4D Printing for Freeform Surfaces: Design Optimization of Origami and Kirigami Structures

Tsz-Ho Kwok[1,2], **Charlie C.L. Wang**[1], **Dongping Deng**[2], **Yunbo Zhang**[1,3], **Yong Chen**[2*]

[1]Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong

[2]Epstein Department of Industrial and Systems Engineering, University of Southern California

[3]School of Mechanical Engineering, Purdue University

A self-folding structure fabricated by additive manufacturing can be automatically folded into a demanding 3D shape by actuation mechanisms such as heating. However, 3D surfaces can only be fabricated by self-folding structures when they are flattenable. Most generally designed parts are not flattenable. To address the problem, we develop a shape optimization method to modify a non-flattenable surface into flattenable. The shape optimization framework is equipped with topological operators for adding interior/boundary cuts to further improve the flattenability. When inserting cuts, self-intersection is locally prevented on the flattened 2D pieces. The total length of inserted cuts is also minimized to reduce artifacts on the finally folded 3D shape.

Keywords: *Additive manufacturing, Flattenable, Self-folding, Origami, Kirigami, Computer-Aided Design*

1 Introduction

Additive manufacturing (AM) is a promising technique for fabricating Three-Dimensional (3D) complex shapes, which are difficult to be fabricated by traditional manufacturing processes. Currently, most AM processes are layer-based. However, such approach has drawbacks such as fabrication speed is slow and the built parts have anisotropic stiffness (i.e., weaker in one direction comparing to others). Recently, inspired by Origami and Kirigami [1,2], a new AM technique based on self-folding structures has been developed to overcome the problems of the conventional layer-based fabrication approach [3,4]. The self-folding approach this is also called *4D printing*. In other fields, self-folding structures have also attracted a lot of attentions in biomedical and robotic applications (e.g., [5,6]). Instead of directly fabricating the designed 3D shape, this new manufacturing method first fabricates a two-dimensional (2D) part. Then, the part will be self-folded into the designed 3D shape using certain stimulating conditions (e.g., heat or magnetic). Models fabricated by this method have reconfigurable shapes, and the procedure of fabrication is fast since only a few active/passive layers need to be made by additive manufacturing.

1.1 Motivation

Not every 3D shape can be made by a 2D self-folding structure. To fabricate a model by self-folding, the 3D shape must be flattenable – i.e., it can be flattened into a 2D pattern without stretching. A 3D surface that has such a geometric property is called flattenable. A designed 3D shape is often non-flattenable, and some geometric details cannot be reconstructed if such a shape is fabricated by a non-optimized self-folding structure (refer to an example in the top row of Fig.1). In the prior work of self-folding structures (e.g., [3]), the 3D models to be fabricated are cut into strips in order to make them flattenable. Nevertheless, artifacts are left at the places where cuts are introduced after folding. The more the cuts, the more artifacts are resulted on the final folded part. In this paper, we aim at tackling this problem by optimizing a designed surface that is fabricated by self-folding structures. The following two questions will be answered:

1. Given a 3D shape M that is not flattenable, how to optimize the shape of M into a flattenable one M' while minimizing the shape similarity error between M and M'?
2. When cuts need to be added in order to get a shape that is more similar to M, how to minimize the number and the length of cuts on M'?

Moreover, self-intersection must be prevented on the flattened 2D piece of M'. Before finding answers of the above questions, we review the related literatures on self-folding structures and geometric computation approaches.

1.2 Related Works

Self-folding structures (also called self-transforming or self-evolving structures) are usually designed to deform their shapes in a pre-defined way, where the shape variation can

[*]Corresponding author: yongchen@usc.edu, (213) 740-7829

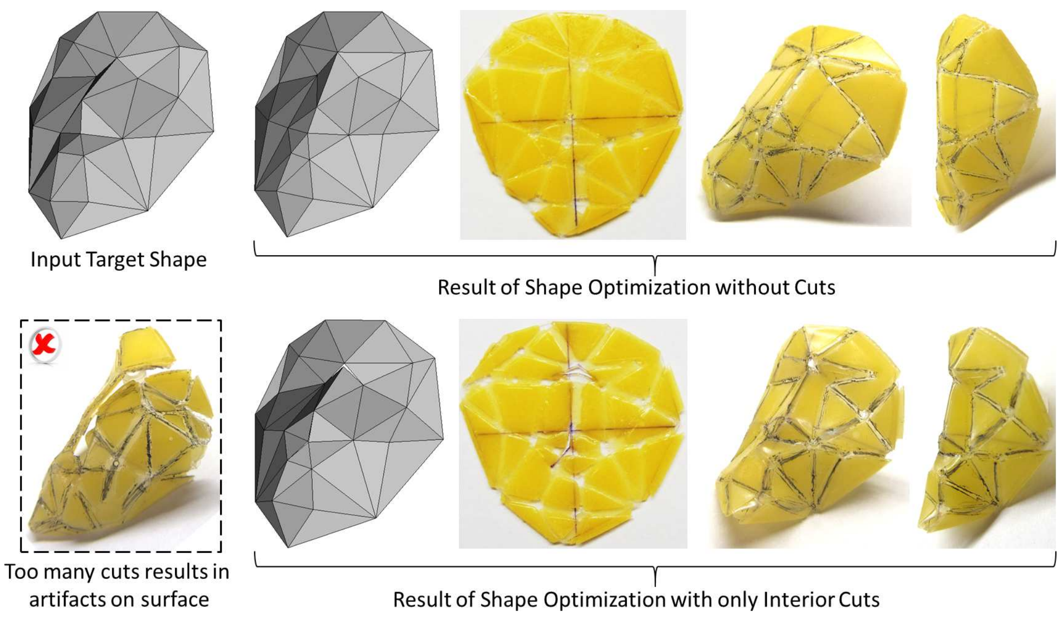
Input Target Shape
Result of Shape Optimization without Cuts
Too many cuts results in artifacts on surface
Result of Shape Optimization with only Interior Cuts

Fig. 1. For an input freeform surface to be fabricated by Origami-structure – the face model, our shape optimization can generate a 2D pattern for producing a self-folding structure by AM, which can be folded into a face model by heating (see the top row). Cuts can be automatically added onto the model to further improve the shape similarity of fabricated surface comparing to the input model. Adding too many cuts can generate unwanted artifacts on the surface of fabricated part – see the bottom-left corner for an example.

be induced by different physical stimulations to fold, expand, shrink and curl. The idea was first developed by using hydrophilic materials that can be activated when being submerged in water. Thereafter, different self-folding mechanisms have been developed in the form of planar sheets by using different folding principles – e.g., shape memory materials [7], bilayer structures [8–10], inhomogeneous materials [11], and Shrinky-Dinks films [12]. They can be triggered by different conditions, including thermal [13,14], microwaves [15], or humidity [16,17]. The Shrinky-Dinks film shrinks when it is exposed to an environment having a temperature higher than the transition temperature of the film. By controlling the area of exposure on a film surface, the bending movements can be designed. Using this idea, our recent paper [4] has calibrated the mapping between the width of a hinge (in the form of exposed film) and the bending angle of this hinge under heating. Many other researches have also demonstrated the self-folding techniques using different approaches. Nevertheless, the previous works mainly focus on the transformation of simple shapes, for which manual modeling was used in their construction (e.g., [18]). Recently, Raviv et al. [19] showed how to construct and simulate a complex solid structure that bends and stretches over time; however, the work only considers a 2D grid skeleton. In this paper, we aim at automatically designing self-folding structures for fabricating 3D freeform surfaces.

Given a 3D shape, computing its corresponding 2D pattern is known as the surface flattening problem in Computer-Aided Design/Manufacturing (CAD/CAM) [20]. The computation of surface flattening is usually formulated as minimizing different surface metrics, such as angle, distance, area. However, no mapping can fully reduce the distortion on a general 3D shape, and the distortion increases significantly when the freeform surfaces are more complex in shape [21]. Kilian *et al.* [22] presented an optimization-based computational framework for the design and reconstruction of general developable surfaces. They optimize a pair of models in the form of 3D mesh and 2D pattern while maintaining an isometric mapping between them. However, models with high surface complexity have not been considered, where topological operations need to be performed. Specifically, cuts need to be added to reduce the distortion. Sheffer [23] proposed a method based on a minimum spanning tree that passes through points with high Gaussian curvature to compute cuts with a minimal total length. Wang *et al.* [24] introduced cutting paths starting from a point with maximum Gaussian curvature to the surface boundary to reduce stretches in surface flattening. In both methods, cuts are added in the way of linking critical points and the surface boundary; however, such cuts maybe long and have the risk of introducing self-intersections on the flattened 2D piece. This paper proposes a different strategy to add cuts in a more effective and safe way.

The approaches based on directly optimizing a 3D shape to improve its flattenability converges slowly. For instance, Wang [25] introduced the Flattenable Laplacian (FL) mesh and used Newton's method to optimize the mapping to be isometric. The problem is formulated as a constrained nonlinear optimization problem, and a scheme akin to multigrid solver was employed to improve the convergence of the computation. Other direct optimization approaches such as [26, 27] also have similar problems. Different from the prior work, we conduct a flattening/folding deformation approach in this work.

1.3 Main Results

We first flatten a 3D surface M into a 2D pattern D with minimal stretching. The 2D pattern is then folded into a shape $M^{(1)}$ similar to M while keeping the rigidity of each fragment on D. After that, $M^{(1)}$ is flattened into a new 2D pattern $D^{(1)}$, which will be folded back to get a better approximation of M. Iteratively applying the aforementioned flattening and the folding steps can result in a pair of 2D and 3D models $< D^{(n)} :: M^{(n)} >$, where $D^{(n)}$ and $M^{(n)}$ are isometric and $M^{(n)}$ gives a good approximation of M.

When cuts are needed to further reduce the error between $M^{(n)}$ and M, most of the prior approaches add cuts from the interior of a surface to its boundary. Our study finds that adding such long cuts may not be optimal in many cases. There are different types of non-flattenable surfaces. While adding cuts to boundary is appropriate for elliptic surfaces, it could lead to unwanted self-intersection for hyperbolic surfaces. It is also known that a 2D pattern with self-intersection cannot be fabricated using a self-folding structure. In this research, a novel interior cutting scheme is introduced. An algorithm for progressively applying interior and boundary cuts has been developed. Based on it, our method can prevent self-intersection on 2D patterns and reducing the total cutting length at the same time.

The technical contributions of this work are:

1. We develop a new computation tool for designing self-folding structures from a general 3D freeform surface.
2. We propose a framework of iterative flattening/folding to optimize the flattenability of an input 3D surface, and simultaneously find the flattenable 3D shape as well as its corresponding 2D pattern.
3. We develop a hybrid cutting algorithm, which can prevent self-intersection on 2D patterns and minimize the surface deflects that caused by the added cuts.

Experimental tests using the fabrication method of 4D printing have been performed to verify the effectiveness of the developed design optimization framework for self-folding structures.

2 Methodology

2.1 Fabrication by Additive Manufacturing

In our previous work [4], a fabrication method based on additive manufacturing has been presented for self-folding structures. This method will be employed in the physical tests for verifying the design results that are generated by our approach. We briefly introduce the principle of the fabrication approach here. A multi-layered structure with both active and passive materials is employed, where different material layers undergo various ratios of volumetric shrinkage after heating. As a result, a structure made in this way can be self-folded into a desired configuration. In our setup, pre-strained polystyrene films that shrink under heating – the Shrinky-Dinks films [12] – are used as the layer of active material. The layers of passive materials are produced from photocurable resin by additive manufacturing – the *mask-image-projection-based stereolithography* (MIP-SL) process

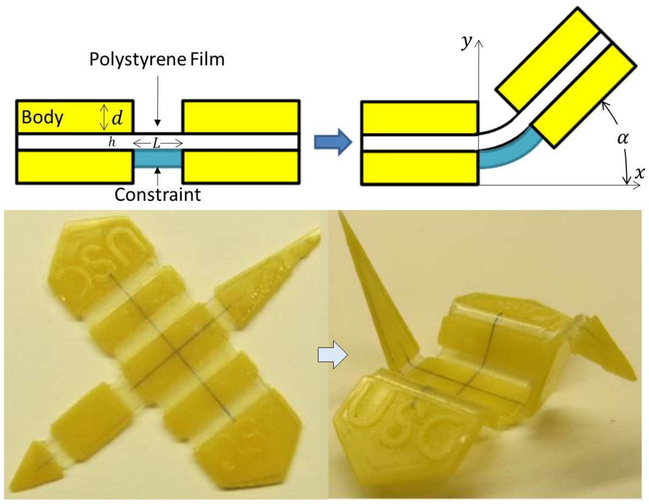

Fig. 2. An illustration for the working principle of self-folding [4]. (Top) The self-folding structure is printed on a shrinking film with body and constraining layers. It is bended when heat is applied. (Bottom) The test case of a self-folded crane model is shown.

is used [28]. Fig.2 illustrates the described working principle and shows a crane example that is self-folded from a designed 2D pattern. After modeling and calibrating the 2D patterns on the active layer, we can control the bending angle on every hinge by designing different 2D patterns on a hinge. Specifically, the bending angle (α) depends on the thicknesses of the active layer (h) and the passive layers (d), as well as the width of a hinge (L).

2.2 Terminology and Mathematical Definitions

The necessary terminology and mathematical definitions of our work are presented in this section. First of all, the computation of design optimization is taken on an abstractive graph of the self-folding structure that is defined as follows.

Definition 1 The abstractive graph of a self-folding structure is represented by a mesh surface $M = \{F, E, V\}$, where F, E and V are the sets of faces, edges and vertices, respectively.

Referring to the self-folding structure, F defines the building blocks on the passive layer, E defines the hinges where the bending happens, and V gives the geometric shape in both 3D and 2D. To simplify the analysis and the computation, only two-manifold mesh surfaces are considered in this study. Each interior edge of a two-manifold mesh has two adjacent faces. The angle between the normals of two neighboring faces in 3D defined the bending angle of the corresponding hinge. When there are n_e interior edges, we need to determine n_e hinges with the set of bending angles as $\{\alpha_1, \alpha_2, \ldots, \alpha_{n_e}\}$.

Definition 2 For each vertex i in V, there is a position $\mathbf{v}_i$ in 3D and a corresponding 2D planar coordinate $\mathbf{p}_i$.

The 2D pattern of a self-folding structure is denoted by

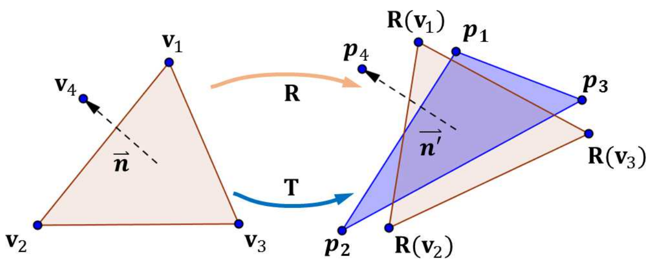

Fig. 3. For a triangle with vertices $(\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3)$ (left) deformed a shape in blue by an affine transformation $\mathbf{T}$, we can extract the pure rotation matrix $\mathbf{R}$ from $\mathbf{T}$ using SVD.

D, on which the vertices are placed at $\{i \in V | \mathbf{p}_i\}$.

Remark 1 The mapping, $\Gamma : M \mapsto D$, need to be *isometric* as the building blocks in F are inelastic.

When giving the 3D and 2D shapes of M and D, whether Γ is isometric can be checked by the invariance of edge length.

Remark 2 The mapping, $\Gamma : M \mapsto D$, is isometric if and only if

$$E_{iso} = \sum_{\forall e \in E} (\|\mathbf{v}_s\mathbf{v}_q\| - \|\mathbf{p}_s\mathbf{p}_q\|)^2 \equiv 0 \tag{1}$$

where s and q are two vertices of the edge e.

From differential geometry [29], it is known that only developable surfaces have isometric mappings to planar shapes. The mathematical definition of developable surface is given on differentiable surfaces relating to Gaussian curvature. Generally, a surface is developable if and only if the Gaussian curvature at any point is zero except the boundary points, which do not have Gaussian curvature. Here, we adopt a discrete interpretation of developable surface and name it as flattenable (or non-flattenable) by using the definitions in [25].

Definition 3 For a triangular mesh vertex v, if and only if the summed inner angle $\theta(v)$ around it is identical to 2π, the triangles around it can be flattened onto a plane without distortion; such a vertex is called *flattenable vertex*.

Remark 3 A triangular mesh patch is flattenable when all its interior vertices are flattenable, which can be measured by a metric

$$E_{flt} = \sum (\theta(v) - 2\pi)^2 \equiv 0 \tag{2}$$

with all interior vertex v.

The problem of the isometric metric E_{iso} and the flattenable metric E_{flt} is that the direct computation based on them does not lead to an optimization framework that can converge fast (see [21, 25]). Instead, we define a new function based on measuring the rigidity of the transformation between the faces in 3D and 2D. Such rigidity measurement can be considered as a weak form of the isometry and flattenability metrics.

For each triangle $f \in M$ with three vertices at $(\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3)$ in 3D and $(\mathbf{p}_1\ \mathbf{p}_2\ \mathbf{p}_3)$ in 2D, we add the fourth accessory point $\mathbf{v}_4$ along the unit normals of the triangle rooted at the center of f. Similar, $\mathbf{p}_4$ is also added for the corresponding triangle in D.

Remark 4 The transformation from 2D to 3D is $\mathbf{T} = \mathbf{P}\hat{\mathbf{P}}^{-1}$ with

$$\mathbf{P} = [\mathbf{v}_1^t - \mathbf{v}_4^t\ \mathbf{v}_2^t - \mathbf{v}_4^t\ \mathbf{v}_3^t - \mathbf{v}_4^t]$$
$$\hat{\mathbf{P}} = [\mathbf{p}_1^t - \mathbf{p}_4^t\ \mathbf{p}_2^t - \mathbf{p}_4^t\ \mathbf{p}_3^t - \mathbf{p}_4^t]$$

by aligning the centers of 3D and 2D coincident.

Note that, the planar positions are also represented as 3D vectors in this formulation. In general, $\mathbf{T}$ is not a rigid transformation (see Fig.3 for an illustration). Its nearest rigid transformation $\mathbf{R}$ can be obtained by first computing the *Singular Value Decomposition* (SVD) of $\mathbf{T}$: $\mathbf{T} = \mathbf{U}\Sigma\mathbf{V}^\top$, and then eliminating the scaling matrix Σ as $\mathbf{R} = \mathbf{U}\mathbf{V}^\top$. Based on this, the rigidity of Γ can be evaluated by the following metric.

Definition 4 The rigidity of mapping $\Gamma : M \mapsto D$ is defined by the Frobenius norm $\|\cdot\|_F$ as

$$E_{rgd} = \sum_{k=1}^{n_f} \|\mathbf{T}_k - \mathbf{R}_k\|_F^2. \tag{3}$$

Similar to this formulation, when a transformation is from 3D to 2D (e.g., in the flattening), $\mathbf{T}$ is formulated in an inverse way as $\mathbf{T} = \hat{\mathbf{P}}\mathbf{P}^{-1}$.

Generally, designed 3D freeform surfaces are non-flattenable. As a result, we need to compute 2D pieces that can be folded back into the shapes similar to the designed surfaces. To conduct such optimization, a metric is needed to evaluate the shape similarity between a shape M' that is folded back from a 2D piece and the designed 3D shape M.

Definition 5 Assume M' and M have the same connectivity. The shape approximation error is defined as

$$E_{apx} = \sum_{i=1}^{n_v} \|\mathbf{v}_i - \mathbf{v}_i'\|^2, \tag{4}$$

where $\mathbf{v}_i$ and $\mathbf{v}_i'$ are the positions of a vertex on M and M'.

2.3 Algorithm Overview

To fabricate a designed 3D shape by a self-folding structure, the current practice is to introduce many cuts on the input surface so that the surface is tessellated into a strips. Nevertheless, as mentioned before, unwanted artifacts are also created for the added cuts. We develop a design optimization approach to address the problem by minimizing the shape approximation error between the designed surface and the surface that is folded from a planer piece. When

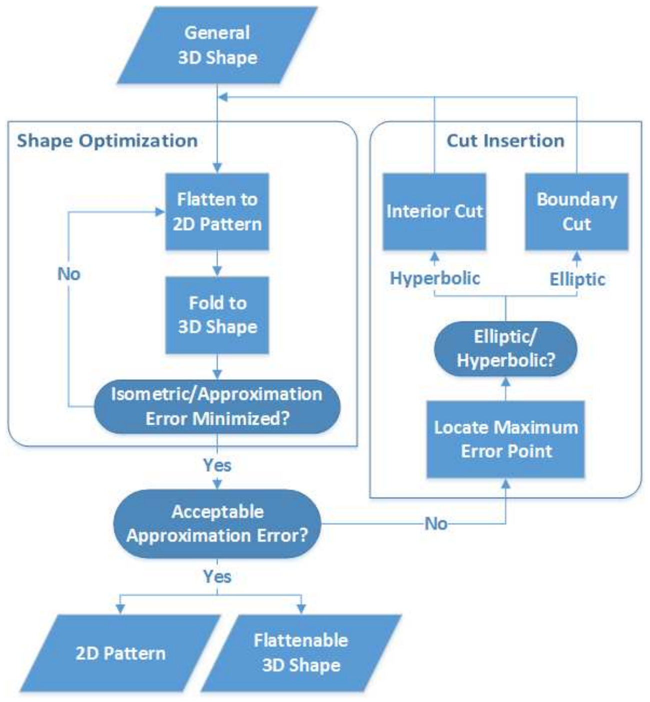

Fig. 4. Algorithm overview. The details of flattening and folding simulation are presented in Sections 3.1 and 3.2 respectively. Our hybrid cutting insertion algorithm will be introduced in Section 4.

cut insertion cannot be avoided, we attempt to add cuts as-short-as-possible. In other words, both the geometry and the topology of the input mesh M will be optimized.

The overview of our algorithm is illustrated in Fig.4. Our algorithm has two nested loops of iterations.

Inner Loop: The inner loop takes care of the shape optimization to improve the flattenablity of a given shape by iteratively applying flattening/folding deformations. The positions of vertices in 3D and 2D meshes are optimized together to minimize both the approximation error E_{apx} and the isometric error E_{iso} (by E_{rgd}).

Outer Loop: When E_{apx} obtained after running the inner loop of the shape optimization is too high, cuts are iteratively added in the outer loop. In each step of iteration, a cut is applied carefully in order to prevent self-intersections in 2D pattern. Operations are developed to generate as-short-as-possible cuts to preserve the quality of the fabricated surface.

After applying cuts in the outer loop, the inner loop of shape optimization is run again to further improve the shape approximation. Working together, the computation taken by these two levels of iterations converges rather quickly. The framework proposed in this paper provides a useful tool for designers when using the 4D-printing technique to fabricate self-folding structures.

3 Shape Optimization

Our idea in optimizing a non-flattenable 3D shape M is to iteratively get 2D patterns that are nearly isometric to M via the repeated process of flattening and folding. When flattening a 3D surface, the stretch caused by the difference between the 2D and 3D surfaces should be minimized. After getting a 2D pattern, it will be folded to approximate the input 3D shape M. The steps of flattening and folding are iteratively applied until both the approximation error E_{apx} and the rigidity error E_{rgd} have been minimized. We stop the iteration when neither E_{apx} nor E_{rgd} drops any more. Note that, in this shape optimization, the mesh topology is not adjusted. Both flattening and folding operations only add deformations on the model. Therefore, only vertices of a mesh are moved during the iterations of the shape optimization. The final positions of vertices are what we are interested in this design stage.

3.1 Elastic Flattening

Flattening a surface M into a 2D pattern is in fact introducing a deformation from 3D to 2D by eliminating one dimension [30]. The flattening process simulates an elastic deformation with the isometric energy minimized. Directly formulating the optimization by E_{iso} will make the problem highly non-linear, and results in a very slow convergence. To overcome this difficulty, a weak form E_{rgd} will be used instead. The 2D shape D can be obtained by moving the mesh points of M and determining the values of the 2D positions $\{\mathbf{p}_i\}$ as

$$\operatorname*{argmin}_{\{\mathbf{p}_i\}\in\Re^2} E_{rgd}. \tag{5}$$

There are two sets of unknowns in this optimization – the set of planar positions $\{\mathbf{p}_i\}$ and the set of rotation matrices on faces $\{\mathbf{R}_k\}$. They are correlated to each other. To decouple this, we linearize the numerical computation in two orthogonal directions by using the local/global strategy. Specifically, when the initial positions in 2D are known, we can treat $\{\mathbf{p}_i\}$ as known and compute $\{\mathbf{R}_k\}$ by SVD applied on $\mathbf{T}_k = \hat{\mathbf{P}}_k\mathbf{P}_k^{-1}$. As SVD is taken locally on each face, the computation can be completed very efficiently. For the first iteration, the initial values of $\{\mathbf{p}_i\}$ are determined by the least square conformal map [31]. In the later iterations, planar positions determined in the previous step are used to calculate $\{\mathbf{R}_k\}$. After obtaining the rotation matrices, new values of $\{\mathbf{p}_i\}$ are computed in the global sense. As $\mathbf{T}_k = \hat{\mathbf{P}}_k\mathbf{P}_k^{-1}$ is in a linear form of the planar positions, Eqn.(5) can be rewritten into a least-square system

$$\operatorname*{argmin}_{\mathbf{x}=\{\mathbf{p}_i\in\Re^2\}} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2,$$

where $\mathbf{A}$ is a sparse matrix derived from $\{\mathbf{T}_k\}$, and $\mathbf{b}$ is a vector containing entries from $\{\mathbf{R}_k\}$.

These two steps are then iterated until converged, and the result is a 2D pattern D flattened from the input 3D shape M. However, if the 3D shape itself is not flattenable, E_{rgd} can hardly be minimized into zero. In such cases, we simulate

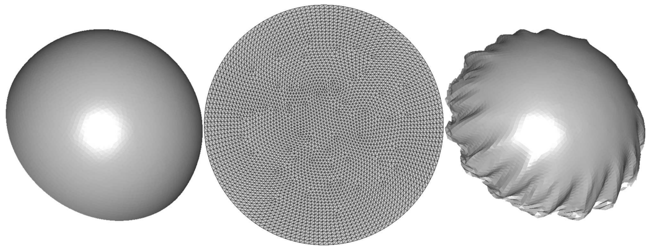

Fig. 5. Shape optimization: (Left) A highly non-flattenable surface – half sphere – is input to our system. (Middle) The 2D pattern computed by our system. (Right) The simulated 3D shape folded up from the 2D pattern.

the folding process on D to get a folded shape M' that is as-similar-as possible to M.

3.2 Folding Simulation

In this simulation, a 2D pattern D is folded to approximate the given 3D shape M. As only bending along the edges are allowed, the isometric energy E_{iso} is demanded to be *zero* during the process of folding. The deformation can be simulated by moving the vertices of D to get the 3D positions $\mathbf{v}_i$ which is driven by minimizing the shape approximation error, that is

$$\underset{\{\mathbf{v}_i \in \Re^3\}}{\operatorname{argmin}} E_{apx} \qquad s.t., \, E_{iso} = 0. \tag{6}$$

Again, to improve the convergence of computation, the weak form of isometric metric – the rigidity energy E_{rgd} is used to replace E_{iso}. After using the Lagrange multiplier, the non-constrained optimization is taken as

$$\underset{\{\mathbf{v}_i \in \Re^3\}}{\operatorname{argmin}} E_{apx} + \lambda E_{rgd} \tag{7}$$

to compute the folded shape of D. The penalty coefficient λ controls the influence of the rigidity during the deformation. If λ is too small, it becomes an elastic deformation that is not what we want. However, if an extremely large coefficient λ is chosen, E_{apx} will be ignored and it is hard to fold D. Therefore, a reasonable large value should be chosen. In our simulation, the penalty coefficient $\lambda = 10.0$ is employed. It is easy to find that the formula in Eqn.(7) is in the quadratic form and results in a fast convergence.

To demonstrate the functionality of our system, we apply the flattening/folding simulation to a highly non-flattenable surface – hemisphere (refer to Fig.5 for an illustration). Although it is impossible to find a 2D pattern without stretch, our approach can find a perfect circular disk that is the best approximation for the unfolded 2D pattern. Also, the shape warped back from the planar disk shows a good approximation of the hemisphere. The major defect comes from the unwanted wrinkles appeared on the folding result; however, this matches well with the physical phenomena happened in nature. In summary, our shape optimization framework can effectively compute a pair of 2D/3D shapes for approximating an input 3D shape. In next section, topological operations will be added to further improve the design optimization on self-folding structures.

4 Cut Insertion

The framework of shape optimization presented in Section 3 can reduce the isometric error in finding a flattenable shape that is a good approximation of a non-flattenable shape. However, if the shape is highly non-flattenable (e.g., the face models as shown in Fig.1), the part fabricated from the computed 2D pattern will have large approximation error. Consequently, cuts must be added to reduce the surface stretch during flattening. Accordingly, a better 2D pattern can be obtained that may result in a better 3D folding result. When inserting a cut, the following two requirements need to be fulfilled:

1. The cuts will not lead to self-intersections (locally) in the 2D pattern;
2. The total length of the inserted cuts needs to be minimized.

For requirement 1, it will be too difficult to foresee any global self-intersections that maybe introduced before the cut is really applied. Therefore, in determining cut insertions, it should at least guarantee no local self-intersection, and we leave the checking of the global self-intersection after inserting the cut.

We are motivated by Definition 3 and Remark 3 to add cuts through those vertices with the sum of surface angle, $\theta(\cdots)$, that are far different from 2π. Specifically, there are two kinds of non-flattenable vertices: one is with $\theta(v_p) < 2\pi$ (called *elliptic vertex*), another one has $\theta(v_p) > 2\pi$ (called *hyperbolic vertex*), which are illustrated in Fig.6. For a non-flattenable vertex with $\theta(v_p) < 2\pi$, when unfolding its adjacent faces into 2D, the incident angles are forced to become larger (stretched). In contrast, for the surface with $\theta(v_p) > 2\pi$, the incident angles are enforced to be smaller under flattening (compressed). These two different non-flattenable local surfaces actually encounter two different types of potential energy, which should be released in different ways. Therefore, we introduce the interior cut and the boundary cut to deal with these two different situations. A boundary cut is a cut linking a non-flattenable vertex and its nearest boundary vertex on the input surface. An interior cut is a cut only added at two neighboring edges of a non-flattenable vertex, which introduces a new interior hole on the input surface. Both operators changes the topology of an input surface.

i) For an elliptic vertex, the results of applying different types of cuts are given as follows.

Interior cut: As the incident angles are under stretched, if an interior cut is made, while the angles are restoring

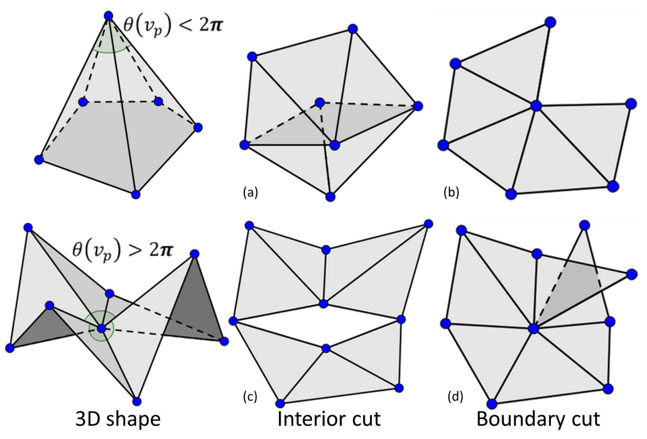

Fig. 6. Different cuts are to be added at vertices with different local shapes. (Top row) For an elliptic vertex, adding an interior cut will lead to self-intersection. Therefore, boundary cuts are usually added to resolve the non-flattenable problem. (Bottom row) The situation is reversed for a hyperbolic vertex. Boundary cuts lead to self-intersection while inserting an interior cut can resolve the problem.

to its original size, the edges will be pushed out and the adjacent faces will intersect with each other (see Fig.6a).

Boundary cut: A boundary cut can separate and extend the boundary of the input surface. Self-intersection is prevented (see Fig.6b).

ii) For a hyperbolic vertex, an inverse conclusion can be made as follows.

Interior cut: Adding an interior cut at the vertex will introduce two boundary vertices with obtuse surface angles. Therefore, the condition with $\theta(v_p) > 2\pi$ on 3D can be satisfied (see Fig.6c).

Boundary cut: A boundary cut is not suitable for the hyperbolic surface. As the local area of the vertex is larger than what can be given by a planar shape, self-intersection cannot be avoided at this vertex $\mathbf{v_p}$ with $\theta(\mathbf{v}_p) > 2\pi$ (see Fig.6d).

4.1 Interior Cut

The interior cut is only applied to a hyperbolic vertex $\mathbf{v}_p$ (i.e., $\theta(\mathbf{v}_p) > 2\pi$). When adding an interior cut, the following criteria are demanded.

1. The cut does not reach any existing boundaries.
2. The cut is as-short-as-possible.
3. The cut is as-straight-as-possible.

The first criterion is obvious; if it is not satisfied, the cut becomes a boundary cut. The second criterion comes from the observation that adding a long cut can easily lead to significant visual artifact on the part that is fabricated by self-folding structures. The last criterion relates to the effectiveness on releasing potential energy by adding a local interior cut. An interior cut actually separates the faces adjacent to $\mathbf{v}_p$ into two groups, and the potential energy is adsorbed by each of the groups. A heuristic, which has also been verified by our experiments, is that the energy distribution is well balanced if the newly created two groups have similar areas. Less stretch will be transferred to other vertices around $\mathbf{v}_p$ when an interior cut is added in this way. The algorithm for inserting an interior cut is presented as follows.

Assume there are n incident edges $\{e_1, \ldots, e_n\}$ around $\mathbf{v}_p$, we need to find a pair of edges, (e_i, e_j), that are the best candidates to fulfill the above criteria. As n is small, a simple brute-force search can be applied. First, edges with any endpoint on the boundary are eliminated from the list of incident edges as they violate the first criterion. After this step, if there are only one edge left in the candidate list, the interior cut on $\mathbf{v}_p$ is skipped. Secondly, we loop through all the incident edges in this list to construct pairs of edges. A pair of edges (e_i, e_j) is considered as a candidate pair if the angle between e_i and e_j is greater than $\frac{\pi}{2}$ at $\mathbf{v}_p$. Lastly, among all these pair of edges, the pair with the smallest total edge length is selected as an interior cut to be inserted.

4.2 Boundary Cut

For an elliptic surface M (i.e., $\theta(\mathbf{v}_p) < 2\pi$), boundary cuts are applied. Starting from the point $\mathbf{v}_p$, the cut is continuously extended until it reaches M's boundary. The shortest path from $\mathbf{v}_p$ to the boundary of M is desired. Hence, we first build an undirected graph G based on the connectivity of M, where vertices of M become vertices on G and edges on G are converted from M's edges with their edge lengths as their weights. Using $\mathbf{v}_p$ as the target and all boundary vertices on M as sources, the multi-source Dijkstra's algorithm [32] is employed to compute the one-to-many shortest path. Then, all edges on the shortest path are opened up to construct a boundary cut.

4.3 Algorithm with Hybrid Cutting

As shown in the algorithm overview (see Fig.4), cuts are added only when the shape approximation error is higher than a threshold. The approximation error is produced when a 2D pattern cannot be folded into the given 3D shape – i.e., the approximation error is directly related to the flattenability of a given surface. To efficiently reduce the approximation error, we identify the most non-flattenable points and insert an appropriate cut to make it flattenable. In addition, we know that a boundary cut is usually much longer than an interior cut. Therefore, we try to minimize the approximation error by using interior cuts first, and boundary cuts are only applied when it is necessary.

Given a threshold of the allowed approximation error (ε), the algorithm with the developed hybrid cutting strategy is given as follows:

1 Run the shape optimization and get a pair of optimized 2D/3D models $M^{(m)}$ and $D^{(m)}$, and compute the approximation error $\varepsilon(\mathbf{v}_i)$ at each vertex $\mathbf{v}_i \in M^{(m)}$ as the distance between $\mathbf{v}_i$ and its corresponding point in the input model M.

2 If $\max_{i \in V}\{\varepsilon(\mathbf{v}_i)\} \leq \varepsilon$ (the given tolerance), the model $M^{(m)}$ is good enough and the algorithm stops. Otherwise, cuts will be computed.

A1 *Routine of interior cuts only*: For all hyperbolic vertices, locate the one with maximal surface angle (i.e., $\mathbf{v}_p = \operatorname{argmax}_{\mathbf{v}_i}(\theta(\mathbf{v}_i))$).

A2 If $\theta(\mathbf{v}_p) > 2\pi + \xi$ is found, apply an interior cut to it and go back to Step 1. Otherwise, output the computed results.

B1 *Routine of boundary/interior cuts*: Locate the vertex with the largest derivation from flattenable (i.e., $\mathbf{v}_p = \operatorname{argmax}_{\mathbf{v}_i}(\theta(\mathbf{v}_i) - 2\pi)^2$).

B2 If $\theta(\mathbf{v}_p) < 2\pi - \xi$, a boundary cut is applied; otherwise, an interior cut is applied if $\theta(\mathbf{v}_p) > 2\pi + \xi$.

B3 If a cut is made, go back to Step 1; otherwise, output the computed results.

A fuzzy condition with a threshold ξ has been added when detecting non-flattenable vertices to avoid over-cutting the nearly flattenable regions. We take $\xi = \pi/16$ in all our tests. In the algorithm, two routines are used to provide a better control on the balance between the approximation error and the length of cuts (i.e., the surface quality of the final folded parts). Also, interior cuts have a higher priority than boundary cuts. For example, in the face model as shown in Fig.7a, the features of nose and mouth disappear in the shape optimization result without introducing any cuts. After applying the routine of interior cuts, we have already been able to see the profile of the nose and mouth (refer to Fig.7b). When sharper profiles are demanded on these features, the routine of boundary/interior cuts are activated to further insert boundary/interior cuts (a result can be seen in Fig.7c).

4.4 Interactive Tools

In the design process, different applications may have different requirements on the inserted cuts. Therefore, our design platform also provides interactive tools for users to specify the regions in which cuts are preferred (or prohibited). This intension from designers can be easily incorporated into the above algorithm by:

1. Removing all the vertices of the prohibited regions from the cut insertion algorithm;
2. Giving a preference weight on the vertices that are in the preferred regions;
3. Modifying the weights of edges in the preferred/prohibited regions when using Dijkstra's algorithm to compute the path of a boundary cut.

Users can also directly pick edges to insert cuts. In addition, our system provides a function to report and visualize the self-intersections on the flattened 2D patterns.

5 Implementation Details for Fabrication

After the given 3D shape is optimized and the corresponding 2D pattern has been found, it is ready to generate information for fabrication. In our previous work [4], we have modeled the elastic deformation on the hinge as large deflections of a wide beam, i.e.,

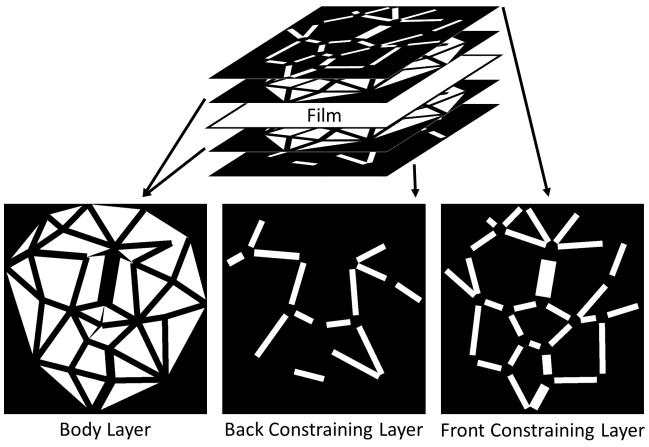

Fig. 8. The mask images generated for fabricating the face model with only interior cut shown in Fig.7. There are in total five layers, including two body layers, two constraining layers, and one layer of Shrinky-Dinks film in the center.

$$\Delta E_d = \frac{1}{2} M\alpha, \quad M = \frac{E^{\top} I}{R(1 - \nu^2)}, \tag{8}$$

where ΔE_d is the energy used in generating the deformation, M is the required moment for bending, $E^{\top}$, ν are the Flexural Modulus and the Poisson ratio of the resin material, and $I = \frac{bd^3}{12}$. In summary, as the thickness of the active layer h and the passive layer d are fixed, all variables in the formulation are constants, except the bending angle α at a hinge and its width L. Therefore, we can perform a set of experiments with different L values to calibrate its relationship with α. For more details of the formulation and experiments, please refer to [4]. Based on the experimental calibrations, we can compute L by a given α (in radian) using

$$L = (\alpha - 0.46)/(0.1\sqrt{\frac{h}{d^3}}).$$

To fabricate self-folding structures using the MIP-SL process, we need to generate mask images with grayscale values at pixels from the 2D patterns [33]. The mask image for the body layers can be generated by shrinking half distance of LW that is computed above on each edge. Since our self-folding structure is a sandwiched design, we have to print on both front and back sides of the shrinking film. The same mask images are used for the body layers at both sides. See Figs.2 and 8 for illustrations. Note that, hinges with nearly zero bending angles are considered as rigid body when generating the mask images. Another mask image to be generated is the constraining layer. When a constraint is added on one side of a hinge, the structure will be bent to the other side (refer to Fig.2 for the principle). As a result, two different mask images have to be produced at opposite sides of the film (see the right column in Fig.8).

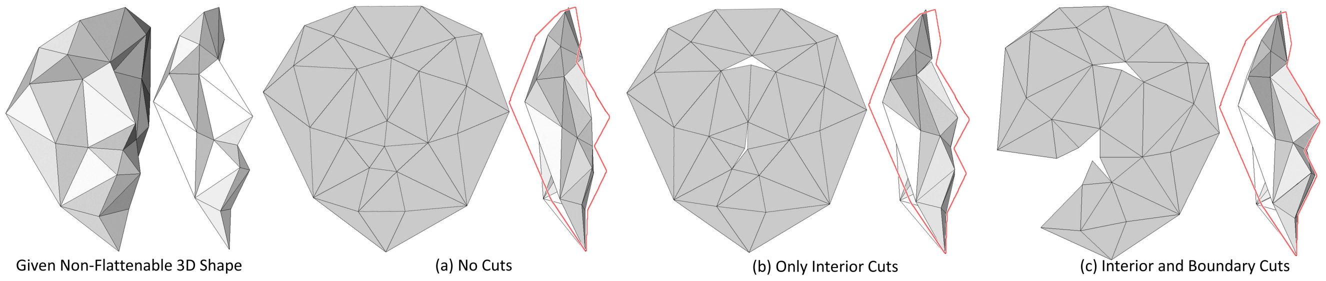
Given Non-Flattenable 3D Shape
(a) No Cuts
(b) Only Interior Cuts
(c) Interior and Boundary Cuts

Fig. 7. For a highly non-flattenable input – the face model, only applying the shape optimization (without cuts) results in a shape with large shape approximation error – the nose and the mouth disappear. The overlaid red lines are the contour of the input shape. The result can be improved by adding an interior cut. A final result with high similarity as input can be obtained by our hybrid cut insertion algorithm, in which both the interior and boundary cuts are added.

6 Experimental Results

We have implemented the presented optimization approach as a prototype software system using C++. TAUCS library [34] is employed as the numerical solver of linear equations. Different examples have been tested in our framework, and both numerical and physical tests have been performed to verify the optimization results. Running on a standard PC with Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, the statistical data on computational time is shown in Table 1. It can be found that the computation complexity is mainly based on the number of triangles of the input model. In general, the algorithm is converged within seconds. Table 1 also shows the necessary number of iterations required for a converged computation in all the examples. It can be seen from the data that as the face model is highly non-flattenable, it takes more steps to converge (i.e., 26 steps in total). In contrast, even the car model has many triangles, as it is modeled for the sheet metal fabrication, if has a better flattenability and takes only 6 steps to converge.

To compare the folded shapes with/without the presented optimization step, we measure the Hausdorff distance between the result and the given 3D shape by the publicly available software, Metro [35]. The statistics are shown in Table 2. Here, the shape approximation errors in terms of the Hausdorff distances are measured in the four different stages throughout the optimization process: (1) before optimization, (2) after flattenability-based shape optimization, (3) with only interior cuts inserted, and (4) with both interior and boundary cuts inserted. As 2D patterns are not available as the input, we take the resultant 3D shape after running the shape optimization for one step (i.e., $M^{(1)}$) as the shape before optimization. Three examples are shown in the table: a flower blade model (Fig.9), a face model (Fig.7), and a car-body model (Fig.11). In all of these examples, the shape approximation errors decrease during all the optimization steps. There is only one exception – the flower blade model, which does not have any hyperbolic vertex. Thus, no data is reported for the stage of inserting interior cuts.

Table 1. Statistics on Convergence and Computational Time

Model	Size (#tri)	#iter	Time (s)
Face	41	26	0.86
Flower	22	5	0.13
Wave	884	14	7.06
Car	758	6	2.71

The size of model is reported as the number of triangles (#tri), and #iter is the number of iteration steps until converged.

Table 2. Statistics on Shape Approx. Errors[†] in Different Stages

Input	Shape Optimization		Cut Insertion	
Model	Before	After	Interior	Both
Face	2.02 / 8.42	1.57 / 9.98	1.08 / 6.04	0.94 / 5.80
Flower	0.71 / 1.86	0.54 / 1.82	–	0.11 / 0.67
Car	1.09 / 4.33	0.25 / 2.13	0.18 / 1.26	0.13 / 1.25

[†]Hausdorff distances are measured for indicating shape approx. errors in different stages, and they are reported in the format of [average / maximum].

In addition to numerical computation, physical tests have been performed to verify the effectiveness of our approach. The tests are taken on the face model (see Fig.1), the flower blade model (see Fig.9), and a wave shape model (see Fig.10). For the blade model, we pick the 2D pattern generated by the shape optimization in the fabrication, which can be folded without any cuts. However, for the face model, the fabricated result is far away from the given shape when no cut is inserted (refer to Fig.1). Therefore, a better result can be obtained when fabricating the model from the 2D pattern by adding interior-only cuts. In Fig.1, we also show the fabricated part with artifacts that are caused by inserted cuts with long edge length. In the wave shape model, the originally designed fillets make it non-flattenable. For such an input shape, the edges become sharpened in the optimized shape, i.e., the fillets are removed with the main edges remained. The fabricated part is also shown in Fig.10. The effectiveness of our approach has been successfully demonstrated by the examples.

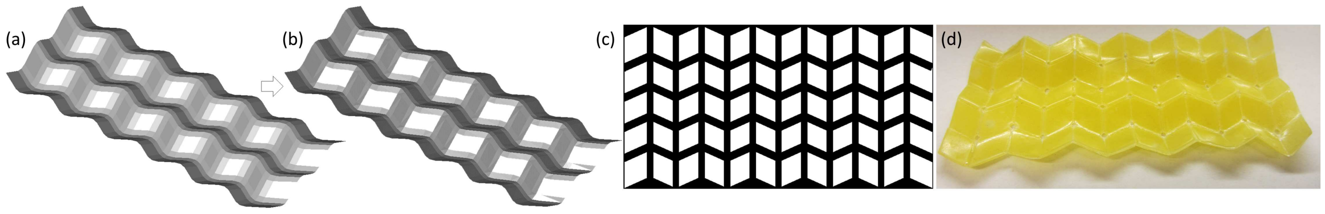

Fig. 10. An example of wave shape. (a) The fillets in the input model make the model non-flattenable. (b) The edges in the optimized shape are sharpened to improve its flattenability, and (c) the mask image that only includes the sharp edges. (d) The fabricated result.

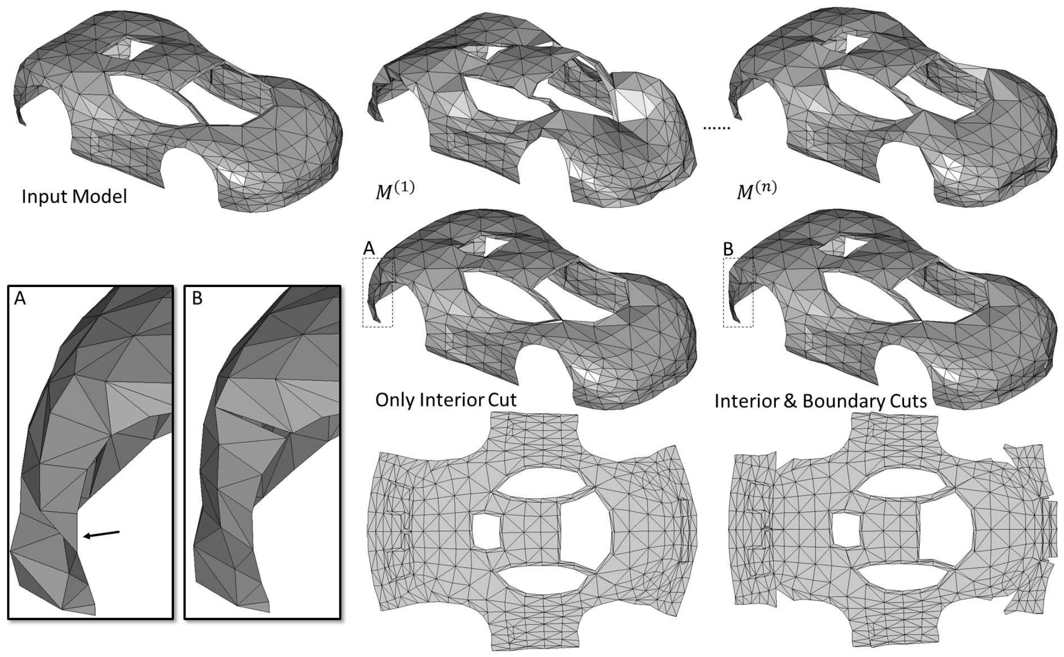

Fig. 11. An example of car-body. Given the input model (top-left) that is not flattenable, the iterations of flattenability-based shape optimization can significantly improve the 3D shape folded from 2D patterns (see the first row, from $M^{(1)}$ to $M^{(n)}$). The shape approximation error can be further improved by first inserting interior-only cuts and then adding both boundary and interior cuts (see the second row and the zoom windows). The corresponding 2D patterns are also shown at the bottom row.

7 Conclusion and Discussion

In this paper, we present a design optimization framework for using additive manufacturing to fabricate 2D Origami or Kirigami structures that can be self-folded into a target 3D shape. The challenge of designing such a self-folding structure is that only flattenable surfaces can be self-folded from a planar Origami or Kirigami structure. However, designers wish to have the flexibility of designing 3D freeform surface according to their needs. A generally designed 3D freeform surface is rarely flattenable. As a result, we propose a shape optimization framework to deform a pair of 3D/2D patches to 1) retain the flattenability and 2) minimize the shape approximation error to the given model. Not only geometric shape but also topology of an input surface are optimized in our approach to meet the design demands. An insight presented in this work is that inserting interior cuts at the local hyperbolic regions will gain more benefits. Together with boundary cuts, a hybrid algorithm has been developed to handle the cut insertion for topological optimization. Experimental results have verified the effectiveness of our framework in designing self-folding structures for 3D freeform surfaces.

Our method still has a few limitations. First, we have optimized the flattenability of a given shape by enforcing the rigidity of folding between a pair of 3D/2D mesh models. However, self-collision may occur during the self-folding process. In our future work, a collision-aware simulation for flattening and folding will be investigated by introducing the variable of time to prevent collision during the dynamic process of deformations. Second, we have simplified the input mesh before applying it to our framework to reduce the degrees of freedom for optimization. This pre-simplification strategy may introduce topological obstacle into the procedure of optimization. We plan to develop a dynamic procedure to adaptively simplify the input mesh according to the behavior during the flattening/folding simulation. Third, our cutting strategies presented in this paper try to reduce the shape approximation error without considering the structure stiffness. The FEM results will be incorporated to develop new cutting strategies in our future work. Finally, we assume there is no fabrication error in this paper, because the main focus of the paper is the design optimization for fabri-

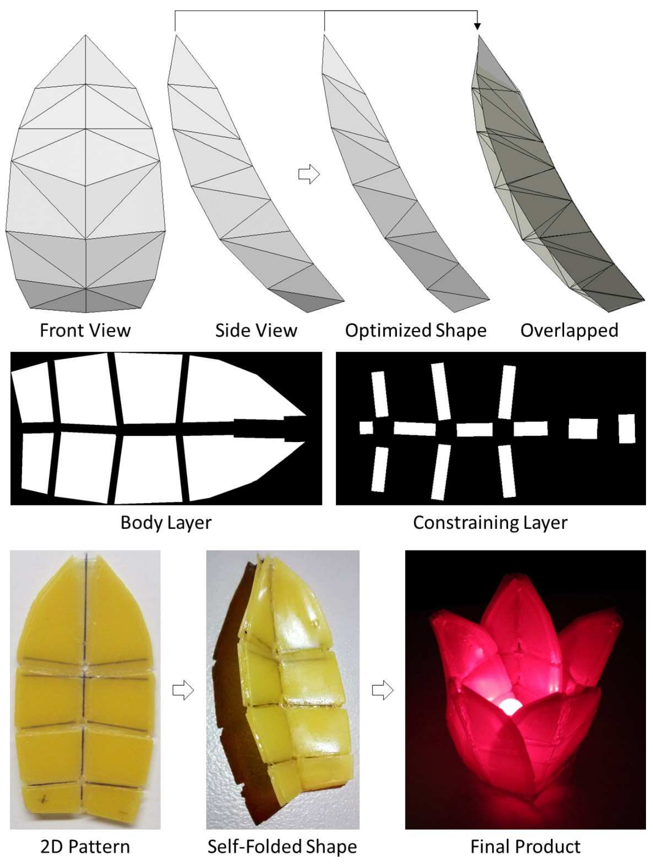

Fig. 9. An example of flower blades. (Top row) The input digital model shown in front and side view, and our framework can optimize it into a flattenable shape. (Middle row) The mask images for body and constraining layers generated by our system. (Bottom row) The 2D pattern is fabricated according to the mask images and then self-folded into the desired 3D shape after heating. Multiple flower blades are put together to get the final product – a lamp.

cation instead of the fabrication itself. To address the fabrication errors and related effects, we plan to investigate some inspection-based research by employing 3D scanners to analyze the fabrication error and to further verify the outcome of our design optimization platform in the future.

Acknowledgements

The research work reported in this paper was partially supported by NSF CMMI 1151191, and the HKSAR Research Grants Council (RGC) General Research Fund (GRF), CUHK/14207414.

References

[1] Lang, R. J., 2011. *Origami Design Secrets: Mathematical Methods for an Ancient Art*. CRC Press, Boca Raton, FL.

[2] Zhang, K., Qiu, C., and Dai, J. S., 2015. "Helical kirigami-enabled centimeter-scale worm robot with shape-memory-alloy linear actuators". *Journal of Mechanisms and Robotics*, **7**(2), p. 021014.

[3] An, B., Miyashita, S., Tolley, M. T., Aukes, D. M., Meeker, L., Demaine, E. D., Demaine, M. L., Wood, R. J., and Rus, D., 2014. "An end-to-end approach to making self-folded 3D surface shapes by uniform heating". In IEEE International Conference on Robotics and Automation (ICRA), pp. 1466–1473.

[4] Deng, D., and Chen, Y., 2015. "Origami-based self-folding structure design and fabrication using projection based stereolithography". *J. Mech. Des.*, **137**(2).

[5] Park, J.-R., Slanac, D. A., Leong, T. G., Ye, H., Nelson, D. B., and Gracias, D. H., 2008. "Reconfigurable microfluidics with metallic containers". *Journal of Microelectromechanical Systems*, **17**(2), April, pp. 265–271.

[6] Azam, A., Laflin, K. E., Jamal, M., Fernandes, R., and Gracias, D. H., 2011. "Self-folding micropatterned polymeric containers". *Biomedical Microdevices*, **13**(1), pp. 51–58.

[7] Peraza-Hernandez, E., Hartl, D., Galvan, E., and Malak, R., 2013. "Design and optimization of a shape memory alloy-based self-folding sheet". *Journal of Mechanical Design*, **135**, p. 111007.

[8] Ionov, L., 2011. "Soft microorigami: self-folding polymer films". *Soft Matter*, **7**, pp. 6786–6791.

[9] Shim, T. S., Kim, S.-H., Heo, C.-J., Jeon, H. C., and Yang, S.-M., 2012. "Controlled origami folding of hydrogel bilayers with sustained reversibility for robust microcarriers". *Angewandte Chemie International Edition*, **51**(6), pp. 1420–1423.

[10] Stoychev, G., Turcaud, S., Dunlop, J. W. C., and Ionov, L., 2013. "Hierarchical multi-step folding of polymer bilayers". *Advanced Functional Materials*, **23**(18), pp. 2295–2300.

[11] Ahmed, S., Lauff, C., Crivaro, A., McGough, K., Sheridan, R., Frecker, M., von Lockette, P., Ounaies, Z., Simpson, T., Lien, J.-M., and Strzelec, R., 2013. "Multi-field responsive origami structures: Preliminary modeling and experiments". In Proceedings of the ASME IDETC/CIECIE, August 4-7, Portland, Oregon, USA, p. V06BT07A028.

[12] Liu, Y., Boyles, J. K., Genzer, J., and Dickey, M. D., 2012. "Self-folding of polymer sheets using local light absorption". *Soft Matter*, **8**, pp. 1764–1769.

[13] Wang, M.-F., Maleki, T., and Ziaie, B., 2008. "Enhanced 3-D folding of silicon microstructures via thermal shrinkage of a composite organic/inorganic bilayer". *Microelectromechanical Systems, Journal of*, **17**(4), Aug, pp. 882–889.

[14] Ge, Q., Qi, H. J., and Dunn, M. L., 2013. "Active materials by four-dimension printing". *Applied Physics Letters*, **103**(13).

[15] Yasu, K., and Inami, M., 2012. "Popapy: Instant paper craft made up in a microwave oven". In *Advances in Computer Entertainment*, A. Nijholt, T. Romo, and D. Reidsma, eds., Vol. 7624 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 406–420.

[16] Smela, E., 2003. "Conjugated polymer actuators for biomedical applications". *Advanced Materials*, **15**(6),

pp. 481–494.

[17] Ionov, L., 2012. "Biomimetic 3d self-assembling biomicroconstructs by spontaneous deformation of thin polymer films". *J. Mater. Chem.*, **22**, pp. 19366–19375.

[18] Tibbits, S., 2014. "4D printing: Multi-material shape change". *Architectural Design*, **84**(1), pp. 116–121.

[19] Raviv, D., Zhao, W., McKnelly, C., Papadopoulou, A., Kadambi, A., Shi, B., Hirsch, S., Dikovsky, D., Zyracki, M., Olguin, C., Raskar, R., and Tibbits, S., 14. "Active printed materials for complex self-evolving deformations". *Sci. Rep.*, **4**.

[20] Wang, C. C. L., Smith, S., and Yuen, M. M. F., 2002. "Surface flattening based on energy model". *Computer-Aided Design*, **34**(11), pp. 823–833.

[21] Sander, P. V., Snyder, J., Gortler, S. J., and Hoppe, H., 2001. "Texture mapping progressive meshes". In SIGGRAPH: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, pp. 409–416.

[22] Kilian, M., Flöry, S., Chen, Z., Mitra, N. J., Sheffer, A., and Pottmann, H., 2008. "Curved folding". *ACM Transactions on Graphics*, **27**(3), pp. #75, 1–9.

[23] Sheffer, A., 2002. "Spanning tree seams for reducing parameterization distortion of triangulated surfaces". In Proceedings of Shape Modeling International, pp. 61–68.

[24] Wang, C. C. L., Wang, Y., Tang, K., and Yuen, M. M. F., 2004. "Reduce the stretch in surface flattening by finding cutting paths to the surface boundary". *Computer-Aided Design*, **36**(8), pp. 665–677.

[25] Wang, C. C. L., 2008. "Towards flattenable mesh surfaces". *Computer-Aided Design*, **40**(1), pp. 109–122.

[26] Decaudin, P., Julius, D., Wither, J., Boissieux, L., Sheffer, A., and Cani, M.-P., 2006. "Virtual garments: A fully geometric approach for clothing design". *Computer Graphics Forum (Eurographics'06 proc.)*, **25**(3), sep.

[27] Liu, Y., Pottmann, H., Wallner, J., Yang, Y.-L., and Wang, W., 2006. "Geometric modeling with conical meshes and developable surfaces". *ACM Trans. Graph.*, **25**(3), July, pp. 681–689.

[28] Zhou, C., Chen, Y., Yang, Z., and Khoshnevis, B., 2013. "Digital material fabrication using mask-image-projection-based stereolithography". *Rapid Prototyping Journal*, **19**(3), pp. 153–165.

[29] do Carmo, M. P., 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.

[30] Liu, L., Zhang, L., Xu, Y., Gotsman, C., and Gortler, S. J., 2008. "A local/global approach to mesh parameterization". In Proceedings of the Symposium on Geometry Processing, SGP '08, pp. 1495–1504.

[31] Lévy, B., Petitjean, S., Ray, N., and Maillot, J., 2002. "Least squares conformal maps for automatic texture atlas generation". *ACM Trans. Graph.*, **21**(3), July, pp. 362–371.

[32] Dijkstra, E., 1959. "A note on two problems in connexion with graphs". *Numerische Mathematik*, **1**(1), pp. 269–271.

[33] Zhou, C., Chen, Y., and Waltz, R. A., 2009. "Optimized mask image projection for solid freeform fabrication". *Journal of Manufacturing Science and Engineering*, **131**(6), p. 061004.

[34] TAUCS: A library of sparse linear solver. http://www.tau.ac.il/ stoledo/taucs/.

[35] Cignoni, P., Rocchini, C., and Scopigno, R., 1998. "Metro: Measuring error on simplified surfaces". *Computer Graphics Forum*, **17**(2), pp. 167–174.