Checkerboard Patterns with Black Rectangles

CHI-HAN PENG*, KAUST
CAIGUI JIANG*, KAUST
PETER WONKA, KAUST
HELMUT POTTMANN, KAUST

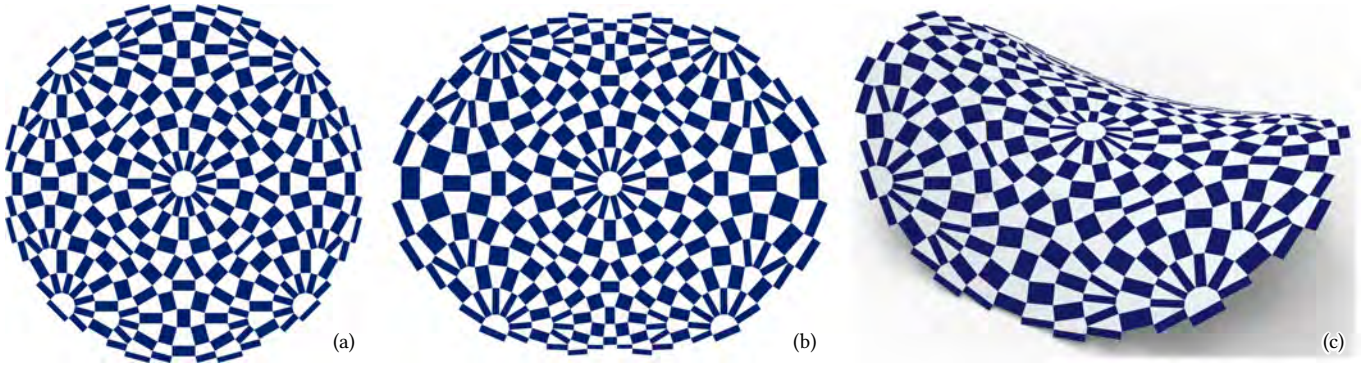


Fig. 1. (a) Our computational design tool can create 2D checkerboard patterns that tile a given boundary in a seamless manner, using exactly the three kinds of black rectangles used in the Tokyo 2020 Olympics Games logo design (see Fig. 2). The pattern has a perfect 4-way rotational symmetry. By allowing each black rectangle to scale uniformly by a different ratio, new designs in 2D (b) and 3D (c) can be created.

Checkerboard patterns with black rectangles can be derived from quad meshes with orthogonal diagonals. First, we present an initial theoretical analysis of these quad meshes. The analysis reveals many possible applications in geometry processing and also motivates the numerical optimization for aesthetic and functional checkerboard pattern design. Second, we describe an optimization algorithm that transforms initial 2D and 3D quad meshes into quad meshes with orthogonal diagonals. Third, we present a 2D checkerboard pattern design framework based on integer programming inspired by the logo design of the 2020 Olympic games. Our results show a variety of 2D and 3D checkerboard patterns that can be derived from 2D or 3D quad meshes with orthogonal diagonals.

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh geometry models**.

Additional Key Words and Phrases: Pattern Design, Discrete Differential Geometry, Architectural Geometry, Quad Meshes

ACM Reference Format:
Chi-Han Peng*, Caigui Jiang*, Peter Wonka, and Helmut Pottmann. 2019. Checkerboard Patterns with Black Rectangles. 1, 1 (August 2019), 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

* Joint first authors.
Authors' addresses: Chi-Han Peng*, pchihan@asu.edu, KAUST; Caigui Jiang*, caigui.j@gmail.com, KAUST; Peter Wonka, pwonka@gmail.com, KAUST; Helmut Pottmann, helmut.pottmann@gmail.com, KAUST.

XXXX-XXXX/2019/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We study the computational design of 2D and 3D checkerboard patterns with black rectangles, i.e., meshes with a black-and-white coloring of the faces for which the shapes of the black faces are constrained to be rectangles. The white faces can either be unconstrained or planar. Our work was originally inspired by the Tokyo 2020 Olympics Games logos design by Japanese artist Asao Tokolo [Committee 2016] (Fig. 2). In these 2D checkerboard patterns, the black faces can be one of three types of rectangles. Rather than generating these patterns directly, they can be derived from 2D rhombic (quad) meshes by subdividing edges and placing one black rectangle inside each rhombus. In particular, most of Tokolo's designs are derived from a tiling of 90°-rhombi (squares), 60°-rhombi, and 30°-rhombi.

Generalizing these patterns to 3D leads to a very interesting class of meshes: quad meshes with orthogonal diagonals. The theoretical analysis part of this paper studies this class of meshes and reveals that these meshes (and their further generalizations) have many applications in geometry processing. In this initial paper we focus on the theory most relevant to checkerboard pattern design to justify the proposed optimization algorithm. The most important practical aspect of these meshes is that they often lead to optimization problems that are numerically stable and have fast convergence.

To tackle the 2D pattern design problem, we propose a computational design tool for rhombic tilings inside a prescribed boundary with three types of tiles. We build on the Integer Programming (IP)-based method for solving square and equilateral triangle tilings in [Peng et al. 2018] and propose several additional components: an accelerated strategy, extensions to control symmetries, and tools to design admissible boundaries from vector graphics.

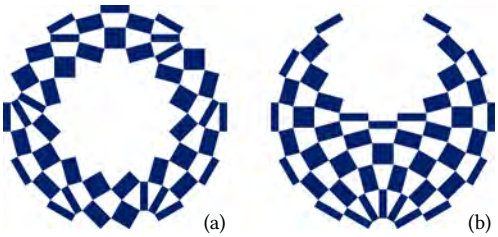


Fig. 2. The Tokyo 2020 Olympic (a) and Paralympic Games (b) logos design by Japanese artist Asao Tokolo [Committee 2016] computed with our framework.

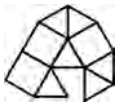
Our paper includes three major contributions:

(1) We study the theoretical background of the geometry of quad meshes whose faces have orthogonal diagonals. These structures have so far been used in discrete complex analysis and are now shown to hold great potential for discrete differential geometry. (See Sec. 3)
(2) A numerical optimization framework to transform 2D or 3D input meshes into quad meshes with orthogonal diagonals. (See Sec. 4)
(3) A 2D rhombic mesh design framework that includes a novel efficient tiling algorithm based on integer programming, a novel boundary design method, and various extensions to control the symmetry of the patterns. (See Sec. 5)

2 BACKGROUND AND RELATED WORK

2.1 Tiling and tessellation

Many tiling problems provide a domain and tile set as input and require that the domain should be covered such that no two tiles overlap. The difficulty of a tiling problem depends on the boundary and the tile set. Grünbaum and Shephard [2016] focus on studying tilings of the infinite Euclidean plane. A popular tiling problem is tiling a given 2D polygon with polyominos (tetris) [Karademir et al. 2016]. Despite the simplicity of polyominos, the corresponding decision problem is already NP-complete and the most natural problem formulation uses integer programming.



While it is easy to imagine a greedy tiling algorithm, it is important to consider that such an algorithm may create narrow spaces that cannot be filled with tiles from the input tile set (see inset for a failed case with equilateral triangles and squares). This happens especially often when growing towards a boundary. Similarly, advancing front/paving-based methods that are commonly used in meshing ([Blacker and Stephenson 1991], [White and Kinney 2007], [Park et al. 2007]) often have problems when multiple growing fronts collide. An often overlooked distinction of tiling problems is whether it is easily possible to enumerate possible tile placements. While this is trivial for polyominos [Karademir et al. 2016], this is difficult for tile sets such as triangle and quads [Peng et al. 2018] as well as the rhombic tile set studied in this paper. Richard Kenyon [1993] proposed an algorithm for tiling a 2D polygon with parallelograms (including rhombi) based on building strips between matched boundary edge pairs. In comparison, our method offers more control over the results, can support polygons with holes, and incorporates triangle tiles as an option to broaden the types of polygons that can be tiled.

2.2 Rhombic Meshes

A more general form of the rhombic-tiling problem is the modeling of rhombic meshes. A closely related concept is a discrete Chebyshev net [Chebyshev 1878], which has been extensively applied in textile models ([Adkins 1956; Aono et al. 1996; Pipkin 1986; Rivlin 1964, 1997; van West et al. 1990]) and computer graphics ([Garg et al. 2014]).

2.3 Architectural geometry

Checkerboard patterns appear in architecture mostly as decorative elements, sometimes on panelizations of flat or cylindrical facades, but to our knowledge not on more general curved skins. However, a significant portion of research in architectural geometry deals with freeform structures from flat panels (see [Pottmann et al. 2015] for an overview). Most closely related to our paper is research on the case where the panels are quads which are as close as possible to rectangles. From a mathematical perspective, these structures are discrete principal curvature parameterizations of surfaces [Bobenko and Suris 2008], which include circular and conical meshes as important cases [Liu et al. 2006; Pottmann et al. 2007]. It is a contribution of the present paper that one can efficiently compute checkerboard patterns whose black faces are precise rectangles, while the white quads are planar and just close to rectangles.

2.4 Fabrication

In fabrication, the use of congruent elements is beneficial for ease of assembly and cost reduction. Examples include Lego block-based modeling ([Mueller et al. 2014], [Luo et al. 2015]) and the use of universal building blocks for cost-efficient 3D printing [Chen et al. 2018]. A major reason is that the building blocks can be manufactured at lower cost due to the fact that rectangular elements pack more easily into square or rectangular sheets of material. Packing problems are also related to tiling as they enforce the non-overlap constraint, but they do not have a constraint to cover the entire domain [Chen et al. 2015].

3 GEOMETRY OF CHECKERBOARD PATTERNS WITH BLACK RECTANGLES

3.1 Control mesh

A checkerboard pattern in 2D or 3D can be constructed from a quad mesh C (called *control mesh*) by edge midpoint subdivision (see inset). This is an immediate consequence of the intercept theorem (or Varignon's theorem [Var 1731]): the edge midpoints of a quad form a parallelogram whose edges are parallel to the diagonals of the quad. Hence, by performing subdivision on C, we obtain a checkerboard pattern where the black faces are parallelograms and the white faces are arbitrary (non-planar) polygons. To restrict the black faces to rectangles, we require that all quads in C have orthogonal diagonals. We therefore use a control mesh to generate seamless checkerboard patterns of arbitrary topology. We can easily show that a control mesh exists for any checkerboard

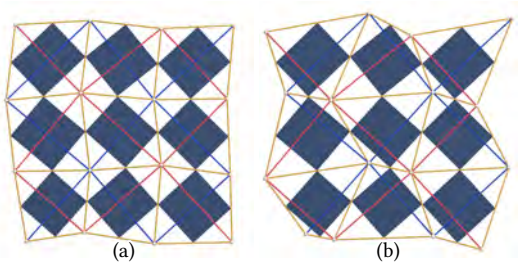


Fig. 3. Two control meshes obtained from the same checkerboard pattern are shown in (a) and (b). The red and blue lines denote the first and second diagonal mesh. While the two control meshes have different shapes, the corresponding diagonal meshes are congruent and related by translations.

pattern with black rectangles if the pattern is simply connected: given a rectangle, we can choose an arbitrary point p_1 as a vertex of a control quad and by consecutive reflection at the four vertices we return to p_1, which again follows from the intercept theorem. This process can be continued over a checkerboard pattern and there will be no closure problems as long as the pattern is simply connected.

To better understand the relation between control mesh and the pattern resulting from edge midpoint subdivision, it is useful to consider the *two diagonal meshes* D_1 and D_2 of C (see Fig. 3). Choosing another starting point $p_1' = p_1 + v$ to construct another control mesh C' for the same pattern, we see immediately that the diagonal mesh which contains the vertex p_1 is translated by the vector v, while the other diagonal mesh is transformed by a translation with vector $-v$. Hence, the control mesh is almost uniquely determined, up to opposite translations of the two diagonal meshes. One can arbitrarily translate the two diagonal meshes and always obtain a checkerboard pattern by computing the midpoints of those vertex pairs in the diagonal meshes which are connected in the control mesh.

More generally, we obtain an entire family of checkerboard patterns via fixed affine combinations $\lambda D_1 + (1-\lambda)D_2$. This family of solutions is useful for design (see Fig. 4), but for studying the patterns it is sufficient to consider midpoint subdivision ($\lambda = 1/2$).

Summarizing, we obtain a checkerboard pattern with black rectangles via edge midpoint subdivision of a quad mesh in which each face has orthogonal diagonals. This works in 2D and 3D.

3.2 Constraints on the black rectangles and maps between surfaces

We get insight into our patterns when we relate them to maps. Two patterns P_1, P_2 with the same combinatorics allow us to set up a one-to-one correspondence between their vertices, edges, and faces. Now the patterns can be seen as discrete surfaces S_1, S_2, with the correspondence setting up a discrete map between S_1 and S_2. These two surfaces may approximate the same underlying smooth reference surface. If one pattern P_1 lies in the plane, we can view P_1 as a parameter domain for P_2. Having only one pattern P_2, we may use part of a regular square grid as parameter domain P_1 in areas of regular combinatorics of P_2.

Let us assume that the patterns P_1, P_2 discretize a smooth map μ. Then, each pair of corresponding rectangles is related by an affine map which is a discretization of the derivative map of μ. This means that *first order properties of μ are seen in the rectangles.*

Consider the mapping from part of a regular square grid P_1 to a combinatorially regular part of a pattern P_2. Due to the rectangles in the pattern, we can view P_2 as a surface parameterization with orthogonal iso-parameter lines. It is not obvious how to express orthogonality of a surface parameterization in the standard discrete setting when working with a single quad mesh. In our approach, it is trivial: We just require orthogonality of diagonals in the control mesh (associated edges in the pair D_1, D_2). The simplicity comes from the mesh pair.

Note that the rectangles are aligned with principal distortion directions of μ. This is great if one wants to involve the field of principal directions in a design or modeling task. It may not be ideal if we want to optimize mappings where the principal distortions are involved, but their directions do not matter. Selecting a combinatorics of P_1, P_2 would mean selection of the combinatorics of the network of principal distortion curves. However, principal distortions are not well defined if the map is *conformal* (angle preserving). Fortunately, conformal maps are the most interesting ones in our context, since we would like to have just a few shapes of rectangles, for example squares only.

Recall that the diagonals in the control mesh are parallel to the edges of rectangles and twice as long. Hence, we can control the map μ between two patterns with the help of the underlying control meshes C_1, C_2, which should have orthogonal diagonals.

a. The map $\mu : P_1 \to P_2$ is discretely conformal if corresponding rectangles have the same aspect ratio, i.e., corresponding diagonals in faces of the control meshes have the same ratio of lengths.
b. The map $\mu : P_1 \to P_2$ is a discrete isometry if corresponding rectangles are congruent and hence corresponding diagonals of the control meshes have the same length.

We have here a form of *discrete conformal equivalence*, which has been studied in mathematics in connection with discrete complex analysis [Bobenko and Skopenkov 2012; Bobenko and Günther 2017; Kenyon 2002; Skopenkov 2013]. However, in 3D and for geometry processing, most work has been based on circle patterns and packing ([Kharevych et al. 2006; Stephenson 2005]) and especially on triangle meshes. For those, a beautiful theory of discrete conformal geometry has been developed, including advanced topics such as relations to hyperbolic geometry and uniformization (see, for example, [Bobenko et al. 2016; Gu et al. 2018a,b; Gu and Yau 2008; Springborn et al. 2008]). There are many applications of this theory in Computer Graphics and beyond. We just mention parameterization, texture mapping, surface deformations, Willmore flow [Crane 2013] and fabrication with auxetic structures [Konaković et al. 2016]. For an introduction into this area and an overview, we refer to [Crane 2019].

The present approach to discrete conformal equivalence is very easy to understand and to implement. One may not even look at the final checkerboard pattern, but just at the pair of diagonal meshes D_1, D_2, which are coupled through orthogonality and obvious relations on edge lengths.

Fig. 4. (a1) The control mesh for the Tokyo 2020 Olympics Games logo design. (a2) to (a4): checkerboard patterns generated from the same control mesh but with different λ for the interpolation of the two diagonal meshes. (a2) is the logo ($\lambda = 0.5$). (b1) and (b2): Two 3D checkerboard patterns generated from the same control mesh but with different λ.

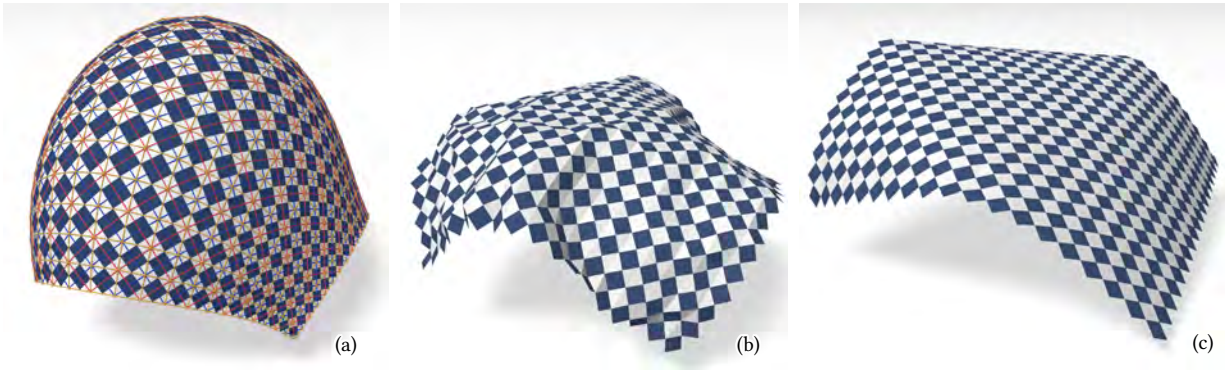


Fig. 5. (a) A pattern from squares, shown with control mesh and the two diagonal meshes, represents a discrete conformal parameterization. Optimizing this pattern for congruent squares, yields a discrete developable surface (b,c), which has a crumpled appearance without (b) and smooth appearance with a fairness term (c).

For theoretical studies and other applications beyond the patterns in this paper, there is a reason to prefer the control mesh with the pair D_1, D_2 over the final pattern. Both meshes D_1, D_2 discretize the same type of mapping as the pattern does. However, the pattern lacks some fairness due to the parallelism of opposite sides in the rectangles. This effect is not seen in the diagonal meshes (see Fig. 6).

If one of two isometric patterns P_1, P_2 is in 2D, the other one must represent a developable surface. In particular, patterns from congruent squares or rectangles are *discrete developable surfaces* (see Fig. 5). Their control meshes are quad meshes with orthogonal diagonals of constant length and constitute a new practically useful discretization of developable surfaces. They provide an alternative to recent work by Rabinovich et al. [2018a; 2018b] and will be studied in detail in future research.

We can use known results on smooth conformal geometry to learn about the behavior of checkerboard patterns under desired design changes and mappings. Important are the following facts. Two simply connected surfaces can always be mapped onto each other by a conformal map. One can even prescribe three corresponding points on their boundaries (Riemann mapping theorem). Also any two closed surfaces of genus zero are conformally equivalent; again one can prescribe three correspondences. Closed surfaces of higher genus or domains which are not simply connected can be conformally mapped onto each other only if they belong to the same conformal class (see e.g. [Gu and Yau 2008]).

Let us summarize typical tasks that can be solved. In all cases, we rely on known facts from the smooth setting:

(1) Mapping a simply connected pattern P_1 onto a pattern P_2 by allowing all rectangles to scale uniformly (with an unspecified scale factor per rectangle) and by specifying a target boundary (see Figures 19, 21, and 22). Here, one may specify three correspondences on the boundary.
(2) Same as above for patterns on closed surfaces of genus zero and three correspondences.

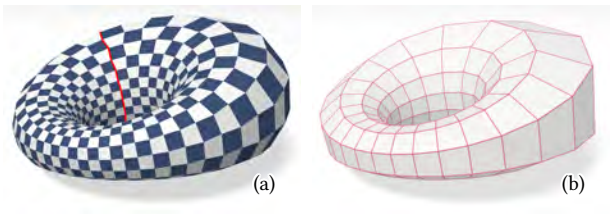


Fig. 6. A checkerboard pattern with black rectangles and white planar quads represents a discrete principal curvature parameterization. While the pattern (a) lacks some fairness (e.g., the red polyline), the diagonal meshes of the control mesh, one shown in (b), do not suffer from this problem and are also discrete principal curvature parameterizations.

(3) For more boundaries or higher genus, one needs to provide more flexibility. One cannot specify a target surface precisely, but optimization will lead towards a possible target.
(4) Mapping a pattern P_1 by keeping the rectangles congruent will only work if the target surface for P_2 is isometric to P_1. Thus, one either needs to know that original surface and target surface are isometric or explore possible target shapes through optimization (Sec. 4). A planar pattern can be mapped isometrically only onto a developable surface.
(5) Taking an arbitrary quad mesh as a control mesh C, one can optimize for a pattern with all rectangles having the same aspect ratio. If C has the topology of a disk or sphere, this can be done without changing the shape of the surface represented by C; the bunny in Fig. 4 has been generated in this way. However, one has to expect a shape change for other topologies (see Fig. 24). This is so since the combinatorics of C already determines the conformal class, which needs not contain the surface represented by C.

3.3 Planar white quads: discrete principal curvature parameterizations

For certain applications, e.g., architecture, it is very useful if not only the black faces, but all faces are planar. We simplify this requirement to only constrain the white quads to be planar and exempt white faces with more than four edges from this requirement. This is achieved if and only if the two diagonal meshes D_1, D_2 in the control mesh are composed of planar quads (PQ meshes). A PQ mesh discretizes a so-called conjugate parameterization of a surface [Bobenko and Suris 2008], but here we also have orthogonality. The only orthogonal conjugate parameterizations are those where the iso-parameter curves are principal curvature lines. This means that *a pattern P from black rectangles and planar white quads is a discrete principal curvature parameterization*, or shortly, a *principal mesh*. Both the pattern P and the two diagonal meshes D_1, D_2 of the control mesh are principal meshes (see Fig. 6).

Principal meshes have received a lot of interest, both within discrete differential geometry [Bobenko and Suris 2008] and applications such as architecture (see e.g. [Liu et al. 2006; Pottmann et al. 2007]). Here we have a new approach to principal meshes. We briefly outline some advantages and show that this is going beyond the currently used discretization.

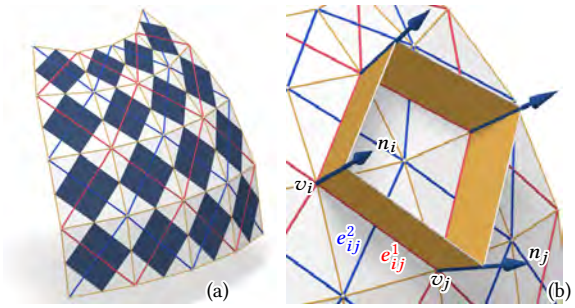


Fig. 7. A checkerboard pattern with black rectangles and planar white faces and both diagonal meshes of the control net, shown in blue and red, are discrete principal curvature parameterizations (a). The diagonal mesh pair allows us to define vertex normals such that connected vertices possess coplanar normals (b). This property facilitates the layout of support structures in architectural applications.

In our approach, the discrete conjugacy, a second order property, constrains the faces of the diagonal meshes D_1, D_2, which are not relevant for the expression of first order properties. This *separation of first and second order properties* in our discrete structure is a big advantage. Note that this is not present in the traditional way of working with a single mesh. Here, one has invented various definitions, such as circular and conical meshes, but these are much less obvious to obtain [Bobenko and Suris 2008; Liu et al. 2006; Pottmann et al. 2007].

We can enforce even stronger conditions in a simple way. An interesting example concerns *checkerboard patterns composed of white planar faces and black rectangles with a fixed aspect ratio* (Fig. 8). As we know from the previous subsection, we have to add the conformality constraint discussed above, i.e., in each quad of the control mesh diagonals have the same ratio of lengths. Such patterns discretize surfaces with a conformal principal curvature parameterization. Known examples of these so-called *isothermic surfaces* include quadrics, rotational surfaces and surfaces with constant mean curvature, in particular minimal surfaces. Isothermic surfaces have been studied very well in differential geometry, both in the smooth and in the discrete setting [Bobenko and Suris 2008]. However, the known discretizations require more sophistication than the present scheme.

Let us add a few basic facts about the new type of principal meshes: Here, we first consider only D_1, D_2 and not the pattern. In a diagonal mesh pair (D_1, D_2) each face in one quad mesh is associated with a vertex of the other. This allows us to define unique *discrete surface normals at vertices* of D_i as normals to the associated face of the other diagonal mesh (see Fig. 7). Consider an edge e^1_{ij} of D_1 (red) connecting vertices v_i, v_j. The normals n_i, n_j at these vertices are orthogonal to the corresponding faces F_i, F_j of D_2 (blue) which share an edge e^2_{ij}. Since the associated edges e^1_{ij} and e^2_{ij} are orthogonal, all three lines n_i, n_j, e^1_{ij} are orthogonal to e^2_{ij} and therefore lie in a common plane. Hence, *along each mesh polyline of D_1 (and D_2) consecutive normals are co-planar and thus form a discrete developable surface.* This is a discrete version of a result from the smooth theory, namely that the normals along a principal

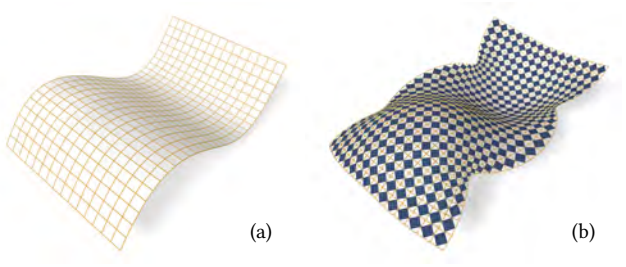


Fig. 8. Starting from an initial control mesh (a), we optimize for a pattern (b) where all black quads are squares and white quads are planar. The generated surface is a discrete isothermic surface.

curvature line form a developable surface. For architecture, these discrete normal developables define torsion-free support structures and are useful for multi-layer constructions; see [Pottmann et al. 2007].

Remark. A special instance of the discussed pairs (D_1, D_2) of principal meshes has been described in connection with the focal geometry of circular and conical meshes [Pottmann and Wallner 2008]. In that case, one has corresponding discrete Gauss images which are polar with respect to the unit sphere S^2 (one is inscribed, the other circumscribed). For our more general pairs of principal meshes (D_1, D_2) one can find parallel meshes (D_1^*, D_2^*) which approximate the unit sphere S^2 and are transformed into each other by polarity with respect to S^2. These discrete Gauss images can be used to define discrete curvatures and a discrete shape operator. As a derivative of the Gauss map, the discrete shape operator is seen explicitly in corresponding rectangles of patterns associated with (D_1, D_2) and (D_1^*, D_2^*). We leave more on that for future research, as it goes far beyond the scope of the present paper.

4 CHECKERBOARD PATTERN OPTIMIZATION

In this section, we describe how to optimize an input quad mesh to generate a quad mesh with orthogonal diagonals in 2D or 3D. As input we consider an initial control mesh C, given as quad mesh (V, E, F). The required checkerboard pattern can be derived directly from the optimized control mesh C by subdivision. As optional input we consider an aspect ratio for the quad diagonals that can be specified for each quad separately. This aspect ratio directly translates to the aspect ratio of the black rectangles in the derived checkerboard pattern. Further, it is possible to optionally require the white faces of the derived checkerboard pattern to be planar as well.

Variables . We denote the vertex coordinates of the control mesh C as $\mathbf{v}_i, i \in V$ and the vertex normal vectors as $\mathbf{n}_i, i \in V$. The vectors $\mathbf{n}_i$ are also the face normals of the white quads.

Diagonal Orthogonality. For all the quads of C, we require their two diagonals to be orthogonal. This is the fundamental constraint for checkerboard patterns with black rectangles. We write the constraint as an energy term

$$E_{orth} = \sum_{k \in F} \left((\mathbf{v}_{k1} - \mathbf{v}_{k3}) \cdot (\mathbf{v}_{k2} - \mathbf{v}_{k4})\right)^2, \tag{1}$$

where $k1$, $k2$, $k3$ and $k4$ are the vertex ids of the face k.

Diagonal Ratio Constraint. The ratio of the two diagonal lengths of a quad in the control mesh can be constrained by the energy term

$$E_{ratio} = \sum_{k \in F} ((\mathbf{v}_{k1} - \mathbf{v}_{k3}) \cdot (\mathbf{v}_{k1} - \mathbf{v}_{k3}) - r_k^2 (\mathbf{v}_{k2} - \mathbf{v}_{k4}) \cdot (\mathbf{v}_{k2} - \mathbf{v}_{k4}))^2, \tag{2}$$

where r_k is the ratio of two diagonal lengths $|\mathbf{v}_{k1}\mathbf{v}_{k3}|$ and $|\mathbf{v}_{k2}\mathbf{v}_{k4}|$.

Planarity. The black rectangles derived from the control mesh are automatically planar. The following constraint encodes the optional planarity for white quads. The planarity of white faces can be expressed by requiring the neighbouring vertices of each vertex $\mathbf{v}_i$ to form a planar quad. We use the same planarity formulation of [Tang et al. 2014] and [Jiang et al. 2015],

$$E_{plan} = \sum_{i \in V} \sum_{(i,j) \in E} ((\mathbf{v}_i - \mathbf{v}_j) \cdot \mathbf{n}_i)^2 + \sum_i (\mathbf{n}_i \cdot \mathbf{n}_i - 1)^2, \tag{3}$$

where $\mathbf{n}_i$ are the face normals of the white quads.

Objective. The objective function is written as combination of the three energy terms:

$$E = E_{orth} + \lambda_1 E_{ratio} + \lambda_2 E_{plan}. \tag{4}$$

To make the parameter settings between different models comparable, we scale the input meshes such that their average edge length is 1. The energy terms E_{ratio} and E_{plan} are optional, so the weights λ_1 and λ_2 are set to 1 or 0 depending on the applications. We implemented the optimization using the Levenberg-Marquardt algorithm. We also experimented with simpler optimization algorithms such as standard gradient descent but observed that the results tend to get less smooth. Only for the pattern in Fig. 5 (c) we used as additional regularizer a fairness energy applied to the diagonal meshes, namely the standard sum of squared 2nd difference vectors of the mesh polylines.

Algorithm extension. To achieve smoother control quad meshes, the algorithm can be extended by replacing the gradient g in the LM algorithm by a modified descent direction d which satisfies the equation $Ld = g$, where L is the graph Laplacian matrix. This scheme essentially takes a gradient step with respect to a Sobolev-like metric, rather than the ordinary L^2 metric. The motivation is that the norm used to define the gradient should account not only for the change in position, but also the change in normals. A similar strategy can be found in [Martin et al. 2013; Schumacher 2017]. The modified algorithm usually takes significantly more iterations to converge and each iteration is slower because of additional calculation for the modified descent direction. Examples comparing the algorithm before and after the modification are shown in Fig. 9 and Fig. 10. Fig. 10 demonstrates a case where the original algorithm produces quite noisy results while the extended algorithm produces more smooth ones when the input mesh is far from having orthogonal diagonals.

In the next section we will show how a variety of different checkerboard patterns can be generated with different parameter settings.

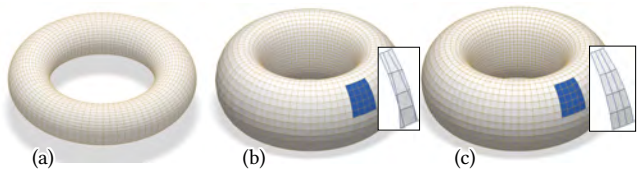


Fig. 9. Diagonal orthogonality optimization of a torus model. (a) Input mesh with elongated quads. (b) and (c) are the optimized meshes by LM and the extended algorithm. We take a random small part of the meshes and compute their Gauss images using the corresponding face normal of black quads. (b) takes 0.15 seconds and (c) takes 72.03 seconds to compute.

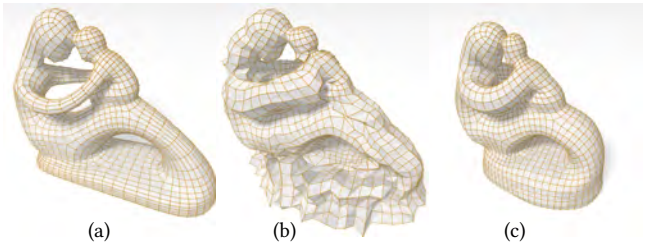


Fig. 10. Diagonal orthogonality optimization. (a) Input mesh. (b) and (c) compare the optimized meshes by LM and the extended algorithm. (b) takes 1.25 seconds and (c) takes 60.35 seconds to compute.

5 2D CHECKERBOARD PATTERN DESIGN

5.1 Rhombic tiling using integer programming

Here, we describe our basic IP-based method for tiling a simply connected 2D domain with rhombi of angles 90°, 60°, and 30°. We denote such tilings as *rhombic tilings*. Our proposed solution builds on the recent tiling algorithm for triangles and quads [Peng et al. 2018], and we discuss the differences at the end of this section. We review key properties first.

There are twelve possible directions for the half-edges in a rhombic tiling. After fixing one half-edge's direction, we can encode each half-edge's direction as an integer $\in [0, 11]$. Assuming an edge length of 2, the coordinate vectors of vertices must have the form:

$$v = (A + B\sqrt{3}, C + D\sqrt{3}), \tag{5}$$

and can be encoded as *discrete 4D coordinate*, (A, B, C, D). Every 4D coordinate can be projected to a 2D coordinate by the "Projection 1" proposed in [Peng et al. 2018] as follows (we slightly modify the projection by dividing coordinates by 2):

$$(A, B, C, D) \mapsto (x, y) = (A + 2B, C + 2D). \tag{6}$$

We denote this 2D plane P^2. Note that multiple 4D coordinates can be mapped to the same 2D coordinate in P^2.

We call a boundary *admissible* for a rhombic-tiling if it is a simple polygon that has edges of length 2 and turning angles as multiples of $30° \in [0, 330]$. Admissible is a necessary but not sufficient condition for a boundary to be *feasible* for a rhombic-tiling.

As input, we assume that an admissible boundary is given as a counterclockwise loop of half edges. As initialization, we embed the boundary in the Euclidean plane and P^2 such that the first boundary half-edge is in the 0-th direction and starts at $(0, 0)$ (see Fig. 11 (a) for an example).

There are two main steps of our method as follows:

(1) *Tile enumeration.* Enumerate a superset S of potential tile placements within the boundary.
(2) *Tiling computation.* Solve an IP problem to select a subset of potential tile placements in S that covers the bounded domain without gaps nor overlaps.

In the following, we describe these steps including an extension to other types of triangular tiles. In subsequent subsections we present mechanisms for local and global control as well as tools for admissible boundary design.

5.1.1 Tile enumeration. We present the following analysis to motivate why the tile enumeration and the subsequent tiling computation can be done in P^2 alone. First, every rhombic tiling embedded in P^2 uniquely maps to a rhombic tiling embedded in E^2. This can be shown by establishing that all paths from the origin to a vertex are consistent. The Euclidean coordinate of a vertex is then obtained by summing up the 4D vectors of the half-edges along the path. Further, all vertices that could be part of a tiling inside the domain in E^2 map to a vertex inside the domain in P^2.

In P^2, a unique tile placement is defined by its tile type (one of three types of rhombi), the 2D coordinate of a fixed vertex in P^2, and its orientation. As shown in Fig 12, there are three, six, and six possible orientations for the 90°-rhombi, 60°-rhombi, and 30°-rhombi, respectively. We can exhaustively enumerate all possible tile placements by iterating over every point inside the admissible boundary in P^2.

5.1.2 Tiling computation. In order for a tiling to be valid, we can simply check that a small circle around each interior vertex is covered by tiles without overlap. For boundary vertices, the subset of the circle inside the domain has to be covered. This test is sufficient, because a hole in the tiling has to touch at least one interior or boundary vertex. Due to the nature of our tile set, we can simply slice each circle into 12 slices and check the coverage of these 12 discrete slices.

We denote the selection of the i-th potential tile placement by a Boolean variable T_i. For every tile placement, we enumerate slices that are covered by the tile. Conversely, for a slice $s_j, 0 \le j < N_s, N_s$ is the number of slices in the bounded domain, we enumerate tiles that cover s_j and denote these tiles as $T_{j,k}, 0 \le k < K_j, K_j$ is the number of tiles that cover s_j. We solve the following IP feasibility problem:

$$\begin{aligned} &\text{find} && T_i,\ 0 \le i < |S|, \\ &\text{s.t.} && \sum_k T_{j,k} = 1, \forall\ s_j. \end{aligned} \tag{7}$$

Solving Eq. 7 gives us a selection of the potential tile placements such that every slice in the bounded domain is covered exactly by one selected tile. An example is shown in Fig. 13 (a). An exhaustive enumeration of all possible solutions can be done by solving the IP multiple times, each time banning all previously retrieved solutions.

Note that Eq. 7 can be infeasible for certain admissible boundaries, which can be helped by adding triangles to the tile set.

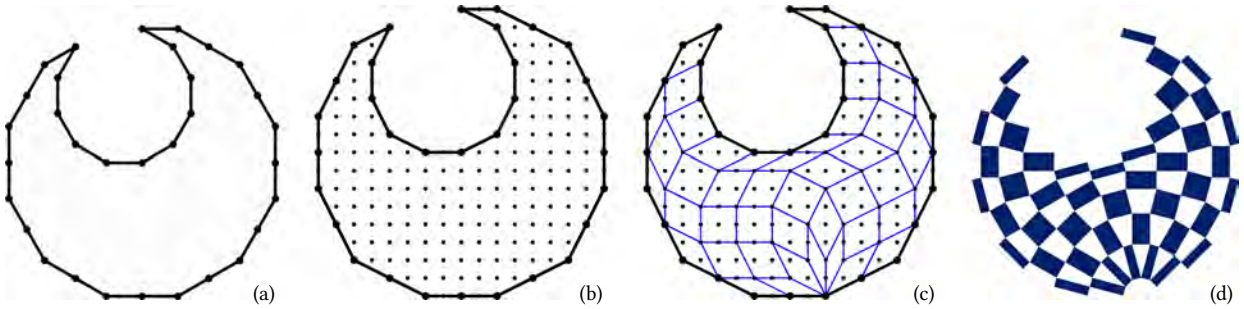


Fig. 11. Pipeline of 2D checkerboard pattern generation. (a) An input admissible boundary in E^2. (b) The admissible boundary is mapped to the 2D projection space P^2. (c) A tiling in P^2 is computed such that all slices around vertices are covered exactly by one tile. (d) The tiling in P^2 is mapped back to E^2 and then a corresponding checkerboard pattern is created by subdivision.

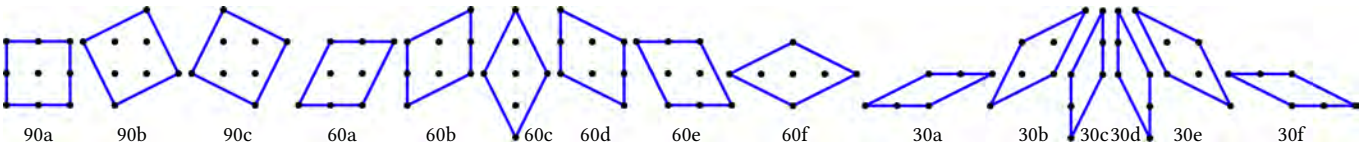


Fig. 12. The three, six, and six possible orientations for the 90°-, 60°-, and 30°-rhombic tiles.

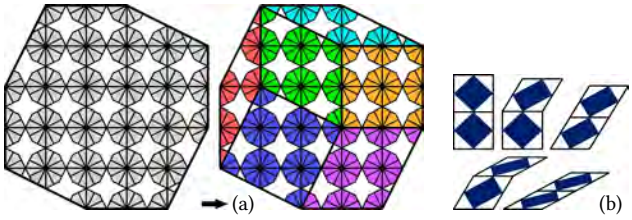


Fig. 13. (a) A covering of slices by tiles. Slices covered by the same tile are drawn with the same color. (b) Adjacent tile combinations that lead to a smooth-looking pattern.

5.1.3 Triangle tiles extension. In this paper, we treat triangle tiles as singularities and their occurrences are minimized. We can add triangle tiles (equilateral triangles of side length 2 in E^2) and revise the IP problem:

$$\begin{aligned} &\text{find} && T_i,\ 0 \le i < |S^{+tri}|, \\ &\text{min.} && \sum_x T_x^{tri},\ 0 \le x < N_{tri} \\ &\text{s.t.} && \sum_k T_{j,k} = 1, \forall\ s_j, \end{aligned} \tag{8}$$

where S^{+tri} denotes the superset of potential tile placements including the three kinds of rhombi plus triangles. T^{tri} denote triangle tile placements and N_{tri} denotes the number of them. Examples of tiling with triangle tiles are shown in Fig. 15.

5.2 User control for the pattern style

We propose five schemes for users to retrieve tiling solutions with certain desired qualities.

a) Symmetry. We support two types of symmetry constraints:

(1) Reflective or 180°-rotational symmetry with respect to the $x-$, $y-$, and 45°-axis in the Euclidean plane.

(2) Arbitrary reflective or rotational symmetry other than the ones specified above.

For (1), this is a special case where we can generate a scalar field at integer coordinates in P^2 with the same symmetry. We identify *paired* tile placements that have matching scalar values at their vertices and then add the following constraints to Eq. 8 to ensure that paired tile placements must be selected at the same time:

$$T_{x,0} = T_{x,1}, \forall \text{ paired tile placements } T_{x,0} \text{ and } T_{x,1}. \tag{9}$$

For (2), the challenge is that a scalar field matching the symmetry cannot be established in P^2. Still, we can identify paired boundary edges by the symmetry and therefore paired tile placements among the tiles adjacent to the boundary. Inspired by this, we solve the tiling problem in a way that is similar to an advancing-front tiling method (see Fig. 16). At each iteration, paired tile placements are identified among the tiles that are adjacent to the advancing front.

Examples of symmetric tiling results are shown in Fig. 14.

b) Global style. We can optimize for two distinct visual styles of checkerboard patterns generated by rhombic-tiling: *fractured* and *smooth* (see Fig. 14 (d) and (e)). To do so, we identify several combinations of adjacent tile placements that lead to a smooth-looking pattern: 1) two 90°-rhombi, 2) one 90°-rhombus and one 60°-rhombus, and 3) two 60°- or 30°-rhombi put together and one obtuse corner is adjacent to one acute corner (see Fig. 13 (b)). For each combination of tiles T_p and T_q, we create a Boolean variable, $C_{p,q}$, and constrain it to be true if and only if both T_p and T_q are selected. This is done by adding the following constraints for each combination to Eq. 8:

$$0 \le T_p + T_q - 2 * C_{p,q} \le 1. \tag{10}$$

To optimize for the fractured or the smooth styles, we add the positive or negative sum of all combination Boolean variables to the objective function in Eq. 8.

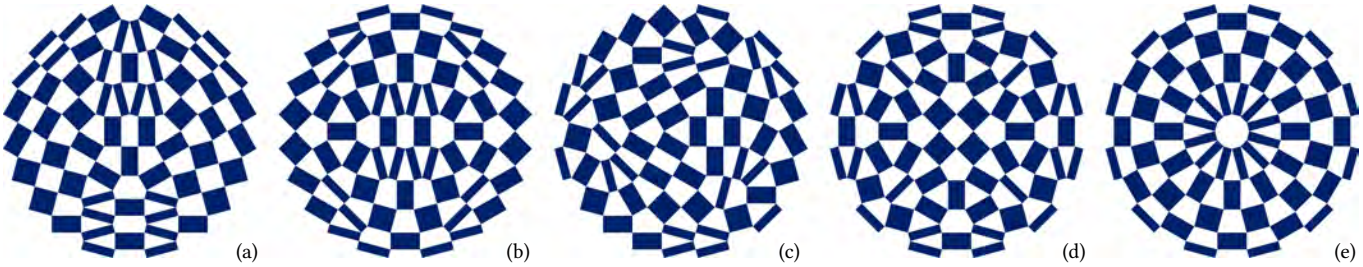


Fig. 14. Checkerboard patterns with a (a) left-right reflective symmetry, (b) left-right and up-down reflective symmetries, (c) three-way rotational symmetry, and (d and e) four-way rotational symmetry. (d) and (e) are further optimized with the fractured and smooth global styles, respectively.

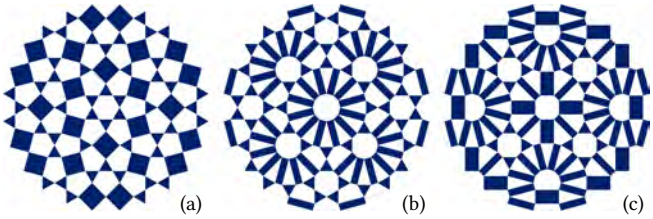


Fig. 15. Tilings generated with (a) 90°-rhombi and triangle tiles, (b) 30°-rhombi and triangle tiles, and (c) 60°- and 30°-rhombi and triangle tiles.

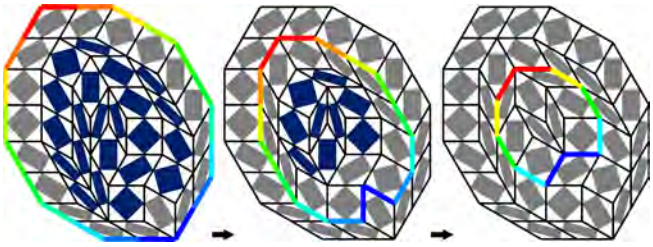


Fig. 16. Generating a symmetric tiling with a reflective symmetry from the top-left to the bottom-right corners. We solve tiling in three iterations. In each iteration, paired edges of the current advancing front are drawn with the same color. Tiles solved at each iteration are drawn in grey.

c) Local control. It is straightforward to enforce or forbid certain tile placements to appear as hard constraints added to Eq. 8. Alternatively, they can be added as a weighted sum to the objective function of Eq. 8 as a soft constraint. One such example are the "eyes" of the Tokyo 2020 Mascots design shown in Fig. 18.

d) Holes. To create holes in the tiling, we first identify points inside the specified hole regions. Then we remove potential tile placements and coverage constraints for these points.

e) Admissible boundary design. Many simple admissible boundaries such as ellipses (including circle), rectangles, and triangles, can be drawn by hand as a sequence of half-edge directions $\in [0, 11]$. One can also make admissible boundaries using interactive editing operations. See Fig. 26 and 27 in the additional materials for examples.

In summary, the rhombic-tiling IP problem takes the form:

$$\begin{aligned} \text{find} \quad & T_i,\ 0 \le i < |S^{+tri}|, \\ \text{min.} \quad & W_t \sum_x T_x^{tri} + W_s \sum_{p,q} C_{p,q} + W_c \sum_c T_c\ , \\ \text{s.t.} \quad & \sum_k T_{j,k} = 1, \forall\ s_j, \\ & Eq.\ 10, \forall\ C_{p,q}, \\ & T_a = 1, \forall \text{ enforced tiles } T_a, \\ & T_b = 0, \forall \text{ forbidden tiles } T_b. \end{aligned} \tag{11}$$

where W_t, W_s, and W_c are the weights for the triangle-minimization, global-style, and soft local-control terms. T_c denotes the tiles for soft local-control constraints. Note that $T_x^{tri} = \emptyset$ and $S^{+tri} = S$ when triangle tiles are excluded.

5.2.1 Comparison to [Peng et al. 2018]. In short, the IP formulation in [Peng et al. 2018] selects a subset of all possible potential edge placements within the admissible boundary that satisfies the requirement that at every potential vertex location, adjacent edges must join in exactly one of the allowed "configurations", which are defined by the sequences of corner angles in counterclockwise order without rotational symmetry. There are 29 such configurations for interior vertices. This approach is not feasible for our rhombic-tiling problem for two reasons. First, the numbers of such configurations with our three kinds of rhombic tiles become prohibitively large. Second, simply regulating ways edges can join at each vertex is no longer enough as two adjacent triangle tiles that occupy the space of a single 60°-rhombus cannot be prevented.

By experiments, we also find that our method is significantly faster than Peng et al.'s for solving the same regular triangle-quad tiling problem. Solved on a machine with nearly identical specs, we solved the tiling problem in Fig. 7 (c) of [Peng et al. 2018] (using only 90°-rhombic and triangle tiles) in 0.84 sec as compared to 21.32 sec reported in their paper, which is about 25 times faster.

6 RESULTS AND APPLICATIONS

6.1 Implementation and running times

The rhombic tiling computation is solved by Gurobi on a machine with Intel Xeon 16 Core 2.30GHz CPU and 128GB RAM. We list the statistics and timing numbers in Table 1.

The running times for the geometric optimization in Sec. 4 of our examples are all under one second using the basic geometric

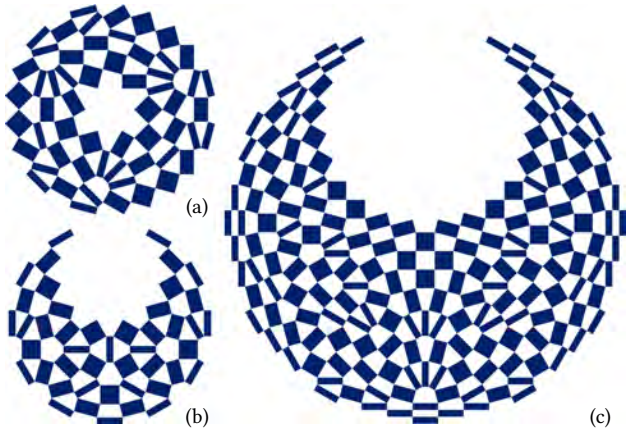


Fig. 17. Variations of the Tokyo 2020 logo design. (a) A thicker version of the ring-shaped Olympics logo with the same 3-way rotational symmetry. (b) A remake of the Paralympic Games logo with a "fractured" global style. (c) A bigger design with a left-right reflective symmetry.

Table 1. Rhombic tiling computation statistics. #PT denotes the number of potential tile placements. #Tiles denotes the number of tiles in a solution.

Model	#PT	#Tiles	Time	Model	#PT	#Tiles	Time
Fig. 1*	10907	240	1.46s	Fig. 1	10907	240	23.75s
Fig. 14 (a)	2347	60	0.55s	Fig. 15 (c)	2566	76	0.24s
Fig. 17 (a)	1940	57	2.87s	Fig. 17 (b)	1413	45	12.2s
Fig. 17 (c)	7357	180	49.61s	Fig. 18 (a)	8128	211	5.896s
Fig. 18 (b)	3999	96	0.73s	Fig. 18 (c)	4832	124	1.27s
Fig. 20-1‡	2901	71	1.22s	Fig. 20-2	3866	75	3.05s
Fig. 20-4	3755	68	0.97s	Fig. 20-5	3606	89	1.19s
Fig. 21 (a)	11743	206	4.27s	Fig. 21 (b)	55067	1068	53.47s

* Without "smooth" global style. ‡ In left to right order.

optimization with an Intel Xeon E5-2687W 3.0GHz machine without parallel processing or other acceleration techniques. The objective functions converge to less than 1e-20 within 10 iterations.

6.2 2D Patterns with three types of rectangles

We begin by showcasing a gallery of 2D checkerboard patterns to complement the Tokyo 2020 logo design. Recall that patterns generated by our rhombic-tiling method (Sec. 5) consist of exactly the same kinds of rectangles as in the Tokyo 2020 logos. In Fig. 17, we propose several variations of the two logos. In Fig. 18 (a), we show a checkerboard version of the five-ringed symbol of the Olympics Games. In Fig. 18 (b) and (c), we show two portrait-like patterns inspired by the Tokyo 2020 Mascots. In Fig. 20, we show more 2D pattern designs with boundaries generated from reference 2D shapes. To reduce clutter, we show the corresponding rhombic-tiling meshes in the additional materials.

A rhombic tiling generated by our 2D method gives a 2D control mesh (Sec. 3.1) with which a family of checkerboard patterns with the same number of types of rectangles can be computed as $\lambda D_1 + (1-\lambda)D_2$, where D_1 and D_2 are the two diagonal meshes. When λ equals 0.5, the rectangles equal to the ones used in the Tokyo 2020 design. In Fig. 4 (a), we show other variations.

6.3 2D Patterns with rectangles of fixed aspect ratios

More general 2D checkerboard patterns can be generated by the numerical optimization scheme described in Sec. 4. We take a 2D rhombic tiling as input and prescribe three types of aspect ratios and changing the boundary. Examples are shown in Fig. 19.

6.4 2D patterns mapped to surfaces

The numerical optimization scheme also enables us to map 2D checkerboard patterns to arbitrary 3D surfaces while requiring that each type of rectangles have the same aspect ratio (with possibly different sizes). See Fig 21 and Fig. 22 for designs enabled by this scheme.

6.5 Checkerboard patterns from quad meshes

Another way to create 3D checkerboard patterns is to directly take a quad mesh of a reference surface as the control mesh and then optimize for the orthogonality of quad diagonals. We tried two schemes to assign the types of black rectangles - 1) all of them are squares and 2) random assignment. For both schemes, we find that the numerical optimization converges very quickly for almost any input quad mesh. See Fig. 4 (b), Fig. 22, Fig. 23 (c), and Fig. 24 for examples.

6.6 Checkerboard patterns with planar tiles

In Fig. 23 and Fig. 25 , we show 3D checkerboard patterns with the additional constraint that all white quad faces are planar. This is a major advantage in architectural geometry as planar quads are strongly preferred.

6.7 Discussion and limitation

We believe that our integer programming approach scales to smaller and medium scale designs as demonstrated in the paper. For bigger scale designs, having symmetry constraints helps to reduce computation cost (our biggest design, Fig.21 (b), has 1068 tiles). However, integer programming does not scale to very large designs, e.g. we failed to generate designs of around 1000 tiles without symmetry. This is an inherent limitation of integer programming. The numerical optimization works efficiently and converges quickly for many input quad meshes. That indicates that quad meshes with orthogonal diagonals are a great surface discretization that is very flexible. However, for difficult inputs such as quad meshes that contain quads with two long diagonals that are far from orthogonal, the slower extended algorithm is required. As expected, additionally requiring the white faces to be planar makes the optimization significantly more constrained. While an input control quad mesh can be deformed by specifying different aspect ratios for different quads, we currently do not have an algorithm that jointly computes an interesting pattern directly on the surface as initialization. We leave this to future work. As already indicated in Sec. 3, the discretization of surfaces based on the two diagonal meshes of a control quad mesh offers numerous possibilities for future research in discrete differential geometry and in geometry processing.

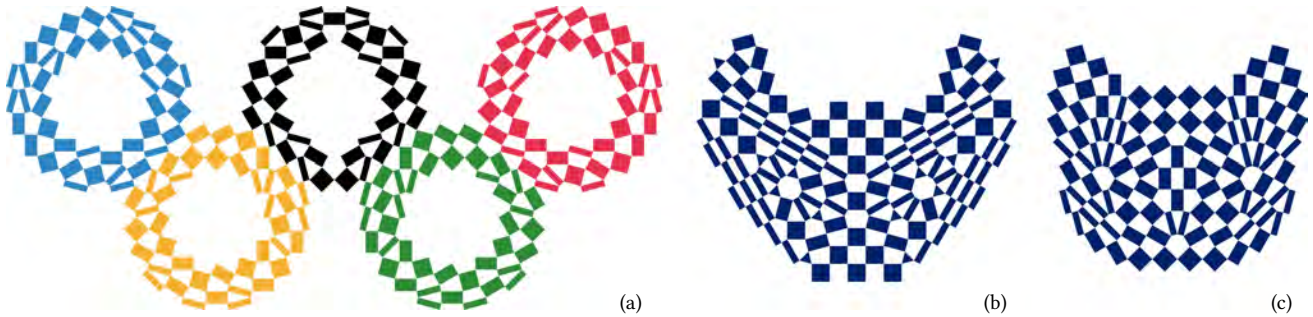


Fig. 18. (a) A checkerboard version of the five-ringed symbol of the Olympics Games. Note that the five rings join seamlessly. (b) and (c) Two portrait-like patterns inspired by the Tokyo 2020 Mascots. The "eyes" are realized by prioritizing tiles near the eye locations to appear.

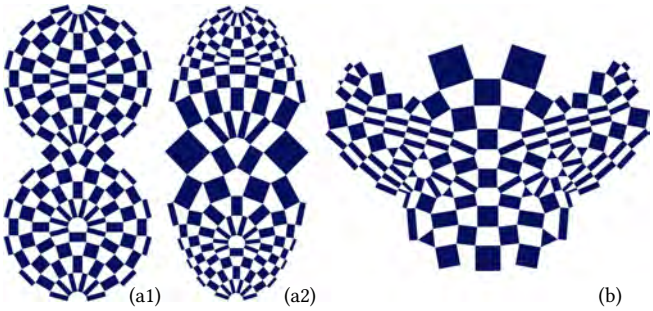


Fig. 19. 2D patterns with arbitrary rectangles. They are created by allowing each black rectangle to scale uniformly by a different ratio. The control meshes of (a1) and (a2) have the same combinatorics but different boundary shapes. (b) is a deformed version of the first Mascot design (Fig. 18 (b)).

7 CONCLUSION

Inspired by the logo design for the Olympic games in Tokyo, we study checkerboard patterns with black rectangles. These patterns can be 2D or 3D. The black faces are always (planar) rectangles and the white faces are arbitrary polygons. To generate these patterns, we propose a generalization of 2D rhombic tilings to 3D: quad meshes with orthogonal diagonals. We analyze the geometry of such meshes and present two novel algorithms. First, we propose a novel 2D tiling algorithm to generate checkerboard pattern with three types of black rectangles based on integer programming. Second, we propose a numerical optimization algorithm to generate checkerboard pattern from input quad meshes.

ACKNOWLEDGMENTS

This research was supported by the KAUST Office of Sponsored Research under contract no. OSR-CRG2018-3730. We thank Alexander Bobenko and Mikhail Skopenkov for pointing out connections of our work to discrete complex analysis and Martin Reis for the architectural rendering. The algorithm extension in Section 4 was proposed by an anonymous reviewer who realized the problems of the basic algorithm applied to difficult input meshes.

REFERENCES

1731. Varignon's theorem. https://en.wikipedia.org/wiki/Varignon%27s_theorem

J. E. Adkins. 1956. Finite Plane Deformation of Thin Elastic Sheets Reinforced with Inextensible Cords. *Philosophical Transactions of The Royal Society B: Biological Sciences* 249 (05 1956), 125–150. https://doi.org/10.1098/rsta.1956.0017

M. Aono, P. Denti, D. E. Breen, and M. J. Wozny. 1996. Fitting a woven cloth model to a curved surface: dart insertion. *IEEE Computer Graphics and Applications* 16, 5 (Sep. 1996), 60–70. https://doi.org/10.1109/38.536276

Ted D. Blacker and Michael B. Stephenson. 1991. Paving: A new approach to automated quadrilateral mesh generation. *Internat. J. Numer. Methods Engrg.* 32, 4 (1991), 811–847. https://doi.org/10.1002/nme.1620320410

Alexander Bobenko, Stefan Sechelmann, and Boris Springborn. 2016. *Discrete Conformal Maps: Boundary Value Problems, Circle Domains, Fuchsian and Schottky Uniformization.* 1–56. https://doi.org/10.1007/978-3-662-50447-5_1

Alexander Bobenko and Mikhail Skopenkov. 2012. Discrete Riemann surfaces: Linear discretization and its convergence. *Journal für die reine und angewandte Mathematik (Crelles Journal)* 0 (10 2012). https://doi.org/10.1515/crelle-2014-0065

Alexander Bobenko and Yuri Suris. 2008. *Discrete differential geometry: Integrable Structure.* American Math. Soc.

Alexander I. Bobenko and Felix Günther. 2017. Discrete Riemann surfaces based on quadrilateral cellular decompositions. *Advances in Mathematics* 311 (2017), 885 – 932. https://doi.org/10.1016/j.aim.2017.03.010

Pafnuty Lvovich Chebyshev. 1878. Sur la coupe des vetements, "On the cutting of garments". *Association francaise pour l'avancement des sciences* (1878), 154–155.

Xuelin Chen, Honghua Li, Chi-Wing Fu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2018. 3D Fabrication with Universal Building Blocks and Pyramidal Shells. *ACM Trans. Graph.* 37, 6, Article 189 (Dec. 2018), 15 pages. https://doi.org/10.1145/3272127.3275033

Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qixing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. 2015. Dapper: Decompose-and-pack for 3D Printing. *ACM Trans. Graph.* 34, 6, Article 213 (Oct. 2015), 12 pages. https://doi.org/10.1145/2816795.2818087

Tokyo 2020 Organizing Committee. 2016. Tokyo 2020 Emblems. https://tokyo2020.org/en/games/emblem/

Keenan Crane. 2013. Conformal Geometry Processing. *PhD thesis, Caltech* (June 2013).

Keenan Crane. 2019. Conformal Geometry of Simplicial Surfaces. *AMS Proceedings of Symposia in Applied Mathematics* (2019).

Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire Mesh Design. *ACM Trans. Graph.* 33, 4, Article 66 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601106

Branko Grünbaum and Geoffrey Colin Shephard. 2016. *Tilings and Patterns: Second Edition.* Dover Publications.

Xianfeng Gu, Ren Guo, Feng Luo, Jian Sun, and Tianqi Wu. 2018a. A discrete uniformization theorem for polyhedral surfaces II. *Journal of Differential Geometry* 109, 3 (07 2018), 431–466. https://doi.org/10.4310/jdg/1531188190

Xianfeng Gu, Feng Luo, Jian Sun, and Tianqi Wu. 2018b. A discrete uniformization theorem for polyhedral surfaces. *Journal of Differential Geometry* (2018).

Xianfeng Gu and Shing-Tung Yau. 2008. *Computational Conformal Geometry.* International Press.

Caigui Jiang, Chengcheng Tang, Amir Vaxman, Peter Wonka, and Helmut Pottmann. 2015. Polyhedral Patterns. *ACM Trans. Graphics* 34, 6 (2015). Proc. SIGGRAPH Asia.

Serdar Karademir, Oleg A. Prokopyev, and Robert J. Mailloux. 2016. Irregular polyomino tiling via integer programming with application in phased array antenna design. *Journal of Global Optimization* 65, 2 (01 Jun 2016), 137–173. https://doi.org/10.1007/s10898-015-0354-8

Richard Kenyon. 1993. Tiling a polygon with parallelograms. *Algorithmica* 9, 4 (01 Apr 1993), 382–397. https://doi.org/10.1007/BF01228510

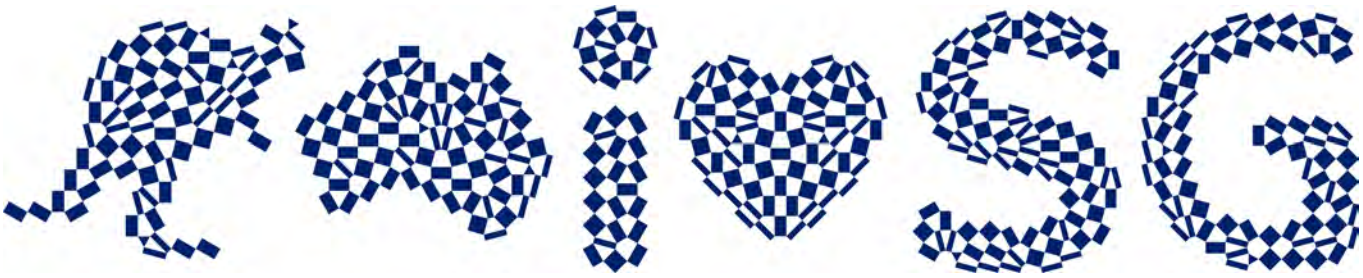


Fig. 20. Checkerboard patterns with boundaries generated from reference 2D shapes.

Fig. 21. 2D checkerboard patterns lifted to 3D while still retaining that the black rectangles of the same type have the same aspect ratio.

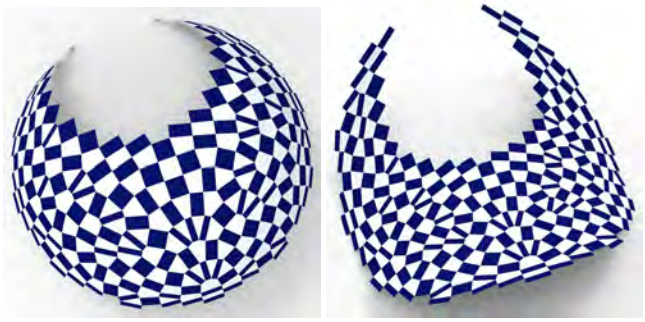


Fig. 22. Two 3D designs created by lifting our bigger version of the Paralympics logo (Fig. 17 (c)) to a sphere (a) and a hyperbolic paraboloid (b).

Richard Kenyon. 2002. The Laplacian and Dirac operators on critical planar graphs. *Inventiones mathematicae* 150, 2 (01 Nov 2002), 409–439. https://doi.org/10.1007/s00222-002-0249-4

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete Conformal Mappings via Circle Patterns. *ACM Trans. Graph.* 25, 2 (apr 2006), 412–438. https://doi.org/10.1145/1138450.1138461

Mina Konaković, Keenan Crane, Bailin Deng, Sofien Bouaziz, Daniel Piker, and Mark Pauly. 2016. Beyond Developable: Computational Design and Fabrication with Auxetic Materials. *ACM Trans. Graph.* 35, 4, Article 89 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925944

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.* 25, 3 (2006), 681–689. Proc. SIGGRAPH.

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. 2015. Legolization: Optimizing LEGO Designs. *ACM Trans. Graph.* 34, 6, Article 222 (Oct. 2015), 12 pages. https://doi.org/10.1145/2816795.2818091

Tobias Martin, Pushkar Joshi, Miklós Bergou, and Nathan Carr. 2013. Efficient Nonlinear Optimization via Multi-scale Gradient Filtering. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 89–100.

Stefanie Mueller, Tobias Mohr, Kerstin Guenther, Johannes Frohnhofen, and Patrick Baudisch. 2014. faBrickation: Fast 3D printing of Functional Objects by Integrating Construction Kit Building Blocks. *Conference on Human Factors in Computing Systems (CHI)* (2014). https://doi.org/10.1145/2556288.2557005

Changhyup Park, Jae-Seung Noh, Il-Sik Jang, and Joe M. Kang. 2007. A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints. *Computer-Aided Design* 39 (2007), 258–267.

Chi-Han Peng, Helmut Pottmann, and Peter Wonka. 2018. Designing Patterns Using Triangle-quad Hybrid Meshes. *ACM Trans. Graph.* 37, 4, Article 107 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201306

A. C. Pipkin. 1986. Continuously distributed wrinkles in fabrics. *Archive for Rational Mechanics and Analysis* 95, 2 (1986), 93–115.

Helmut Pottmann, Michael Eigensatz, Amir Vaxman, and Johannes Wallner. 2015. Architectural Geometry. *Computers and Graphics* 47 (2015), 145–164. https://doi.org/10.1016/j.cag.2014.11.002

Helmut Pottmann, Yang Liu, Johannes Wallner, Alexander Bobenko, and Wenping Wang. 2007. Geometry of Multi-layer Freeform Structures for Architecture. *ACM Trans. Graph.* 26 (2007), #65,1–11. Proc. SIGGRAPH.

Helmut Pottmann and Johannes Wallner. 2008. The focal geometry of circular and conical meshes. *Adv. Comp. Math* 29 (2008), 249–268.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018a. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM Transactions on Graphics* 37, 2, Article 16 (2018), 17 pages. https://doi.org/10.1145/3180494

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018b. The Shape Space of Discrete Orthogonal Geodesic Nets. *ACM Transactions on Graphics* 37, 6, Article 228 (2018), 17 pages.

R. S. Rivlin. 1964. Networks of Inextensible Cords. *Nonlinear Problems of Engineering* (1964), 51–64.

R. S. Rivlin. 1997. Plane Strain of a Net Formed by Inextensible Cords. *Collected Papers of R.S. Rivlin* (1997), 511–534.

Henrik Schumacher. 2017. On H^2-gradient Flows for the Willmore Energy. *arXiv preprint arXiv:1703.06469* (2017).

Mikhail Skopenkov. 2013. The boundary value problem for discrete analytic functions. *Advances in Mathematics* 240 (2013), 61 – 87. https://doi.org/10.1016/j.aim.2013.03.002

Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal Equivalence of Triangle Meshes. *ACM Trans. Graph.* 27, 3, Article 77 (aug 2008), 11 pages. https://doi.org/10.1145/1360612.1360676

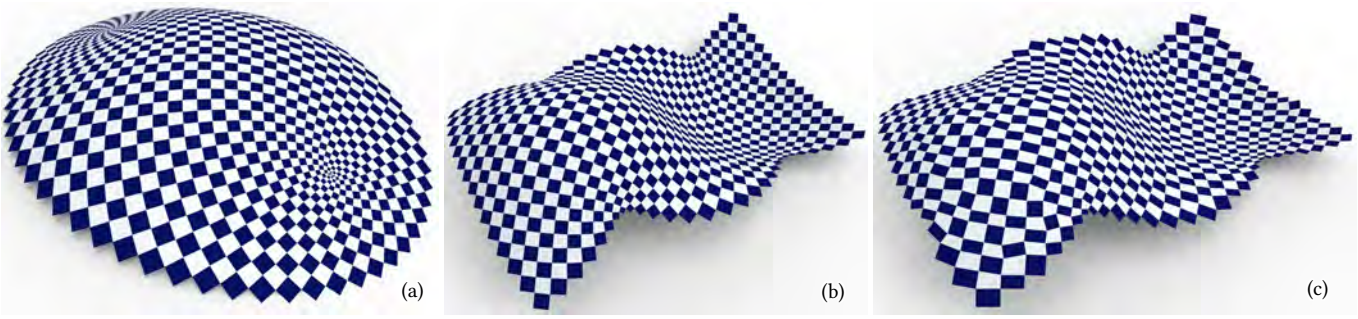


Fig. 23. 3D checkerboard patterns with black rectangles and white planar faces. In (a) and (b), all black faces are squares. In (c), the black faces are randomly assigned to one of three prescribed aspect ratios: 1, $\sqrt{2}$, and $\sqrt{3}$.

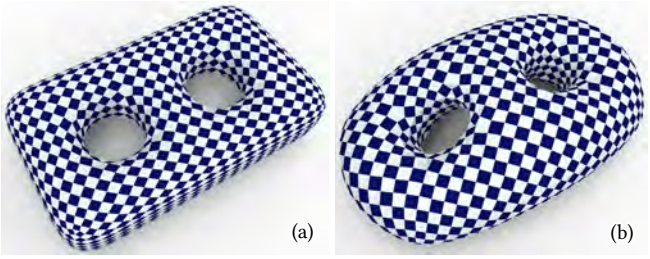


Fig. 24. Shape change due to topology. (a) A control mesh of a double torus, after edge midpoint subdivision, yields a pattern with black parallelograms. (b) Optimizing for black squares changes the shape.

Fig. 25. An architectural model inspired by the department of Islamic art structure in the Louvre.

Kenneth Stephenson. 2005. *Introduction to Circle Packing: The Theory of Discrete Analytic Functions.* Cambridge University Press.

Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with Polyhedral Meshes Made Simple. *ACM Trans. Graphics* 33, 4 (2014). Proc. SIGGRAPPH.

B. P. van West, R. B. Pipes, and M. Keefe. 1990. A Simulation of the Draping of Bidirectional Fabrics over Arbitrary Surfaces. *The Journal of The Textile Institute* 81, 4 (1990), 448–460.

David R. White and Paul Kinney. 2007. Redesign of the Paving Algorithm : Robustness Enhancements through Element by Element Meshing. *6th International Meshing Roundtable* (2007), 323–335.

ADDITIONAL MATERIALS

7.1 Boundary Editing

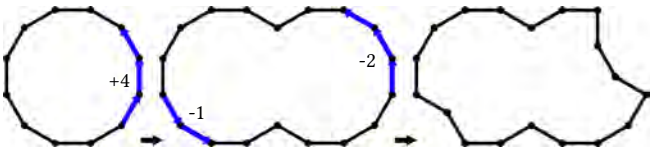


Fig. 26. Boundary editing example. The first edit expands by a magnitude of 4 and with the additional "circular" constraint. The second edit made two subtractions at two different locations with magnitudes of −2 and −1.

An initial boundary can be given as an arbitrary 2D piece-wise linear loop using the approximation method described in Section 3.4 in [Peng et al. 2018]. In addition, we propose an interactive tool to design admissible boundaries. Starting from an admissible boundary, the tool allows users to iteratively edit the boundary while keeping it admissible.

An editing operation replaces a subset of the current boundary with a new sequence of half-edges. For an operation, the user specifies: 1) the subset of the current boundary to replace and 2) the *magnitude* of the operation, which is a signed non-zero integer. The sign of the magnitude denotes whether the operation is an *expansion* or a *subtraction*. For simplicity, we assume the new sequence of half-edges is a subset of a convex loop (i.e., an admissible boundary without turning angles > 180°). We denote the directions of the first and last half-edges of the current subset as d_0 and d_1. After the replacement, the directions of the first and last half-edges becomes $(d_0 - x)$ mod 12 and $(d_1 + x)$ mod 12, shortened as D_0 and D_1. If $D_0 \leq D_1$, we encode the directions of a subset of a convex loop's boundary in counterclockwise order as $\{E_{D_0}, E_{D_0+1}, E_{D_0+2}, ..., E_{D_1}\}$ where these E_i vectors, $i \in [0, 11]$, are one of the twelve 4D direction vectors for half-edges (see Eq. 5.1). Otherwise, we encode it in clockwise order as $\{E_{D_0}, E_{D_0-1}, E_{D_0-2}, ..., E_{D_1}\}$. We then solve the following IP problem for the counterclockwise case:

$$\sum X_{D_0} E_{D_0} + X_{D_0+1} E_{D_0+1} + ... + X_{D_1} E_{D_1} = Z, \qquad (12)$$

and for the clockwise case change the order. Length variables X_{D_0} to X_{D_1} denote the numbers of half-edges in the convex loop at the specific directions. Z is the 4D offset vector from the first to the last vertices of the boundary subset to be replaced. Additionally, a *circular* constraint can be added such that all the length variables are non-zero. In summary, solving Eq. 12 gives us a subset of half-edges of a convex loop that seamlessly replace the original subset of boundary. See Fig. 26 and Fig. 27 for examples.

7.2 Additional Figures

In Fig. 28, we show all of our variations for the Tokyo 2020 logo design. In Fig. 29, we show the quad mesh of the bunny example. In Fig. 30, we show the control meshes of the 2D pattern results shown in the paper.

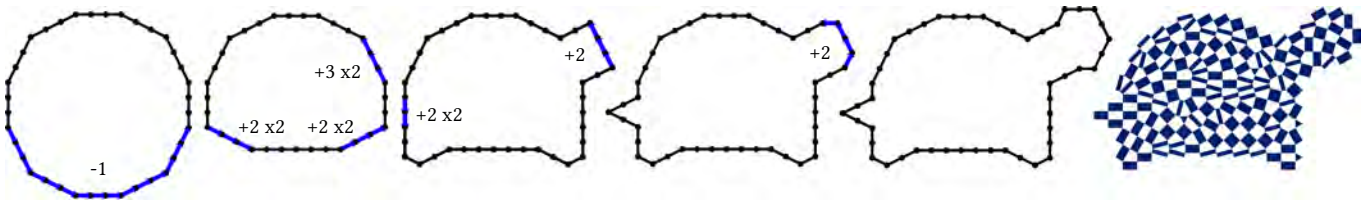


Fig. 27. Making a turtle-shaped boundary using a series of editing operations. "x2" means applies an operation two times.

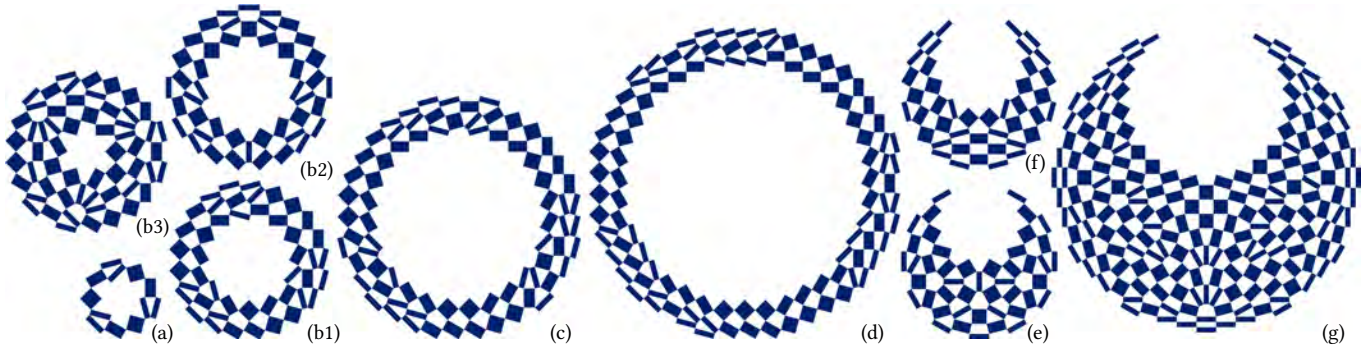


Fig. 28. All of our variations of the Tokyo 2020 logo design. (a), (b1), (c), and (d): we create a series of ring-shaped patterns, similar to the Olympics logo (Fig. 2 (a)), with the same 3-way rotational symmetry but in different scales (measured by the width of the outer boundary). (b1), (b2), and (b3) have the same scale as the original Tokyo design, of which (b2) is a ring-shaped pattern with a left-right reflective symmetry and (b3) is a thicker ring-shaped pattern. (e): a remake of the Paralympic Games logo (Fig. 2 (b)) with a "fractured" global style. (f): a design with the same scale but with a slightly different boundary. (g): a bigger design with a left-right reflective symmetry.

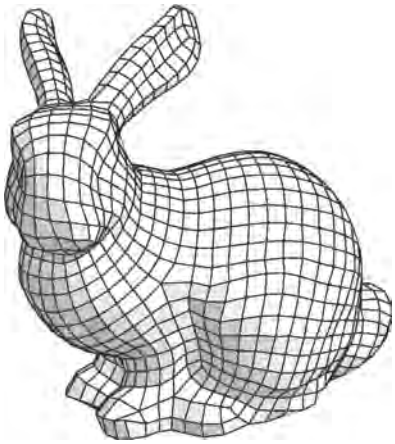


Fig. 29. The quad mesh of the bunny model.

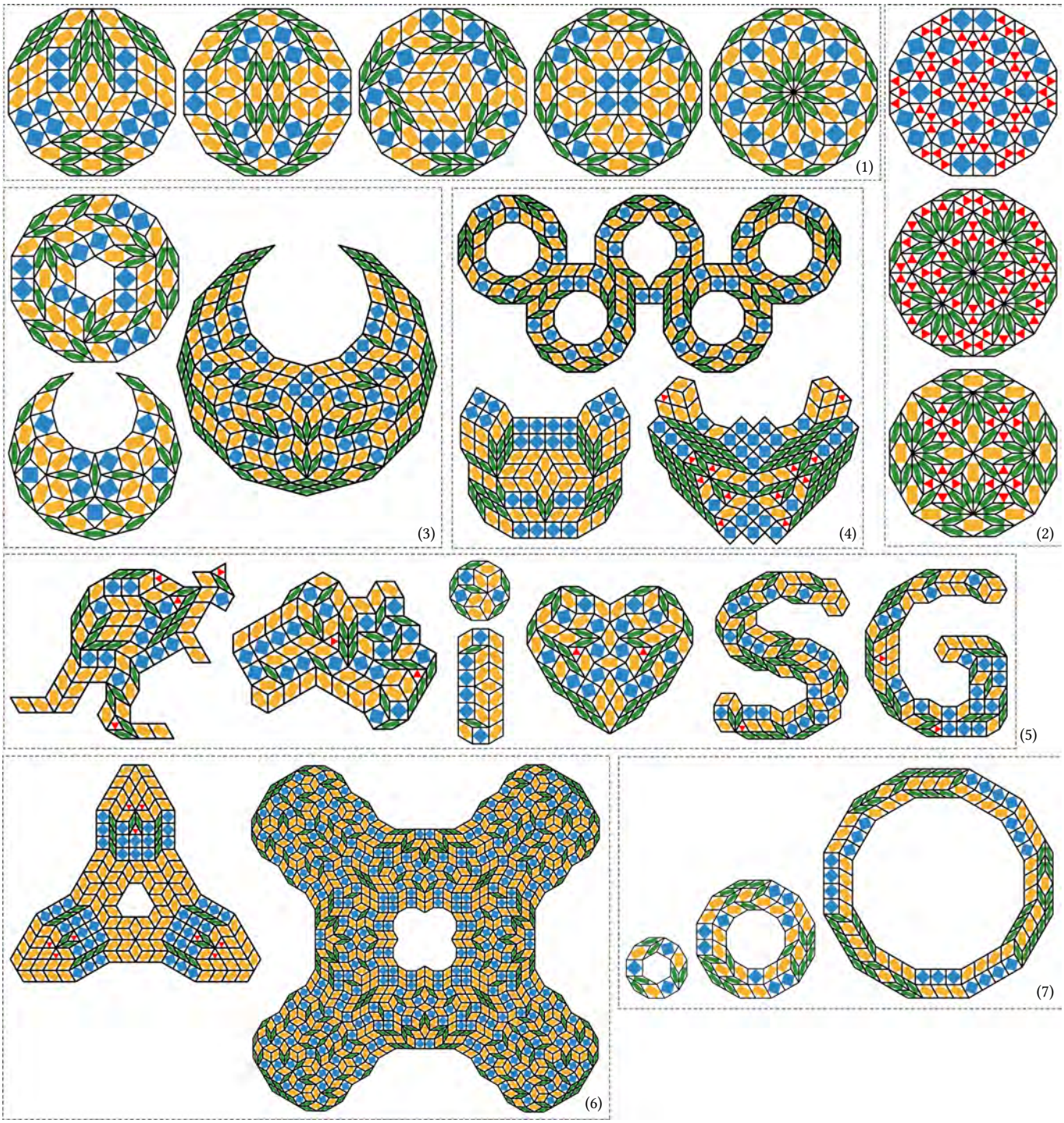


Fig. 30. Control meshes of 2D pattern results shown in the paper. (1): Fig. 14. (2): Fig. 15. (3): Fig. 17. (4): Fig. 18. (5): Fig. 20. (6): Fig. 21. (7): Fig. 28 (a), (b1), and (c). Each type of faces is colored with a different color.