Forward and inverse D-Form modelling based on optimisation

Fei Huang[a], Caigui Jiang[b,*], Tony Wills[c], Yong-Liang Yang[a]

[a] *University of Bath, Bath, United Kingdom*
[b] *Xi'an Jiaotong University, Xi'an, China*
[c] *Wills Watson+Associates, London, United Kingdom*

Abstract

D-Form is a special piece-wise developable surface formed by aligning the boundaries of two planar domains. It has been widely used in different design scenarios. In this paper, we study how to computationally and intuitively model D-Forms. We present an optimisation-based framework that can efficiently generates D-Form shapes. Our framework can model D-Forms with two approaches based on two different user inputs, including the forward modelling from two given planar domains and, more importantly, the inverse modelling from a given space curve where the planar domains are no longer needed. Our optimisation is devised based on two critical characteristics of D-Forms. Firstly, the constituent developable surfaces of a D-Form are isometrically deformed from planar domains. Secondly, there is a close relationship between a D-Form and the convex hull of its seam. Through extensive evaluation, we demonstrate that our approach can model plausible D-Forms efficiently from various inputs with different geometric properties.

Keywords: D-Form, developable surfaces, isometric deformation, optimisation

1. Introduction

Developable surfaces with zero Gaussian curvature are popular geometric design choices as they can be directly fabricated by bending common materials such as paper and flat metal sheets. The D-Form, obtained by simply conjoining the boundaries of two planar domains with the same perimeter length, is known as an important type of piece-wise developable surfaces. It has been widely used to model shapes in different design contexts such as furniture, industrial object, decoration, etc. (see Fig. 1). Although flat materials are relatively easy to deal with, the form-finding stage still largely depends on the skills and experiences of designers to specify planar domains and manipulate (cut, bend, connect) materials. Satisfactory designs are usually achieved by a long and troublesome trial-and-error process.

To facilitate the design process, researchers have attempted to simulate the D-Form generation computationally from different aspects. Some common D-Form shapes were made from Surface Evolver [3], an interactive physical simulator for modelling liquid surfaces, by adding various forces and constraints [4]. A computational approach was presented to model a particular type of D-Form with constant discrete Gaussian curvature [1]. Optimisations were also employed to generate D-Form shapes from planar domains with various discrete representations [5, 6].

Figure 1: Examples of real D-Form designs, including mechanical part [1], decorative furniture [2], and art work [2] created by designers.

However, all previous works merely follow the traditional design pipeline, i.e., define planar domains in 2D and then generate the corresponding D-Form in 3D. Moreover, to change the 3D D-Form shape, the 2D domains need to be updated beforehand. Hence it is difficult to either expect or control the shape of the resultant D-Form, making the design exploration far from intuitive.

Our goal is to model a D-Form shape computationally with more intuitive control. To achieve this goal, we

*Corresponding author
Email addresses: fh463@bath.ac.uk (Fei Huang), cgjiang@xjtu.edu.cn (Caigui Jiang), tony@wills-watson.co.uk (Tony Wills), y.yang@cs.bath.ac.uk (Yong-Liang Yang)

first devise a purely geometric approach that can model D-Form in a forward way. The key idea is to deform two planar domains isometrically while satisfying boundary alignment constraints. This is formulated as an optimisation problem that can be efficiently solved by solving a non-linear least-square objective with Cholesky factorisation. Moreover, we present an inverse modelling technique that allows intuitive D-Form control over its seam (i.e., the 3D space curve conjoined by the two planar domain boundaries in 3D) without knowing the 2D planar domains beforehand. More specifically, given a user-prescribed 3D space curve as the target seam, we first initialise the piecewise developable surfaces based on the convex hull of the input curve. Then we flatten both developable pieces to generate the two initial planar domains. Finally, we employ another isometric deformation optimisation (varied from the forward modelling process) to refine the planar domains and generate the resultant D-Form shape that meets the seam requirement. Our forward and inverse modelling techniques can be used to handle general D-Form shapes with complicated settings in both 2D and 3D, such as the planar domain with non-convex boundaries, non-convex D-Form bodies, and complex seam configurations, as demonstrated in the experiments.

Overall our work makes two major contributions:

- We present an optimisation-based framework that can efficiently model D-Form and its variants.
- We propose a novel inverse modelling approach that can intuitively generate D-Form shapes from space curves.

The rest of the paper is organised as follows. The related works are discussed in Section 2. Important definitions and problem statements regarding D-Form are presented in Section 3. The forward D-Form modelling and inverse D-Form modelling processes are detailed in Section 4 and Section 5, respectively. The presented framework will be extensively evaluated in Section 6. Finally, we conclude and discuss our work in Section 7.

2. Related Work

2.1. D-Forms

D-Forms were proposed by [7] and drew attention to many design and research works [8, 9, 5, 10, 11]. Demaine and Price [12] solved two theoretical problems regarding D-Forms. They indicated that the D-Form was always free of creases when being the convex hull of its seam under some general assumptions. They suggested constructing a D-Form from two flat sheets from the mathematical perspective using Alexandrov's theorem. Other works attempted to generate D-Forms computationally. Orduno et al. [13] presented a simple yet effective method for generating D-Form shapes to guide the fabrication. Physical simulation and tension-based energy minimisation are other options for D-Form generation [3, 1]. Leduc et al. [1] proposed to generate specific D-Forms with a procedural computation approach. Besides, A D-Form also appeared as a representative example in some works related to developable surface modelling, such as optimising D-Form shapes in [6] and remeshing D-Form shapes in [14].

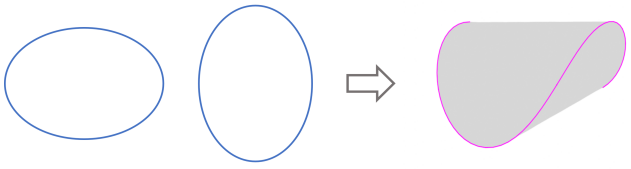
2.2. Developable Surface Modelling

The developable surface is a special type of smooth surface with zero Gaussian curvature everywhere; thus it can be isometrically mapped onto a 2D plane [15, 10]. Developable surfaces are widely used in geometric design [5]. They can be easily fabricated by simply bending real-world materials, such as metal, paper, and cold-bent glass. In the geometry processing field [16], researchers are more interested in discrete developable surfaces with piece-wise linear mesh representations.

Various methods have been proposed for directly modelling discrete developable surfaces. Solomon et al. [17] utilised the fact that the rulings of a developable surface are along with the directions of the non-zero principal curvatures. Based on this, a planar sheet with a set of given rulings was used to model discrete developable surfaces by bending across the rulings. Liu et al. [18] studied planar quadrilateral (PQ) meshes. They presented an alternating subdivision and PQ optimisation approach to model discrete developable surfaces as a PQ strip. Tang et al. [19] introduced a method to model developable surfaces interactively. The modelling may start from different forms, such as planar sheets or standard cylinders. Rabinovich et al. [20, 21] proposed to parameterise developable surfaces through orthogonal geodesics. They used geodesic net to represent and reshape developable surfaces. Some other works used free-form quad meshes to represent and model discrete developable surfaces by isometric deformation [22, 6, 23].

Approximating a given shape with piece-wise developable surfaces is another important research area. Wang and Tang [24] deformed a given shape and decomposed it into a group of developable surfaces in an iterative manner. Julius et al. [25] decomposed the input mesh into a set of quasi-developable charts. Kilian et al. [26] reconstructed developable surfaces by explicitly discretising from their rulings from a given shape.

Tang et al. [19] presented an interactive developable surface modelling tool to approximate a given shape. Stein et al. [27] defined a new developability term 'hinge'. With the new definition, they transferred a triangular mesh to assemble developable surfaces. Ion et al. [28] wrapped developable sheets according to the target mesh and leveraged a global optimisation to refine the result. Verhoeven et al. [14] estimated the rulings of a given triangular developable surface and produced a simplified representation with PQ strips.



(a) (b)

Figure 2: A typical D-Form generated from two ellipses. (a) Two flat components. (b) D-Form. The seam is highlighted in pink color.

3. Definitions and Problem Statement

In this section, we first introduce the key concepts that help the formulation, then state the fundamental problems to be handled in the present work.

3.1. Definitions

Definition 1 (Flat components). *A flat component is a disk-like planar domain. Two flat components with the same perimeter length are used to define a D-Form shape as follows. Note that the name 'flat component' follows the discussion in [12] (see Fig. 2a).*

Definition 2 (D-Form). *A D-Form $\mathbb{D}$ is obtained by bending two flat components D_1 and D_2 while aligning their domain boundaries. Two constituent developable surfaces $\hat{D}_1$ and $\hat{D}_2$ of the D-Form $\mathbb{D}$ are isometric to D_1 and D_2, respectively.*

Definition 3 (Starting pair). *A starting pair is a pair of vertices, each from the boundaries of the one flat components. Matching the two vertices in the pair (thus the rest of the two boundaries sequentially) can uniquely determine the alignment of the two boundaries for D-Form generation.*

Definition 4 (Seam). *The space curve shared by developable surfaces $\hat{D}_1$ and $\hat{D}_2$ in D-Form $\mathbb{D}$ is called the seam $\mathbb{S}$. The length of the seam is equal to the perimeter of the two flat components D_1 and D_2 (see the pink curve in Fig. 2b).*

3.2. Problems to Solve

We divide D-Form modelling into two sub-problems. The first is the *forward* modelling problem: given two flat components with the same perimeter length, how to bend both of them by aligning their boundaries to form a 3D D-Form shape. The second is the *inverse* modelling problem: given a 3D space curve, how to generate a 3D D-Form shape whose seam conforms to the space curve. The forward modelling follows the traditional D-Form modelling pipeline. In contrast, the inverse modelling allows intuitive control of D-Form modelling by specifying the seam as its prominent feature. Both of them use optimisation strategy to generate the result. In the following part of this work, we will elaborate on how to solve the forward problem (Section 4), followed by the detailed solution of the inverse problem (Section 5).

4. Forward Modelling

We formulate the forward modelling as an optimisation problem. The basic idea is to evolve the two flat components in 3D by aligning their boundaries while preserving their intrinsic distance metrics. In other words, the two flat components are deformed isometrically to meet the physical requirement of forming 3D D-Form shapes in practice, i.e., no stretching or tearing of the flat material, such as processing paper or metal sheets.

4.1. Initialisation

As shown in Figure 3, the input of the forward modelling is two flat components D_1 and D_2. Their boundaries are discretised as polylines B_1 and B_2 with the same number of vertices. These vertices are distributed along the boundary uniformly according to the arc length parameterisation. The average length of the boundary line segment is normalised to a unit value. The inner region of the flat component is further discretised using constrained Delaunay triangulation by fixing the boundary vertices and allocating an adequate number of internal vertices, resulting in two triangular meshes M_1 and M_2. To align B_1 and B_2, we also need to specify the boundary vertex correspondences by indicating the starting pair, then the rest vertices can be matched one by one along the two boundaries from the starting pair.

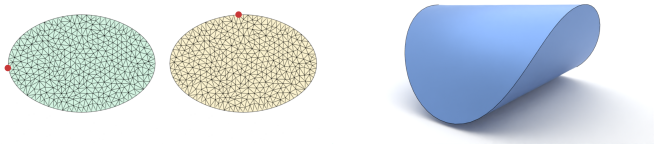
4.2. Optimisation

4.2.1. Variables

Let $\hat{M}_1(\hat{V}_1, E_1, F_1)$ and $\hat{M}_2(\hat{V}_2, E_2, F_2)$ be the two constituent 3D meshes of the 3D D-Form shape. They are isometrically deformed from the two 2D meshes $M_1(V_1, E_1, F_1)$ and $M_2(V_2, E_2, F_2)$, thus have topologically equivalent edge sets E_1 and E_2, and face sets F_1 and F_2, respectively. The variables of the optimisation include the 3D vertices $\hat{\mathbf{v}}_1^i \in \hat{V}_1$ and $\hat{\mathbf{v}}_2^i \in \hat{V}_2$. Note that the 2D vertices $\mathbf{v}_1^i \in V_1$ and $\mathbf{v}_2^i \in V_2$ are also treated as variables to add more degrees of freedom for vertex distribution in 2D, which allows more flexibility for isometric deformation during D-Form construction.

4.2.2. Energy Terms

Here we define energy terms that take into account various properties of D-Form constructed from flat components.



(a) (b)

Figure 3: The input and output of the forward modelling. (a) The two input flat components are discretised by Delaunay triangulation. Red dots represent the starting pair of vertices. (b) The output D-Form.

Isometry term. The isometry term is defined to ensure isometric deformation from M_1 to $\hat{M}_1$ and from M_2 to $\hat{M}_2$. This term is realised by preserving the length of edges in each mesh during the deformation as:

$$f_{iso} = \sum_{e_1^{ij} \in E_1} (\|\hat{\mathbf{v}}_1^i - \hat{\mathbf{v}}_1^j\|^2 - \|\mathbf{v}_1^i - \mathbf{v}_1^j\|^2)^2 + \sum_{e_2^{ij} \in E_2} (\|\hat{\mathbf{v}}_2^i - \hat{\mathbf{v}}_2^j\|^2 - \|\mathbf{v}_2^i - \mathbf{v}_2^j\|^2)^2. \quad (1)$$

Boundary alignment term. This term is defined to align the boundaries of two flat components while bending them in 3D, as required by modelling D-Forms. As mentioned, we discretise the two flat component boundaries to B_1 and B_2, which have the same number of vertices. Therefore, the alignment of the boundaries can be measured by the closeness between the corresponding boundary vertices as follows:

$$f_{align} = \sum_{\hat{\mathbf{v}}_1^i \in \hat{B}_1} \|\hat{\mathbf{v}}_1^i - \hat{\mathbf{v}}_2^i\|^2, \quad (2)$$

where $\hat{B}_1$ represents the boundary of $\hat{M}_1$ formed by boundary vertices $\hat{\mathbf{v}}_1^i \in \hat{B}_1$, whose corresponding boundary vertices are $\hat{\mathbf{v}}_2^i \in \hat{B}_2$. Note that the boundary alignment is measured in 3D (i.e., on $\hat{M}_1$ and $\hat{M}_2$), while the boundaries of M_1 and M_2 remain fixed in 2D (see below).

Boundary fix term. We require the boundaries of M_1 and M_2 (marked as B_1 and B_2) to be fixed due to two reasons. First, the boundaries of two flat components are key inputs of the forward modelling; thus, they are required to keep fixed during optimisation. Second, M_1 and M_2 may degenerate into one point, leading to a naive global minimum if no boundary condition added. For each vertex $\mathbf{v}_1^i \in B_1$, we store its original position $\mathbf{b}_1^i$. In optimisation, we minimise their distance in-between to preserve the boundary shape of B_1. The same also applies to B_2. The overall term is defined as:

$$f_{fix} = \sum_{\mathbf{v}_1^i \in B_1} \|\mathbf{v}_1^i - \mathbf{b}_1^i\|^2 + \sum_{\mathbf{v}_2^i \in B_2} \|\mathbf{v}_2^i - \mathbf{b}_2^i\|^2. \quad (3)$$

Surface smoothness term. This term ensures the mesh quality by preventing flipped and degenerated triangles. The smoothness is required for all meshes to be optimised, including M_1, M_2 and $\hat{M}_1$, $\hat{M}_2$. Its definition is based on the discrete Laplacian operator [29] per mesh vertex:

$$f_{smooth} = \omega_{2D} \sum_{\mathbf{v}_*^i \in V_*} \| \sum_{\mathbf{v}_*^j \in N_*^i} w_*^{ij} (\mathbf{v}_*^j - \mathbf{v}_*^i) \|^2 + \omega_{3D} \sum_{\hat{\mathbf{v}}_*^i \in \hat{V}_*} \| \sum_{\hat{\mathbf{v}}_*^j \in \hat{N}_*^i} \hat{w}_*^{ij} (\hat{\mathbf{v}}_*^j - \hat{\mathbf{v}}_*^i) \|^2. \quad (4)$$

Here we use N_*^i to represent the one-ring neighbor of vertex $\mathbf{v}_*^i$ of 2D mesh $M_*(* = 1, 2)$. w_*^{ij} is the cotangent weight between current vertex $\mathbf{v}_*^i$ and its neighboring vertex $\mathbf{v}_*^j$. The same also applies for the smoothness of 3D mesh $\hat{M}_*(* = 1, 2)$. ω_{2D} and ω_{3D} are two weights to balance between M_* and $\hat{M}_*$.

Algorithm 1: Forward modelling optimisation

Data: M_1, M_2, initial value for $\hat{M}_1$ and $\hat{M}_2$
Fix threshold f_{min}, maximum iterations i_{max}
Initialise $i = 0$, ω_{iso}, ω_{align}, ω_{fix}, ω_{seam}
while $f \geq f_{min}$ *and* $i < i_{max}$ **do**
  $f \leftarrow f_{smooth} + \omega_{iso} f_{iso} + \omega_{align} f_{align} + \omega_{fix} f_{fix} + \omega_{seam} f_{seam}$
  $i \leftarrow i + 1$
  Update V_1, V_2, $\hat{V}_1$, $\hat{V}_2$
end
Result: V_1, V_2, $\hat{V}_1$, $\hat{V}_2$

Seam smoothness term. To penalise the irregular shape of the D-Form seam (see Definition 4) by aligning the boundaries of $\hat{M}_1$ and $\hat{M}_2$, we also add a seam smoothness term by minimising the second-order differences of consecutive boundary vertices as:

$$f_{seam} = \sum_{\hat{\mathbf{v}}_1^i \in \hat{B}_1} \|\hat{\mathbf{v}}_1^{i-1} + \hat{\mathbf{v}}_1^{i+1} - 2\hat{\mathbf{v}}_1^i\|^2 + \sum_{\hat{\mathbf{v}}_2^i \in \hat{B}_2} \|\hat{\mathbf{v}}_2^{i-1} + \hat{\mathbf{v}}_2^{i+1} - 2\hat{\mathbf{v}}_2^i\|^2 \quad (5)$$

4.2.3. Objective

The forward D-Form modelling formulation is an optimisation of the overall objective, which is a weighted sum of all the terms defined above: $\omega_{iso} f_{iso} + \omega_{align} f_{align} + \omega_{fix} f_{fix} + f_{smooth} + \omega_{seam} f_{seam}$. Our formulation is essentially geometric by following the definition and geometric properties of D-Form modelling. The isometric term and boundary alignment term coincide with the modelling process, while the other terms avoid trivial solutions and ensure the quality of the optimisation result. In practice, we utilise the Cholesky decomposition to solve the optimal solution of the non-linear least-squares problem. Alg. 1 breaks down the optimisation procedure, and the detailed evaluation can be found in Section 6.

5. Inverse Modelling

Unlike the forward modelling, where the two flat components are given, the inverse modelling (see Fig. 4) aims to automatically determine two flat components guided by the given design specification, i.e., a space curve that represents the target seam of the D-Form. It is also formulated as an optimisation problem. As in the forward modelling, we require the boundaries of the two flat components to align with each other after deformation in 3D.

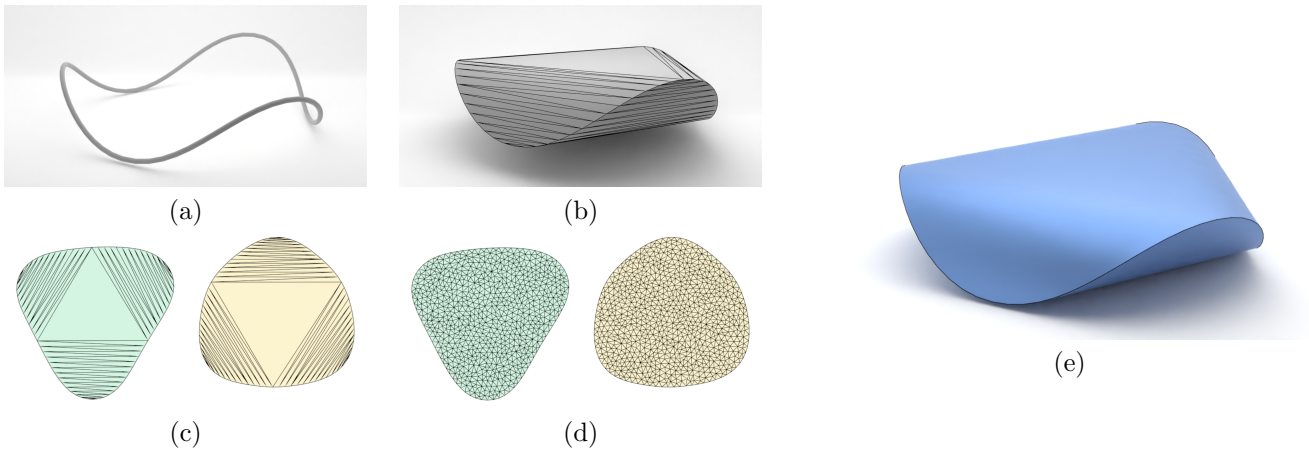


Figure 4: The inverse modelling pipeline. (a) The input is a smooth space curve. (b) The convex hull of the space curve. (c) The convex hull is divided into two pieces by the space curve. Each piece is unfolded onto the 2D plane, resulting the initial flat components. (d) The flat components are discritised by Delaunay triangulation. (e) The output d-from after optimisation constrained on the input space curve.

Additionally, the seam (by aligning the two boundaries) also needs to fit the given space curve. The critical part is how to initialise the two flat components and update them in the optimisation process.

5.1. Initialisation

The input of our inverse modelling is a space curve in 3D (see Fig. 4a), which is implemented as a Bezier curve but can be any smooth curve in general. The space curve represents the target seam (can also be treated as the sharp feature) of the output D-Form. The curve is parameterised by arc length and discretised into an ordered list of vertices C. As in the forward modelling, the average edge length between consecutive vertices in C is normalised to a unit value. The basic idea for our initialisation is to form a preceding geometric form (we call it pre-D-Form) based on the convex hull of C (see Fig. 4b), then use the two parts on the pre-D-Form divided by C to initialise the two flat components of the D-Form (see Fig. 4c). According to its relationship with the convex hull, we classify C into two categories: *on-hull* space curve and *non-on-hull* space curve, which are detailed below.

On-hull space curve. D-Form has been studied from the perspective of the convex hull of its seam. [12] proved that as long as the metric space of the D-Form is locally convex, the D-Form is always the convex hull of its seam. [30] and [31] also show that the triangulation approximates a developable surface when most edges are locally convex. Inspired by the above, we first construct the convex hull H of C. If C fully resides on H, we call it an on-hull space curve, and the pre-D-Form is exactly the convex hull of C in this case. Then it is easy to divide the pre-D-Form into two parts by C, resulting in two surfaces $\mathring{D}_1$ and $\mathring{D}_2$ in 3D. $\mathring{D}_1$ and $\mathring{D}_2$ can be treated as the initialisation of $\hat{D}_1$ and $\hat{D}_2$, the two constituent developable surfaces of the D-Form.

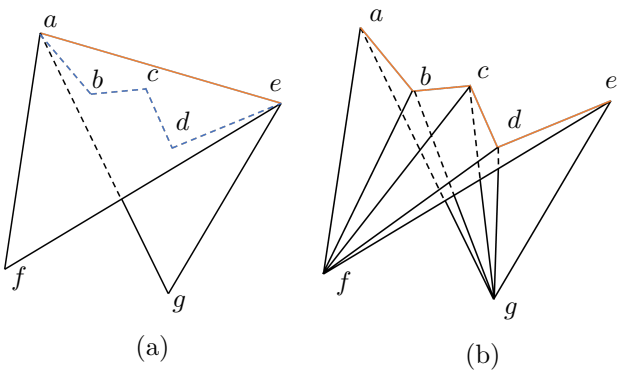


Figure 5: Local modification of the convex hull to construct the pre-D-Form. The edge e_{ae} on the convex hull but not on the input space curve is replaced with four new edges along the space curve. The faces f_{afe} and f_{aeg} are removed and replaced by eight new faces.

Non-on-hull space curve. Unlike the on-hull space curve case, C may not entirely reside on its convex hull H. In other words, only part of the vertices in C lies on H, while other vertices are inside H. Here we call C a non-on-hull space curve. In this case, we construct the pre-D-Form by locally modifying the convex hull H, such that it can still be divided into two parts by C. The basic idea is to extract space curve segments on H first and keep their associated convex hull parts only as a partial pre-D-Form (this is similar to what we did for the on-hull space curve, where it only has one space curve segment on H which is associated with the whole convex hull). Then we fill in the missing parts of the pre-D-Form by adding triangle fans based on those vertices in C and inside the convex hull H. One example of triangle fan filling is shown in Fig 5. Here edge e_{ae} lies on the convex hull. However, it links two non-consecutive vertices $\mathbf{v}_a$ and $\mathbf{v}_e$ in C. Hence we delete their associated triangles (f_{afe} and f_{aeg}) in H. Vertices $\mathbf{v}_b$, $\mathbf{v}_c$ and $\mathbf{v}_d$ form one curve segment of C which lies inside H. In this case, we add two triangle fans. One is formed by

f_{afb}, f_{bfc}, f_{cfd}, and f_{dfe} to replace the missing f_{afe}. The other is formed by f_{abg}, f_{bcg}, f_{cdg} and f_{deg} to replace the missing f_{aeg}. In total eight new faces are added to the pre-D-Form. Note that after the triangle fan filling process, all the vertices in C now reside on the pre-D-Form.

After the pre-D-Form is achieved (by handling either on-hull or non-on-hull space curve), we can easily divide it into two surfaces $\mathring{D}_1$ and $\mathring{D}_2$ in 3D by cutting along C. Both $\mathring{D}_1$ and $\mathring{D}_2$ are formed by triangle strips/fans constructed by all vertices in C. Thus they can be flattened onto the 2D plane without any distortion, leading to two planar domains, which can be treated as the initialisation of the two flat components of the D-Form. We keep the boundaries of two planar domains and remesh them by applying Delaunay triangulation to generate M_1 and M_2 as in the forward modelling (see Fig. 4d). Then we re-project the vertices of M_1 and M_2 back onto $\mathring{D}_1$ and $\mathring{D}_2$ based on the barycentric coordinates of each vertex in their flattened triangles, resulting in the initialised $\hat{M}_1$ and $\hat{M}_2$. C, M_1 and M_2 are all forwarded into the following optimisation to generate the final D-Form result (see Fig. 4e).

5.2. Optimisation

5.2.1. Variables

After initialisation, the variables to be optimised in the inverse modelling are the same as those in the forward modelling described in Section 4.2.1.

5.2.2. Energy Terms

Here we define energy terms used in the following-up optimisation after initialisation.

Seam closeness term. To meet the requirement of the target seam, we define a term that minimises the closeness between the seam and the two boundaries $\hat{B}_1$ and $\hat{B}_2$ of 3D meshes $\hat{M}_1$ and $\hat{M}_2$ as follows:

$$f_{close} = \sum_{\hat{\mathbf{v}}_1^i \in \hat{B}_1} \|\hat{\mathbf{v}}_1^i - \mathbf{c}^i\|^2 + \sum_{\hat{\mathbf{v}}_2^i \in \hat{B}_2} \|\hat{\mathbf{v}}_2^i - \mathbf{c}^i\|^2, \quad (6)$$

where $\mathbf{c}^i$ is one of the vertex in C and $\mathbf{v}_1^i$ and $\mathbf{v}_2^i$ are its corresponding boundary vertices on $\hat{B}_1$ and $\hat{B}_2$, respectively.

Isometry term. This term is adopted from the forward modelling process (Eqn. 1) to ensure isometric deformation when generating D-Form.

Surface smoothness term. This term is also defined as in the forward modelling (Eqn. 4) to ensure mesh quality for D-Form generation.

5.2.3. Objective

The objective we minimise in the inverse modelling is $f_{smooth}+\omega_{close}f_{close}+\omega_{iso}f_{iso}$. Note that the inverse modelling is different from the forward modelling in the sense that the target seam of the D-Form is prescribed. As a result, the boundary alignment term and seam smoothness term do not need to apply here. Although the seam is fixed, the two flat components have more degrees of freedom because their boundaries are not fixed during the optimisation. Therefore, it better allows the isometric deformation between M_* and $\hat{M}_*$ while meeting the seam requirement. The overall inverse modelling optimisation procedure is shown in Alg. 2.

Algorithm 2: Inverse modelling optimisation

Data: Input curve C, initial value for M_1, M_2, $\hat{M}_1$, $\hat{M}_2$
Fix threshold f_{min}, maximum iterations i_{max}
Initialise $i = 0$, ω_{iso}, ω_{close}
while $f \geq f_{min}$ *and* $i < i_{max}$ **do**
  $f \leftarrow f_{smooth} + \omega_{iso} f_{iso} + \omega_{close} f_{close}$
  $i \leftarrow i + 1$
  Update V_1, V_2, $\hat{V}_1$, $\hat{V}_2$
end
Result: V_1, V_2, $\hat{V}_1$, $\hat{V}_2$

6. Evaluation

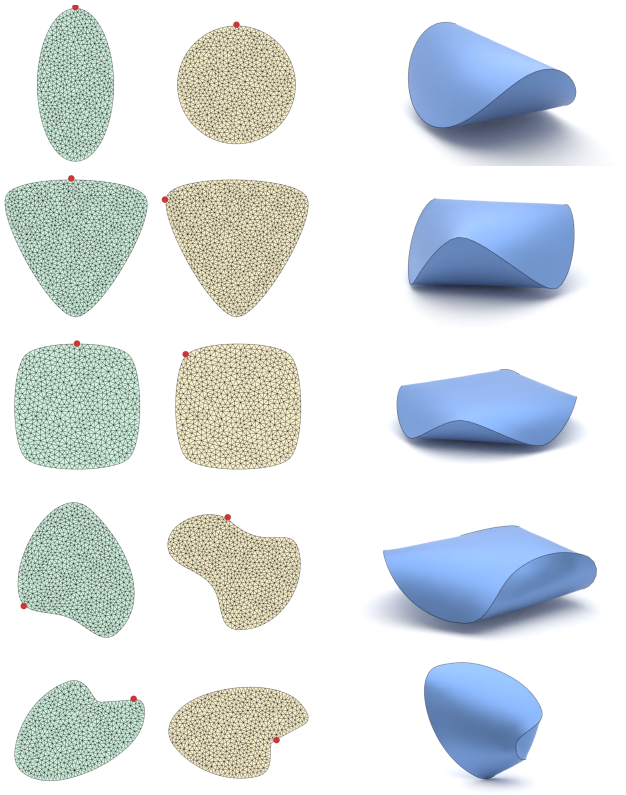
In this section, we extensively evaluate our work by 1) testing forward modelling and inverse modelling from various inputs and specifications; 2) measuring the quality of the results using different metrics and/or under different scenarios; and 3) demonstrating other design possibilities. In addition, we list the related statistics, including evaluation metrics and weights for optimising all examples presented in this paper in Table 1.

6.1. Forward Modelling Results

We show a variety of forward modelling results in Fig. 6. Our work is capable of modelling D-Forms composed of two developable surfaces deformed from two flat components isometrically.

Different starting pairs for boundary alignment. In Fig. 7, we show examples of the forward modelling using the same flat components with different starting pairs (shown as red dots). We can see that different starting pairs lead to different D-Form shapes. The same two flat components can generate various D-Forms with different boundary alignment configurations determined by the start pairs.

Flat components with non-convex boundaries. Prior works usually demonstrate D-Forms based on flat components with convex boundaries. We also test our forward modelling algorithm with non-convex boundaries, as shown by the last two cases in Fig. 6. We can see that non-convex flat components can also produce reasonable D-Forms. [12] proved that the D-Form generated from two convex flat components was free of creases, which means the D-Form has no sharp edge except its seam. However, this does not



(a) (b)

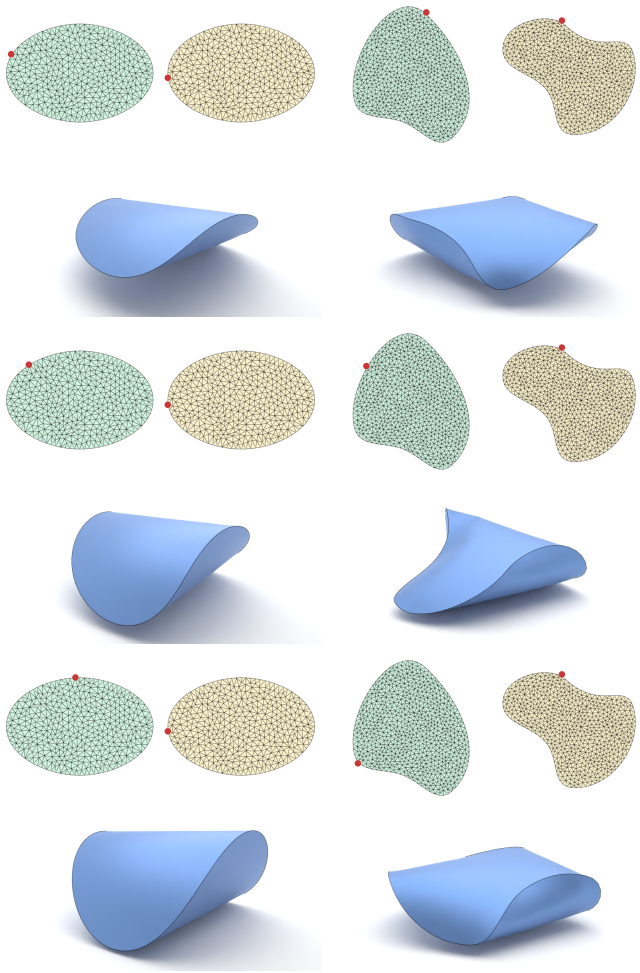
Figure 6: Forward modelling results. (a) Flat components triangulated from input boundaries. Red dots represent the starting pair for each example. (b) D-Forms generated from the input.

hold for D-Forms generated from non-convex flat components, where additional creases may be introduced. Designers may want to take this problem into account when designing D-Forms.

6.2. Inverse Modelling Results

Apart from the forward modelling, our work can further be employed to generate plausible inverse D-Form modelling. Fig. 8 shows various inverse modelling results by not specifying the two flat components but using a space curve to denote the target seam, where the flat components and their corresponding D-Form shape are automatically generated. The flexibility of specifying the target seam allows more intuitive control of the resultant D-Form shape without trying out different flat components.

Non-on-hull seams. The second, fourth, and fifth cases in Fig. 8 are generated from non-on-hull space curves (seams). From [12], we know that D-Forms are often the convex hull of their seams (i.e., on-hull seams), such as the first and third cases in Fig. 8. However, more complicated seams are under-explored. We show that plausible results can be generated in such cases.

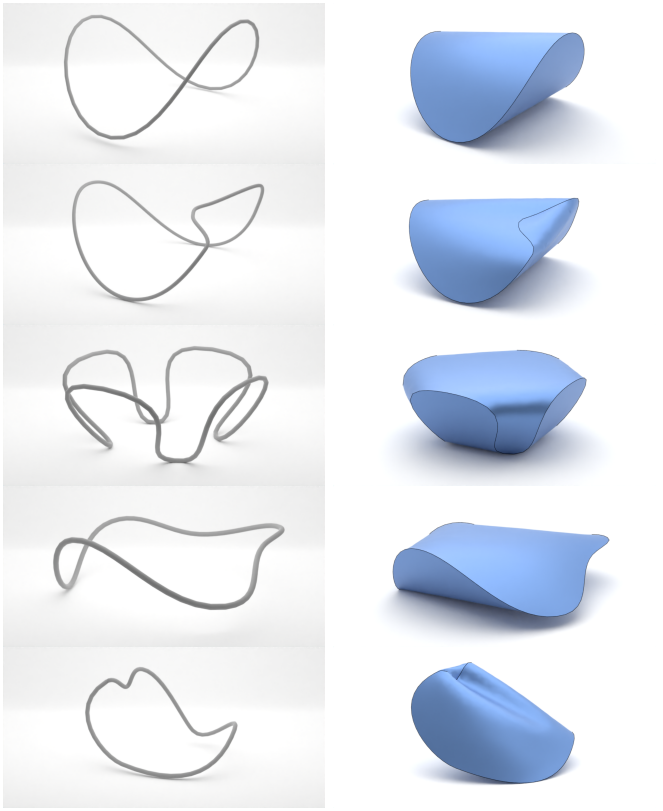


(a) (b)

Figure 7: With the same flat components, different starting pairs result in different D-Forms. The top row of each sub-figure shows the specified starting pairs (denoted by red dots). The bottom row shows corresponding results.

6.3. Quality Validation

Developability. We evaluate the developability of the resultant D-Forms in two ways. Firstly, we visualise the Gaussian curvature shown in Fig. 9. We can see that the resultant D-Form has zero Gaussian curvature almost everywhere except the seam, meaning the two constituent surfaces bent from flat components are developable. Secondly, we evaluate the developability error based on edge length variations as in [6]. The L_{error} is defined as $||\hat{L} - L||/||L||$. $\hat{L}$ refers to the vector composed by all the edge lengths of the 3D meshes $\hat{M}_1$ and $\hat{M}_2$. L refers to that of 2D meshes M_1 and M_2. The developability error for each example is shown in Table 1. All the examples have error value less than 1%, most of which are less than 0.5%. We also show the maximum developability error per edge L_{max} in Table 1. It is defined as the maximum value of $(\hat{L}_i - L_i)/L_i$, where i denotes the component index within a vector. All the examples have $L_{max} < 3\%$ except one special demi D-Form in Fig. 12 which is 7.18%.



(a) (b)

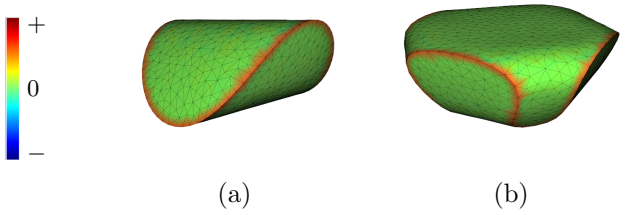
Figure 8: Inverse modelling results. (a) Input space curve. (b) D-Form generated from the left input.

Curve matching. In the inverse modelling we minimise the distance between two boundaries and the input space curve. We calculate the (normalised) average error D_{error} between vertices of the input space curve and their corresponding vertices on the two boundaries as:

$$\hat{D} = \hat{D}_1 + \hat{D}_2 = \{\hat{d}_1, \hat{d}_2, ..., \hat{d}_n\}^T, \quad D_{error} = \frac{\sum \hat{d}_i / n}{L_{curve}}, \tag{7}$$

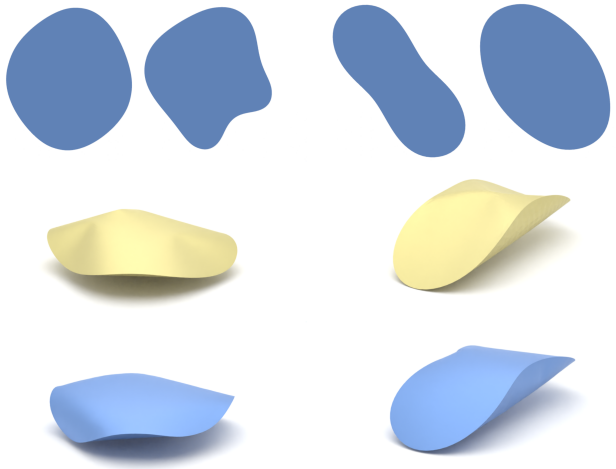
where $\hat{D}_1$ is the vector composed of distances between each vertex on the boundary of $\hat{M}_1$ and its corresponding vertex on the input curve. $\hat{D}_2$ represents the vector composed of distances between the boundary of $\hat{M}_2$ and the input curve. L_{curve} refers to the length of the input curve. Values of D_{error} are shown in Table 1. All values are smaller than 0.0001%. Our algorithm can produce the D-Form whose seam matches the input curve with a reasonable-small error.

Comparison with previous work. We compare our work with [6], a recent method for quad-mesh-based developable surface modelling, which can also be adopted for (only) the forward D-Form modelling. The qualitative results are shown in Fig. 10. The quantitative comparisons on computational time, developability, boundary matching, etc. can be found in Table 2. From both qualitative and



(a) (b)

Figure 9: Developability validation by visualising the Gaussian curvature of two resultant D-Forms.



(a) (b)

Figure 10: We compare our forward modelling result (bottom row) with the output from [6] (middle row). Top row shows the two groups of input flat components.

quantitative results, we can see that both methods generate smooth D-Form shapes that are visually pleasing and similar. The computational costs are comparable given both methods are based on non-linear least-squares optimisation. While the errors are reasonably small (<1.5e-3) for both methods, the D-Form shapes generated from our method have slightly better developability and better-matched boundaries. We attribute this to the better geometric representation ability of triangle mesh over quad mesh.

Fabrication. We also validate our results by actual fabrication using paper to create physical objects. Fig. 11 shows the virtual D-Forms, and their corresponding paper objects, where D-Forms in (a), (b) and (d) are generated from the forward modelling while D-Form in (c) is generated from the inverse modelling. We can see that the virtual D-Forms and their physical counterparts are visually close, which verifies the quality of our results, in particular the developability. This can largely benefit artists for creating design prototypes in practice.

6.4. Other Design Possibilities

Anti D-Forms and demi D-Forms. The D-Form we discussed so far is formed by matching the outer boundaries

Fig.	$\|V\|$	$\|F\|$	I	t(s)	L-Error(%)	L-Max(%)	D-Error(%)	w_{iso}	w_{align}	w_{fix}	w_{close}	$w_{smooth3D}$	$w_{smooth2D}$	w_{seam}
3	0.8k	1.5k	10	1.06	0.03	0.33	-	1	1	1	-	0.01	1	0.01
6(1)	1.2k	2.4k	7	1.79	0.21	1.05	-	1	1	1	-	0.01	1	0.01
6(2)	1.3k	2.4k	10	2.36	0.17	0.99	-	1	1	1	-	0.015	1	0.01
6(3)	0.8k	1.5k	12	1.38	0.06	0.62	-	2	1	1	-	0.01	1	0.01
6(4)	0.9k	1.6k	15	1.86	0.43	2.66	-	1.5	1	1	-	0.015	1	0.01
6(5)	1.1k	2k	10	1.82	0.19	2.92	-	1	1	1	-	0.01	1	0.01
4	1.7k	3.3k	3	0.82	0.006	0.07	7.2e−7	1	-	-	1	0.001	1	-
8(1)	1.7k	3.3k	3	0.82	0.002	0.02	2.9e−7	1	-	-	1	0.001	1	-
8(2)	1.5k	3k	3	0.65	0.06	0.61	1.3e−5	1	-	-	1	0.005	1	-
8(3)	1.2k	2.2k	3	0.28	0.0003	0.09	8.9e−7	1	-	-	1	0.001	1	-
8(4)	1.7k	3.3k	7	1.81	0.11	0.71	4.5e−5	1	-	-	1	0.02	1	-
8(5)	1.6k	3k	7	1.52	0.14	1.34	8.6e−5	1	-	-	1	0.01	1	-
7(a)(1)	0.8k	1.5k	10	1.06	0.02	0.29	-	1	1	1	-	0.01	1	0.01
7(a)(2)	0.8k	1.5k	10	1.03	0.03	0.32	-	1	1	1	-	0.01	1	0.01
7(b)(1)	0.9k	1.6k	15	1.86	0.43	2.24	-	1.5	1	1	-	0.01	1	0.01
7(b)(2)	0.9k	1.6k	15	1.92	0.44	2.37	-	1.5	1	1	-	0.01	1	0.01
10(a)	1.1k	2k	15	3.62	0.12	2.16	-	2	1	1	-	0.014	1	0.01
10(b)	1.2k	2.2k	15	2.83	0.046	0.55	-	2.2	1	1	-	0.01	1	0.01
11(d)	0.9k	1.6k	15	1.78	0.46	2.16	-	1.5	1	1	-	0.01	1	0.01
12(a)	2.7k	5.4k	20	15.79	0.31	2.13	-	1	1	1	-	0.08	1	0.1
12(b)	2.1k	4k	20	12.08	0.25	7.18	-	1	1	1	-	0.08	1	0.1
14(b)	0.8k	1.4k	7	1.10	0.18	1.13	-	1	1	1	-	0.01	1	0.01

Table 1: The statistics of the results in this paper. We show the figure index, number of vertices $|V|$, number of faces $|F|$, number of optimisation iterations I, time for optimisation, developability error L_{error}, maximum developability error per edge L_{max}, seam matching error D_{error}, and optimisation weights.)

Fig.	$\|V\|$	I	$t_{\mathbf{v}}$	L-Error(%)	B-Error(%)
10(a) (middle)	11k	10	8.18e-4	0.15	2.56e-3
10(a) (bottom)	1.1k	15	3.29e-3	0.12	2.75e-9
10(b) (middle)	2.5k	20	2.08e-3	0.054	2.63e-2
10(b) (bottom)	1.2k	15	2.35e-3	0.046	8.41e-10

Table 2: Quantitative comparison between our method and [6] in the forward modelling setting. We compare number of vertices $|V|$, number of optimisation iterations I, computational time per vertex $t_{\mathbf{v}}$, developability error L_{error}, and boundary matching error B_{error}. The B-Error is defined similarly to the curve/seam matching error D_{error} in the inverse modelling, but using one boundary (not the target seam) as the other boundary's target.

of two disk-like flat components. By matching the inner boundary instead of the outer boundary, two variations of D-Form can be created, including *anti D-Form* formed by matching the inner boundaries of two annulus-like flat components with holes, and *demi D-Form* constructed by matching the outer boundary of one disk-like flat component and the inner boundary of the other annulus-like flat component. Our work can be adapted to generate anti D-Form and demi D-Form shapes, as shown in Fig. 12. In our implementation, we do not take the annulus-like flat component as input directly. Instead, we first convert it to a disk-like component by filling the hole while keeping the marks of inner boundary vertices. Then we still use the marked vertices for boundary alignment. After optimisation, we remove those faces added for hole filling to generate the final output.

Intuitive design exploration. Our work can benefit intuitive design exploration. Here we show an example of the easy creation of design variations from an existing design. Based on an existing D-Form [2] as shown in Fig. 13a, we can easily modify its seam and then employ the inverse modelling to create a new D-Form (see Fig. 13b). This new D-Form matches the prescribed seam constraints without tuning the shape of the flat components, making the design process much more efficient. A demonstration of applying the new D-Form for street furniture design is in Fig. 13c.

6.5. Implementation Details

We implement our framework with C++ on a desktop PC with i7-8700K CPU and 32GB RAM. The input meshes for optimisation typically have about 2k triangles, and the input space curves have around 100 vertices. The approximate running time of the algorithm is usually around 2s. The most time-consuming part is the iterative optimisation.

6.6. More on Optimisation

Our current optimisation solution penalises the error of the isometry term (Eqn. 1) and boundary alignment terms (Eqn. 2, 3, 6) by specifying much bigger weights (∼100 times larger than mesh smoothness in Eqn 4 and seam smoothness in Eqn 5), as shown in Table 1. The

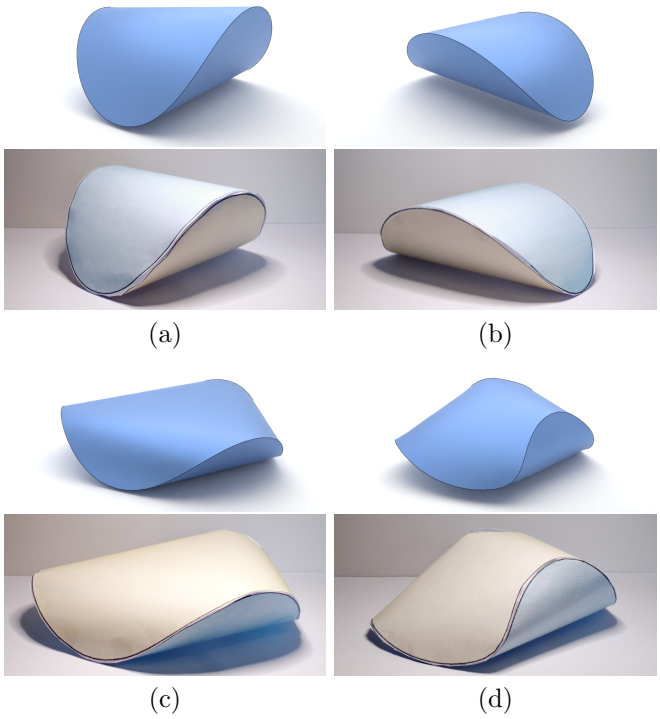


Figure 11: The resultant D-Form shapes can be easily fabricated using paper. Top row in each sub-figure shows the virtual shape. Bottom row shows the paper object. (a), (b), and (d) are generated from forward modelling. (c) is generated from inverse modelling.

Figure 12: Two special D-Forms generated from our work. (a) anti D-Form. (b) demi D-Form.

optimisation process can be treated as a 'geometric simulation' of how D-Form is fabricated in practice (smoothly bending two flat components while aligning boundaries). Different terms in the objective function are not mutually conflict and they can be effectively minimised altogether. As a result, our solution achieves satisfactory results with very small errors (see Tables 1 and 2). Another possible solution is to only keep smoothness-related terms in the objective function while modelling isometry- and alignment-related terms as hard (equality) constraints. Such a problem can be solved by constrained optimisation algorithms such as Augmented Lagrangian Method (ALM). This solution would perform better on hard constraints but the computational cost would also increase.

Also, in our current inverse modelling formulation, we expect the two boundaries to match the target seam, thus using the penalty term of seam closeness in Eqn. 6. Given the pre-D-Form already meets this requirement (see Section 5.1) and the boundaries of the two flat components

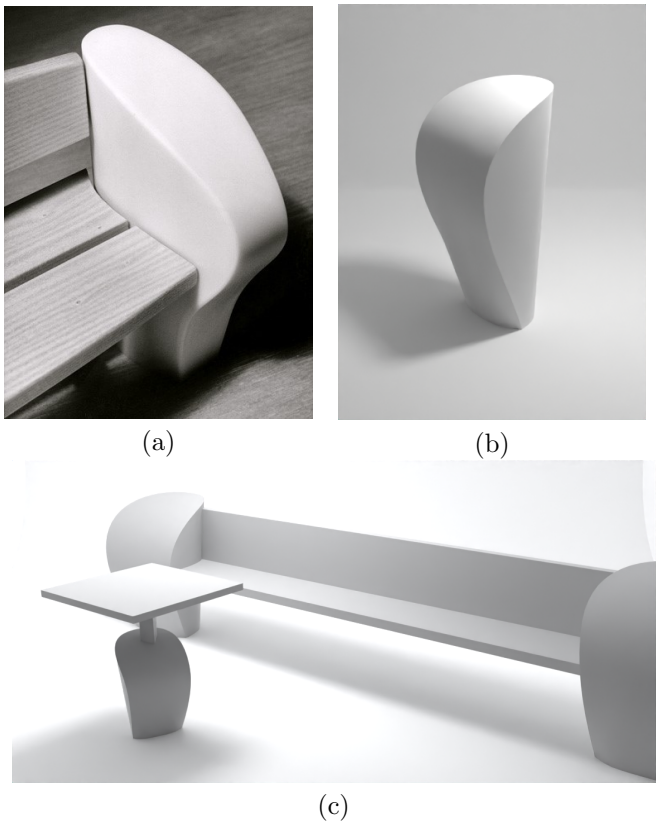


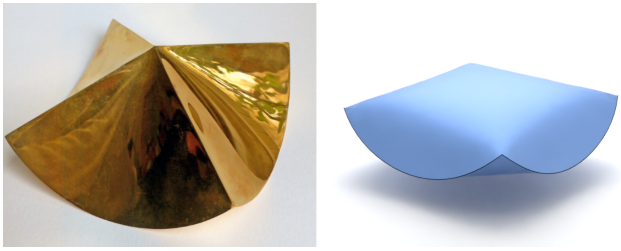
Figure 13: Intuitive design exploration inspired existing designs. (a) Westminster street furniture collection [2]. (b) A D-Form variant inspired by (a). We use a varied space curve as input and generate a design variation based on inverse modelling. (c) A street furniture scene design based on the resultant D-Form.

are not fixed during the optimisation (see Section 5.2.3), such an initialisation along with the additional degrees of freedom produce good target seam matching results as validated by the very small D_{error} in Table 1. On the other hand, for potential cases where the seam closeness term cannot be effectively minimised, relaxing the seam matching energy and forcing no gap at the seam with large penality or hard constraints would be interesting to explore.

7. Conclusion and Discussion

In this paper, we present a computational framework for modelling D-Forms. Unlike prior work, our framework can model D-Forms not only forwardly from two planar domains but also inversely from a space curve, making the modelling and exploration process more intuitive. Various outputs with different design constraints demonstrate the efficacy and efficiency of our work. We hope our work can inspire following-up research works on D-Form and other types of pieces-wise developable surfaces.

Discussion. Our forward D-Form modelling is based on automated smooth isometric deformation of the flat components (recall the isometric term and the smoothness



(a) (b)

Figure 14: D-form shape formed by one disk and one square. (a) The Squaricle D-Form design [2], where four sharp edges were introduced to bend the disk component. (b) The output generated from our algorithm, which produces smooth deformation result from the disk component.

term). As a result, we cannot generate D-Form with sharp features in addition to the seam, such as the Squaricle, the D-Form shape formed by one disk and one square (see Fig. 14). In the future, we plan to allow user sketches to further split 2D flat components and their 3D developables into smaller segments to handle such cases. Also, our inverse D-Form modelling is automatically initialised by constructing the convex hull. Although this approach successfully provides proper initialisation, there can be other initialisation possibilities, which is especially true for complicated space curves, given there can be finitely many developable surfaces having such a boundary [32]. We will further explore how to perform interactive initialisation similar to [31] for sketches. Further, there is no guarantee that any space curve can be the boundary of a developable surface, especially for space curves that are non-smooth and winding around. We only provide a practical method to find one of the solutions if it exists, but what space curve can correctly serve as the seam of a D-Form is still an open question.

Acknowledgement. The authors would like to thank anonymous reviewers for their suggestive comments, and Bailin Deng for helpful discussions on constrained optimisation. This paper is partly supported by RCUK grant CAMERA (EP/M023281/1, EP/T022523/1), the Fundamental Research Funds for the Central Universities under Grant xtr072022001 and a gift from Adobe.

References

[1] N. Leduc, C. Douthe, B. Vaudeville, S. Aubry, K. Leempoels, L. Hauswirth, O. Baverel, Dforms with constant discrete gaussian curvature, in: Advances in Architectural Geometry, Presses des Ponts, 2020.

[2] T. Wills, D-form street furniture, https://www.wills-watson.co.uk/portfolio-item/d-form-street-furniture/.

[3] K. Brakke, The surface evolver, http://facstaff.susqu.edu/brakke/evolver/evolver.html (2013).

[4] P. Bourke, dforms, http://paulbourke.net/geometry/dform/.

[5] H. Pottmann, A. Asperl, M. Hofer, A. Kilian, Architectural geometry, Bentley Institute Press, 2007.

[6] C. Jiang, C. Wang, F. Rist, J. Wallner, H. Pottmann, Quad-mesh based isometric mappings and developable surfaces, ACM Transactions on Graphics 39 (4) (2020).

[7] T. Wills, et al., Dforms: 3d forms from two 2d sheets, in: Bridges London: Mathematical Connections in Art, Music, and Science, 2006, pp. 503–510.

[8] J. Sharp, D-forms: Surprising new 3-D forms from flat curved shapes, Tarquin Publications Hertfordshire, 2009.

[9] E. D. Demaine, J. O'Rourke, Geometric folding algorithms: linkages, origami, polyhedra, Cambridge University Press, 2007.

[10] H. Pottmann, J. Wallner, Computational line geometry, Springer Science & Business Media, 2009.

[11] K. A. Seaton, Sphericons and d-forms: a crocheted connection, Journal of Mathematics and the Arts 11 (4) (2017) 187–202.

[12] E. D. Demaine, G. N. Price, Generalized d-forms have no spurious creases, Discrete & Computational Geometry 43 (1) (2010) 179.

[13] R. R. Orduño, N. Winard, S. Bierwagen, D. Shell, N. Kalantar, A. Borhani, E. Akleman, A mathematical approach to obtain isoperimetric shapes for d-form construction, in: Proceedings of Bridges 2016: Mathematics, Music, Art, Architecture, Education, Culture, Tessellations Publishing, 2016, pp. 277–284.

[14] F. Verhoeven, A. Vaxman, T. Hoffmann, O. Sorkine-Hornung, Dev2pq: Planar quadrilateral strip remeshing of developable surfaces, ACM Transactions on Graphics 41 (3) (2022) 1–18.

[15] M. P. Do Carmo, Differential geometry of curves and surfaces: revised and updated second edition, Courier Dover Publications, 2016.

[16] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Lévy, Polygon mesh processing, CRC Press, 2010.

[17] J. Solomon, E. Vouga, M. Wardetzky, E. Grinspun, Flexible developable surfaces, in: Computer Graphics Forum, Vol. 31, Wiley Online Library, 2012, pp. 1567–1576.

[18] Y. Liu, H. Pottmann, J. Wallner, Y. Yang, W. Wang, Geometric modeling with conical meshes and developable surfaces, ACM Transactions on Graphics 25 (3) (2006) 681–689.

[19] C. Tang, P. Bo, J. Wallner, H. Pottmann, Interactive design of developable surfaces, ACM Transactions on Graphics 35 (2) (2016) 1–12.

[20] M. Rabinovich, T. Hoffmann, O. Sorkine-Hornung, Discrete geodesic nets for modeling developable surfaces, ACM Transactions on Graphics 37 (2) (2018a).

[21] M. Rabinovich, T. Hoffmann, O. Sorkine-Hornung, The shape space of discrete orthogonal geodesic nets, ACM Transactions on Graphics 37 (6) (2018b).

[22] C. Peng, C. Jiang, P. Wonka, H. Pottmann, Checkerboard patterns with black rectangles, ACM Transactions on Graphics 38 (6) (2019) 171:1–171:13.

[23] C. Jiang, H. Wang, V. C. Inza, F. Dellinger, F. Rist, J. Wallner, H. Pottmann, Using isometries for computational design and fabrication, ACM Transactions on Graphics 40 (4) (2021) 42:1–42:12.

[24] C. C. L. Wang, K. Tang, Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches, The Visual Computer 20 (8-9) (2004) 521–539.

[25] D. Julius, V. Kraevoy, A. Sheffer, D-charts: Quasi-developable mesh segmentation, in: Computer Graphics Forum, Citeseer, 2005, pp. 581–590.

[26] M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, H. Pottmann, Curved folding, ACM Transactions on Graphics 27 (3) (2008) 75.

[27] O. Stein, E. Grinspun, K. Crane, Developability of triangle meshes, ACM Transactions on Graphics 37 (4) (2018) 1–14.

[28] A. Ion, M. Rabinovich, P. Herholz, O. Sorkine-Hornung, Shape approximation by developable wrapping, ACM Transactions on Graphics 39 (6) (2020) 200:1–200:12.

[29] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in: Visualization and Mathematics III, 2003, pp. 35–57.

[30] W. H. Frey, Boundary triangulations approximating developable surfaces that interpolate a closed space curve, International Journal of Foundations of Computer Science 13 (2) (2002) 285–302.

[31] K. Rose, A. Sheffer, J. Wither, M.-P. Cani, B. Thibert, Developable surfaces from arbitrary sketched boundaries, in: Eurographics Symposium on Geometry Processing, Eurographics Association, 2007, pp. 163–172.
[32] M. Alberich-Carramiñana, J. Amorós, F. Coltraro, Developable surfaces with prescribed boundary, in: Extended Abstracts GEOMVAP 2019, Springer, 2021, pp. 127–132.