Generation of Organic Textures with Controlled Anisotropy and Directionality via Packing Rectangular and Elliptical Cells

Takayuki Itoh
IBM Research, Tokyo Research Lab.
1623-14 Shimotsuruma, Yamato-shi
Kanagawa 242-8502 JAPAN
+81-46-215-4925
itot@computer.org

Kazunori Miyata
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi
Ishikawa 923-1292 JAPAN
+81-761-51-1818
miyatak@acm.org

Kenji Shimada
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
+1-412-268-3614
shimada@cmu.edu

Abstract

This paper presents a computational method for generating organic textures. The method first tessellates a region into a set of pseudo-Voronoi polygons using a particle model and then generates the detailed geometry of each of the polygons using Loop's subdivision surface with fractal noise. Unlike previous particle models, which are designed for creating hexagonal cell arrangements, this particle model can also create rectangular cell arrangements, often observed in organic textures. In either cell arrangement, the method allows a user to control the anisotropy of the cell geometry and the directionality of the cell arrangements. A detailed three-dimensional cell geometry is then created by adjusting a set of parameters that control the cells' height and degree of skewing and tapering. A user can create various types of realistic looking organic textures with ease, by simply choosing a cell arrangement type, anisotropy, and directionality, along with the geometry control parameters.

Keywords
texture synthesis, texture mapping, rendering, anisotropic meshing, Voronoi tessellation, subdivision surface

1. Introduction

Texture is one of the attributes of an object's surface, and it is often the key to generating realistic images in computer graphics. Textures represent various aspects of an object's surface properties, including optical properties, such as colors and glossiness, and geometric properties, such as repeating bumps and dents. Adding texture produces richer and more realistic surface image than graphics with flat colored surfaces.

It would be tedious and time consuming, and thus not practical, for a graphics designer to manually draw an organic texture. One common way to avoid manual drawing is to scan a photograph of a real organic texture and map it onto a surface geometry. This approach, however, has the following problems:

(1) When the surface of an organic texture has small geometric features, such as bumps and dents, their shades and shadows do not match the lighting conditions used for rendering the rest of the scene in a real photograph.
(2) When the aspect ratio of the real photograph is different from that of the target surface, there will be unnatural distortion of the mapped texture.
(3) When the boundary of the real photograph is different from that of the target surface, which is usually the case, there will be undesirable cut-offs or gaps around the boundary.

The method proposed in this paper solves these problems by: (1) tessellating a region to be texture-mapped into a set of polygons using a particle model, (2) generating a texture cell geometry for each polygon using Loop's subdivision surface with fractal noise, and (3) rendering an image to be texture-mapped onto the object's surface. Because a particle system and physically-based simulation is used to tessellate a region into a set of pseudo-Voronoi polygons, the proposed method may seem similar to some of the previous work in texture generation especially the cellular texture method proposed by Fleischer et al. [7]. However, this particle method, called Bubble Mesh and originally proposed for mesh generation for FEM analysis, is unique for two aspects that make this method particularly suitable to organic texture generation: (1) the user can specify an anisotropic force field, enabling anisotropic texture cell generation, and (2) the user can specify not only hexagonal cell arrangements, but also rectangular arrangements with controlled directionality. After a region is tessellated into a set of pseudo-Voronoi polygons, an initial polyhedral shape is defined for each of the polygons by sweeping each polygon with tapering and skewing effects. The polyhedral geometry is then refined and rounded by Loop's subdivision scheme, while fractal noise is used to add fine geometric features.

With this method, a user can automatically create an organic texture consisting not only of hexagonally arranged cells, but also of orthogonally arranged cells with controlled anisotropy and directionality. Because the method first tessellates a region into a

set of polygonal cells, it generates a texture for an arbitrarily shaped surface domain by simply specifying the geometry, cell directionality, and a few rendering parameters, without manual intervention.

The reminder of this paper is organized as follows: Section 2 describes the characteristics of organic textures, and Section 3 discusses some related work. The overview of this method is then presented in Section 4. Section 5 presents an anisotropic meshing technique that generates a pseudo-Voronoi tessellation. Section 6 describes a method of generating realistic organic surfaces and textures for each of the polygons. Section 7 shows some results, and finally, Section 8 gives conclusions.

Figure 1. Examples of organic textures: crocodile (left), lizard (right).

2. Characteristics of organic textures

By observing real organic textures, such as the ones shown in Figure 1, the following characteristic are observed:

- An organic texture consists mostly of cells in hexagonal or rectangular arrangements. The basic shape of each cell is a hexagon or a rectangle, respectively.
- In orthogonal cell arrangements, organic textures often show distinct directionality, forming "stream lines."
- Cell geometry is often stretched in a certain direction. The aspect ratio of the anisotropy and its orientation may vary over the entire surface domain.

There are two constraints making the generation of such organic textures difficult: (1) the rectangular arrangement of the cells, and (2) the anisotropic cell geometry. These two issues are closely related to how a region is tessellated, and they must be addressed in order to generate realistic looking organic textures. Once a set of points or generators is located in a two-dimensional domain, the popular Voronoi method can be used to tessellate the domain into a set of polygons [11]. The larger challenge, however, is to find a set of points with a rectangular arrangement in which the rectangles' orientations are aligned with specified stream lines.

Such point locations cannot be created with the physically-based particle systems previously proposed in computer graphics research. Prior inapplicable particle systems include re-tiling of polygon models as described by Turk [17], surface modeling by Szeliski and Tonnesen [18], representation of implicit surfaces by Figueiredo et al. [19], and by Witkin and Heckbert [20], and anisotropic triangulation by Bossen and Heckbert [21]. This is because each of these particle systems is designed to create a hexagonal arrangement of points.

A cellular texture method [7] has also been proposed for the generation of surface details; it can be applied to a rectangular and an anisotropic tessellation. Because the method is an energy-optimization approach, rather than a physically-based approach, it requires expensive computations.

In this work, therefore, a different physically-based particle system is used, called Bubble Mesh, which was originally devised for solving various meshing problems in FEM analysis. Bubble Mesh is devised to create different types of meshes, triangular, quadrilateral, isotropic or anisotropic, by packing cells of various shapes—circles for isotropic triangular meshes [1], ellipses for anisotropic triangular meshes [2], squares for isotropic quadrilateral meshes [3], and rectangles for anisotropic quadrilateral meshes [4].

3. Related work

Previous research related to the present work has been published in three areas: (1) tiling textures, which subdivide a surface into sub-regions and generate procedural textures for each sub-region; (2) modeling and rendering of organic materials, which increase scene richness; and (3) cellular textures, which apply particle systems for modeling surface details. Among these, the cellular texture approach is most closely related to our method because our method also uses a particle system.

3.1 Tiling textures

A reference book on visual geometry surveys various aspects of patterns and tiling [9]. Yessios has presented methods of generating common materials, such as stones and wood, with two-dimensional line patterns [13]. Miyata has proposed an enhanced method for automatically generating three-dimensional stone wall patterns [12]. A limitation of such tiling textures is that regions must be aligned carefully, or visible discontinuities, such as seams and gaps, may be apparent.

3.2 Modeling and rendering of organic materials

Various methods have been proposed for the modeling and rendering of organic materials. Reaction-diffusion equations, which were originally proposed as a model of morphogenesis by Turing, have been applied to the texture synthesis problem [14, 15]. Fowler et al. have reported a method of generating seashell patterns using reaction-diffusion equations [16]. Worley reported a cellular texture basis function [25], which complements Perlin's noise function [26].

3.3 Cellular texture

Fleischer et al. proposed a cellular texture method [7] that can model surface details such as scales, feathers, and thorns. Their method computes the locations, orientations, and other properties associated with cellular particles. After a hexagonal arrangement of a system of particles is obtained by optimization, each particle is converted to a geometric unit with user-defined appearance parameters. The detail of each texture cell is then rendered.

There are several similarities between their work and ours:

- Both methods use particle systems, and each particle has energy potential.
- The energy potential is calculated based on particle-to-particle distance and direction.
- The total energy potential is minimized iteratively.

By defining inter-particle or inter-cellular forces explicitly, we speed up the cell tessellation process significantly. In the cellular texture method of Fleischer et al., cells are located using an

energy-optimization approach, and the energy of each cell is calculated by a cost function consisting of several energy terms. This requires extensive computations. This approach, on the other hand, calculates the inter-cellular force explicitly and solves the equation of motion, a second order ordinary differential equation, using the fourth-order Runge-Kutta method. Consequently, this method converges very quickly—all the examples shown in this paper required only 10 to 20 seconds to find the cell arrangements with an Intel Pentium III 650 MHz PC.

4. Method overview

The proposed method consists of three steps, as depicted in Figure 2:

1. **packing pattern generator:** makes a packing pattern of skin cells, a pseudo-Voronoi tessellation, using an anisotropic triangular or quadrilateral meshing technique. The resultant tessellation can be hexagonal or rectangular. (described in Section 5)
2. **skin geometry generator:** generates each skin cell's three-dimensional geometry using Loop's subdivision surface method with fractal noise. (described in Section 6)
3. **renderer:** renders a final texture image based on the skin geometry and lighting parameters.

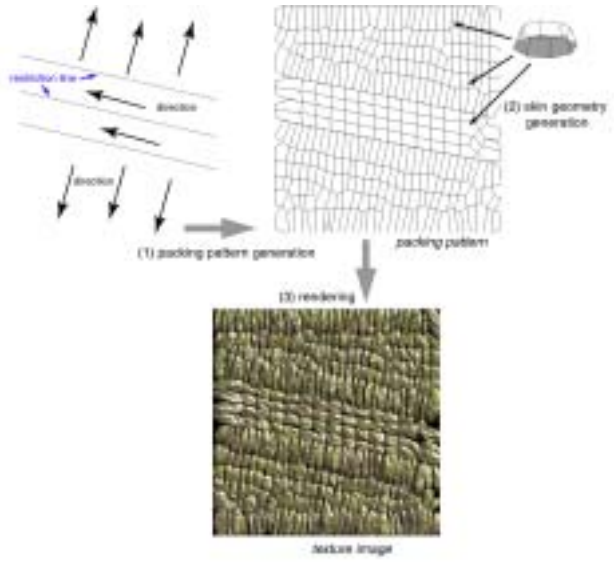


Figure 2. Method overview: After a packing pattern is generated, the skin geometry is generated. A textured image is then obtained by rendering with specified lighting conditions.

5. Pseudo-Voronoi tessellation via anisotropic meshing

This section describes the present approach to generating packing patterns, pseudo-Voronoi tessellations, for organic texture. This problem can be stated as follows.

Given:

- a two-dimensional geometric domain,
- a desired size distribution of cells, given as a scalar field,
- a desired directionality, given as a vector field, and
- a desired anisotropy, given as a scalar field,

Generate:

- a polygonal tessellation consisting primarily of quadrilateral or hexahedral polygons compatible with the specified cell sizes, directionality, and anisotropy.

The input size distribution can be specified by the user, or automatically calculated according to the curvature of the input domain or other metrics. The input directionality can be specified by the user, or automatically calculated by interpolating from the direction of the input boundary edges.

This approach is based on the observation that natural-looking hexahedral or rectangular tessellations are geometric duals of well-shaped triangular or quadrilateral meshes, as illustrated in Figure 3. The algorithm performs pseudo-Voronoi tessellation in three steps:

1. packing elliptic or rectangular cells closely in the domain (Figure 3 left).
2. generating a triangular or quadrilateral mesh by connecting the centers of the ellipses or rectangles (Figure 3 center).
3. tessellating the domain into mostly hexagonal and rectangular polygonal cells (Figure 3 right).

Figure 4 shows an example of a pseudo-Voronoi tessellation. Figures 4 (a) and 4 (b) show the preferred cell size distribution and preferred directionality of rectangular packing, respectively. Figure 4 (c) depicts a closely packed set of rectangles. Figure 4 (d) shows the mostly quadrilateral mesh created by connecting the centers of the rectangles. Figure 4 (e) shows the pseudo-Voronoi tessellation.

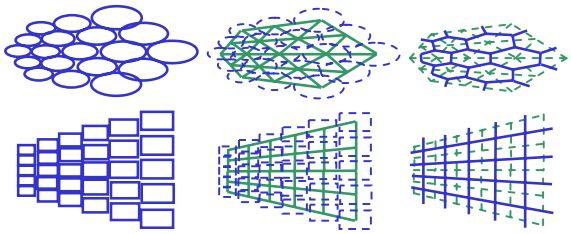
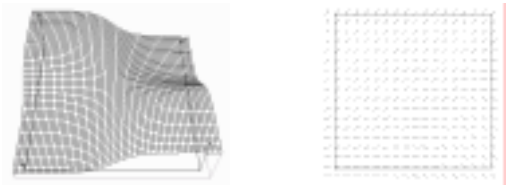


Figure 3. (upper) Elliptic cells, a triangular mesh, and hexahedral Voronoi polygons. (lower) Rectangular cells, a quadrilateral mesh, and quadrilateral Voronoi polygons.



(a) Heights of grid points represent preferred cell sizes.

(b) Preferred directionality of rectangular packing

(c) Closely packed rectangles **(d) Primarily Quadrilateral mesh** **(e) Pseudo-Voronoi tessellation**

Figure 4. Four sub-steps of packing pattern generation

5.1 Close cell packing with proximity-based forces

The particle model implemented here is designed to efficiently obtain a closely packed arrangement of elliptic, or rectangular cells. A proximity-based force field is defined between pairs of cells, so that the force field exerts a repelling force when the two cells are located too close; an attracting force is exerted when the two cells are separated by more than a specified distance. The approach presented in this paper is based on the Bubble Mesh method, originally proposed for triangular mesh generation [1]. This method tightly packs a set of circles using a force field similar to the van der Waals force.

In the original Bubble Mesh method, the stable distance between the centers of two adjacent circular cells is calculated as the sum of the desired radii of the two circular cells. A user-given scalar field specifies the radii of the cells. Here, we denote the radii of the two cells as r_i and r_j; the distance between the centers of the cells as l; the stable distance between the cells as $l_0 = r_i + r_j$; the ratio of the current distance and l_0 as $w = l / l_0$; and the corresponding linear spring constant at the target distance as k_0. The force model used in the Bubble Mesh method is described as a function of w. The essential characteristic of this approach is that a repulsive force is applied when $w < 1$, and an attractive force is applied when $w > 1$. As shown in Figure 5(a), the force $f(w)$ defined satisfies the following conditions:

$$f(1) = f(1.5) - 0,\ f'(0) = 0,\ f'(1) = -k_0$$

By solving for the above conditions, the function is written as:

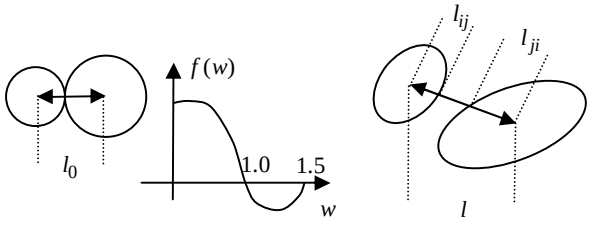
$$f(w) = \begin{cases} \dfrac{k_0}{l_0}\left(\dfrac{5}{4}w^3 - \dfrac{19}{8}w^2 + \dfrac{9}{8}\right) & 0 \le w \le 1.5 \\ 0 & 1.5 \le w \end{cases}$$

The Bubble Mesh method was later extended to provide close packing of elliptic cells [2]. This approach calculates the directions of the major and minor axes of a cell from the given directionality, and calculates the radii along these axes from the given cell sizes and anisotropy. The effective distance between two cells is calculated as the sum of the two lengths, measured along the line segment that connects the centers of the two cells from the center to the boundary of each cell. If the effective distance is less than the desired stable distance, then the force is negative (the cells are repelled from each other), otherwise the force is positive. Let these two lengths be l_{ij} and l_{ji}, as shown in Figure 5 (b), then the effective distance is written as $l_0 = l_{ij} + l_{ji}$.

The method was also extended to achieve close packing of square cells [3]. Here we denote a potential field around a cell, which is obtained as an integration of the above force field, as Ψ_{P_0}. In order for square cells to align in an orthogonal arrangement, we add four sub-potential fields Ψ_{P_1}, Ψ_{P_2}, Ψ_{P_3}, and Ψ_{P_4}, at the four corners of the square cell P_1, P_2, P_3, and P_4, to the original potential field Ψ, as shown in Figure 6 (a). If the desired cell size is locally uniform, the radii of the four sub-potential fields should be $(\sqrt{2}-1)r_0$, where r_0 is the radius of the central potential field Ψ_{P_0}. If graded cell sizes are specified, however, the radii of the sub-potential fields should be adjusted accordingly. The potential field shown in Figure 6(a) is expressed as a weighted linear combination of the central potential field and the four sub-potential fields, i.e., $\Psi = \Psi_{P_0} + (\sqrt{2}-1)(\Psi_{P_1} + \Psi_{P_2} + \Psi_{P_3} + \Psi_{P_4})$.

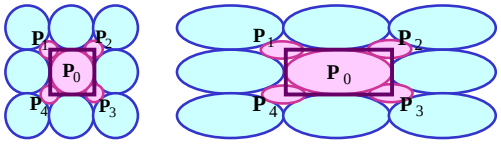
Similar to the elliptic cell packing shown in Figure 5 (b), this square cell packing approach was extended to rectangular cell packing [4], as shown in Figure 6 (b).

In all cases, a point mass at the center of each cell, and the effect of viscous damping are assumed. We then solve the equation of motion numerically to find a tightly packed configuration of cells using a standard numerical integration scheme, the fourth-order Runge-Kutta method. The integration process is terminated when the displacements of all cells in an iteration become small. While solving the equation, the number of cells in the domain is adjusted by checking the population density. One cell is added around another cell when the smallest w value between the cell and its adjacent cells is larger than a user-specified value. One cell is deleted when the smallest w value between that cell and its adjacent cells is smaller than a user-specified value.



(a) Stable distance and force function of circular cells **(b) Stable distance of elliptic cells**

Figure 5. Circular cell packing and elliptic cell packing



(a) A square cell realized using five circular force fields.

(b) A rectangular cell realized using five elliptic force fields.

Figure 6. Square cell packing and rectangular cell packing.

Input geometric models consist of a set of 3D polygons. As shown in Figures 7(a)(b), cells are first packed along the boundaries of the polygons, then packed inside the polygons. If the input model is too fine, cells are packed using a simplified model then projected onto the original input model.

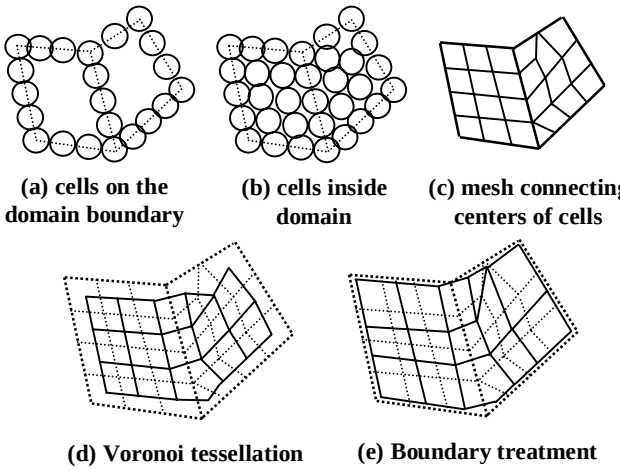
The complexity of the cell packing process is regarded as $O(mn)$, where n denotes the total number of square cells, and m denotes the average number of cells that suffer attractive or repulsive forces from another cell. To accomplish this, cells are stored into a grid surrounding the entire input domain, in order to search for the adjacent cells surrounding a given cell. Since it extracts an almost constant number of adjacent cells, m is treated as a constant. Consequently, the complexity of the process is linear to the total number of cells, and therefore the method should be faster than optimization-based methods such as the cellular texture method.

5.2 Pseudo-Voronoi tessellation from packed cells

Given closely packed elliptic or rectangular cells, the present approach generates anisotropic triangular or quadrilateral meshes by connecting the centers of the cells.

To obtain the configuration of an anisotropic triangular mesh from the centers of elliptic cells, an anisotropic Delaunay triangulation algorithm was applied [2, 4]. While the original Delaunay triangulation algorithm satisfies the condition that no other vertices lie inside the circumcircle of a triangular element, this anisotropic triangulation uses circumellipses defined by the user-given directionality and anisotropy, instead of circumcircles.

To obtain an anisotropic quadrilateral mesh by connecting the centers of a set of packed rectangular cells, a triangular-to-quadrilateral mesh conversion algorithm was used [3, 4]. The mesh conversion algorithm first tests all the possible quadrilaterals formed by coupling two adjacent triangular elements and then computes a score that measures the quality of each resultant quadrilateral element. The algorithm then converts the pairs of triangles to a set of quadrilaterals in the order of their scores. It is acceptable to have some isolated triangles remaining in the mesh because create a pseudo-Voronoi tessellation can still be created for such a mixed mesh, and the effect of a small number of triangles adds just a few non-four-sided polygons to the final tessellation.



(a) cells on the domain boundary **(b) cells inside domain** **(c) mesh connecting centers of cells**

(d) Voronoi tessellation **(e) Boundary treatment**

Figure 7. Cell packing and Voronoi tessellation processes.

Finally, this approach generates a pseudo-Voronoi tessellation from these anisotropic triangular, quadrilateral, or mixed meshes. This tessellation is obtained by connecting the center of each mesh element to the centers of all adjacent mesh elements. As shown in Figure 7(d), each internal mesh node is thus enclosed by a polygon, formed by connecting the centers of all the mesh elements that share the node.

The above method does not fill the entire given domain by the Voronoi polygons, and therefore it requires special treatment around the domain boundary. We have named the edge of the Voronoi polygon, not shared by any other Voronoi polygon a *boundary edge*. Boundary edges can be defined along a given domain's perimeter or at a set of user-specified line segments inside the domain. When a node is connected to such boundary edges, this approach connects the center of the boundary edges instead of connecting the centers of mesh elements. Also, the algorithm does not form pseudo-Voronoi polygons for a vertex on a boundary edge. Consequently this approach generates pseudo-Voronoi polygons which are well aligned along the mesh boundary and along other user-specified line segments.

6. Organic texture generation

The organic texture is obtained by generating a skin texture for each of the pseudo-Voronoi cells created by the method described in the previous section. Each skin texture is generated in the following three steps:

(1) initial skin mesh generation for each pseudo-Voronoi polygon
(2) smoothing of the initial skin mesh by a subdivision surface method
(3) small geometric features generation with fractal noise

Sections 6.1, 6.2, and 6.3 describe each step in more detail.

6.1 Initial skin mesh generation

Each pseudo-Voronoi polygon, or cell polygon, generated by the anisotropic cell packing technique described in Section 5, must be transformed into a realistic skin surface geometry.

First, depending on the type of target organic texture, it is

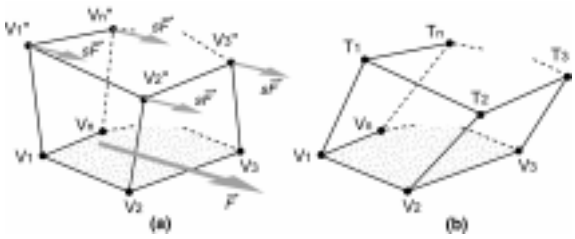


Figure 9. Skewing operation: (a) flow vector that defines the direction of skewing, and (b) skewed prism.

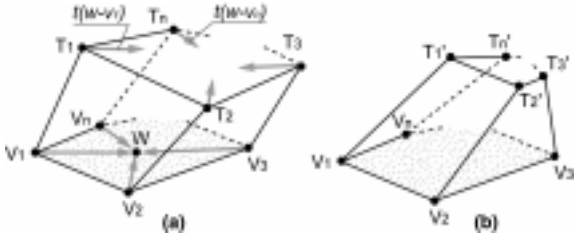


Figure 10. Tapering operation: (a) centripetal displacement, and (b) tapered prism.

possible to either add a gap between the cells, or pack the cells more tightly by scaling each of the cells. The amount of scaling is specified by a scaling parameter, *Sv*.

Next, an initial three-dimensional skin mesh is obtained by sweeping the polygon of each cell in the normal vector direction, $\vec{n}$. The height of this sweeping operation, *H*, is given by:

$$H = SkinSize \times Sr ,$$

where *SkinSize* is the average distance from the center of gravity of a cell polygon to the corners of the polygon, and *Sr* a user-specified sweeping parameter. The sweeping operation defines a prism.

This prism is then deformed by displacing the top corners, V_1', V_2', …, and $V'n$, randomly within a specified displacement range, as shown in Figure 8 (b). The distortion parameter, *Dr*, can be adjusted to create various skin shapes. We define the range of this random displacement by:

$$DisplacementRange = SkinSize \times Dr \times RandomNumber ,$$

where *RandomNumber* is a random number between 0 and 1.

After the initial skin shape is deformed, the prism can be skewed, as shown in Figures 9 (a) and 9 (b), according to a specified flow vector $\vec{F}$. Each of the top corners, V_1'', V_2'', …, and V_n'', is displaced by vector $s\vec{F}$, where *s* is a user-defined skewing parameter. This process adds some overlapping layers of skin cells. Choosing a larger skewing parameter can thus generate scale-like skins.

The prism can be further deformed by a tapering operation, as shown in Figures 10 (a) and 10 (b), by displacing each of the top corners of the prism, T_1, T_2, …, and T_n. Each top corner, T_i, is displaced by a centripetal vector $t(\vec{w} - \vec{v}_i)$, where *t* is the user-defined tapering parameter and $\vec{w}$ is the center of gravity of the base polygon. Setting a larger tapering parameter yields thorny skins. The inter-element collisions are not computed on the final

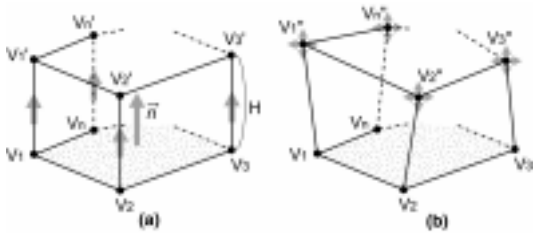


Figure 8. Initial skin shape: (a) sweeping base polygon, and (b) adding small random displacement

element shapes, and therefore if the skew variable is set high enough, elements will intersect.

After all of these deformations are applied, the shape of each skin cell is represented as a coarse triangular mesh, called the initial skin mesh. The final skin mesh is obtained by refining and smoothing the initial skin mesh using Loop's surface subdivision scheme, as described in the next section.

6.2 Mesh refining and smoothing

After initial skin meshes, or initial control meshes, are generated, their sharp corners must be smoothed out. This smoothing is archived by using a surface subdivision method.

Several surface subdivision methods have been proposed, including Doo-Sabin's method [6], Catmull-Clark's method [5], and Loop's method [10]. Loop's method is used here because it generates triangular patches and works well with this implementation.

In Loop[illegible] is black dots in Figure [illegible]ch edge of a given cont[illegible] to divide an initial tria[illegible]

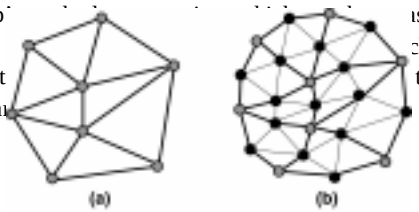


Figure 11. Mesh subdivision: (a) Initial mesh. (b) After subdivision.

6.3 Surface displacement

In parallel with the mesh smoothing process, skin mesh nodes can be further displaced with fractal noise in order to add small geometric features, such as bumps and dents, to a surface.

Three-dimensional fractal noise is generated by recursively subdividing a triangular mesh element into smaller triangles, as shown in Figure 12. A new node is added at the midpoint of each side of the triangle and the nodal point is then displaced in the vertical direction [8]. The displacement vectors for nodal points at each subdivision level are stored and merged into the subdivision surface geometry, as shown in Figure 13.

Figure 12. Three-dimensional fractal noise: The left triangle is subdivided into four smaller triangles by displacing each midpoint of the edge. This subdivision process is repeated recursively.

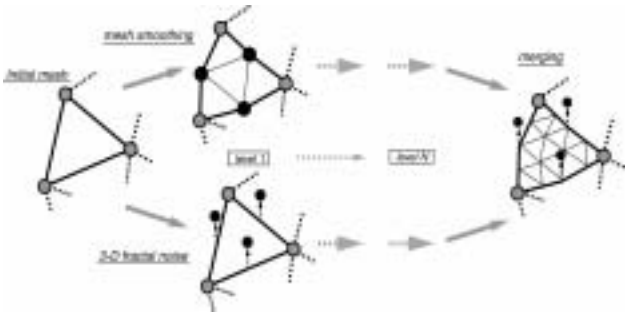


Figure 13. The final skin shape is obtained by adding fractal noise to the subdivision surface mesh.

7. Results

This section shows various organic textures that demonstrate the advantages of this proposed texture generation method.

As described in Sections 5 and 6, there are many parameters that can be adjusted, to create various types of organic textures:

- *Ca*: Degree of anisotropy, or aspect ratio of cell geometry
- *Sv*: Scaling parameter of cell polygon
- *Sr*: Sweeping parameter of initial skin mesh
- *Sc*: Skewing parameter of initial skin mesh
- *Tc*: Tapering parameter of initial skin mesh
- *Dr*: Distortion parameter of initial skin mesh ($0.0 < Dr < 1.0$)
- *Fd*: Fractal dimension of fractal noise controlling skin's surface roughness ($1.0 < Fd < 2.0$)
- *Af*: Amplitude of fractal noise

Table 1 summarizes all the parameters used to create the textures shown in Figures 14, 15, 16, and 17.

Changing the degree of anisotropy, or aspect ratio, *Ca*

One unique capability of this texture generation method is to control the anisotropy of the texture cells. Figure 14 shows how texture appearances change with different aspect ratios. By setting a high aspect ratio, the obtained textures are stretched in specified flow directions.

Changing the skew coefficient, *Sc*

Figure 15 shows the effect of changing the skewing parameter, *Sc*. Specifying a larger skewing parameter leads to some overlapping layers of texture cells, yielding scale-like skins.

Changing the amplitude of fractal noise, *Af*

Textures shown in Figure 15, compared with those in Figure 14, have bumpier surfaces due to the larger amplitude of fractal noise.

Changing the tapering parameter, *Tc*

Figure 16 shows the effect of changing the tapering parameter. The skins bulge out using a larger taper parameter..

Hexagonal cell arrangement and rectangular cell arrangement

Figure 17 shows more complicated organic textures. One of the advantages of this method is to be able to create cells packed in both hexagonal arrangements and rectangular arrangements. If a triangular mesh is used for generating the pseudo-Voronoi tessellation, the resultant cell arrangement becomes hexagonal, and if a quadrilateral mesh is used, then the resultant cell arrangement becomes rectangular. Figures 17 (a), (b), and (c) show organic textures with both hexagonal and rectangular cell arrangements in a single texture. All the examples shown in Figure 17 have some internal restriction lines. Note that the cells are well aligned along these restriction lines.

Application of organic textures to three-dimensional geometry

The proposed method can generate organic textures and map them onto a three-dimensional object; figures 18 and 19 show examples of this. In both examples, cells are packed in a rectangular arrangement and its directionality is specified so that the texture cells align well along the longitudinal direction of the legs.

Computational time

Computational time changes depending on the number of texture cells to be generated. In the examples shown in Figures 14, 15, 16, and 17, it takes 10 to 40 seconds to generate pseudo-Voronoi tessellation; it takes approximately 3 to 6 more seconds to generate the detailed skin geometry on an Intel Pentium III 933 MHz processor. The size of the generated texture images in all the examples is 512 by 512. Table 2 summarizes the computation time.

Rendering method

For the examples shown in Figures 14, 15, 16, and 17, the three dimensional meshes are laid on a plane and rendered in parameter space using in-house software. For the examples shown in Figure 18 and 19, the three dimensional meshes are imported into the CG software, 3D Studio MAX, and rendered. The number of generated micro-triangles depends on the number of the cells and their shapes, and is about 0.5 million on average.

Visual justification

The examples of Fleischer's method [7] were obtained by performing texture map on the surfaces. The methods treated herein do not use texture maps; therefore, the visual quality of these results might be lower than Freischer's results, but the geometrical richness(or variety) is competitive with them. The intention is to continue extending this method to generate surface attributes, such as colors and optical features, procedurally for each skin cell.

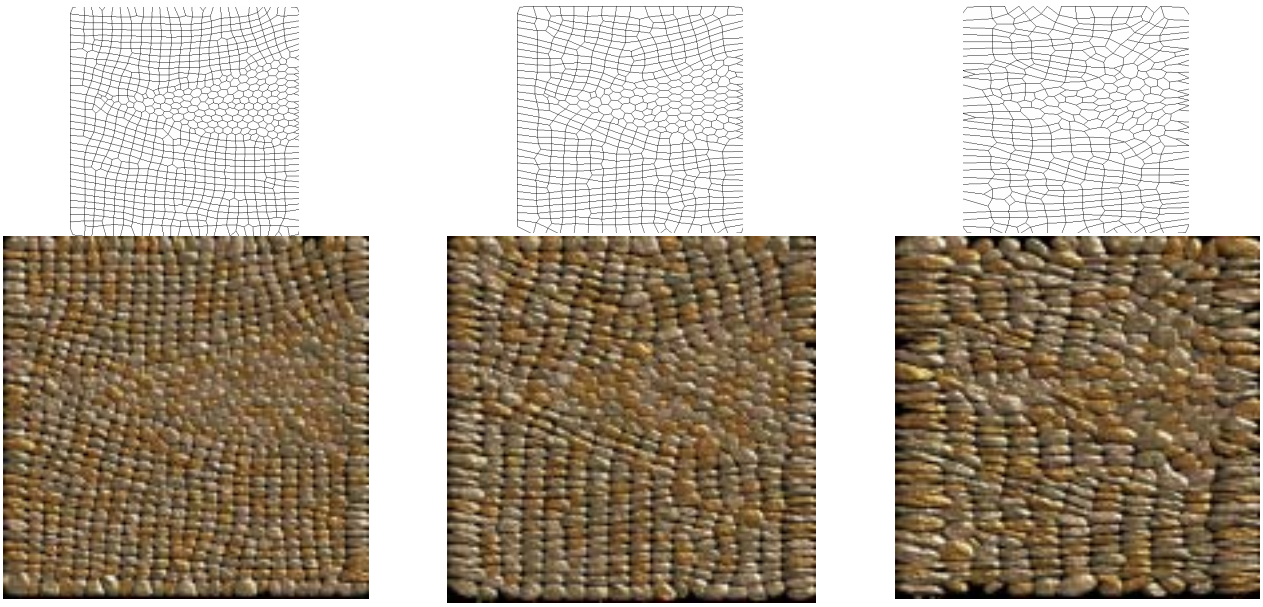
Table 1. List of parameters used for generating textures shown in Figures 14, 15, 16, and 17.

Fig. No.	*Ca*	*Sv*	*Sr*	*Sc*	*Tc*	*Dr*	*Fd*	*Af*
Fig.14 (a)	1.5	1.4	1.0	0.0	0.2	0.0	1.1	0.1
Fig.14 (b)	2.0	1.4	1.0	0.0	0.2	0.0	1.1	0.1
Fig.14 (c)	3.0	1.4	1.0	0.0	0.2	0.0	1.1	0.1
Fig.15 (a)	2.0	1.4	2.0	0.0	0.8	0.05	1.1	0.2
Fig.15 (b)	2.0	1.4	2.0	2.0	0.8	0.05	1.1	0.2

Fig.15 (c)	2.0	1.4	2.0	3.0	0.8	0.05	1.1	0.2
Fig.16 (a)	2.0	1.3	1.0	0.0	0.0	0.05	1.1	0.1
Fig.16 (b)	2.0	1.3	1.0	0.0	0.2	0.05	1.1	0.1
Fig.16 (c)	2.0	1.3	1.0	0.0	0.5	0.05	1.1	0.1
Fig.17 (a)	2.0	3.0	1.0	2.0	0.7	0.05	1.1	0.1
Fig.17 (b)	2.0	1.4	0.8	0.0	0.2	0.05	1.1	0.1
Fig.17 (c)	2.0	1.6	2.0	0.0	0.2	0.1	1.1	0.1
Fig.17 (d)	2.0	1.4	1.0	0.0	0.1	0.0	1.1	0.1

Table 2. List of computation times for generating textures shown in Figures 14 and 17.

Fig. No.	Time for pseudo-Voronoi tessellation (sec.)	Time for detailed skin generation (sec.)
Fig. 14(a)	31	5.3
Fig. 14(b)	34	4.2
Fig. 14(c)	32	3.8
Fig. 17(b)	32	4.8
Fig. 17(c)	16	4.0
Fig. 17(d)	10	3.9



(a) ***Ca*** **= 1.5** **(b)** ***Ca*** **= 2.0** **(c)** ***Ca*** **= 3.0**

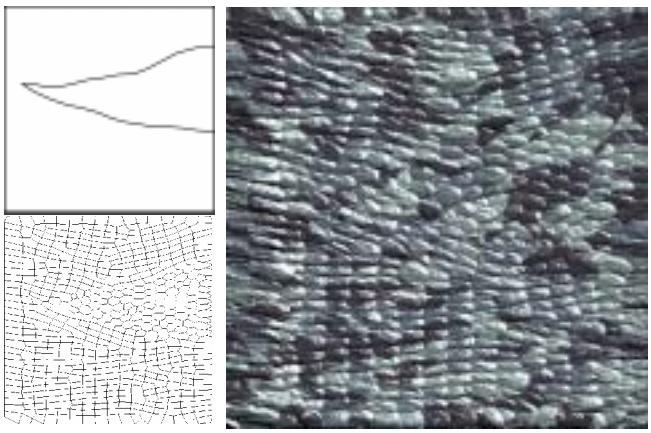
Figure 14: Textures with different degrees of anisotropy, or aspect ratios

(a) ***Sc*** **= 0.0** **(b)** ***Sc*** **= 2.0** **(c)** ***Sc*** **= 3.0**

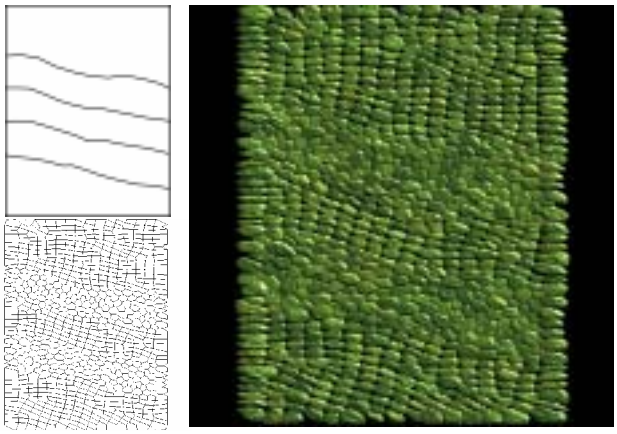
Figure 15: Textures with different values of the skewing parameter

(a) ***Tc*** **= 0.0** **(b)** ***Tc*** **= 0.2** **(c)** ***Tc*** **= 0.5**

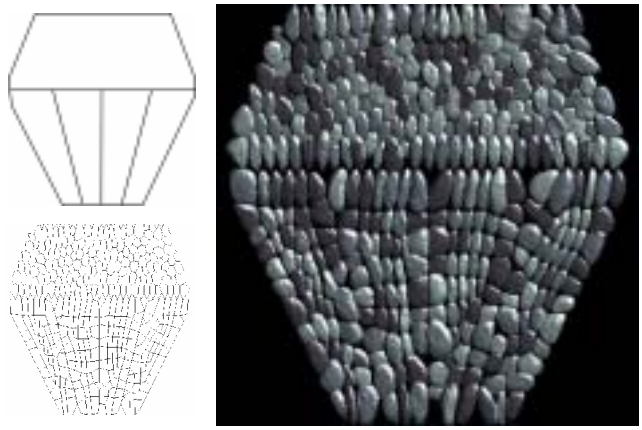
Figure 16: Textures with different values of the tapering parameter



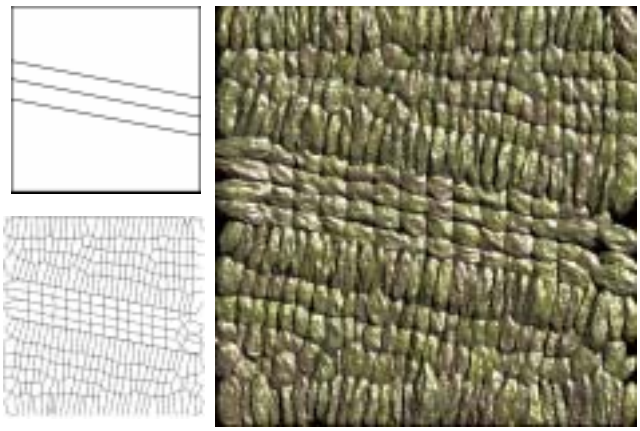
(a) Texture #1



(b) Texture #2



(c) Texture #3



(d) Texture #4

Figure 17: Various organic textures: In each example, input boundaries are shown at top left, Pseudo-Voronoi polygons at bottom left, and a generated texture image on the right.

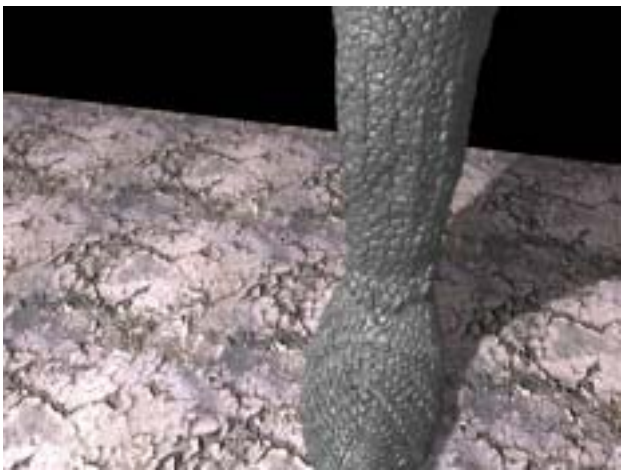


Figure 18: Textured leg 1.

Figure 19: Textured leg 2.

8. Conclusion

A method has been presented for generating a variety of organic textures with controlled anisotropy and directionality. A new particle model is adopted so that texture cells can be packed, not only in hexagonal arrangements, but also in rectangular arrangements. By simply specifying a boundary shape, texture cell directionality, anisotropy, and a few parameters that control the geometric details of each texture cell, a realistic image of an organic texture can be generated automatically.

In the future, we would like to extend our work in the following areas:

Weathering

The organic textures presented in this paper are pristine and immutable, even though real ones are not. Some weathering effects [22, 23] such as wear, abrasion, and wounds are also important factors in improving the reality of a computer-rendered organic texture image.

Varied skin conditions

In our method, a cell's shape is generated with a procedural approach. An interesting consideration for future work is the generation of an organic texture from real sample images or the simulation of various situations such as wet [24] and dirty skins.

References

[1] K. Shimada, D. C. Gossard, Bubble Mesh: Automated Triangular Meshing of Non-manifold Geometry by Sphere Packing, *Third Symposium on Solid Modeling and Applications*, pp. 409-419, 1995.

[2] K. Shimada, A. Yamada, T. Itoh, Anisotropic Triangulation of Parametric Surfaces via Close Packing of Ellipses, *International Journal on Computational Geometry & Applications*, Vol. 10, No. 4, pp. 417-440, 2000.

[3] K. Shimada, J. Liao, T. Itoh, Quadrilateral Meshing with Directionality Control through the Packing of Square Cells, *7th International Meshing Roundtable*, pp. 61-76, 1998.

[4] N. Viswanath, K. Shimada, T. Itoh, Quadrilateral Meshing with Anisotropy and Directionality Control via Close Packing of Rectangular Cells, *9th International Meshing Roundtable*, pp. 227-238, 2000.

[5] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes," *Computer Aided Design*, Vol.10, No.6, pp. 350-355, 1978

[6] D. Doo and M. Sabin, "Analysis of the behavior of recursive division surfaces near extraordinary points," *Computer Aided Design*, Vol.10, No.6, pp. 356-360, 1978

[7] K.W. Fleischer, et al., "Cellular texture generation," *Proceedings of SIGGRAPH '95*, pp. 239-248, 1995

[8] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Communications of the ACM*, Vol. 25, No. 6, pp. 371-384, 1982

[9] B. Grünbaum and G.C. Shephard, *Tiling and Patterns*, W.H. Freeman and Co., New York, 1987

[10] C. Loop, "Smooth subdivision surfaces based on triangles," Master's thesis, University of Utah, Department of Mathematics, 1987

[11] K. Mehlhorn and S. Näher, LEDA: a platform for combinatorial and geometric computing, pp. 686-707, 1999

[12] K. Miyata, "A method of generating stone wall patterns," *Proceedings of SIGGRAPH '90*, pp. 387-394, 1990

[13] C.I. Yessios, "Computer drafting of stones, wood, plant and ground materials," *Computer Graphics*, Vol.13, No.2, pp. 190-198, 1979

[14] G. Turk, "Generating textures for arbitrary surfaces using reaction-diffusion," *Proceedings of SIGGRAPH '91*, pp. 289-298, 1991

[15] A. Witkin and M. Kass, "Reaction-diffusion textures," *Proceedings of SIGGRAPH '91*, pp. 299-308, 1991

[16] D.R. Fowler, H. Meinhardt, and P. Prusinkiewicz, "Modeling seashells," *Proceedings of SIGGRAPH '92*, pp. 379-388, 1992

[17] G. Turk, "Re-tiling polygonal surfaces," *Proceedings of SIGGRAPH '92*, pp. 55-64, 1992

[18] R Szeliski and D Tonnesen, Surface modeling with oriented particle systems, *Proceedings of SIGGRAPH '92*, pp. 185-194, 1992

[19] L. H. Figueiredo, J. M. Gomes, D. Terzopoulos, and L. Velho, Physically-based methods for polygonization of implicit surfaces, *Proceedings of Interface '92*, pp. 250-257, 1992

[20] A. P. Witkin and P. S. Heckbert, Using particles to sample and control implicit surfaces, *Proceedings of SIGGRAPH '94*, pp. 269-277, 1994

[21] F. Bossen and P.S. Heckbert, A pliant method for anisotropic mesh generation, *Proceedings of 5th International Meshing Roundtable*, pp. 63-74, 1996

[22] J. Dorsey and P. Hanrahan (1996), "Modeling and rendering of metallic patinas," *Proceedings of SIGGRAPH '96*, pp. 387–396. 1996

[23] J. Dorsey, et al (1999), "Modeling and Rendering of Weathered Stone," *Proceedings of SIGGRAPH '99*, pp. 225-234. 1999

[24] H.W. Jensen, J. Legakis, and J. Dorsey, Rendering Wet Materials, *Proceedings of Tenth Eurographics Workshop on Rendering*, Granada, Spain, pp. 273-282, 1999

[25] S. Worley, A Cellular Texture Basis Function, *Proceedings of SIGGRAPH '96*, pp.191-294, 1996

[26] K. Perlin, An image synthesizer, *Proceedings of SIGGRAPH '85*, pp. 287-296, 1985