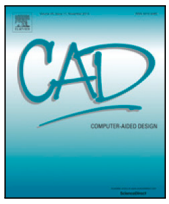
Computer-Aided Design 193 (2026) 104031

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad



Research Paper

Paver: Element-based pattern creation on 3D free-form surfaces

Weidan Xiong [a], Ziyu Hu [a], Yongli Wu [a], Peng Song [b], Jianmin Zheng [c,*]

[a] CSSE, Shenzhen University, China
[b] Singapore University of Technology and Design, Singapore
[c] Nanyang Technological University, Singapore

ARTICLE INFO

Keywords:
Parametric design
3D geometric pattern
Interactive tool

ABSTRACT

This paper presents *Paver*, an interactive tool for designing structured patterns composed of 3D elements on free-form 3D surfaces. To address the inherent complexity of direct pattern creation in 3D space, we introduce a parametric design model that allows users to define patterns on a 2D parameter plane. Each pattern is defined by an underlying tessellation and isometric elements. A bijective mapping then lifts these 2D designs onto the target input surface. To ensure structural coherence and accommodate surface curvature, we model the tessellation as a 2D mass–spring system and propose an optimization method to adjust the particle positions on the 2D parameter plane under well-designed system forces. Once optimized, the tessellation is transferred to the 3D surface, generating the final element-based pattern. Paver supports diverse pattern types and offers real-time feedback, enabling designers to iteratively explore and refine complex surface decorations with ease. Experimental results are presented to demonstrate the effectiveness of our *Paver*.

1. Introduction

The product design cycle often involves a sequence of steps, including creation, modification and optimization. Designers and engineers must consider the aesthetics and cost of products throughout design cycles, making the process complicated and time-consuming [1].

A notable example of architectural facade design is the Selfridges building in London, which is covered by 15,000 identical discs, each with a diameter of 60 cm [2]. In such cases, the regularity of both elements (shape, size, etc.) and placement pattern is crucial for the construction cost and design beauty. Various design problems in other domains, such as jewelry design and fiber mold design, are akin to the problem mentioned above, as shown in Fig. 1.

Such problems aim to position equivalent classes of rigid elements with 3D shapes over a 3D free-form surface, adhering to specific patterns and element-wise separation. Subsequently, these elements can be encoded onto the model surface in an additive or subtractive manner [3].

Computational methods for circle packing or surface sampling have been utilized to initiate points/spheres directly on free-form surfaces. However, existing research mainly addresses functional or engineering concerns, such as randomness, point density and object stiffness. Moreover, the heavy computation makes them unsuitable for interactive design. Another way is to design 2D textures and map them onto the surface via parameterization. However, current 2D texture design approaches mainly focus on the vector/tile-based/non-stationary textures [4–6], neglecting the structural distortion after mapping. The design and preservation of delicate regular patterns consisting of 3D rigid elements over 3D surfaces remains an open and challenging task.

This paper poses the problem of designing and placing 3D elements over a free-form surface with regular patterns. We choose to perform the design and optimization of structural patterns in a 2D parameter plane, and map the refined pavement onto the surface afterwards. The main contributions include the following:

- An interactive tool called *Paver*, which lets designers quickly create 3D structural patterns consisting of 3D elements over an input surface.
- A novel parametric model including grammar rules and interactive operators, which enables designers to quickly design element-based patterns.
- A numerical optimization method based on a 2D mass–spring system, which refines the 3D pavement such that the structure of the designed pattern is preserved and curvature-adaptive compactness is achieved.

* Corresponding author.
E-mail addresses: xiongweidan@gmail.com (W. Xiong), huziyubm05@gmail.com (Z. Hu), ngligo16@gmail.com (Y. Wu), peng_song@sutd.edu.sg (P. Song), ASJMZheng@ntu.edu.sg (J. Zheng).

https://doi.org/10.1016/j.cad.2026.104031
Received 27 July 2025; Received in revised form 7 October 2025; Accepted 7 January 2026
Available online 3 February 2026

Fig. 1. Left: Jewellery designs. A diamond bracelet with uniform hexagonal pavement (top, ©GlitzUK); A bracelet with a grid pattern (bottom, ©Yue Guang). Right: The free-form facade of the building of Selfridges in Birmingham, UK [11].

2. Related work

Circle/sphere packing. Extensive research has been conducted on the geometry of circle (disk) or sphere packings in a plane or a sphere [7], with extensions to surface geometries [8]. The mathematics of surface meshes endowed with circles or spheres is a topic of study within the field of geometry processing [9]. In this domain, circle packing metrics and discrete Ricci flow are employed to investigate discrete conformal mappings and parameterization for triangulated surfaces [10]. However, the goal of surface packing methods is to generate random patterns without considering visual regularity, largely overlooking the packing structure and its induced design, which is the opposite of the goal here.

Architectural facades often incorporate packing-like arrangements of circles or spheres on surfaces [12]. Schiftner et al. [13] explored the packing of spheres on free-form surfaces and its application in architectural design. They introduced a structure called circle packing mesh (CP mesh), which is a triangle mesh whose in-circles form a packing. However, the size of the optimized in-circles is not controllable, and the resulting pattern of circles is heavily influenced by the topology of the input surface mesh, thus unsuitable for design problems.

Surface sampling. Besides circle and sphere packings, there has been a lot of research on surface sampling [14], such as blue noise sampling [15,16] and Centroidal Voronoi Tessellation [17,18]. These methods are often employed as the initializations for sphere packing. Given the heavy computational expense of applying these sampling techniques to 3D surfaces, it usually does not satisfy the requirement for real-time interactions. Moreover, these sampling methods do not emphasize the creation of patterns or fail to meet aesthetic and functional constraints at the same time.

Pattern design. Lots of works and tools have been introduced for the design of patterns and textures in different applications and representations [4,6,19,20]. Ma et al. [21] presented a methodology for the creation of a dynamic simulation process based on a given 2D pattern. Bian et al. [5] proposed a tile-based method that synthesizes vector patterns with visual appearance and topological structures specified by users via a 2D drawing for physical manufacturing. While Jiang et al. [22] focused on the deformation of texture with polygonal topology, Peng et al. [23] considered the creation and optimization of checkerboard-style patterns. Song et al. [24,25] introduced a tool for the design of 3D self-supporting reciprocal frame structures by reducing the dimensionality and performing the design in 2D. Zheng et al. [26] proposed to engrave cylinders on thin-shell structures following a simple hexagonal or polygonal pattern via parameterization, while Hu et al. [27] presented a parametric design framework for engraving customized shapes following Central Voronoi Tessellation on thin-shell structures, ensuring structural stiffness.

In this work, we focus on element-based pattern design over 3D surfaces, as well as curvature-aware and structure-preserved density control for both aesthetic considerations and alleviating intersections among elements.

Mesh Parameterization. Surface parameterization targets the generation of a mapping between a suitable domain and the surface. Generation of a planar mapping to three-dimensional meshes is a fundamental task in computer graphics. Usually, the surfaces triangular meshes. Many parameterization algorithms have been developed. Conformal parameterization preserves the angles [28–30] and equal-area parameterization preserves the areas [31,32]. If a parameterization algorithm is both conformal and area-preserving, it is isometric, which means that it can flatten the surface to a plane without distortion. It is known in mathematics that most surfaces cannot be mapped to a plane without distortion, except for developable surfaces [33–35].

Our input surfaces are assumed to have an existing open border. We seek a parameterization method that can preserve the geodesic distance as much as possible. We implement a deformation method called ARAP [32,36], which keeps the transformation for the surface in each cell as rigid as possible. The ARAP algorithm applies to a manifold that is homeomorphic to a planar domain.

Mass–spring system. Consisting of masses connected by springs, the dynamics of such a system is governed by the forces exerted by the springs based on Hooke's law. The equations of motion can be solved using numerical integration methods, enabling efficient and stable real-time simulations [37]. Li et al. [38] proposed a variational integrator based on Incremental Potential Contact for modeling nonlinear elastic solids with guaranteed non-interpenetration. Existing methods widely explore the use of mass spring systems to simulate the behavior of deformable objects, such as cloth [39], hair [40], and elastic solids [41]. These physics-based simulation methods encode high computational efficiency, which is beneficial for interactive design. However, none of them simultaneously achieves specific compactness and structure-preserving patterns among masses, which are crucial for design tasks. In this work, we adapt the mass–spring system to the design task of generating element-based structural patterns on 3D free-form surfaces.

3. Overview

Given an open free-form surface, represented as a triangle mesh $M = (V, F)$ with vertices V and faces F, our target is to create a visually appealing and structural pavement design located on M. Different from conventional packing or sampling problems, we aim to develop a design tool that allows the user to easily design various pavements consisting of 3D elements over a 3D free-form surface. A straightforward approach is to perform this design process in a 3D space. Nevertheless, it is usually difficult due to the nature of the problem. Its high computational cost makes it less feasible for interactive pattern design applications.

This paper presents a new approach consisting of two stages: pavement creation and placement optimization. In the first stage, we introduce a parametric design model to let the designer create a pavement pattern in the 2D parameter plane (Section 4). The structure of a pattern is denoted as a mesh tessellation, and the centers of elements are designed to be located on the vertices and edges of the mesh. The created pavement will be inversely mapped back onto the input surface as a set of 3D points. These 3D points serve as the center points, at which the design elements (e.g., diamonds) are instantiated. In the second stage, we present an optimization method to refine the 3D pavement and adapt it to curvature variation (Section 5). The positions of the mesh tessellation vertices are optimized to alleviate element-wise collision (especially in high-curvature surface regions) and achieve pavement with curvature-adaptive compactness, while the topology of the mesh tessellation remains fixed. In Section 6, we perform experiments to show the performance of the proposed method.

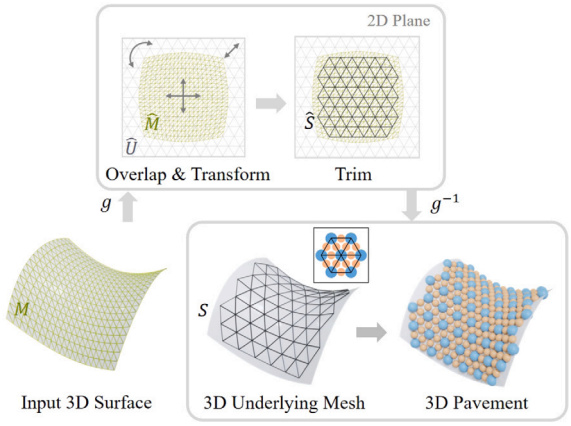


Fig. 2. The overview of generating a 3D pavement following hexagonal tessellation for a 3D saddle surface from the 2D parameter plane.

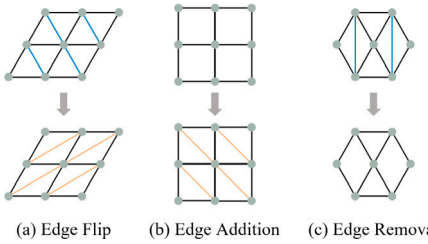


Fig. 3. Three types of interactive operations in the topology design.

4. Parametric pavement creation

In this section, we introduce a parametric model that enables users to create pavement by interactively specifying and varying a 2D underlying tessellation and elements (Section 4.1). After that, a fast initialization of 3D pavement is developed by lifting the 2D underlying mesh to 3D space and placing elements according to the design parameters (Section 4.2). Fig. 2 shows the creation of a pavement for a saddle surface. The user can modify the parameters, preview the appearance, customize the shape of the 3D elements and experiment with different design variations in our tool.

4.1. Interactive pattern design

Topology design. Pavement patterns often follow a triangular or polygonal tessellation. The user can initialize a tessellation by specifying the tessellation category and edge length D (black edge in Fig. 3). We introduce three types of interactive operations to extend these two basic 2D tessellations. Users can flip, add, or remove arbitrary edges to create various tessellations, as illustrated in Fig. 3. The resultant tessellation is then organized into an underlying mesh $\widehat{U}=(p)$ with vertices p (gray points).

Element design. Given a specified tessellation as an underlying mesh $\widehat{U}$, the user can initiate two types of elements: (i) one V-element locating on the vertex of $\widehat{U}$, and (ii) multiple E-elements locating along the edge of $\widehat{U}$. The user can adjust the position of E-element E_k along its locating edge $p_a p_b$ with a position parameter $t \in [0, 1]$, and adjust the diameter of both types with a scale parameter $s \in [0, 1]$.

The final settings will be propagated to the entire underlying mesh. With our parametric model, users can easily create a wide variety of aesthetic patterns, as shown in Fig. 4.

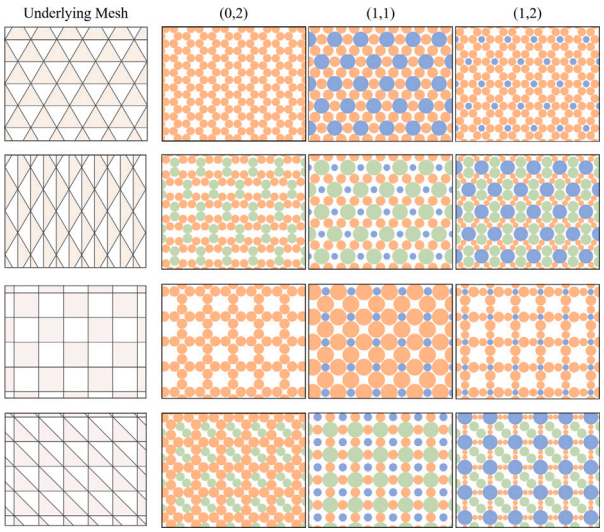


Fig. 4. Patterns created by varying the topology of the underlying mesh and the setting on elements (labels correspond to the number of elements located on the vertex and edge, respectively). The color of elements encodes the type (elements on the vertex are shown in blue, and elements on the edge are shown in orange and green). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.2. 3D pavement initialization

We establish a bijective mapping between the 2D plane and the 3D surface by parameterizing the input surface mesh M. For our task, we seek a parameterization method that can preserve the geodesic distance as much as possible. Among current methodologies, we find that as-rigid-as possible parameterization [32] algorithm delivers feasible bijective mapping $g(\cdot)$ between M and $\widehat{M}=(v, f)$. For a pair of mesh vertex $V_i \in V$ and $v_i \in v$, we have: $g(V_i) = v_i, g^{-1}(v_i) = V_i$.

We overlay the 2D triangle mesh $\widehat{M}$ generated by the parameterization with the underlying mesh $\widehat{U}$. Users can interactively transform the $\widehat{M}$ to control the position, orientation and scale of placement as shown in Fig. 2. Once the users are satisfied with the settings, the underlying mesh $\widehat{U}$ (gray mesh lines in Fig. 2) will be trimmed by the boundary of the 2D input mesh. The trimmed underlying mesh is denoted as $\widehat{S}$ (black mesh lines in Fig. 2).

The current mapping function only provides a direct mapping between v and V. Using the barycentric coordinates, the mapping $g(\cdot)$ can be extended to arbitrary points on the meshes. For a 3D point P_a locating inside triangle $F = (V_i, V_j, V_k)$, we can have its relative 2D point p_a inside triangle $f = (v_i, v_j, v_k)$ as:

$$g(P_a) = w_i v_i + w_j v_j + w_k v_k, \tag{1}$$

where (w_i, w_j, w_k) is its barycentric coordinate.

Similarly, consider a 2D point p_a lying on the triangle $f = (v_i, v_j, v_k)$, we calculate its corresponding 3D point P_a as:

$$g^{-1}(p_a) = w_i V_i + w_j V_j + w_k V_k. \tag{2}$$

A hash table is constructed to accelerate the mapping. Finally, using inverse mapping, we can quickly map the 2D underlying mesh $\widehat{S}$ back to 3D space and obtain a 3D triangle mesh S for placing elements. Fig. 5 shows multiple pavement designs on a pendant surface after initialization.

5. Pavement optimization

Note that the mapping between 2D and 3D is not isometric. The geometric shape of the input surface in the parameter domain can

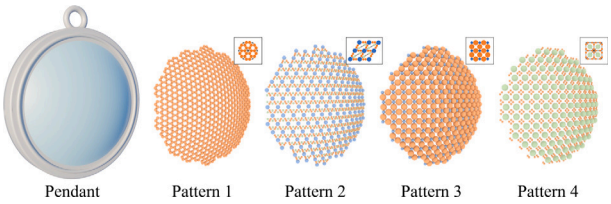


Fig. 5. Four designs of diamond pavement given the same pendant surface following the designated element pattern shown as an inset.

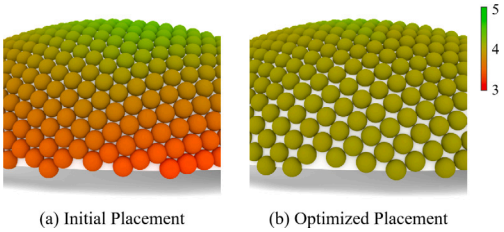


Fig. 6. Pendant examples paved with spheres (V-element only) following a grid pattern before (a) and after (b) optimization. We let the edge length $D = 4$. The sphere diameter is set to four as well. The color of a sphere visualizes the center-to-center distance value to its closest neighbor, from small (red) to large (light green). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

typically be different from the shape of the original surface. The length of the lifted edge P_aP_b can largely deviate from the predefined edge length D. See Fig. 6, it is clear that there exist red elements intersecting with others. Moreover, for design applications, the pavement often exhibits various levels of compactness over curvy and flat surface regions. This section introduces an optimization method to address the intersection issue and achieve a curvature-aware pattern.

Objectives. Given the 2D underlying mesh $\hat{S}$=(p) and the lifted 3D underlying mesh $S = (P)$, we aim to adjust the positions of $\{P_a\}$ to: (i) generate curvature-aware compactness; (ii) preserve the user-designated pattern; and (iii) avoid element collisions.

The straightforward method is to handle such problems in three-dimensional space. However, this is time-consuming and could easily generate vertices away from the target surface. Instead, we choose to perform the optimization in the two-dimensional plane by adjusting the 2D underlying mesh $\hat{S}$.

We treat $\hat{S}$ as a mass–spring system, where each point is considered as a particle and each edge is considered as a spring. Our method considers the geometric curvature of M and the distortion introduced by the parameterization process to create guidance fields for the computation of system forces. We iteratively perturb the locations of 2D points with increments $\Delta p = (\Delta u, \Delta v)$ until the system reaches balance. Finally, the optimized 2D underlying mesh $\hat{S}^*$ is lifted to form the 3D mesh S^* according to the parameters in the element design stage (Section 4.1).

5.1. Guidance fields

To achieve our objectives, we compute a curvature field and propose a target edge length field based on the lifted 3D pavement to guide the adjustment of particle positions in the 2D domain.

Curvature field. For a point P_a lying on the triangle $F = (V_i, V_j, V_k)$ on the mesh surface M with barycentric coordinates (w_i, w_j, w_k). We define the curvature of point P_a as:

$$\kappa(P_a) = w_i\kappa(V_i) + w_j\kappa(V_j) + w_k\kappa(V_k), \tag{3}$$

where $\kappa(V_i)$, $\kappa(V_j)$ and $\kappa(V_k)$ are the discrete mean curvature of vertex V_i, V_j, and V_k respectively, defined by the discrete Laplace–Beltrami operator [42]. The vertex located in the curvy region receives a large curvature value and the vertex in the flat regions receives a small value.

Thus, we obtain a static curvature field $\{\kappa\}$ for M. The $\{\kappa\}$ is used to guide the adjustment of the separation distance of neighboring points following the variation in curvature. For example, the larger the curvature is, the looser the distribution of elements needs to be. We denote the maximum curvature by κ_{max} and the minimum curvature by κ_{min}. We employ κ_{max} and κ_{min} to normalize the curvature into the range [0,1].

Target length field. Next, given the curvature field, we introduce a target length field as dynamic guidance to adjust the separation distance of neighboring points. The target length consists of a pre-defined length and a further apart distance.

Let $N(a)$ denote the set of indices of the vertices adjacent to p_a in the 2D underlying mesh. Firstly, for each edge p_ap_b, $b \in N(a)$, we define D_{ab} as its pre-defined length in the design step. Note that for an edge p_ap_b which follows the initial regular topology, i.e., the black edges in Fig. 3, D_{ab} is equal to D. The length of the other edges p_ap_b that are generated with topology operations can be easily calculated given D. For example, the pre-defined length of the blue edges in Fig. 3(b) is equal to $D\sqrt{2}$. For an edge p_ap_b on the 2D plane, we can find its corresponding edge P_aP_b in 3D space by the inverse mapping in Eq. (2). A uniform compact design indicates that the Euclidean distance between adjacent P_a and P_b should be similar to D_{ab}.

To enable designs with adaptive compactness, we first define $\kappa(P_aP_b)$ to be the larger curvature of the surface at points P_a and P_b, which approximately measures the large variation of the surface around the edge P_aP_b:

$$\kappa(P_aP_b) = max(\kappa(P_a), \kappa(P_b)). \tag{4}$$

Then, we define a further apart distance, $D(P_aP_b)$, to enable adaptive compactness. The $D(P_aP_b)$ is designed to further separate the points P_a and P_b in addition to D_{ab}, varying according to the curvature of edge, $\kappa(P_aP_b)$, and a user-specified additional element-wise margin:

$$D(P_aP_b) = \left(c + m\frac{\kappa(P_aP_b) - \kappa_{min}}{\kappa_{max} - \kappa_{min}}\right) D_{ab}, \tag{5}$$

where c and m are constants controlling the level of separation. In particular, the value of c controls a uniform margin between the elements, which is a positive value. The value of m controls the adaptive margin between elements according to the curvature of the edge, which can be negative. The further apart distance $D(P_aP_b)$ should be a non-negative value. Thus, our tool requires $c + m \geq 0$.

For the 3D edge P_aP_b, we define its target length L_{ab} in 3D space over the input surface to be the sum of the further apart distance $D(P_aP_b)$ and its pre-defined length D_{ab}:

$$L_{ab} = D(P_aP_b) + D_{ab}. \tag{6}$$

5.2. System forces

Instead of adjusting particles in 3D space, we perturb all the particles in the 2D plane. For P_aP_b, we can easily find its corresponding spring p_ap_b in the 2D plane with mapping function g. Next, given its target length L_{ab}, considering the edge distortion brought by the mapping, we set the target length l_{ab} of p_ap_b as: $l_{ab} = L_{ab}|p_ap_b|/|P_aP_b|$.

In our planar mass–spring system, the rest length of each spring is set as its target length. All the particles will move under the influence of three categories of forces (see Fig. 7): (i) restoring force, (ii) local-centric force, and (iii) global geocentric gravity. Fig. 9 is an example showing the effect of each of these forces.

Restoring force. We intend to avoid particle collision via pushing intersected particles away, and encourage particle compactness via

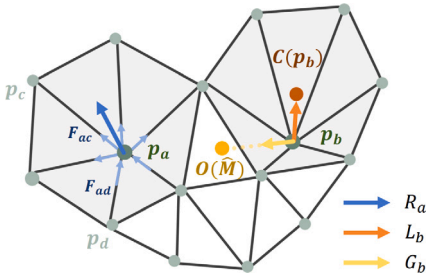


Fig. 7. Each particle p in the mass–spring system $\widehat{M}$ is affected by three 2D forces: (i) The restoring force, R_a, is the sum of connected spring force F of p_a; (ii) Local centric force, L_b, is pushing p_b towards the center of its one-ring neighbors; (iii) Global geocentric gravity, G_b, will push p_b towards the center of the mass–spring system $O(\widehat{M})$.

pulling further-apart particles closer. To this end, we first formulate a restoring force R, following Hooke's Law. Each particle will be influenced by all its connected springs.

For a neighbor p_b of particle p_a, the restoring force of particle p_b on particle p_a is proportional to the stretch or compression of the spring compared to its rest length l_{ab}:

$$F_{ab} = k_c(r_{ab})(|p_a p_b| - l_{ab})\frac{p_a p_b}{|p_a p_b|}, \tag{7}$$

where $k_c(\cdot)$ is a spring constant function.

We hope very few springs are compressed (particle collision) or overly stretched (loose pattern). In molecular dynamics, the *Van der Waals Force* can keep two molecules positioned at a close but separative distance [43]. Inspired by this, we introduce the spring constant function $k_c(\cdot)$ to dynamically assign a value to the spring constant based on the spring state, impose a strong penalty on collisions and looseness. We define the stretching ratio of $p_a p_b$ as $r_{ab} = |p_a p_b|/l_{ab}$ to reflect the spring state. Springs in their resting position have $r = 1$. $r > 1$ related to a stretching spring, and $r < 1$ related to a compressing one. The spring constant function is formulated as:

$$k_c(r) = \frac{\alpha}{r^{12}} - \frac{\beta}{r^6} + \gamma. \tag{8}$$

This spring constant function $k_c(\cdot)$ assigns a large value to k_c if the spring is stretched, and a larger value to k_c when the spring is compressed. Experimentally, we found $\alpha = 10$, $\beta = 5$, $\gamma = 1$ can achieve feasible particle-wise separation.

Then, the restoring force of a particle p_a (the dark-blue arrow in Fig. 7) can be computed as the normalized sum of all the adjoining spring forces F:

$$R_a = \frac{\sum_{b\in N(a)} F_{ab}}{\sum_{b\in N(a)} k_c(r_{ab})}. \tag{9}$$

This restoring force succeeds in alleviating particle collision, and encourages particle compactness. Meanwhile, the local pattern structure can be slightly distorted. Our target is regular patterns, which indicates that a particle p_a should not deviate too much from the center of its one-ring patch.

Local centric force. Next, to preserve the local structure of pattern, we introduce a local virtual force L_a that pulls the particle p_a towards the centroid $C(p_a)$ (brick-red circle in Fig. 7) of the local configuration of p_a :

$$L_a = C(p_a) - p_a, \tag{10}$$

$$C(p_a) = \frac{\sum_{b\in N(a)} p_a/l_{ab}}{\sum_{b\in N(a)} 1/l_{ab}}. \tag{11}$$

Note that if all l_{ab} are identical, $C(p_a)$ is just a simple average of the neighboring points. With the Local centric force, we can retain the local structure of the target pattern. However, the result still lacks global compactness.

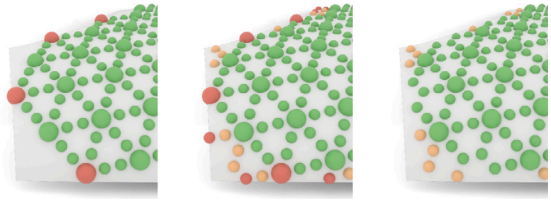


Fig. 8. The results before post-processing (left), after filling boundary regions (middle) and eliminating particles (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Global geocentric gravity. To yield a compact pavement, we further introduce a global virtual force that pulls all the particles towards the center of the bounding box of the 2D input mesh $\widehat{M}$, denoted as $O(\widehat{M})$ (see the orange circle in Fig. 7). The global virtual force G_a of particle p_a is computed as:

$$G_a = \frac{\sum_{b\in N(a)}(|p_a p_b| - l_{ab})}{|N(a)|}\frac{O(\widehat{M}) - p_a}{|O(\widehat{M}) - p_a|}, \tag{12}$$

where $\sum_{b\in N(a)}(|p_a p_b| - l_{ab})$ measures the distortion on the local configuration of p_a using the sum of differences between the current and target edge lengths in its one-ring patch, and $|N(a)|$ denotes the number of particles in $N(a)$. The particle p_a with a local configuration that largely deviates from the target will receive a strong global virtual force.

5.3. Point update

The movement of all the particles is simulated under the system forces. At each time step t, the increment Δp_a for point p_a is calculated as the weighted sum of the three forces:

$$\Delta p_a = \lambda_r R_a + \lambda_l L_a + \lambda_g G_a, \tag{13}$$

where λ_r, λ_l and λ_g are the balancing coefficients.

We iteratively adjust the planar position of particle p_a. The particle position p_a^{t+1} at time $t+1$ is adjusted from p_a^t at time t as follows:

$$p_a^{t+1} = p_a^t + \Delta p_a \Delta t, \tag{14}$$

where Δt is the step size. The optimization stops when the lengths of all increments $|\Delta p|$ are smaller than a predefined threshold, or the number of iterations exceeds a predefined number.

However, the parameterization is not isometric, and an edge $p_a p_b$ can span several triangles of $\widehat{M}$ on the 2D plane. The value of l_{ab} is inaccurate and can be regarded as a noisy signal across iterations. An adaptive filter is a digital filter that has self-adjusting characteristics, capable of noise cancellation [44,45]. We design an adaptive filter w to dynamically adjust the target length l_{ab}:

$$l_{ab}^{t+1} = w^{t+1} l_{ab}^t, \tag{15}$$

$$w^{t+1} = 1 + \mu(1 - \frac{|P_a P_b|}{L_{ab}^t}), \tag{16}$$

where μ is the learning rate in $[0, 1]$. In practice, we let l_{ab}^{t+1} in the range of $[l_{ab}, 1.5 * l_{ab}]$ for numerical stability.

5.4. Post-processing

After optimization, post-processing is introduced to fill in the missing elements at the cross-boundary edges and to ensure that all the elements are located within the boundary of M.

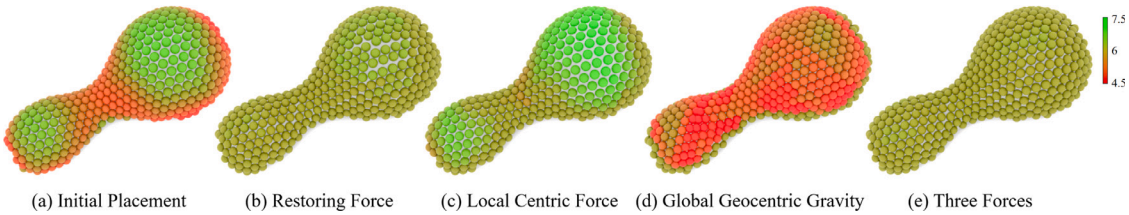


Fig. 9. The pavements following hexagonal tessellation (edge length $D = 6$) over a dome surface, consisting of spheres (diameter equal to D). From left to right shows the initial pavement (a) and optimized pavements under restoring force R (b), local centric force L (c), global geometric gravity G (d) and all three forces (e). The color of a sphere visualizes the center-to-center distance value to its closest neighbor, from small (red) to large (light green). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

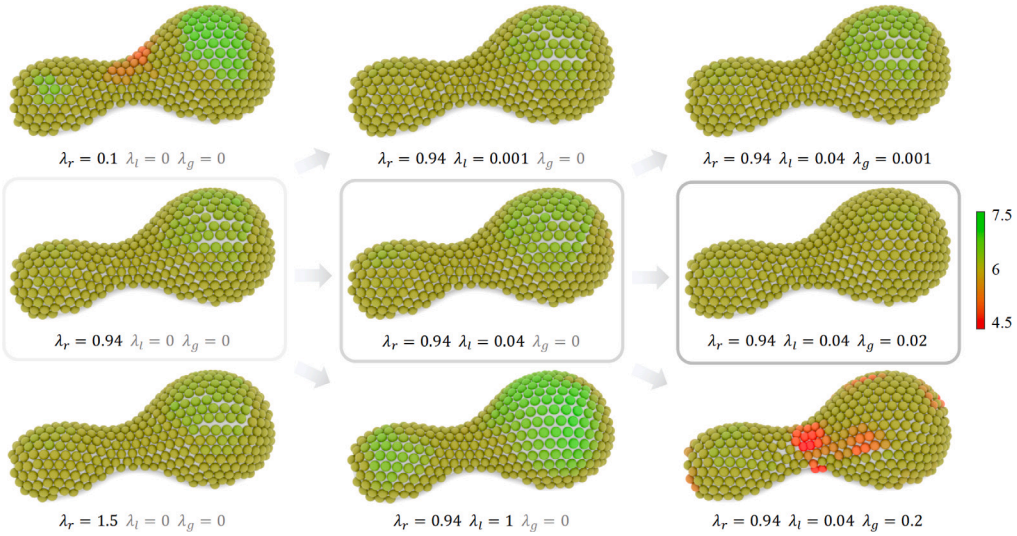


Fig. 10. The pavements following hexagonal tessellation over a dome surface, consisting of spheres (diameter $D = 6$). From left to right shows the pavements optimized with different values on balancing coefficients λ_r, λ_l and λ_g. We explore the values of one coefficient at a time, then fix the value that generates the most feasible result. The color of a sphere visualizes the center-to-center distance value to its closest neighbor, from small (red) to large (light green). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Filling boundary regions. The boundary $\mathcal{B}$ of $\widehat{M}$ may intersect with the edges of $\widehat{U}$. Here, we split those intersected edges in $\widehat{U}$ into two parts using the plane sweep algorithm [46] to find intersection points. Then, we place elements on the edge portion located within $\widehat{M}$, according to the position parameters t.

Eliminating particles. However, the particles can intersect with $\mathcal{B}$. For particles around the border $\mathcal{B}$, we calculate the shortest distance from the center of each element to the boundary. Any vertex element with a distance less than the radius of the element will be removed.

After the post-processing step, as demonstrated in the third column of Fig. 8, the empty regions near the boundary are filled with elements (orange). Meanwhile, all the elements crossing the boundary (red) are eliminated.

6. Results

To validate the proposed algorithm, we conduct a series of experiments on free-form surfaces with varying shapes. These surfaces are created by modelers or come from public sources [47–50]. All the presented experimental results are obtained on a desktop computer equipped with an Intel i7-7700k processor with 3.6 GHz and 32 GB RAM.

6.1. Implementation

Our tool is implemented with C++ using OpenGL, CGAL and Qt. We partially parallelize the optimization process utilizing Intel TBB. To prevent topological issues and preserve the structure of patterns, we introduce extra virtual edges to the underlying mesh during the optimization [37,51].

We first explore the setting of learning rate μ by an empirical study over a set of examples with a small step size. For the results in this paper, we let $\mu = 1$ for the compressed spring and $\mu = 0.5$ for the stretched spring. Next, we explore the value of step size Δt. The selection of the step size is open to the users. According to the empirical results, we recommend letting $\Delta t = 0.01, \ldots, 0.6$, and using a large value for flat surfaces while a small value for curvy ones. The statistics on step size employed in the optimization for examples in Figs. 12 and 13 are provided in Table 1.

6.2. Ablation study

We use the simplest design pattern with only V-elements to examine the effects of key ingredients in the optimization, where the diameter of the element and the edge length of the underlying tessellation are equal to D.

System forces. Firstly, we initialize a pavement consisting of spheres following a hexagonal tessellation over a dome surface. The initial pavement has obvious element collisions (red spheres in Fig. 9(a)). Then, we explore the influence of optimizing the initial pavement with individual force. See Fig. 9(b), the restoring force R can alleviate the element-wise collisions. However, it brings pattern distortion (see the large gaps between spheres). Meanwhile, the local centric force L helps

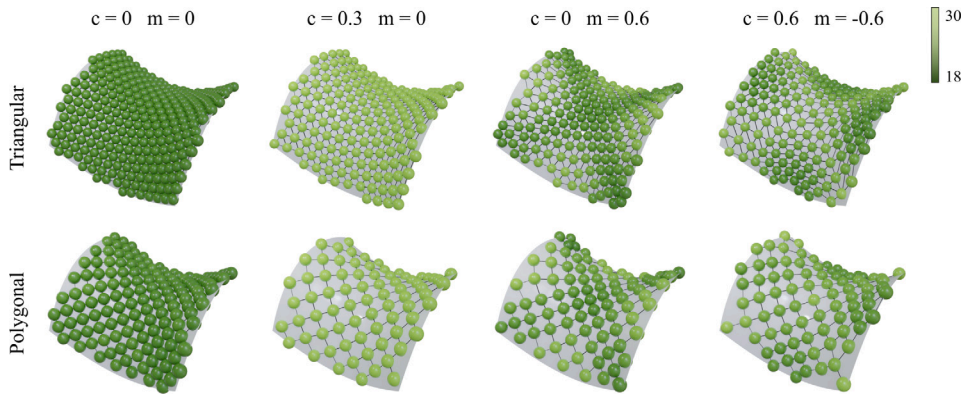


Fig. 11. The optimized pavements with $D = 18$ and spheres (diameter equal to D) over a saddle surface, in triangular (first row) or polygonal (second row) tessellation. The first column shows uniform pavements optimized with $c = 0$ and $m = 0$. The second to fourth columns show adaptive paves optimized with different values of c and m. The color of a sphere visualizes the center-to-center distance value to its closest neighbor, from small (dark green) to large (light green). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

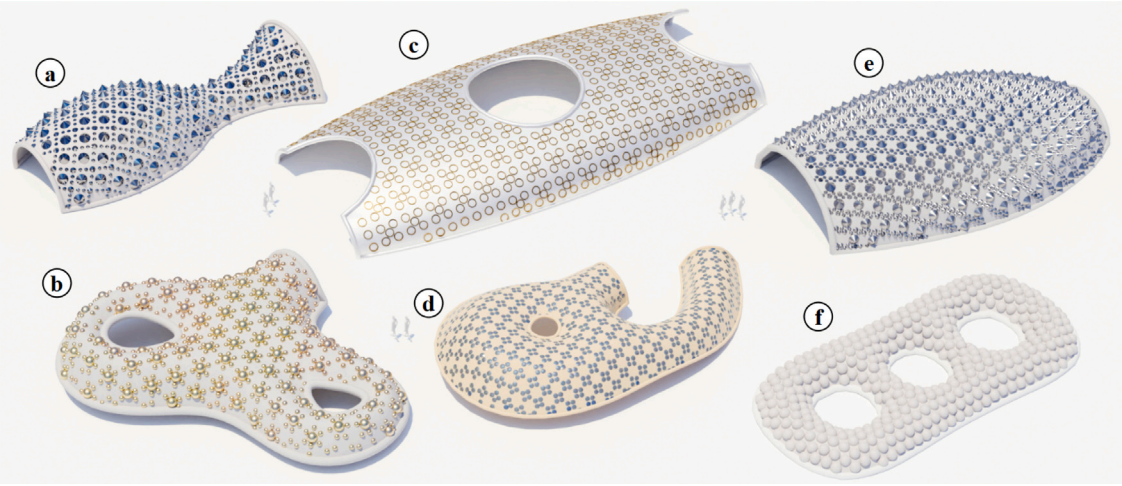


Fig. 12. Six designs of architectural facades with uniform compactness.

to preserve the local pattern, but fails to increase the local element compactness, see the light green spheres in Fig. 9(c). The global geocentric gravity G succeeds in encouraging higher global compactness, while neglecting collisions, see Fig. 9(d). After combining the three forces in Fig. 9(e), we simultaneously improve the regularity of the pattern and relieve the element-wise collision issue.

Next, we explore the optimization results generated with various combinations of coefficients in Fig. 10, with the same step size and number of iterations. We first vary the value of λ_r while setting λ_l and λ_g to zero in the first column of Fig. 10. The results show that the smallest value of λ_r in the first row fails to achieve element-wise separation (see the red and light green spheres). The other two values of λ_r yield similar results. Hence, we fix $\lambda_r = 0.94$, and explore the value of λ_l. The results in the second column show that a small λ_l is ineffective in improving the regularity of local pattern structure (first row), while a large λ_l preserves the structure but yields a locally loose pavement (third row). Therefore, we set $\lambda_l = 0.04$, and vary the value of λ_g in the third column of Fig. 10. The result optimized with a small λ_g is ineffective in achieving better compactness (first row), while a larger value achieves higher density, but does not meet the requirement on element-wise separation (red spheres in the third row). Empirical results indicate that setting the parameters to $\lambda_r = 0.94$, $\lambda_l = 0.04$, and $\lambda_g = 0.02$ produces feasible solutions that achieve uniform compactness. For the balancing coefficients among the system forces in Eq. (13), we use the above values by default for uniform compactness while setting $\lambda_g = 0.001$ for curvature-aware compactness.

Curvature-aware separation. We can manipulate the local compactness of placement according to curvature by varying the parameter values of c and m in Eq. (5). Then, we show pavements optimized with different combinations of c and m over a saddle example. The experimental results are visualized in Fig. 11. Our algorithm is designed to generate a margin of $c * D$ between elements, see the second column in Fig. 11. We will receive a uniform placement of spheres with $m = 0$ and curvature-adaptive pavement with varying local compactness otherwise. With $m > 0$, the algorithm will produce pavement with lower element compactness in high-curvature regions, see the third column in Fig. 11, and vice versa (Fig. 11, fourth column).

6.3. Designs

We present ten architectural facade designs where the elements are combined with the surface in an additive manner. Six architectural facade designs that follow uniform patterns are visualized in Fig. 12. In Fig. 13, we provide four architectural facade designs with curvature-aware patterns. For Wave (h), the elements are placed with higher density in the curvy surface regions, while placed with lower density over relatively flat regions. Meanwhile, Pentagon (g), Conical (i) and Squidward (j) have high-density pavement over flat regions, and low-density pavement in the curvy areas. We list the values of c, m, and

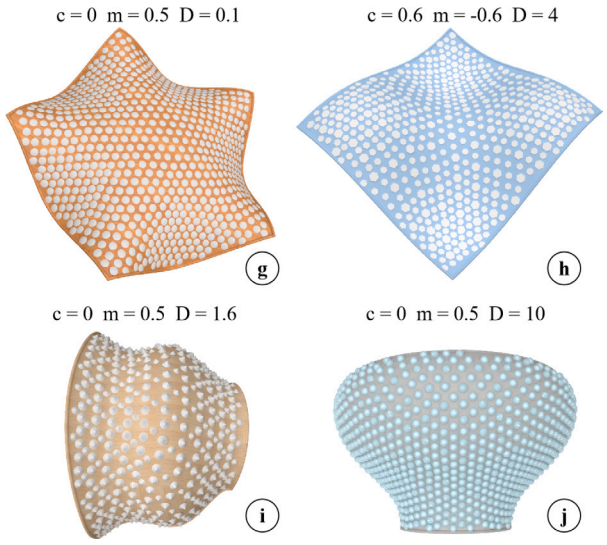


Fig. 13. Four surfaces paved with 3D elements following curvature-aware compactness optimized via different values of c, m and D.

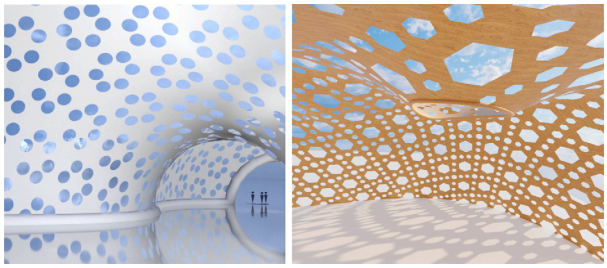


Fig. 14. Two architectural surfaces excavated by spheres (left) and hexagonal prisms (right), respectively.

Table 1
The statistics of results. $|F|$ denotes the number of triangle faces. $|P|$ denotes the number of elements. Time is in minutes.

Surface model		Pavement			
Name	$\|F\|$	$\|P\|$	Time	# Iterations	Δt
Vase (a)	2112	1209	0.36	200	0.5
Aquadom (b)	2576	1906	0.50	200	0.5
Gym (c)	665	1521	0.23	200	0.5
Six (d)	656	1099	0.34	200	0.5
Theatre (e)	432	1834	0.15	200	0.5
Torus (f)	618	606	0.24	700	0.1
Pentagon (g)	1980	802	0.82	2200	0.1
Wave (h)	480	554	0.87	3000	0.03
Conical (i)	1980	599	0.96	2400	0.1
Squidward (j)	2834	642	1.57	3000	0.05

D used in the optimization for the reader's reference. Moreover, we include two architectural designs where the elements are combined with the surface in a subtractive manner, in Fig. 14.

The results exhibit high coherence following the target pattern, target element compactness, and specified separation distance. The statistics of input models and the designs are reported in Table 1. Note that the time cost includes parameterization, initialization and optimization of pavements.

Extension. By default, we do not allow the placement of intersected E-elements in the design stage. Here, we explore an additional option for users to allow a combination of elements in the pattern design step.

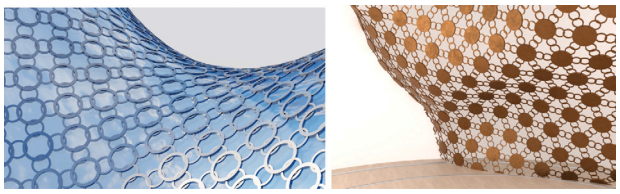


Fig. 15. Two architectural surfaces with a pattern allowing intersection among elements.

Fig. 16. Two jewelry pendants consist of diamonds.

In this design scheme, we can easily create patterns with high diversity. Fig. 15 shows two designs on architectural surfaces allowing the combination of 3D elements. Fig. 16 shows two designs of jewellery [3].

7. Conclusion and future work

We described a design method and a tool called Paver, for paving 3D elements over 3D free-form surfaces. This method is of interest to designers in the fields of jewelry design, architectural facade design and surface models with micro-structure, etc. Our approach has the following features: (i) a framework for placing elements on a 3D free-form surface following user-designed patterns; (ii) an interactive parametric model to let users easily create aesthetic patterns via 2D interactions; (ii) a computationally efficient method to adjust and optimize the initial 3D pavement in 2D, considering local compactness of elements and pattern regularity. Various designs are provided to demonstrate the effectiveness of our approach.

One of our future directions is to accelerate the method using GPU-based parallel computing. Another direction could be to extend this framework to more tessellations (such as pentagonal mesh or irregular tessellation), introduce an option for a user-specific guidance field, and allow placing elements inside the mesh faces as well. In addition, our current work focuses on pattern design and the preservation of pattern structures, without analyzing combinatorial singularities. Detecting and addressing combinatorial singularities represents an interesting problem for future exploration.

CRediT authorship contribution statement

Weidan Xiong: Writing – original draft, Supervision, Software, Conceptualization. **Ziyu Hu:** Writing – review & editing, Visualization, Validation, Software. **Yongli Wu:** Writing – review & editing, Visualization. **Peng Song:** Writing – review & editing. **Jianmin Zheng:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank the reviewers for their constructive comments. This work was supported in parts by The National Natural Science Foundation of China (NSFC) (62302313), Guangdong Basic and Applied Basic Research Foundation (2023B1515120026), Scientific Development Funds from Shenzhen University, Singapore MOE AcRF Tier 2 Grant (MOE-T2EP20222-0008), the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore through Alibaba-NTU Global e-Sustainability CorpLab (ANGEL).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2026.104031.

Data availability

Data will be made available on request.

References

[1] Zhou T, Xiong W, Obata Y, Lange C, Ma Y. Digital product design and engineering analysis techniques. In: Digital manufacturing. Elsevier; 2022, p. 57–96.
[2] Pronk A. The morphology of fluid architecture. In: Flexible forming for fluid architecture. Springer; 2021, p. 141.
[3] Manavis A, Minaoglou P, Aidinli K, Efkolidis N, Kyratsis P. Cad based design for the jewellery industry: A case study. In: IOP conference series: materials science and engineering. vol. 1009, IOP Publishing; 2021, 012036.
[4] Tu P, Wei L-Y, Zwicker M. Clustered vector textures. ACM Trans Graph 2022;41(4).
[5] Bian X, Wei L-Y, Lefebvre S. Tile-based pattern design with topology control. Proc ACM Comput Graph Interact Tech 2018;1(1):1–15.
[6] Zhou Y, Xiao R, Lischinski D, Cohen-Or D, Huang H. Generating non-stationary textures using self-rectification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024, p. 7767–76.
[7] Stephenson K. Introduction to circle packing: the theory of discrete analytic functions. Cambridge University Press; 2005.
[8] Shimada K, Gossard DC. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. In: Proceedings of the third ACM symposium on solid modeling and applications. 1995, p. 409–19.
[9] Gu XD, Luo F, Yau S-T. Recent advances in computational conformal geometry. In: IMA international conference on mathematics of surfaces. Springer; 2009, p. 189–221.
[10] Jin M, Kim J, Luo F, Gu X. Discrete surface ricci flow. IEEE Trans Vis Comput Graphics 2008;14(5):1030–43.
[11] Bs0u10e0. Selfridges building, birmingham. 2012, https://www.flickr.com/photos/bs0u10e0/6783635687/in/photostream/.
[12] Pottmann H, Brell-Cokcan S, Wallner J. Discrete surfaces for architectural design. Curves Surf: Avignon 2006;213(1):e234.
[13] Schiftner A, Höbinger M, Wallner J, Pottmann H. Packing circles and spheres on surfaces. ACM Trans Graph 2009;28(5):1–8.
[14] Chen R, Gotsman C. Parallel blue-noise sampling by constrained farthest point optimization. In: Computer graphics forum. vol. 31, Wiley Online Library; 2012, p. 1775–85.
[15] Yan D-M, Guo J, Jia X, Zhang X, Wonka P. Blue-noise remeshing with farthest point optimization. In: Computer graphics forum. vol. 33, Wiley Online Library; 2014, p. 167–76.
[16] Cline D, Jeschke S, White K, Razdan A, Wonka P. Dart throwing on surfaces. In: Computer graphics forum. vol. 28, Wiley Online Library; 2009, p. 1217–26.
[17] Wang X, Ying X, Liu Y-J, Xin S-Q, Wang W, Gu X, Mueller-Wittig W, He Y. Intrinsic computation of centroidal voronoi tessellation (cvt) on meshes. Comput-Aided Des 2015;58(1):51–61.
[18] Zong C, Wang P, Yan D-M, Chen S, Xin S, Tu C, Hu Q. Parallel post-processing of restricted voronoi diagram on thin sheet models. Comput-Aided Des 2023;159(1):103511.
[19] Hertz A, Hanocka R, Giryes R, Cohen-Or D. Deep geometric texture synthesis. ACM Trans Graph 2020;39(4). 108:1–108:11.
[20] Tu P, Wei L-Y, Zwicker M. Compositional neural textures. In: SIGGRAPH Asia. 2024, p. 1–11.
[21] Ma C, Wei L-Y, Lefebvre S, Tong X. Dynamic element textures. ACM Trans Graph 2013;32(4):1–10.
[22] Jiang C, Tang C, Vaxman A, Wonka P, Pottmann H. Polyhedral patterns. ACM Trans Graph 2015;34(6):1–12.
[23] Peng C-H, Jiang C, Wonka P, Pottmann H. Checkerboard patterns with black rectangles. ACM Trans Graph 2019;38(6):1–13.
[24] Song P, Fu C-W, Goswami P, Zheng J, Mitra NJ, Cohen-Or D. Reciprocal frame structures made easy. ACM Trans Graph 2013;32(4):1–13.
[25] Song P, Fu C-W, Goswami P, Zheng J, Mitra NJ, Cohen-Or D. An interactive computational design tool for large reciprocal frame structures. Nexus Netw J 2014;16(1):109–18.
[26] Zheng J, Xiong W, Zeng J, Davis ED. Design element placement on 3d surfaces. 2024, uS Patent App. 18/279, 130.
[27] Hu J, Wang S, He Y, Luo Z, Lei N, Liu L. A parametric design method for engraving patterns on thin shells. IEEE Trans Vis Comput Graphics 2024;30(07):3719–30.
[28] Gu X, Yau S-T. Global conformal surface parameterization. In: Proceedings of the eurographics/ACM SIGGRAPH symposium on geometry processing. 2003, p. 127–37.
[29] Sheffer A, Lévy B, Mogilnitsky M, Bogomyakov A. Abf++: fast and robust angle based flattening. ACM Trans Graph 2005;24(2):311–30.
[30] Springborn B, Schröder P, Pinkall U. Conformal equivalence of triangle meshes. ACM Trans Graph 2008;27(3):1–11.
[31] Zhao X, Su Z, Gu XD, Kaufman A, Sun J, Gao J, Luo F. Area-preservation mapping using optimal mass transport. IEEE Trans Vis Comput Graphics 2013;19(12):2838–47.
[32] Liu L, Zhang L, Xu Y, Gotsman C, Gortler SJ. A local/global approach to mesh parameterization. In: Computer graphics forum. vol. 27, Wiley Online Library; 2008, p. 1495–504.
[33] Fu X-M, Su J-P, Zhao Z-Y, Fang Q, Ye C, Liu L. Inversion-free geometric mapping construction: A survey. Comput Vis Media 2021;7(3):289–318.
[34] Xiong W, Zhang H, Peng B, Hu Z, Wu Y, Guo J, Huang H. Twintex: Geometry-aware texture generation for abstracted 3d architectural models. ACM Trans Graph 2023;42(6):1–14.
[35] Xiong W, Wu Y, Zeng B, Guo J, Lischinski D, Cohen-Or D, Huang H. Diff-tex: Differentiable texturing for architectural proxy models. ACM Trans Graph 2025;44(6):1–13.
[36] Sorkine O, Alexa M. As-rigid-as-possible surface modeling. In: Symposium on geometry processing. vol. 4, 2007, p. 109–16.
[37] Liu T, Bargteil AW, O'Brien JF, Kavan L. Fast simulation of mass–spring systems. ACM Trans Graph 2013;32(6):1–7.
[38] Li M, Ferguson Z, Schneider T, Langlois T, Zorin D, Panozzo D, Jiang C, Kaufman DM. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. ACM Trans Graph 2020;39(4):1–49.
[39] Wang H, O'Brien J, Ramamoorthi R. Multi-resolution isotropic strain limiting. ACM Trans Graph 2010;29(6):1–10.
[40] Selle A, Lentine M, Fedkiw R. A mass spring model for hair simulation. ACM Trans Graph 2008;27(3):1–11.
[41] Xu S, Liu XP, Zhang H, Hu L. An improved realistic mass–spring model for surgery simulation. In: IEEE international symposium on haptic audio visual environments and games. 2010, p. 1–6.
[42] Reuter M, Biasotti S, Giorgi D, Patanè G, Spagnuolo M. Discrete laplace–beltrami operators for shape analysis and segmentation. Comput Graph 2009;33(3):381–90.
[43] Slater JC, Kirkwood JG. The van der waals forces in gases. Phys Rev 1931;37(6):682.
[44] Zhu Y-S. Applications of adaptive filtering to ecg analysis: noise cancellation and arrhythmia detection. IEEE Trans Biomed Eng 1991;38(8):785–94.
[45] You X, Crebbin G. A robust adaptive estimator for filtering noise in images. IEEE Trans Image Process 2022;4(5):693–9.
[46] De Berg M. Computational geometry: algorithms and applications. Springer Science & Business Media; 2000.
[47] Xiong W, Cheung CM, Sander PV, Joneja A. Rationalizing architectural surfaces based on clustering of joints. IEEE Trans Vis Comput Graphics 2022;28(12):4274–88.
[48] Chen R, Qiu P, Song P, Deng B, Wang Z, He Y. Masonry shell structures with discrete equivalence classes. ACM Trans Graph 2023;42(4):1–12.
[49] Dellinger F, Li X, Wang H. Discrete orthogonal structures. Comput Graph 2023;114(1):126–37.
[50] Chen T, Panetta J, Schnaubelt M, Pauly M. Bistable auxetic surface structures. ACM Trans Graph 2021;40(4):1–9.
[51] Provot X, et al. Deformation constraints in a mass–spring model to describe rigid cloth behaviour. In: Graphics interface. Canadian Information Processing Society; 1995, p. 147.