Origamizing Polyhedral Surfaces

Tomohiro Tachi

Abstract—This paper presents the first practical method for "origamizing" or obtaining the folding pattern that folds a single sheet of material into a given polyhedral surface without any cut. The basic idea is to tuck fold a planar paper to form a three-dimensional shape. The main contribution is to solve the inverse problem; the input is an arbitrary polyhedral surface and the output is the folding pattern. Our approach is to convert this problem into a problem of laying out the polygons of the surface on a planar paper by introducing the concept of tucking molecules. We investigate the equality and inequality conditions required for constructing a valid crease pattern. We propose an algorithm based on two-step mapping and edge splitting to solve these conditions. The two-step mapping precalculates linear equalities and separates them from other conditions. This allows an interactive manipulation of the crease pattern in the system implementation. We present the first system for designing three-dimensional origami, enabling a user can interactively design complex spatial origami models that have not been realizable thus far.

Index Terms—Origami, origami design, developable surface, folding, computer-aided design.

1 Introduction

Origami is an art of folding a single piece of paper into a variety of shapes without cutting or stretching it. Creating an origami with desired properties, particularly a desired shape, is known as *origami design*. Origami design is a challenge faced not only by origami artists but also by designers and engineers who apply origami to industrial purposes. The objective of this research is to *origamize* a given polyhedral surface, defined in this paper as an origami design to obtain a folding pattern that folds into a given three-dimensional shape.

1.1 Developable Surfaces

A *folded state* of origami is roughly defined as a topological disk in three-dimensional space onto which a square can be isometrically mapped with a one-to-one mapping such that any portions of the surface do not intersect but are allowed to touch each other. A precise mathematical description of origami is derived by Demaine and O'Rourke [1].

In an engineering sense, this isometric mapping represents the deformation of the surface of hard thin sheet materials such as paper, plastic, and metal sheet. The surface obtained by such a deformation is termed developable surface, which traditionally implies a smooth, i.e., C^1 continuous, developable surface.

C^1 developable surfaces have been used for architectural and other industrial designs since they can be produced by simply bending the surfaces of continuous sheet materials. Methods to design a freeform surface by assembling developable or nearly developable surfaces have been proposed. Mitani and Suzuki [2] and Massarwi et.al. [3] constructed an arbitrary surface with triangle strips, each of which approximates a C^1 developable surface. Other methods have been proposed to obtain nearly developable patches represented as triangle meshes, either by segmenting the surface through the fitting of the patches to cones, as proposed by Julius [4], or by minimizing the Gauss area, as studied by Wang [5]. Subag and Elber [6] approximated NURBS surfaces with piecewise C^1 developable surfaces. A method with planar quad mesh [7] successfully creates a discrete representation of a C^1 developable surface, and Pottmann et.al. [8] use planar quad mesh to approximate a freeform surface with developable panels.

On the other hand, an origami surface is a C^0 developable surface, which allows a finite number of creases. It is known that creating folds or creases onto a single surface results in the formation of a wide variety of forms such as curved folding created by Huffman [9], periodic folding created by Resch [10], [11], and various other contemporary origami sculptures. In the 2000s, sheet metal designs have been developed that apply curved folding, such as column covers by Lalvani [12] and a car model by Epps [13]; these designs suggest a new field of origami design for industrial purposes.

Thus far, several studies have been conducted to analyze or simulate three-dimensional surfaces with folds. Huffman [14] described creases on a developable surface using the Gauss map and introduced a method to create a special case of curved folding. Also there have been studies to computationally represent folds between piecewise linear approximation of developable surfaces. Kergosien et.al. [15] simulated the creases created on a curved paper, and Kilian et.al. [16] first succeeded in discretely representing a general case of curved folding. Piecewise linear origami is defined as rigid origami, whose local behavior around a vertex is investigated by Belcastro and Hull [17]. The rigid origami simulator [18] is an interactive system that animates piecewise linear origami in three-dimensional space. Another representation of origami is to use a discrete shell simulation, as used by Burgoon et.al. [19], which takes the elasticity of paper into account.

Despite conducting these analytical studies and developing simulational methods, no practical method for designing an

• *The author is with the Department of Architecture, the University of Tokyo, Tokyo, Japan. E-mail: ttachi@siggraph.org*

origami surface has been developed. The objective of this study is to realize a freeform shape out of a single developable surface with creases. Although the key application of the proposed method is to create a new artistic origami, this method is also expected to be applicable for industrial purposes in the future. Prospective applications include the design of architectural elements such as facades, roofs, furniture, and light fixtures, which require both designability and manufacturability

1.2 Flat Folding

Flat-foldable origami is a special type of origami where every point on a folded state lies on a plane. In the case of flat-foldable origami, a *crease pattern*, i.e., the inverse image of foldlines, is described as a simple straight-edged two-dimensional graph drawn on a plane, each line segment of which represents a mountain or a valley. The crease pattern represents the points on which the points on a paper are mapped; however, if the points are folded in a manner in which they overlap, determining the overlapping ordering of the paper is not a trivial problem. It has been proved by Bern and Hayes [20] that determining a valid configuration of a folded state only from a crease pattern is an NP-complete problem with worst-case complexity.

However, some general design problems are solved by dividing a global crease pattern into fragments, each of which is bounded by a polygonal region. The crease pattern of such a fragment is designed to have a characteristic such that desired points and lines are folded to overlap each other. In an origami design, such a fragment is termed *molecule*, coined by Meguro [21]. For example, a molecule that folds the perimeter of a given polygon onto a common line is applied for the fold and cut problem [22]. A further extended molecule termed *universal molecule* provides the flexibility of deciding which parts of the perimeter must overlap; the correspondence between the parts is represented with a circle packing pattern [23]. The concept of universal molecule is applied for designing a tree-shaped origami [23] (Section 1.3.1), solving the fold-and-cut problem [24], and flat-folding polyhedra [25].

In this paper, another type of molecule termed *tucking molecule* is used. Tucking molecules are partially flat-foldable structures that assemble corresponding edges of polygons on a three-dimensional surface.

1.3 Previous Origami Design Methods

1.3.1 Tree Method

The *tree method* has been the only existing practical computational origami design method for realizing desired shapes. Its basic concept was first introduced in [21], which states that an arbitrary tree-shaped origami figure can be constructed from a pattern of circles and rivers packed into a square. Lang [23] described the theory of the tree method with some proofs and proposed a computational algorithm. The proposed method generates a crease pattern that folds into a *base*, i.e., a folded shape whose projection to a plane is exactly the same as the given tree shape with arbitrary edge lengths and connectivity. The algorithm is implemented as an origami design software TreeMaker [26]. The tree method enables the creation of origami with vast complexities; however, it is only possible to control one-dimensional properties, i.e., the lengths of flaps. An intuitive process wherein an experienced origami artist performs the “shaping” is essential to transform such an origami base into the final shape. In addition, it is virtually impossible even for experienced origami artists to create a desired three-dimensional shape using this approach. In contrast, our method can precisely represent three-dimensional shapes without additional shaping.

1.3.2 Folding a Polyhedron

Demaine et al. [27] proved that any polyhedron can be created by folding a square piece of paper. The basic idea of the algorithm provided in the proof is to fold the square sheet of paper into a thin strip, and then, wrap the strip around the desired polyhedron. It is practically impossible to design any actual model using this approach, because of the inefficiency arising from the algorithm, which relies on an extremely narrow strip. In addition, the strip does not stably sustain the three-dimensional shape if it is constructed by the abovementioned approach. Section 9.2 discusses the efficiency and structural stiffness of the folded model.

Tanaka [28] proposed another technique based on folding a paper into the development of the given polyhedron. This technique is also not practical for the same reasons — the generated crease pattern is inefficient, and the folded shape is unstable, because the algorithm begins with the formation of a complex tree-shaped polygon.

1.3.3 Tucking

It is a well-known fact that a flat sheet of paper can be curved by tuck folding. Some origami artists empirically apply the tuck-folding technique to shape a three-dimensional surface. Based on the idea of tuck-folding, the author has previously proposed a technique for designing a three-dimensional origami surface by tucking and hiding the unwanted areas of a paper [29]. The work proposes the concept of tucking molecules, i.e., fragments of crease pattern specially designed for tucking. However, no algorithm or system has been developed, and the entire design process relies on trial and error using a conventional paper craft software [30] and a vector drawing software. It sometimes takes several weeks to create a crease pattern for realizing a three-dimensional origami model using this technique.

1.4 Contributions

In this study, the first practical method and the system for designing a three-dimensional origami model have been developed. Although the method does not guarantee the origamization of an arbitrary polyhedron, it is practical enough to enable elaborate three-dimensional origami models (Figure 1). Following are the main contributions of this study.

1.4.1 Polygons and Tucking Molecules

Our technique is developed on the basis of the concept of tucking molecules proposed in the author's previous work [29]. We additionally formalize the conditions and parameterize the configuration. This parameterization translates the problem of designing three-dimensional origami into a problem of laying out polygons on a plane. In order to implement this technique, sufficient conditions for obtaining a valid crease pattern are investigated. It is shown that the conditions are represented by non-linear equations and inequalities.

1.4.2 Mapping and Edge Splitting

In this study, a novel algorithm has been developed to solve the above-mentioned conditions. The algorithm is based on two-step mapping and edge splitting.

The first step of mapping fixes the rotations of the polygons. This converts non-linear equations into linear equations, which are then precalculated. Under these precalculated linear equations, the remaining inequalities are solved by an iterative method. The precalculation enables a fast computation for each iterative step, which also allows a real-time human interaction.

Edge splitting is a recursively applicable procedure that locally changes the crease pattern without changing the original surface. This procedure decreases the number of conditions to be solved. Mapping is performed under these relaxed conditions, following which edge splitting is performed recursively on the given pattern.

1.4.3 Interactive Design System

Our algorithm was implemented as an interactive system for origami design. The system automatically generates a crease pattern from an input polyhedral surface while allowing users to edit the topology of the three-dimensional mesh and to modify the resulting crease pattern. These tools are designed to be consistent with the required disk topology and the conditions for origamization.

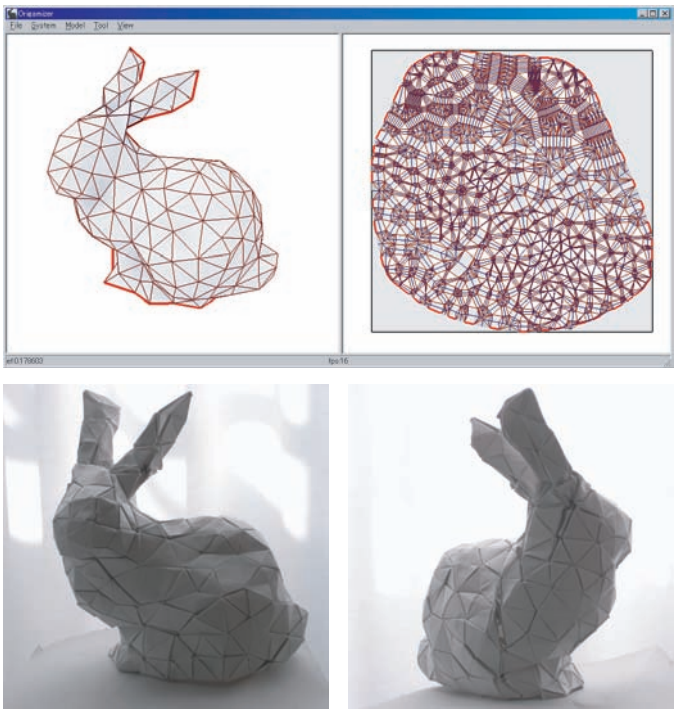


Fig. 1. Top: Desired polyhedral model (Stanford Bunny) and the created crease pattern. Bottom: World's first origami Stanford Bunny.

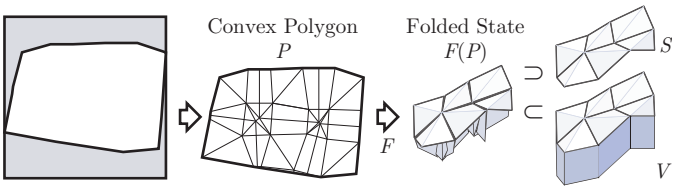


Fig. 2. Origamization is achieved by creating a crease pattern representing $F(P)$ such that $F(P) \subset V$ and $F(P) \supset S$. The convex polygon P can be further folded from a square.

2 Method Overview

2.1 Description of the Problem

In this paper, the origamization of a polyhedral surface is defined as, obtaining a crease pattern of origami that constructs a polyhedral surface with hidden tucks, instead of constructing only the exact surface.

Let S and $V(\supset S)$ denote a set of all points on the given polyhedral surface and the solid enclosed by S, respectively. Then, the goal is to create the crease pattern on a planar convex polygon P so that the folded state $F(P)$ is contained in V and S is contained in $F(P)$. S can be an orientable manifold with a boundary that does not completely enclose a volume, in which case V is given to indicate a finite region hidden behind S.

Note that the planar convex polygon P can be obviously folded from a square containing P by folding and hiding the unnecessary area behind. The minimum size of such a square paper is used when measuring the efficiency of the model (Section 9.2.3).

2.2 Procedure

2.2.1 Cutting to a Disk

The first step toward origamizing a surface not homeomorphic to a disk is to construct a polygonal schema, which is a polyhedral surface homeomorphic to a disk that covers the original surface. A polygonal schema is represented by cutlines along the edges of the surface which correspond to the boundary of the disk. Several algorithms are proposed to obtain "nice" polygonal schemata [31], [32], [33], and in the system proposed, a combination of flooding and modification by the user is adopted, as discussed later in Section 8.1.

2.2.2 Mapping Surface Polygons

The next major step is to separate individual polygons of the given polyhedral mesh and map each of them congruently on a plane. The problem here is to obtain a folding that folds the mapped polygons properly back to S and the gap between the polygons to $V \setminus S$. The shape of the gap, defined by the layout

of the polygons, determines whether such a folding is possible or not, the conditions on which are investigated in Section 4. The layout is determined by solving these conditions as shown in Section 6.

2.2.3 Generating a Crease Pattern

In order to tuck fold the gap to make it completely hidden behind the surface, the gap is subdivided into simple polygons with crease patterns termed *tucking molecules*, as described in Section 3. The crease pattern on a tucking molecule is created such that it "glues" separated vertices or a pair of edges by folding itself. The combination of these molecules enables a folding pattern that hides the unwanted gaps and makes the separated polygons connected again (Figure 3).

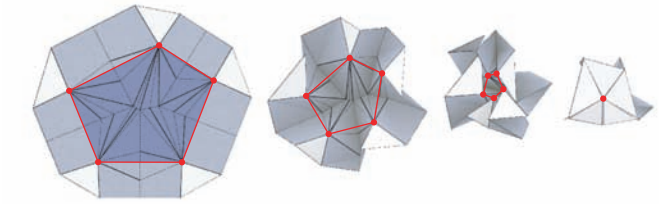


Fig. 3. Folding motion of the crease pattern. The tuck is hidden and the surface polygons are glued together as the corresponding vertices are folded to the same position.

3 Tucking Molecules

Two types of tucking molecules are used in this study. One type is the *edge-tucking molecule*, which is a quadrilateral with a crease pattern inserted between a pair of edges corresponding to a single edge on S. The second type is the *vertex-tucking molecule*, which is an N-gon with a crease pattern surrounded by N edge-tucking molecules, whose N vertices correspond to a single vertex on S. Therefore, a planar region is tessellated into the original polygons of the surface, edge-tucking molecules, and vertex-tucking molecules (Figure 4). This derived tessellation is termed *molecule mesh*.

3.1 Edge-Tucking Molecule

Assume that surface polygons $F(P_0)$ and $F(P_1)$ share the same segment AB on S, as shown in Figure 5 Left. Let $\mathrm{A}_0\mathrm{B}_0$ and $\mathrm{A}_1\mathrm{B}_1$ denote the edges that correspond to AB. Then, the edge-tucking molecule corresponds to quadrilateral $\mathrm{A}_0\mathrm{B}_0\mathrm{B}_1\mathrm{A}_1$. In general, the crease pattern can be defined if and only if the quadrilateral is a simple polygon with a positive signed area, i.e., without flipping, and the condition $\min(\mathrm{A}_0\mathrm{B}_1, \mathrm{A}_1\mathrm{B}_0) \geq \mathrm{AB}$ is satisfied.

In addition, the proposed algorithm assumes that the edge-tucking molecules are symmetric, i.e., each quadrilateral is an isosceles trapezoid, where $\mathrm{A}_0\mathrm{A}_1$ is parallel to $\mathrm{B}_0\mathrm{B}_1$ (Figure 5 Right). In this case, the crease pattern of an edge-tucking molecule is simply a single valley crease ($+\pi$) on the axis of reflection in which the edges are reflected onto each other.

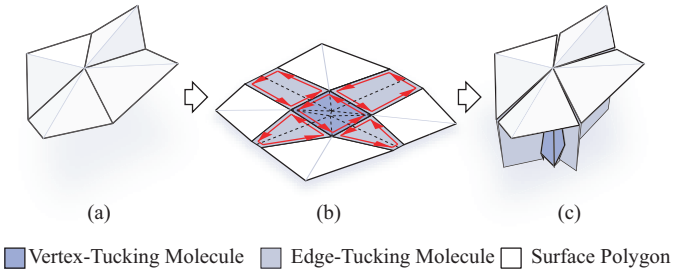


Fig. 4. (a) Polyhedral surface S. (b) Tessellation derived from S, constructed by inserting tucking molecules between mapped surface polygons. (c) Gap is tuck-folded to be completely hidden behind the surface, thereby the surface polygons are "glued" together again.

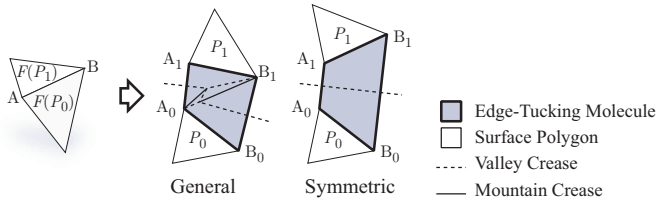


Fig. 5. Edge-tucking molecules. Segments $\mathrm{A}_0\mathrm{B}_0$ and $\mathrm{A}_1\mathrm{B}_1$ are folded onto the same edge AB.

3.2 Vertex-Tucking Molecule

A vertex-tucking molecule corresponds to an N-gon that joins N pieces of the edge-tucking molecules sharing a single vertex. The surrounding vertices of a vertex-tucking molecule are folded onto a single vertex by folding along its crease pattern.

Such a crease pattern can be generated for a given shape of molecule by the Voronoi folding, i.e., folding along the Voronoi diagram. The concept of Voronoi folding is based upon the idea of the circumcentric folding of a triangle, i.e., folding the triangle along the perpendicular bisectors of its sides and folding three vertices onto a single point. Following are the procedure (Figure 6).

1) Draw the Voronoi diagram with valley creases ($+\pi$) whose generating points are the vertices of the molecule.
2) Connect each Voronoi vertex and adjacent generating points with mountain creases (the fold angles are determined by the tuck proxy, defined later in Section 4.3).
3) Add crimp folds that makes the folded state fit the curvature of the surface polygons (discussed later in Section 4.3).

4 Conditions

The sufficient conditions for achieving origamization are described by equalities and inequalities on the configuration of the mapped surface polygons. The conditions can be stated as follows.

1) All surface polygons are isometrically mapped, and all pairs of mapped edges are symmetrically aligned (Equality conditions: Section 4.1).

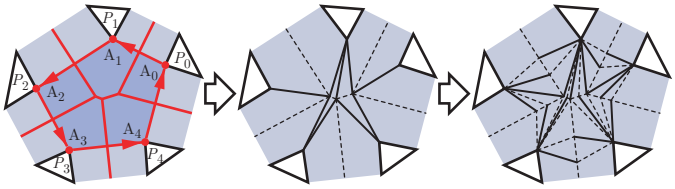


Fig. 6. Crease pattern for creating a vertex-tucking molecule is generated using the Voronoi diagram.

2) The mapped polygons and tucking molecules tessellate a convex polygon, and the molecule mesh yields a valid crease pattern (2D inequality conditions : Section 4.2).
3) The folded state follows the curvature of S and is contained in V (3D inequality conditions: Section 4.3).

4.1 Equality Conditions

All mapped surface polygons are isometric and the edge-tucking molecules are symmetric. The configuration of the molecule mesh can be represented by distances and angles of the edge-tucking molecules.

For a pair of adjacent vertices i, j, angle $\theta(i,j)$ is defined as the angle between a pair of edges corresponding to edge ij, whose sign is assigned according to the orientation of the rotation about vertex i. The width $w(i,j)$ is defined as the signed length of the edge on the boundary of $\mathrm{VTM}(i)$ (vertex-tucking molecule corresponding to i) shared by $\mathrm{ETM}(i,j)$ (edge-tucking molecule corresponding to ij), whose negative value indicates that the molecule is crossed or flipped. Since edge-tucking molecules are represented by isosceles trapezoids, $\theta(j,i) = -\theta(i,j)$ and $w(j,i) = w(i,j) + 2\ell(i,j)\sin(\frac{1}{2}\theta(i,j))$, where $\ell(i,j)$ is the length of edge ij (Figure 7).

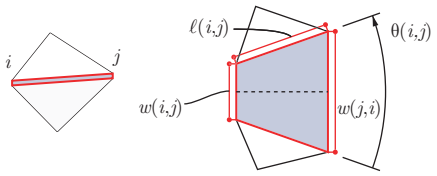


Fig. 7. Edge-tucking molecule is represented in terms of two parameters: $w(i,j)$ and $\theta(i,j)$.

We assume that an extra edge-tucking molecule termed *boundary molecule* is connected to each vertex-tucking molecule on the boundary of the surface. Then, all vertex-tucking molecules are surrounded by polygons and molecules. Similarly, variables of a boundary molecule adjacent to $\mathrm{VTM}(i)$ are denoted by $\theta(i,o)$ and $w(i,o)$, where the nominal vertex o represents the boundary.

The variables are constrained at each vertex surrounded by facets. Let j_n ($n = 0, \cdots, N-1$, where N is the valency of vertex i) denote the vertex adjacent to vertex i or boundary o connected counterclockwise in this ordering, and let $\alpha(i,j_n)$ denote the sector angle on S between $ij_{n-1(\mathrm{mod}N)}$ and ij_n.

Then, equality conditions around vertex i are given as follows.

$$\sum_{n=0}^{N-1} \theta(i,j_n) = 2\pi - \sum_{n=0}^{N-1} \alpha(i,j_n) \tag{1}$$

and

$$\sum_{n=0}^{N-1} w(i,j_n) \begin{bmatrix} \cos\left(\sum_{m=1}^{n} \Theta_m\right) \\ \sin\left(\sum_{m=1}^{n} \Theta_m\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{2}$$

where Θ_m is the external angle between the adjacent edges of a vertex-tucking molecule (Figure 8), given by

$$\Theta_m = \frac{1}{2}\theta(i,j_{m-1}) + \alpha(i,j_m) + \frac{1}{2}\theta(i,j_m) \tag{3}$$

The right-hand side of equation (1) is the Gauss area of the original surface at the vertex, which is modified to be zero by adding or subtracting extra angles using edge-tucking molecules. Equation (2) ensures that the vertex-tucking molecule forms a closed polygon.

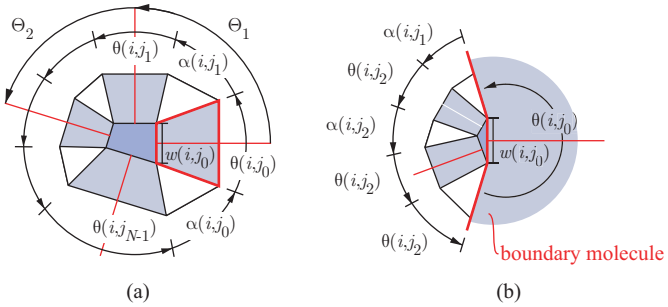


Fig. 8. (a) Vertex-tucking molecule surrounded by polygons and edge-tucking molecules. (b) Vertex-tucking molecule on the boundary.

4.2 2D Inequality Conditions

In order to ensure a valid crease pattern, it is necessary that a convex paper can be tessellated into surface polygons and tucking molecules (Section 4.2.1), and a valid crease pattern can be constructed from the molecule mesh (Section 4.2.2).

4.2.1 Convex Paper and Non-overlapping Conditions

The condition that satisfies the convexity of paper is given as follows. For every boundary molecule $\mathrm{ETM}(i,o)$,

$$\theta(i,o) \geq \pi \tag{4}$$

$$w(i,o) \geq 0 \tag{5}$$

Because the boundary of the molecule mesh is convex, no overlapping occurs without the local intersection or flipping of the tessellating elements, i.e., surface polygons, edge-tucking molecules, and vertex-tucking molecules. Thus the necessary and sufficient condition for obtaining a valid molecule mesh without overlapping is that every edge-tucking and vertex-tucking molecule is a simple polygon. Following conditions are used. For every $\mathrm{ETM}(i,j)$,

$$-\pi < \theta(i,j) < \pi \tag{6}$$

$$\min(w(i,j), w(j,i)) \geq 0, \tag{7}$$

and for every vertex of every $\mathrm{VTM}(i)$,

$$0 \leq \Theta_m < \pi \tag{8}$$

Here, we are using the sufficient condition for vertex-tucking molecules; conditions (7) and (8) keep every vertex-tucking molecule convex.

4.2.2 *Crease Pattern Validity*

It must be ensured that (i) a crease pattern is properly generated for each tucking molecule and that (ii) the patterns of adjacent molecules do not intersect, in order to obtain a valid crease pattern. It is obvious that every isosceles-trapezoidal edge-tucking molecule yields a valid crease pattern, i.e., a single extendable segment on the axis of reflection. Thus we will focus on the conditions (i) and (ii) for vertex-tucking molecules.

Condition (i) is satisfied for any vertex-tucking molecule that satisfies convexity conditions (7) and (8). This is because the convexity of the vertex-tucking molecule ensures that the perpendicular bisectors of all the sides of the molecule lie on the Voronoi edges.

Condition (ii) is satisfied by avoiding an intersection between the crease patterns of two adjacent vertex-tucking molecules (Figure 9). This type of intersection inside the edge-

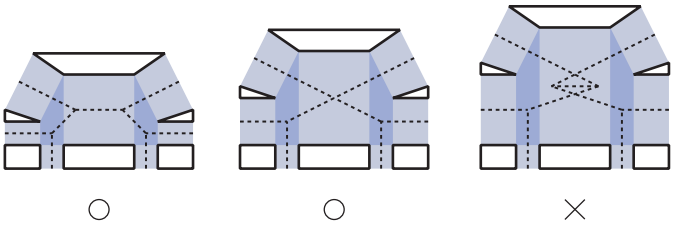


Fig. 9. Intersection of the crease pattern.

tucking molecule occurs when a Voronoi vertex of the adjacent vertex-tucking molecule traverses a certain distance across the boundary of the edge-tucking molecule. This distance is represented by the opposite angle of the Delaunay triangle incident to the concerned boundary (Figure 10). A sufficient condition for $\mathrm{ETM}(i,j)$ is expressed as the following inequality of this angle denoted by $\phi(i,j)$.

$$\phi(i,j) \leq \gamma(i,j) + 0.5\pi, \tag{9}$$

where $\gamma(i,j)$ is the angle between the border segment and the diagonal of $\mathrm{ETM}(i,j)$.

$$\gamma(i,j) = \gamma(j,i) = \arctan \frac{2\ell(i,j)\cos\frac{1}{2}\theta(i,j)}{w(i,j)+w(j,i)}$$

4.3 3D Inequality Conditions

It must be ensured that the folded shape finally follows the original surface S, and every part of the folded shape is contained in the solid V (Figure 2). The 3D conditions are defined as follows.

1) The folded tuck does not intersect and is contained in V.

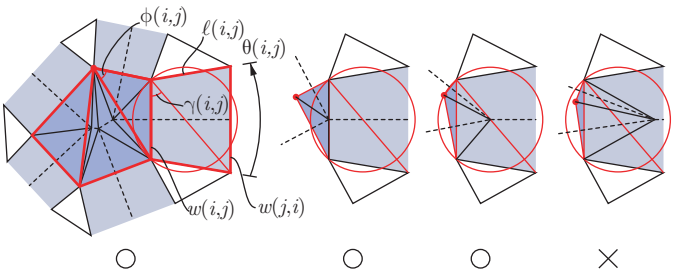


Fig. 10. Sufficient condition for avoiding the intersection of crease patterns is represented as an inequality using $\phi(i,j)$ and $\gamma(i,j)$.

2) The folded tuck forms the curvature of S.

We propose the idea of "tuck proxy" (Section 4.3.1) that translates the former condition into the inequalities between width and depth (Section 4.3.3) and the latter into the inequalities between angles (Section 4.3.2).

4.3.1 *Tuck Proxy*

Instead of directly estimating all the degrees of freedom in the three-dimensional configuration of a folded tuck, we assume that the folded tuck is contained in *tuck proxy*, i.e., the precalculated shape of the tuck, which is a subset of V, thereby deriving the sufficient conditions. Tuck proxy is defined as the union of connected triangle strips generated by inwardly extruding the edges of the surface (Figure 11 (a)).

Tuck proxy is generated in the following manner. First, we define the direction of each *joint axis*, i.e., the axis to which the triangle strips are connected at their terminal segments, by shooting a ray inward from each vertex. Subsequently, the ray is trimmed by an inward offset of the surface, so that the generated tuck proxy does not intersect itself. We connect each pair of the joint axes generated from a pair of adjacent vertices with a triangle strip. Then, we obtain a valid tuck proxy inside the volume (Figure11 (b)).

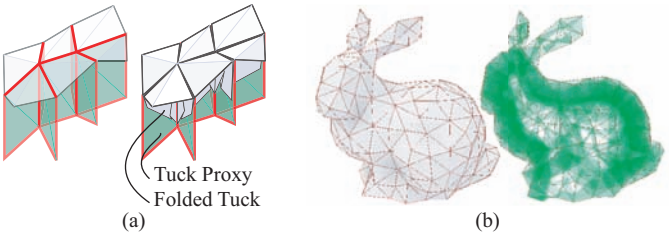


Fig. 11. (a) Folded tuck is assumed to be contained in the tuck proxy, which is an inward extrusion of the edges on the surface. (b) Surface polygons (left) and generated tuck proxy free of intersection (right).

4.3.2 *Tuck Angle Conditions*

As a consequence of 2D conditions, a vertex-tucking molecule can be folded such that the vertex and surrounding edges are connected (Figure 12 (a)). In order to ensure that the folded edge exactly follows the edge on S, the folded tuck is first

crimp-folded to enable the terminal segments of the strips to lie along the joint axis of the tuck proxy (Figure 12 (b)). Subsequently, the *tuck angle*, defined as the angle between the joint axis and the edge, is further adjusted by introducing an additional crimp fold. The addition of crimp fold reduces the tuck angle, but cannot increase it. Therefore, the conditions required to fit the edge $e(i,j)$ accurately on the original surface is expressed as the following inequality between the potential tuck angle along the folded paper $\tau(i,j)$ and the desired tuck angle along the tuck proxy $\tau'(i,j)$ (Figure 12).

$$\tau(i,j) \geq \tau'(i,j)$$

$\tau(i,j)$ can be expressed in terms of $\theta(i,j)$, and $\phi(i,j)$, and the inequality can be represented as follows:

$$\phi(i,j) - \frac{1}{2}\theta(i,j) \leq \pi - \tau'(i,j) \tag{10}$$

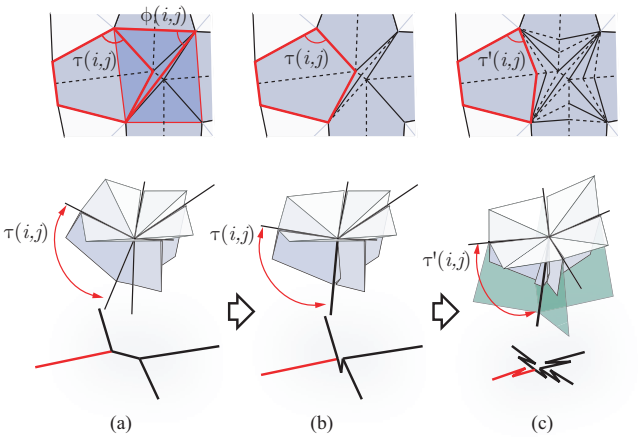


Fig. 12. Adjusting the tuck angle with crimp folds.

4.3.3 *Tuck Width Conditions*

The folded edge-tucking molecules and vertex-tucking molecules must be contained in the tuck proxy. This results in the formation of two inequalities for each vertex.

The width of the edge-tucking molecule is limited by the depth of the tuck proxy. This condition is described by the inequality between the length of the joint axis $d'(i)$ and the length of the crease on the edge-tucking molecule to be folded onto the axis $d_{\text{edge}}(i,j)$ (Figure 13).

$$d_{\text{edge}}(i,j) \leq d'(i),$$

which can be rewritten as

$$w(i,j) \leq 2\sin(\tau'(i,j) - \frac{1}{2}\alpha(i,j))d'(i). \tag{11}$$

The depth of the vertex-tucking molecule is also limited. Depth $d_{\text{vert}}(i)$ of $\text{VTM}(i)$ is defined as the maximum distance from vertex i to a point on the folded vertex-tucking molecule projected to the joint axis, which is also limited by $d'(i)$.

$$d_{\text{vert}}(i) \leq d'(i) \tag{12}$$

However, the latter condition (12) is reduced to the former one (11) by introducing a *sink fold* procedure. Sink fold is a common folding technique that reflects a part of the folded state about a plane. Here, we apply an open sink, which is a sink fold where all the mountain and valley assignments of the creases on the plane of reflection are the same, about a plane perpendicular to the joint axis. This maintains the developability of the surface and the flexibility of the folded shape, that is, edge-tucking molecules can be rotated along the joint axis after the sink fold.

Assume that every $\text{ETM}(i,j)$ adjacent to $\text{VTM}(i)$ satisfies inequality (11); then, $\text{VTM}(i)$ is sink-folded at depth $\max(d_{\text{edge}}(i,j))$ (Figure 13). This assumption sets $d_{\text{vert}}(i) = \max(d_{\text{edge}}(i,j))$, resulting in satisfying condition (12).

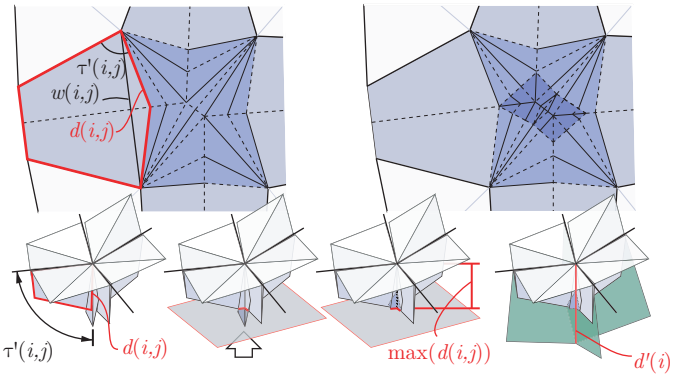


Fig. 13. If any $\text{ETM}(i,j)$ adjacent to $\text{VTM}(i)$ satisfies condition (11), the depth of the $\text{VTM}(i)$ can be adjusted with a sink fold.

5 Preprocess

5.1 Additional Edges

In order to avoid overconstraints by convex paper conditions, an extra edge-tucking molecule, a vertex-tucking molecule, and a digon are added to each edge on the boundary (Figure 14).

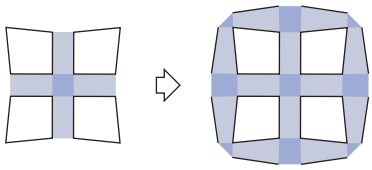


Fig. 14. Adding molecules to the boundary.

5.2 Triangulation

All concave polygons are triangulated, because concave polygons do not satisfy the following necessary conditions for (10) at the concave vertex between $\text{ETM}(i,j_0)$ and $\text{ETM}(i,j_1)$.

$$\tau'(i,j_0) + \tau'(i,j_1) < 2\pi - \alpha(i,j_1) \tag{13}$$

$$\max(\tau'(i,j_0), \tau'(i,j_1)) < \pi \tag{14}$$

Triangulation of polygons also helps solving the conditions for origamiziation by inserting flexible edge-tucking molecules between facets. For this reason, other convex polygons are also triangulated before mapping. However, the mapping algorithm itself can be applied to polyhedra with quadrilaterals, which was tested to yield valid results in practice (Section 9).

6 MAPPING

6.1 Variables and Constraints

Let N_{vert} and N_{edge} denote the number of vertex-tucking molecules and the number of edge-tucking molecules plus boundary molecules, respectively. The configuration of the molecule mesh is represented by two N_{edge}-vectors $\boldsymbol{\theta}$ and $\mathbf{w}$, whose n-th elements are $\theta(i,j)$ and $\min(w(i,j), w(j,i))$, respectively, where $\mathrm{ETM}(i,j)$ is assigned a number n.

These $2N_{\text{edge}}$ variables are constrained by $3N_{\text{vert}}$ equalities represented by (1) and (2). This constructs an underdetermined system, within which the other inequalities are solved. The degree of freedom of this system is calculated as $N_{\text{edge}} - 3$, because $N_{\text{vert}} = \frac{1}{3}N_{\text{edge}} + 1$ when all the polygons are triangulated and boundary edges are added.

The actual mapping algorithm is performed in the following two steps; (i) rotations of the polygons, represented by $\boldsymbol{\theta}$, are determined and (ii) translations of the polygons, represented by $\mathbf{w}$, are calculated.

6.2 Rotation

In the rotational phase, we obtain $\boldsymbol{\theta}$ that satisfies linear equation (1) and linear inequalities (4), (6), and (8).

For each vertex, $\boldsymbol{\theta}$ is constrained by equation (1), which can be represented by the following linear equation.

$$\mathbf{C}_\theta \boldsymbol{\theta} = \mathbf{g}, \tag{15}$$

where $\mathbf{C}_\theta$ is an $N_{\text{vert}} \times N_{\text{edge}}$ matrix whose m-th row corresponds to equation (1) for the m-th vertex, and $\mathbf{g}$ is the N_{vert}-vector whose element represents the Gauss area at each vertex. This is an underdetermined system having $N_{\text{edge}} - N_{\text{vert}} = \frac{2}{3}N_{\text{edge}} - 1$ degrees of freedom. The solution space is given as,

$$\boldsymbol{\theta} = \mathbf{C}_\theta^{+}\mathbf{g} + (\mathbf{I}_{N_{\text{edge}}} - \mathbf{C}_\theta^{+}\mathbf{C}_\theta)\boldsymbol{\theta}_0, \tag{16}$$

where ${\mathbf{C}_\theta}^{+} = \mathbf{C}_\theta^{\mathrm{T}}(\mathbf{C}_\theta\mathbf{C}_\theta^{\mathrm{T}})^{-1}$ is the pseudo-inverse of $\mathbf{C}_\theta$, and $\boldsymbol{\theta}_0$ is an arbitrary configuration. Within this constrained space, inequalities represented by (4), (6), and (8) are solved.

We solve the inequalities under the constraints by iteratively updating $\boldsymbol{\theta}_0$ and projecting it to the constrained space. $\boldsymbol{\theta}_0$ is updated according to the gradient of the penalty function E_θ defined as the sum of squares of the errors from linear inequalities (4), (6), and (8). Due to the convexity of these conditions, the calculation always converges to the global minimum. Additionally, although not proved, it is observed in every tested model, that the global minimum is 0, i.e., all the conditions are satisfied.

6.3 Translation

In the next translational step, we obtain the configuration $\mathbf{w}$ that satisfies equation (2) and inequalities (5), (7), and (19), where (19) (discussed in Section 7.2) is a sufficient condition for (9), (10), and (11).

Since the rotations of the surface polygons are fixed, i.e., $\theta(i,j)$ is constant, equation (2) is a linear equation. The constraints for the entire model can be represented as,

$$\mathbf{C}_w\mathbf{w} = \mathbf{b}, \tag{17}$$

where $\mathbf{C}_w$ is a $2N_{\text{vert}} \times N_{\text{edge}}$ matrix whose $2m$-th and $(2m-1)$-th rows correspond to equation (1), for the m-th vertex. Equation (17) also forms an underdetermined system having $N_{\text{edge}} - 2N_{\text{vert}} = \frac{1}{3}N_{\text{edge}} - 2$ degrees of freedom. The solution space is calculated as,

$$\mathbf{w} = \mathbf{C}_w^{+}\mathbf{b} + (\mathbf{I}_{N_{\text{edge}}} - \mathbf{C}_w^{+}\mathbf{C}_w)\mathbf{w}_0, \tag{18}$$

where $\mathbf{C}_w^{+} = \mathbf{C}_w^{\mathrm{T}}(\mathbf{C}_w\mathbf{C}_w^{\mathrm{T}})^{-1}$ is the pseudo-inverse of $\mathbf{C}_w$, and $\mathbf{w}_0$ is an arbitrary value.

Similarly to the rotational phase, we solve the inequalities under the abovementioned constraints by iteratively updating $\mathbf{w}_0$ and projecting it to the constrained space. The projection is performed in $O(N_{\text{edge}}^2)$ using the precalculated LU factorization of matrix $\mathbf{C}_w\mathbf{C}_w^{\mathrm{T}}$.

$\mathbf{w}_0$ is updated according to the gradient of the penalty function calculated from inequalities (5), (7), and (19). We set the penalty function as a linear combination of the sum of squares of the errors and the external work from the user input. The penalty function is expressed as,

$$E_w = c_0 E_{(5,7)} + c_1 E_{(19)} + c_2 \left|\Delta\mathbf{w}\right|^2,$$

where $\Delta\mathbf{w}$ is a change in widths provided by the user input, and $E_{(5,7)}$ and $E_{(19)}$ represent the sum of squares of the errors calculated from the inequalities (5) and (7), and (19), respectively. c_0, c_1, or c_2 can attain any constant positive value. To eliminate scale dependency, $c_0 : c_1 : c_2 = 1 : \ell_{\text{ave}}^2 : 1$ is chosen, where ℓ_{ave} is the average length of the edges of the polyhedron.

The combination of iterative update in the configuration and projection to the constraint space enables a system that allows a user to edit the crease pattern interactively while satisfying the conditions (2), (5), (7), and (19).

6.4 Existence of the Solution

As linear inequalities (5) and (7) are dominant in the translational step, we did not experience any problem in the convergence to the global minimum. However, the global minimum is not always the valid solution. In general, the penalty function in the translational phase does not ensure the existence of the solution.

Here, we focus on the dominant conditions (5) and (7) for the existence of the solution; they are represented as $\mathbf{w}_n \geq 0$, for any n. There exists a solution that satisfies this condition, if there exists any solution of the homogeneous equation $\mathbf{C_w w} = \mathbf{0}$ such that $\mathbf{w}_n > 0$ for any n. The homogeneous solution corresponds to the graph derived by the molecule

mesh with infinitely small surface polygons. If any of the elements of $\mathbf{w}$ is not positive, the graph is crossed or reversed.

6.5 Remapping

Our mapping algorithm discards the degrees of freedom when fixing the rotations of the polygons, which is the main cause of the non-existence of the solution. If there exists no solution in the translational step, the initial angles must be modified by recalculating the rotation in order to reduce the penalty $E_{(5,7)}$. The initial angles are modified according to the following steps (Figure 15): (i) A valid graph free of intersections is determined by ignoring the constraints. Such a graph is obtained by a robust parameterization method such as barycentric embedding [34]. (ii) Then, the increase $\Delta\boldsymbol{\Theta}$ in the angle between the adjacent edges of the graph is measured. (iii) The increase in the angles of the edge-tucking molecules $\Delta\boldsymbol{\theta}$ is determined from $\Delta\boldsymbol{\Theta}$ by solving equation (3), in a least square sense. (iv) Finally, we recalculate equation (16) using these modified angles as the initial configuration $\boldsymbol{\theta}_0$. The overall process is performed iteratively until the system achieves a valid configuration. This iteration may not converge to a solution, in which case the surface must be topologically modified (normally, by adding cuts) by user interaction (Section 8.1). Algorithm 1 describes the mapping and remapping processes.

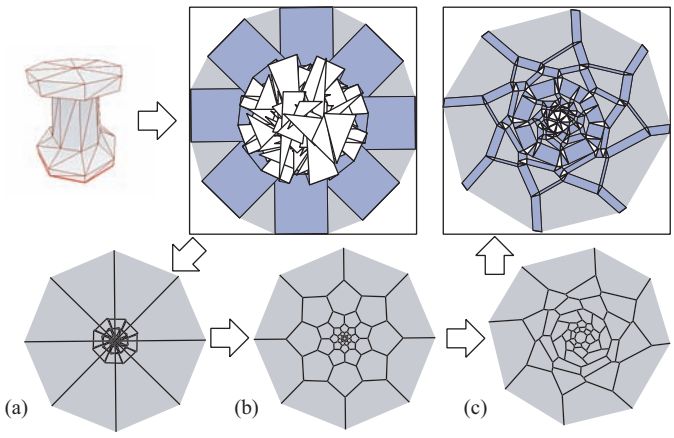


Fig. 15. Example of remapping. (a) Homogeneous solution. (b) Barycentric embedding. (c) Modified valid graph.

Algorithm 1 Mapping: Calculate $\boldsymbol{\theta}$ and $\mathbf{w}$ from a preprocessed mesh M

update $\mathbf{C}_\theta$ and $\mathbf{g}$ from M
LU factorize $\mathbf{C}_\theta\mathbf{C}_\theta^{\mathrm{T}}$ (precalculation of $\mathbf{C}_\theta^{+}$)
$\boldsymbol{\theta} \leftarrow \mathbf{0}$
loop
  $\boldsymbol{\theta} \leftarrow \mathbf{C}_\theta^{+}\mathbf{g} + \left(\mathbf{I}_{N_{\text{edge}}} - \mathbf{C}_\theta^{+}\mathbf{C}_\theta\right)\boldsymbol{\theta}$
  while conditions (4) (6) (8) NOT satisfied **do**
    calculate E_θ and ∇E_θ from conditions (4) (6) (8)
    $\nabla E_\theta \leftarrow (\mathbf{I}_{N_{\text{edge}}} - \mathbf{C}_\theta^{+}\mathbf{C}_\theta)\nabla E_\theta$
    calculate step size α by line search
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\nabla E_\theta$
  end while
  update $\mathbf{C}_w$ and $\mathbf{b}$ from $\boldsymbol{\theta}$
  LU factorize $\mathbf{C}_w\mathbf{C}_w^{\mathrm{T}}$ (precalculation of $\mathbf{C}_w^{+}$)
  $\mathbf{w} \leftarrow \mathbf{C}_w^{+}\mathbf{b}$
  repeat
    if conditions (5) (7) (19) satisfied **then**
      end
    else
      calculate E_w and ∇E_w from conditions (5) (7) (19)
      $\nabla E_w \leftarrow (\mathbf{I}_{N_{\text{edge}}} - \mathbf{C}_w^{+}\mathbf{C}_w)\nabla E_w$
      calculate step size α by line search
      $\mathbf{w} \leftarrow \mathbf{w} - \alpha\nabla E_w$
    end if
  until $|\alpha\nabla E_w|$ is sufficiently small
  while conditions (5) NOT satisfied **do**
    calculate E_w and ∇E_w from conditions (5)
    $\nabla E_w \leftarrow (\mathbf{I}_{N_{\text{edge}}} - \mathbf{C}_w^{+}\mathbf{C}_w)\nabla E_w$
    calculate step size α by line search
    $\mathbf{w} \leftarrow \mathbf{w} - \alpha\nabla E_w$
  end while
  generate barycentric graph using the convex boundary.
  calculate $\Delta\boldsymbol{\Theta}_{\text{target}}$ from the graph
  set $\Delta\boldsymbol{\theta}$ to minimize $\left|\Delta\boldsymbol{\Theta}_{\text{target}} - \dfrac{\partial\boldsymbol{\Theta}}{\partial\boldsymbol{\theta}}\Delta\boldsymbol{\theta}\right|^2$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end loop

7 Edge Splitting

7.1 Splitting

We introduce the concept of *edge splitting* for solving conditions represented by inequalities (9), (10), and (11). An edge-tucking molecule can split into two edge-tucking molecules with one digon between them without changing the original polygons of the surface. This technique is performed locally, i.e., without changing the location of the other part of the molecule mesh (Figure 16). The digon is inserted to divide the sector angle between the pair of edges of the original edge-tucking molecule. The width of the derived edge-tucking molecule with angle θ_{div} is expressed as

$$w_{\text{div}} = \frac{\sin(\theta_{\text{div}}/2)}{\sin(\theta/2)}w,$$

where θ and w denote the angle and width of the original edge-tucking molecule, respectively. Therefore, edge splitting is performed to subdivide edge-tucking molecules into sufficiently narrow edge-tucking molecules (Figure 16(c)), which satisfy the crease pattern intersection (9) and width conditions (11).

7.2 Relaxed Conditions

For each edge-tucking molecule, we obtain a pair of arcs on which the end points of an added digon is located. Assume that we obtain the configuration of the original molecule mesh such that the arcs do not intersect, and each arc is finitely

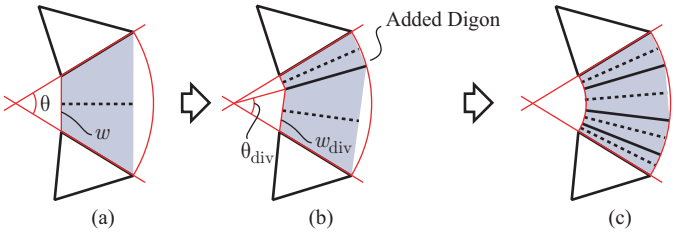


Fig. 16. Edge splitting.

separated from the other arcs, excluding any pair of adjacent arcs sharing a vertex (Figure 17(a)).

The edge-tucking molecules can be subdivided such that any corner vertex of the original vertex-tucking molecule belongs to a single Delaunay triangle. Assume that conditions (13) and (14) are satisfied; then, the shape of the corner Delaunay triangle can be flexibly changed such that $\tau_{\text{div}}(i, j_0)$ and $\tau_{\text{div}}(i, j_1)$ satisfy condition (10), by controlling $w_{\text{div}}(i, j_0)$ and $w_{\text{div}}(i, j_1)$ (Figure 17(b)). Any of other Delaunay triangles is constructed by connecting two adjacent vertices on an arc and a point on another arc. In the case of such a triangle, $\phi_{\text{div}}(i, j)$ can be set to a sufficiently small value by subdividing the edge-tucking molecules. Therefore, the tuck angle condition (10) can be satisfied if and only if the tuck proxy satisfies conditions (13) and (14).

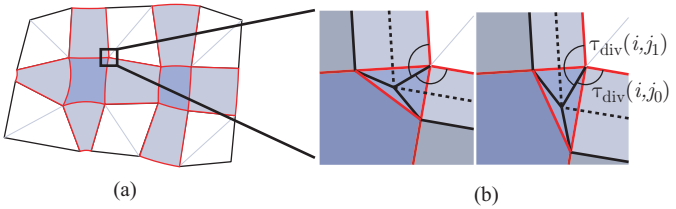


Fig. 17. (a) Configuration of the molecule mesh in which no arcs intersect. (b) Delaunay triangle at the corner can be transformed by controlling the width of the edge-tucking molecules.

Therefore, edge splitting relaxes conditions (9), (10), and (11), and replaces them with a sufficient condition that every arc does not intersect. The following sufficient condition is used.

$$\min(\beta_0(i,j), \beta_1(i,j)) \geq -\frac{1}{2}\theta(i,j), \tag{19}$$

where $\beta_0(i,j)$ and $\beta_1(i,j)$ denote the base angles of the Delaunay triangle whose base segment is located between $\text{VTM}(i)$ and $\text{ETM}(i,j)$. Condition (19) forces each arc to be contained in a single Delaunay triangle.

8 System

The proposed method is implemented as an interactive three-dimensional origami design system (Figure 1 Top). The input figure is given as a polygon mesh (Figure 18 (a)). First, the system constructs cuts to obtain the polygonal schema, which can be further modified by a user (Figure 18 (b)). Then, the system maps the polygons of the input polyhedral surface based on the mapping algorithm and generates the crease pattern (Figure 18 (c)). The user can manipulate the configuration and the crease pattern through a standard pointing device (e.g. mouse). Since the system both updates and displays the crease pattern and the resulting shape of the folded tuck, a user can interactively design three-dimensional origami models using this system.

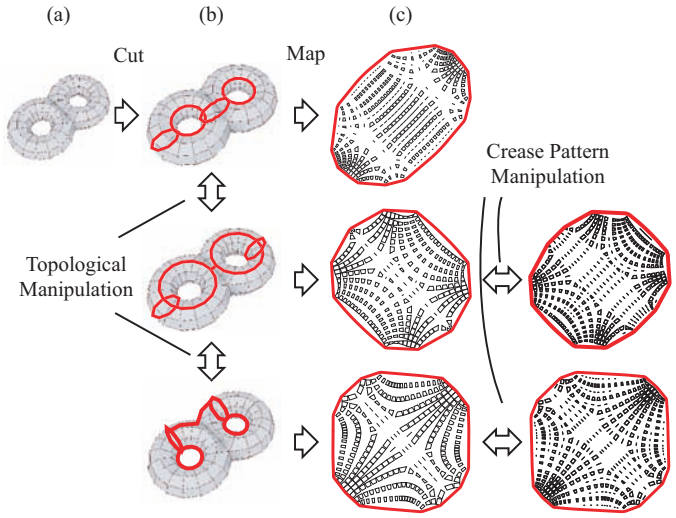


Fig. 18. The system for designing three-dimensional origami models. Changing cuts yields different visible seams and crease patterns for the same genus 2 polyhedron.

8.1 Topological Manipulation

The input is given as an orientable polygon mesh of arbitrary genus, which is cut to a polygonal schema ready to be mapped. As the cut, i.e., the boundary, is visible as a separated seam on the folded model, the system provides valid cuts controllable according to the preference of the user, in the following manner.

First, the system constructs a tree shape from a given mesh by breadth-first flooding. The boundary is then simplified by removing branches and jags. The user can specify the root of the tree, which will be located approximately in the middle of the paper.

Then, the boundary can be further modified. The modification is performed through the following two types of tools: (i) a tool to add (or delete) a selected edge into the existing boundary by connecting it via the shortest path and (ii) a tool to move the boundary by altering the connectivity around selected facets (Figure 19). A combination of these tools enables the user to freely modify the boundary while maintaining the surface homeomorphic to a disk throughout the manipulation.

8.2 Crease Pattern Manipulation

While the system automatically solves the inequalities, a user can modify the crease pattern in order to improve the aesthetics

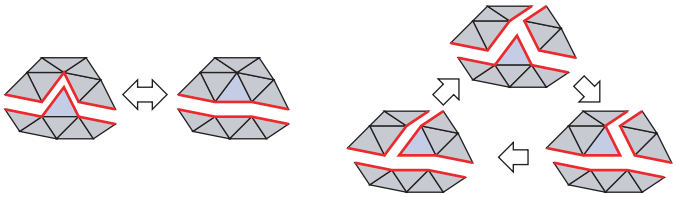


Fig. 19. Relocation of the perimeter by altering the connectivity around a facet.

of the model, including the shape of the folded tuck and the crease pattern itself. In order to modify the crease pattern, the system provides a tool to pick and drag mapped polygons and a tool to increase or decrease the widths of specified edge-tucking molecules. This tool allows a user to modify the crease pattern without changing the representing three-dimensional polyhedral surface.

The user input is converted into penalty forces that are applied to edge-tucking molecules when the translational configuration is being updated (Section 6.3). Iterative calculations are performed at an interactive frame rate for models with less than 700 edge-tucking molecules, examples of which are shown in Section 9.3.

The system facilitates the automatic splitting of edge-tucking molecules, in order to solve the inequalities. A user can also obtain the desired crease pattern by manually specifying the molecules to be split or merged. After splitting, the LU factorization for solving (18) is recalculated, which allows further manipulation of the crease pattern.

The modification of the crease pattern alters the shape of the folded tuck behind the surface, which is visible when folding a surface with a boundary (Figure 20). The folded tuck is visualized and continuously updated to follow the modified layout; this allows a user to understand the result while designing.

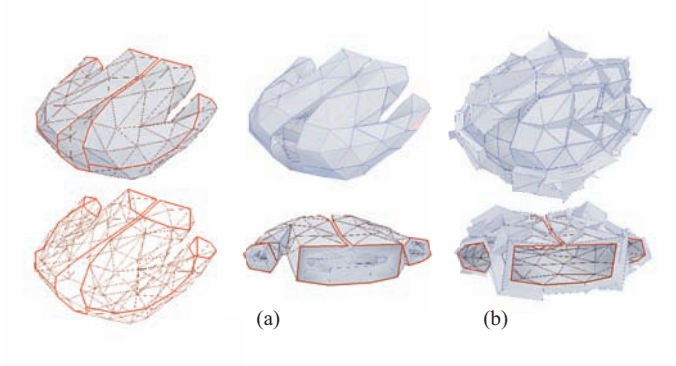


Fig. 20. Orientable surface with a boundary. (a) Tuck is folded inside. (b) Same mesh with opposite orientation. The tuck is folded outside.

9 Results

9.1 Examples

The proposed method was applied to six polyhedral surfaces. The generated crease patterns were realized to form real paper models, each of which was folded from one piece of paper, as shown in Figure 22. In models (a)-(d), the resulting crease patterns were automatically determined from the original topological disk meshes. Since model (e) was homeomorphic to a sphere, the software automatically determined the cut path on the surface. The cut path was then symmetrically oriented through a user interface for aesthetic reason. Thus far, the bunny model (f) is the most complicated models tested and folded. Although the original mesh was homeomorphic to a disk, additional user interaction for editing cut path was necessary, because the automatically generated crease pattern was highly inefficient (Figure 24). The paths behind the ears were chosen intuitively according to an empirical strategy in designing origami to place long flaps onto the perimeter of the paper. In these 6 models, the layouts were determined by the two-step approach, requiring no remapping process.

9.2 Results of Folding

The author has folded three-dimensional origami models from actual sheets of paper using the crease patterns generated by the design system (Figure 22). The original shape of the paper was a convex polygon. First, the generated crease pattern data is sent to a cutting plotter, which scores a sheet of paper along the crease pattern with its blade. Then, the paper is precreased and folded into a three-dimensional model, without the need for extra folds for shaping. The approximate time needed to fold a sheet of paper into the desired three-dimensional shape is shown in Figure 23.

The scoring by a cutting plotter helps the folder to fold precisely along the crease pattern. A laser cutter is also suitable for this purpose. In the given examples, the paper is scored from one side for aesthetic reason; however, scoring the paper from both sides can help to fold it easily.

Visible seams on a folded model lie on the lines of the three-dimensional mesh, which has a certain aesthetic quality, especially when backlit (Figure 21). The similar quality is pursued by the existing origami technique known as “origami tessellations” [35], [36], which expresses the geometric beauty of synthetic and natural patterns through repetitive pattern on a plane. Our results extend this type of expression to a three-dimensional one.

The origami models were practically realizable because of the following three reasons: thickness (Section 9.2.1), structural stiffness (Section 9.2.2), and efficiency (Section 9.2.3).

9.2.1 Thickness

From the viewpoint of thickness, the feasibility in folding is examined by considering the number of layers of paper folded at the same fold. In the proposed method, this type of folding occurs only when the folded tuck composed of 2 layers of paper is crimp-folded.

The bunny model shown in Figure 22(f) is folded from a sheet of convex paper that is approximately 500 mm in

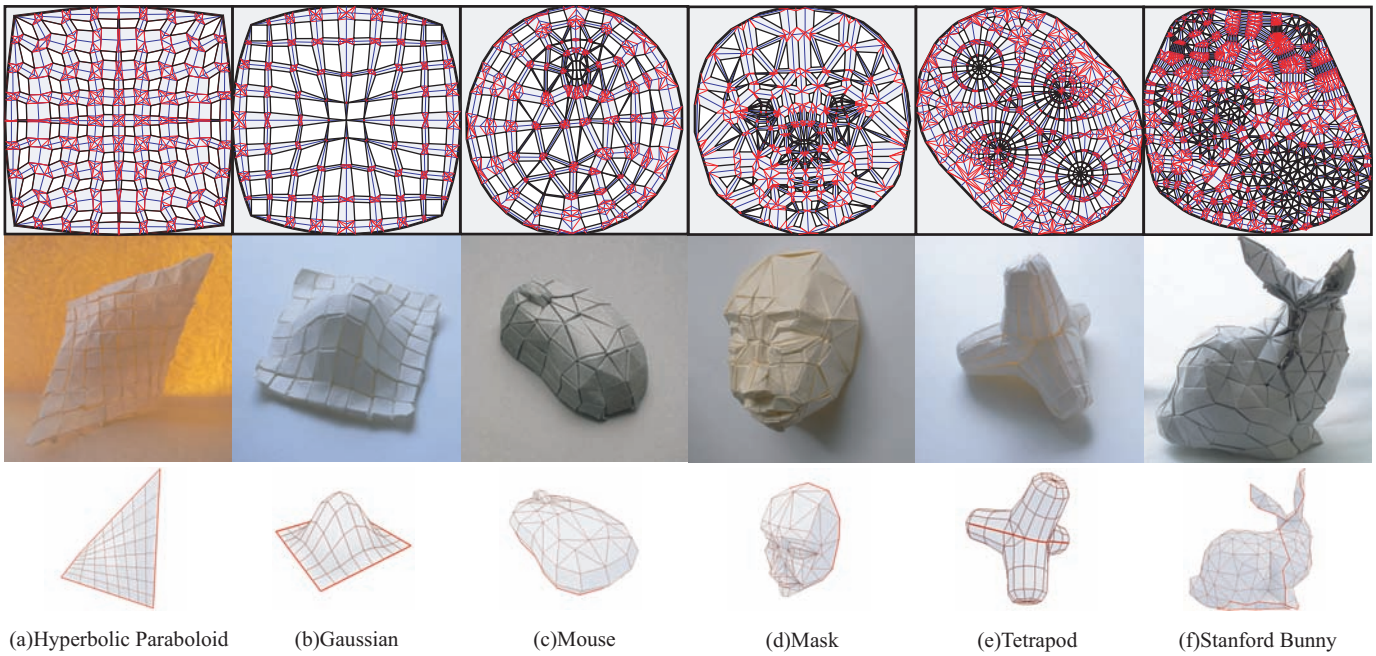


Fig. 22. Origami models designed using the system. Top row shows the crease patterns. Middle row shows the actual models folded from a sheet of paper (each).

Fig. 21. Backlit origami models.

diameter and $116\ \mathrm{g/m^2}$ in thickness. This paper is thicker than that used for folding some conventional super complex origami models, which require paper with a thickness of $19\ \mathrm{g/m^2}$.

9.2.2 Structural Stiffness

The resulting folded shape holds its own shape, because the crimp-folded tucking molecules prevent each vertex to split apart. The stiffness of the shape can be measured in terms of the stability of the target polyhedral surface homeomorphic to a disk.

A truss model is used to briefly examine the stability of the surface polyhedron. The surface is triangulated and all the edges of the polyhedron are regarded as rigid bars, which result in an unstable structure. In general, the number of degrees of freedom or degrees of instability of this structure is given by,

$$\textit{Degrees of Instability} = 3v - e - 6 = e_0 - 3,$$

where v, e, and e_0 are the number of vertices, edges, and edges on the boundary of the triangulated surface, respectively. The calculated degrees of instability of the folded models in this study and the estimated value of those in previous study [27] are compared, as shown in Figure 23. The degrees of instability in our study are substantially lower, which indicates a higher stiffness of the surface.

9.2.3 Efficiency

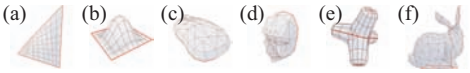
Although the example models were folded from a convex piece of paper, they could also be folded from a square piece of paper because the boundary of the crease pattern is approximately circular in shape, which can be efficiently packed in a square.

The efficiency of a crease pattern can be defined as the ratio of the surface area of the input model to that of the square paper to be folded. The efficiency of the crease patterns of the example models is within the range of 0.1 to 0.5, which is considerably better than previous studies (Figure 23). For example, if a Gaussian model, as shown in Figure 23(b), with efficiency of 0.415 is to be folded by the method described in [27], the most optimistic estimation of the efficiency is 0.0017 as the method requires a strip narrower than $1:1100$.

9.3 Time

Figure 23 shows the amount of time required for performing calculations on a laptop PC with a Pentium M 1.60 GHz using SSE2 instructions through ATLAS. Precalculation was completed in a few seconds, and the iterative calculation was performed at an interactive frame rate. The time required to converge varied with models, and it was within 20 s for the models used as examples in this study. The longer convergence time was attributed to the use of the gradient descent method, which can be replaced with a faster method. Nevertheless, this was a fairly reasonable amount of time over which we could

repeatedly modify the boundary and perform remapping until we obtain a desired crease pattern, before actually folding the model for several hours.



	(a)	(b)	(c)	(d)	(e)	(f)
# Facets: Quadrilateral	64	64	40	32	144	0
Triangles	0	0	62	154	72	374
# Boundary Edges	32	32	16	14	20	50
# Edge-Tucking Molecules	208	208	213	317	446	606
Deg. Instability	29	29	13	11	17	47
Deg. Instability in method[8]	382	382	548	960	1222	1868
Efficiency	0.180	0.415	0.272	0.265	0.143	0.172
Folding Time (approx.) [h]	2	2	3	6	4	10
Precalculation Time [s]	0.047	0.047	0.16	0.16	0.40	1.2
Iteration Frame Rate [fps]	128	128	93.0	51.1	34.2	16.5
Convergence Time [s]	0.156	0.094	3.81	1.91	4.03	15.0

Fig. 23. Results of the system

10 Drawbacks and Future Works

It should be noted that the proposed algorithm does not always guarantee a solution, as stated in Section 6.5. Some models fail to be validly mapped even after the iteration of remapping or result in highly inefficient crease patterns. These failure and inefficiency occur mainly because a solution is dependent on the topology of the surface and the mesh.

A problem of this kind is often solved by appropriately modifying the boundary on the surface using tools provided by the system (Figure 24). In general, adding boundary edges increases the degrees of freedom and enables a valid solution. It is further observed that placing the boundary onto acute points on the surface increases the efficiency of the crease pattern; this is a similar strategy used in conventional origami designs. However, the process still relied on intuitive trial and error by an experienced origami designer. We intend to develop a method for automatically finding an appropriate cutlines from a given mesh.

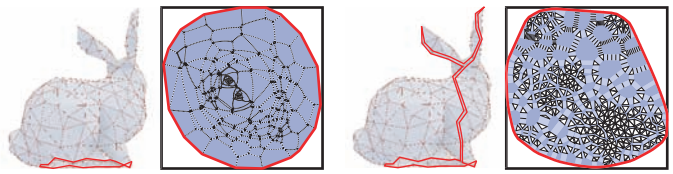


Fig. 24. Left: Bunny without modification of the boundary results in the generation of a crease pattern with a low efficiency (0.007). Right: Efficiency of the bunny with modified boundary behind the ears is 0.172.

When applying our technique to develop art and industrial designs, the objective is often in building an approximation of a smooth surface. Suppose the desired shape is given as a dense triangular mesh, it must be sufficiently simplified for later calculation and folding by a tessellation method such as triangular remeshing [37], [38], [39], [40], [41], quandrangulation [42], [7], and segmentation with developable patches [43], [8].

The tessellation itself can be a variable that is optimized within the origamization procedure. In order to improve robustness of the system by changing the tessellation, we intend to characterize a “nice” mesh that ensures a valid solution and improves the efficiency. An empirical observation indicates that this characteristic is not simply the regularity of triangles and connection, as opposed to a naive expectation.

As mentioned in the introduction, our method can potentially be used in a wide variety of applications since it provides a solution for the general and critical problem of achieving an arbitrary three-dimensional shape from a single sheet of material. However, it is necessary to further improve our method in order to apply it to widespread industrial purposes. This is because our method produces a number of crimp-folds, thus locking other folds; such crimp-folds prevent a material from transforming smoothly into a desired state without stretching, and they may potentially allow thick or brittle materials to rupture at the vertices.

One of the improvements to our method that is worth considering is the optimization of the configuration of the mesh and the tuck proxy such that all the crimp-folds disappear. This would significantly simplify the manufacturing of a desired surface and it would be more suitable for on-demand or mass production and for on-site construction. This will also lead to further studies on transformable or deployable structures that can tightly cover freeform surfaces; such structures could potentially be used for the designs of prefabricated architectures for use in severe environments such as in space, medical devices used in minimally invasive surgery, and several types of watertight containers that can be compactly packaged.

11 Conclusion

This paper presented the first practical method for obtaining a crease pattern for folding a piece of paper into a given polyhedral surface, without cutting the paper. The algorithm is implemented as an interactive origami design system, which enabled the generation of crease patterns from several computer-generated three-dimensional models with less than 400 polygons, in a short time span. The generated crease patterns were folded in practice to produce highly complex three-dimensional origami models, for the first time.

The method sometimes fails to satisfy conditions or generates highly inefficient crease patterns. In future, we intend to develop a more robust algorithm to avoid such failure and to improve the efficiency of the generated crease pattern.

Acknowledgement

The development of the design system was supported by IPA (Information-technology Promotion Agency, Japan). This research is also partially supported by a Grant-in-Aid for JSPS (Japan Society for the Promotion of Science) Fellows. The original bunny model is courtesy of the Stanford 3D Scanning Repository.

REFERENCES

[1] E. D. Demaine and J. O'Rourke, *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, July 2007.

[2] J. Mitani and H. Suzuki, "Making papercraft toys from meshes using strip-based approximate unfolding," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 259–263, 2004. Proc. SIGGRAPH.

[3] F. Massarwi, C. Gotsman, and G. Elber, "Papercraft models using generalized cylinders," in *PG '07: Proc. the 15th Pacific Conference on Computer Graphics and Applications*, pp. 148–157, 2007.

[4] D. Julius, V. Kraevoy, and A. Sheffer, "D-charts: Quasi-developable mesh segmentation," in *Proc. Eurographics*, pp. 581–590, 2005.

[5] C. C. L. Wang, "Towards flattenable mesh surfaces," *Computuer-Aided Design*, vol. 40, no. 1, pp. 109–122, 2008.

[6] J. Subag and G. Elber, "Piecewise developable surface approximation of general NURBS surfaces with global error bounds," pp. 143–156, 2006.

[7] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, "Geometric modeling with conical meshes and developable surfaces," *ACM Trans. Graphics*, vol. 25, no. 3, pp. 681–689, 2006.

[8] H. Pottmann, A. Schiftner, P. Bo, H. Schmiedhofer, W. Wang, N. Baldassini, and J. Wallner, "Freeform surfaces from single curved panels," *ACM Trans. Graphics*, vol. 27, no. 3, 2008. Proc. SIGGRAPH.

[9] M. Wertheim, "Cones, curves, shells, towers: He made paper jump to life," *The New York Times*, june 22 2004. http://www.theiff.org/press/NYThuffman.html.

[10] R. D. Resch and H. Christiansen, "The design and analysis of kinematic folded plate systems," in *Proc. the Symposium for Folded Plate Structures*, International Association for Shell Structures, 1970.

[11] R. D. Resch, "Ron resch dot com." http://www.ronresch.com/.

[12] J. Lobell, "The milgo experiment: an interview with haresh lalvani," *Architectural Design*, vol. 76, no. 4, pp. 52–61, 2006.

[13] RoboFold, "Robofold." http://www.robofold.com/design.html.

[14] D. Huffman, "Curvature and creases: a primer on paper," *IEEE Trans.Computers*, vol. C-25, no. 10, pp. 1010–1019, 1976.

[15] Y. Kergosien, H. Gotoda, and T. Kunii, "Bending and creasing virtual paper," *IEEE Computer Graphics and Applications*, vol. 14, no. 1, pp. 40–48, 1994.

[16] M. Kilian, S. Flöry, N. J. Mitra, and H. Pottmann, "Curved folding," *ACM Trans. Graphics*, vol. 27, no. 3, 2008. Proc. SIGGRAPH.

[17] S.-M. Belcastro and T. Hull, "A mathematical model for non-flat origami," in *Origami3: Proc. the 3rd International Meeting of Origami Mathematics, Science, and Education*, pp. 39–51, 2002.

[18] T. Tachi, "Rigid origami simulator," 2007. http://www.tsg.ne.jp/TT/software/.

[19] R. Burgoon, Z. J. Wood, and E. Grinspun, "Discrete Shells Origami," in *Proc. Computers And Their Applications*, pp. 180–187.

[20] M. Bern and B. Hayes, "The complexity of flat origami," in *Proc. the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 175–183, 1996.

[21] T. Meguro, "The method to design origami," *Origami Tanteidan Newspaper*, 1991.

[22] E. D. Demaine, M. L. Demaine, and A. Lubiw, "Folding and cutting paper," in *Revised Papers from the Japan Conference on Discrete and Computational Geometry (JCDCG'98)*, pp. 104–117, 1998.

[23] R. J. Lang, "A computational algorithm for origami design," in *SCG '96: Proc. the twelfth annual symposium on Computational geometry*, pp. 98–105, 1996.

[24] M. Bern, E. Demaine, D. Eppstein, and B. Hayes, "A disk-packing algorithm for an origami magic trick," in *Origami3: Proc. the 3rd International Meeting of Origami Mathematics, Science, and Education*, pp. 17–28, 2002.

[25] M. Bern and B. Hayes, "Origami embedding of piecewise-linear two-manifolds.," in *LATIN*, vol. 4957 of *Lecture Notes in Computer Science*, pp. 617–629, 2008.

[26] R. J. Lang, *Tree Maker 4.0 : A Program for Origami Design*, 1998. http://www.langorigami.com/science/treemaker/TreeMkr40.pdf.

[27] E. D. Demaine, M. L. Demaine, and J. S. B. Mitchell, "Folding flat silhouettes and wrapping polyhedral packages: New results in computational origami," *Computational Geometry: Theory and Applications*, vol. 16, no. 1, pp. 3–21, 2000.

[28] M. Tanaka, "Possibility and constructive proof through origami," *Hyogo University Journal*, no. 11, pp. 75–82, 2006.

[29] T. Tachi, "3D origami design based on tucking molecule," in *Origami^4: Proceedings of The Fourth International Conference on Origami in Science, Mathematics, and Education*, 2009. to appear.

[30] Tama Software, "Pepakura designer," 2004. http://www.tamasoft.co.jp/pepakura-en/.

[31] T. K. Dey, "A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection," in *Proc. ACM SoCG '94*, pp. 277–284, 1994.

[32] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust, "Computing a canonical polygonal schema of an orientable triangulated surface," in *Proc. ACM SoCG '01*, pp. 80–89, 2001.

[33] X. Gu, S. J. Gortler, and H. Hoppe, "Geoemtry images," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 355–361, 2002.

[34] W. Tutte, "How to draw a graph," in *Proc. the London Mathematical Society*, pp. 743–768, 1963.

[35] A. Bateman, "Computer tools and algorithms for origami tessellation design," in *Origami3: Proc. the 3rd International Meeting of Origami Mathematics, Science, and Education*, p. 121, 2002.

[36] E. Gjerde, *Origami Tessellations: Awe-Inspiring Geometric Designs*. AK Peters, 2008.

[37] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Proc. ACM SIGGRAPH '93*, pp. 19–26, 1993.

[38] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," in *Proc. ACM SIGGRAPH '95*, pp. 173–182, 1995.

[39] A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schroder, "Multiresolution mesh morphing," in *Proc. ACM SIGGRAPH 99*, pp. 343–350, 1999.

[40] P. Alliez, M. Meyer, and M. Desbrun, "Interactive geometry remeshing," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 347–354, 2002.

[41] G. Peyré and L. Cohen, "Geodesic remeshing using front propagation," in *Proc. Variational and Level Set Methods in Computer Vision 2003*, pp. 33–40, 2003.

[42] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart, "Spectral surface quadrangulation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1057–1066, 2006.

[43] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 905–914, 2004.