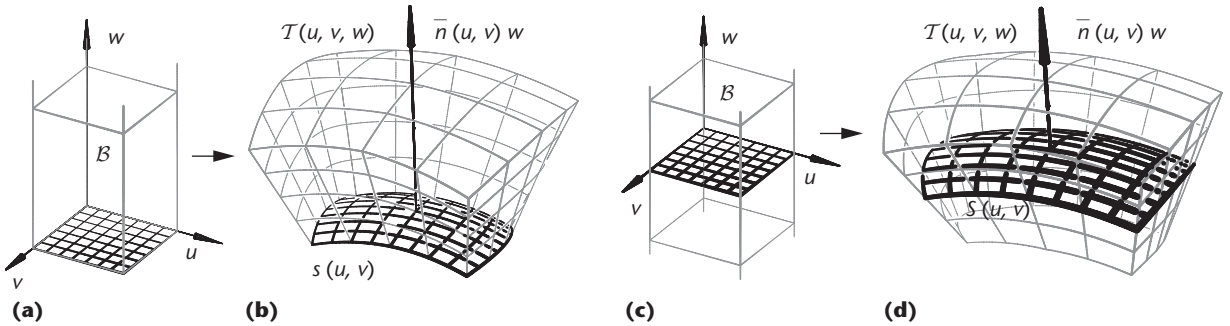
Alternatively, you can symbolically multiply $n(u, v)$ by a function that approximates the inverse magnitude of n

$$m(u,v) \approx \frac{1}{\left\| \frac{\partial S(u,v)}{\partial u} \times \frac{\partial S(u,v)}{\partial v} \right\|}$$

Now $n(u, v)m(u, v)$ approximates the unit magnitude of $\bar{n}(u, v)$, while it fully preserves the directional information of the field. This product can be computed via products of Bezier or B-spline basis functions.[1] By applying refinement[2] to $n(u, v)$, you can improve the approximation of m and, in fact, converge as closely as needed to a unit size normal field with precise directions.

References

1. G. Elber, "Symbolic and Numeric Computation in Curve Interrogation," *Computer Graphics Forum*, vol. 14, no.1, 1995, pp. 25-34.
2. E. Cohen, R.F. Riesenfeld, and G. Elber, *Geometric Modeling with Splines: An Introduction*, A K Peters, 2001.



1 (a) Parametric domain used to define (b) the trivariate $\mathcal{T}(u, v, w)$ function above surface $S(u, v)$. (c) The parametric domain of $\mathcal{T}$ can also be mapped to be (d) both below and above surface S.

ric modeling tools. The "Unit Normal Field Approximation" sidebar prescribes schemes to approximate a polynomial or rational unit normal field to a given polynomial or rational surface.

In the discrete, piecewise-linear domain, parameterizations of polygonal meshes are also readily available, with a large body of recent research work on this topic, for example Sheffer and de Sturler.[5] As a result, continuous unit normals over the polygonal discrete domain can be approximated, much like the classic Phong shading scheme,[2] by interpolating the normals of the vertices with the aid of barycentric weighing coordinates in each triangle.

Having a parameterization for S and a continuous unit normal field $\bar{n}$ define

$$\mathcal{T}(u,v,w) = S(u,v) + \bar{n}(u,v)w \tag{2}$$

a trivariate function above S for $w > 0$ (see Figures 1a and 1b). The 3D parametric space of $\mathcal{T}$ is the infinite box $\mathcal{B}$: $(u, v, w), u, v \in [0, 1]$, and arbitrary $w \in \mathbb{R}^+$. $\mathcal{T}(u, v, w)$ parameterizes the volumetric neighborhood of $S(u, v)$ and equals S exactly for $w = 0$. Similarly, $\mathcal{T}(u, v, w)$ could also be defined below surface $S(u, v)$, having $w \in \mathbb{R}$ (see Figures 1c and 1d).

The notion of above or below the base surface is hence defined as

Definition 1: A point P is above base surface $S(u, v)$ if there exists (u_0, v_0, w_0), $w_0 > 0$, such that $P = \mathcal{T}(u_0, v_0, w_0)$. $S(u_0, v_0)$ is the support position of P.

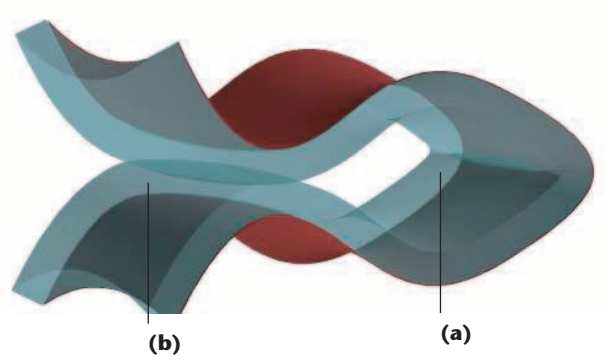
Similarly,

Definition 2: A point P is below base surface $S(u, v)$ if there exists (u_0, v_0, w_0), $w_0 < 0$, such that $P = \mathcal{T}(u_0, v_0, w_0)$. $S(u_0, v_0)$ is the support position of P.

The mapping function of $\mathcal{T}: \mathbb{R}^3 \Rightarrow \mathbb{R}^3$ can self-intersect, both locally due to highly curved regions in S, or globally when two unrelated but close regions of S interact in $\mathcal{T}$ (see Figure 2). Local self-intersections can be identified by the singularity event of the vanishing Jacobian of $\mathcal{T}$. Gain and Dodgson employed this

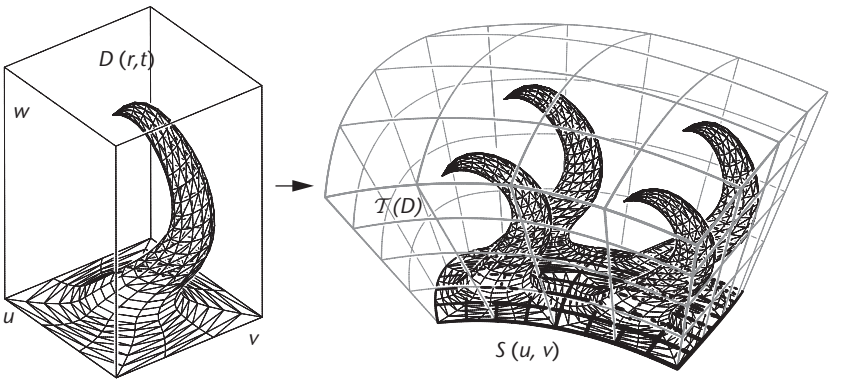
constraint on the Jacobian to detect local singular conditions in FFDs.[6] They used the zero set of the determinant of the Jacobian of the trivariate function to derive and extract the sweep surfaces' boundary, using envelope theory. The possibility of self-intersections in $\mathcal{T}$ is due to point P possibly being above one support position of the base surface and below another. Alternatively, P can be above (or below) two (or more) different support positions.

The mapping $\mathcal{T}$ is well defined while its inverse might not be. Hence, the texture construction is well defined as well though it could end up with self-intersections. This article is mainly concerned with the 3D border layer, or the volume near the base surface, for small values of w. Therefore, if the base surface is smooth and self-intersection free, local and global self-intersections in $\mathcal{T}$ are expected to be rare. Moreover and as already stated, local self-intersections could be identified by examining the Jacobian of $\mathcal{T}$ whereas global self-intersections could be examined by testing for self-intersections in the surface S offsets.



2 Trivariate function $\mathcal{T}(u, v, w)$ (in cyan) can present (a) a local self intersection due to a highly curved region in S (in red), or (b) global self intersections when two unrelated regions of S are too close.

Recall Equation 1. The self-intersection problem also existed in traditional displacement mapping whereas here, with the aid of the Jacobian's determinant analysis, we offer a method to detect such cases. Bivariate displacement maps are, traditionally, mappings from $\mathbb{R}^2$ to $\mathbb{R}$, possibly serving as modulators on the magnitude of a unit vector field such as the base surface's normal. As Figure 3 shows, consider a bivariate parametric displacement texture function $D(r, t)$: $\mathbb{R}^2 \Rightarrow \mathbb{R}^3$, $D(r, t) = (u(r, t), v(r, t), w(r, t))$. The embedding of D in the parametric domain of $\mathcal{T}$, $\mathcal{B}$, yields



3 Mapping of the geometric texture $D(r, t)$ (a bent thorn) using the trivariate function $\mathcal{T}(u, v, w)$ as four thorn tiles in a 2×2 grid over surface S.

$$\begin{aligned}\mathcal{T}(D) &= \mathcal{T}\left(u(r,t), v(r,t) w(r,t)\right) \\ &= S\left(u(r,t), v(r,t)\right) + \bar{n}\left(u(r,t), v(r,t)\right) w(r,t)\end{aligned} \tag{3}$$

For the special case where $u = r$ and $v = t$, $D(r, t)$ is reduced to the traditional displacement mapping technique, becoming an explicit height field above the $(r, t) = (u, v)$ plane. In this restricted case, Equation 3 assumes the form of

$$\begin{aligned}\mathcal{T}(D) &= \mathcal{T}\left(r, t, w(r,t)\right) \\ &= S(r,t) + \bar{n}(r,t) w(r,t)\end{aligned}$$

(compare with Equation 1), while $w(r, t)$ serves as the height field displacement along the direction of the normal. Hence, the traditional displacement mapping technique is a special, explicit case of the parametric form presented in Equation 3. The additional benefits that you might gain by this more general parametric displacement representation are the main interests of this article.

Two types of geometric textures maps, $D(r, t)$, can now be employed, types we denote as *covering texture* maps and *casual texture* maps:

Definition 3: $D(r, t) = (u(r, t), v(r, t), w(r, t))$, $r, t \in [0, 1]$ is considered a covering geometric texture map if $\forall u_0, v_0 \in [0, 1]$, $\exists r_0, t_0 \in [0, 1]$ such that $u(r_0, t_0) = u_0$, $v(r_0, t_0) = v_0$.

In other words, any geometric texture map that is on, and hence spans, the entire surface is a covering texture map. An obvious example of a covering map is the traditional displacement maps. A covering texture map can be discontinuous in $w(r, t)$ and hence still preserve gaps in its range.

Definition 4: $D(r, t)$, $r, t \in [0, 1]$ is considered a casual geometric texture map if it's not a covering geometric texture map.

If $D(r, t)$ is not a covering geometric texture map, typically the base surface will not be replaced by the texture map, but will merely be augmented by it. That is, the texture map serves supplementary purposes only. An obvious example of a casual texture is hair or fur.

Polynomial/Rational Composition Algorithm

Let $D(r, t) = (u(r, t), v(r, t), w(r, t)) \subset \mathcal{T}$ be a polynomial parametric geometric texture tile. Having a polynomial representation to $\mathcal{T}$, $\mathcal{T}(D(r, t))$ could be precisely evaluated into a composed surface. Here, we present the composition process for the polynomial domain whereas the extension to rationals is simple.[1] Without loss of generality, assume S, D, and $\mathcal{T}$ are all in the Bezier representation. Then, the trivariate mapping equals

$$\mathcal{T}(D) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v}\sum_{k=0}^{n_w} P_{ijk} B_{i,n_u}(u(r,t)) B_{j,n_v}(v(r,t)) B_{k,n_w}(w(r,t))$$

where

$$B_{i,n_u}(u) = \binom{n_u}{i}(1-u)^{n_u-i}u^i$$

are the Bezier basis functions of degree n_u. Further let

$$u(r,t) = \sum_{p=0}^{n_p}\sum_{q=0}^{n_q} u_{pq} B_{p,n_p}(r) B_{q,n_q}(t)$$

Substituting in, we have

$$\begin{aligned} & B_{i,n_u}(u(r,t)) \\ &= B_{i,n_u}\left(\sum_{p=0}^{n_p}\sum_{q=0}^{n_q} u_{pq} B_{p,n_p}(r) B_{q,n_q}(t)\right) \\ &= \binom{n_u}{i}\left(1-\sum_{p=0}^{n_p}\sum_{q=0}^{n_q} u_{pq} B_{p,n_p}(r) B_{q,n_q}(t)\right)^{n_u-i} \\ &\quad \left(\sum_{p=0}^{n_p}\sum_{q=0}^{n_q} u_{pq} B_{p,n_p}(r) B_{q,n_q}(t)\right)^{i} \end{aligned}$$

Hence, the composition of $\mathcal{T}(D)$ in the Bezier domain was reduced to sums and products of Bezier basis functions. See Cohen, Riesenfeld, and Elber[1], and Elber[2] on the computation of these two operations. This forward evaluation process of $\mathcal{T}(D)$ circumvents the possible difficulties in handling the singularities in the mapping function. Hence, you can conduct this computation as a simple forward composition evaluation.

The composition of B-spline basis functions can be similarly derived. Only now the sums and products of B-spline basis functions need to be computed; see Cohen et al. for more information.[1]

References

1. E. Cohen, R.F. Riesenfeld, and G. Elber, *Geometric Modeling with Splines: An Introduction*, A K Peters, 2001.
2. G. Elber, "Symbolic and Numeric Computation in Curve Interrogation," *Computer Graphics Forum*, vol. 14, no.1, 1995, pp. 25-34.

Definition 5: Consider a C^0 continuous $D(r, t)$, $r, t \in [0, 1]$. If $D(1, t) = D(0, t) + \mathbf{t}_u$ and $D(r, 1) = D(r, 0) + \mathbf{t}_v$, such that $\mathbf{t}_u$ and $\mathbf{t}_v$ are translation vectors in the XY plane, then $D(r, t)$ is a C^0 continuous periodic geometric texture map, with a period of $(\mathbf{t}_u \times \mathbf{t}_v)$.

A periodic geometric texture map can seamlessly tile the base surface. Higher continuity constraints, such as tangent plane continuity, could be imposed as well, providing a G^k continuous periodic geometric texture map.

The advantage of using periodic geometric texture maps can be found in the smaller size of the representation. Because we expect the final geometry to be large, the benefits are evident. We only have to express details in the surface once and thereafter can duplicate and tile these geometric details as necessary. For example, tiles that are far from the viewing position could be approximated in a low resolution whereas tiles in hidden surface regions can be purged altogether. Similarly, when contouring or slicing the final object, possibly toward layered manufacturing processes, only geometric tiles near the current contouring level need to be generated and processed.

Clearly, both covering and casual texture maps could be made periodic, with the advantage that periodic covering texture maps can replace the original surface. Here, you can find another advantage in using trivariate functions as geometric detailing tools. If $S(u, v)$ and $D(r, t)$ are C^k continuous, $\mathcal{T}(D)$ is C^{k-1} continuous, because $\mathcal{T}$ employs first-order derivatives of S. In other words, the continuity of the final geometry is completely governed by the base surface, S, and its (tiled) texture, D. For a tiled texture, D must be a G^k continuous periodic geometric texture map.

The "Polynomial/Rational Composition Algorithm" sidebar provides a short review of the necessary composition computation steps of $\mathcal{T}(D)$, in the (piecewise) polynomial and/or rational domains. Similarly, the "Polygonal Composition Algorithm" sidebar presents the necessary computation steps of this composition when D and/or S are the surfaces of polygonal meshes.

With this capability to map both polygonal and spline geometry through $\mathcal{T}$, D can be formed out of a polygonal mesh (2-manifold or not), or a whole polygonal model like the many models available on the Web. Alternatively, D could be a single NURBS surface or a spline geometric model formed from a trimmed NURBS surface set, as many contemporary geometric modeling methods generate.

Given a geometry that undergoes some nonlinear transformation $\mathcal{T}$, the normal at some vertex could be either mapped directly using $\mathcal{T}$ if possible, or approximated as the average of the normals of the mapped polygons sharing this vertex. While clearly feasible, such normal averaging could be time consuming. For reasons that we will reveal shortly, the direct normal estimation using the mapping $\mathcal{T}$ is impossible, in general—a problem shared by most FFD mapping schemes. Hence, this article seeks the conditions under which an approximation could still be derived from the original normals by mapping these normals directly through $\mathcal{T}$.

A mapping $\mathcal{T}$ is called *conformal* if it preserves

Polygonal Composition Algorithm

In many cases, the provided geometry is not readily available in the polynomial or rational forms, and therefore, we also seek an alternative that approximates this composition for polygonal geometry.

Let D be a polygonal geometric texture. You can sequentially transform every polygon $P_i \in D$ by applying the mapping function $\mathcal{T}$ to all the vertices of P_i, V_i^j, $j = 1, \ldots, q$. The algorithm follows:

Input:
S(u, v), A surface to add details to;
D, A polygonal geometric texture;
Output:
The modeled geometry texture over S;
Begin
$\mathcal{T}(u, v, w) \Leftarrow S(u, v) + \bar{n}(u, v)\, w$;
For all *polygons* $P_i \in D$
Do
For each *vertex* $V_i^j = (x_i^j, y_i^j, z_i^j)$ in P_i
Do
$V_i^j \Leftarrow (x_i^j, y_i^j, z_i^j)$
Od
Od
End

Notice that in the algorithm, the polygonal mesh is modified in place. Moreover, the trivariate function, $\mathcal{T}$, need not be defined explicitly. Given a vertex location, (x_i^j, y_i^j, z_i^j), the unit normal, $\bar{n}(x_i^j, y_i^j)$, could be evaluated on the fly. Therefore, the evaluation of the mapping of polygonal models in the algorithm is, again, a highly efficient forward process.

$\mathcal{T}$ could also be defined over a polygonal surface mesh S with an available parameterization. The parameterization could help identify the triangle, $\mathcal{T} \in S$, vertex V_i^j is above, by detecting the support position of (x_i^j, y_i^j) in the parameterization of S. The normal of S through V_i^j could then be approximated with the aid of the barycentric coordinates of the (x_i^j, y_i^j) coefficients of V_i^j in $\mathcal{T}$. These barycentric coordinates blend the normals of the vertices of $\mathcal{T}$, yielding a continuous approximation of the normal of S through V_i^j.

If $q > 3$, the mapped polygon, $\mathcal{T}(P_i)$ is not necessarily planar anymore. Splitting all polygons in D into triangles would resolve this problem.

Being piecewise linear C^0 continuous, the polygonal approximation of the composition might introduce large errors. Gain and Dodgson proposed adaptive approaches that could be employed so polygons in D would achieve a prescribed tolerance via refinement.[1]

Reference

1. J.E. Gain and N.A. Dodgson, "Adaptive Refinement and Decimation under Free-Form Deformation," Eurographics, 1999; http://www.cl.cam.ac.uk/~nad/pubs/EGUK99JG.pdf.

angles.[7] Specifically, a conformal mapping will preserve the orthogonality between the normal vector and the tangent plane of the surface. While, for example, rigid motion is clearly conformal, a general trivariate function is hardly so. Nonetheless, the type of trivariate functions already presented (see Equation 3) does preserve angles to a certain extent, and thus could be considered almost conformal under some conditions.

Elsewhere, I examine the conditions under which direct mapping of normal vectors through $\mathcal{T}$ can yield a reasonable approximation.[8] Let (n_0^u, n_0^v, n_0^w) be the coefficients of the normal vector. If $\partial S(u_0, v_0)/\partial u$ and $\partial S(u_0, v_0)/\partial v$ are approximately orthogonal and of similar magnitude, and if $\bar{n}(u_0, v_0)$ is of similar length as well, you can employ

$$\hat{\mathcal{T}}(N_0) = n_0^u \frac{\partial S(u_0, v_0)}{\partial u} + n_0^v \frac{\partial S(u_0, v_0)}{\partial v} + n_0^w \bar{n}(u_0, v_0) \quad (4)$$

as a reasonable approximation to the normal mapping. Small values of w, near the base surface layer S, will further increase this approximation's accuracy.

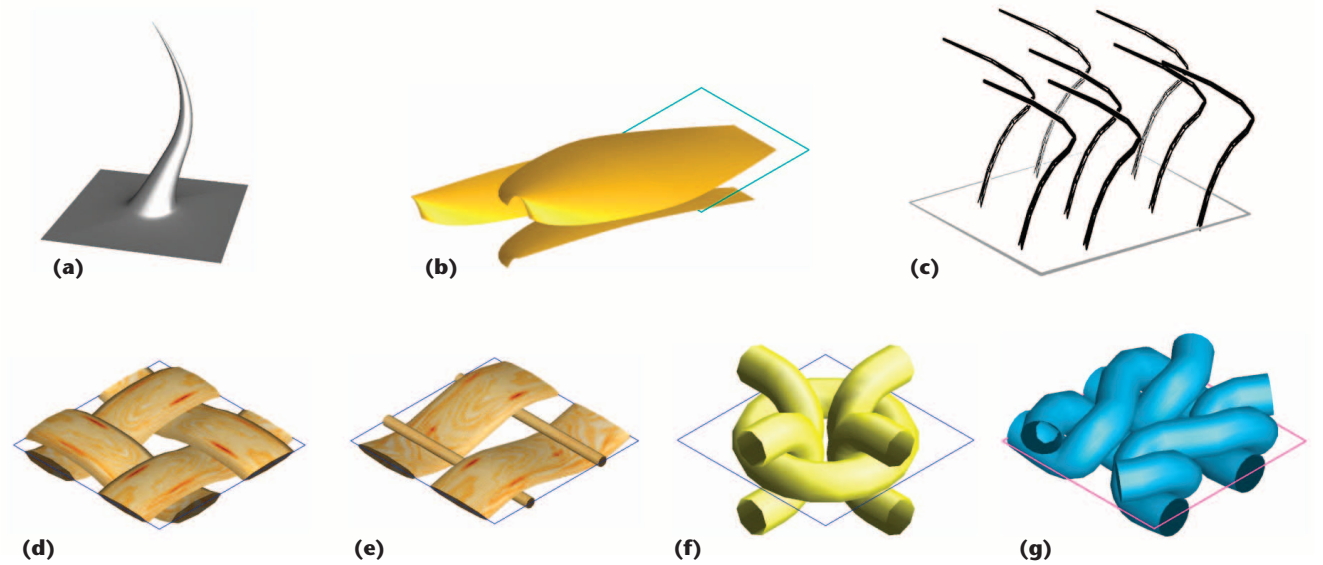
To estimate the normal field of the mapped texture geometry, as $\hat{\mathcal{T}}(N_0)$ in Equation 4, you need not only the surface normal at each vertex V_i^j, but also the surface's two partial derivatives. This data is not typically provided by contemporary rendering tools. In most cases, however, it's readily available in geometric modeling environments. In the continuous, polynomial, or rational case, these partials can be easily extracted from S via differentiation, an operation supported by all contemporary modeling systems. In the polygonal, discrete case, you can approximate $\partial S/\partial u$ ($\partial S/\partial v$) by examining the first-order-divided difference of the positions of a triangle's vertices with respect to the vertices' u (v) parametric values.

Texture modeling examples

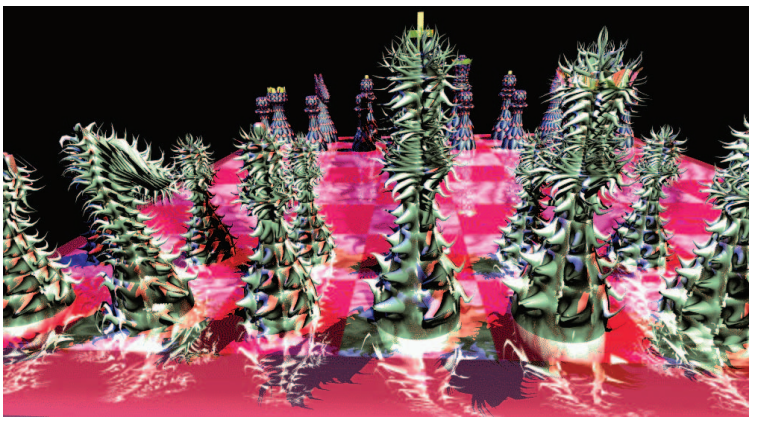
We created all examples presented in this section using an implementation based on the IRIT (http://www.cs.technion.ac.il/~irit) solid modeling environment developed at the Technion. All the ray-traced images presented in this article were created using the POV-Ray (http://www.povray.org) ray tracer.

Figure 4 (next page) shows a few examples of the geometric tiles used in this section. You can create these tiles using any modern geometric modeling system. We constructed these examples, in a few minutes each, using the IRIT solid modeling environment.

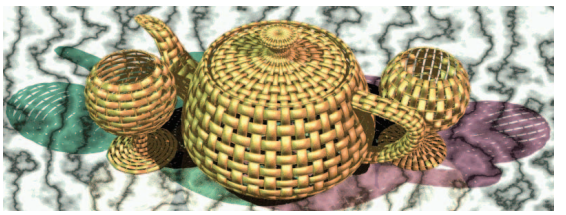
In Figure 5, the light chess pieces are covered by thorny bent geometric texture tiles (shown in Figure 4a). The dark pieces use the scale geometry tiles seen in Figure 4b, almost completely covering the pieces. The fact that the texture is a regular geometry allows us to preserve the geometric context and modify the attributes of the texture geometry at will. The light pieces have thorns that are color modulated with a low-frequency volumetric noise function, globally and contin-



4 Periodic tiles that serve as geometric textures in this article and in the coming figures: (a) thorny tile, (b) scaly tile, (c) furry tile, (d-e) wicker tiles, (f) chain link tile, and (g) knitted-looking tiles. In all subfigures, the domain of the period ($t_u \times t_v$) is also shown as a square.



5 Chess set covered with a geometric texture modeling in the shapes of thorns for the light pieces and a casual geometric texture of scales for the dark pieces (this scene has more than a million polygons).



6 Utah teapot with two wine glasses that are all made of wicker. The glasses show two different styles. See Figures 4d and 4e for the geometric texture tiles used.

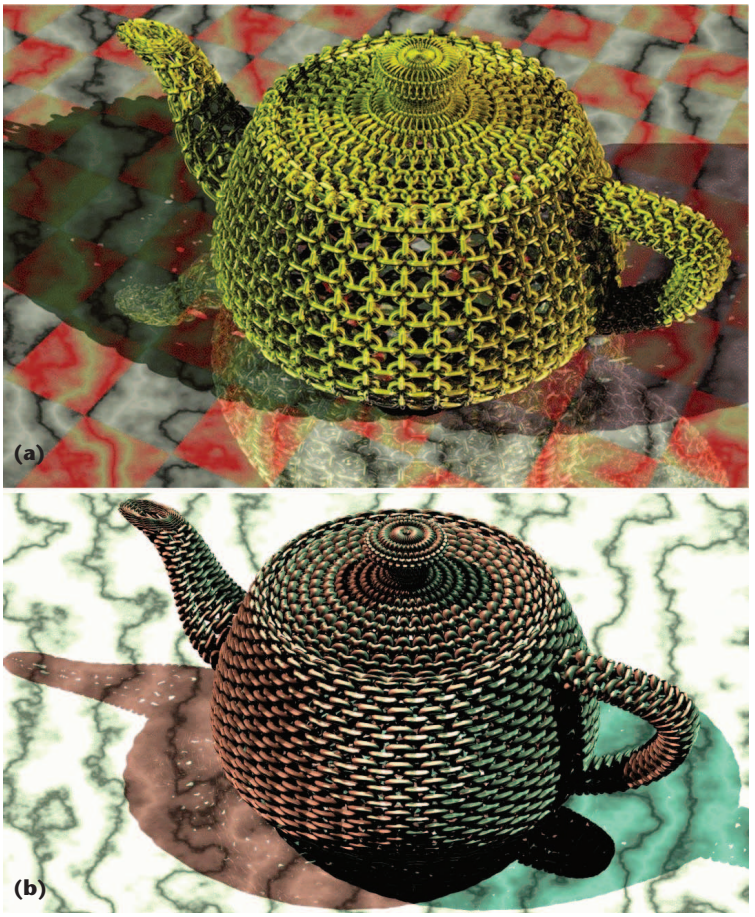
uously over each chess piece, and are reflective. In contrast, the texture of the dark pieces is formed out of two types of scales, each with different color and color-modulation functions.

The geometric texture tiles in Figure 4 are not confined to their canonic (0, 1) domain. The scale geometry, as shown in Figure 4b, clearly extends beyond its canonic domain with the expected effect of proper mapping using the close neighborhood of T. The capability to support such a domain extension allows the definition of simpler tiles with overlapping, as is the case with the scaly tiles in Figure 4b used in the dark pieces of the chess set in Figure 5.

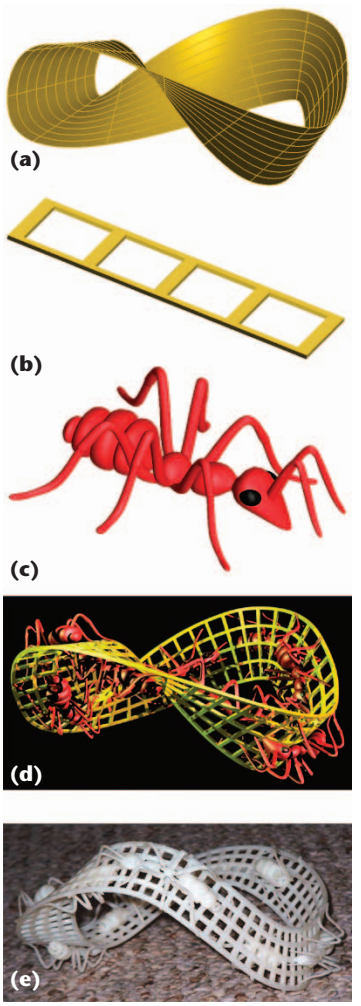
Because it has the ability to support full 3D geometry as texture, the presented texture modeling scheme enables the modeling with ease of geometry that would otherwise be painstakingly difficult to model. One example of such geometry is wicker-worked objects. By designing a geometric texture tile that captures one atomic element of the wicker style, you can texture any surface with wicker geometry of that same style. In Figure 6, we modeled the Utah teapot and two wine glasses out of wicker. Figures 4d and 4e show the tiles used. When the texture's geometry is tiled along the surface, it clearly follows the surface's shape. Notice that the wicker style of one glass is along the latitude lines of the glass and in the other glass it is along the longitude lines.

Figures 7 and 8 show other examples of geometry difficult to model but made simple when using the presented geometric texture tool. In Figure 7a, the Utah teapot is reconstructed out of golden chains (a single tile is presented in Figure 4f). Figure 7b shows the Utah teapot knitted using a simple knot shown in Figure 4g.

Figure 8 is a reconstruction of the *Moebius Strip II* drawing by M.C. Escher. Here again, a difficult object to recreate with the aid of contemporary geometric modeling tools is built in a few minutes with ease as geometry texture. Figure 8a shows a straightforward Moebius strip. The simple geometric texture tile in Figure 8b allows the easy re-creation of a hollow Moebius strip by placing this tile 45 times along the strip. A regular ant geometric model, shown in Figure 8c, is also interwoven nine times on both sides of the Moebius strip as a



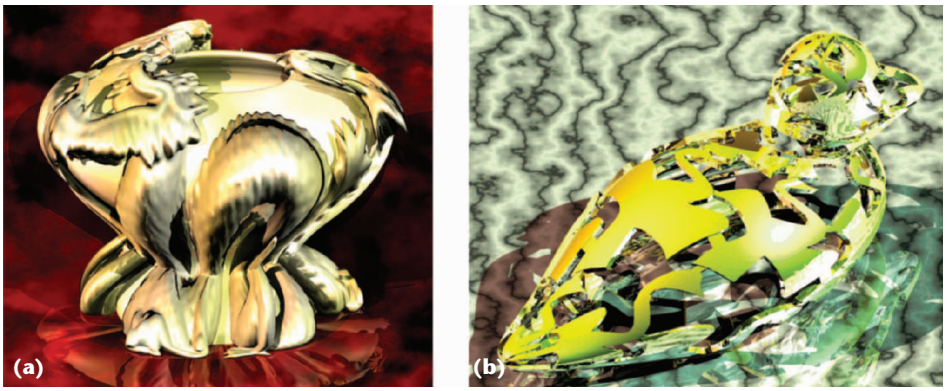
7 Utah teapot modeled using (a) a golden chain texture geometry (see Figure 4f), and (b) knitting style knots (see Figure 4g).



8 Using (a-c) as the input geometry, (d) a geometric texture reconstruction of the *Moebius Strip II* by M.C. Escher (a ray-traced version) and (e) a real plastic manufactured model are presented. (The *Moebius Strip II* model was manufactured with the aid of Sam Drake, University of Utah.)

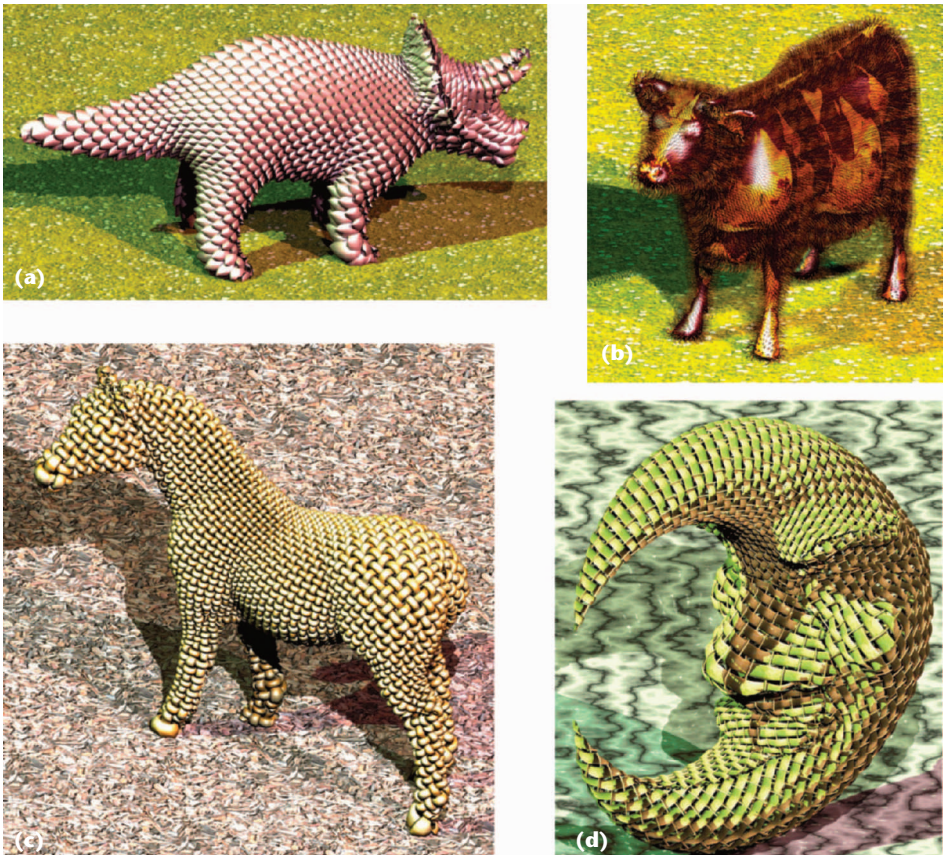
geometric texture. Figure 8d shows the final (and valid) geometric model. The precise placement of the hollow tiles along the strip, and moreover, the accurate placement of the ants so their legs are in proper contact with the strip, are difficult tasks that are easily accomplished using the proposed geometric texture modeling scheme. Figure 8e shows the manufactured realization of this model.

Figure 9 presents two additional examples of complex geometric texture models that are properly laid over the base surface S. Models of a dragon and a flying duck are employed as geometric texture tiles and placed over the surfaces of a vase and a duck, respectively.

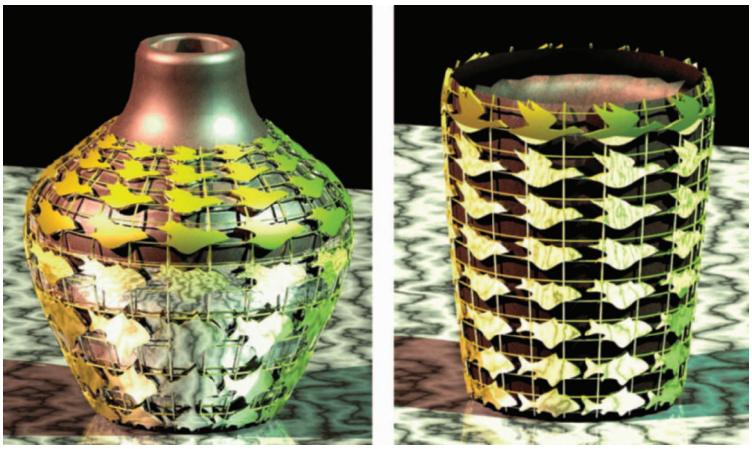


9 (a) Three dragons wrapped around the surface of the golden vase, serving as its base. In (b), the shape of a flying bird is tiled along a free-form base surface, in the shape of a duck. (The model of the dragon used as the geometric texture map was provided courtesy of The Stanford 3D Scanning Repository, http://www-graphics.stanford.edu/data/3Dscanrep.)

Attempts to glue geometry onto surfaces are known mostly for decorative purposes. Typically, some primary-orientation curves are used to guide the warping function along the base surface, an easier to maintain approach compared to the approach presented here. Nevertheless, the glued geometric is typically not a covering geometric texture, the continuity conditions cannot always be guaranteed, and the behavior above the surface is not easily controlled, resulting in less general schemes. Figures 8 and 9 demonstrate the level of precision possible with texture-tiling complex geometry over prescribed surfaces. The presented modeling techniques of placing a given texture geometry over a prescribed surface geometry S is a powerful modeling



10 Examples of texture modeling over polygonal meshes. (a) A scaly geometric texture (triceratops courtesy of http://www.ocnus.com), (b) a hairy cow (cow courtesy of http://www.ocnus.com), and (c-d) wicker geometry tiled over the models (horse model courtesy of http://www.3dcafe.com and moon model courtesy of the anonymous source). (Alla Sheffer, UBC, provided the parameterizations for the models.)



11 Geometric texture modeling using metamorphosis sequences between flying-duck tiles (top row) and fish-looking tiles (bottom row), is shown over two types of vases.

capability that makes the modeling of complex scenery with repeated texture geometry simple.

Finally, Figure 10 shows a few examples of geometric texture tiles over polygonal meshes. The scaly texture tile from Figure 4b is placed over the triceratops in Figure 10a. Hair is simulated over the cow model in Figure 10b with the aid of the geometric casual texture tile shown in Figure 4c. The wicker textures from Figures 4d and 4e are used as geometric textures in Figures 10c and 10d over the horse mesh and moon mesh. The tiles are placed over the polygonal mesh after a global parameterization was assigned to it. In these examples the angle-based flattening (ABF) parameterization scheme was employed to parameterize the geometry.[5] Using this parameterization, we derive a continuous normal field by blending the three normals of the triangle's vertices that contain the support position with the barycentric coordinates of the support position. This completes the definition of the C^0 continuous $\mathcal{T}$ function (recall Equation 3) over the polygonal meshes, which were used in Figure 10. See also the "Polygonal Composition Algorithm" sidebar.

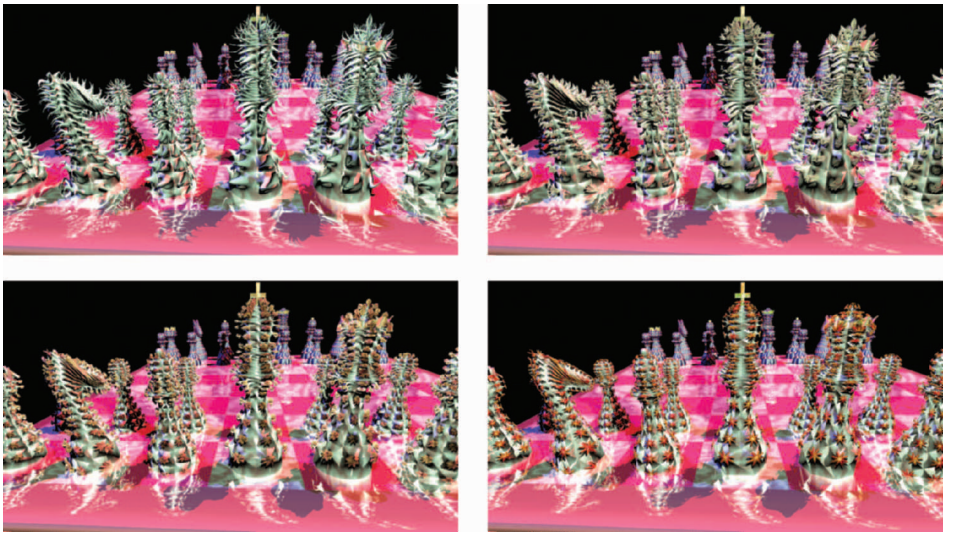
Animated and metamorphed texture modeling examples

We have already demonstrated that recognized techniques for processing geometry could be applied to geometric texture tiles. These include attribute settings such as colors, reflectivity, and translucency, but the list is clearly not limited to such settings. Next, we will explore applying known animation and morphing techniques over a prescribed geometry for geometric texture tiles, further expanding the modeling capabilities of the proposed tools.

In Figure 11, a tile with an outline in the shape of a flying bird is metamorphed continuously into a fish-like tile. Figure 11 shows two examples of vases decorated by a golden grid with rows that continuously change from flying birds (top row) to fish (bottom row).

A second type of possible animation is continuous interpolation over time between two geometric texture tiles that offer a smooth transition over time between

two texture tiles. Each tile will change over time in place, for example, by smoothly interpolating two types of thorns or even between a thorn and a flower. You can easily create a smooth animation of metamorphed geometric texture, using any available geometry metamorphosis method. This simple extension has a clear advantage over the traditional, image-based displacement mapping where the geometric context is lost. Due to the generality of the presented texture modeling technique, any available geometric metamorphosis technique can be employed between the source and target tiles to create such interpolation sequences. Figure 12 shows one such example, presenting a few snapshots from an animation of geometric texture tiles that are metamorphed from a thorn to a flower.



12 Snapshots from an animated metamorphosis sequence presenting geometric texture in the shape of thorns (top left) that continuously change into flowers (bottom right).

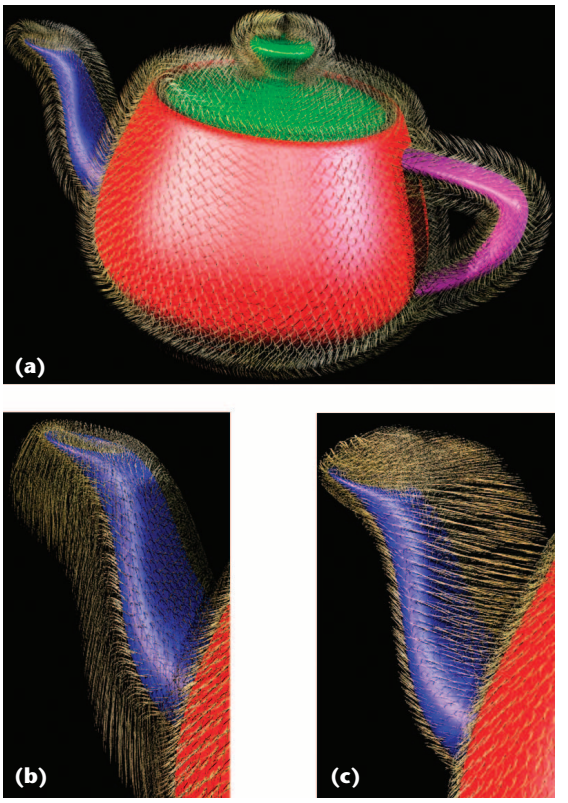
You can apply a different type of animation to the trivariate function itself—not to the geometric texture tile. The functions defined so far (see Equation 2) assumed that the volume above the surface is linearly parameterized along w and in the direction of the normal to (the tangent plane of) the surface. Clearly, this need not be the case. For example, consider the trivariate function above the surface S that is defined as

$$\mathcal{T}_{\psi}(u,v,w) = S(u,v) + \bar{n}(u,v)w + \psi w^{k}, w > 0$$

where ψ is some prescribed attraction vector having its effect controlled to the kth power with respect to w. The influence of ψ increases as you move away from the surface. One intuitive view of this effect is gravity (vertical ψ) or wind in the direction of ψ. All the volume above the surface S will be attracted to the direction of ψ, bending all the texture geometry with it.

Figure 13 shows several examples of simulation of fur over the Utah teapot, with and without the attraction vector ψ. The basic casual fur geometric tile used in Figure 13 appears in Figure 4c.

The complexity of the presented texture modeling approach is linearly dependent on the number of tiles placed along the base surface. In all presented examples, this number was in the thousands, placing dozens of tiles along each of the two parametric directions. Having this many tiles, the geometry of all the examples presented here was synthesized in a few seconds on a modern PC workstation.



13 Utah teapot with casual fur textures: (a) fur oriented along the surface normals (about 600,000 polygons). (b-c) gravity and wind bias added to the created fur.

Conclusions and future work

The presented method generates precise geometry. Nonetheless, this same geometry is also typically large. Methods should be sought to either compress or reduce this excessive amount of data in the order of hundreds of thousands of polygons. Because the geometric texture mapping presented here is typically nonlinear, it's difficult to compute its inverse function, in an attempt to follow the approach of Kajiya and Kay,[9] for ray-tracing applications.

While it's impossible to guarantee a fixed tile size in Euclidean space, you could alleviate these distortions. The notion of arc length does not extend to surfaces, yet you can still change the speed of the parameterization, seeking, for example, as isometric as possible mapping. Alternatively, you can modulate the height of the tiled geometry, providing another degree of freedom to this texture modeling process. ■

Acknowledgments

The research related to this article was supported in part by the Fund for Promotion of Research at the Technion–Israel Institute of Technology, Haifa, Israel, and in part by the Israeli Ministry of Science Grant No. 01-01-01509.

References

1. E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces*, tech. report UTEC-CSc-74-133, doctoral dissertation, Computer Science Dept., Univ. of Utah, 1974.
2. J.D. Foley et al., *Fundamentals of Interactive Computer Graphics*, 2nd ed., Addison-Wesley, 1990.
3. J.F. Blinn, "Simulation of Wrinkled Surfaces," *Proc. ACM Siggraph*, vol. 12, no. 3, ACM Press, 1978, pp. 286-292.
4. T.W. Sederberg and S.R. Parry, "Free-Form Deformation of Solid Geometric Models," *Proc. ACM Siggraph*, vol. 20, ACM Press, 1986, pp. 151-160.
5. A. Sheffer and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening," *Engineering with Computers*, vol. 17, no. 3, Springer, 2001, pp. 326-337.
6. J.E. Gain and N.A. Dodgson, "Preventing Self-Intersection under Free-Form Deformation," *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 4, 2001, pp. 280-298.
7. M. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.
8. G. Elber, "Geometric Deformation-Displacement Maps," *Proc. 10th Conf. Pacific Graphics*, IEEE Press, 2002, pp. 156-165.
9. J.T. Kajiya and T.L. Kay, "Rendering Fur with Three Dimensional Textures," *Proc. ACM Siggraph*, ACM Press, 1989, pp. 271-280.

Gershon Elber *is an associate professor in the Computer Science Department at Technion in Israel. His research interests include computer-aided geometric designs and computer graphics. Elber has a BS in computer engineering and an MS in computer science from Technion, and a PhD in computer science from the University of Utah. He is a member of the IEEE and ACM. He serves on the editorial boards of* Computer Aided Design, Computer Graphics Forum *and the* International Journal of Computational Geometry and