Surface and Contour-Preserving Origamic Architecture Paper Pop-Ups

Sang N. Le, Su-Jun Leow, Tuong-Vu Le-Nguyen, Conrado Ruiz Jr., and Kok-Lim Low

Abstract—Origamic architecture (OA) is a form of papercraft that involves cutting and folding a single sheet of paper to produce a 3D pop-up, and is commonly used to depict architectural structures. Because of the strict geometric and physical constraints, OA design requires considerable skill and effort. In this paper, we present a method to automatically generate an OA design that closely depicts an input 3D model. Our algorithm is guided by a novel set of geometric conditions to guarantee the foldability and stability of the generated pop-ups. The generality of the conditions allows our algorithm to generate valid pop-up structures that are previously not accounted for by other algorithms. Our method takes a novel image-domain approach to convert the input model to an OA design. It performs surface segmentation of the input model in the image domain, and carefully represents each surface with a set of parallel patches. Patches are then modified to make the entire structure foldable and stable. Visual and quantitative comparisons of results have shown our algorithm to be significantly better than the existing methods in the preservation of contours, surfaces, and volume. The designs have also been shown to more closely resemble those created by real artists.

Index Terms—computer art, papercraft, paper architecture, surface segmentation, shape abstraction, pop-up foldability, pop-up stability

1 Introduction

Paper pop-up books have long fascinated people of all ages. Historically, "movable books" had been created for scientific and historical illustration. Recently, pop-ups have employed creative mechanisms to convey stories or even portray art. Some notable works include Carter's series of Dot books [1], Alice's Adventures in Wonderland [2], and ABC3D [3]. In addition, pop-ups also have practical scientific applications. In microelectromechanics, pop-up techniques could be used to transform 2D patterns into 3D surfaces for fabricating microstructures [4], [5].

Origamic Architecture (OA) is an intriguing type of paper pop-ups developed by Masahiro Chatani in the early 1980s. Each OA pop-up uses only a single piece of paper, and is constructed by only cutting and folding without the need for gluing. These strict geometric constraints make it highly challenging for ones to manually design paper layouts for nontrivial OA pop-ups. However, artists have been able to create interesting and intricate 3D structures, such as the two examples shown in the leftmost column of Fig. 2.

Essentially, a "valid" OA pop-up must be *stable* and *foldable*. This means that the structure must not only pop up when opened, but also fold completely flat when closed, and all this would require only holding and moving the outermost area on each half of the paper.

Unlike other types of papercraft, such as *origami*, studies of the mathematical and computational aspects of OA pop-ups are relatively scarce. There is considerable literature on how to manually design OA pop-ups [8], [9], [10], [11], and there have also been a number of works that developed computer-aided tools to assist users in the design process [12], [13], [14], [15]. Nonetheless, these tools require the user to manually position the individual patches, which can still be very labor-intensive and skill-demanding. Recently, Li et al. [7] proposed a completely automatic method for generating OA designs from 3D models. However, the class of pop-up geometry allowed by their algorithm is quite limited, which we have observed to be clearly insufficient to approximate some common shapes. Their algorithm also produces inaccurate patch contours due to the voxel discretization. Using a higher resolution 3D grid may mitigate the problem, but it incurs large memory requirement, and often results in pop-ups with an excessive amount of cuts and folds. This makes the OA designs impractical for construction into real pop-ups.

In this paper, we present a method to automatically generate an OA design that closely depicts an input 3D model. We propose a new set of geometric conditions for the foldability and stability of OA designs. The generality of the conditions allows valid pop-up structures that are previously not accounted for by other algorithms. This formulation serves as the foundation of our automatic algorithm for producing valid OA pop-ups. Our algorithm takes a novel image-domain approach to convert the input model to an OA design. The result is a 2D layout (called an *OA plan*) that is marked with lines and curves to indicate where to cut and fold to construct the pop-up. The steps of our algorithm are summarized in Fig. 3.

Some of our results are shown in Figs. 1 and 2 (middle column). In particular, in the OA plans in Fig. 2, the red and green lines indicate folds, and black lines indicate cuts. Also in Fig. 2, one can compare our results with the designs from artists [6], and with the results from the OA

• *The authors are with the National University of Singapore, School of Computing, Computing 1, 13 Computing Drive, Singapore 117417. E-mail: lnsang@comp.nus.edu.sg, {sujun1984, tuongvu}@gmail.com, {conrado, lowkl}@comp.nus.edu.sg.*

Manuscript received 2 Aug. 2012; revised 11 May 2013; accepted 4 July 2013; published online 2 Aug. 2013.
Recommended for acceptance by S. Takahashi.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-08-0151.
Digital Object Identifier no. 10.1109/TVCG.2013.108.

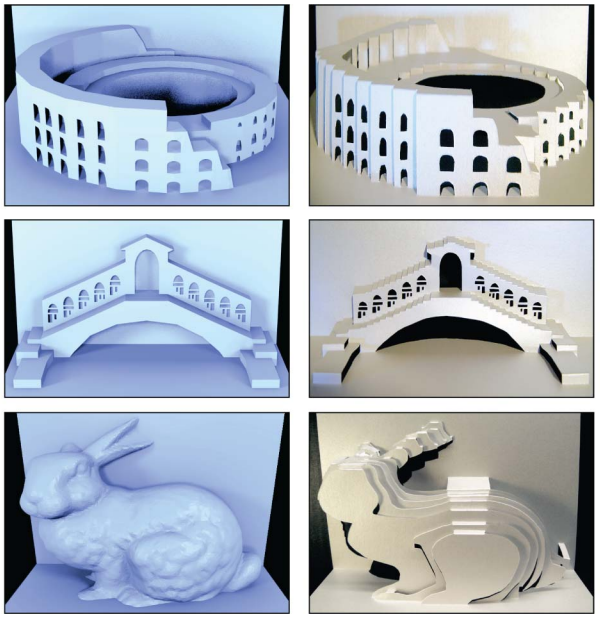

Fig. 1. Hand-made paper pop-ups (right) constructed from the OA designs produced by our algorithm given the input 3D models (left).

tool by Li et al. [7]. Note that all results (including the artists' designs) are based on the same input 3D models shown in the left column of Fig. 1. In general, our method is better than [7] in preserving the contours, surfaces, and volume of the input models, and the resulting pop-ups often require significantly fewer cuts and folds.

The main contributions of this work are as follows:

1. We present a set of more general geometric conditions for the foldability and stability of OA designs. These conditions cover a wide range of pop-up geometry, allowing our automatic OA design algorithm to closely approximate more shapes than the existing methods.
2. We propose a novel algorithm for creating OA designs using an image-domain approach that is better in preserving the contours, surfaces, and volume of the input models.
3. We propose an effective algorithm for checking and fixing the stability of any arbitrary foldable OA. This stabilization algorithm can be easily embedded in other OA design systems.

2 Related Work

Besides paper pop-ups, other types of papercraft have also been studied in the fields of mathematics and computing. For example, *kirigami*, the Japanese art of cutting and fastening pieces of paper, has been the subject of a number of studies. Several interactive methods for designing 3D objects from pieces of paper have been proposed by [16], [17], [18]. Similarly, algorithms for *Chinese paper-cutting*, an art form that is often used for designing decorative patterns and figures, have been presented by [19], [20].

Origami, the Japanese art of folding, is another well-studied papercraft technique. Notable recent books on the mathematical formulations of origami are [21] and [22]. Recently, an interactive system for origami design from polyhedral surfaces has been presented in [23]. The formulations in origami and OA differ significantly and cannot be easily ported to each other. In particular, OA allows cutting and its folding mechanisms are much more restricted than those in origami.

On paper pop-ups, Glassner [13], [24] described the use of simple geometry to create various pop-up mechanisms. Hendrix and Eisenberg [14] designed a computer application called "Pop-up Workshop" to introduce children to the

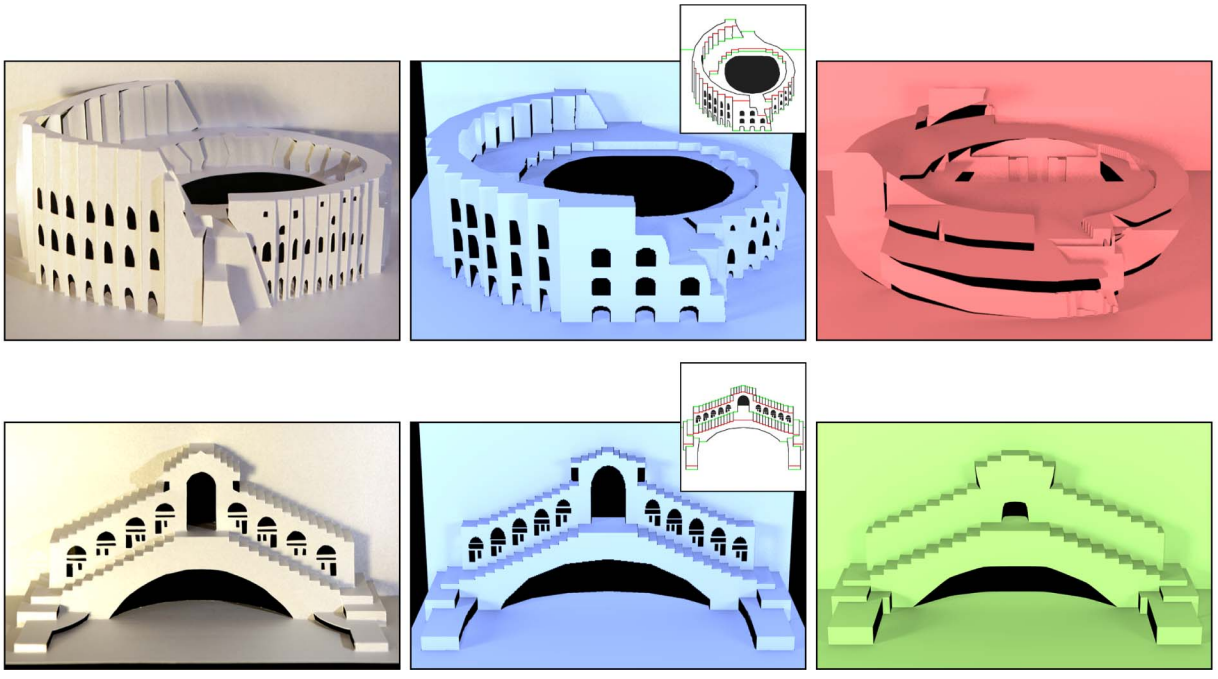

Fig. 2. (Left) OA pop-ups of the Colosseum and the Rialto bridge designed by artists [6], (middle) by our system, and (right) by the method of Li et al. [7].

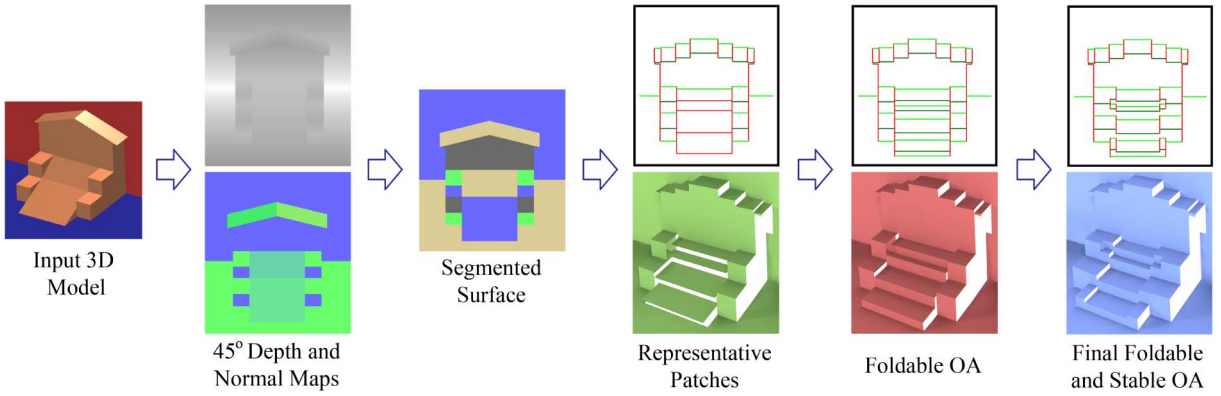

Fig. 3. Steps in our automatic OA design algorithm.

crafting and engineering discipline of paper pop-ups. Iizuka et al. [25] presented an interactive system that detects collisions and protrusions for v-folds and parallel folds. Lee et al. [12] also developed a model for simulating the opening and closing of parallel v-folds. Recently, Abel et al. [26] proposed a polynomial-time algorithm that creates pop-ups by subdividing a polygon into single-degree-of-freedom linkage structures.

Only a few studies have focused on origamic architecture in particular. The pioneering work in OA came from Mitani and Suzuki [15], who created a computer application that allows users to construct OA models by positioning and designing the horizontal and vertical faces. Although it can check for the validity of the pop-up designs in a number of cases, it does not work with cases that have dangling parts. Other similar systems include [27] and [28]. All these approaches for OA design require heavy user interactions and the users need to have adequate knowledge of the OA geometric constraints.

The most notable recent work on OA was from Li et al. [7]. They are the first to have an algorithm that fully automates the design of OA pop-ups to resemble the input 3D models. However, as mentioned previously in this paper, their method has some major shortcomings, which we aim to address in our current work. In their later work [29], Li et al. extended the same notion of validity to a more general class of v-style pop-ups.

Our formulation of the new OA geometric conditions is very much inspired by that of Li et al. [7]. However, our algorithm is inspired by the work of slice-based shape abstraction [30]. Perception and vision research has strongly emphasized the impact of contours created by slicing smooth surfaces in the visualization of 3D models [31], [32]. These contours have been shown to be effective in capturing the important shape information of the objects [33], [34]. They may even appear more appealing than the original models, which can be visually cluttered [35]. In the same spirit, our algorithm explicitly slices each smooth surface of the input model into a set of parallel patches to capture the contours.

3 OA Formulation

An OA pop-up is created from a single rectangular sheet of paper by cutting and folding. More specifically, the sheet of paper is divided into a number of nonoverlapping regions called *patches*, whose boundaries are marked with cut lines and straight fold lines. Among these patches, the *back patch* p_B and the *floor patch* p_F are two special outermost patches that share a *central fold line* (see Fig. 4a). We call this paper layout an *OA plan*.

To aid our subsequent presentation, we position the OA pop-up in a right-handed *orthogonal* coordinate system, as shown in Figs. 4c and 4d, in which the x-axis is parallel to the central fold line, and the z-axis is perpendicular to the x-axis and parallel to the floor patch. We also call the angle between p_F and p_B the *opening angle* θ. When $\theta = 180°$, the OA is said to be *fully opened*; when $180° > \theta > \epsilon$ for an arbitrary small positive angle ϵ, the OA is in a pop-up state; and when $\theta \leq \epsilon$, the OA is said to be *fully closed* or *folded flat*. Note that the opening angle cannot be exactly $0°$ because the patches are not allowed to overlap.

In this work, we focus only on the design of a common type of OA called *parallel OA*, in which all the fold lines must be parallel to the central fold line. This is also the type of OA addressed by Li et al. in [7].

3.1 OA Plans

To formally define an OA plan, we first define the followings:

Definition 1. *Two patches $p_a \neq p_b$ are said to be* adjacent *if and only if they share a common fold line.*

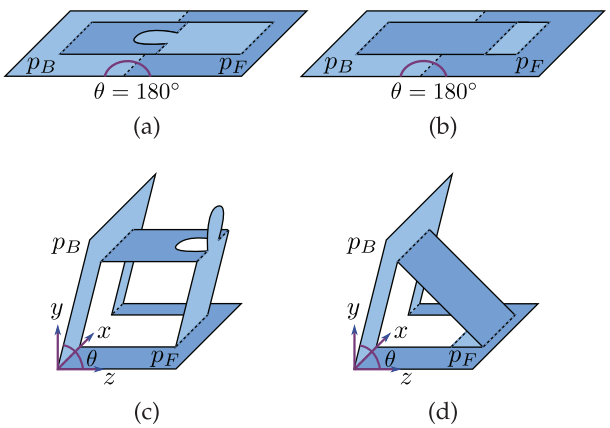

Fig. 4. (a) An OA plan with cuts (solid lines) and folds (dashed lines), (c) which can pop up at any arbitrary angle. (b) A nonfoldable OA plan, (d) which is stuck during folding.

Definition 2. *Let* $\mathbb{P} = \{p_0 = p_a, p_1, \ldots, p_{n-2}, p_{n-1} = p_b\}$ *be an ordered set of* n *distinct patches. If* p_i, p_{i+1} *are adjacent for all* $0 \leq i \leq n-2$*, we say that* p_a *and* p_b *are* connected *and* $\mathbb{P}$ *is an* n-path *from* p_a *to* p_b*.*

We now define an OA plan as

Definition 3. *An* OA plan *is a set of patches, where*

1. *all patches are coplanar and form a rectangular domain with possible holes.*
2. *they are nonintersecting, except at their boundaries.*
3. *for every patch* p*, there exists a path from* p_B *to* p_F *that contains* p*.*

The first two properties are universal for general origamic architecture layouts [7], [15]. However, such general layouts may contain "floating" patches, which are not adjacent to any other patch, or "dangling" patches, which are not connected to p_B and p_F. Hence, Property 3 is defined to keep all the patches connected to both p_B and p_F.

3.2 Foldable OA Plans

For our OA formulation, we make the assumption that the paper has zero thickness, and each patch is rigid. A fold line between two adjacent patches acts like a hinge that allows the patches to freely rotate about each other.

An OA plan, as defined above does not guarantee that the OA can be folded from a fully opened state to a fully closed state without violating the patch rigidity assumption. Figs. 4b and 4d show such an OA plan, which is stuck during its folding process. A valid OA plan should be foldable from $\theta = 180^\circ$ to $\theta = \epsilon$ without bending the patches and affecting the pairwise adjacency and nonintersection of the patches. If so, we say the OA plan is *foldable*.

At each opening angle, a possible set of relative positions of the patches, with respect to p_F and p_B, is called a *configuration*. When all the patches are parallel to p_F or p_B, we have a *parallel configuration*. By considering this special configuration, we have the condition for a foldable OA plan as follows:

Proposition 1. *An OA plan is* foldable *if and only if this plan is the projection of a parallel configuration along vector* $\mathbf{w}$ *onto the* xz*-plane, where* $\mathbf{w}$ *is perpendicular to the* x*-axis and bisecting the corresponding opening angle.*

The proof of the sufficiency of Proposition 1 can follow the one given in [7]. Its necessity can be proved trivially: as no patch is floating or dangling, the opening of an OA from the considered parallel configuration to 180° is equivalent to projecting the patches along $\mathbf{w}$ onto the OA plan.

Proposition 1 is used in our automatic OA design algorithm to guarantee that when the representative patches are modified, the resulting OA is foldable (see Section 4.3). In addition, its necessity condition provides an easy way to detect a nonfoldable OA.

3.3 Stable OA Plans

In practice, an OA can be fully opened, popped up, and fully closed simply by turning only the back and floor patches, without the need to apply any external forces to the other patches. If the back and floor patches are held stationary, other patches must also remain stationary. In other words, the OA needs to be *stable*. In this section, we present a new stability condition for parallel OAs. We first define the following:

Definition 4. *A* fold line *is said to be* stable *if it is not movable at each opening angle of the OA. A* patch *is said to be* stable *if at least two noncollinear fold lines on it are stable. An OA* plan *is said to be stable if all of its patches are stable.*

Existing studies have only defined a narrow set of conditions for the stability of parallel OA. Specifically, a patch is considered stable if it is the back or floor patch, or it lies in a 3-path or 4-path that connects two stable patches [7], [26]. These conditions limit the set of possible valid patch arrangements and could seriously handicap the preservation of shapes and volume, as demonstrated in Section 5.

To facilitate our discussion, we specifically define the parallel configuration at $\theta = 90^\circ$ the *orthogonal configuration*, because every pair of adjacent patches are orthogonal to each other. The orthogonal configuration is significant because the input model is approximated at this opening angle.

We consider an OA plan initially constructed from the projection of an orthogonal configuration in the $(0, -1, -1)$ direction onto the xz-plane. This OA plan is foldable (by Proposition 1), and we further formulate the conditions that make it stable.

Let $\mathbb{P} = \{p_0 = p_B, p_1, \ldots, p_n, p_{n+1} = p_F\}$ be a path connecting the back and floor patches. Along $\mathbb{P}$, we mark p_{2k} as *even* and p_{2k+1} as *odd* patches, where $0 \leq k \leq \lfloor n/2 \rfloor$. In a parallel configuration, the even and odd patches are, respectively, parallel to p_B and p_F. As a result, if a patch is even (odd) in one path, it is also even (odd) in all other paths from p_B to p_F. We further define two special sets of patches.

Definition 5. *The* B*-set (*F*-set) is the set of patches that are always parallel to* p_B *(*p_F*) at any opening angle.*

It is clear that if an even (odd) patch is stable, it will be in the B-set (F-set). We aim to make every patch belong to either of these two sets. To achieve that, we define a new type of connection.

Definition 6. *Two patches* p_1 *and* p_2 *are said to be doubly connected if there exist noncoplanar patches* q_1 *and* q_2 *such that* p_i *and* q_j *are adjacent for all* $i, j \in \{1, 2\}$*. Patches* q_1 *and* q_2 *are also doubly connected and the structure* $\{p_1, q_1, p_2, q_2\}$ *is called a* double connection*.*

Corollary 1. *Consider two doubly connected patches* p_1 *and* p_2*. If* p_1 *is in* B*-set (*F*-set), then* p_2 *must also be in the same set.*

Proof. Recall that the OA is temporarily in a parallel configuration. The double connection between p_1 and p_2 immediately makes them always parallel to each other. Hence, both belong to B-set (F-set). □

Corollary 2. *In an* n*-path* $\mathbb{P}$ *from* p_B *to* p_F*, if*

1. *one of its even (odd) patches is in* B*-set (*F*-set), and*
2. $\lfloor n/2 \rfloor$ *pairs of even (odd) patches are doubly connected,*

then all the even (odd) patches in $\mathbb{P}$ *belong to* B*-set (*F*-set). We call* $\mathbb{P}$ *a* B*-path (*F-path*).*

Proof. Path $\mathbb{P}$ consists of patches alternately parallel to p_B and p_F. Thus, if it contains n patches, the number of even

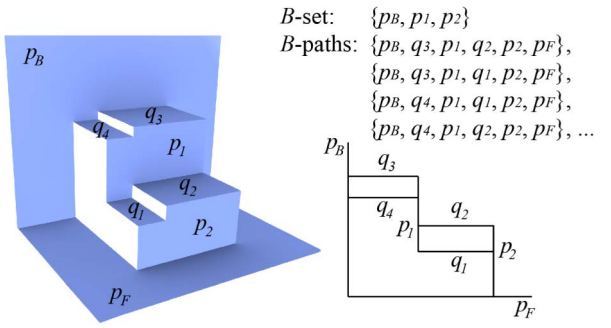

Fig. 5. (Left) An OA containing two double connections, and (right) its side view.

patches and the number of odd patches are both at most $\lfloor n/2 \rfloor + 1$. From this observation and Corollary 1, it is clear that Corollary 2 is also true. □

Fig. 5 illustrates a B-set and a set of B-paths that are constructed from double connections. Note that the even (odd) patches in a B-path (F-path) are always parallel to p_B (p_F). Hence, we can easily "stabilize" that path by making its odd (even) patches also parallel to p_F (p_B). To do so, we utilize *monotonic paths* and *near-monotonic paths*, which are defined as follows:

Definition 7. *A B-path (F-path) is said to be* monotonic *if the perpendicular distances from its even (odd) patches to p_B (p_F) form a monotonic sequence.*

Fig. 6a illustrates an OA with a monotonic path, which can be used to approximate staircase-like models.

Definition 8. *A B-path $\mathbb{P} = \{p_0, p_1, \ldots, p_{2k+1}\}$ is said to be* near-monotonic *if $d_0, d_2, \ldots, d_{2k}$, the perpendicular distances from its even patches to p_B, satisfy*

1. $d_0 < d_{2k} < d_{2k-2} < \cdots < d_4 < d_2$, *or*
2. $d_0 > d_{2k} > d_{2k-2} > \cdots > d_4 > d_2$.

Similarly, an F-path $\mathbb{P} = \{p_0, p_1, \ldots, p_{2k+1}\}$ is said to be near-monotonic if $d_1, d_3, \ldots, d_{2k+1}$, the perpendicular distances from its odd patches to p_F, satisfy

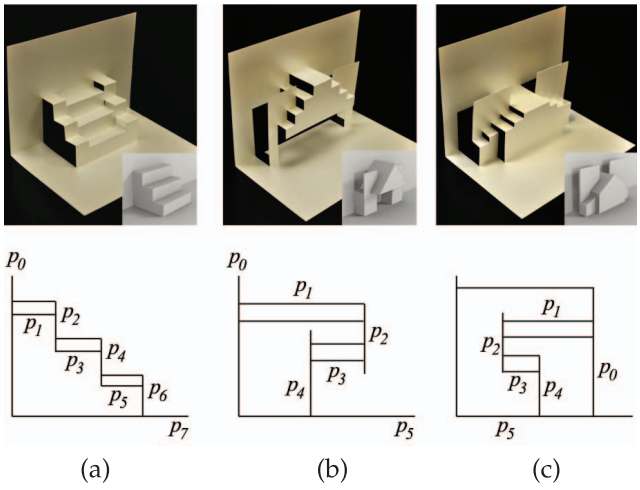

Fig. 6. (Top) (a) An OA that contains a monotonic path, and (b, c) the two cases of near-monotonic B-paths. (Bottom) Side views of the OAs.

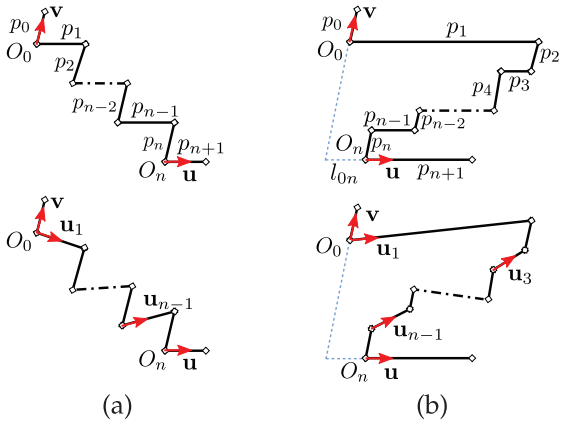

Fig. 7. (a) Side views of a monotonic and (b) a near-monotonic B-paths connecting stable patches p_0 and p_{n+1}. At each opening angle, the parallel configuration (above) is shown to be the only possible configuration. The nonparallel ones (below) are impossible.

1. $d_1 < d_{2k+1} < d_{2k-1} < \cdots < d_5 < d_3$, *or*
2. $d_1 > d_{2k+1} > d_{2k-1} > \cdots > d_5 > d_3$.

Figs. 6b and 6c illustrate respectively the first and second cases of near-monotonic B-paths. Near-monotonic paths are important for preserving concave shapes in the input models.

We now present the conditions for stable patches.

Proposition 2. *If a path connects two stable patches and is monotonic or near-monotonic, then all of its patches are stable.*

Proof. Here we prove Proposition 2 for B-paths, as F-paths can be proved similarly. Fig. 7 illustrates the (a) monotonic and (b) near-monotonic paths as seen from the side view. In this view, the patches are represented as line segments.

Consider a B-path $\mathbb{P} = \{p_0, p_1, \ldots, p_n, p_{n+1}\}$ connecting stable patches p_0 and p_{n+1}. Without loss of generality, we assume that the first and last even patches in $\mathbb{P}$ are p_0 and p_n. At each opening angle, there exists a parallel configuration of $\mathbb{P}$, since the OA plan is created from the $(0, -1, -1)$ projection of the orthogonal configuration. We show that this parallel configuration is also the only possible configuration, which leads to the conclusion that the patches are stable.

Let l_k be p_k's length and O_k be the fold between p_k and p_{k+1}. We also denote $\mathbf{u}$ ($\mathbf{v}$) as the unit vector parallel to p_{n+1} (p_0) and pointing away from O_n (O_0). In a general configuration, let $\mathbf{u}_{2i+1}$ be the unit vector parallel to p_{2i+1}, pointing away from O_{2i}. As $\mathbb{P}$ is a B-path, p_{2i} are always parallel to $\mathbf{v}$.

We consider the two types of paths.

Monotonic. If $\mathbb{P}$ is monotonic, then $d_0 < d_2 < \cdots < d_n$ (Fig. 7a). For the parallel configuration, we have

$$O_n = O_0 + \sum_{i=0}^{\lfloor n/2 \rfloor - 1} (l_{2i+1}\mathbf{u}) - \sum_{i=1}^{\lfloor n/2 \rfloor} (l_{2i}\mathbf{v}).$$

For a general configuration, we have

$$O_n = O_0 + \sum_{i=0}^{\lfloor n/2 \rfloor - 1} (l_{2i+1}\mathbf{u}_{2i+1}) - \sum_{i=1}^{\lfloor n/2 \rfloor} (l_{2i}\mathbf{v}).$$

Equating the RHS of the above two equations leads to

$$\sum_{i=0}^{\lfloor n/2\rfloor-1}(l_{2i+1}\mathbf{u}_{2i+1}) = \sum_{i=0}^{\lfloor n/2\rfloor-1}(l_{2i+1}\mathbf{u}).$$

This equality occurs if and only if $\mathbf{u}_{2i+1} = \mathbf{u}$ for all $0 \le i \le \lfloor n/2\rfloor - 1$, which means all the odd patches are parallel to p_{n+1}. As $\mathbb{P}$ forms a parallel configuration and p_0, p_{n+1} are stable, all its other patches are also stable.

Near-Monotonic. If $\mathbb{P}$ is near-monotonic, without loss of generality, we assume the case where $d_2 > d_4 > \cdots > d_{n-2} > d_n > d_0$ (Fig. 7b). For the parallel configuration, we have

$$\begin{aligned} O_1 &= O_0 + l_1\mathbf{u} \\ &= O_n + \sum_{i=1}^{\lfloor n/2\rfloor-1}(l_{2i+1}\mathbf{u}) + \sum_{i=1}^{\lfloor n/2\rfloor}(l_{2i}\mathbf{v}). \end{aligned} \tag{1}$$

In addition,

$$l_1 = l_{0n} + \sum_{i=1}^{\lfloor n/2\rfloor-1} l_{2i+1}, \tag{2}$$

where l_{0n} is the difference between O_0 and O_ns coordinates along the $\mathbf{u}$-axis.

Equations (1) and (2) lead to

$$O_0 = O_n + \sum_{i=1}^{\lfloor n/2\rfloor}(l_{2i}\mathbf{v}) - l_{0n}\mathbf{u}. \tag{3}$$

For a general configuration, we have

$$\begin{aligned} O_1 &= O_0 + l_1\mathbf{u}_1 \\ &= O_n + \sum_{i=1}^{\lfloor n/2\rfloor-1}(l_{2i+1}\mathbf{u}_{2k+1}) + \sum_{i=1}^{\lfloor n/2\rfloor}(l_{2i}\mathbf{v}). \end{aligned} \tag{4}$$

Substituting (2) and (3) into (4), we obtain

$$l_{0n}\mathbf{u}_1 + \sum_{i=1}^{\lfloor n/2\rfloor-1}(l_{2i+1}\mathbf{u}_1) = l_{0n}\mathbf{u} + \sum_{i=1}^{\lfloor n/2\rfloor-1}(l_{2i+1}\mathbf{u}_{2k+1}).$$

This equality occurs if and only if $\mathbf{u} = \mathbf{u}_{2k+1} = \mathbf{u}_1$ for all $1 \le i \le \lfloor n/2\rfloor - 1$, which means all the odd patches, including p_1, are parallel to p_{n+1}. Similar to the case of monotonic paths, this condition makes all the patches on $\mathbb{P}$ stable. □

Proposition 2 leads to an effective approach for OA stabilization, as presented in Section 4.4. Unlike [7], in which the OA is created to simultaneously guarantee foldability and stability, we can take as input an arbitrary foldable OA and make it stable. As a result, our stabilization technique can be embedded into other OA design systems and allows independent improvements of other steps.

4 Automatic OA Design

In this section, we present our algorithm for automatic OA design. The input is a 3D model, represented as a polygonal mesh, which is positioned by the user between the two orthogonal planes (the xy- and xz-planes). The output of our system is a foldable and stable OA plan that can be popped up into the desired structure at $90°$ opening angle.

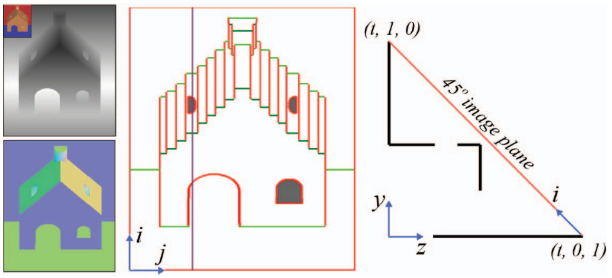

Fig. 8. A house model is being converted to an OA. (a) Its depth map $\mathbb{D}$ and normal map $\mathbb{N}$, (b) OA plan, and (c) cross section of the patches along the purple line.

We compute this OA plan by first constructing the patches in the orthogonal configuration, and then projecting them in the $(0, -1, -1)$ direction onto the xz-plane. Our method is carried out in four steps (shown in Fig. 3).

1. *Surface Segmentation.* We divide the input model into nonoverlapping, smooth surface segments. The surface segmentation facilitates the creation of patches in the next step.
2. *Generating representative patches.* A set of parallel patches is generated to estimate the curvatures and details of each surface segment.
3. *Constructing a foldable OA plan.* We connect the representative patches so that their projections along $(0, -1, -1)$ onto the xz-plane is a foldable OA plan, according to Proposition 1.
4. *Stabilizing the OA plan.* Utilizing Proposition 2, we check whether the patches are stable. If they are not, we stabilize them by constructing extra supporting patches.

Our algorithm operates on a depth map of the input model instead of directly on the polygonal mesh. To obtain the depth map, it sets up a $45°$ orthographic view, which looks at the central fold line along the projection vector $(0, -1, -1)$. The image plane is placed perpendicular to this vector so that it intersects with the xz- and xy-planes at $(t, 0, 1)$ and $(t, 1, 0)$ lines, respectively (see Fig. 8c). We define i and j as the orthonormal basis of this image plane, which are parallel to $(0, 1, -1)$ and $(1, 0, 0)$, respectively. Our OA plan can be computed completely in the $45°$ orthographic view (Figs. 8a and 8b).

From this view, we render the depth map $\mathbb{D}$ and normal map $\mathbb{N}$ of the visible input surfaces enclosed by the two orthogonal planes. The normal vectors in $\mathbb{N}$ are all scaled to unit length. The depth values in $\mathbb{D}$ are measured from the image plane and range from 0 (points on the image plane) to $\sqrt{2}/2$ (points on the central fold line). The details of the four steps are elaborated in the following sections:

4.1 Surface Segmentation

To better preserve the curvatures and boundaries of the input surfaces, we hope each surface can be represented by a separate set of parallel patches. To extract the surfaces, some mesh segmentation algorithms [36], [37] can be used,

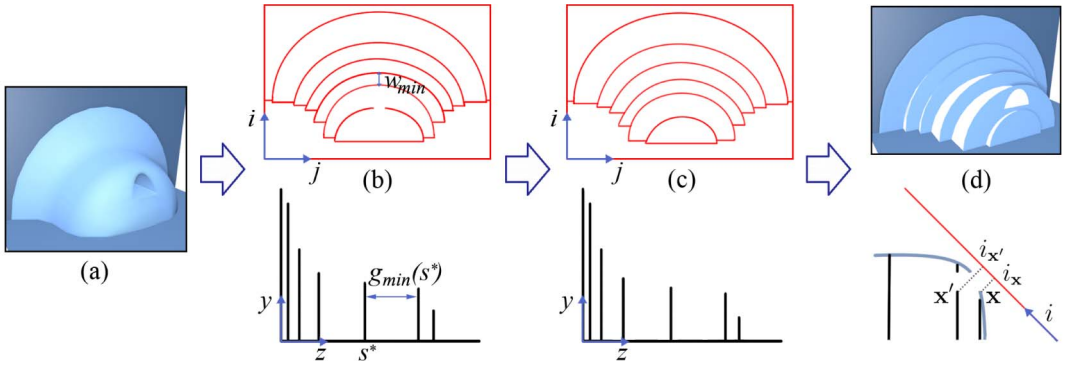

Fig. 9. (a) An input surface is to be represented by a set of vertical slices, which are (b) first positioned to satisfy the minimum patch width threshold and constant gap-gradient ratio. (c) The slice positions are then optimized to minimize the contour discontinuity, while maintaining the gap-gradient ratio. (d) Finally, the contour of the original hole is projected onto the corresponding patch. The bottom row shows the side view for each step.

but our algorithm simply uses image segmentation to partition the depth map $\mathbb{D}$ into surface segments.

Essentially, the surface segmentation works by locally fitting a quadratic surface on the segmented pixels in the neighborhood of a candidate pixel. The candidate pixel is considered to be in the same segment if it is near this quadratic surface.

Our algorithm uses the depth map $\mathbb{D}$ and normal map $\mathbb{N}$ to perform the segmentation. It determines whether each candidate pixel $\mathbf{x}$ should remain in the current segment by thresholding $f(\mathbf{x})-q(\mathbf{x})$, where $f(\mathbf{x})$ is a vector consisting of the depth value and the x-, y-, z-components of the normal vector at $\mathbf{x}$, and $q(\mathbf{x})$ is the quadratic approximation from the previously segmented pixel $\mathbf{x}_0$:

$$q(\mathbf{x}) = f(\mathbf{x}_0) + f'(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) + \frac{1}{2}f''(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0)^2.$$

The derivatives of f are estimated from the neighboring pixels in the current segment. We make pixel $\mathbf{x}$ belong to a new segment if $f(\mathbf{x})-q(\mathbf{x})$ exceeds the predefined thresholds in $\mathbb{D}$ or $\mathbb{N}$. Oversegmentation may occur due to noise and small geometric features. It is, however, tolerable because the slicing in Section 4.2 will produce patches in the same orientations if the segments have similar gradients.

4.2 Representative Patches Generation

After the input model is divided into distinct smooth segments, we want to generate a set of parallel representative patches to preserve both overall curvature and detailed contours of each segment.

To achieve that, we first determine whether the patches should be horizontal or vertical, then create them by "slicing" each segment using a number of parallel planes in the selected orientation. Each patch is the union of all the cross sections of the surface segment between two consecutive slices, and is entirely visible in the 45° orthographic view. The process of generating the representative patches consists of three main steps.

4.2.1 Determining the Slicing Orientation

We observe that a segment can be represented better by vertical patches, which are parallel to the xy-plane, if its gradient changes more significantly along the y-axis than along the z-axis. Otherwise, it should be represented by horizontal patches.

From this observation, we determine the slicing orientation by comparing $\overline{dz/di(\mathbf{x})}$ and $\overline{dy/di(\mathbf{x})}$, the average z- and y-gradients along the i-axis (see Fig. 8c) of all pixels $\mathbf{x}$ in depth map $\mathbb{D}$ that belong to the considered segment. We slice using vertical planes if $\overline{dz/di(\mathbf{x})} > \overline{dy/di(\mathbf{x})}$, and horizontal planes otherwise.

4.2.2 Positioning the Slices

Having determined the orientation for the slices, we proceed to find their positions along the y-axis (horizontal slices) or z-axis (vertical slices). They are computed according to three criteria.

Criterion 1: We aim to make the OA plan easy to cut by preventing the patches bounded by two consecutive slices from being thinner than w_{min}, which is a threshold proportional to the segment's area. We constrain the width of a patch in the 45° image space using the dilation operation in morphological image processing [38]. At each slicing position s, we estimate the minimum gap $g_{min}(s)$ such that the width of the patch bounded by two slices at s and $s+g_{min}(s)$ is at least w_{min} (see Fig. 9b).

Criterion 2: Using only Criterion 1 may result in too many slices, especially on steep surface. To control this, each slicing gap is kept proportional to the square root of the average surface gradient perpendicular to the slice. We observe that maintaining a constant ratio $r = g(s)/\sqrt{grad(s)}$ over every slice s results in a more pleasant approximation, where $g(s)$ and $grad(s)$ are the slicing gap and the average surface gradient perpendicular to s.

Criterion 3: We want to avoid slicing through holes by minimizing the total discontinuity along the slice contours. We measure contour discontinuity as the maximum distance between two contour pixels that are both adjacent to another surface segment. We denote this as $dis(s)$ for each slice s.

To achieve Criteria 1 and 2, we start from the position s^* where the surface gradient is greatest. We place two slices at s^* and $s^* + g_{min}(s^*)$, and compute the desired gap-gradient ratio, $r = g_{\min}(s^*)/\sqrt{grad(s^*)}$. We then estimate the positions of other slices such that the corresponding ratio at every slice is equal to r. These initial positions guarantee

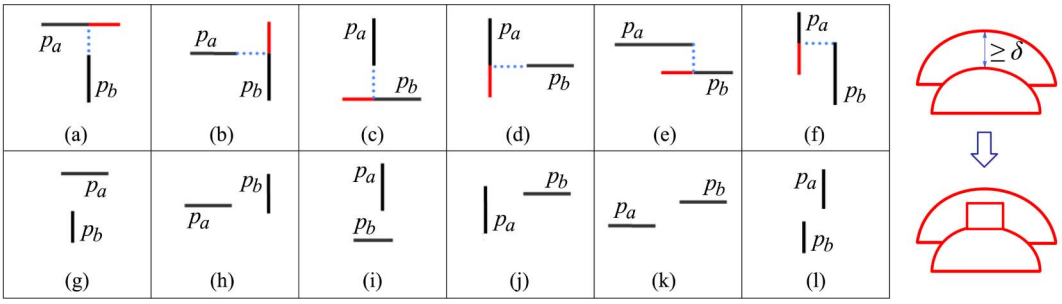

Fig. 10. All possible relative positions of p_a and p_b. The first row shows connectable patches. Red segments represent the parts on existing patches that need to be removed. Black and dashed blue segments represent the remaining parts and the new connections to be added, respectively. The second row shows nonconnectable patches. The rightmost diagram illustrates the connecting result for Case (f) in the 45° view.

that all the patches' widths are not smaller than w_{min}, and a constant ratio r is maintained over the segment (see Fig. 9b).

To optimize the slice positions for Criterion 3, we use dynamic programming to minimize the discontinuity on the slices. Let $s_1, s_2, \ldots, s_n$ be the initial slice positions that satisfy Criteria 1 and 2, and $[s_1], [s_2], \ldots, [s_n]$ be small ranges around them, which allow flexibility in slicing. We define $\mathcal{D}is(k, s)$ as the total contour discontinuity up to slice k at position $s \in [s_k]$.

We compute the minimum total discontinuity as

$$\mathcal{D}is(k, s) = \mathcal{D}is(k-1, s_{prev}(k, s)) + dis(s),$$

where $s_{prev}(k, s)$ is the position of slice $k-1$ that produces the minimum discontinuity up to that slice:

$$s_{prev}(k, s) = \arg\min_{s'}(\mathcal{D}is(k-1, s')).$$

To maintain the gap-gradient ratio, s' can only lie within the intersection $[s_{k-1}] \cap [s - r \cdot \sqrt{grad(s)}]$. The latter range is computed from $[s]$ so that the slicing gap satisfies the desired ratio r.

Eventually, after obtaining the minimum $\mathcal{D}is(n, s)$, we use s_{prev} to find the optimal positions for all the slices (see Fig. 9c).

In our slice positioning algorithm, the slices may still cut through holes in the input surfaces if they are big. In our system, the user can handle such case by using an appropriate w_{min} and interactively adjusting the affected slices based on their computed positions.

4.2.3 Projecting Surface Contours

After slicing the surface segment to create the representative patches, we reconstruct the holes and sharp corners by projecting their contours onto the patches (see Fig. 9d). In the 45° orthographic view, we project pixel $(i_{\mathbf{x}}, j_{\mathbf{x}})$ on the original surface to pixel $(i'_{\mathbf{x}}, j'_{\mathbf{x}})$ on the corresponding patch p using the transformation

$$\begin{cases} i'_{\mathbf{x}} = i_{\mathbf{x}} + (-1)^k (d(p) - d(\mathbf{x}))/\sqrt{2} \\ j'_{\mathbf{x}} = j_{\mathbf{x}} \end{cases}$$

where d is the distance measured from the xy (xz) plane and $k = 1$ ($k = 0$) if p is vertical (horizontal). Note that we only need to reconstruct the parts that are visible in the 45° orthographic view.

4.3 Foldable OA Plan Construction

The representative patches have been created as nonoverlapping continuous regions in the 45° view. Their projections along the $(0, -1, -1)$ direction onto the xz-plane satisfy Properties 1 and 2 of an OA plan. We connect the patches to fulfill Property 3.

Consider p_a and p_b, two patches sharing a nonvertical boundary in the 45° image space. Without loss of generality, we assume that p_a and p_b lie, respectively, above and below their shared boundary in this space. These two patches are adjacent if they also touch each other at their actual boundary in 3D space. If they are not adjacent, we may need to connect them to form a path from p_B to p_F. However, because all the patches need to be visible in the 45 degree view, p_a and p_b are only connectable if they satisfy

$$\begin{cases} m(y(p_b)) + \delta\sqrt{2} \leq m(y(p_a)) \\ m(z(p_a)) + \delta\sqrt{2} \leq m(z(p_b)), \end{cases}$$

where δ is the minimal width allowed for each patch in the 45° view. If p_b is vertical and p_a is horizontal, then m is the $\min$ function (Figs. 10a and 10b, otherwise m is the $\max$ function (Figs. 10c, 10d, 10e, and 10f).

If two connectable patches are not parallel, we create a new connection by extending one of them in the plane of the patch (Figs. 10a, 10b, 10c, and 10d). If they are both vertical (Fig. 10f), their orthogonal connecting patch is created to replace the bottom portion of p_a. We make the connecting patch as wide as possible such that the new fold line that replaces the top boundary of p_b is still a good approximation (within a predefined gap threshold) of the original boundary. The case when both p_a and p_b are horizontal (Fig. 10e) are handled analogously.

We also define the *connecting cost* between p_a and p_b as the total length of the parts added and removed when connecting p_a and p_b. This cost is zero if they are adjacent.

Given the connectability of patches, we keep constructing the least-cost paths from p_B to p_F until no more patch can be added. Each path is evaluated based on the total connecting cost. If a patch cannot be added to any path from p_B to p_F, we merge it into the nearest parallel patch that shares with it a nonvertical boundary in the 45° view. If it cannot be merged, we simply discard it.

The connecting-merging process may not preserve the input surfaces well. In Fig. 11, Patch 1 is merged into Patch 2, which fills up the original alcove. However, the resulting patches now satisfy both Definition 3 and Proposition 1, hence, their projection along $(0, -1, -1)$ on the xz-plane forms a foldable OA plan.

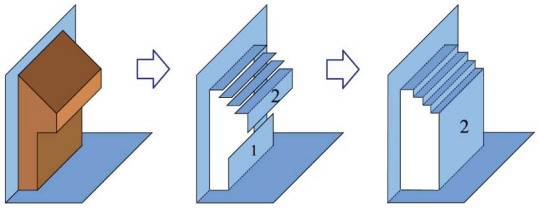

Fig. 11. A 3D model is approximated by a set of representative patches, which then go through the connecting-merging process to form a foldable OA.

4.4 OA Plan Stabilization

To make the OA plan stable, we repeatedly stabilize the paths of patches in three steps: 1) sorting the paths, 2) stabilizing the best one, and 3) updating them. We divide the paths so that each of them starts and ends with stable patches, but does not visit any other stable ones.

4.4.1 Sorting the Paths

First, we need to determine a metric for path comparison. If an n-path $\mathbb{P}$ contains n_B pairs of doubly connected even patches that belong to the B-set, we can make the other $n_B^* = \lfloor n/2 \rfloor - n_B$ even pairs doubly connected by adding n_B^* odd patches to them. According to Corollary 2, $\mathbb{P}$ will then become a B-path. Similarly, we denote n_F^* as the number of even patches we can add to make $\mathbb{P}$ an F-path.

These values give us $n_{\mathbb{P}}^* = \min(n_B^*, n_F^*)$, a measurement for path comparison. We say that path $\mathbb{P}_1$ is better than path $\mathbb{P}_2$ if one of the following conditions is true:

1. $n_{\mathbb{P}1}^* < n_{\mathbb{P}2}^*$;
2. $n_{\mathbb{P}1}^* = n_{\mathbb{P}2}^*$, and $\mathbb{P}_1$ is monotonic but not $\mathbb{P}_2$;
3. $n_{\mathbb{P}1}^* = n_{\mathbb{P}2}^*$, neither $\mathbb{P}_1$ nor $\mathbb{P}_2$ is monotonic, and $\mathbb{P}_1$ is near-monotonic but not $\mathbb{P}_2$;
4. $n_{\mathbb{P}1}^* = n_{\mathbb{P}2}^*$, $\mathbb{P}_1$s and $\mathbb{P}_2$s monotonicity and near-monotonicity are the same, but $\mathbb{P}_1$ visits more patches than $\mathbb{P}_2$.

Based on these conditions, we sort the paths from the best to the worst.

4.4.2 Stabilizing the Best Path

We stabilize the patches along the best path $\mathbb{P}$ by making it monotonic or near-monotonic. First, we need to ensure that $\mathbb{P}$ is a B-path or F-path. Assuming $n_{\mathbb{P}}^* = \min(n_B^*, n_F^*) = n_B^* > 0$, we consider n_B^* pairs of nondoubly connected even patches with the smallest pairwise distances. We add a new connection for each pair to make it doubly connected.

Suppose p_a and p_b are one of those pairs, in which p_b's coordinate is greater. Let p_c be their existing connection and p_d be the new one to be added. We choose a set of potential positions for p_d along p_b's upper boundary where the gradients along the boundary are the smallest, so that the horizontal fold line between p_b and p_d does not alter p_b's boundary significantly. From these positions, we pick the one farthest from p_c to avoid a degenerated double connection.

At the final position of p_d, we maximize p_d's width such that the horizontal fold line between p_b and p_d is still a good approximation of p_b's boundary within a given threshold. However, in staircase-like structures, such as the one in Fig. 6a, p_a, p_b, and p_c span the same interval in the j-coordinate, within which p_d is also created. In such a case, we limit p_d's width to $1/6$ of p_c's.

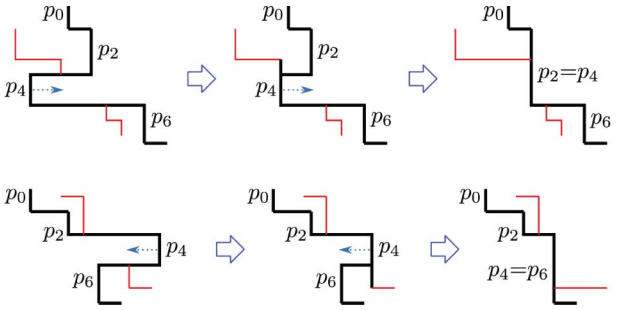

Fig. 12. Two B-paths (thick lines) are made monotonic by merging forward (above) and backward (below). The dotted arrows show the merging direction. Other patches not in these paths (thin lines) may also be involved in the merging process.

After the double connections are created to make $\mathbb{P}$ a B-path, if it is monotonic or near-monotonic then all of its patches are stable, according to Proposition 2. Otherwise, we use merging to transform $\mathbb{P}$ into a monotonic path, which we describe below.

Similar to the foldable OA construction, merging of a patch p is done with the nearest patch p' that is parallel to p. When $z_p < z_{p'}$, we have a forward merging. Otherwise, the merging is backward.

In details, let $\mathbb{P} = \{p_0, p_1, \ldots, p_n\}$ be a B-path with p_{2i} in B-set. We assume that the last even patch has greater z-coordinate than the first one. For each i starting from 0, we find the first $j > i$ such that $z_{p_{2j}} > z_{p_{2i}}$. If $i + 1 < j \leq \lfloor n/2 \rfloor$, along the subpath $\{p_{2i+2}, \ldots, p_{2j-2}\}$, we repeatedly merge the even patch having the smallest z-coordinate forward until we reach p_{2i} or a stable patch (see Fig. 12 (top row)). If j cannot be found, along $\{p_2, \ldots, p_{2i}\}$, we repeatedly merge the even patch having the greatest z-coordinate backward until we reach $p_{2\lfloor n/2 \rfloor}$ or a stable patch (see Fig. 12 (bottom row)). Note that the merging process may involve patches not belonging to $\mathbb{P}$.

As no patch in $\mathbb{P}$ is stable, except the first and the last ones, and the merging stops when we reach a stable patch, this process does not break any existing stable paths while making $\mathbb{P}$ monotonic.

4.4.3 Updating the Paths

After the best path $\mathbb{P}$ is stabilized, we remove it from the list of sorted paths. For each remaining path that contains any newly stabilized patch, we break it into smaller paths, each starting and ending with stable patches, but not visiting any other stable ones. If no more unstable path is found, the algorithm ends. Otherwise, we go back to the first step and sort the remaining paths. As the number of patches is finite, this process finally stabilizes the whole OA.

During our OA creation, the patches are created in the $45°$ image plane, which results in a compression in the image's vertical direction. Therefore, we need to scale the image space by $\sqrt{2}$ along its i-axis to obtain the final OA plan.

5 Results

We show a number of OA designs automatically generated by our system and compare them with the results from Li

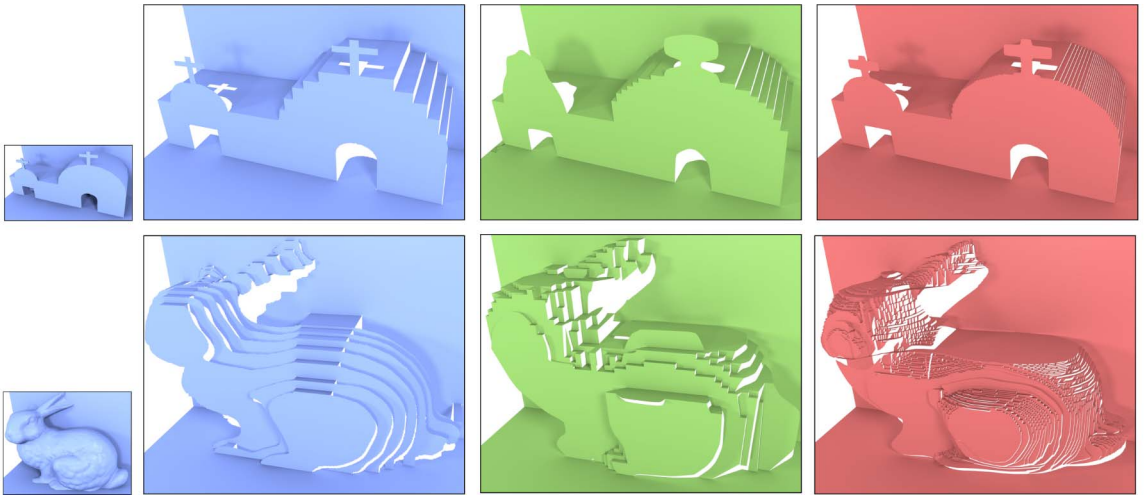

Fig. 13. OAs of the Stanford bunny and a chapel model. (Left) Our results, (middle) Li's [7] results using $64 \times 64 \times 64$ grid, (right) Li's results using $256 \times 256 \times 256$ grid.

et al. [7], which is the only other automatic OA design system. In their implementation, an OA can be generated at several voxel grid resolutions, from $8 \times 8 \times 8$ to $256 \times 256 \times 256$. For comparison, we choose their results at $256 \times 256 \times 256$, the most detailed, and $64 \times 64 \times 64$, the most balanced between contour preservation and voxel size, and the best match to our slicing resolution.

In Fig. 2, the designs from both systems are compared with artists' creations [6]. We observe that Li's method [7] is unable to preserve the small archways in the Rialto Bridge model, unlike in our system where we can preserve them automatically. Furthermore, Li's method cannot produce a good approximation of the Colosseum model. This may be due to their stability formulation, which does not preserve large concave regions.

Figs. 13 and 14 compare our results with Li's for more input models. We smooth the highly bumpy bunny model as a preprocessing step for both systems. The figures illustrate

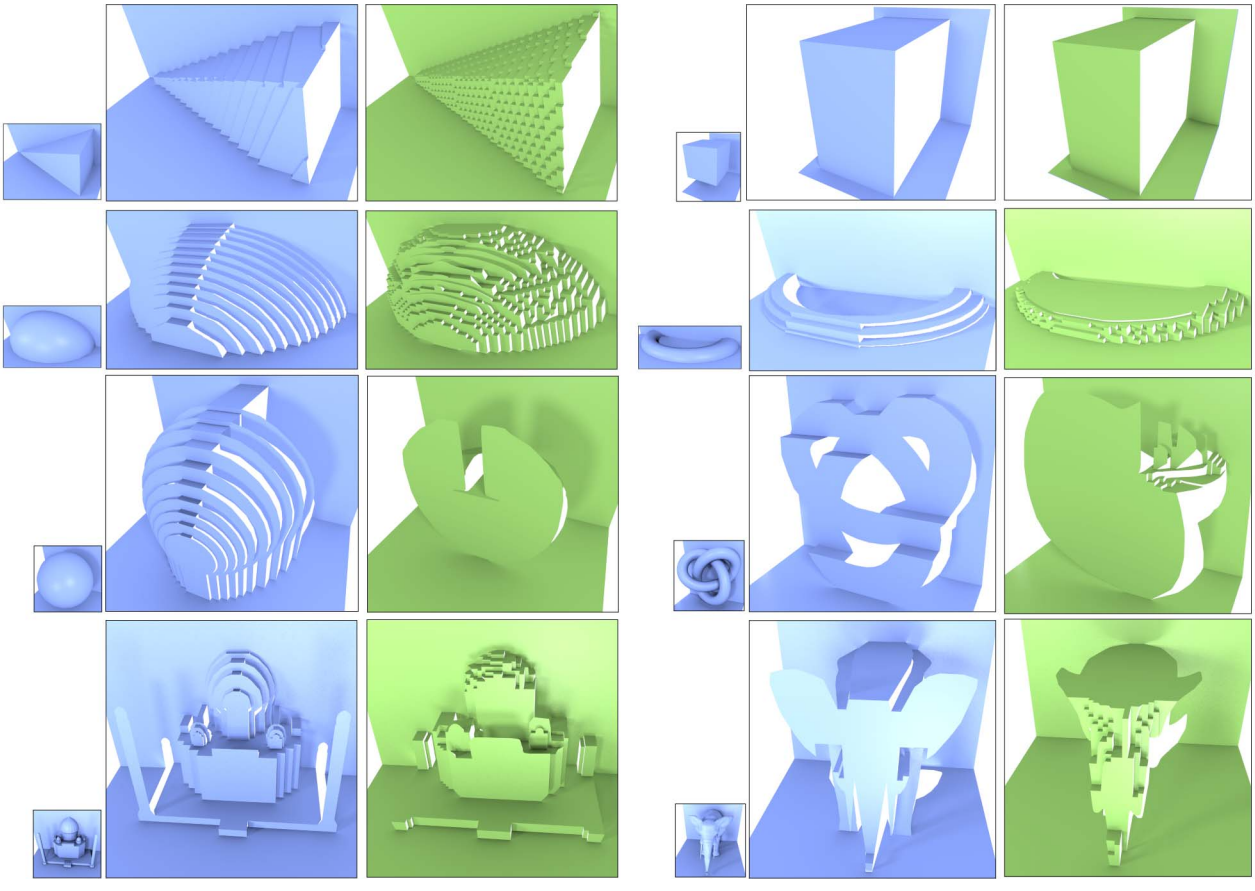

Fig. 14. (Blue) OAs designed by our method, and (green) by the method of Li et al. [7] using $64 \times 64 \times 64$ grid to match our slicing resolution.

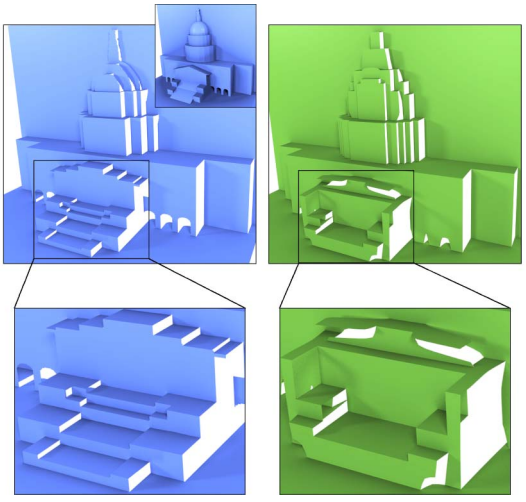

Fig. 15. (Left) Our result for the Capitol Building model, and (right) Li's [7] results using $64 \times 64 \times 64$ grid. The bottom row shows close ups of the staircase structures.

TABLE 1
Deviations from the Input Surfaces

Percentage Difference (%)	Li et al. [7]				Ours
	32	64	128	256	
Chapel	13.78	9.41	4.46	2.63	2.4
Colosseum	44.03	45.3	41.93	40.9	5.59
Quarter-Sphere	25.72	10.1	8.72	8.12	8.7
Rialto Bridge	52.88	23.59	15.64	14.57	12.27
Stanford Bunny	15.44	10.77	9.95	8.17	11.6
Torus	139.49	91.28	84.38	81.19	21.57
Triangular Prism	13.66	6.63	3.95	2.16	3.32
Capitol Building	26.85	14.28	9.65	5.9	4.93

Smaller value means better approximation. Li's method [7] is run with various grid resolutions.

TABLE 2
Time (in mins) to Physically Construct the OAs Designed by Our Algorithm

Model	Time		Model	Time
Stanford Bunny	35		Torus	9
Rialto Bridge	36		Complete Sphere	24
Colosseum	35		Trefoil Knot	10
Chapel	10		Taj Mahal	29
Triangular Prism	14		Elephant	26

the ability of our algorithm in preserving smooth surfaces (e.g., the bunny, the sphere, the torus), as well as sharp contours (e.g., the cross of the chapel) and important creases (e.g., the edge between two faces of the triangular prism).

Our novel stabilization is best demonstrated on the Capitol Building model (Fig. 15). Its main staircase is stabilized as a monotonic path, which minimizes the change in volume and shape, as compared to the stabilization technique used in Li's method [7]. In our implementation, we construct double connections on both sides of the staircase to maintain symmetry.

Monotonic paths are also used for the Stanford bunny, the Rialto Bridge (Fig. 1), and the trefoil knot (Fig. 14) models, while near-monotonic paths are used for the gate (Fig. 6) and the elephant (Fig. 14) models.

TABLE 3
Number of Cuts and Folds in the OAs

Model	Li et al. [7]		Ours	
	Cuts	Folds	Cuts	Folds
Chapel	19	35	13	23
Stanford Bunny	77	188	29	57
Triangular Prism	246	517	25	55
Quarter Sphere	223	496	22	44
Torus	93	202	9	18
Trefoil Knot	12	27	8	17
Taj Mahal	43	85	34	53
Elephant	29	33	14	19

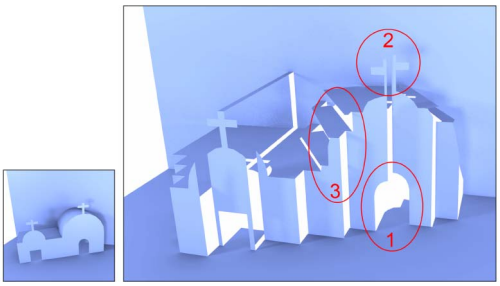

Fig. 16. OA of the chapel model rotated by 30° about the y-axis.

Besides a qualitative visual comparison, we compare our results quantitatively by using the volumetric percentage difference to measure the surface deviation between an OA and the original input surfaces. This is computed by accumulating the depth difference at each pixel, and the sum is then divided by the total volume of the depth map of the original surfaces. The percentage differences of the OAs designed by Li's method [7] and our method are shown in Table 1. Even though we use considerably fewer cuts and folds, our volumetric percentage differences are comparable to Li's at their highest resolution for most models. Especially, our results are significantly better for models with big concave surfaces, since our algorithm can preserve them well.

To demonstrate the practicality of the OAs designed by our algorithm, we also measure the time required to physically construct the real OAs from the 2D designs. Table 2 shows the approximate time in minutes for a novice to cut and fold the pop-ups. In addition to construction time, we count the number of cuts and folds as another measurement of the complexity of the designs. Except the Rialto Bridge model (whose archways are not reconstructed by [7]), the Colosseum, and the complete sphere models (whose shapes are not preserved by [7]), other models show that our OAs require significantly fewer cuts and folds than the corresponding OAs from [7] (Table 3). All of the results presented were generated in at most 15 seconds running on a PC with an Intel Pentium (R) Dual-Core 2 GHz CPU and 4 GB RAM.

Fig. 16 reveals some of the weaknesses of our algorithm. It shows the OA of the chapel model rotated by 30° about the y-axis. Note that the OA is foldable and stable. The major flat surfaces of the model are now not aligned with the patch orientations, resulting in unpleasant fragmentation of the surfaces in the OA. Even though our algorithm

does try to avoid slicing through holes, sometimes this is inevitable when the holes are large, as highlighted by the red ellipse "1". Moreover, the slicing can break small details, such as the cross highlighted by the red ellipse "2". In addition, because we slice each segment locally, the slices on adjacent segments may not align well. This may cause undesirable result when two adjacent segments have multiple slices in different orientations. Such a case is highlighted in the red ellipse "3" in Fig. 16.

6 Conclusion and Discussion

In this paper, we present a method for automatic OA generation grounded on a more comprehensive geometric formulation. Sufficient and necessary conditions for the foldability of parallel OAs are presented. These support our novel image-domain approach for creating 2D plans that can fully pop up into parallel patches, whose positions and contours closely depict the geometric shapes of the 3D input models.

We also formulate a set of conditions for stabilizing OA structures. Our stability conditions are more general than those in recent studies [7], [15], [26], and are able to include all the structures we have seen used by pop-up artists [6], [10], [11]. Utilizing our novel double connections, we can stabilize OAs without significantly affecting their shapes.

Visual and quantitative analyses show the ability of our approach to design foldable and stable OAs that both approximate the desired structures well and are practical for real construction. Since we carefully compute the slice positions and contours, our designs can preserve the input surfaces while creating significantly fewer cuts and folds, as compared to the automatic OA tool of Li et al. [7].

Limitations and Future Work. Although our algorithm produces visually pleasing designs, it may not always resemble some of the designer's artistic choices. For instance, in the Rialto Bridge model (Fig. 2), the sharp tip of the roof is depicted by the artist, but not in our design, because it is projected to a horizontal fold line. Similarly, in our implementation, the slicing width thresholds computed automatically may not be desirable for users. These choices are subjective and require further user studies. A possible solution is to wrap our algorithm in an interactive system, in which users can interactively specify the desired details and patch width thresholds. Our foldable patches construction and stabilization technique can also be integrated into other OA design systems, such as [13], [14], allowing the user to have more control over the design while keeping the pop-ups valid.

In addition, due to the OA's constraint of a single piece of paper, we are not always able to preserve highly concave regions, as shown with the floating cube, complete sphere, and trefoil knot models (Fig. 14).

Our work also offers interesting possibilities for future research. While the foldability conditions are proved to be sufficient and necessary, it is still unknown whether necessary conditions for stability checking are achievable. Besides, generalizing the current conditions to nonparallel OA will allow more flexibility in patch creation and shape approximation.

We can extend our work to allow other types of input representations, such as drawings and photographs. An algorithm for OA design from such inputs will be exciting and may require single-view reconstruction techniques [39].

In our current geometric formulation, we do not take into account the real physical characteristics of paper. However, in practice, the thickness, mass, strength, and elasticity of paper are important considerations for OA design. It will be useful to estimate the strength of an OA. If it encounters physical weakness, careful adjustments will be needed to strengthen the structure, while maintaining its geometric foldability and stability.

Acknowledgments

This work was supported by the Singapore MOE Academic Research Fund (Project No. T1-251RES1104).

References

[1] D.A. Carter, *One Red Dot: A Pop-Up Book for Children of All Ages.* Little Simon, 2005.
[2] R. Sabuda and L. Carroll, *Alice's Adventures in Wonderland: A Pop-up Adaptation.* Simon & Schuster, 2003.
[3] M. Bataille, *ABC3D.* Roaring Brook Press, 2008.
[4] E.E. Hui, R.T. Howe, and M.S. Rodgers, "Single-Step Assembly of Complex 3-D Microstructures," *Proc. IEEE 13th Ann. Int'l Conf., Micro Electro Mechanical Systems (MEMS '00),* pp. 602-607, 2000.
[5] J.P. Whitney, P.S. Sreetharan, K.Y. Ma, and R.J. Wood, "Pop-up Book MEMS," *J. Micromechanics Microeng.,* vol. 21, no. 11, pp. 115021-115027, 2011.
[6] M. Bianchini, I. Siliakus, and J. Aysta, *The Paper Architect: Fold-It-Yourself Buildings and Structures.* Crown Publishing Group, 2009.
[7] X.-Y. Li, C.-H. Shen, S.-S. Huang, T. Ju, and S.-M. Hu, "Popup: Automatic Paper Architectures from 3D Models," *ACM Trans. Graphics,* vol. 29, no. 4, pp. 111:1-111:9, 2010.
[8] P. Jackson and P. Forrester, *The Pop-Up Book: Step-by-Step Instructions for Creating over 100 Original Paper Projects.* Henry Holt and Co., 1993.
[9] D.A. Carter and J. Diaz, *Elements of Pop-Up.* 1999.
[10] C. Barton, *The Pocket Paper Engineer: How to Make Pop-Ups Step-by-Step.* Popular Kinetics Press, 2008.
[11] D. Birmingham, *Pop-Up Design and Paper Mechanics: How to Make Folding Paper Sculpture.* Guild of Master Craftsman Publications Ltd., 2011.
[12] Y. Lee, S. Tor, and E. Soo, "Mathematical Modeling and Simulation of Pop-up Books," *Computers & Graphics,* vol. 20, no. 1, pp. 21-31, 1996.
[13] A. Glassner, "Interactive Pop-up Card Design, Part 1," *IEEE Computer Graphics and Applications,* vol. 22, no. 1, pp. 79-86, Jan. 2002.
[14] S.L. Hendrix and M.A. Eisenberg, "Computer-Assisted Pop-up Design for Children: Computationally Enriched Paper Engineering," *Advanced Technology Learning,* vol. 3, no. 2, pp. 119-127, Apr. 2006.
[15] J. Mitani and H. Suzuki, "Computer Aided Design for Origamic Architecture Models with Polygonal Representation," *Proc. Computer Graphics Int'l (CGI '04),* pp. 93-99, 2004.
[16] J. Mitani and H. Suzuki, "Making Papercraft Toys from Meshes Using Strip-Based Approximate Unfolding," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 259-263, Aug. 2004.
[17] I. Shatz, A. Tal, and G. Leifman, "Paper Craft Models from Meshes," *Visual Computer: Int'l J. Computer Graphics,* vol. 22, no. 9, pp. 825-834, Sept. 2006.
[18] F. Massarwi, C. Gotsman, and G. Elber, "Papercraft Models Using Generalized Cylinders," *Proc. 15th Pacific Conf. Computer Graphics and Applications,* pp. 148-157, 2007.
[19] J. Xu, C.S. Kaplan, and X. Mi, "Computer-Generated Papercutting," *Proc. 15th Pacific Conf. Computer Graphics and Application,* pp. 343-350, 2007.
[20] Y. Li, J. Yu, K.-l. Ma, and J. Shi, "3D Paper-Cut Modeling and Animation," *Computer Animation and Virtual Worlds,* vol. 18, nos. 4/5, pp. 395-403, Sept. 2007.

[21] E. Demaine and J. O'Rourke, *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge Univ. Press, 2007.
[22] J. O'Rourke, *How to Fold It: The Mathematics of Linkages, Origami and Polyhedra.* Cambridge Univ. Press, 2011.
[23] T. Tachi, "Origamizing Polyhedral Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 2, pp. 298-311, Mar. 2010.
[24] A. Glassner, "Interactive Pop-up Card Design, Part 2," *Computer Graphics and Application,* vol. 22, no. 2, pp. 74-85, 2002.
[25] S. Iizuka, Y. Endo, J. Mitani, Y. Kanamori, and Y. Fukui, "An Interactive Design System for Pop-up Cards with a Physical Simulation," *Visual Computer,* vol. 27, nos. 6-8, pp. 605-612, http://dx.doi.org/10.1007/s00371-011-0564-0, June 2011.
[26] Z. Abel, E.D. Demaine, M.L. Demaine, S. Eisenstat, A. Lubiw, A. Schulz, D.L. Souvaine, G. Viglietta, and A. Winslow, "Universality Results for Pop-Up Cards," rapid post, Universita di Pisa. http://www.di.unipi.it/~vigliett/, 2012.
[27] Tama Software. PePaKuRa Designer, http://www.tamasoft.co.jp/pepakura-en, 2007.
[28] J.-m. Chen and Y.-z. Zhang, "A Computer-Aided Design System for Origamic Architecture," *Proc. Int'l Conf. Supercomputing,* 2006.
[29] X.-Y. Li, T. Ju, Y. Gu, and S.-M. Hu, "A Geometric Study of V-Style Pop-ups: Theories and Algorithms," *ACM Trans. Graphics,* vol. 30, no. 4, pp. 98:1-98:10, July 2011.
[30] J. McCrae, K. Singh, and N.J. Mitra, "Slices: A Shape-Proxy Based on Planar Sections," *ACM Trans. Graphics,* vol. 30, no. 6, pp. 168:1-168:12, Dec. 2011.
[31] K.A. Stevens, "The Visual Interpretation of Surface Contours," *Artificial Intelligence,* vol. 17, no. 13, pp. 47-73, 1981.
[32] J. Todd and F. Reichel, "Visual Perception of Smoothly Curved Surfaces from Double-Projected Contour Patterns," *J. Experimental Psychology. Human Perception and Performance,* vol. 16, pp. 665-674, 1990.
[33] J. Koenderink, *Solid Shape.* MIT Press, 1990.
[34] J. Feldman and M. Singh, "Information along Contours and Object Boundaries," *Psychological Rev.,* vol. 112, pp. 243-252, 2005.
[35] R. Mehra, Q. Zhou, J. Long, A. Sheffer, A. Gooch, and N.J. Mitra, "Abstraction of Man-Made Shapes," *ACM Trans. Graphics,* vol. 28, no. 5, pp. 137:1-137:10, Dec. 2009.
[36] E. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis, "3D Mesh Segmentation Methodologies for CAD Applications," *Computer-Aided Design and Application,* vol. 4, no. 6, pp. 827-842, 2007.
[37] A. Shamir, "A Survey on Mesh Segmentation Techniques," *Computer Graphics Forum,* vol. 27, no. 6, pp. 1539-1556, 2008.
[38] R.C. Gonzalez and R.E. Woods, *Digital Image Processing,* second ed. Prentice Hall, 2002.
[39] D. Hoiem, A.A. Efros, and M. Hebert, "Automatic Photo Pop-up," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 577-584, July 2005.

Sang N. Le received the BSc degree in computer science, with a minor in mathematics, from NUS in 2006. He is working toward the PhD degree and is a teaching staff at the School of Computing of the National University of Singapore (NUS). He has worked in a number of research areas, ranging from biomechanics and human motion synthesis to 3D reconstruction. His current focus is on automatic design of paper pop-ups.

Su-Jun Leow received the BSc degree in computer science with a major in communications and media, and the MSc degree in computer science from the National University of Singapore (NUS), in 2007 and 2010, respectively. She is working toward the PhD degree at the School of Computer Engineering of Nanyang Technological University (NTU). Her research interests include automatic speech recognition, spoken term detection image processing, and computational photography.

Tuong-Vu Le-Nguyen received the BSc degree from Vietnam National University in 2005, and the MSc degree in computer science from the National University of Singapore in 2012. He has worked in a number of projects in the areas of computer graphics and computer vision. He is running a startup company that provides cloud-based software solutions.

Conrado Ruiz Jr. received the BSc degree in computer science from De La Salle University (DLSU)-Manila, the MSc degree from NUS. He is working toward the PhD degree at the School of Computing of the National University of Singapore (NUS). He is currently on leave from his faculty position at DLSU. He has coauthored papers in the areas of computer graphics and multimedia retrieval.

Kok-Lim Low received the MSc and BSc (Honors) degrees in computer science from NUS, the PhD degree in computer science from the University of North Carolina at Chapel Hill. He is an assistant professor at the Department of Computer Science of the National University of Singapore (NUS). His research interests include computational art, real-time rendering, and computational photography.