Image-Based OA-style Paper Pop-up Design via Mixed-Integer Programming

Fei Huang, Chen Liu, Kai-Wen Hsiao, Ying-Miao Kuo, Hung-Kuo Chu, and Yong-Liang Yang

Abstract—Origami architecture (OA) is a fascinating papercraft that involves only a piece of paper with cuts and folds. Interesting geometric structures 'pop up' when the paper is opened. However, manually designing such a physically valid 2D paper pop-up plan is challenging since fold lines must jointly satisfy hard spatial constraints. Existing works on automatic OA-style paper pop-up design all focused on how to generate a pop-up structure that approximates a given target 3D model. This paper presents the first OA-style paper pop-up design framework that takes 2D images instead of 3D models as input. Our work is inspired by the fact that artists often use 2D profiles to guide the design process, thus benefited from the high availability of 2D image resources. Due to the lack of 3D geometry information, we perform novel theoretic analysis to ensure the foldability and stability of the resultant design. Based on a novel graph representation of the paper pop-up plan, we further propose a practical optimization algorithm via mixed-integer programming that jointly optimizes the topology and geometry of the 2D plan. We also allow the user to interactively explore the design space by specifying constraints on fold lines. Finally, we evaluate our framework on various images with interesting 2D shapes. Experiments and comparisons exhibit both the efficacy and efficiency of our framework.

Index Terms—origami architecture, paper pop-up, image-based design, foldable structure, mixed-integer programming

1 Introduction

Paper is one of the most popular media that plays a significant role in our lives with a long history. Besides common usages such as writing, drawing, and printing, people also utilize paper for creative design, resulting in various artistic forms, including origami, kirigami, paper cutting, etc. These artistic forms fascinate people due to the easy access to paper. Therefore, they have been widely used for various applications, such as craft education, indoor/outdoor decoration, greeting card and bookmaking.

Each form of paper art typically defines specific user manipulations. For instance, origami only involves folding a single piece of paper, while kirigami also allows cutting. Those restrictions on each artistic form make the design process time-consuming and skill-demanding, thus motivating researchers, especially those in the computer graphics field, to develop computational systems to facilitate the design process.

In recent years, the computational design of a unique art form, the *origami architecture (OA)* structure, has gained researchers' attention. Such a papercraft stems from the traditional origami architecture, which Masahiro Chatani initially introduced in the 1980s [1]. It is a design of cuts and folds on a single piece of paper such that interesting geometric structures automatically 'pop up' when the paper is opened (see Figure 2). The restriction of using only cuts and folds to construct a physically valid and stable 3D structure makes the design process extremely challenging. Early computational design work [2] focused on providing an easy-to-use virtual design environment to assist the design process. However, the ultimate placement of cuts and folds still depends on the user. Li et al. [3] introduced a novel formulation that defines the foldability and stability of OA-style paper pop-up structures. In addition, they proposed the first computational framework to automatically generate a physically valid paper pop-up design from an input 3D model. By generalizing the formulation, researchers have presented a series of works that either improve the approximation accuracy [4] or resolve paper pop-ups with new styles [5], [6].

Unlike prior works where paper pop-up designs approximate input 3D models, we aim at automatically generating OA-style 3D pop-up structures from 2D images. The motivation behind our work is two-fold. First, paper pop-up designs created by artists are often inspired by 2D sketches [7] (see Figure 2). Second, the reliance on 3D models limits the scope of the methods above and cannot benefit from the availability and popularity of 2D images. Besides, the lack of 3D geometric information in the input 2D image makes image-based OA-style paper pop-up design incompatible with prior works because it violates the assumption of the formulation therein.

In this work, we present a novel foldability formulation in 2D by investigating how to 'unproject' the input 2D shape into a foldable configuration in 3D. Our formulation requires a set of topological and geometric constraints specified among fold lines in the paper pop-up design. To facilitate the specification of constraints, we propose a novel graph representation to abstract the paper pop-up design, where graph nodes represent patches bounded by cut/fold lines and graph edges represent fold lines. In contrast to prior works that achieve foldability by 3D par-

- *F. Huang, Y. Yang are with the Department of Computer Science, University of Bath, Bath, UK.*
  Email: fh463@bath.ac.uk, y.yang@cs.bath.ac.uk.
- *C. Liu is with Meta - Facebook Reality Labs, USA.*
  Email: chenliu@wustl.edu.
- *K. Hsiao, Y. Kuo, and H. Chu are with Department of Computer Science, National Tsing Hua University, HsinChu, Taiwan.*
  Email:{kevin30112, jollytreeskuo}@gmail.com, hkchu@cs.nthu.edu.tw.

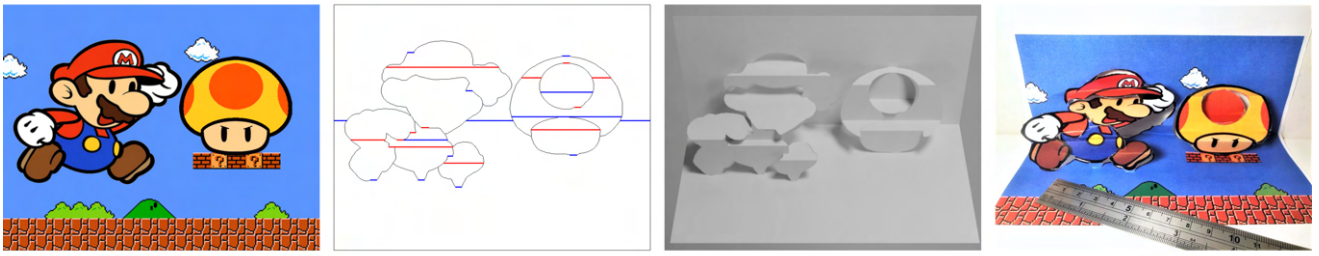


Fig. 1. We present the first computational design framework that can automatically generate physically valid paper pop-up designs from 2D images. From left to right: the input image; the generated 2D pop-up plan with mountain/valley fold lines in red/blue, and cut lines in black; the corresponding 3D pop-up geometry; the realized paper pop-up with textures in practice.

allel projection, we present a novel graph-based topology and geometry optimization framework via mixed-integer programming to automatically generate a foldable paper pop-up design (see Figure 1). We also introduce an additional sufficiency condition on stability and a design space exploration strategy to ensure stability while preserving the shape semantics of the input 2D image. We further investigate how to initialize and optimize fold lines for feature preservation. Finally, we test our image-based paper pop-up design framework on various input images. The results and comparisons both demonstrate the effectiveness of our framework.

In summary, our work makes three major contributions:

- We present the first image-based OA-style paper pop-up design framework via mixed-integer optimization, which can automatically generate physically valid results from 2D images.
- We introduce a novel image-based paper pop-up formulation based on geometric and topological analysis of a paper pop-up graph representation.
- We derive the foldability conditions using a novel 'unprojection'-based approach and improve stability conditions together with a simple yet effective design space exploration strategy.

The rest of the paper is organized as follows. Section 2 reviews prior works relevant to ours. Section 3 introduces the basic concepts of the paper pop-up plan and its abstracted graph and the mathematical formulation of geometric and topological properties of a valid pop-up plan. Section 4 presents our computational paper pop-up design framework using mixed-integer programming based on the formulation. Section 5 evaluates the presented framework.

Fig. 2. Practical paper pop-up designs manually created by artist [7]. The inset shows the corresponding 2D pop-up plan, where the cut lines and fold lines are inspired by 2D sketches/images, and are highlighted in black and blue/red respectively (zoom in for details).

Section 6 concludes the paper and discusses potential future directions.

2 Related Work

2.1 Papercraft Design

Papercraft is a delicate art form that creates 3D shapes using paper as the artistic medium. How to computationally design paper crafts of different styles has been an active research topic.

Origami. Origami, the traditional oriental art of folding, has been well studied in the field. The mathematical formulations of foldability together with practical folding algorithms are comprehensively discussed in [8] and [9]. Tachi [10] presented an interactive system that can create origami designs for polyhedral surfaces. By analyzing developable surfaces and their discrete counterpart, various methods have been proposed for modeling origami designs composed of multiple developables joined by straight and curved creases. The employed discrete developable surface representations include triangle meshes [11], quad meshes [12], spline surfaces [13], orthogonal geodesic nets [14], [15], and pleated structures [16]. In addition to geometric modeling, the physical simulation and realization of folding structures have also been investigated [17], [18], [19].

Kirigami. Kirigami is a variation of origami that also includes paper cutting rather than solely folding. Mina et al. [20] employed conformal geometry to model the flat auxetic material with small incisions such that the material can be expanded to construct freeform surfaces. Mina et al. [21] presented a more general class of cut/fold patterns for generating structures at the limit of the material expansion. Choi et al. [22] deformed a regular square grid and cut along its edges to deploy 3D surfaces with a checkerboard pattern that alternates the original quad faces with quad holes. Jiang et al. [23] performed geometric modeling with a corrugated watertight box kirigami representation, which allows an isometric unfolding into a planar domain after introducing appropriate cuts.

Paper Cutting/Intersecting/Gluing. Paper cutting, a special folk art, has been widely used as decorative patterns. Xu et al. [24] presented an automatic image-based paper cutting design framework. Li et al. [25] studied paper cutting in 3D and explored how to animate paper cuts. Several works

have been presented to generate paper statues using mutually intersecting planar paper sections. Mccrae et al. [26] presented several principles inferred from user studies to approximate a given shape using planar sections. Hildebrand et al. [27] utilized an extended binary space partitioning tree to generate planar sections with guaranteed castability for real fabrication. Cignoni et al. [28] proposed a method that aligns planar sections to a vector field defined on the given shape. How to approximate a 3D shape by gluing a set of developable paper segments has also been studied. Mitani and Suzuki [29] and Liu et al. [30] abstracted a given shape using triangle strips. Julius et al. [31] decomposed a mesh model into a set of quasi-developable charts. Shatz et al. [32] used planes and conics as proxies to approximate a given shape. Massarwi et al. [33] employed generalized cylinders to generate developable parts. Wang and Tang [34] and Stein et al. [35] iteratively deformed a given shape to be developable while introducing creases. Ion et al. [36] wrapped a given shape with multiple developable patches based on a global optimization for effectively finding the placement of patches.

2.2 Computational Paper Pop-ups

Compared with kirigami, fold line generation is more restrictive for OA-style paper pop-up design due to the simplicity of the 'one-fold-all-pop-up' process. Glassner [37], [38] introduced a simple rule-based method to generate pop-up cards using simple geometry. Hendrix et al. [39] developed a similar system for children. Other early works mainly focused on interactive design, where the user has to intervene if needed, and the validity of the pop-up design is not guaranteed. Mitani and Suzuki [2] proposed a seminar work specific to origami architectures. Their system allows the user to design patches with vertical/horizontal orientation in an interactive manner. The validity of the design can be judged by condition verification. However, the resultant design is not guaranteed to be valid, and the user has to resolve failure cases in a trial-and-error manner. Similar systems such as [40] have also been presented, but they all require cumbersome user interactions with specialized domain knowledge. The first work that guarantees design validity was proposed in [3]. The authors invented a set of desirable conditions to ensure foldability and stability, resulting in plausible pop-up architecture designs. However, due to voxelization, the resultant design cannot capture the geometric details of the input architecture model, especially for curved contours. Le et al. [4] presented a novel surface-and-contour-aware method based on feature preserving segmentation and patch connection, leading to high-quality pop-up outcomes. They also proposed additional sufficiency conditions for stability to better preserve surface features. Li et al. [5] extended the prior work to v-style pop-ups. In addition to single-style pop-ups, Ruiz et al. [6] proposed an approach to convert a 3D model into a multi-style paper pop-up. How to use different mechanisms of paper pop-ups to convey motion information from 3D animation has been studied in [41]. In contrast to previous works that rely on a 3D model to guide the design process, we initiate the computational pop-up design using 2D shapes. The large availability of 2D image data enriches design possibilities. We also present novel geometrical and topological conditions on foldability and stability for image-based pop-up design that are orthogonal to prior works.

In a separate scenario of transforming pop-up books, Xiao et al. [42] studied how to transform one 2D pattern into another 2D pattern through a 3D pop-up interface, which comprises several pieces of paper patches that can be glued and crossed. This scenario is different from our paper pop-up design, which only allows folding and cutting on a single piece of paper.

2.3 Photo Pop-up

In single view reconstruction, 'photo pop-up' refers to estimating 3D properties of non-local image regions for higher-level single image understanding and is mainly used to understand global structural cues of outdoor and indoor scenes. Different scene models have been proposed to generate photo pop-up results, such as boxed model [43], staggered model [44], [45], and layered model [46], [47]. Due to the involvement of interleaved orthogonal planes, the staggered model is the closest to our paper pop-up structure. However, our work is fundamentally different from the above as we consider how to pop up a 2D shape under strong fabrication constraints for paper cutting and folding, rather than recovering a rough scene model from a single image using geometry cues, e.g., depth, wall, and ground.

3 Formulation

In this section, we first introduce the preliminary concepts of OA-style paper pop-up structure, such as background patches, fold lines, the folding angle, etc., in Section 3.1. Then we present the novel graph-based representation that encodes both geometric and topological properties of the pop-up plan in Section 3.2. We discuss the foldability conditions of the paper pop-up plan in Section 3.3 based on the graph representation. Finally, we present the stability conditions of the paper pop-up plan in Section 3.4. The following formulation serves as the foundation of the OA-style paper pop-up generation algorithms in Section 4. In particular, the foldability conditions are employed to set constraints during the optimization in Section 4.4, while the stability conditions are utilized to guide the design exploration in Section 4.5.

3.1 Preliminaries

Our goal is to generate a physically realizable OA-style pop-up design from a 2D image. To this end, we adopt the same type of pop-up design presented in [3], [4], where the design domain D is a single rectangular sheet of paper divided into a set of non-overlapping patches, P. The boundaries of patches are marked with either cut lines or straight fold lines. Two special background patches, denoted as p_S and p_E, are bounded by the outlines of D and share a common main fold line f_M (see Figure 3a).

As shown in Figure 3b, 3D structures pop up when holding p_E and rotating p_S about f_M. We call the angle formed by p_S and p_E the folding angle θ, which is valued between $180°$ (fully opened) and ε (fully folded), where ε

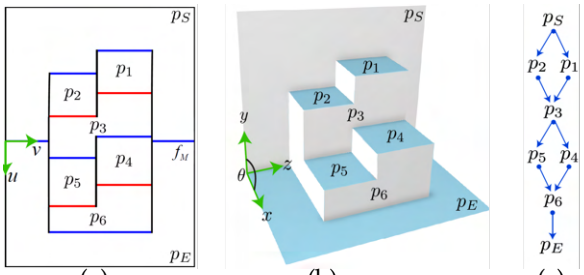


Fig. 3. (a) A 2D pop-up plan with cut lines (black) and fold lines (red and blue represent mountain and valley folds, respectively), and (b) the corresponding 3D pop-up structure at folding angle $\theta = 90°$. (c) The constructed pop-up graph.

is a small positive value. We also address a common type of pop-up structure where all the fold lines must be parallel to the main fold line during the folding process. To aid the following presentation, we set up orthogonal coordinates in 2D (u- and v-axis) and 3D (x-, y-, and z-axis) domains, as shown in Figure 3a and 3b, in which the main fold line f_M coincides with the v- and z-axis, and the x- and y- axis are perpendicular to the z-axis and parallel to the patch p_E and p_S, respectively.

3.2 Pop-up Plan

To facilitate the definition and formulation of pop-up plans, we propose to encode the important geometric and topological properties of a popup plan using a graph representation (see Figure 3c).

Definition 3.1. *A* pop-up graph *is a directed graph $G = \{V, E\}$, where the nodes V correspond to the set of patches $P = \{p_1, ..., p_n\} \cup \{p_S, p_E\}$ and the directed edges E represent the set of fold lines excluding f_M. Two nodes, p_i and p_j, are connected by an edge if and only if they share a common fold line. The direction of the edge is determined by the linear order of its incident nodes p_i and p_j in a directed path from p_S to p_E that contains p_i and p_j.*

Definition 3.2. *The* parity *(odd/even property) of graph nodes and edges is defined as follows. First, if the node p_S is labeled odd (even), then the node p_E is labeled even (odd). A node is labeled odd (even) if all its incoming edges are from even (odd) nodes, otherwise we say the parity of the node is undefined. A directed edge is labeled odd (even) if its head node is labeled odd (even), otherwise we say the parity of the edge is undefined.*

We now formally define a pop-up plan as follows.

Definition 3.3. *A* pop-up plan *is a set of co-planar patches where:*

- *All patches are connected by parallel fold lines and separated from cut lines, overall forming a rectangular domain.*
- *The patches are non-intersecting, except at the shared boundaries.*
- *Each patch p_i must lie in at least one directed path from p_S to p_E.*

The above three properties are essentially inherited from [3], [4] as we all address the same type of paper pop-up design. For simplicity, we call the last property the connectivity of a pop-up plan.

3.3 Foldability

A valid pop-up plan should be continuously foldable from $\theta = 180°$ to $\theta = \varepsilon$ without introducing artifacts such as bending or intersecting (see Figure 4 in [4]). To ensure the foldability of resultant pop-up plans, a formal definition as well as a set of geometric conditions have been proposed in prior works [3], [4]. However, the previous systems all deal with 3D models while our work aims at 2D images. This fundamental difference in the input makes their approaches infeasible for our problem on the verification of foldability. Before introducing the novel necessity and sufficiency conditions of foldability tailor-made for the image-based pop-up design, we will first review the conditions proposed in prior works.

According to [4], a foldable pop-up plan will form a parallel configuration with all patches being parallel to p_S or p_E at any folding angle θ. When $\theta = 90°$, it is called an orthogonal configuration. Then the condition for a foldable pop-up plan is as follows:

Proposition 3.1. *A 2D pop-up plan is foldable if and only if the plan is the projection of a parallel configuration along direction $\mathbf{d}$ onto the xz-plane, where $\mathbf{d}$ is perpendicular to the z-axis and bisecting the folding angle θ.*

The proof of the sufficiency and necessity of Proposition 3.1 can be found in [3] and [4], respectively. In Figure 3, the 2D pop-up plan (a) is actually a $45°$ parallel projection of its orthogonal configuration (b).

Since the input to their methods is a 3D model, constructing and checking a valid foldable pop-up plan can be done trivially by projecting the input 3D geometry onto the 2D plane along the vector perpendicular to the main fold line from a $45°$ orthographic view. However, our system starts with a 2D pop-up plan, which contains a foreground 2D shape (from the input 2D image) and two background patches defined by the main fold line specified by the user. Due to the lack of a corresponding 3D geometry of the input 2D shape, the projection-based approach is thus infeasible. In addition, for an intriguing pop-up design, the fold lines and cut lines should respect the shape semantics (e.g., shape contour, semantic part boundary) of the input 2D shape when folded. To this end, the system has to create (i) fold lines that attach the foreground object to two background patches; and (ii) fold lines inside the foreground object to better capture semantic features of the input 2D shape during the folding process. Therefore, we propose a set of novel geometric conditions based on the 2D configuration of fold lines to model the foldability of the pop-up plan.

The key idea behind our foldability formulation is to 'unproject' the 2D pop-up plan into a parallel configuration in 3D, and prove the uniqueness of such a mapping. Based on the graph representation, we now present the geometric and topological conditions for a foldable pop-up plan.

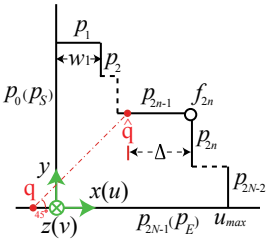
Proposition 3.2. *A 2D pop-up plan is foldable if and only if all the nodes (patches) in the pop-up graph have unique parity, and satisfy regularity: if the node p_S is labeled odd (even), then all the odd (even) and even (old) patches must be respectively parallel to p_S and p_E at any folding angle.*

The necessity of the above proposition is easy to prove, since otherwise we can easily find a counter-example that

is not foldable. Now we want to prove its sufficiency based on the geometry and topology of the unprojected parallel configuration in 3D. To simplify the discussion, here we assume (i) the node p_S is labeled even, and (ii) the unprojecting angle is $45°$. Note that (ii) implies an orthogonal configuration in 3D with folding angle $\theta = 90°$. The only difference from other parallel configurations is essentially the orientation of the y-axis (thus θ) [3].

Unique parity This property can be proved trivially as it naturally follows the statement in [4]: in a parallel configuration, if a patch is even (odd) in one directed path, it is also even (odd) in all other directed paths from p_S to p_E.

Regularity Given a point $\mathbf{q}$ in the 2D plan, we would like to prove that there exists a bijective mapping between $\mathbf{q}$ and its unprojected 3D position $\hat{\mathbf{q}}$ in the orthogonal configuration. The proof for $\mathbf{q} \in \{p_S, p_E\}$ is trivial because p_E is fixed in the same plane during the folding process, while p_S is rotated rigidly along the main fold line f_M during the folding process, thus we can always compute a unique corresponding point in the orthogonal configuration by the $45°$ unprojection. For other patches, given a directed path $\mathbb{P} = \{p_0(p_S), p_1, ..., p_{2N-1}(p_E)\}$ in the graph, suppose $\mathbf{q}$ lies in an odd node (patch) p_{2n-1}, where $1 \leq n < N$ (side view in the inset). Then $\mathbf{q}$'s coordinates in 2D are:



$$\mathbf{q} = (u_{max} - \Sigma_{i=n+1}^{N-1} w_{2i-1} - \Sigma_{i=n}^{N-1} w_{2i} - \Delta, v), \quad (1)$$

where u_{max} is the coordinate of fold line in 2D along u-axis, which connects p_{2N-2} and p_E. The term w_k indicates the width of node (patch) p_k and is defined as the distance between two consecutive fold lines along the path $\mathbb{P}$. Δ is the distance in u-axis between $\mathbf{q}$ and the fold line connecting p_{2n-1} and p_{2n}. If the regularity constraints are satisfied, $\mathbf{q}$'s 3D coordinates in the orthogonal configuration are:

$$\hat{\mathbf{q}} = (u_{max} - \Sigma_{i=n+1}^{N-1} w_{2i-1} - \Delta, \Sigma_{i=n}^{N-1} w_{2i}, v) \quad (2)$$

It is easy to prove that $\mathbf{q}$ is equal to the $45°$ projection of $\hat{\mathbf{q}}$, as the projection maps any 3D point (x, y, z) to $(x - y, 0, z)$. Similarly we can prove that this equality also holds when $\mathbf{q}$ lies in even node (patch) p_{2n} with $1 \leq n < N$. Overall we have proved that all the points in the 2D plan can be generated by $45°$ parallel projection from an orthogonal configuration in 3D. Then from the sufficiency of Proposition 3.1, we know that the 2D plan is foldable. According to Proposition 3.2, we have the following facts.

Corollary 3.1. *For an odd (even) node p_i in a directed path from p_S to p_E, the sum of widths for all even (odd) nodes between p_i and p_E (p_S) is consistent across all other directed paths that contain p_i.*

Proof. Here we prove only the case where p_i is odd, as the case of even nodes can be prove similarly. From the proof of Proposition 3.2, it is easy to show that the distance d between odd node p_i and p_E is simply the y coordinate of any point lying in p_i, i.e., $d = \Sigma_{i=n}^{N-1} w_{2i}$, thus equals to the sum of widths for all even nodes between p_i and p_E. Since p_i is always parallel to p_E, d is constant no matter which directed path is used for estimation. □

Corollary 3.2. *For an edge f_i in the graph with u_i and (x_i, y_i) denoting respectively the location of the corresponding fold line in the 2D plan and 3D orthogonal configuration, we have:*

$$x_i - y_i = u_i \quad (3)$$

Proof. Given an edge f_{2n} that connects p_{2n-1} and p_{2n} in a directed path from p_S to p_E, and let's assume p_{2n-1} is an odd node, which makes f_{2n} an even edge (fold line). Then according to the proof of Proposition 3.2, the 2D and 3D coordinates of points lying on the f_{2n} can be computed by setting Δ to 0, which leads to:

$$u_{2n} = u_{max} - \Sigma_{i=n+1}^{N-1} w_{2i-1} - \Sigma_{i=n}^{N-1} w_{2i} \quad (4)$$

in the 2D plan, and

$$(x_{2n}, y_{2n}) = (u_{max} - \Sigma_{i=n+1}^{N-1} w_{2i-1}, \Sigma_{i=n}^{N-1} w_{2i}) \quad (5)$$

in the 3D orthogonal configuration. We can see that $u_{2n} = x_{2n} - y_{2n}$, thus Corollary 3.2 holds for f_{2n}. Note that the case of an odd edge can be proved similarly. □

3.4 Stability

In addition to foldability, another key property that renders a physically realizable pop-up design is the stability of pop-up plans. A pop-up plan is stable if the folding process requires no external forces except holding and turning two background patches, p_S and p_E, along the main fold line, and all other patches remain stationary as p_S and p_E are held steady at any folding angle. Here we adopt the same definition of stability as in [3], [4].

Definition 3.4. *A fold line is said to be* stable *if it has a unique 3D location at any folding angle when fixing p_E and rotating p_S along the main fold line, and vice versa. A patch is said to be* stable *if it carries at least two non-collinear and stable fold lines. A pop-up plan is said to be stable if all of its patches are stable.*

Sufficient conditions on stability have been studied comprehensively in previous works. There are two main strategies to make a foldable popup plan stable, including stability propagation from background patches [3], and reinforced

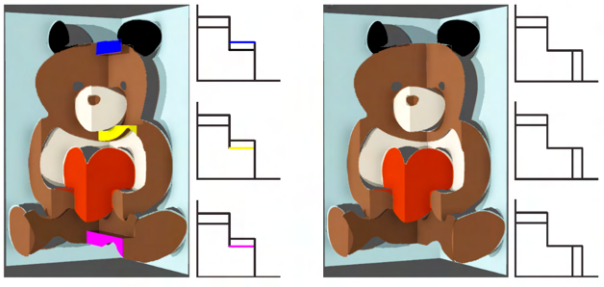


Fig. 4. To ensure stability, (Left) the previous approach [4] reinforces the structure by introducing extra cut and fold lines (blue/yellow/pink patch), which damages shape semantics. (Right) A novel condition for stability is introduced to respect the shape semantics of individual patches as well as the interrelation between patches.

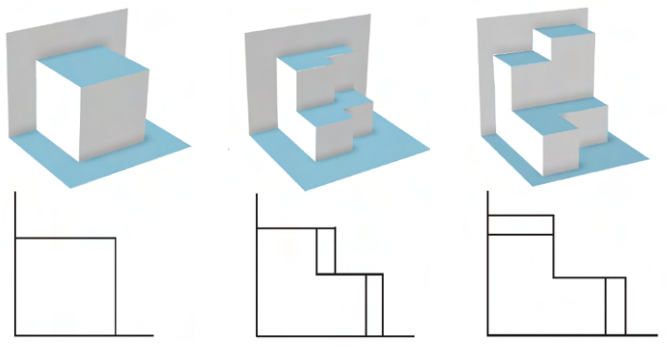


Fig. 5. (Top) Illustration of different sufficient conditions for stability. (Bottom) The corresponding side views.

stability based on a special doubly-connected structure between two patches with the same parity [4]. In general, we can directly enforce these conditions in our system to guarantee the stability of a foldable pop-up plan. However, we found that such enforcement can easily introduce extra fold and cut lines, thus damage the overall input shape in the pop-up structure (see Figure 4-Left). On the other hand, by observing the pop-up designs crafted by artists, we discover a novel sufficient condition for stability that better preserves shape semantics of individual patches, and the spatial relations between patches (see Figure 4-Right). In the following, we present the sufficient conditions on stability used in our system. Please refer to previous works for the details of those adopted conditions.

1) A patch is stable if it is connected to two parallel, non-coplanar stable patches (Proposition 2 and Figure 4a in [3]).
2) A patch is stable if it is connected to a stable patch and another patch that is further connected a stable patch (Proposition 2 in [3], illustrated in Figure 5-Left).
3) A patch is stable if it is on a B-path or F-path. (Proposition 2 in [4], illustrated in Figure 5-Middle).
4) A patch is stable if it is directly doubly-connected with a stable patch, and connected with another patch that is directly doubly-connected with another stable patch (see Figure 5-Right).

To prove the proposed condition 4, we first define the notations as shown in the inset. Let $\mathbf{l}_E = (x_E, 0)$ represent fold line $f_E(x = x_E, y = 0)$ in 3D, $\mathbf{l}_S = (0, y_S)$ represent fold line $f_S(x = 0, y = y_S)$ in 3D, and w_i be the width of patch p_i. The term $\mathbf{u}$ represents the unit vector orthogonal to p_E, $\mathbf{v}$ represents the unit vector orthogonal to p_S. We also define $\mathbf{d}_i$ to be the unit vector lies in p_i that is aligned to $\mathbf{u}$ or $\mathbf{v}$ in a parallel configuration where the patches are parallel to p_S or p_E (one such configuration with $\theta = 90°$ is shown in the inset). Then for parallel configuration we have:

$$\mathbf{l}_S = \mathbf{l}_E + w_1\mathbf{u} - w_2\mathbf{v} + w_3\mathbf{u} - w_4\mathbf{v}. \quad (6)$$

On the other hand, in a general configuration where patches may not be parallel to p_S or p_E, and $\mathbf{d}_i$ may no longer be aligned to $\mathbf{u}$ or $\mathbf{v}$, we have:

$$\mathbf{l}_S = \mathbf{l}_E + w_1\mathbf{d}_1 - w_2\mathbf{d}_2 + w_3\mathbf{d}_3 - w_4\mathbf{d}_4. \quad (7)$$

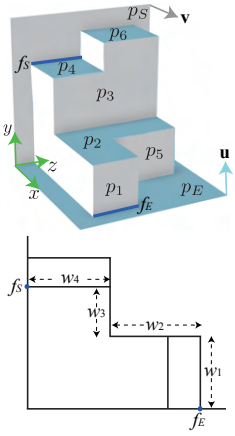
Note that here for simplicity we omit the z coordinate ($z = 0$) of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{d}_i$.

By equating the right hand side of the above two equations, we can derive:

$$w_1\mathbf{u} - w_2\mathbf{v} + w_3\mathbf{u} - w_4\mathbf{v} = w_1\mathbf{d}_1 - w_2\mathbf{d}_2 + w_3\mathbf{d}_3 - w_4\mathbf{d}_4 \quad (8)$$

Because p_2 is doubly-connected to stable patch p_E, $\mathbf{d}_2$ is always parallel to to $\mathbf{v}$. Similarly, $\mathbf{d}_3$ is always parallel to $\mathbf{u}$ as p_3 is doubly-connected to stable patch p_S. Substituting $\mathbf{d}_2 = \mathbf{v}$ and $\mathbf{d}_3 = \mathbf{u}$ into the above equation, we have:

$$w_1\mathbf{u} - w_4\mathbf{v} = w_1\mathbf{d}_1 - w_4\mathbf{d}_4 \quad (9)$$



As $\mathbf{u}, \mathbf{v}, \mathbf{d}_1, \mathbf{d}_4$ are all unit vectors, the above equality holds if and only if $\mathbf{d}_4 = \mathbf{v}$ and $\mathbf{d}_1 = \mathbf{u}$, which means p_1 and p_4 are always parallel to stable patches p_S and p_E, respectively, thus p_1 and p_4 are stable. In the same way, we can also prove that both p_5 and p_6 are stable, therefore the whole structure is stable.

4 Algorithms

4.1 Overview

Figure 6 illustrates the overview of our algorithm. Given an input image containing a 2D shape of interest, our system generates a physically realizable pop-up design through the following steps. In a preprocessing step, the 2D shape is divided into meaningful image parts that respect the shape semantics (Section 4.2). The system then leverages the shape of image parts and the user specified main fold line (located in the middle by default) to generate a set of candidate fold lines with initial patches bounded in-between (Section 4.3). Utilizing Proposition 3.2, we model the foldability and connectivity of the 2D paper pop-up plan, and optimize the fold line properties (validity/parity/location) accordingly via a mixed-integer programming (MIP) (Section 4.4). Stability is ensured afterwards by exploring the design space spanning foldable configurations, and verifying stability conditions listed in Section 3.4 (Section 4.5). The final pop-up plan is constructed by altering the shape of image parts to match the locations of optimized fold lines using a constrained curve deformation method (Section 4.6).

4.2 Pre-processing

Our system takes input as either a raster or vector image containing a foreground object. We first employ simple color thresholding or advanced image matting techniques [48] to extract the 2D shape of the foreground object. Then the 2D shape is segmented into meaningful image parts in order to preserve shape semantics in the final popped-up structure. For vector images, this can be done in a straightforward manner by directly using the color block information encoded in the image file. For raster images, we first obtain an over-segmentation by performing a conventional Watershed algorithm, followed (optionally) by grouping segments into an image part with user assists if needed. Note that for some input images where the 2D shape is depicted with thin-black outlines, we simply merge these outlines into the adjacent

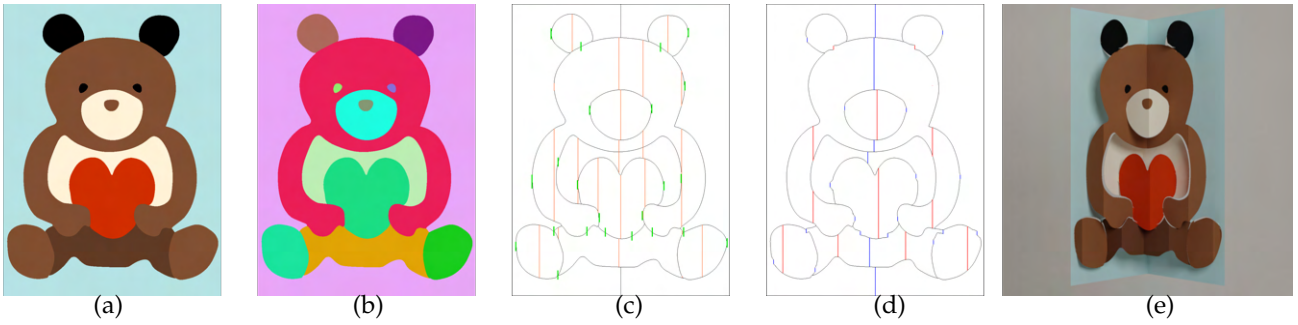


Fig. 6. Algorithm overview. (a) Input 2D image; (b) The semantic parts generated by segmenting the input image using color information; (c) According to the shape semantics, a set of candidate fold lines are generated, including boundary and inner fold lines highlighted in green and orange, respectively; (d) The resultant pop-up plan is generated automatically by optimizing the properties (e.g., validity, parity, location) of the candidate fold lines; (e) The 3D structures popped up from 2D plan.

image parts to explicitly define the spatial relationships between image parts. An example of pre-processing can be found in Figure 6b.

Note that for small image parts entirely lies in another part, we call them 'island' patches (see the eyes and nose in the pig example in Figure 7). Based on the observation of real designs created by artists (see Figure 2), we merge each island patch into its surrounding neighbour, such that it represents a decorative pattern or might be hollowed out in the final pop-up plan.

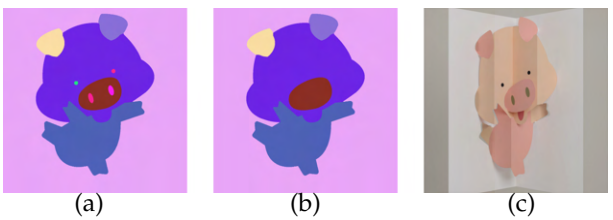


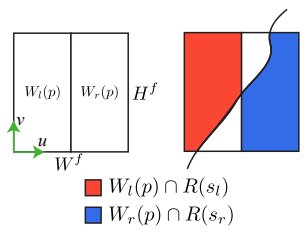
Fig. 7. (a) Original segmentation. (b) We merge island patches into adjacent neighbours before optimization (e.g. eyes of the pig are merged into the head part). (c) The texture of island patches will be preserved in the final result.

4.3 Generating Candidate Fold Lines

Given the extracted image parts and a specified main fold line, a valid pop-up plan can be obtained by placing a proper set of fold lines at proper locations on the image plane. We call such a set of fold lines the fold line configuration. Apparently, without imposing any restrictions, there are numerous fold line configurations. To make the problem tractable and preserve the shape semantics in the final pop-up plan, we propose to exploit the shapes of image parts to generate a set of candidate fold lines that will be used in the later optimization stage. The process involves two steps. First, boundary fold lines are generated on boundaries between adjacent image parts. We further introduce inner fold lines within each image part to increase the diversity of the solution space. We elaborate on each step as follows.

Boundary Fold Lines. To initialize boundary fold lines, we first extract boundary curves between neighboring image parts, including boundaries between image parts and the background (see Figure 8a). The extracted curves are usually with complicated shapes. And it is not desirable to place fold line at an arbitrary location along one curve. To ease computation, we initialize a fold line (a line segment parallel to v-axis) by determining its u coordinate and looking up its v range along the boundary curve. Note that multiple fold lines could be initialized along one boundary curve, especially when the curve is long and with a complicated shape. Figure 8b illustrates how we initialize fold lines on a boundary curve. Formally, we first calculate a confidence score for placing a fold line at each potential location on the curve as explained here.

For a boundary fold line, the 'ideal' case would be that it is exactly part of the curve. This means that the curve must contain a straight line parallel to the main fold line, which is usually not the case. Thus we can only estimate a 'preferable' location along the curve. To do this, we define a confidence score to measure the preference over different locations. For each location, we use a local window centered at that location for measurement (see the inset). The size of the window has clear physical meanings. The dimension along v-axis indicates fold line length and has to be long enough to connect two patches robustly. Also, we want to leave enough marginal space for incident patches sharing the fold line, such that the



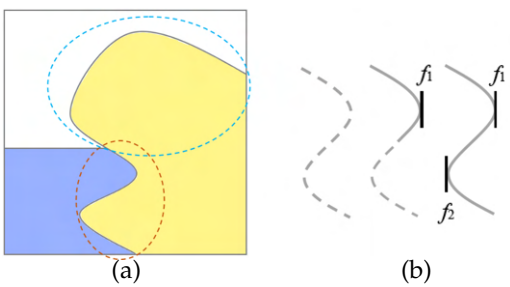


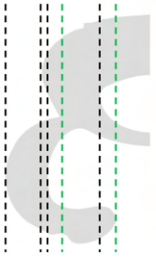
Fig. 8. (a) The boundary of each image part is decomposed according to its neighboring patches. For example, the boundary of the yellow part is divided into two boundary curves shared with the blue and white part, respectively. (b) Initial boundary fold line placement for one boundary curve. First, the location with maximum confidence score is chosen to place a candidate fold line f_1. Then the solid part is identified as the support region for f_1 and excluded. f_2 is initialized similarly for the remaining part. The procedure ends as the entire boundary curve is covered by the support regions of f_1 and f_2.

fold line is easy to fold by hand. As such, the v dimension of the window H^f is chosen to be twice the minimal fold line length, so as the u dimension W^f. For a fold line f placed at location p, suppose the image part on its left is s_l and the part on the right is s_r, the preference score is defined as follows.

$$S(f, p, s_l, s_r) = \frac{|W_l(p) \cap R(s_l)||W_r(p) \cap R(s_r)|}{(0.5H^f W^f)^2} \quad (10)$$

We place a candidate boundary fold line at the location with maximum score, and exclude its local support region along the curve from the subsequent consideration. We repeat the same procedure until the entire boundary curve is covered by support regions or a sufficient number of fold lines are initialized.

Inner Fold Lines. In addition to boundary fold lines which connect image parts at their boundaries, we further introduce inner fold lines inside image parts in order to add more degrees of freedom to enable foldability during the optimization (otherwise the subsequent mixed-integer programming can easily become infeasible).



More specifically, for each image part, we decompose it into sub-regions based on a Reeb-graph like approach. As shown in the inset, we estimate level sets of a 'height' function that indicate topology changes along the height direction of the image part (black/green lines imply topology split/merge). The height direction is always orthogonal to the main fold line. After decomposition, each sub-region corresponds to a bounded area between dashed lines (a branch in the Reeb graph). We place a candidate inner fold line within each major sub-region with adequate size. Such initialization avoids inconsistent topology change of the pop-up graph when optimizing the locations of inner fold lines within the sub-regions. Note that inner fold lines divide the original image part into multiple patches in the initial 2D plan.

4.4 Optimization

Given an initial 2D plan and its corresponding graph G, we optimize the geometry of the graph elements (i.e., properties of individual candidate fold lines and the patches in-between) and the topology of the graph (validity of the candidate fold lines), such that the resultant plan represented by the optimized graph is *foldable*. First, we define a set of variables to represent the properties of each candidate fold line f_i and each patch p_m as follows.

- *fold line validity*: binary variable $a_i = 1$ indicates that f_i is valid (i.e., f_i will actually be folded in the final pop-up plan), otherwise f_i is invalid when $a_i = 0$.
- *patch parity*: binary variable $o_m = 1$ indicates that p_m is odd (i.e., p_m is parallel to the background patch p_E with $y = 0$), otherwise $o_m = 0$ indicates p_m is even (i.e., parallel to the the other background patch p_S with $x = 0$).
- *fold line parity*: binary variable $e_i = 1$ indicates that the corresponding (directed) edge of f_i in the pop-up graph heads to a node that is an odd patch, otherwise $e_i = 0$ indicates that the corresponding edge heads to a node that is an even patch.
- *fold line location*: float variable u_i indicates the location of the fold line f_i satisfying $u = u_i$ in the 2D pop-up plan.

Based on the discussion in Section 3 (Proposition 3.2), the 2D pop-up plan is foldable if and only if it satisfies parity and regularity. These conditions are formulated into parity and regularity constraints as described next. Note that during the optimization, we also ensure the connectivity and the balance of the pop-up plan by adding connectivity constraints and balance constraints.

4.4.1 Constraints

Parity Constraints. According to Definition 3.2, we add parity constraints to all the candidate fold lines (i.e., boundary and inner fold lines) such that the parity is consistent for any two consecutive fold lines along any path connecting p_S and p_E in the pop-up graph G. Suppose f_i and f_j are two consecutive edges (i.e., f_i is one of incoming fold lines and f_j is one of outgoing fold lines of a patch), we have the following constraint **P1**.

$$\begin{aligned} e_i &= 1 - e_j, \quad \text{if } a_j = 1 \\ e_i &= e_j, \qquad \text{otherwise} \end{aligned} \quad (11)$$

In the above, if a candidate fold line is invalid, the constraint is to make sure its parity is the same as its ascendant, such that the overall parity can be consistent over the whole configuration. For all the candidate fold lines incident to p_S or p_E, the parity is set to odd by default.

Apart from fold lines, the parity of patches also needs to be considered. Suppose p_m and p_n are two adjacent patches connected by a common fold line f_i, the following constraint **P2** needs to be satisfied.

$$\begin{aligned} o_m &= 1 - o_n, \quad \text{if } f_i \text{ is valid} \\ o_m &= o_n, \qquad \text{if inner fold line } f_i \text{ is invalid} \end{aligned} \quad (12)$$

Note that no constraint is added if f_i is an invalid boundary fold line as a cut line will be generated therein along the boundary. Also, the parity of patches p_S and p_E are respectively set as even and odd by default.

Regularity Constraints. In addition to parity constraints, we also enforce regularity constraints to all candidate fold lines and patches to enable foldability. To simplify the formulation. We add auxiliary float variables x_i and y_i to indicate the location of $f_i(x = x_i, y = y_i)$ in 3D, and auxiliary float variable d_m to indicate the underlying plane equation of a patch p_m. If p_m is an odd patch, the plane is $y = d_m$, otherwise $x = d_m$. The regularity constraints consist of three parts as follows.

The first part **R1** enforces the fold line location consistency between 2D pop-up plan and 3D orthogonal configuration. Based on Corollary 3.2, each valid fold line should satisfy:

$$x_i - y_i = u_i \quad (13)$$

The second part **R2** ensures the correct 3D spatial relation between patches and fold lines. Suppose a valid fold

line f_i connects patch p_m and p_n in sequence, then it should satisfy the following constraints:

$$\begin{aligned} y_i = d_m,\ x_i = d_n, &\quad \text{if } p_m \text{ is odd} \\ x_i = d_m,\ y_i = d_n, &\quad \text{if } p_m \text{ is even} \end{aligned} \tag{14}$$

Also, fold lines f_i and f_j that lie on the same odd/even patch p_m should share the same y/x coordinate, and the shared coordinate is determined by the plane equation of p_m. For example, if two fold lines lie in an odd patch with $y = d_m$, then $y_i = d_m$ and $y_j = d_m$.

The third part **R3** restricts the relative locations between all candidate fold lines such that the relative locations of consecutive fold lines (within each path from p_S to p_E in the pop-up graph G) are consistent between 2D and 3D. Suppose f_i and f_j are two consecutive fold lines in the pop-up graph (i.e., f_i is one incoming fold line of a patch p_m while f_j is one outgoing fold line), the following constraints restrict their topology order after optimization in 3D.

$$\begin{aligned} y_i > y_j, &\quad \text{if } p_m \text{ is even} \\ x_i < x_j, &\quad \text{if } p_m \text{ is odd} \end{aligned} \tag{15}$$

Connectivity Constraints. In Definition 3.3, the connectivity property of a pop-up plan guarantees that all patches, including two background patches p_S and p_E, are connected. In our case, we have two types of candidate fold lines, boundary fold line and inner fold line. The connectivity is only affected by boundary fold lines. This is because cut lines can only be introduced if a boundary fold line is invalid and the two adjacent patches will be disconnected. To ensure connectivity, for initial image parts generated from the segmentation, we add the followings constraints **C1** to its boundary fold lines:

$$\begin{aligned} \sum_{m \in \mathbb{E}_{bin}} a_m \geq 1, \\ \sum_{n \in \mathbb{E}_{bout}} a_n \geq 1, \end{aligned} \tag{16}$$

where $\mathbb{E}_{in}$ indicates all incoming boundary fold lines, $\mathbb{E}_{out}$ indicates all outgoing fold lines for an image part. The above constraints guarantee that at least one incoming/outgoing boundary fold line is valid for each image part. As invalid inner fold lines do not affect the connectivity, all patches are ensured to be connected.

Balance Constraints. In addition to the above constraints that ensure the physical validity of the resultant pop-up plan, we also introduce additional balance constraints to control both the number and location of valid inner fold lines, making the result more plausible. There are two types of balance constraints to control the number of valid inner fold lines in each image part, and the position of valid inner fold lines, respectively. Note that we do not consider boundary fold lines here because they will be formulated in the objectives later.

In practice, we initialize a set of candidate inner fold lines as a complement to boundary folder lines to allow a feasible pop-up plan optimization. However, generating too many valid inner fold lines in an image part can easily lead to an overly zig-zag folding structure, which is largely inconsistent with the original image semantics. Thus we use **B1** to limit the number of valid inner fold lines of each image part:

$$\sum_{k \in \mathbb{E}_{inner}} a_k \leq \sum_{m \in \mathbb{E}_{bin}} a_m + \sum_{n \in \mathbb{E}_{bout}} a_n, \tag{17}$$

where we inherit the same notations from the connectivity constraints for $\mathbb{E}_{bin}$ and $\mathbb{E}_{bout}$, and $\mathbb{E}_{inner}$ indicates all candidate inner fold lines for an image part.

Apart from their number, we also realize that the balanced distribution of valid inner fold lines for each image part and for the whole input are also important. For symmetric image part or input, the balance of inner fold line locations can help preserve shape symmetry. For the non-symmetric image part or input, this prevents the optimized patches from having uneven widths in the resultant structure. We employ **B2** that comprises two components to achieve this, including a local component for balanced fold line distribution of an image part, and a global component for the balanced distribution of the whole input as follows:

$$\begin{aligned} |\sum_{l \in \mathbb{E}_{local}} (u_l - c_p)| \leq \lambda w_i, \\ |\sum_{g \in \mathbb{E}_{global}} (u_g - c_i)| \leq \lambda w_i, \end{aligned} \tag{18}$$

where c_p and c_i denote the center of an image part and the whole input, w_i is the width of the input image and the weight $\lambda = 0.1$ by default in our implementation.

4.4.2 Objectives

In our optimization, one of the objectives is to make the 2D shape consistent with its approximation in the pop-up plan. In particular, we try to avoid large shape variation caused by relocating fold lines. For simplicity, we use $(u_i - u_i^0)^2$ to indicate the shape variation due to the relocation of boundary fold line f_i, where u_i^0 denotes the initial location of f_i on an image part boundary in 2D. And we optimize the shape consistency using the following energy term:

$$E_{consist} = \sum_{f_i \in \mathbb{F}_{bdry}} (u_i - u_i^0)^2, \tag{19}$$

where $\mathbb{F}_{bdry}$ contains all the boundary fold lines. Note that we only consider boundary fold line in $E_{consist}$ since it is shared by two image parts and would affect their shapes in 2D (the shapes will be deformed in the post-processing stage as in Section 4.6), whereas inner fold line has no such influences.

4.4.3 MIP Formulation

Based on the constraints and objectives defined above, the overall optimization is formulated as a mixed-integer programming:

$$\begin{aligned} &\min_{a_i, e_i, u_i, x_i, y_i, o_m, d_m} E_{consist} \\ &\text{subject to } \mathbf{P1},\ \mathbf{P2},\ \mathbf{R1},\ \mathbf{R2},\ \mathbf{R3},\ \mathbf{C1},\ \mathbf{B1},\ \mathbf{B2} \end{aligned} \tag{20}$$

By solving the MIP, we achieve the validity, parity, and balanced location (both in the 2D and 3D domain) of each fold line and the incident patches between fold lines. Thus the final pop-up plan can be formed subsequently.

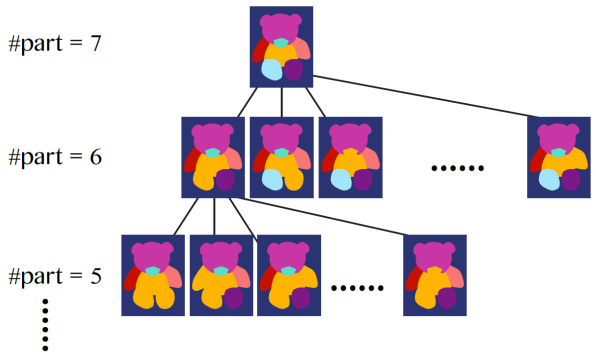


Fig. 9. We merge image parts in a bottom-up manner to explore the pop-up design space with different number of image parts.

4.5 Design Exploration for Stability

In previous work [3], [4], the stability is enforced after a foldable plan is generated. The basic idea is to revise the foldable structure according to the sufficiency conditions on stability. We apply a similar strategy to ensure stability based on the optimization results. The difference is that instead of revising the foldable structure, we explore the pop-up design space by changing the image part combinatorics (see Figure 9). This is based on an observation that, pop-up plan with fewer image parts can not only simplify the optimization, but also avoid visual artifacts introduced by foldable structure modification.

The exploration is performed in a bottom-up manner as follows. We first optimize all candidate fold lines initialized from all image parts as in Section 4.4 to generate a foldable pop-up plan. Then we verify the stability of the plan according to the sufficiency conditions in Section 3.3. If the stability is not satisfied, we iteratively merge image parts and perform optimization and stability verification again. The whole process ends when all the image parts are merged into one part. It is easy to see that foldability and stability can be achieved (with the simplest single stair-like popup) if there is only one image part. This means our algorithm has the ability to ensure a physically plausible pop-up plan after exploration. And thanks to the exploration in the design space, we provide the user with a set of valid pop-up plans to choose from. Figure 10 shows a number of user selected pop-up designs by exploring the design space of the same input image.

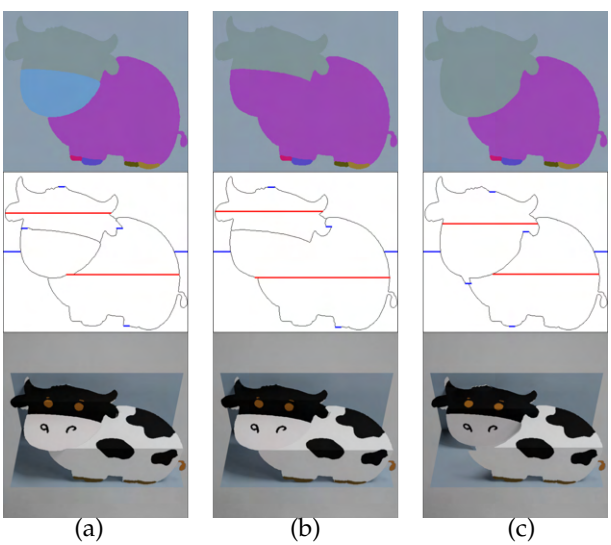


Fig. 10. Our framework can generate plausible pop-up designs with different layouts from the same input image by exploring the design space, and reach a stable solution (c).

4.6 Post-processing

Although our optimization tries to preserve the initial location of each boundary fold line, small variation can still be introduced to the optimized location, leading to misalignment between the fold line and the boundary which prevents compact 2D pop-up plan generation (see Figure 11a). To solve this problem, we further employ curve deformation to align a boundary to the optimized fold line (see Figure 11b) in a way that (i) the two endpoints of the optimized fold line lie on the boundary; and (ii) the deformed boundary preserves its original shape. More specifically, we first project the two endpoints of the optimized fold line onto the original boundary. Suppose the two projection points are $\mathbf{v}_1$ and $\mathbf{v}_2$, we apply a 2D rigid curve deformation (an adaption of the state-of-the-art MLS image deformation [49]) under the constraint that $\mathbf{v}_1$ and $\mathbf{v}_2$ coincide with the two endpoints. Based on the aligned fold line and its corresponding boundary, we can easily generate a compact foldable pop-up plan (see Figure 11c).

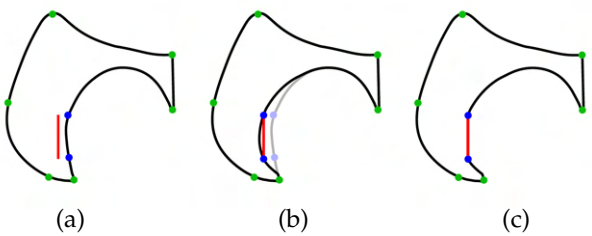


Fig. 11. We employ a constrained curve deformation to align the boundary to its optimized boundary fold line to generate a compact pop-up plan. (a) The optimized boundary fold line may not align with the associated boundary; (b) A constrained curve deformation is performed for the alignment with blue and green points represent moving and fixed control anchors, respectively; (c) Compact 2D pop-up plan with cut and fold lines in black and red, respectively.

5 Evaluation

Results. We evaluate our image-based paper pop-up design framework on various 2D images. Figure 17 shows a gallery of results, including automatically generated 2D pop-up plans and the corresponding 3D structures with 90° fold angles. We also demonstrate the physical validity of the resultant pop-up plans by generating animated pop-up sequences (see the supplemental video) and real fabrications in practice (see Figure 12). The results have validated the effectiveness of our framework.

Besides generating automatic results, our framework provides a set of interactive tools to involve both novices and professionals in the creative design process (see also the supplemental video). We allow the user to (i) adjust the location of a fold line; (ii) specify the validity of a fold

Fig. 12. Our framework generates physically valid paper pop-up designs that can be fabricated in practice.

line; (iii) specify the parity of a fold line. Figure 13 shows an example of automatic and interactive results generated from the same image. Note that too many prescribed parity constraints may cause conflict, leading to infeasible optimization flagged by our MIP solver. But this rarely happens in practice since the user only needs to specify very few parity constraints and gains satisfactory results.

Comparisons. Since no prior work addresses image-based paper pop-up generation, we only compare our results with manually generated designs [7] as shown in Figure 14. To make a fair comparison, depending on the quality of the input, we either use the 2D sketch image (the 'heart' example) or color the pop-up plan (the 'bear' example) provided by the artist as the input of our algorithm. Our framework can generate valid paper pop-ups (see Figure 14) efficiently (Bear fully automatic within a few seconds and Heart within 3 mins with user interaction to enforce symmetry). The results are close to manual results but without the cumbersome trial-and-error process for placing fold lines which often takes hours or even days to polish, depending on the properties of the shape such as symmetry, size, proportion, etc. Note that despite the automation of our framework, a few simple interactions on fixing the location

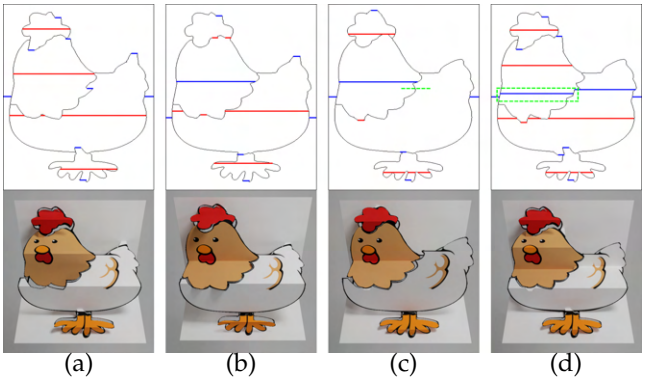


Fig. 13. Our framework not only enables automated result (a), but also allows simple user specifications to generate interactive results: (b) The user specifies the position of the main fold line. (c) The user specifies that the fold line in the green dash line is invalid. (d) The user specifies that the fold line in the green dash rectangle is valid and odd.

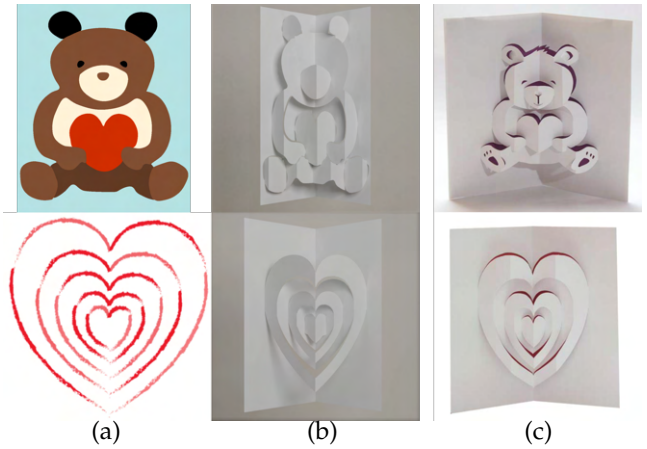


Fig. 14. Comparison between our results and manual results generated by artist. (a) Input images; (b) Our results; (c) Manual results in [7].

and validity of candidate fold lines (see Table 1) are applied here to mimick manual design preferences (e.g., symmetry). More discussions on design choices are included at the end of this section.

TABLE 1
A summary of user interactions for producing some of the results in our work, including the number of fold lines affected and the interaction type. Interaction types include location, validity and parity.

figure ID	2D shape	# fold lines	interaction type
Figure 1	MARIO	4	location
Figure 13b	HEN	1	location
Figure 13c	HEN	1	validity
Figure 13d	HEN	1	validity/parity
Figure 14	HEART	5	validity
Figure 16	REINDEER1	1	validity

Ablations. To validate the effectiveness of our image-based paper pop-up optimization framework, we perform ablation studies on optimization initialization and constraints, as shown in Figure 15. We demonstrate two examples here, including the Goat and the Reindeer2. The final resulting pop-up plan (odd fold lines in blue while even fold lines in red) is shown in Figure 15a. In contrast, without initializing inner fold lines (see Figure 15b), it is not feasible to achieve a foldable solution only based on boundary fold lines (MIP cannot resolve the validity of the fold line shared by the green path and the purple path for Goat, or the green path for Reindeer2). Without parity constraints (see Figure 15c), not all patches are foldable due to the conflicting parity of fold lines (e.g., the yellow image part in both examples contains consecutive fold lines with the same parity, thus is not foldable. Without regularity constraints (see Figure 15d), positions of fold lines can violate foldability (e.g., fold lines in the yellow image parts result in unfoldable structures). Without connectivity constraints (see Figure 15e), no patch lies in a connected path from p_S to p_E in the pop-up graph. The image parts can even be all isolated in such a situation. Without balance constraints (see Figure 15f), there can be unbalanced locations and numbers of fold lines distributed either in a local region or across the main fold line, as highlighted in the Goat and Reindeer2 examples, respectively.

Implementation Details and Performance. Our framework is implemented using C++ on a desktop machine with

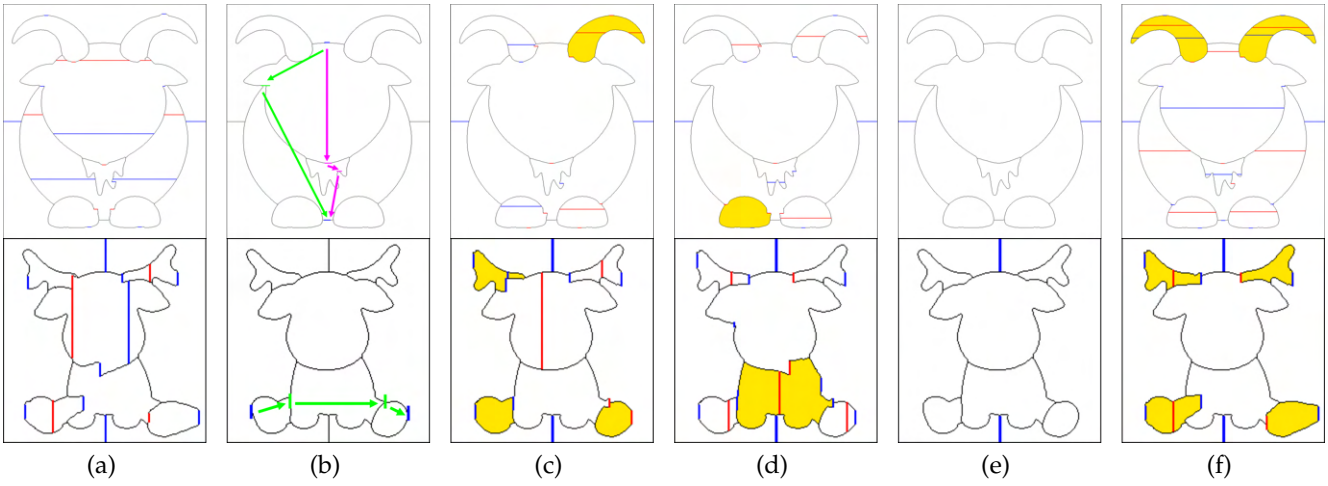


Fig. 15. Ablation comparisons of Goat and Reindeer2 to verify the pop-up plan optimization. (a) full model (Equation 20); (b) initialization w/o inner fold lines; (c) model w/o parity constraint; (d) model w/o regularity constraint; (e) model w/o connectivity constraint; (f) model w/o balance constraint.

TABLE 2
A summary of computational performance statistics for all automatic results shown in Figure 17. We show the computational time (in seconds) for initialization, optimization, and post-processing together with the number of image parts, and the number of initialized boundary fold lines (BFL) and inner fold lines (IFL).

2D shape	init.(s)	opt.(s)	post.(s)	# image parts	# init. BFL	# init. IFL
BEAR1	0.114	1.622	0.276	18	21	9
BEAR2	0.158	3.769	0.410	28	24	17
BABY1	0.098	0.595	0.188	15	13	8
BABY2	0.136	4.196	0.272	20	18	12
HEN	0.080	1.189	0.169	19	11	13
PIG	0.100	0.308	0.248	16	14	9
GOAT	0.117	1.107	0.330	24	19	15
PENGUIN	0.112	0.760	0.157	14	11	8
COW	0.065	0.333	0.117	12	9	7
FROG	0.104	0.561	0.207	18	15	10
MAN1	0.088	1.272	0.236	24	18	15
MAN2	0.073	0.496	0.181	18	15	11
REINDEER1	0.061	0.722	0.168	15	11	9
REINDEER2	0.100	1.238	0.236	21	15	13
BIRD	0.147	1.879	0.614	19	13	12
DOLPHIN	0.085	1.361	0.199	14	11	9

a hex-core AMD Ryzen 5 CPU (3.8GHz) and 16GB RAM. The MIP is solved using a branch-and-bound algorithm [50] implemented in the Gurobi solver [51]. The detailed timings performance is reported in Table 2. All results are generated within 10 seconds. The major consumption is the MIP optimization due to the complex nature of optimizing integer numbers and real values at the same time. As Gurobi uses relaxation and branch-and-bound strategies to solve the mixed-integer problem, the optimization time is only roughly proportional to the complexity of 2D shape semantics and 2D pop-up plan, which are represented (in a way) by the number of image parts and fold lines, respectively.

Discussions. Although our framework can generate physically valid pop-up structures directly from 2D shapes, the lack of underlying 3D geometry information would lead to results that do not coincide well with the 3D shape in regular human impression (see Figure 16a). Our current system allows the user to prescribe constraints such as validity and parity to adjust the convexity/concavity of the popped-up structure locally (see Figure 16b). However, the strong physical constraints may still cause shape variation/distortion. A similar issue is also reported in 3D-based methods [3], [4]. We also noticed that for manual

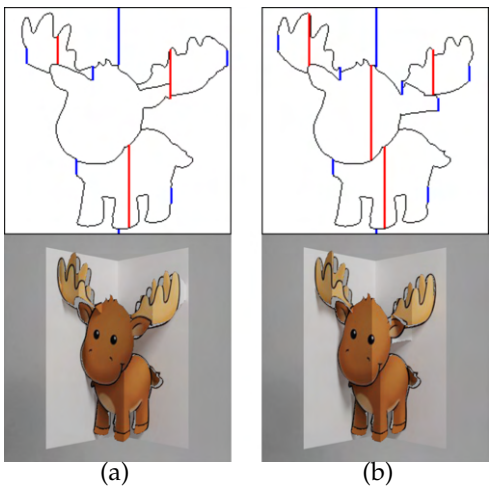


Fig. 16. (a)The automatic result may produce flat shapes (e.g., reindeer's face), which do not coincide well with the underlying 3D shape in regular human impression. (b) The user could simply specify only a few inner fold line properties to adjust the popped up geometry.

designs created by artists, symmetric fold lines (w.r.t. the main fold line or center of image parts) are often utilized as an additional design choice to particularly improve the aesthetic of results (see Figure 14). This observation motivates us to add balance constraints. However, there exists a trade-off between plausibility and foldability. In some cases, a foldable structure can only be generated at the cost of aesthetics (e.g., adding additional inner fold lines) due to the nature of the input. Further, our work can be easily extended to handle multiple shape pop-ups rather than one based on a simple divide-and-conquer approach that fuses the pop-up plans from individual shapes in 2D (see Figure 1 for an example of popping up two shapes).

Lastly, our work focuses more on the geometry and topology of the paper pop-up plan. Physical properties are addressed through foldability and stability. In practice, other physical conditions such as paper thickness and construction solutions can also affect the design choices, such as the minimum fold line length (currently set as a parameter proportional to the image size in our implementation).

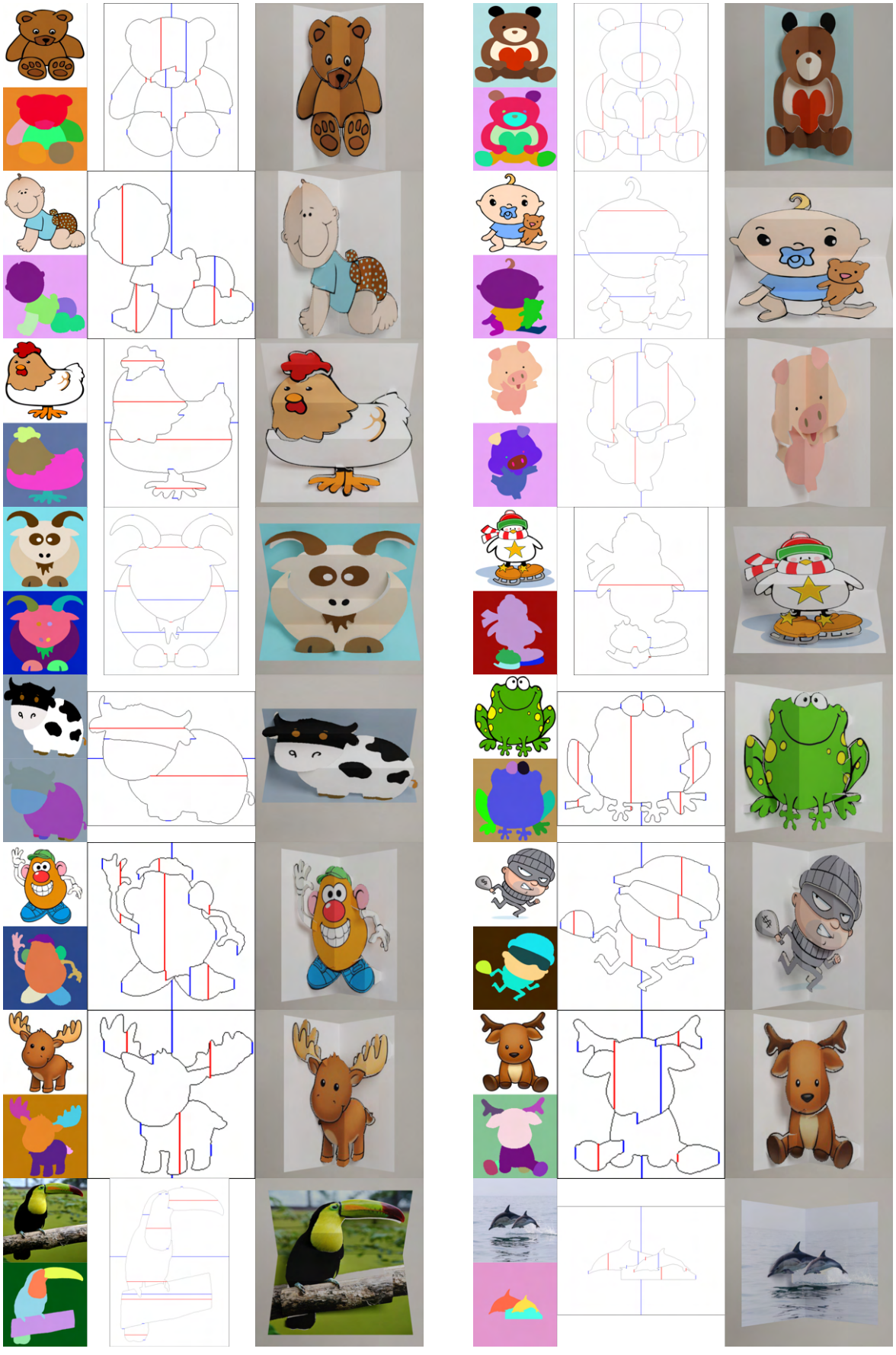


Fig. 17. A variety of plausible pop-up designs generated automatically by our framework. We show the original image, segmentation, the 2D pop-up plan and the illustration of 3D popped up structure respectively.

6 Conclusion and Future Work

In this paper, we present the first computational design framework that can automatically generate OA-style paper pop-up designs from 2D images. This work is inspired by pop-up designs created by artists and utilizes the availability of 2D images over 3D models. Unlike prior works in which the goal is to approximate a given 3D model using the pop-up structure, we optimize the pop-up plan to preserve the semantics of the input image while still allowing physically valid 3D pop-ups. We derive new sufficiency and necessity conditions on foldability in our problem setting and formulate the problem as a constrained topology and geometry optimization. We solve the proposed optimization using mixed-integer programming. In addition, we propose an additional sufficiency condition on stability and a design space exploration strategy while preserving shape semantics. User interactions are also allowed for creative design. We evaluate our framework on various images, and the results demonstrate the effectiveness of our framework.

In the future, we would like to improve the 'perceptual' quality of the results by taking into account 3D geometry priors (e.g., depth) inferred from the input 2D image or specified by the user, such that the popped-up structure accords more with prior human knowledge on 3D geometry. Furthermore, the techniques bridging the gap between 2D and 3D, such as single-view reconstruction and 2D sketch-based modeling, can be involved to constrain the optimization and enhance the results. Also, we would like to further improve the visual quality of the results by considering other aesthetic metrics. We plan to conduct comprehensive user studies by asking novices and professionals to evaluate our results and use our framework. We hope to propose some quantitative measurements that can be used in the optimization to improve automated results and/or to achieve some high-level design guidelines that can be used to improve interactive results. Further, we hope to exploit our work in other application domains. For example, it would benefit the design of customized foldable furniture, which also considers functionalities based on foldable structures. It would be interesting to use our framework to design foldable furniture with different shape/texture contexts, such as a bear chair, a penguin table, etc.

Acknowledgments

The authors would like to thank Kwok Cheung Chow for design inspirations and helpful discussions, and all the anonymous reviewers for their comments and suggestions. The work was funded in part by CAMERA, the RCUK Centre for the Analysis of Motion, Entertainment Research and Applications (EP/M023281/1 and EP/T022523/1), the Ministry of Science and Technology of Taiwan (110-2221-E-007-061-MY3 and 110-2221-E-007-060-MY3), and a gift from Adobe.

References

[1] M. Chatani, *Pop-up Origamic Architecture*. Ondori-Sha Publishers Ltd, 1985.

[2] J. Mitani and H. Suzuki, "Computer aided design for origamic architecture models with polygonal representation," in *Computer Graphics International, CGI 2004*. Crete, Greece: IEEE, 2004, pp. 93–99.

[3] X.-Y. Li, C.-H. Shen, S.-S. Huang, T. Ju, and S.-M. Hu, "Popup: automatic paper architectures from 3d models." *ACM Trans. Graph.*, vol. 29, no. 4, pp. 111–1, 2010.

[4] S. N. Le, S.-J. Leow, T.-V. Le-Nguyen, C. Ruiz, and K.-L. Low, "Surface and contour-preserving origamic architecture paper pop-ups," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 2, pp. 276–288, 2014.

[5] X.-Y. Li, T. Ju, Y. Gu, and S.-M. Hu, "A geometric study of v-style pop-ups: Theories and algorithms," *ACM Trans. Graph.*, vol. 30, no. 4, 2011.

[6] C. R. Ruiz Jr, S. N. Le, J. Yu, and K.-L. Low, "Multi-style paper pop-up designs from 3d models," *Computer Graphics Forum*, vol. 33, no. 2, pp. 487–496, 2014.

[7] K.-C. Chow, "Amazing popup," 2021. [Online]. Available: http://www.amazingpopup.com

[8] E. D. Demaine and J. O'Rourke, *Geometric folding algorithms: linkages, origami, polyhedra*. Cambridge university press, 2007.

[9] J. O'Rourke, *How to fold it: the mathematics of linkages, origami, and polyhedra*. Cambridge University Press, 2011.

[10] T. Tachi, "Origamizing polyhedral surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 2, pp. 298–311, 2010.

[11] J. Solomon, E. Vouga, M. Wardetzky, and E. Grinspun, "Flexible developable surfaces," *Computer Graphics Forum*, vol. 31, no. 5, p. 1567–1576, 2012.

[12] M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann, "Curved folding," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–9, 2008.

[13] C. Tang, P. Bo, J. Wallner, and H. Pottmann, "Interactive design of developable surfaces," *ACM Trans. Graph.*, vol. 35, no. 2, 2016.

[14] M. Rabinovich, T. Hoffmann, and O. Sorkine-Hornung, "Discrete geodesic nets for modeling developable surfaces," *ACM Trans. Graph.*, vol. 37, no. 2, 2018.

[15] ——, "The shape space of discrete orthogonal geodesic nets," *ACM Trans. Graph.*, vol. 37, no. 6, 2018.

[16] C. Jiang, K. Mundilova, F. Rist, J. Wallner, and H. Pottmann, "Curve-pleated structures," *ACM Trans. Graph.*, vol. 38, no. 6, 2019.

[17] T. Tachi, "Interactive form-finding of elastic origami," in *International Association for Shell and Spatial Structures (IASS) Symposium*, 2013.

[18] L. Zhu, T. Igarashi, and J. Mitani, "Soft folding," *Computer Graphics Forum*, vol. 32, no. 7, pp. 167–176, 2013.

[19] M. Kilian, A. Monszpart, and N. J. Mitra, "String actuated curved folded surfaces," *ACM Trans. Graph.*, vol. 36, no. 3, pp. 1–13, 2017.

[20] M. Konaković, K. Crane, B. Deng, S. Bouaziz, D. Piker, and M. Pauly, "Beyond developable: Computational design and fabrication with auxetic materials," *ACM Trans. Graph.*, vol. 35, no. 4, 2016.

[21] M. Konaković-Luković, J. Panetta, K. Crane, and M. Pauly, "Rapid deployment of curved surfaces via programmable auxetics," *ACM Trans. Graph.*, vol. 37, no. 4, 2018.

[22] G. P. T. Choi, L. H. Dudte, and L. Mahadevan, "Programming shape using kirigami tessellations," *Nature Materials*, vol. 18, pp. 999–1004, 2019.

[23] C. Jiang, F. Rist, H. Pottmann, and J. Wallner, "Freeform quad-based kirigami," *ACM Trans. Graph.*, vol. 39, no. 6, 2020.

[24] J. Xu, C. S. Kaplan, and X. Mi, "Computer-generated paper-cutting," in *The 15th Pacific Conference on Computer Graphics and Applications, PG 2007*. Maui, HI, USA: IEEE, 2007, pp. 343–350.

[25] Y. Li, J. Yu, K.-l. Ma, and J. Shi, "3d paper-cut modeling and animation," *Computer Animation and Virtual Worlds*, vol. 18, no. 4-5, pp. 395–403, 2007.

[26] J. McCrae, K. Singh, and N. J. Mitra, "Slices: a shape-proxy based on planar sections," *ACM Trans. Graph.*, vol. 30, no. 6, p. 168, 2011.

[27] K. Hildebrand, B. Bickel, and M. Alexa, "crdbrd: Shape fabrication by sliding planar slices," *Computer Graphics Forum*, vol. 31, no. 2pt3, pp. 583–592, 2012.

[28] P. Cignoni, N. Pietroni, L. Malomo, and R. Scopigno, "Field-aligned mesh joinery," *ACM Trans. Graph.*, vol. 33, no. 1, pp. 1–12, 2014.

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

[30] Y.-J. Liu, Y.-K. Lai, and S. Hu, "Stripification of free-form surfaces with global error bounds for developable approximation," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 4, pp. 700–709, 2009.

[31] D. Julius, V. Kraevoy, and A. Sheffer, "D-charts: Quasi-developable mesh segmentation," *Computer Graphics Forum*, vol. 24, no. 3, pp. 581–590, 2005.

[32] I. Shatz, A. Tal, and G. Leifman, "Paper craft models from meshes," *The Visual Computer*, vol. 22, no. 9, pp. 825–834, 2006.

[33] F. Massarwi, C. Gotsman, and G. Elber, "Papercraft models using generalized cylinders," in *The 15th Pacific Conference on Computer Graphics and Applications, PG 2007*. Maui, HI, USA: IEEE, 2007, pp. 148–157.

[34] C. C. L. Wang and K. Tang, "Achieving developability of a polygonal surface by minimum deformation: A study of global and local optimization approaches," *The Visual Computer*, vol. 20, no. 8–9, p. 521–539, 2004.

[35] O. Stein, E. Grinspun, and K. Crane, "Developability of triangle meshes," *ACM Trans. Graph.*, vol. 37, no. 4, 2018.

[36] A. Ion, M. Rabinovich, P. Herholz, and O. Sorkine-Hornung, "Shape approximation by developable wrapping," *ACM Trans. Graph.*, vol. 39, no. 6, 2020.

[37] A. Glassner, "Interactive pop-up card design. 1," *IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 79–86, 2002.

[38] ——, "Interactive pop-up card design. 2," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 74–85, 2002.

[39] S. L. Hendrix, M. Eisenberg *et al.*, "Computer-assisted pop-up design for children: computationally enriched paper engineering," *Advanced Technology for Learning*, vol. 3, no. 2, pp. 119–127, 2006.

[40] J.-m. Chen and Y.-z. Zhang, "A computer-aided design system for origamic architecture," in *International Conference on Supercomputing*, 2006.

[41] C. Ruiz, S. N. Le, and K.-L. Low, "Generating animated paper pop-ups from the motion of articulated characters," *The Visual Computer*, vol. 31, no. 6, pp. 925–935, 2015.

[42] N. Xiao, Z. Zhu, R. R. Martin, K. Xu, J.-M. Lu, and S.-M. Hu, "Computational design of transforming pop-up books," *ACM Trans. Graph.*, vol. 37, no. 1, 2018.

[43] Y. Horry, K. Anjyo, and K. Arai, "Tour into the picture: using a spidery mesh interface to make animation from a single image," in *The 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*. Los Angeles, CA, USA: ACM, 1997, pp. 225–232.

[44] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 577–584, 2005.

[45] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," *International Journal of Computer Vision*, vol. 76, no. 1, p. 53–69, 2008.

[46] J. Y. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing*, vol. 3, no. 5, p. 625–638, 1994.

[47] L. Zhao, M. Hansard, and A. Cavallaro, "Layered scene models from single hazy images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 7, pp. 2167–2179, 2018.

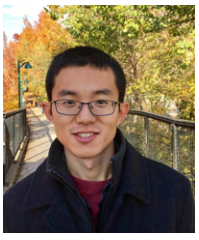
[48] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, "A global sampling method for alpha matting," in *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011*. Colorado Springs, CO, USA: IEEE, 2011, pp. 2049–2056.

[49] S. Schaefer, T. McPhail, and J. Warren, "Image deformation using moving least squares," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 533–540, 2006.

[50] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*. John Wiley & Sons, 1999, vol. 55.

[51] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021. [Online]. Available: http://www.gurobi.com

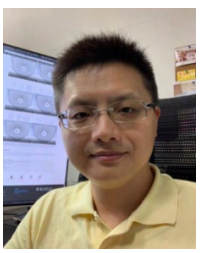
Fei Huang is a second-year PhD candidate in the Department of Computer Science at University of Bath, United Kingdom. She received the MS degree in computer graphics, vision and imaging from University College London, United Kingdom. She is working on computer graphics, particularly in geometry optimization.



Chen Liu is an applied research scientist at Meta - Facebook Reality Labs. He received his PhD degree from Washington University in St. Louis in 2019. His research interests include 3D vision and scene understanding. He is particularly interested in geometry reasoning using learning techniques.

Kai-Wen Hsiao received the MS degree in computer science from National Tsing Hua University, Taiwan, in 2018. Now, He is pursuing his PhD degree at Department of Computer Science, National Tsing Hua University. His research interests include computer graphics, 3D modeling and photorealistic reandering.

Ying-Miao Kuo received the MS degree in computer science from National Tsing Hua University, Taiwan, in 2017. Her research interests include computer graphics, non-photorealistic rendering, and recreational graphics.



Hung-Kuo Chu is an associate professor at the Department of Computer Science, National Tsing Hua University. He received BS and PhD degrees from Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests focus on scene understanding, shape analysis, smart geometry, recreational graphics, and extended reality.

Yong-Liang Yang is a senior lecturer in the Department of Computer Science at University of Bath, United Kingdom. He received the BS degree and PhD degree in Computer Science from Tsinghua University, Beijing, China. His research area is broadly in visual computing, with particular interests in shape modeling, computational design, and interactive techniques.