Dual Poisson-Disk Tiling: An Efficient Method for Distributing Features on Arbitrary Surfaces

(minor revision version)

Hongwei Li lihw@cse.ust.hk
Kui-Yip Lo csfelix@cse.ust.hk
Man-Kang Leung cskang@cse.ust.hk
Chi-Wing Fu cwfu@cse.ust.hk

The Hong Kong University of Science and Technology

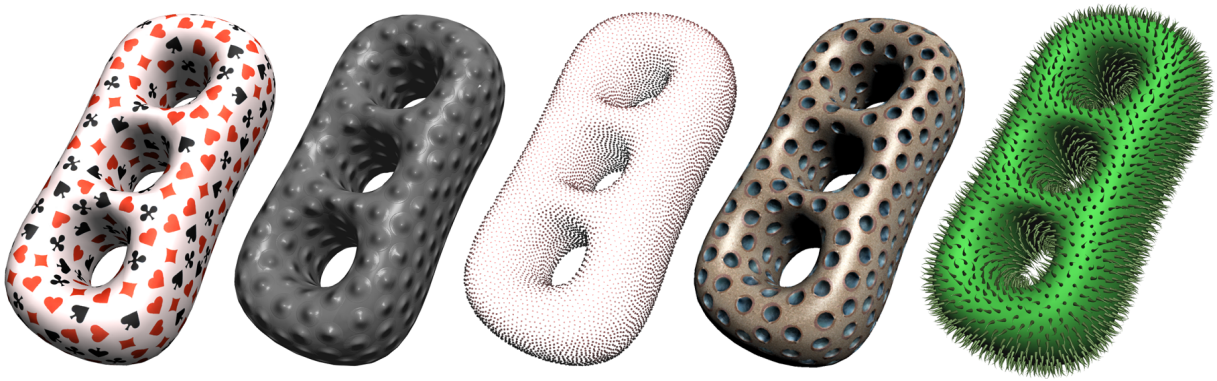
Hongwei Li
Department of Computer Science and Engineering,
The Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong.
Email: lihw@cse.ust.hk

Kui-Yip Lo
Department of Computer Science and Engineering,
The Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong.
Email: csfelix@cse.ust.hk

Man-Kang Leung
Department of Computer Science and Engineering,
The Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong.
Email: cskang@cse.ust.hk

Chi-Wing Fu (* Corresponding author)
Department of Computer Science and Engineering,
The Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong.
Tel: +852-2358-6991 Fax: +852-2358-1477 Email: cwfu@cse.ust.hk / cwfu@acm.org

Dual Poisson-Disk Tiling: An Efficient Method for Distributing Features on Arbitrary Surfaces



Surface modeling applications: Distributing different kinds of features on HOLES3.

Abstract

This paper introduces a novel surface-modeling method to stochastically distribute features on arbitrary topological surfaces. The generated distribution of features follows the Poisson disk distribution, so we can have a minimum separation guarantee between features and avoid feature overlap. With the proposed method, we not only can interactively adjust and edit features with the help of the proposed Poisson disk map, but can also efficiently re-distribute features on object surfaces.

The underlying mechanism is our dual tiling scheme, known as the Dual Poisson-Disk Tiling. First, we compute the dual of a given surface parameterization, and tile the dual surface by our specially-designed dual tiles; during the pre-processing, the Poisson disk distribution has been pre-generated on these tiles. By dual tiling, we can nicely avoid the problem of corner heterogeneity when tiling arbitrary parameterized surfaces, and can also reduce the tile set complexity. Furthermore, the dual tiling scheme is non-periodic, and we can also maintain a manageable tile set. To demonstrate the applicability of this technique, we explore a number of surface-modeling applications: pattern and shape distribution, bump-mapping, illustrative rendering, mold simulation, the modeling of separable features in texture and BTF, and the distribution of geometric textures in shell space.

Index Terms

I.3.7 Three-Dimensional Graphics and Realism, I.3.5 Computational Geometry and Object Modeling, I.3.8 Applications

I. INTRODUCTION

Surface modeling is a significant element in computer graphics. It helps to increase the surface complexity on 3D objects, and hence improve the visual realism in the rendering. Since most 3D objects we employed nowadays are still surface-based, surface modeling is yet a highly significant issue to address at present.

In principle, surface modeling can be categorized into the following three major approaches: texture mapping approach, geometry approach, and shell mapping approach. The texture mapping approach, pioneered by Catmull [4], defines a mapping between a 2D rectangular image and a 3D target model, and maps surface data, such as colors [4], normals [2], the bidirectional texture functions (BTF) [10], [53], [62], the irradiance field in shell texture function [5], the BSSRDF in sub-surface light transport [22], [52], and even volumetric features such as fur [24], [34], onto the related model surface. Unlike texture mapping, the geometry approach [17], [1], [28], [63] truly constructs geometric details on model surfaces, and thus can avoid problems such as the non-uniformity in surface map, texture aliasing, and texture filtering; however, due to the complexity of the introduced geometric details, the amount of geometry primitives to be processed by the graphics hardware could be excessively large. The shell mapping approach [45], [47], [16], [46] bridges the gap between the above two approaches by attaching a shell volume on the model surface. By using this shell volume as a volumetric space for mapping, we can precisely map geometric models onto the object surface as if we perform texture mapping in a continuous volumetric fashion. Hence, we can attain higher flexibility and efficiency in the surface modeling process, and employ both discrete and continuous surface data originated from the texture mapping and geometry approaches at the same time.

A. Motivation

This paper explores the use of *Wang tiling* [55], [56] and the *Poisson disk distribution* to enrich the surface modeling capability provided by the above three approaches. We introduce the *Dual Poisson-Disk tiling* scheme to stochastically distribute features on surfaces of 3D models. The generated distribution pattern follows the Poisson disk distribution and our method can work on parameterized surfaces of arbitrary topology. To demonstrate the applicability of our method, we explore several different surface modeling applications, covering the three surface modeling approaches mentioned above; related rendering results are presented in Section V.

Since we focus on tiling parameterized surfaces of arbitrary topology, the tiling mechanism is quite different as compared to tiling on planar domain. The vertex (tile corner) incidence on the surface parameterization grid is no longer a constant, i.e., 4; rather, the incidence could be 3, 5, or even 6. To overcome this *tile corner heterogeneity* problem, if we simply exhaust all possible corner, edge, and interior tile decompositions, e.g., by extending the tiling scheme in [31], we could end up creating an enormous tile set that is too complicated for management. On the other hand, if we just employ conventional Wang tiling [48], [8] or the template Poisson disk tiling methods [30], which promises a very small tile set, we could not properly handle the surface topology and maintain distribution correctness across tile boundaries.

B. Contributions

This paper presents a novel tiling method, called the *Dual Poisson-Disk Tiling*, to handle surfaces of arbitrary topology. Here we list the major contributions of this paper:

1) The major contribution of this paper is the Dual Poisson-Disk Tiling scheme; this dual tiling idea makes tile-based Poisson disk distribution applicable to parameterized surfaces of arbitrary topology. Its advantages include: 1) The proposed dual tiling scheme can nicely avoid the tile corner heterogeneity problem and allows us to properly tile parameterized surfaces of arbitrary topology; 2) The size of the tile set is relatively small and manageable; 3) It allows us to efficiently layout tiles on parameterized surfaces non-periodically.
2) Secondly, we propose a rendering mechanism called the *Poisson disk map*; it enables interactive feature editing through the support of GPU fragment programming.
3) Lastly, our distribution method brings in a wide range of applications, especially in the area of surface modeling, see Section V for the demonstration.

C. Related work

The distribution pattern we employed in our surface modeling applications is the *Poisson disk distribution*. Due to its blue noise properties [60], [61], [9], it is generally accepted as one of the best distribution pattern for sampling, and is widely used in computer graphics: anti-aliasing [12], [9], [40], ray tracing [41], and primitive distribution in illustrative rendering [11].

In general, the Poisson disk distribution is a uniform distribution of points, where all point-to-point distances are no less than a certain limit. In other words, if we put a circular disk of

radius equal to half of this limit at each distributed point, we can guarantee no overlapping disks. Based on this property, if we use the Poisson disk distribution as a mean to distribute features on surfaces, we can effectively avoid feature overlap.

Generating the Poisson disk distribution Traditionally, the generation of the Poisson disk distribution is based on the *dart throwing algorithm* [9]. Points are successively placed in a finite area at random, and a point is accepted only if the point-to-point distance limit is not violated. Since this algorithm requires numerous amount of checking, it is computationally very expensive. To address this issue, some faster algorithms [12], [40], [41], [26] have been proposed for speedup, and in particular, McCool and Fiume [39] developed an improved version of dart throwing by gradually reducing the distance limit. *Lloyd's relaxation* method [36] is further employed to optimize the point distribution by jittering the generated points towards their corresponding Voronoi centroids. At present, research in speeding up the dart throwing algorithm is still very active; Jones [23] applied Voronoi diagram; Dunbar and Humphreys [14] applied the idea of scalloped sectors, to represent available regions for dart throwing; White [59] improved the traditional dart-throwing algorithm by using quadtree subdivision on the sampling domain.

The Wang tile approach Stam [50] was the first in applying *Wang tiles* [55], [56] to computer graphics; by synthesizing matchable texture patterns on Wang tiles, textures can be arranged on two-dimensional manifolds in a non-periodic manner. Furthering this idea, Hiller et al. [21] invented a new approach to generate the Poisson disk distribution using tiling. Rather than generating one point at a time, they employed Wang tiles as point set containers with the Poisson disk distribution pre-synthesized on tiles; thus, we can match color-coded edges between Wang tiles, and quickly generate a tile-based Poisson disk distribution on a 2D domain. Cohen et al. [48], [8] furthered this approach and presented a stochastic tiling algorithm so that we can efficiently generate the Poisson disk distribution. Recently, Kopf et al. [27] proposed the recursive Wang tile method so that we can control distribution density at different spatial scales. In addition to two-dimensional domain [50], [21], [48], [8], [57], Wang tiles have been generalized to handle spaces such as the three-dimensional domain using Wang cubes [49], [37], [33], [38] and arbitrary topological surface using signed Wang tiles [19], [35]. Other than Wang tiling, Ostromoukhov et al. [43] applied Penrose tiling to distribute points for importance sampling.

In addition, Ostromoukhov [42] proposed using polyomino subdivision to generate distributions with blue-noise properties.

The Poisson disk tiles Lagae and Dutré [29] developed a procedural approach for efficiently evaluating the Poisson disk distribution. The proposed *Poisson disk tiles* is a variation of Wang tiles, where the square tile region is divided into three disjoint parts: corner tiles, edge tiles, and interior tiles. These three kinds of tiles are pre-generated sequentially conforming to the Poisson disk distribution, and they are finally re-assembled together to tile a given 2D domain by color matching. Since corner and edge regions are introduced in this tiling scheme, we can improve the distribution quality, particularly around corner regions.

In addition to Poisson disk tiles, Lagae and Dutré [30] invented another interesting tiling scheme, called the *Template Poisson disk tiles*. It is a toroidal version of Wang tiles with one single edge color in the entire tile set, so that we can create a single tileable frame (with margin border only) and synthesize additional tiles by using the same frame as the boundary condition.

Recently, Lagae and Dutré [31] proposed an alternative tile-based method to generate the Poisson disk distribution; it is a *corner-based tiling* scheme with color-coded corners. Hence, we can further improve the distribution quality around corner regions, see [32] for a detailed analysis and comparison of various methods that generate the Poisson disk distribution in 2D. Moreover, Lagae and Dutré [33] also explored their tile-based schemes in three-dimensional space and proposed the generation of Poisson sphere distributions by tiling Wang cubes.

Wang tiles on arbitrary surfaces Fu and Leung [19], [35] generalized the conventional Wang tiling method using signed Wang tiles, and made texture tiling applicable to arbitrary topological surfaces. However, since this tiling scheme does not take corner heterogeneity into account, this scheme cannot be used to generate the Poisson disk distribution on arbitrary surfaces.

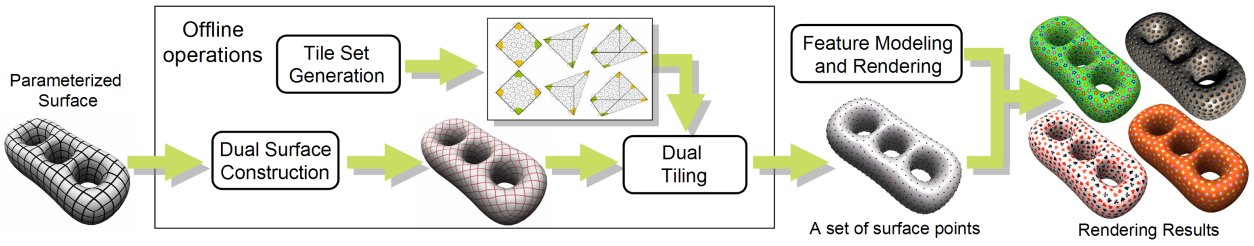


Fig. 1. Overview of our surface modeling method.

D. Overview

Figure 1 outlines our surface modeling method. It involves four main tasks in total, and takes a parameterized surface as its input. Any surface parameterization method [15], [18], [7], [3], [51], [20], [25], [13] can be used to create this input as long as the given parameterization is quad-based and has low distortion in the surface mapping.

Following this input, our first task is to compute the dual of the parameterized surface and pass the generated dual surface to the stochastic tiling task later on. However, unlike other offline tasks, the tile set generation is surface independent, so we can pre-construct a dual Poisson-disk tile set that stores the Poisson disk distribution. Section III details the generation process, while Section IV introduces the stochastic tiling task for arranging the generated tiles on dual surfaces. Finally, based on the distributed points on object surface, we can construct the Poisson disk map, and implement various surface modeling applications, see Section V.

II. The Dual Surface

As pointed out in Section I-A, when tiling a parameterized surface of arbitrary topology, tile corners could be shared by three, four, five, or even six tiles [13], see Figure 2; we cannot properly ensure distribution correctness across tile corners if we simply apply conventional tiling schemes on parameterized surfaces of arbitrary topology. To address this problem, our first strategy is to create a *dual surface*, so that we can make every tile corner to be shared exactly by 4 tiles all over the surface. By this means, we can avoid complex cases like irregular tile corner regions with 3, 5 or even more incident tiles, and thus can significantly reduce the tile set size because we only have one form of corner tiles in our scheme, see Section III.

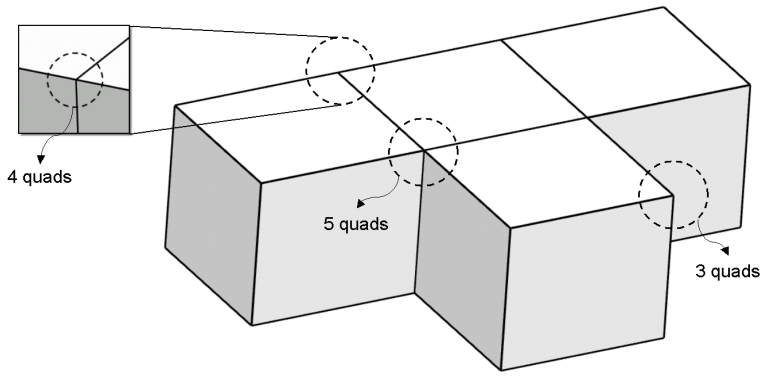


Fig. 2. The tile corner heterogeneity.

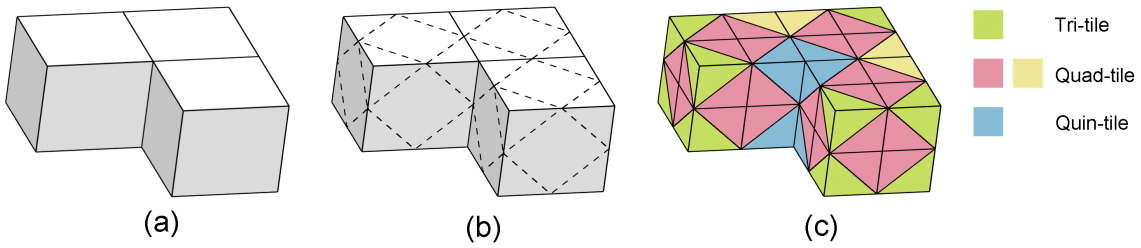


Fig. 3. (a) Input surface; (b) We connect neighboring edge midpoints on the input surface to generate the dual surface. The dotted lines are edges of the dual surface; (c) The dual surface is composed of Tri-tiles, Quad-tiles, and Quin-tiles.

To construct dual surfaces, we first take edge midpoints in the input surface parameterization as vertices of the dual surface to be generated. Then, we connect each midpoint to every neighboring midpoint on the same subquad belonging to the original surface parameterization, see Figure 3 for an illustrative example. After this dual transform, we have three kinds of interior tiles in our tile set: *Tri-tile*, *Quad-tile*, and *Quin-tile*, corresponding to the interior regions bounded by 3, 4, and 5 tile corners, respectively, but now, we only have one kind of corner junction over the entire model surface. Depending on the surface parameterization, we somehow could have *Sex-tile* for interior regions bounded by 6 tile corners.

III. Constructing a Dual Tile Set

The dual Poisson-disk tile set is generated using the corner-edge-interior decomposition [29], see Figure 9 for an example tiling. With the dual transform, *we only have one form (shape) of corner tiles (and edge tiles)*, while interior tiles can take several forms: Tri-tile, Quad-tile, Quin-tile, etc. The size of corner and edge tiles are determined by the Poisson disk radius, so that any Poisson disk placed on the interior tiles cannot affect any other disk in neighboring interior tiles. Hence, we first create corner tiles, and then edge tiles in our tile set construction, and employ these tiles to generate Tri-tiles, Quad-tiles, Quin-tiles, etc.

Furthermore, unlike regular Wang tiles that have a fixed orientation, we have to allow tile rotation in our case because we work on surfaces of arbitrary topology. Therefore, *our corner tiles, edge tiles, and interior tiles are all rotatable*, and the tile matching condition is determined not only by the *tile colors*, but also by the *tile orientations* relative to corner tiles as well.

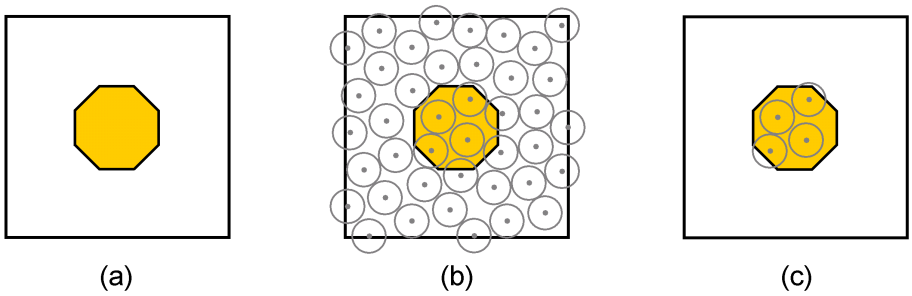


Fig. 4. The construction of a corner tile.

A. Constructing Corner Tiles

Similar to the idea of colored corners in [31], we arrange colors at corners, rather than on edges. Here we have C colors and hence C different corner tiles. Each corner tile takes the shape of a regular octagon with side length exactly equal to $2R$, where R is the Poisson disk radius. Figure 4 shows the corner tile construction procedure. Firstly, we employ *relaxation dart throwing* to dissipate N points onto a unit square region, and *Lloyd's relaxation* to optimize the distribution. Then, an octagon region is cut out of the distribution to form a corner tile; by independently repeating this construction process C times, we can generate all C corner tiles.

To guarantee a good distribution quality in the final tiling, we further have to compare these C corner tiles after the synthesis, and re-generate any corner tile (among the C corner tiles) that is too similar in distribution as compared to others in the corner tile set.

B. Constructing Edge Tiles

After creating corner tiles, the next step is to generate edge tiles. Figure 5 shows the edge tile construction procedure. Here we first put two corner tiles at the two opposite ends of an edge tile to be generated (Figure 5(a)). Then, we constrain the point distribution by the points pre-generated inside the two corner tiles, and we apply relaxation dart throwing to fill the remaining empty area. Next, we carry out a number of Lloyd's relaxation to fine-tune the distribution (Figure 5(b)). Note that during the dart throwing and relaxation, new points are prohibited from entering the corner regions, so that we can ensure proper distribution matching while matching corner tiles and edge tiles in the tiling. Finally, we extract the points inside the edge tile region and produce an edge tile. The thickness of an edge tile is $2R$.

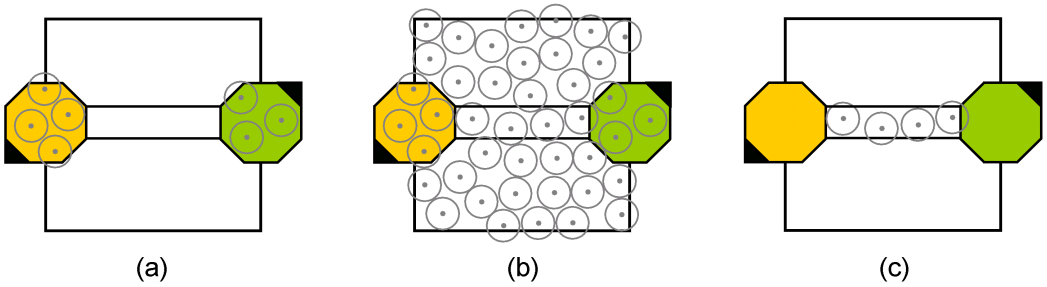


Fig. 5. The construction of an edge tile. The black triangles attached to the corner tiles indicate the corner-tile orientation.

Rotational Symmetry in Edge Tiles Since corner tiles are rotatable, each corner tile could have four possible axis-to-axis orientations relative to the edge tile being constructed. Hence, we could have as many as $(4C)^2$ different edge tiles in total. However, because of rotational symmetry, some edge tiles are in fact *equivalent in terms of tileability*. Figure 6 shows an example edge tile set with one single corner color; we label top, bottom, left, and right sides of a corner tile as T, B, L, and R, respectively; for instance, edge 1 and edge 11 shown in the figure is a symmetric pair, while edge 3 is self-symmetric.

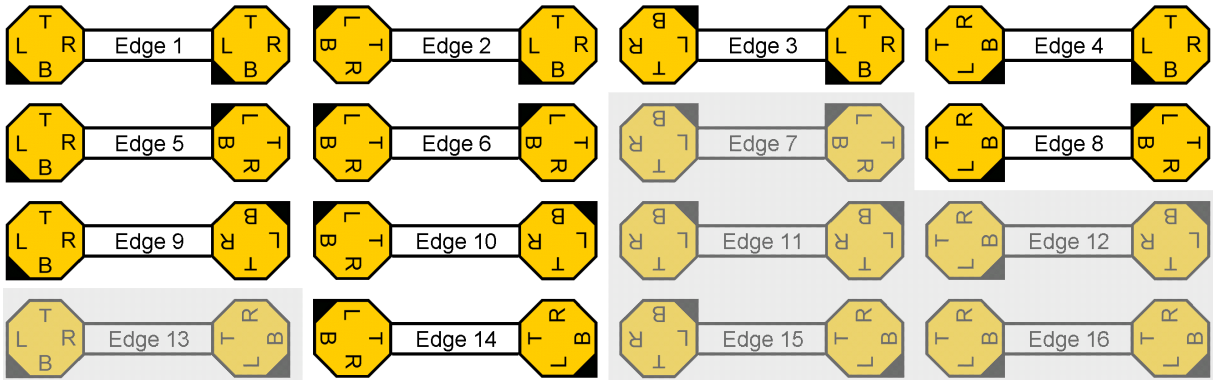


Fig. 6. Rotational symmetry in edge tiles; we have six symmetric pairs: 1 and 11, 2 and 15, 4 and 7, 5 and 12, 10 and 13, as well as 6 and 16, whereas the other four edges (3, 8, 9, and 14) are self-symmetric. Thus, there are totally 10 unique edges.

Since our tile matching scheme is based solely on *corner color* and *corner orientation*, we have to enforce edge tile uniqueness, so that after we layout all corner tiles on the dual surface, we can uniquely determine which edge tile to be placed in-between each pair of corner tiles, as well as which interior tile to be placed inside a ring of corner tiles. Otherwise, we will have to add additional color code (diversity) to edge tiles, and escalate the tile set complexity and size. In this way, within the 16 possible combinations shown in the figure above, we have to determine the symmetric pairs and generate only one edge tile for each pair found.

Further than that, we need to specially handle self-symmetric edges, e.g., edge 3 shown in the figure. *The point distribution inside a self-symmetric edge tile has to be rotationally symmetric*. Otherwise, we could have two ways in placing these edge tiles in a tiling, and escalate the tile set complexity; this is similar to tagging additional colors on edges and expanding the tile set size. Note that to construct a self-symmetric edge tile, we have to constrain the initial dart throwing and the Lloyd's relaxation process to enforce rotational symmetry during the tile construction.

C. Constructing Interior Tiles

After corner tiles and edge tiles, we can move on to create the interior tiles: Tri-tiles, Quad-tiles, Quin-tiles, etc.

Quad-tiles Figure 7 shows the construction procedure of Quad-tiles. Similar to edge tiles, we first assemble an appropriate configuration of 4 corner tiles and pick up related edge tiles to form a frame (Figure 7(a)). Then, we apply relaxation dart throwing followed by Lloyd's relaxation to generate the point distribution in the interior region (Figure 7(b)). Again, new points cannot enter the corner and edge regions during the dart throwing and relaxation process, while points originally in the corner and edge tiles are fixed throughout the construction process.

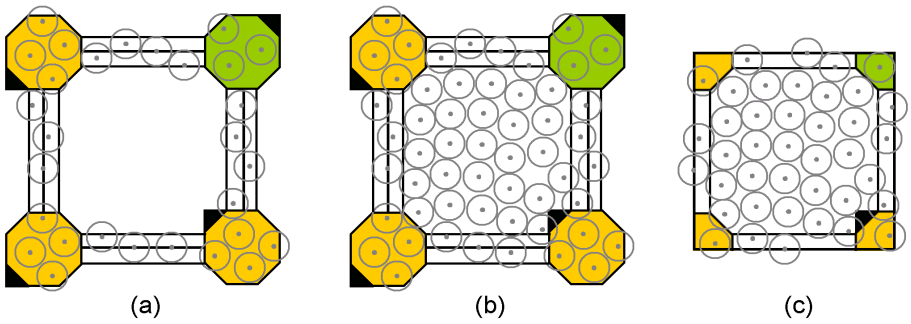


Fig. 7. The construction of a Quad-tile. The black triangles attached to the corner tiles indicate the corner-tile orientation.

Furthermore, we also explore rotational symmetry in interior tiles and generate only unique interior tiles. However, it is worth to note that removing duplicated interior tiles, e.g., Quad-tiles, is not as critical as removing duplicated edge tiles. For edge tiles, if we ignore tile uniqueness, we could escalate the tile set complexity because we need more interior tiles to match different edge tiles introduced, even when the colors and orientations of the boundary corner tiles are the same; but for interior tiles, since the points added inside the interior tiles never affect points

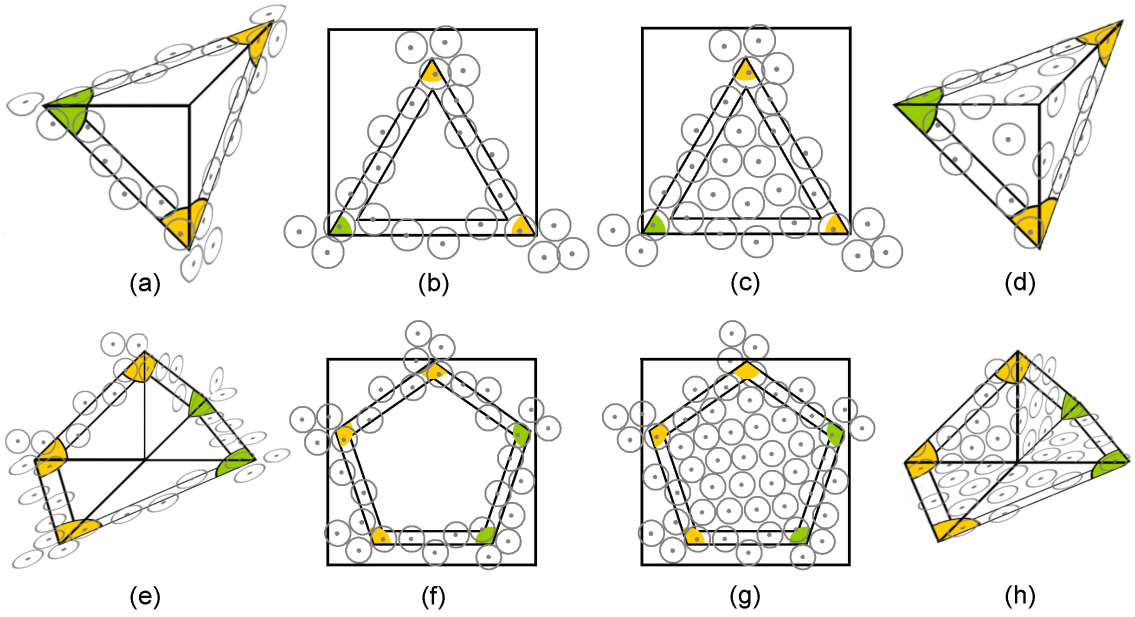


Fig. 8. The construction of a Tri-tile (above) and a Quin-tile (below); the warping method is employed here.

elsewhere in any other tile (different from points in the edge tiles), having duplicated interior tiles will increase the tile set size, but not the tile set complexity.

Tri-tiles and Quin-tiles As compared to Quad-tiles, the construction of Tri-tiles and Quin-tiles are more complicated because these tiles are intrinsically non-planar, see Figures 8(a) and (e). To address this problem, we propose two methods to generate non-planar interior tiles:

- The first method employs *geodesic distance* and jitters the distributed point samples directly on the non-planar domain. After we make up a tile frame using corner tiles and edge tiles, see Figures 8(a) and (e), we apply dart throwing in the interior area, and employ geodesic distances to construct a Delaunay triangulation for all points distributed so far, including also the points on the corner tiles and edge tiles. Finally, we apply Lloyd's relaxation in this non-planar domain and fine-tune the point distribution inside.
- The second method is based on the observation that when we arrange non-planar tiles such as Tri-tiles and Quin-tiles on object surfaces, the surface regions covered by these tiles are usually highly smooth and planar. Hence, if we employ the first approach to arrange point samples with same density as that on Quad-tiles, we could end up having a relatively denser point distribution on the surface regions covered by non-planar tiles. Therefore, we propose to warp the non-planar region to a planar domain, distribute and relax points directly on

it, and finally map the distributed points back to the object surface. In detail, we employ equilateral triangle and regular pentagon, as the warped domains for Tri-tiles and Quin-tiles, respectively, see Figures 8(b) and (c), and *barycentric coordinate mapping* to perform the warping. Furthermore, like what we have done for Quad-tiles, we also forbid any newly added point to enter the corner tile and edge tile regions in the warped domain during the tile construction process. Also, we can construct the Sex-Tiles in a similar way, but rather than using an equilateral triangle or pentagon as in the cases of the Tri-tiles and Quin-tiles, respectively, we use a regular hexagon on the planar domain.

D. Size of a dual Poisson-disk tile set

In this subsection, we examine the size of a complete dual Poisson-disk tile set; the *Cauchy-Frobenius lemma* [58] (also known as the orbit-counting theorem) is used in the analysis, so that we can take rotational symmetry into consideration, and determine the number of unique edge tiles and interior tiles in a dual Poisson-disk tile set.

Edge tiles Let E be the set of all possible edge tile configurations without considering rotational symmetry, i.e., $|E| = (4C)^2$. Let G be a finite group that acts on E. For each $g \in G$, let E^g denote the set of elements in E that are fixed by g. For edge tiles, we have two g's in G: one is an identity, say g_0, and hence, $E^{g_0} = (4C)^2$; the other one is an 180-degree rotation (in 2D), say g_1, and we have $E^{g_1} = 4C$ because only four edge tiles per color (the two neighboring corner tiles having the same color) are identical after an 180-degree rotation. Therefore, by using the Cauchy-Frobenius lemma, the number of unique edge tiles under rotational symmetry is:

$$|E/G| \;=\; \frac{1}{|G|}\sum_{g\in G}|E^g| \;=\; \frac{1}{2}((4C)^2 \;+\; 4C) \;=\; 8C^2 \;+\; 2C\ .$$

Interior tiles For Quad-tiles, we have four possible operations: an identity, a 90-degree clockwise rotation, an 180-degree clockwise rotation, and a 270-degree clockwise rotation. By counting the number of fixed elements under each of them, we can again apply the Cauchy-Frobenius lemma to find out the number of unique Quad-tiles under rotational symmetry:

$$|Q/G| \;=\; \frac{1}{|G|}\sum_{g\in G}|Q^g| \;=\; \frac{1}{4}((4C)^4 \;+\; 4C \;+\; (4C)^2 \;+\; 4C) \;=\; 64C^4 \;+\; 4C^2 \;+\; 2C\ ,$$

where G is a finite group acting on the full Quad-tile set Q. Using a similar method, we can also compute the number of unique Tri-tiles and Quin-tiles under rotational symmetry. Table I summarizes the size of a dual Poisson-disk tile set. Note that we normally do not need a full set of Quin-tiles; a relatively small amount of them are sufficient for stochastic tiling, see Section IV.

TABLE I

THE SIZE OF A DUAL POISSON-DISK TILE SET.

Num. of Colors	Num. of Corner Tiles	Num. of Edge Tiles	Num. of Tri-tiles	Num. of Quad-tiles	Num. of Quin-tiles
1	1	10	24	70	208
2	2	36	176	1044	6560
3	3	78	584	5226	49776
C	C	$8C^2 + 2C$	$\frac{1}{3}(64C^3 + 8C)$	$64C^4 + 4C^2 + 2C$	$\frac{16}{5}(64C^5 + C)$

Comparing to standard tiling scheme Without using dual tiling, conventional tiling schemes, for example, the corner-edge-interior decomposition, will result in different kinds of corner tiles because we work on surfaces of arbitrary topology. A corner tile could meet with three, four, five, or even six edge tiles unlike the case we have in dual tiling. As a result, the set of corner tiles will become more complex even we consider only a small number of colors in the tiling. Note that these corner tiles need to be rotatable; hence, such a complexity could further affect the size of the edge tile set and the interior tile set as well.

For instance, if C is the number of colors we have for each kind of corner tiles, that is, we have C 3-corner tiles, C 4-corner tiles, and C 5-corner tiles, etc. When we consider edge tiles, we have to consider several kinds of edge tiles: edge tiles joining two 3-corner tiles, edge tiles joining one 3-corner tile and one 4-corner tile, etc. It unavoidably complicates the edge tile set as well as the interior tile set that follows. Moreover, it will also significantly increase the tile set size as well. By dual tiling, we can avoid such a complexity and keep a compact corner tile set; hence, we can also maintain manageable sets of edge tiles and interior tiles.

IV. THE DUAL TILING ALGORITHM

A. The Dual Tiling Algorithm

We have the following steps in applying our dual Poisson-disk tile set to generate the Poisson disk distribution on dual surfaces; *stochastic tiling* [31] is used in our algorithm.

- First of all, we properly assign Quin-tiles available in our tile set onto the Quin-tile slots on the dual surface. Since conventional surface parameterization methods normally produce very few 5-corner (as well as 6-corner) junctions (as compared to 4-corner junctions) on the surface parameterizations, we normally do not have many Quin-tile slots (as well as Sex-tiles) on the dual surface, as compared to Quad-tile slots. Therefore, a relatively small amount of Quin-tiles (and Sex-tiles) are sufficient in our stochastic tiling. Furthermore, in extreme cases where Quin-tile slots dominate, we can make use of one or two corner colors to generate a full set of Quin-tile set (See Table I); then, we can still guarantee an agreeing tiling.
- After arranging Quin-tiles, we randomly assign colors and orientations to the remaining corner tile slots on the dual surface. Since we have a complete set of Tri-tiles and Quad-tiles, we do not need to constrain the corner tile assignment as we did for Quin-tiles.
- Finally, based on the arranged corner tiles, we can lookup related edge tiles and then Quad-tiles and Tri-tiles to fill the remaining space on the dual surface.

In our actual implementation, we develop a tile-reassembling scheme to facilitate the tiling process. After we generate the whole dual tile set, we expand the region of each interior tile to include parts of the related corner and edge tiles. Since interior tiles and edge tiles are unique, given any interior tile, we can uniquely determine its boundary corner and edge tiles, pack half of the related edge tiles and a quarter of the related corner tiles around it, and generate a reassembled tile for each interior tile, see Figure 7(c), Figure 8(d), and Figure 8(h) for a reassembled Quad-tile, Tri-tile, and Quin-tile, respectively. In this way, we can replace corner tiles, edge tiles, and interior tiles by the reassembled tiles in our implementation, and hence simplify our data structure, and make the tiling more convenient.

B. Tiling Examples

Figure 9 presents a tiling example to exemplify how we tile a non-planar surface in the form of a T-shape brick with our dual Poisson-disk tiles. Here we have $C = 2$ and $N = 32$, where C is the number of colors (corner tiles) and N is the point distribution density. In this example, we can see that we have four Quin-tiles and twelve Tri-tiles, whereas all corner tiles meet exactly four edge tiles over the entire model surface due to dual tiling.

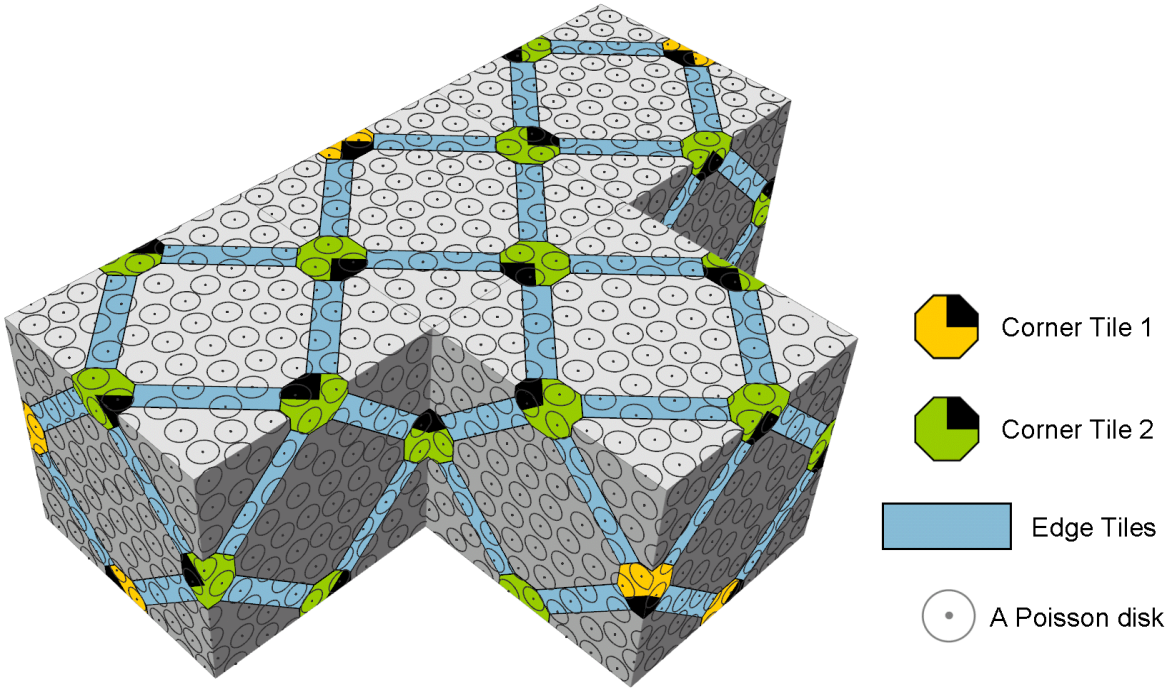


Fig. 9. A tiling example – Generating the Poisson disk distribution on a dual parameterized surface in the form of a T-shape brick; $C = 2$ and $N = 32$. The black quadrants on the corner tiles indicate the corner-tile orientation.

In addition, we also present the tiling of a two-dimensional plane using our tiling scheme, see Figure 10. Since the tiling domain is now planar, we only need corner tiles, edge tiles, and Quad-tiles, but not Tri-tiles and Quin-tiles. Moreover, the corner tiles can be placed at the grid junctions, rather than at the edge mid-points. Furthermore, since only corner tiles, edge tiles, and Quad-tiles are involved, this 2D tiling structure is very similar to the colored corner approach proposed in [31]; however, our corner tiles are rotatable and our tiling is dual-based, so that our method can handle parameterized surfaces of arbitrary topology.

V. Implementation, Analysis and Applications

A. Implementation issues

The Poisson disk map To support interactive control on distributed features, we developed the idea of *Poisson disk map*. It is basically a texture map defined on a parameterized surface, with each texel storing three values: (d, θ) is the polar coordinate of the pixel center measured from the nearest distributed feature point on object surface using geodesics (see Figure 11 for an example Poisson disk map), and *feature index* could be a unique feature ID associated with

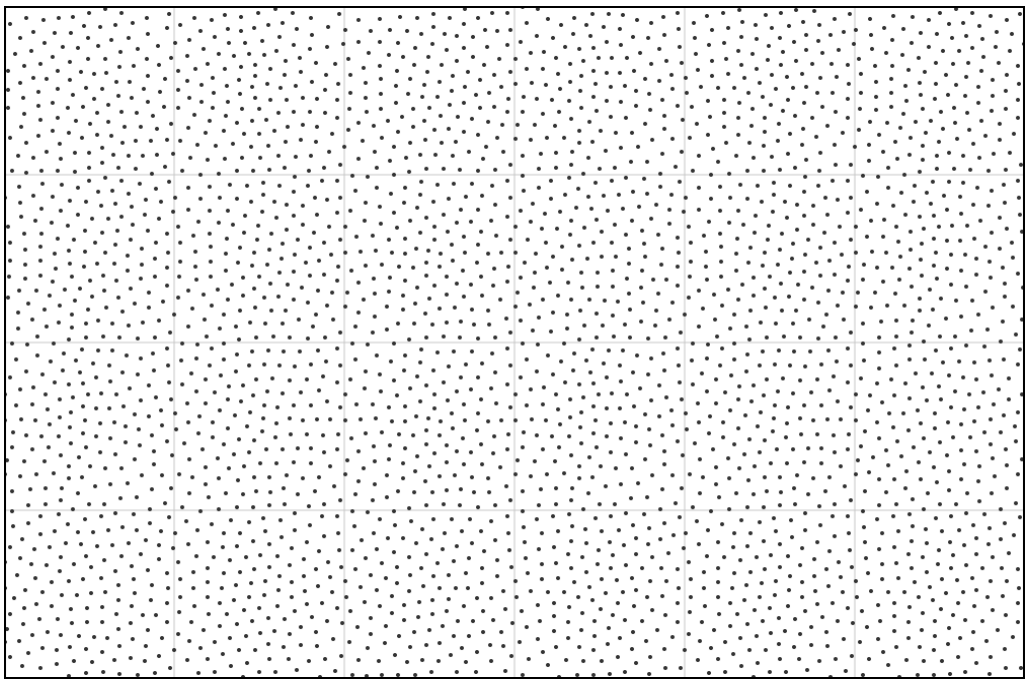


Fig. 10. A tiling example – Generating the Poisson disk distribution on a two-dimensional plane; $C = 2$ and $N = 128$.

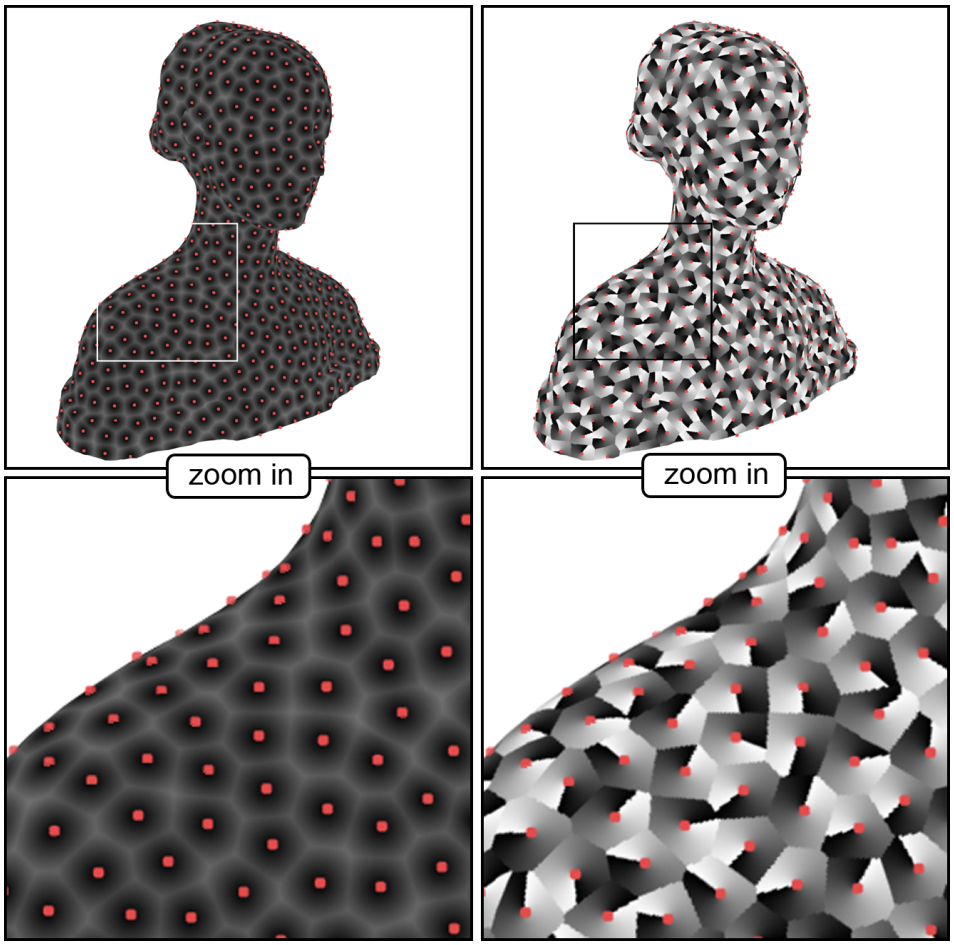


Fig. 11. An example Poisson disk map: the distance component d (left) and angular component θ (right) on LAURANA. Note that there are 874 points distributed on LAURANA, and the reference direction for θ is picked randomly at each point.

that feature point or an integer denoting the color index resulted from k-coloring. Note that in the figure, red dots are additionally drawn on the LAURANA surface to indicate the location of feature points, and also, by using k-coloring, we can assign multiple colors and patterns to the distributed features and ensure no two neighboring features share the same color or pattern.

In the renderings, we employ fragment programs on the GPU: 1) to retrieve (d, θ) and feature index for each pixel fragment, and 2) to compute the texture coordinates on the corresponding feature texture if the pixel fragment falls within an effective Poisson disk from a feature point. The advantage of having a Poisson disk map is that we can interactively deform and modify the features, for example, scaling or re-orientating the features without needing to build any form of static geometries for them, as well as editing the feature textures and reloading the feature appearance on objects interactively at run-time.

Distribution refinement on object surface With our tiling method, we can precisely guarantee a valid Poisson disk distribution on parameterized surfaces, and this distribution guarantee can be carried to the object surface, provided the surface parameterization does not introduce any distortion. But as one might expect, since surfaces of 3D models are seldom flat, this is, however, rarely the case in practice. To address this issue so that we can produce a valid Poisson disk distribution on object surface, certain refinement process is unavoidable after the tiling. In our current implementation, we first refine the Delaunay triangulation (from the corresponding Voronoi diagram) mapped from the parameterized surface to the object surface, so that the resultant connectivity information can be truly defined on the object model; note that geodesics must be computed for connected edges on high curvature regions so as to ensure connectivity correctness. Then, we can precisely apply Lloyd's relaxation locally at each distributed point and jitter them towards their Voronoi centroids on the object surface. Since the surface parameterization we used are of low distortion, the entire refinement process takes no more than a few minutes (around $10-20$ iterations, depending on the number of distributed points and the parameterization complexity) to finish on a commodity PC: an HP xw4400 workstation with Intel Core(TM)2 CPU 6400 at 2.13GHz and 1GB Memory, see Table II for the timing result. Also, to fairly compare the refinement time against different 3D objects, and also against different number of distributed points, we use 20 iteration steps in all testing cases shown. Furthermore, note also in the table that the number of points in parameterization denotes the model complexity.

TABLE II

TIMING ON THE REFINEMENT PROCESS.

3D models	Tile set	Total number of distributed points	Number of points in parameterization	Total time for refinement (sec.)
HOLES3	$C = 2, N = 16$	3142	25596	5.4
	$C = 2, N = 32$	6360	25596	9.0
	$C = 2, N = 64$	12724	25596	19.2
LAURANA	$C = 2, N = 16$	7039	14587	9.6
	$C = 2, N = 32$	14082	14587	19.2
	$C = 2, N = 64$	27986	14587	39.0
BUNNY	$C = 2, N = 16$	10121	82434	15.0
	$C = 2, N = 32$	20380	82434	28.8
	$C = 2, N = 64$	40983	82434	58.8

Now, one may argue why we employ tiling to generate the Poisson disk distribution on 3D models, while after-tiling refinement is still needed. In fact, if we do not use tiling, we have to do dart throwing and Lloyd's relaxation to distribute points on object surfaces. It is a one-point-at-a-time process, and could be exceedingly time-consuming. By tiling, we can efficiently initialize a point distribution that is nearly a Poisson disk distribution, and this distribution can, in turn, be readily refined on object surface within a few iteration steps.

B. Examining the Generated Distribution

Spectral Analysis One common way to examine the quality of Poisson disk distributions is by using periodogram [54]. Here we experiment six different color and point density settings in our dual Poisson-disk tiling, and generate around 16,384 points on a 2D plane for each Poisson disk distribution being analyzed, see Figure 12. In addition, we also generate one Poisson disk distribution purely using dart throwing as a reference. From the figure, we can see that increasing the number of colors and distribution density can improve the distribution quality, which means that the resultant distribution could become less periodic. Moreover, if we compare the periodograms generated by our method against those by other tile-based methods [32], our method has one crucial difference; even with the same number of color code, we have additional variation in generating the Poisson disk distribution because our tiles are rotatable.

Radius Statistics In addition, we also study the Poisson disk distributions that generated on surfaces using radius statistics [31]. Here we first have to compute two radius data: R_{max} and r,

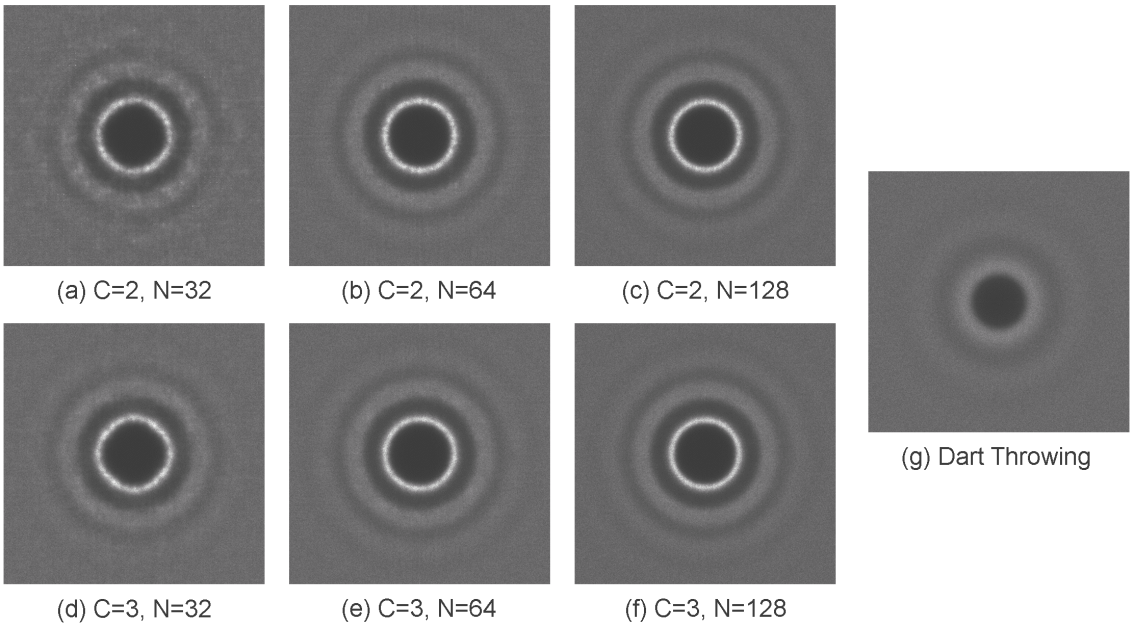


Fig. 12. Periodograms of Poisson disk distributions generated with dual Poisson-disk tiles (a-f) and dart throwing (g).

where R_{max} is an upper bound of the Poisson disk radius given the total surface area and the distribution density, whereas r is the Poisson disk radius we have in the generated Poisson disk distribution.

$$R_{max} = (2\sqrt{3}K/S)^{-1/2} ,$$

where S is the total surface area and K/S is the distribution density. Here r is half of the shortest distance among neighboring point pairs distributed on the object surface. Furthermore, radius statistics measure the ratio $\rho = r/R_{max}$ to estimate the regularity of the generated distributions. A small ρ indicates an uneven distribution, while a large ρ indicates a regular distribution. According to Section 3.1 in [32], Poisson disk distributions should have a relative radius that is large ($\rho > 0.65$), but not too large ($\rho > 0.90$). Furthermore, it is worth noting that since the equation for estimating R_{max} is originally for 2D planar domain, we have to assume sufficiently large point sets in our experiments, so that R_{max} can serve as a good upper bound approximation even on non-planar surfaces. In practice, we can obtain ρ values of around 0.3 to 0.6 (on surfaces) after the tiling depending on the quality of the surface parameterization, and can quickly optimize ρ to be around 0.7 to 0.8 after distribution refinement. Table III presents some related results for HOLES3, BUNNY, and LAURANA before and after relaxation, while Figure 13 shows the generated distributions on BUNNY before and after relaxation. Note that we also compute m_{before}

and m_{after} (see the table); m refers to the average half-distance between neighboring point pairs on the object surface.

TABLE III

RADIUS STATISTICS.

	Tile set	R_{max}	r_{before}	r_{after}	m_{before}	m_{after}	ρ_{before}	ρ_{after}
HOLES3 ($S = 1.7258$)	$C = 2, N = 16$	0.0128	0.0077	0.0100	0.0131	0.1300	0.6052	**0.7653**
	$C = 2, N = 32$	0.0089	0.0050	0.0067	0.0091	0.0090	0.5677	**0.7558**
	$C = 2, N = 64$	0.0063	0.0033	0.0048	0.0064	0.0064	0.5310	**0.7674**
BUNNY ($S = 2.3775$)	$C = 2, N = 16$	0.0085	0.0031	0.0063	0.0086	0.0086	0.3660	**0.7371**
	$C = 2, N = 32$	0.0060	0.0016	0.0044	0.0061	0.0061	0.2591	**0.7358**
	$C = 2, N = 64$	0.0042	0.0012	0.0031	0.0043	0.0042	0.2749	**0.7315**
LAURANA ($S = 1.5414$)	$C = 2, N = 16$	0.0082	0.0029	0.0061	0.0083	0.0083	0.3594	**0.7363**
	$C = 2, N = 32$	0.0057	0.0024	0.0042	0.0058	0.0057	0.4219	**0.7342**
	$C = 2, N = 64$	0.0041	0.0015	0.0030	0.0041	0.0041	0.3658	**0.7339**

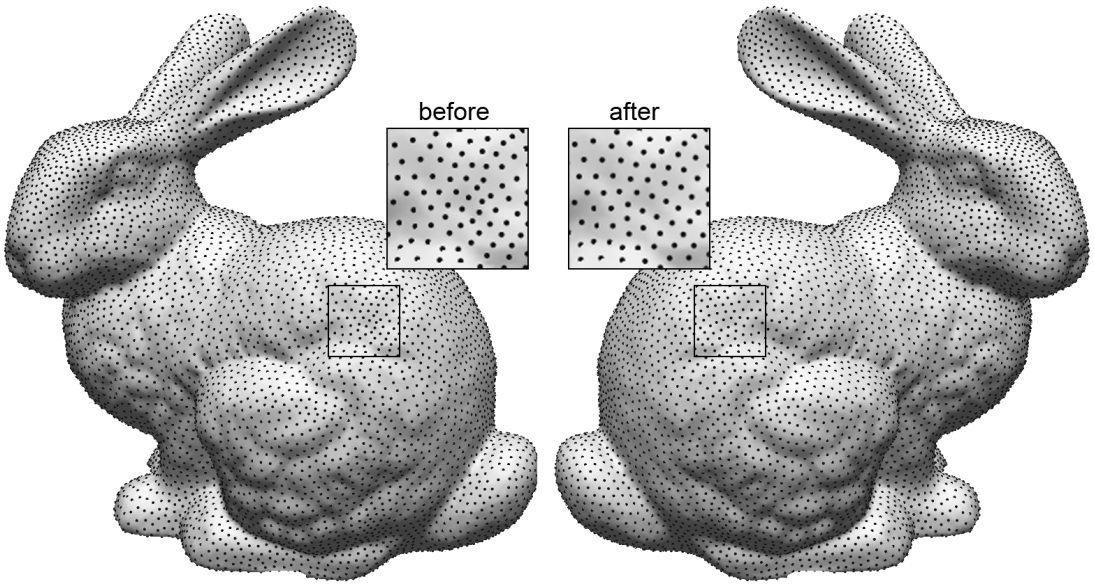


Fig. 13. An example showing the distributions before and after the relaxation; the tile set in use is Bunny, $C = 2, N = 16$.

C. Applications in Surface Modeling

Figures 14-19 present the rendering results of five different surface modeling applications we briefly explored:

Pattern and Shape distribution The most straightforward application of our tiling method is to distribute texture-based features like patterns and shapes on 3D models; Figure 14 demonstrates three different kinds of patterns distributed on HOLES3, BUNNY, and LAURANA. By storing patterns and shapes as small image textures, we can apply fragment program to lookup texture values based on the per-fragment values, (d, θ), sampled from the Poisson disk map. Furthermore, since the image textures in use are for texture lookup only, we can interactively modify, deform, and reload the texture-based patterns during the program run-time. Similarly, we use small bump maps (normal maps) storing distorted surface normals instead of colors, and apply a modified fragment program to lookup distorted surface normals. In this way, we can shade the object surfaces with bumpy appearance (see the bumpy BUNNY, LAURANA, and HOLES3 in Figure 15).

Illustrative rendering To create the illustrative renderings shown in Figure 16, points are first distributed onto the object surface according to our tiling method; compared to other applications, we need far more points in illustrative rendering; in detail, they can be generated by subdividing the dual surface before the tiling, i.e., more tiles on the initial parameterized surfaces. For the examples shown in the figure, $27,986$ and $40,983$ points are generated on LAURANA and BUNNY, respectively. In addition, it is worth to note that since all distributed points have well-defined 3D locations on object surface (in object space), we can easily attain frame-to-frame coherency in the renderings subject to viewpoint changes.

Mold Simulation Moreover, we also apply our distribution method to simulate mold growth. Similar to illustrative rendering, we first distribute a dense set of points on object surface, and compute the accessibility [6] (like ambient occlusion) at each distributed point in object space. Note that to compute the accessibility, we put a hemisphere aligned with the surface normal at each distributed point, and randomly send out hundreds of rays (over the hemisphere) from the distributed point. The accessibility is the percentage of rays that are not blocked by the object itself. Hence, the lower accessibility, the higher the chance (density) for mold to appear on that specific location. Hence, we apply the accessibility map as a filter to assign colors and transparency to distributed points, and simulate the mold growing process. Figure 17 shows molded HOLES3 and BUNNY.

BTF synthesis Furthermore, we can also apply our tiling method to arrange the bidirectional texture functions (BTF) by keying BTF-based features on object surfaces. To create the BTF renderings shown in Figure 18, we first associate a BTF hole data with each distributed point

on the object surface. Then, we can synthesize the BTF on the unfilled surface area on the 3D model by using constrained texture synthesis in the parameterized space. Hence, we can have BTF elements defined for all pixels on the parameterized surface, and hence, can render the model surface under variable illumination and viewing directions.

However, we have to be very careful when transforming the per-fragment viewing and lighting vectors; the TBN transform [2], [44] has to align with the parameterized grid (rather than the local surface curvature) so that we can correctly lookup the surface reflectance defined in the BTF data space. Furthermore, it is worth to note that the advantage of using tiling for BTF is that we only need to keep a very small BTF data for the entire 3D model; in the rendering examples shown, the BTF hole data only has spatial resolution: 128×128 (29. 3MB in total, without spatial compression among BTF data elements).

Geometric Texture and Shell mapping Lastly, we explored the distribution of geometric textures in shell space of object surfaces. Since the generated Poisson disk distribution can guarantee non-overlapping disks (or cylinders in shell space), we can see from the rendering results in Figure 19 that the distributed thorns and pyramids (geometric features) never collide with each other on object surfaces.

VI. Conclusion

In conclusion, this paper presents a practical and efficient solution for tiling the Poisson disk distribution on arbitrary topological surfaces. Since tileable Poisson disk distributions have been pre-synthesized on dual Poisson-disk tiles, we can efficiently tile points on an input parameterized surface, and readily distribute features and patterns on the corresponding 3D model. This is the first paper exploring the use of tiling to generate the Poisson disk distribution on parameterized surfaces of arbitrary topology. Using dual tiling, we can carefully avoid the tile corner heterogeneity problem, so that we only have one kind of corner tiles even when the parameterized surfaces being tiled is of arbitrary topology. Hence, we can precisely generate the Poisson disk distribution on parameterized surfaces, and also on their corresponding 3D models after distribution refinement. The proposed method can nicely address the surface topology, and can maintain a valid distribution with a manageable tile set.

To support interactive control on distributed features, we develop the idea of Poisson disk map based on GPU support. With the Poisson disk map, we can interactively deform distributed

features on object surface without incurring any feature geometry locally on object model (provided the feature is not a geometric texture). To demonstrate the applicability of this approach, a wide range of computer graphics applications, particularly in the area of surface modeling, are presented. Example applications that we have briefly explored in this paper include pattern and shape distribution, bump mapping, illustrative rendering, mold simulation, texture and BTF synthesis, and also the distribution of geometric textures in shell space.

Future work Other than the applications listed above, the proposed tiling method can bring in far more applications yet to be investigated: surface remeshing based on the distributed points, stochastic sampling of surface attributes, Voronoi partitioning on surfaces, approximate collision detection using surface point clouds, as well as stereological technique for textures.

ACKNOWLEDGMENTS

In this research work, we would like to thank Tarini et al. for the PolyCube-Mapped models, the MSRA for the BTF hole data and Lagae and Dutré for the valuable discussion.

REFERENCES

[1] Pravin Bhat, Stephen Ingram, and Greg Turk. Geometric texture synthesis by example. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 41–44, 2004.

[2] James F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH 78 Proceedings)*, volume 12, pages 286–292, Aug. 1978.

[3] Ioana Boier-Martin, Holly Rushmeier, and Jingyi Jin. Parameterization of triangle meshes over quadrilateral domains. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 193–203, 2004.

[4] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Department of CS, University of Utah, Dec. 1974.

[5] Yanyun Chen, Xin Tong, Jiaping Wang, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Shell texture functions. *ACM Transactions on Graphics*, 23(3):343–353, 2004.

[6] Siu chi Hsu and Tien-Tsin Wong. Simulating dust accumulation. *IEEE Computer Graphics and Applications*, 15(1):18–22, jan. 1995.

[7] Ulrich Clarenz, Nathan Litke, and Martin Rumpf. Axioms and variational problems in surface parameterization. *Computer Aided Geometric Design*, 21(8):727–749, 2004.

[8] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *ACM Transactions on Graphics*, 22(3):287–294, 2003.

[9] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986.

[10] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, Jan. 1999.

[11] Oliver Deussen, Stefan Hiller, Cornelius van Overveld, and Thomas Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(3):40–51, 2000.

[12] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. In *Computer Graphics (SIGGRAPH 85 Proceedings)*, pages 69–78. ACM Press, 1985.

[13] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM Transactions on Graphics*, 25(3):1057–1066, 2006.

[14] Daniel Dunbar and Greg Humphreys. A spatial data structure for fast Poisson-disk sample generation. *ACM Transactions on Graphics*, 25(3):503–508, 2006.

[15] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH 95*, pages 173–182, 1995.

[16] Gershon Elber. Geometric texture modeling. *IEEE Computer Graphics and Applications*, 25(4):66–76, Jul./Aug. 2005.

[17] Kurt W. Fleischer, David H. Laidlaw, Bena L. Currin, and Alan H. Barr. Cellular texture generation. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, ACM, pages 239–248, 1995.

[18] Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, pages 157–186. Springer Verlag, 2005.

[19] Chi-Wing Fu and Man-Kang Leung. Texture tiling on arbitrary topological surfaces using Wang tiles. In *Proceedings of Eurographics Symposium on Rendering 2005*, pages 99–104, Jun. 2005.

[20] Xiaohu Guo, Xin Li, Yunfan Bao, Xianfeng Gu, and Hong Qin. Meshless thin-shell simulation based on global conformal parameterization. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):375–385, 2006.

[21] Stefan Hiller, Oliver Deussen, and Alexander Keller. Tiled blue noise samples. In *Proceedings of Vision, Modeling, and Visualization Conference 2001*, pages 265–271, 2001.

[22] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of ACM SIGGRAPH 2001*, Annual Conference Series, ACM, pages 511–518, 2001.

[23] Thouis R. Jones. Efficient generation of Poisson-disk sampling patterns. *Journal of Graphics Tools*, 11(2):27–36, 2006.

[24] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In *Computer Graphics (SIGGRAPH 89 Proceedings)*, pages 271–280. ACM Press, 1989.

[25] Liliya Kharevych, Boris Springborn, and Peter Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics*, 25(2):412–438, 2006.

[26] R. Victor Klassen. Filtered jitter. *Computer Graphics Forum*, 19(4):223–230, 2000.

[27] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. Recursive Wang tiles for real-time blue noise. *ACM Transactions on Graphics*, 25(3):509–518, 2006.

[28] Ares Lagae, Olivier Dumont, and Philip Dutré. Geometry synthesis. In *SIGGRAPH 2004 Sketch*, August 2004.

[29] Ares Lagae and Philip Dutré. A procedural object distribution function. *ACM Transactions on Graphics*, 24(4):1442–1461, Oct. 2005.

[30] Ares Lagae and Philip Dutré. Template Poisson disk tiles. Tech. Rep. CW 413, May 2005. Katholieke Universiteit Leuven.

[31] Ares Lagae and Philip Dutré. An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics*, 25(4):1442–1459, 2006.

[32] Ares Lagae and Philip Dutré. A comparison of methods for generating Poisson disk distribution. Tech. Rep. CW 459, Aug. 2006. Katholieke Universiteit Leuven.

[33] Ares Lagae and Philip Dutré. Poisson sphere distributions. In *Proceedings of Vision, Modeling, and Visualization Conference 2006*, pages 373–379, 2006.

[34] Jerome Lengyel, Emil Praun, Adam Finkelstein, and Hugues Hoppe. Real-time fur over arbitrary surfaces. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 227–232, March 2001.

[35] Man-Kang Leung, Wai-Man Pang, Chi-Wing Fu, Tien-Tsin Wong, and Pheng-Ann Heng. Tileable BTF. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 13(5):953–965, 2007.

[36] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[37] Aidong Lu and David S. Ebert. Example-based volume illustrations. In *Proceedings of IEEE Visualization 2005*, pages 655–662, Oct 2005.

[38] Aidong Lu, David S. Ebert, Wei Qiao, Martin Kraus, and Benjamin Mora. Volume illustration using Wang cubes. 26(2), 2007. Article 11.

[39] Michael McCool and Eugene Fiume. Hierarchical Poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface 92*, pages 94–105. Morgan Kaufmann Publishers Inc., 1992.

[40] Don P. Mitchell. Generating antialiased images at low sampling densities. In *Computer Graphics (SIGGRAPH 87 Proceedings)*, pages 65–72. ACM Press, 1987.

[41] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. In *Computer Graphics (SIGGRAPH 91 Proceedings)*, pages 157–164. ACM Press, 1991.

[42] Victor Ostromoukhov. Sampling with polyominoes. *ACM Transactions on Graphics*, 26(3), 2007. Article No. 78.

[43] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3):488–495, Aug. 2004.

[44] Mark Peercy, John Airey, and Brian Cabral. Efficient bump mapping hardware. In *Proceedings of ACM SIGGRAPH 97*, Annual Conference Series, ACM, pages 303–306, 1997.

[45] Jianbo Peng, Daniel Kristjansson, and Denis Zorin. Interactive modeling of topologically complex geometric detail. *ACM Transactions on Graphics*, 23(3):635–643, 2004.

[46] Fabio Policarpo and Manuel M. Oliveira. Relief mapping of non-height-field surface details. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, pages 55–62, 2006.

[47] Serban D. Porumbescu, Brian Budge, Louis Feng, and Kenneth I. Joy. Shell maps. *ACM Transactions on Graphics*, 24(3):626–633, 2005.

[48] Jonathan Shade, Michael Cohen, and Don P. Mitchell. Tiling layered depth images, 2002. Technical Report, University of Washington, Seattle.

[49] Peter Sibley, Philip Montgomery, and G. Elisabeta Marai. Wang cubes for video synthesis and geometry placement. In *ACM SIGGRAPH 2004 Poster Compendium*, August 2004.

[50] Jos Stam. Aperiodic texture mapping. Technical Report R046, 1997. European Research Consortium for Informatics and Mathematics.

[51] Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. PolyCube-Maps. *ACM Transactions on Graphics*, 23(3):853–860, 2004.

[52] Xin Tong, Jiaping Wang, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Modeling and rendering of quasi-homogeneous materials. *ACM Transactions on Graphics*, 24(3):1054–1061, 2005.

[53] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics*, 21(3):665–672, 2002.

[54] Robert Ulichney. *Digital halftoning*. MIT Press, 1987. Cambridge, MA, USA.

[55] Hao Wang. Proving theorems by pattern recognition II. *Bell Systems Technical Journal*, 40:1–42, 1961.

[56] Hao Wang. Games, logic, and computers. *Scientific American*, pages 98–106, Nov. 1965.

[57] Li-Yi Wei. Tile-based texture mapping on graphics hardware. In *SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 55–63, 2004.

[58] Eric W. Weisstein. Cauchy-Frobenius Lemma (a.k.a. orbit-counting theorem). From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/Cauchy-FrobeniusLemma.html`.

[59] Kenric B. White, David Cline, and Parris K. Egbert. Poisson disk point sets by hierarchical dart throwing. In *IEEE Symposium on Interactive Ray Tracing, 2007*, pages 129–132, 2007.

[60] John I. Jr. Yellott. Spectral analysis of spatial sampling by photoreceptors: Topological disorder prevents aliasing. *Vision Research*, 22:1205–1210, 1982.

[61] John I. Jr. Yellott. Spectral consequences of photoreceptor sampling in the Rhesus retina. *Science*, 221:382–385, 1983.

[62] Kun Zhou, Peng Du, Lifeng Wang, Jiaoying Shi, Baining Guo, and Heung-Yeung Shum. Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):519–528, 2005.

[63] Kun Zhou, Xin Huang, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. Mesh quilting for geometric texture synthesis. *ACM Transactions on Graphics*, 25(3):690–697, 2006.

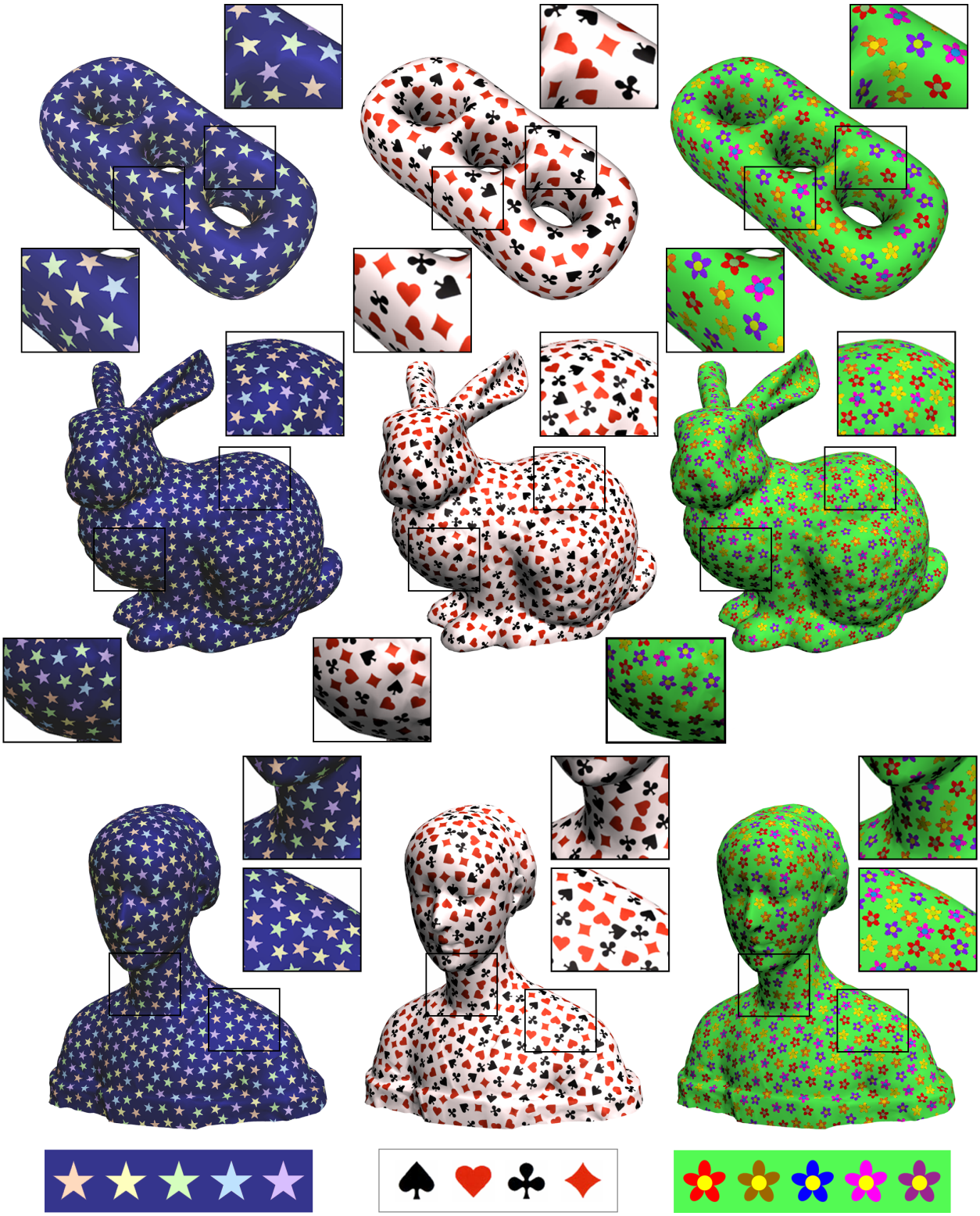


Fig. 14. Surface modeling application (1) — Pattern and shape distribution. Here we have 397, 1289, and 874 patterns distributed on HOLES3, BUNNY, and LAURANA (from top to bottom), respectively.

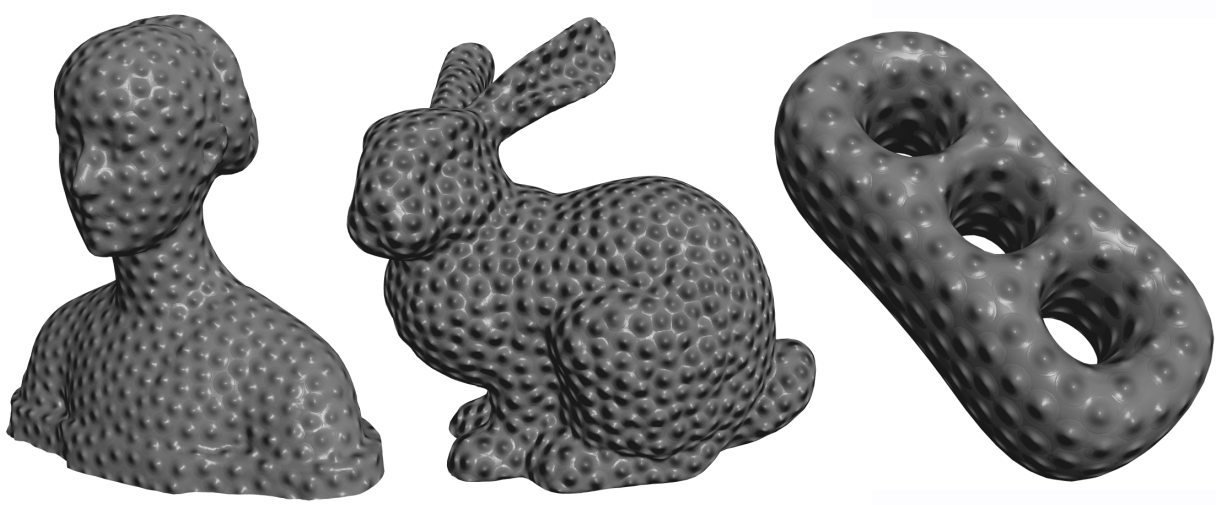


Fig. 15. Surface modeling application (1) — Bump mapping. In this example, we distribute 874, 1289, and 397 Gaussian bumps on LAURANA, BUNNY, and HOLES3 (from left to right), respectively.

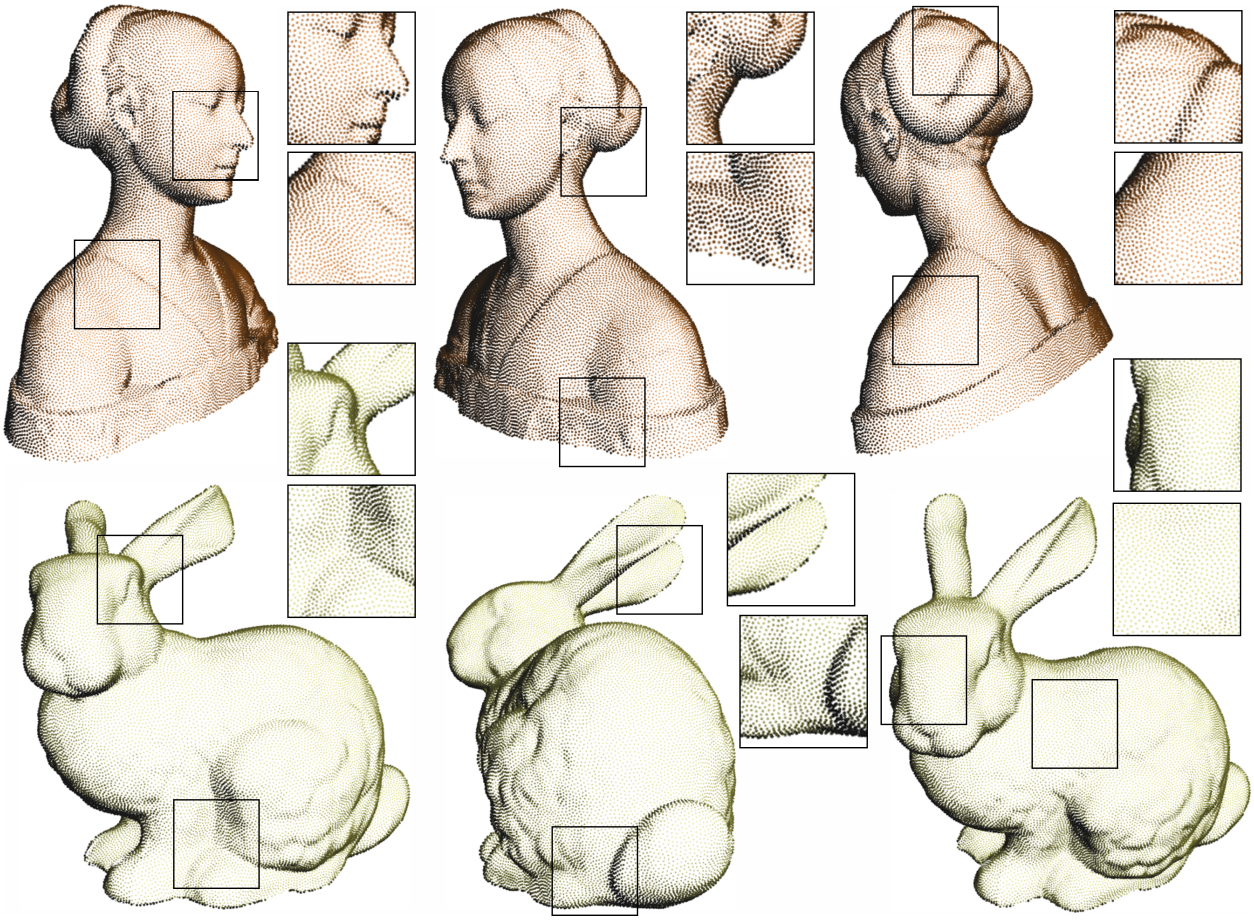


Fig. 16. Surface modeling application (2) — Illustrative rendering. To create the above renderings, we pre-generate 27986 and 40983 points on LAURANA (top) and BUNNY (bottom), respectively.

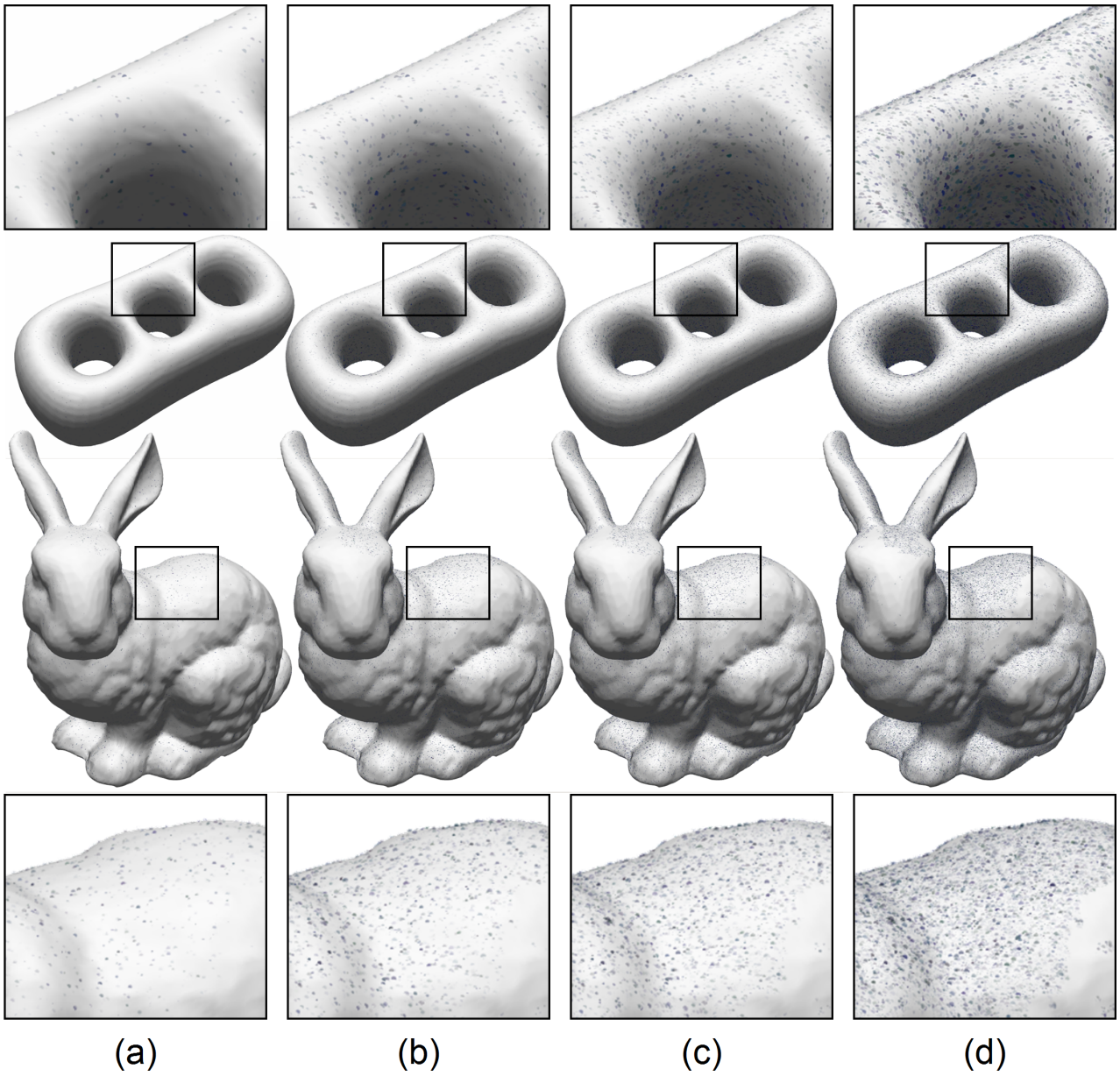


Fig. 17. Surface modeling application (3) — Mold simulation. The subfigures (a), (b), (c), and (d) show two sequences of mildewing on HOLES3 (top) and BUNNY (bottom). In this simulation, we pre-generate distributions of 102, 390 and 329, 573 points on HOLES3 and BUNNY, respectively.

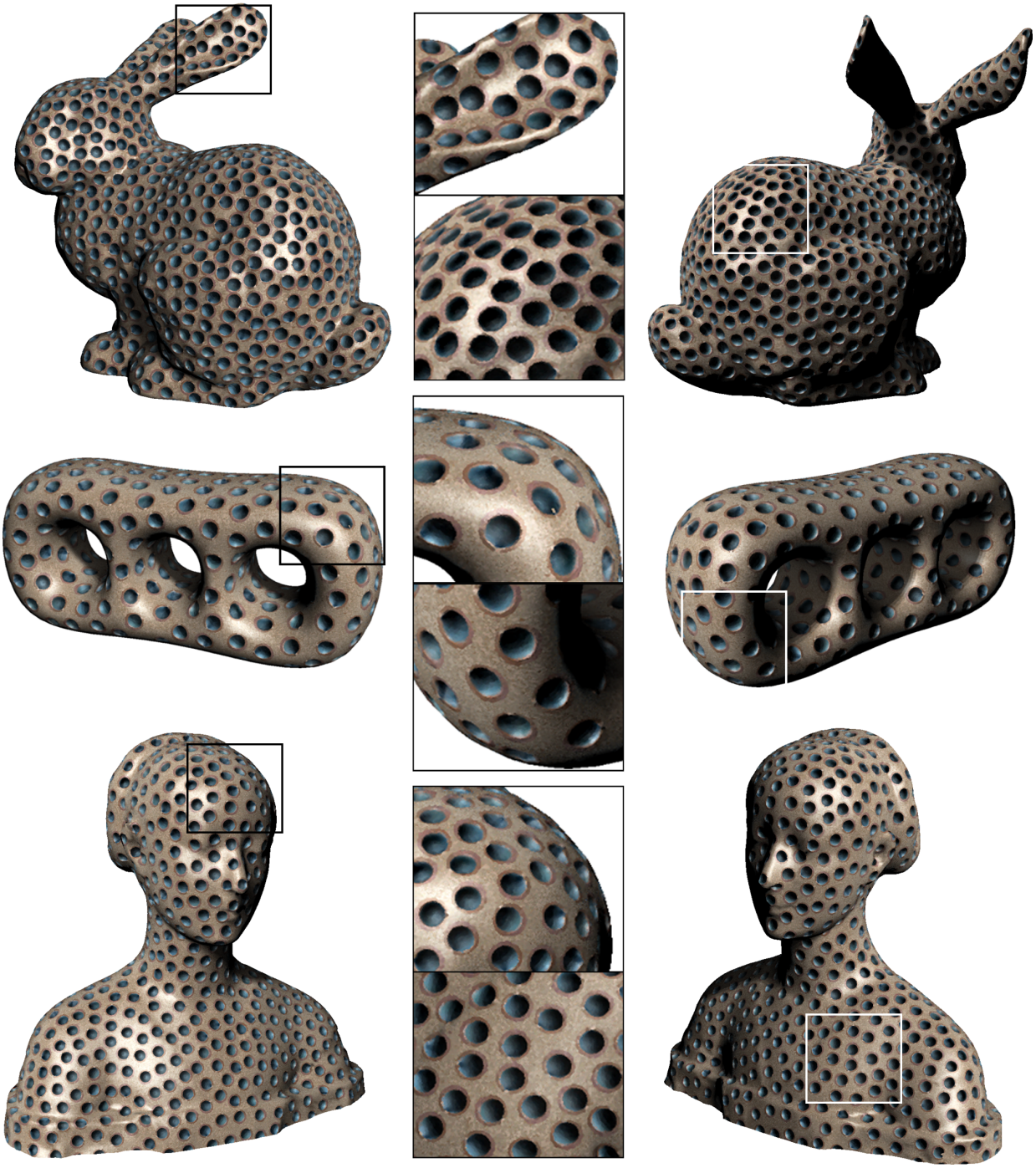


Fig. 18. Surface modeling application (4) — Distributing BTF-based features. We can quickly arrange holes (BTF-based feature) on object surface by keying them at each distributed point generated by our tiling method.

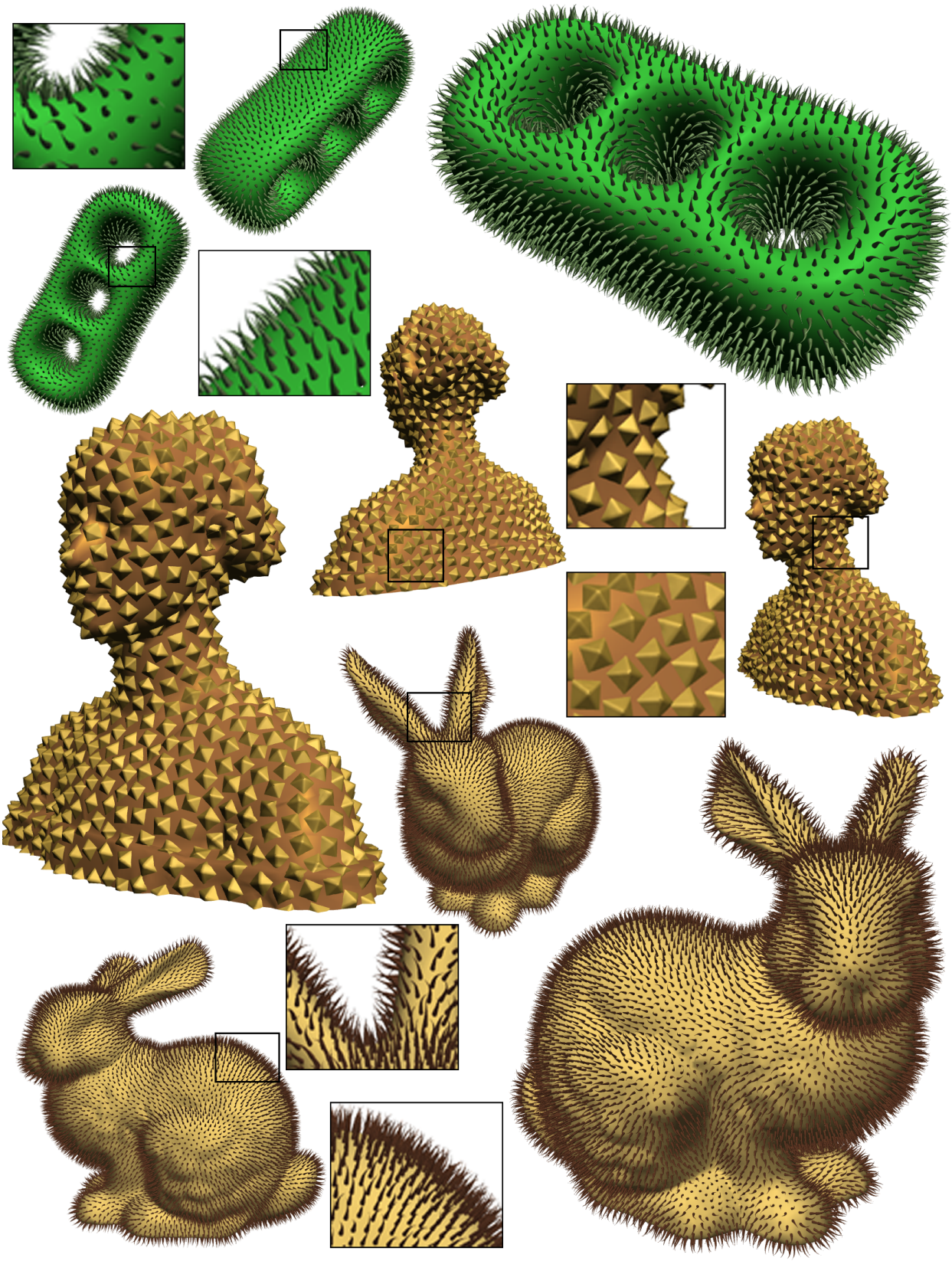


Fig. 19. Surface modeling application (5) — Geometric textures: thorns and pyramids in shell space. Note that we have 3142, 874, and 10121 geometric features placed on HOLES3, LAURANA, and BUNNY (from top to bottom), respectively.