Computational Peeling Art Design

HAO LIU*, University of Science and Technology of China, China
XIAO-TENG ZHANG*, University of Science and Technology of China, China
XIAO-MING FU†, University of Science and Technology of China, China
ZHI-CHAO DONG, University of Science and Technology of China, China
LIGANG LIU†, University of Science and Technology of China, China

(a) A parrot standing on a branch **(b)** Lizard **(c)** Fern

Fig. 1. Peeling artworks generated by our design system. Crafting along the drawn curves on the citruses (top left), the citruses are unfolded into a parrot standing on a branch (a), a lizard (b), or a fern (c).

Some artists peel citrus fruits into a variety of elegant 2D shapes, depicting animals, plants, and cartoons. It is a creative art form, called *Citrus Peeling Art*. This art form follows the conservation principle, i.e., each shape must be created using one entire peel. Central to this art is finding optimal cut lines so that the citruses can be cut and unfolded into the desired shapes. However, it is extremely difficult for users to imagine and generate cuts for their desired shapes. To this end, we present a computational method for citrus peeling art designs. Our key insight is that instead of solving the difficult cut generation problem, we map a designed input shape onto a citrus in an attempt to cover the entire citrus and use the mapped boundary to generate the cut paths. Sometimes, a mapped shape is unable to completely cover a citrus. Consequently, we have developed five customized ways of interaction that are used to rectify the input shape so that it is suitable for citrus peeling art. The mapping process and user interactions are iteratively conducted to satisfy a user's design intentions. A large number of experiments, including a formative user study, demonstrate the capability and practicability of our method for peeling art design and construction.

CCS Concepts: • **Computing methodologies → Shape modeling**.

Additional Key Words and Phrases: peeling art, mesh cutting, deformation, parameterizations, mapping

ACM Reference Format:
Hao Liu, Xiao-Teng Zhang, Xiao-Ming Fu, Zhi-Chao Dong, and Ligang Liu. 2019. Computational Peeling Art Design. *ACM Trans. Graph.* 38, 4, Article 64 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323000

1 INTRODUCTION

Computational art assists users to easily visualize, design, and create generative art in algorithmic and programmatic ways [Bickel et al. 2018; Wood et al. 2016]. *Citrus Peeling Art*, created by Yoshihiro Okada [Okada 2010], allows users to peel a citrus fruit along guidance lines and unfold the whole citrus into desired 2D shapes, as shown in Fig. 2. [1] This fascinating art form offers an opportunity for enjoyment and creativity and has been widely used in children's education.

However, only a limited number of pieces have been created so far [Okada 2010]. To create new samples of this art form, a designer needs a fertile imagination and a large number of trial-and-error experiments, which is non-trivial and time consuming. Therefore, it is highly challenging for general users to peel citruses into customized shapes.

*Joint first authors
†The corresponding authors

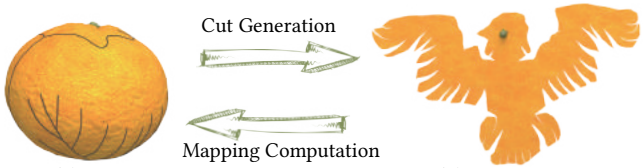
Authors' addresses: Hao Liu, University of Science and Technology of China, China, pllh@mail.ustc.edu.cn; Xiao-Teng Zhang, University of Science and Technology of China, China, zxt2015@mail.ustc.edu.cn; Xiao-Ming Fu, University of Science and Technology of China, China, fuxm@ustc.edu.cn; Zhi-Chao Dong, University of Science and Technology of China, China, dzc206@mail.ustc.edu.cn; Ligang Liu, University of Science and Technology of China, China, lgliu@ustc.edu.cn.

0730-0301/2019/7-ART64 $15.00
https://doi.org/10.1145/3306346.3323000

[1] More examples can be found at https://youtube.com/user/yosigirito/videos.

Crafting Mantis Horse Frog

Fig. 2. Some real examples of citrus peeling art designed by Yoshihiro Okada [Okada 2010].



(a) Citrus **(b)** 2D shape

Fig. 3. The mapping computation and the cut generation are two reciprocal processes. It is very difficult to construct cut paths so that the unfolded shape of the cut citrus is similar to the input. But, it is easier to map the input 2D shape onto the citrus to try to completely cover it.

Our goal is to develop an algorithmic approach to assist home users to create cut paths and peeling works for their desired input shapes. From the point of view of geometry processing, this problem seeks to find a way to generate cut paths on a citrus fruit in order to unfold it into a 2D shape that is close to a user-designed input.

Existing research on computing cut paths on surfaces aims to minimize cut length and unfolded shape distortion [Li et al. 2018; Poranne et al. 2017; Sharp and Crane 2018]; however, our proposed problem is much more difficult. The reasons are twofold. First, it is difficult to obtain a computable metric that can be efficiently used in the optimization to measure the closeness or similarity between the unfolded shape and the desired shape. Second, there may be no possible cut paths to make the unfolded shape adequately match the given input shape. Users should be able to modify an input shape to find a suitable solution. Nevertheless, it is non-trivial to create an easy-to-use design system to freely allow users to modify input shapes and generate desired results.

In this paper, we present a novel method for citrus peeling art design. Instead of directly computing the cut paths to satisfy the shape similarity requirements, we study the problem from a different perspective (Fig. 3). In fact, if we map an unfolded shape back onto a citrus, the citrus can be entirely covered and the boundary of the mapped shape represents the cut paths. Thus, we map the input shape onto the citrus and try to completely cover it. Then, the cuts are determined based on the boundary of the mapped input shape. The *key insight* of our method is to convert the cut generation problem to a mapping problem that is easier to solve. Therefore, we propose a novel method to map an input shape onto a citrus with low isometric distortion, which ensures high similarity between the unfolded shape and the input shape.

However, only using this low isometric distortion mapping is not always successful for every input shape. To modify the input shapes suitable for citrus peeling art, we integrate users into the design process. Based on an analysis of the principles of citrus peeling art, five customized interaction methods are provided, including *shape augmentation*, *angle augmentation*, *part deletion*, *curvature reduction*, and *pre-processing*. Then, the mapping process and interactions are alternatively performed to meet a user's design intentions. Fig. 4 shows an example.

To the best of our knowledge, our method is the first to study the cut generation problem, wherein the unfolding contains low isometric distortion and the unfolded shape is similar to the desired input shape. The provided computational tool for citrus peeling art design and construction is able to create elegant shapes. We demonstrate the feasibility and practicability of our technique through a number of experiments (60 examples) and a formative user study (10 participants).

2 RELATED WORK

Mesh cutting. Previous mesh cutting methods usually focus on the low distortion requirement and the short cut path requirement. Some methods first detect vertices that may introduce high distortion and then connect the detected vertices using a minimum spanning tree to determine the cut [Chai et al. 2018; Sheffer 2002; Sheffer and Hart 2002]. An iterative algorithm is developed, which alternately unfolds the mesh and finds the shortest path from the vertex with maximum distortion to the existing boundary [Gu et al. 2002]. Mesh segmentation approaches [Julius et al. 2005; Lévy et al. 2002; Sander et al. 2002, 2003; Zhang et al. 2005; Zhou et al. 2004] divide an input mesh into multiple patches, and each patch can be unfolded with very low distortion. Cut length and unfolded distortion are simultaneously and jointly optimized [Li et al. 2018; Poranne et al. 2017]. A global variational approach is presented to generate the cut [Sharp and Crane 2018]. These methods cannot generate an unfolded shape that is similar to an input shape.

Low distortion mappings. A lot of methods for low distortion mappings have been proposed (cf. the surveys in [Botsch and Sorkine 2008; Floater and Hormann 2005; Li and Iyengar 2015; Sheffer et al. 2006]). Usually, the task of computing low distortion mappings is formulated as an optimization problem that tries to minimize distortion energy functions with some specified constraints. To preserve the isometry of a mapping, the isometric distortion energy is selected for optimization. There are many isometric distortion metrics, such as the ARAP energy [Liu et al. 2008], the isometric AMIPS energy [Fu et al. 2015], and the symmetric Dirichlet energy [Smith and Schaefer 2015]. To optimize these energies, many geometric optimization algorithms have been proposed, such as local/global methods [Liu et al. 2008; Sorkine and Alexa 2007], bounded distortion methods [Kovalsky et al. 2015; Lipman 2012], representation-based methods [Fu and Liu 2016; Sheffer and de Sturler 2001], coarse-to-fine methods [Hu et al. 2018; Praun and Hoppe 2003], and maintenance-based methods [Liu et al. 2018; Rabinovich et al. 2017]. The BFF method [Sawhney and Crane 2017] provides direct control over boundary length or angle of the unfolded shape, but it does not study the cut generation problem. In our mapping process, we combine the ARAP energy with a novel metric to slightly modify input shapes and effectively shrink the part of a citrus that is excluded from the mapped region.

Foldable art design. Various computational systems have been proposed recently for art design (cf. the surveys in [Bickel et al. 2018]). Here, we only review prior works about foldable art design and fabrication. Papercraft design consists of cutting, bending, folding,

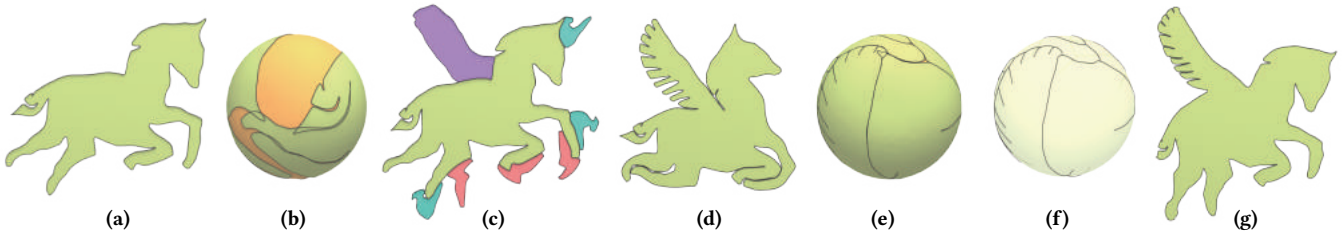


Fig. 4. Workflow of our method. (a) An input 2D shape $\mathcal{S}$. (b) Mapping the input shape $\mathcal{S}$ onto the citrus $\mathcal{M}$ (the whole sphere-like shape). Since the input shape is not suitable for peeling art, the mapped shape $\mathcal{S}^m$ (green parts) cannot completely cover the citrus $\mathcal{M}$. (c) Unfolding the mapped shape $\mathcal{S}^m$ and the uncovered regions (orange parts in (b)) of a citrus onto a plane for interactions. The purple, red, and cyan regions are for interactions. (d) After interactions, a new 2D shape is generated. The mapping process and interactions are iteratively conducted until the area of the uncovered regions is almost zero or the user does not want to interact anymore. (e) The mapped shape completely covers the citrus. (f) Using the boundary of the mapped shape to generate the resulting cut C. (g) Cutting the citrus to a disk topology mesh that is unfolded onto the plane.

and gluing surface patches [Kilian et al. 2008, 2017; Massarwi et al. 2007; Mitani and Suzuki 2004b; Shatz et al. 2006; Skouras et al. 2014; Takezawa et al. 2016]. These methods unfold an input 3D mesh into a planar domain via introducing cuts; however, the folded shape is not constrained so it dose not match the user's desired shape. Some automatic tools have been developed for origami, which is a traditional papercrafting art [Tachi 2009, 2010]. Tools for designing paper pop-up illustrations and wire meshes are also presented [Garg et al. 2014; Jr. et al. 2014; Le et al. 2014; Li et al. 2011, 2010; Mitani and Suzuki 2004a]. In this paper, we compute cut paths on a citrus so that it can be unfolded into a user's expected shape.

3 MAPPING TO SURFACE

Input. We denote the input 2D simply-connected shape as $\mathcal{S}$ and the citrus shape as a genus-zero surface $\mathcal{M}$. $\mathcal{S}$ is represented as a planar triangular mesh that consists of a set of vertices $\mathcal{V} = \{\mathbf{v}_i\}$ and triangles $\mathcal{F} = \{\mathbf{f}_i\}$.

3.1 Problem and challenge

Problem and formulation. Our goal is to compute a cut path C on $\mathcal{M}$ and cut $\mathcal{M}$ to a disk topology mesh $\mathcal{M}^c$ so that the unfolded shape $\mathcal{S}^c$ of $\mathcal{M}^c$ looks similar to $\mathcal{S}$. The process is formulated as follows:

$$\min_{C} \; E_{\text{sim}}(\mathcal{S}, \mathcal{S}^c) = E_{\text{sim}}\left(\mathcal{S}, \Psi(\mathcal{M}^c)\right), \tag{1}$$

where E_{sim} indicates a similarity error between two 2D shapes, and Ψ is the unfolding operation. Since we are unfolding citruses, Ψ can be treated as a determined parameterization, e.g., [Liu et al. 2008].

Challenges. Solving (1) is very challenging. It is non-trivial to choose a E_{sim} that is easy to optimize. The definitions of E_{sim} can be classified into two categories according to whether the correspondence between $\mathcal{S}$ and $\mathcal{S}^c$ is achieved. If the correspondence is given, the distortion of the correspondence or the summation of squares for the Euclidean distance between corresponding vertices can be used to define E_{sim}; however, computing the correspondence is difficult and costly. If the correspondence is not built, we can define E_{sim} as the Hausdorff distance or the difference between descriptors of the two shapes, but it is hard to optimize E_{sim} with respect to C.

3.2 Our method

Key idea. In fact, if we map $\mathcal{S}^c$ back onto $\mathcal{M}$ with low isometric distortion, $\mathcal{M}$ can be completely covered. Then, the boundary of the mapped $\mathcal{S}^c$ is similar to the cut C. According to this fact, we can map $\mathcal{S}$ onto $\mathcal{M}$ and try to completely cover $\mathcal{M}$. If $\mathcal{M}$ is entirely covered, the boundary of the mapped $\mathcal{S}$ approximates the desired cut C. This is our *key idea*: converting the cut generation problem to a mapping problem that is easier to solve.

Goals and formulation. We denote the mapping from $\mathcal{S}$ to its mapped shape $\mathcal{S}^m$ on $\mathcal{M}$ as Φ, i.e., $\mathcal{S}^m = \Phi(\mathcal{S})$. The set of vertices and triangles of $\mathcal{S}^m$ are denoted as $\mathcal{V}^m = \{\mathbf{v}_i^m\}$ and $\mathcal{F}^m = \{\mathbf{f}_i^m\}$, respectively. If $\mathcal{S}$ is suitable for generating citrus peeling art, the final unfolded shape can be approximated as $(\Psi \circ \Phi)(\mathcal{S})$. We use the isometric distortion of the mapping $(\Psi \circ \Phi)$ as the similarity metric. Since Ψ is determined, we only use the isometric distortion of Φ as the similarity metric. Thus, our goal is to compute $\mathcal{S}^m$ so that it contains low isometric distortion from $\mathcal{S}$ and completely covers $\mathcal{M}$ with no overlaps. The procedure is formulated as follows:

$$\min_{\mathcal{S}^m} \; E_{\text{iso}}(\mathcal{S}^m, \mathcal{S}) + \omega E_{\text{shr}}(\mathcal{R}), \tag{2}$$

where $E_{\text{iso}}(\mathcal{S}^m, \mathcal{S})$ indicates the isometric distortion from $\mathcal{S}$ to $\mathcal{S}^m$. $\mathcal{R}$ (see the orange region in Fig. 4 (b)) denotes a region that shares the same boundary with $\mathcal{S}^m$ and includes $\mathcal{M} \setminus \mathcal{S}^m$. $E_{\text{shr}}(\mathcal{R})$ is an energy that is used to shrink $\mathcal{R}$ and resolve the overlaps of $\mathcal{S}^m$, that is, to try to make $\mathcal{S}^m$ completely covers $\mathcal{M}$. ω is a positive weight balancing the two terms.

Isometric distortion energy. We use the ARAP distortion metric [Liu et al. 2008] to measure $E_{\text{iso}}(\mathcal{S}^m, \mathcal{S})$. By introducing a local coordinate system $\mathbf{F}_i^m$ on $\mathbf{f}_i^m$, the mapping from $\mathbf{f}_i$ to $\mathbf{f}_i^m$ is an affine transformation with a Jacobian matrix of 2×2, denoted as J_i.

$$E_{\text{iso}}(\mathcal{S}^m, \mathcal{S}) = \sum_{\mathbf{f}_i \in \mathcal{F}} \text{Area}(\mathbf{f}_i) \|J_i - R_i\|_F^2, \tag{3}$$

where $\|\cdot\|_F^2$ is the Frobenius norm, and R_i is an auxiliary variable representing the closest projection of J_i onto the 2D rotation group.

Shrink energy. We triangulate $\mathcal{R}$ to obtain a triangular mesh that has a set of vertices $\mathcal{V}^r = \{\mathbf{v}_i^r\}$ and triangles $\mathcal{F}^r = \{\mathbf{f}_i^r\}$. Each $\mathbf{f}_i^r$ is equipped with a local coordinate system $\mathbf{F}_i^r$. Our goal is to degenerate each $\mathbf{f}_i^r$. One naive choice is to use the area of $\mathbf{f}_i^r$ as an additional energy term for optimizing the mapping. However, it

works badly (see the comparison in Fig. 10). Assuming that there is a 2D regular triangle $\mathbf{t}$ with an edge length of 1, the mapping from $\mathbf{t}$ to $\mathbf{f}_i^r$ (in $\mathbf{F}_i^r$) is an affine transformation with a 2×2 Jacobian matrix, denoted as A_i. Then, the zero area requirement of $\mathbf{f}_i^r$ is transformed to require the rank of A_i to be one. Based on this concept, we propose a novel rank-one energy:

$$E_{\text{shr}}(\mathcal{R}) = \sum_{\mathbf{f}_i^r \in \mathcal{F}^r} \text{Area}(\mathbf{t}) \|A_i - B_i\|_F^2, \tag{4}$$

where B_i represents an auxiliary variable for the closest projection of A_i onto a rank-one 2×2 matrix space.

Reformulation. We denote the boundary of $\mathcal{R}$ as $\partial\mathcal{R}$ and the boundary of $\mathcal{S}^m$ as $\partial\mathcal{S}^m$. $\partial\mathcal{R}$ and $\partial\mathcal{S}^m$ are always the same. The mapping problem can be reformulated as the following problem:

$$\begin{aligned} \min_{\mathcal{S}^m, \mathcal{R}} \quad & E_{\text{iso}}(\mathcal{S}^m, \mathcal{S}) + \omega E_{\text{shr}}(\mathcal{R}) \\ \text{s.t.} \quad & \mathbf{v}_i^m \in \mathcal{M}, \quad \forall \mathbf{v}_i^m \in \mathcal{V}^m, \\ & \mathbf{v}_j^r \in \mathcal{M}, \quad \forall \mathbf{v}_j^r \in \mathcal{V}^r, \\ & \partial\mathcal{R} = \partial\mathcal{S}^m. \end{aligned} \tag{5}$$

Solver. We collect $\mathcal{V}^m$ and $\mathcal{V}^r$ into one set, which is denotd as $\widehat{\mathcal{V}} = \{\widehat{\mathbf{v}}_i\}$. Similarly, $\widehat{\mathcal{F}} := \{\widehat{\mathbf{f}}_i\}$. The local coordinate system for $\widehat{\mathbf{f}}_i$ is denoted as $\mathbf{F}_i$. Since the form of $E_{\text{shr}}(\mathcal{R})$ is similar to $E_{\text{iso}}(\mathcal{S}^m, \mathcal{S})$, we adopt the local-global solver [Liu et al. 2008] to solve (5).

- *Local step.* Assuming $\widehat{\mathbf{v}}_k$ are fixed, local coordinate systems, $\mathbf{F}_i$ and R_i (or B_i) are computed per triangle. The computation method of R_i is the same as [Liu et al. 2008]. We denote $A_i = U_i S_i V_i^T$ as the singular value decomposition of A_i, in which $S_i = \text{diag}(\sigma_i, \tau_i)$ is a diagonal matrix with singular values on the diagonal. Then, when A_i is given, we compute $B_i = U_i \text{diag}(\sigma_i, 0) V_i^T$. By choosing this type of B_i, we are actually minimizing τ_i^2.
- *Global step.* Assuming R_i, B_i, $\mathbf{F}_i$ are fixed, we first update $\widehat{\mathbf{v}}_k$ and then project them onto $\mathcal{M}$.

Different from [Liu et al. 2008], our variables are 3D vertices $\widehat{\mathbf{v}}_k$. So, $\mathbf{F}_i$ also changes in the global step. It is difficult to solve the global step, so we design an approximation method.

First, $\mathbf{F}_i$ is assumed to be fixed in the global step. Second, the movement of $\widehat{\mathbf{v}}_k$ is locally approximated as a 2D vector. We denote the local frame at $\widehat{\mathbf{v}}_k$ as $\mathbf{F}_k^v$ and set the third axis of $\mathbf{F}_k^v$ as the normal at $\widehat{\mathbf{v}}_k$. Then, the movement $\delta\widehat{\mathbf{v}}_k$ of $\widehat{\mathbf{v}}_k$ is a 3D vector in $\mathbf{F}_k^v$. We restrict $\delta\widehat{\mathbf{v}}_k$ in the tangent plane of $\widehat{\mathbf{v}}_k$, indicating that its third component is 0. For one triangle $\widehat{\mathbf{f}}_k^i$ within the one-ring neighbors of $\widehat{\mathbf{v}}_k$, its local frame is denoted as $\mathbf{F}_k^i$, and the movement of $\widehat{\mathbf{v}}_k$ in $\mathbf{F}_k^i$ is $\delta\widehat{\mathbf{v}}_k^i = (\mathbf{F}_k^i)^T \mathbf{F}_k^v \delta\widehat{\mathbf{v}}_k$. We only consider the first two components of $\delta\widehat{\mathbf{v}}_k^i$ for approximating the 2×2 matrices J_i and A_i. Then, the problem is still with a quadratic objective function and some linear equality constraints (i.e., $\partial\mathcal{R} = \partial\mathcal{S}^m$), so we solve its linear KKT system to obtain the movements $\delta\widehat{\mathbf{v}}_k$. Since the movements $\delta\widehat{\mathbf{v}}_k$ are usually small, this approximation works well in practice. In order to introduce as small distortion as possible, we set $\omega = 0.001$. Fig. 5 shows an example.

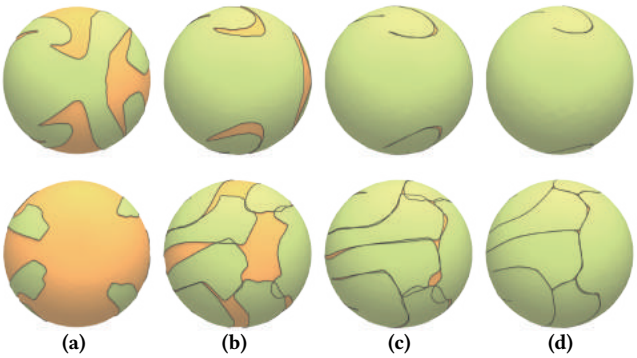


Fig. 5. Progressive results of our optimization on the Frog shape (Fig. 6 (a)). (a) Initialization. (b) After the 10th iteration. (c) After the 75th iteration. (d) Final result after 255 iterations. We show the front and back views of the citrus in the first and second rows, respectively.

Fig. 6. Initialization process. We first put the input shape into a square (a). Then, we map the square onto a regular octahedron (b). Finally, we map the octahedron onto $\mathcal{M}$ to initialize the mapped shape (c). In (c), the green and orange parts indicate the initial $\mathcal{S}^m$ and $\mathcal{R}$, respectively. Here, we scale $\mathcal{M}$ 2.5 times for visualization in (c).

Initialization. To initialize $\mathcal{S}^m$, one direct solution is to use stereographic projection; however, this usually produces large distortion, thereby requiring more iterations to solve (5). We use a regular octahedron (denoted as $\mathbf{O}$) as an intermediate domain (Fig. 6 (b)). We first map $\mathcal{S}$ onto $\mathbf{O}$ to produce $\widetilde{\mathcal{S}}$, and then map $\widetilde{\mathcal{S}}$ onto $\mathcal{M}$ to generate the initial $\mathcal{S}^m$ based on the one-to-one correspondence between $\mathbf{O}$ and $\mathcal{M}$. As we know, $\mathbf{O}$ can be unfolded into a square with small distortion (see Fig. 6 of [Praun and Hoppe 2003]). Thus, we put $\mathcal{S}$ in a square and center it, and then $\widetilde{\mathcal{S}}$ can be easily computed (Fig. 6). Since the initial $\mathcal{S}^m$ has no overlaps, the initial $\mathcal{R}$ is just $\mathcal{M} \setminus \mathcal{S}^m$. In our experiments, the side length of the square is set as $\max\{\alpha w_{\mathcal{S}}, \alpha h_{\mathcal{S}}\}$, wherein $w_{\mathcal{S}}$ and $h_{\mathcal{S}}$ are the width and height of the bounding box of $\mathcal{S}$. To produce free spaces to update $\mathcal{S}^m$, we set $\alpha = 1.05$ in our experiments. If $\mathcal{S}$ can be mapped to completely cover $\mathcal{M}$, then their areas are equal. Based on this consideration, we initialize the scale of $\mathcal{M}$.

Termination. Our algorithm terminates when the relative error of the objective function value is less than 10^{-4} or $\text{Area}(\mathcal{R}) < \epsilon_a$. In our experiments, we set $\epsilon_a = 10^{-2}\text{Area}(\mathcal{M})$, and we do not observe any non-convergent examples.

Dynamic remeshing. During the optimization process, the mesh quality of $\mathcal{R}$ often worsens, which affects the convergence ratio. Inspired by [Jiang et al. 2017], we dynamically remesh $\mathcal{R}$ using [Botsch and Kobbelt 2004]. After performing three global steps, we conduct a remeshing step. This remeshing is necessary, as shown in Fig. 7.

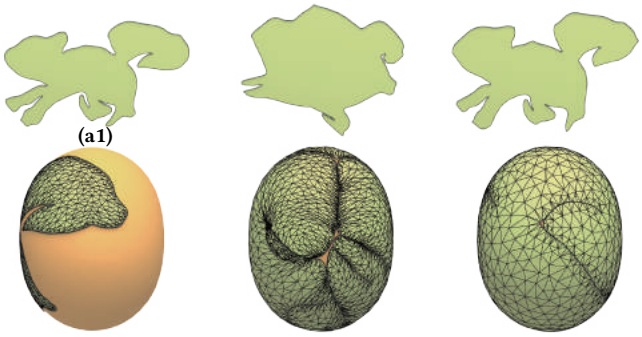
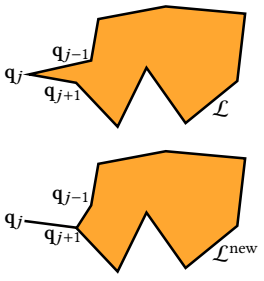


Fig. 7. Dynamic remeshing. A 2D input shape (a1) is initially mapped onto a citrus (a2). In the last two columns, the second row shows the resulting $\mathcal{S}^m$, and the unfolded shapes are in the first row.

3.3 Cut generation

If the input shape $\mathcal{S}$ is suitable for citrus peeling art, the area of $\mathcal{R}$ is very small after solving (5). We use $\partial\mathcal{R}$ to compute the cut C.

Collapse. Our construction of C is based on an operation called *collapse* (see right inset). The collapse operation is defined on a cycle $\mathcal{L}$ that contains $m > 3$ vertices $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$. The subscripts of the vertices are defined in counterclockwise order, and they are expanded periodically. That is to say, $\mathbf{q}_j = \mathbf{q}_{j+km}, \forall k \in Z$. The collapse operation is defined at a vertex $\mathbf{q}_j$, and we select one edge $\overline{\mathbf{q}_j\mathbf{q}_i}$ (e.g., $i = j+1$ in the right inset) adjacent to $\mathbf{q}_j$ for the operation. After performing a collapse operation on $\mathcal{L}$ at vertex $\mathbf{q}_j$ along the edge $\overline{\mathbf{q}_j\mathbf{q}_i}$, we get a new cycle $\mathcal{L}^{\text{new}}$ and the edge $\overline{\mathbf{q}_j\mathbf{q}_i}$ is added into the cut. We recursively apply the collapse operation on the cycle until the new cycle contains three vertices. For these three vertices that represent a triangle, we select two shorter edges that are added into the cut.



Given a cycle $\mathcal{L}$, one must find the vertex that is applied to the collapse operation. In fact, if the interior angle at $\mathbf{q}_j$ is equal to zero, this indicates that the shorter one of its adjacent edges belongs to C. So each time we select the vertex with the smallest interior angle to perform a collapse operation along its shorter adjacent edge.

Details. In our experiments, the initial cycle is $\partial\mathcal{R}$. The interior angle may be negative due to overlaps, so we use the absolute value of the interior angle as the criterion. Since C consists of the partial boundary edges of $\partial\mathcal{R}$, it may not be smooth. In our experiments, we apply five iterations of Laplacian smoothing to the vertices of C while fixing the vertices, whose valences do not equal to 2. We show an example in Fig. 4 (f).

3.4 Implementation details and discussions

Representations of $\mathcal{M}$. $\mathcal{M}$ looks like an oblate sphere, so we approximate it as a surface of revolution with center at the origin by default. One of its points (x, y, z) can be represented as follows:

$$x = r(\phi)\cos(\theta)\cos(\phi),\ y = r(\phi)\sin(\theta)\cos(\phi),\ z = r(\phi)\sin(\phi), \quad (6)$$

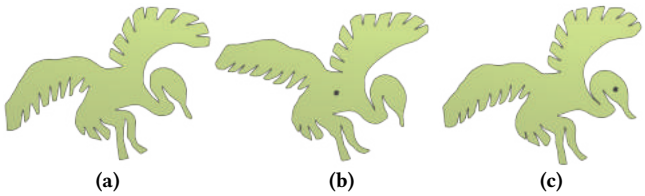


Fig. 8. Given an input shape (a), we compute the resulting 2D unfolded shapes with different stalk locations (see the black dots in (b) and (c)).

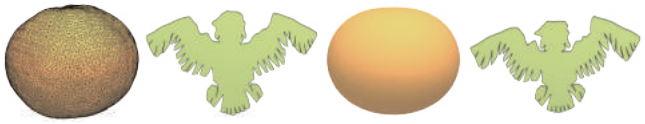


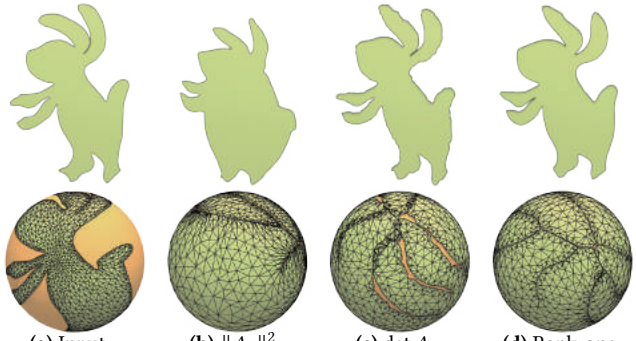
Fig. 9. Representations of $\mathcal{M}$. The input 2D shape is shown in Fig. 3 (b). For each row, the left image indicates $\mathcal{M}$ and the right image is the resulting 2D shape. Similar results are obtained for the two representations. The mapping process takes 30 minutes for (a) and 6 minutes for (b).

where $\theta \in (0, 2\pi]$, $\phi \in [-\pi/2, \pi/2]$, and $r(\phi) = \left((1-\rho)\cos^2(\phi) + \rho\right) r_0$. Here, r_0 is the radius of the equator, and ρ indicates the oblateness (0.8 in our experiments). The point $(0, 0, \rho r_0)$ is the position of the stalk of the citrus. Before running our algorithm, users select a point on $\mathcal{S}$ that will be mapped to the stalk location. During the mapping process, the updated vertices of $\mathcal{S}^m$ are projected back onto $\mathcal{M}$ using radial projection. In Fig. 8, two different points on $\mathcal{S}$ are selected as stalk positions, and the resulting 2D shapes are almost same.

However, if one wants to accurately represent $\mathcal{M}$, a real citrus ready for peeling can be scanned by a high precision scanner and reconstructed as a triangular mesh. In our mapping process, we revise the initial scale of $\mathcal{M}$ so that the scaled $\mathcal{M}$ contains the same area as $\mathcal{S}$. For the updated vertices, we use closest point projection to project them onto $\mathcal{M}$. Fig. 9 shows a comparison between two different representations. The final results are similar. Although our default representation, i.e., a surface of revolution, has some approximation errors, the elasticity of citrus peel can compensate for these errors. In addition, scanning and reconstructing each real citrus is expensive and not practical for general users. During the mapping computation process, closest point projection is more expensive than radial projection (see the time comparison in Fig. 9). Thus, representation as a surface of revolution makes sense.

Area shrink energies. There are some other energies that can be used to shrink $\mathcal{R}$, e.g., the determinant $\det A_i$ or the squared Frobenius norm $\|A_i\|_F^2$. Note that $\det A_i$ also represents the ratios between the areas of $\mathbf{f}_i^r$ and $\mathbf{t}$. In Fig. 10, we show a comparison. Our rank-one energy works better than others, because it produces a smaller distortion mapping. The $\det A_i$ energy reaches the minimum when one of the two singular values of A_i is zero; thus, the solver often oscillates back and forth near the two minimum points. It does not converge in the example shown in Fig. 10 (c), which causes serrated boundary. The $\|A_i\|_F^2$ energy requires the three edges for $\mathbf{f}_i^r$ to be zero, so $\partial\mathcal{R}$ also tries to shrink. Since $\partial\mathcal{R}$ and $\partial\mathcal{S}^m$ are always the same, the shrunk $\partial\mathcal{R}$ leads to a severely distorted result (Fig. 10 (b)).

Median axis of $\mathcal{R}$. During the cut generation process, we can also use the median axis of $\mathcal{R}$ as the final cut. Robustly computing a



(a) Input **(b)** $\|A_i\|_F^2$ **(c)** $\det A_i$ **(d)** Rank-one

Fig. 10. Different energies for shrinking $\mathcal{R}$. Aside from the input column, we show the resulting 2D unfolded shapes and the mapping results $\mathcal{S}^m$ in the first and second rows, respectively.

median axis curve on $\mathcal{S}$ may be costly. Since $\mathcal{R}$ has almost zero area, our generated cut adequately approximates the median axis. Furthermore, it is very fast and robust, and it works well in practice.

4 INTERACTIONS

Iterative design process. If an input shape is suitable for peeling art, the mapping process can make Area($\mathcal{R}$) $< \epsilon_a$. In order to handle unsuitable input shapes, we integrate users into our computational pipeline. Then, the design process is iterative as follows:

(1) We solve (5) to generate $\mathcal{S}^m$ and $\mathcal{R}$. If Area($\mathcal{R}$) $< \epsilon_a$, we terminate the algorithm. Otherwise, go to Step (2).
(2) If users want to modify $\mathcal{S}$, interactions are performed to generate a new $\mathcal{S}$, and go to Step (1) after interacting. Otherwise, go to Step (3).
(3) The requirement Area($\mathcal{R}$) $< \epsilon_a$ is gradually satisfied by increasing the weight ω.

4.1 Interaction places

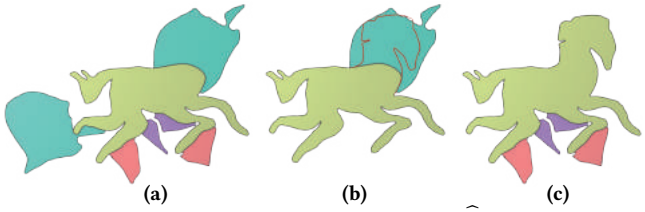
It is unintuitive and hard to interact on $\mathcal{M}$; however, the 2D plane is a good choice for interactions. Thus, we unfold $\mathcal{S}^m$ and $\mathcal{R}$ onto the plane for interactions after solving (5).

We unfold $\mathcal{S}^m$ using [Liu et al. 2008] to generate a 2D shape, which is denoted as $\widehat{\mathcal{S}}$ (see the green shape in Fig. 4 (c) or Fig. 11 (a)). Since our goal is to shrink $\mathcal{R}$ to zero area, large subregions of $\mathcal{R}$ provide useful hints for interactions. Thus, we discard the triangle of $\mathcal{R}$ if it contains two vertices that satisfy the following two conditions: (i) the two vertices are in $\partial\mathcal{R}$, and (ii) the geodesic distance between the two vertices along $\partial\mathcal{R}$ exceeds a threshold ϵ_d. Then, the left region of $\mathcal{R}$ is decomposed into $N_r \geq 1$ disk topology patches, denoted as $\{\mathbf{R}_i, i = 1, \ldots, N_r\}$. We compute the unfolded shapes of $\mathbf{R}_i$ as follows:

(1) We generate several segments as connected components of the intersections between the boundary of $\mathbf{R}_i$ and $\partial\mathcal{S}^m$.
(2) For each segment, we fix its unfolded position to the corresponding position in $\widehat{\mathcal{S}}$, and then unfold $\mathbf{R}_i$ using [Liu et al. 2008] while treating these fixed positions as hard constraints.
(3) After Step (2), $\mathbf{R}_i$ has several unfolded shapes. If $\mathbf{R}_i$ does not overlap with $\mathcal{S}^m$, we discard the unfolded shapes that overlap with $\widehat{\mathcal{S}}$. Otherwise, we reserve all unfolded shapes. If all of the unfolded shapes from $\mathbf{R}_i$ are discarded, we retain the unfolded shape with the least overlap area for interaction.

The regions with the same color, except the green one in Fig. 4 (c) or Fig. 11 (a), denote the unfolded shapes of one $\mathbf{R}_i$. Based on these unfolded shapes, we propose five interaction modes as follows. Note that among all of the unfolded shapes from $\mathbf{R}_i$, only one can be operated. We set $\epsilon_d = 3\bar{l}_b$, wherein $\bar{l}_b$ is the average of boundary edge length of $\mathcal{S}$.

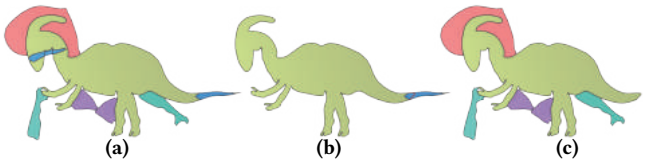
4.2 Interaction modes



(a) **(b)** **(c)**

Fig. 11. Shape augmentation. The green part denotes $\widehat{\mathcal{S}}$, and the other color regions are the interaction places. (a) The shape before interactions. (b) Users select the cyan part for augmentation and draw some red curves to design the desired shapes. (c) The shape after performing a shape augmentation.

Mode 1: shape augmentation. The user-designed shape $\mathcal{S}$ may be supplemented in order to be suitable for peeling art. Users specify the missing parts according to the computed unfolded shapes. As shown in Fig. 11 (a), the green region denotes $\widehat{\mathcal{S}}$, and the other regions indicate the unfolded shapes of all $\mathbf{R}_i$. During an interaction, users select an unfolded shape to augment $\mathcal{S}$. Furthermore, we allow users to draw curves that are represented as B-Splines on the selected unfolded regions in order to define their desired shapes (see the head region in Fig. 11 (b)).



(a) **(b)** **(c)**

Fig. 12. Part deletion. (a) The shape before interactions. (b) Users select the blue part for deletion and draw some red curves to design their target boundary. (c) The shape after performing a part deletion.

Mode 2: part deletion. Some parts of the input shape $\mathcal{S}$ may be redundant in citrus peeling art. We first identify the overlapping regions in $\mathcal{S}^m$ and then color them blue (Fig. 12 (a)). Users select which one to delete during the interaction and design their expected boundary curves (see the red curves in Fig. 12 (a)).

Mode 3: angle augmentation. The cut C is a tree. The interior angle at one boundary vertex of $\mathcal{M}^c$, which corresponds to the leaf node $\mathbf{n}$ in C, is approximately equal to $2\pi - \kappa(\mathbf{n})$, where $\kappa(\mathbf{n})$ indicates the angle defect of $\mathbf{n}$ on $\mathcal{M}$. Therefore, our interaction should follow this constraint.

During the interaction, users first select one unfolded shape, then pick a vertex on the junction line L between the selected unfolded shape and $\widehat{\mathcal{S}}$, Finally, they draw a curve across the selected unfolded shape starting from the chosen vertex (see the red curve

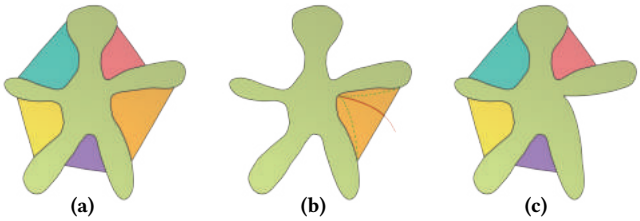


Fig. 13. Angle augmentation. (a) The shape before interactions. (b) Users select the orange part and draw the red curve. Our method automatically generates the dotted green curves. (c) The shape after performing augmentation on one angle.

in Fig. 13 (b)). After drawing the curve, our method automatically generates two new curves (see the dotted green curves in Fig. 13 (b)) that satisfy the following constraints: (1) the two curves start from the user-selected vertex and end at the ending vertices of L; (2) the angle between the tangents of the two curves at the user-selected vertex is equal to $2\pi - \kappa(\mathbf{v})$, where $\mathbf{v}$ is the vertex on $\mathcal{M}$ that corresponds to the user-selected vertex. These curves are represented as B-Splines,and the two requirements are formulated as linear constraints. We solve each new curve by optimizing an objective function, which measures the squares of the gradient difference between the user-drawn curve and the new curve, under the two linear constraints.

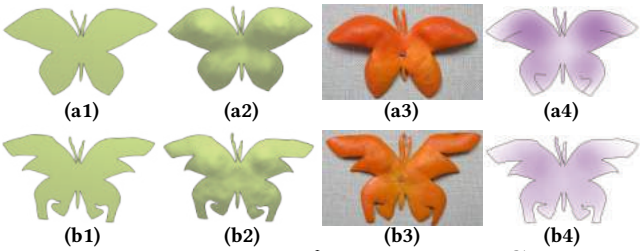


Fig. 14. Curvature reduction. (a1) $\widehat{\mathcal{S}}$. (a2) The computed $\widetilde{\mathcal{S}}$. (a3) The real peeled shape. (a4) Users draw black curves to design their desired cuts. The color encodes the z-value of $\widetilde{\mathcal{S}}$, with white being zero. With the introduced cuts, the new $\widehat{\mathcal{S}}$, $\widetilde{\mathcal{S}}$, the peeled shape, and the encoded z-value are shown in the second row. The curvature from (a2) to (b2) is significantly reduced.

Mode 4: curvature reduction. Usually, the distortion between $\widehat{\mathcal{S}}$ and $\mathcal{S}^m$ is not exactly zero. Although the elasticity of a citrus peel can compensate for a slight distortion, a large distortion may cause the real peeled shape to be curved, which may violate the user's design intention. Thus, we solve the following problem to recover a curved surface (denoted as $\widetilde{\mathcal{S}}$) from $\widehat{\mathcal{S}}$ for interaction:

$$\min_{\widetilde{\mathcal{S}}} \quad E_1(\widetilde{\mathcal{S}}, \mathcal{S}^m) + \beta E_2(\widetilde{\mathcal{S}}), \tag{7}$$

where we use the ARAP distortion of [Sorkine and Alexa 2007] to define $E_1(\widetilde{\mathcal{S}}, \mathcal{S}^m)$, and $E_2(\widetilde{\mathcal{S}})$ indicates the summation of squares of the z-value for each vertex of $\widetilde{\mathcal{S}}$. During the optimization, the z-value of the turning vertices, where the boundary doubles back on themselves, are fixed to zero. The local-global solver of [Sorkine and Alexa 2007] is used to solve (7) and $\widetilde{\mathcal{S}}$ is initialized as $\widehat{\mathcal{S}}$. We set $\beta = 0.001$. Fig. 14 shows an example. Since the Gaussian curvature of each vertex of $\mathcal{S}^m$ is positive, and our E_1 tries to maintain local rigidity, the Gaussian curvature of each vertex of $\widetilde{\mathcal{S}}$ tends to be positive. Besides, as some z-values are fixed to zero, the z-values of all vertices are usually greater than or equal to zero. As shown in Fig. 14, the real peeled shape is slightly different from $\widetilde{\mathcal{S}}$. Performing physical simulation can reduce this difference, but it is very difficult. Thus, we use E_2 to make a rough approximation to simulate the effects of gravity.

We provide an interaction mode, where some cuts are introduced to help flatten the curved surface $\widetilde{\mathcal{S}}$. For ease of operation, the interactions are still performed on the plane. So we show the z-value of $\widetilde{\mathcal{S}}$ using a color map (Fig. 14 (a4)), with white being zero. During the interaction, users draw curves from the outside to the inside of $\widetilde{\mathcal{S}}$ to define the introduced cuts (see the black curves in Fig. 14 (a4)).

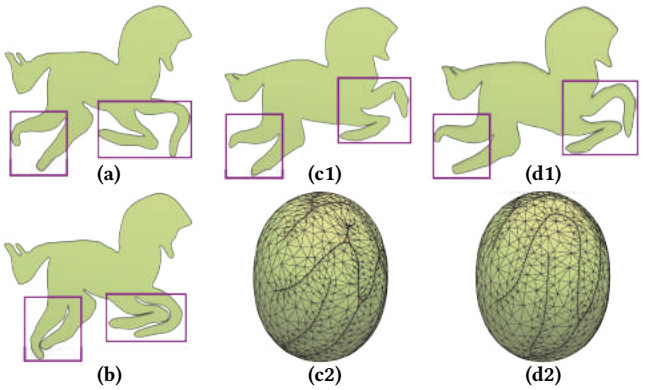


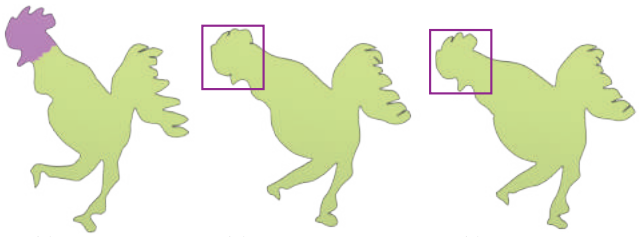
Fig. 15. Pre-alignment. Users want the horse's legs to be aligned. (a) The input shape. (b) The pre-aligned shape. In the last two columns, we show the unfolded shapes and the mapping results $\mathcal{S}^m$ in the first and second rows, respectively. (c1&c2) Using (a) to initialize $\mathcal{S}^m$. (d1&d2) Using (b) for initialization. Without pre-alignment, our mapping process automatically folds the legs. The pre-alignment process prevents the legs from folding during the mapping process.

Mode 5: Pre-processing. The pre-processing process consists of two components: pre-alignment and user-specified weights.

Since our mapping problem is non-linear and non-convex, the solver may be trapped by a local minimum, thereby causing misaligned results. We propose a tool for pre-aligning two parts of the boundary of $\mathcal{S}$ (Fig. 15). It is an ARAP handle-based deformation. Users can deform $\mathcal{S}$ to make pairs of two parts close to each other. Then, we use the deformed shape to initialize $\mathcal{S}^m$. During the optimization of (5), the rank-one energy shrinks $\mathcal{R}$, so these pre-aligned pairs get together. We show a comparison in Fig. 15. Note that the shape after the pre-alignment process is only used to initialize $\mathcal{S}^m$ but not treated as a new $\mathcal{S}$.

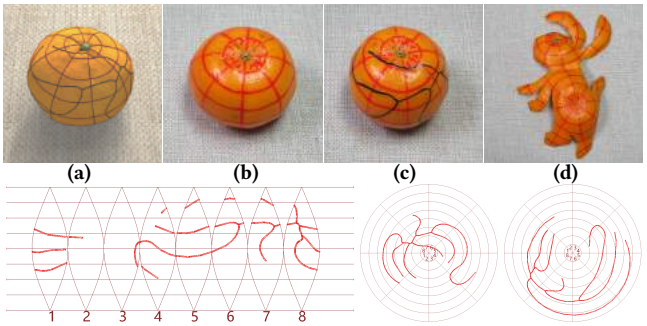
Some regions of $\mathcal{S}$ may be more important than other regions, so we allow users to specify importance weights on $\mathcal{S}$. The regions with greater weights exhibit less distortion. A comparison is shown in Fig. 16.

Adaptive scaling. Given a citrus $\mathcal{M}$ with fixed size, a small $\mathcal{S}$ may lead to no overlaps. Conversely, large overlaps may occur with a very large $\mathcal{S}$. Thus, we compute an appropriate scale for $\mathcal{S}$ to mitigate overlaps. Users input an acceptable threshold (denoted



(a) Input **(b)** Without weights **(c)** With weights

Fig. 16. User-specified weights. Users try to preserve the shape of the head. Users specify the regions with highly important weights (in purple) in the input shape. With weights, our algorithm preserves the beak and cockscomb well.



Cut along longitudes and expand Exponential maps from two poles

Fig. 17. Real peeling process. (a) Graticules (red curves) and cuts (black curves) on a virtual citrus. (b) Graticules (red curves) on a real citrus. (c) Drawn cuts (black curves) on a real citrus. (d) Final real peeling shape. The second row shows the printed guidance for drawing cuts on a real citrus.

as ϵ_o) as an overlap ratio (denoted as ρ_o), i.e., the ratio between the area of the overlaps and $\mathcal{S}$. We propose a practical solution to adjust the scale of $\mathcal{S}$ in the mapping process. The mapping process alternatively solves (5) and scales $\mathcal{S}$ until $\rho_o \leq \epsilon_o$. After solving (5), if $\rho_o > \epsilon_o$, $\mathcal{S}$ is scaled by 0.9 times, then we go back to solve (5). Zooming out $\mathcal{S}$ increases the distortion between $\mathcal{S}^m$ and $\mathcal{S}$. Then, the next optimization of (5) reduces the distortion, thereby making $\mathcal{S}^m$ smaller. Since $\mathcal{M}$ is fixed during the scaling, the smaller $\mathcal{S}^m$ may reduce overlaps. Note that due to overlaps, $\mathcal{R}$ is not exactly $\mathcal{M} \setminus \mathcal{S}^m$ during the mapping process.

5 REAL PEELING PROCESS

To peel a real citrus, we first draw the generated cuts on it, and then craft along the drawn curves with a knife. We use graticules as guidance (Fig. 17). Our painting process contains three steps:

(1) We generate a graticule on a digital citrus (Fig. 17 (a)). Then, the digital citrus is cut along equally spaced longitudes to produce eight pointed ellipses that are printed onto papers as guidance (Fig. 17 - bottom left). In addition, the exponential maps from the north and south poles are also printed to serve as auxiliaries (Fig. 17 - bottom right). During the process, the computed cut paths (red curves in Fig. 17 - bottom) are also printed.
(2) We draw graticules on a real citrus (Fig. 17 (b)).
(3) According to the correspondence between the drawn graticules and the graticules on the printed papers, and the relative relation between the cuts and the graticules on the printed papers, we draw the cuts on the real citrus (Fig. 17 (c)).

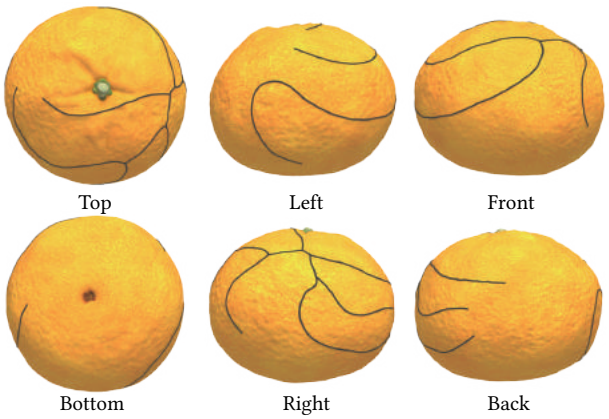


Fig. 18. Using the six primary views as real peeling guidance. Here ,we show the six primary views of the cuts for the Rabbit shape in Fig. 17.

Dove Eagle Shrimp

Fig. 19. Comparison to Yoshihiro Okada. The first row shows the results of Yoshihiro Okada, and ours are in the second row.

6 EVALUATIONS

Our method provides a computational tool for peeling art design and construction. We begin by evaluating the real peeling guidance, continue with a variety of elegant designs generated by authors and a formative user study to demonstrate the capability of our method, and end with a discussion of computational timings.

Performance. To analyze our design process quantitatively, we record four timings during the design process, including the cut generation time (t^{cut}), the graticule drawing time (t^{gra}), the cut path drawing time (t^{path}), and the crafting time (t^{craft}). In addition, the number of iterations in the design process and the interaction times for each interaction mode are recorded. Table 1 summarizes the statistics and timings.

6.1 Peeling guidance

To draw the generated cuts on a real citrus, other guidance can also be used. Here, we select six primary views, which are used in the book [Okada 2010], for comparison. Fig. 18 shows the six primary views of the computed cuts for the Rabbit shape in Fig. 17. We invited two users to peel this example using two types of guidance. One user was skilled and had peeled more than 20 shapes, while another was a novice. The skilled user finished the peeling, but produced a result

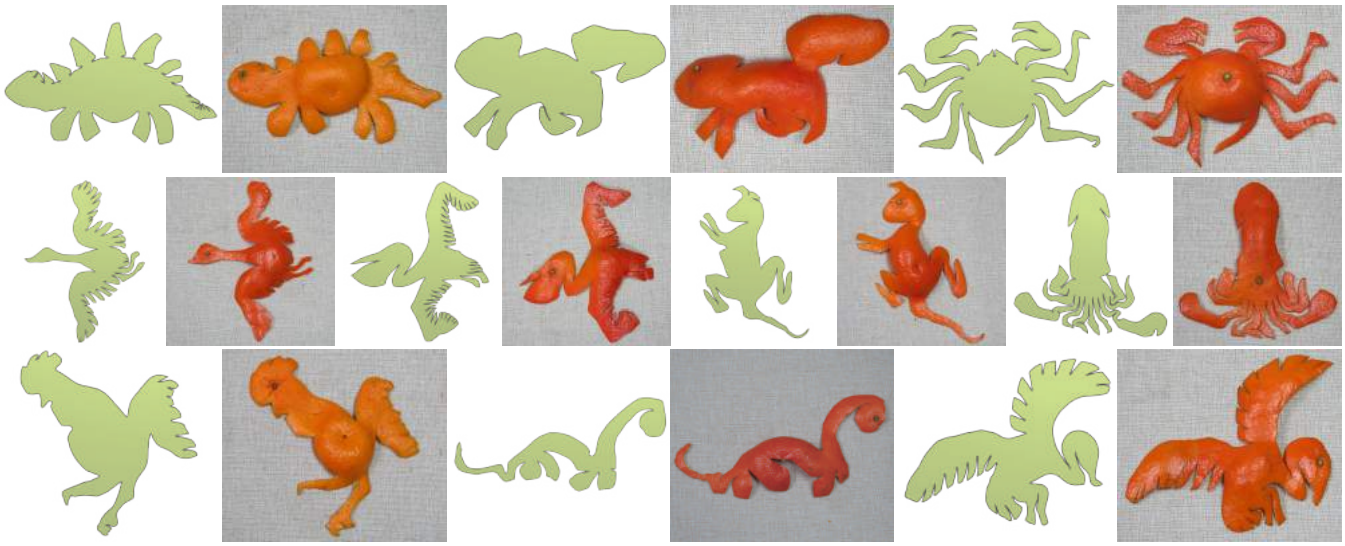


Fig. 20. Ten shapes designed by the Yoshihiro Okada. For each shape, we show the resulting 2D unfolded shape and the real peeling result cut by ourselves.

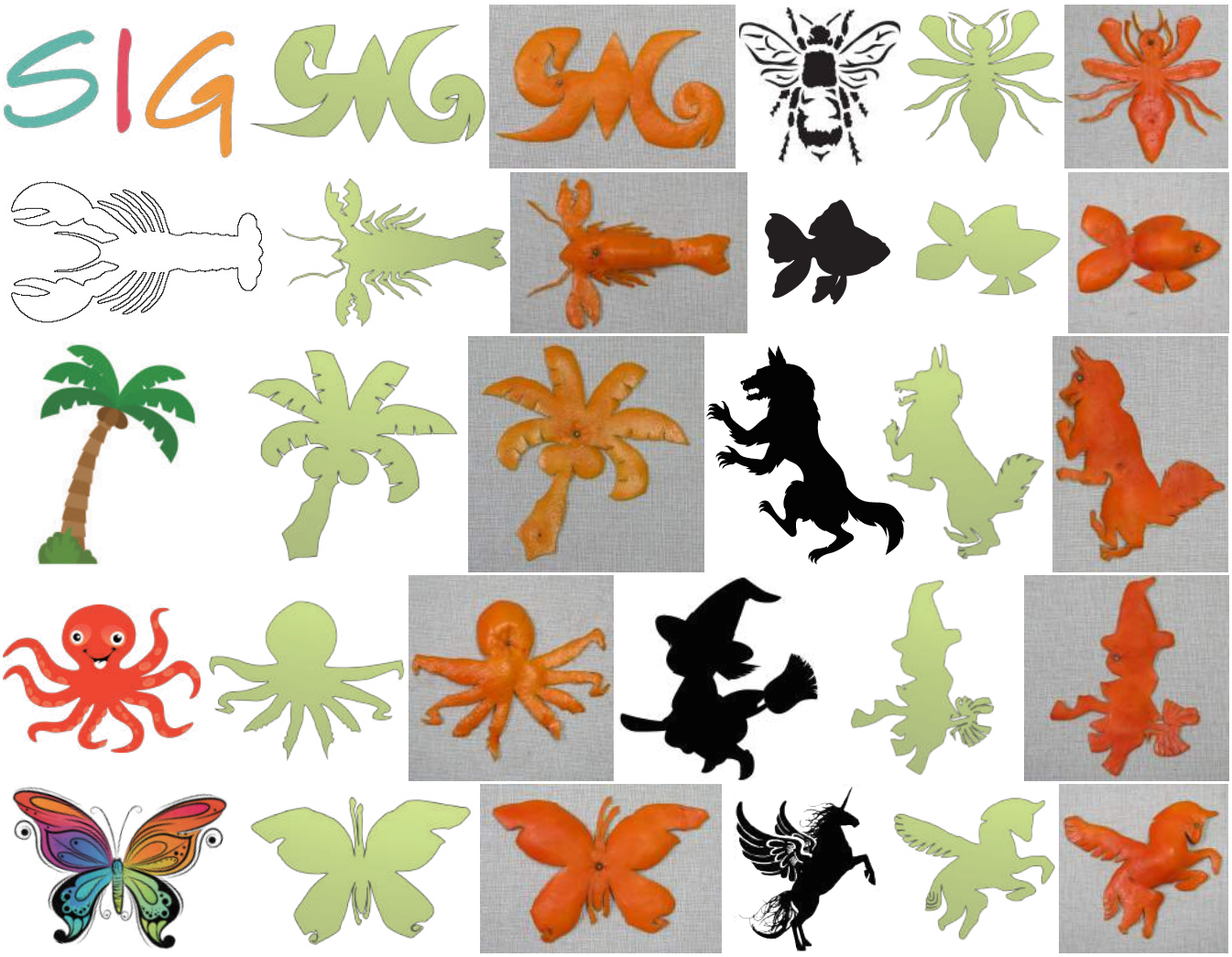


Fig. 21. Ten shapes designed by us. For each shape, we show the reference image (left), the resulting unfolded shape (middle), and the peeling result (right).

Fig. 22. Four example models designed and cut by the participants in the user study.

that differed greatly from the designed shape. By using six primary views for guidance, the novice failed. However, he took 31 minutes to succeed in peeling the Rabbit shape by using our graticule-based guidance. The novice told us that based on six primary views he could not imagine how to draw the cuts on the citrus. Our graticule-based guidance is much better, and we provide guidance for all shapes in the supplementary material. In our experiments, users usually take about 40 minutes to peel a citrus, including graticule drawing, cut drawing, and crafting.

6.2 Designs by authors

Shapes from Yoshihiro Okada. In total, we implement 25 shapes from [Okada 2010] or the Youtube webpage of Yoshihiro Okada. In Fig. 20, we show 10 examples, and all results are provided in the supplementary materials. Since these 2D shapes have been successfully completed by Yoshihiro Okada, this indicates that they are suitable for the citrus peeling art. Thus, we do not modify the input shapes and only use pre-alignments to avoid a local minimum when solving (5). Fig. 19 shows a comparison to Yoshihiro Okada. The peeling results are similar and comparable.

Shapes designed by ourselves. We collect 25 shapes from classical cartoon images and silhouette pictures. We show three examples in Fig. 1,and ten shapes in Fig. 21. All of the results are provided in the supplementary materials. Since these shapes have not been generated before, most of them are not suitable for the citrus peeling art. We use our system to design and construct them. The details of the resulting shapes may be quite different from the input shapes.

6.3 User study

We conducted a formative user study to evaluate our design system.

Participants. For the user study, we enrolled ten participants between the ages of 20 and 30. Their average age is 24 and the standard deviation of their ages is 1.79. The number of males and females are 6 and 4, respectively. No participant had used our design system and peeled a citrus into an elegant 2D shape before.

Task. The task for each participant is to first design a 2D shape, then generate cuts using our system, and finally peel a real citrus. Most participants had no experience or ability to paint their desired shapes. Thus, we asked them to search for images on the Internet and trace the boundary of the content to define the input 2D shape. Before using our design system, we first taught them how to use our software, and then they used a provided example to familiarize themselves with the software. After generating the cuts, we told them how to draw the cuts.

Subjective metrics. To test our design system, we recorded some subjective and objective metrics for each participant. The subjective metrics are evaluated by the post-interview-based method. After finishing the task, we asked them to measure their level of satisfaction with the peeled results, which are denoted as δ^{sat} (out of 100). For the design system, they were asked if it is easy to learn and operate. This score is denoted as δ^{sys} (out of 100). Finally, some comments were recorded to make note of their thoughts and opinions. Beyond the previously mentioned objective metrics, we also consider whether the user has successfully generated the desired shape.

Results. All participants succeeded in producing the designed shapes. Fig. 22 shows four examples and others are provided in the supplementary materials. Table 1 shows the objective metrics, and Table 2 presents the subjective metrics. The average and standard deviation of δ^{sat} over all participants are 86.4 and 6.65, respectively. This indicates that our system helps the participants successfully design their desired shapes. For δ^{sys}, the average and standard deviation over all participants are 85.5 and 6.93, respectively. According to these statistics, our system is learnable and easy to operate for novices. Most comments from the participants were positive (see the rightmost column in Table 2). The participants took about 14

Table 1. Objective metrics. N reports the iteration number of our iterative design process. N_1, N_2, and N_3 are the total number of interactions for the shape augmentation, the part deletion, and the angle augmentation, respectively. The last two columns indicate whether curvature reduction and pre-processing were done. The timings are reported in minutes.

Shape	t^{cut}	t^{gra}	t^{path}	t^{craft}	N	$N_1/N_2/N_3$	Cur	Pre
Fig. 1 - Parrot	17	19	18	19	8	14/7/6	Yes	Yes
Fig. 1 - Lizard	14	19	16	15	6	16/5/4	Yes	Yes
Fig. 1 - Fern	12	20	40	35	4	10/8/8	Yes	Yes
Fig. 17 - Rabbit	5	9	10	7	0	0/0/0	No	No
Fig. 19 - Dove	6	14	10	11	0	0/0/0	No	No
Fig. 19 - Eagle	6	18	17	16	1	0/0/0	No	Yes
Fig. 19 - Shrimp	7	20	20	20	1	0/0/0	No	Yes
Fig. 20 - Stegosaurus	5	14	13	17	0	0/0/0	No	No
Fig. 20 - Squirrel	4	10	11	10	0	0/0/0	No	No
Fig. 20 - Crab	7	16	18	18	1	0/0/0	No	Yes
Fig. 20 - Crane	5	18	21	14	0	0/0/0	No	No
Fig. 20 - Pelican	6	18	20	14	0	0/0/0	No	No
Fig. 20 - Kangaroo	5	16	15	10	1	0/0/0	No	Yes
Fig. 20 - Squid	6	12	15	16	1	0/0/0	No	Yes
Fig. 20 - Chicken	6	11	12	15	1	0/0/0	No	Yes
Fig. 20 - Tanystropheus	6	15	11	11	1	0/0/0	No	Yes
Fig. 20 - Egret	5	18	17	14	0	0/0/0	No	No
Fig. 21 - SIG	15	22	21	17	7	15/6/3	Yes	Yes
Fig. 21 - Bee	16	19	16	21	7	21/9/0	No	Yes
Fig. 21 - Lobster	12	20	18	24	4	11/5/2	No	Yes
Fig. 21 - Fish	7	8	6	12	2	4/1/3	No	No
Fig. 21 - Coconut tree	6	16	11	14	2	0/2/1	No	Yes
Fig. 21 - Wolf	14	16	14	17	8	19/8/6	Yes	Yes
Fig. 21 - Octopus	7	13	10	12	3	7/4/0	No	Yes
Fig. 21 - Witch	18	20	20	22	11	22/12/6	Yes	Yes
Fig. 21 - Butterfly	8	13	14	15	2	6/2/2	Yes	Yes
Fig. 21 - Unicorn	10	16	15	11	4	8/3/1	No	Yes
Fig. 22 - Mermaid	8	14	7	8	2	4/0/2	No	Yes
Fig. 22 - Flower	10	13	14	13	3	9/2/2	Yes	Yes
Fig. 22 - Fairy	11	18	12	16	4	7/3/0	No	Yes
Fig. 22 - Rabbit	6	11	6	10	1	0/0/0	No	Yes

minutes, 8 minutes, and 10 minutes to draw a graticule, to draw a completed cut, and to craft a citrus, respectively. The participants performed three iterations in the design process on average.

6.4 Timings

The algorithm was implemented in C++ using the Intel® Math Kernel Library for the linear solve. The design experiments were performed on a desktop PC with an Intel Core i7-4790K processor and 8GB memory. Typically, each local-global iteration takes about 0.80 seconds for $\mathcal{S}^m$ with 2000 vertices and $\mathcal{R}$ with 1500 vertices, while solving the global step (about 0.60 seconds) takes most of time. Then, the mapping process may take about one minute to solve (5). Since we perform remeshing during the optimization process, the size of the linear system in the global step changes, making the pre-factorization impossible. Thus, it is difficult to accelerate the solver. However, we note that our code is not optimized and we believe some speedups are possible. We will release our source code of the current implementation.

7 CONCLUSION

Our method provides a computational tool for peeling art design and construction. To solve the non-trivial cut generation problem, we novelly formulate it as a mapping problem that is simpler to solve.

Table 2. Subjective metrics for the user study.

Shape	Age/Gender	$\delta^{\text{sat}}/\delta^{\text{sys}}$	Comment
Fig. 22 - Mermaid	24/Female	90/92	It is easy to generate cut paths through some simple operations of the software, and the final result looks very similar to the input. I think this process is very interesting.
Fig. 22 - Flower	28/Male	92/80	The shape generated by the software is beyond my expectations and the final result is very satisfactory. The software interface is simple but not beautiful enough.
Fig. 22 - Fairy	20/Female	90/88	Even if the input shape is complex, the result is still impressive, and the main features are maintained. The peeling process is intuitive and effective.
Fig. 22 - Rabbit	26/Male	85/84	The software is easy to operate, and it is easy to peel a real citrus according to the graticule. However it takes a long time between two interactions.

(a) Input **(b)** **(c)** **(c)**

Fig. 23. User's design intention. A user wants to design a slender person (a), but due to the conservation rule, the resulting unfolded shapes (b,c,d) depict a larger person, which does not stay true to the user's design intention.

Unsuitable input 2D shapes are rectified by an iterative process that alternates in each iteration a pass to map $\mathcal{S}$ onto $\mathcal{M}$ and a pass to perform interactions for designing a new $\mathcal{S}$. We have demonstrated the capability of our method on 60 shapes.

Conservation principle. Given an input shape, our method can always generate an unfolded shape. However, due to the conservation principle, the unfolded shape may not meet a user's design intention (Fig. 23). If the conservation rule is not considered, we can easily construct the designed shape, but it would waste a lot of peels. Therefore, wasting as few peels as possible to meet user design needs would be an intriguing direction for future research.

Convergence analysis. Although our method for solving (5) converges for all testing examples in practice, there is no theoretical guarantee of convergence for any shape. However, if there are ideal assumptions (e.g., remeshing stops after certain steps, $\mathcal{M}$ is C^∞), we can prove the convergence (see the supplementary material).

Computational cost. Our current implementation cannot provide interactive feedback during user interactions, thus users may be frustrated between two interactions. Therefore, it would be interesting to make our implementation more efficient or to present a faster mapping method.

Potential applications. Although the proposed method is dedicated to peeling art design, the presented methodology has the following potential applications: (1) the mapping method can be used to map a pre-designed texture onto a 3D shape with constraints; (2) the interactive design system can be used to generate wrapping covers for 3D objects with near-developable surfaces; (3) and the proposed

rank-one energy may be useful for improving packing efficiency in atlas refinement [Limper et al. 2018; Liu et al. 2019] or removing sheets/chords in the structure optimization of all-hex meshes [Gao et al. 2017]. It is worthwhile to study these practical applications in the future.

ACKNOWLEDGMENTS

We would like to thank user study participants for evaluating our system and the anonymous reviewers for their constructive suggestions and comments. This work is supported by the National Natural Science Foundation of China (61802359, 61672482, 11626253), the Anhui Provincial Natural Science Foundation (1808085QF208), the Fundamental Research Funds for the Central Universities (WK0010460006, WK0010450004), and the One Hundred Talent Project of the Chinese Academy of Sciences.

REFERENCES

Bernd Bickel, Paolo Cignoni, Luigi Malomo, and Nico Pietroni. 2018. State of the Art on Stylized Fabrication. *Comput. Graph. Forum* 37 (2018).

Mario Botsch and Leif Kobbelt. 2004. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 185–192.

Mario Botsch and Olga Sorkine. 2008. On linear variational surface deformation methods. *IEEE. T. Vis. Comput. Gr.* 14, 1 (2008), 213–230.

Shuangming Chai, Xiao-Ming Fu, Xin Hu, Yang Yang, and Ligang Liu. 2018. Sphere-based Cut Construction for Planar Parameterizations. *Computer & Graphics (SMI 2018)* 74 (2018), 66–75.

Michael S. Floater and Kai Hormann. 2005. Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling*. Springer, 157–186.

Xiao-Ming Fu and Yang Liu. 2016. Computing Inversion-Free Mappings by Simplex Assembly. *ACM Trans. Graph. (SIGGRAPH ASIA)* 35, 6 (2016).

Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015. Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 71:1–71:12.

Xifeng Gao, Daniele Panozzo, Wenping Wang, Zhigang Deng, and Guoning Chen. 2017. Robust Structure Simplification for Hex Re-meshing. *ACM Trans. Graph. (SIGGRAPH ASIA)* 36, 6 (2017), 185:1–185:13.

Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire Mesh Design. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014), 66:1–66:12.

Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. 2002. Geometry Images. *ACM Trans. Graph. (SIGGRAPH)* 21, 3 (2002), 355–361.

Xin Hu, Xiao-Ming Fu, and Ligang Liu. 2018. Advanced Hierarchical Spherical Parameterizations. *IEEE. T. Vis. Comput. Gr.* 24, 6 (2018), 1930–1941.

Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. 2017. Simplicial Complex Augmentation Framework for Bijective Maps. *ACM Trans. Graph. (SIGGRAPH ASIA)* 36, 6 (2017), 186:1–186:9.

Conrado R. Ruiz Jr., Sang N. Le, Jinze Yu, and Kok-Lim Low. 2014. Multi-style Paper Pop-up Designs from 3D Models. *Comput. Graph. Forum (EG)* 33, 2 (2014), 487–496.

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. In *Comput. Graph. Forum*, Vol. 24. 581–590.

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J. Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved Folding. *ACM Trans. Graph.* 27, 3 (2008), 75:1–75:9.

Martin Kilian, Aron Monszpart, and Niloy J. Mitra. 2017. String Actuated Curved Folded Surfaces. *ACM Trans. Graph.* 36, 4 (2017).

Shahar Z. Kovalsky, Noam Aigerman, Ronen Basri, and Yaron Lipman. 2015. Large-scale bounded distortion mappings. *ACM Trans. Graph. (SIGGRAPH ASIA)* 34, 6, Article 191 (2015), 10 pages.

Sang N. Le, Su-Jun Leow, Tuong-Vu Le-Nguyen, Conrado Ruiz, and Kok-Lim Low. 2014. Surface and Contour-Preserving Origamic Architecture Paper Pop-Ups. *IEEE. T. Vis. Comput. Gr.* 20, 2 (2014), 276–288.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. (SIGGRAPH)* 21, 3 (2002), 362–371.

Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. 2018. OptCuts: Joint Optimization of Surface Cuts and Parameterization. *ACM Trans. Graph. (SIGGRAPH ASIA)* 37, 6 (2018).

Xin Li and SS Iyengar. 2015. On computing mapping of 3d objects: A survey. *ACM Computing Surveys (CSUR)* 47, 2 (2015), 34.

Xian-Ying Li, Tao Ju, Yan Gu, and Shi-Min Hu. 2011. A Geometric Study of V-style Pop-ups: Theories and Algorithms. *ACM Trans. Graph.* 30, 4 (2011), 98:1–98:10.

Xian-Ying Li, Chao-Hui Shen, Shi-Sheng Huang, Tao Ju, and Shi-Min Hu. 2010. Popup: Automatic Paper Architectures from 3D Models. *ACM Trans. Graph.* 29, 4 (2010), 111:1–111:9.

Max Limper, Nicholas Vining, and ALLA SHEFFER. 2018. Box Cutter: Atlas Refinement for Efficient Packing via Void Elimination. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 153:1–153:13.

Yaron Lipman. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. (SIGGRAPH)* 31, 4 (2012), 108:1–108:13.

Hao-Yu Liu, Xiao-Ming Fu, Chunyang Ye, Shuangming Chai, and Ligang Liu. 2019. Atlas Refinement with Bounded Packing Efficiency. *ACM Trans. Graph. (SIGGRAPH)* 38, 4 (2019), 33:1–33:13.

Ligang Liu, Chunyang Ye, Ruiqi Ni, and Xiao-Ming Fu. 2018. Progressive Parameterizations. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 41:1–41:12.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A local/global approach to mesh parameterization. *Comput. Graph. Forum (SGP)* 27, 5 (2008), 1495–1504.

Fady Massarwi, Craig Gotsman, and Gershon Elber. 2007. Papercraft models using generalized cylinders. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. 148–157.

J. Mitani and H. Suzuki. 2004a. Computer aided design for Origamic Architecture models with polygonal representation. In *Proceedings Computer Graphics International, 2004.* 93–99.

Jun Mitani and Hiromasa Suzuki. 2004b. Making Papercraft Toys from Meshes Using Strip-based Approximate Unfolding. *ACM Trans. Graph.* 23, 3 (2004), 259–263.

Yoshihiro Okada. 2010. *Atarashii mikan no mukikata : zennijūgoshu* (1st ed.). Tōkyō : Shōgakukan.

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Trans. Graph. (SIGGRAPH ASIA)* 36, 6 (2017).

Emil Praun and Hugues Hoppe. 2003. Spherical Parametrization and Remeshing. *ACM Trans. Graph. (SIGGRAPH)* 22, 3 (2003), 340–349.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 2 (2017), 16:1–16:16.

Pedro V. Sander, Steven J. Gortler, John Snyder, and Hugues Hoppe. 2002. Signal-specialized Parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering*. 87–98.

P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. 2003. Multi-chart Geometry Images. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 146–155.

Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Trans. Graph.* 37, 1 (2017), 5:1–5:14.

Nicholas Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 156:1–156:13.

Idan Shatz, Ayellet Tal, and George Leifman. 2006. Paper craft models from meshes. *The Visual Computer* 22, 9 (2006), 825–834.

Alla Sheffer. 2002. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Shape Modeling International*. 61–66.

Alla Sheffer and Eric de Sturler. 2001. Parameterization of faceted surfaces for meshing using angle-based flattening. *Eng. Comput.* 17, 3 (2001), 326–337.

Alla Sheffer and John C Hart. 2002. Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of the conference on Visualization'02*. 291–298.

Alla Sheffer, Emil Praun, and Kenneth Rose. 2006. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.* 2, 2 (2006), 105–171.

Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. 2014. Designing Inflatable Structures. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014), 63:1–63:10.

Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 70:1–70:9.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*. 109–116.

Tomohiro Tachi. 2009. 3D origami design based on tucking molecule. In *The Fourth International Conference on Origami in Science, Mathematics, and Education, R. Lang, ed., Pasadena*. 259–272.

T. Tachi. 2010. Origamizing Polyhedral Surfaces. *IEEE. T. Vis. Comput. Gr.* 16, 2 (2010), 298–311.

Masahito Takezawa, Takuma Imai, Kentaro Shida, and Takashi Maekawa. 2016. Fabrication of Freeform Objects by Principal Strips. *ACM Trans. Graph.* 35, 6 (2016), 225:1–225:12.

Zoe J. Wood, Paul Muhl, and Katelyn Hicks. 2016. Computational Art: Introducing High School Students to Computing via Art. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. 261–266.

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2005. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.* 24, 1 (2005), 1–27.

Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: Stretch-driven Mesh Parameterization Using Spectral Analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 45–54.