Interactive modeling of lofted shapes from a single image

Congyue Deng[1] (✉), **Jiahui Huang**[1], **and Yong-Liang Yang**[2]

Abstract Modeling the complete geometry of general shapes from a single image is an ill-posed problem. User hints are often incorporated to resolve ambiguities and provide guidance during the modeling process. In this work, we present a novel interactive approach for extracting high-quality freeform shapes from a single image. This is inspired by the popular lofting technique in many CAD systems, and only requires minimal user input. Given an input image, the user only needs to sketch several projected cross sections, provide a "main axis", and specify some geometric relations. Our algorithm then automatically optimizes the common normal to the sections with respect to these constraints, and interpolates between the sections, resulting in a high-quality 3D model that conforms to both the original image and the user input. The entire modeling session is efficient and intuitive. We demonstrate the effectiveness of our approach based on qualitative tests on a variety of images, and quantitative comparisons with the ground truth using synthetic images.

Keywords interactive modeling; freeform shapes; image-guided modeling; lofting

1 Introduction

3D modeling is a key research area in computer graphics. Recent developments have resulted in advanced CAD systems that can help to create intricate 3D models in different design scenarios. However, this conventional design process still requires professional skill to specify numerous parameters for modeling freeform shapes with complicated geometry. Moreover, to achieve the level of detail required for realistic objects, modeling often proceeds via iterative refinements requiring significant effort. The above factors limit the applicability of traditional modeling techniques, and motivate research into more efficient tools which can make models from a single image. This can not only bridge the gap between having a vision of a 3D object and modeling its geometry, but also benefit novices by providing intuitive control over the results.

The cognition of geometric shapes from a single image is an easy task for humans who have profound knowledge of sensing the 3D world. Nonetheless, 3D shape reconstruction from a single image is an ill-posed problem in general, due to the lack of depth cues and possible self-occlusion or other occlusion in a single view. A suitable solution could be established by leveraging prior knowledge of the object shape. Numerous efforts have been devoted to combining the cognitive abilities of active users with the accuracy and efficiency of computational algorithms. However, most of these methods focus on modeling limited types of shapes with regular geometry (e.g., cuboids, swept surfaces, extruded objects, etc.) or a specific shape class (e.g., architecture, garments, trees, etc.).

In this work, we present a novel interactive image-guided modeling approach which greatly extends the capability of existing methods. To model more complicated shapes given a single image, our approach is closely related to the lofting technique used for creating freeform surfaces in CAD systems. Our method is also based on the observation that most man-made or natural objects have irregular 3D shapes but rather regular 2D cross sections. By exploiting the regularities in these sections as well as additional geometric constraints, we can model a large variety of objects from a single image with minimal user input. Such lofted shapes can be readily found in

1 Tsinghua University, Beijing, 100084, China. E-mail: C. Deng, dengcy16@mails.tsinghua.edu.cn (✉); J. Huang, huang-jh18@mails.tsinghua.edu.cn.

2 University of Bath, Bath, BA2 7AY, UK. E-mail: y.yang@cs.bath.ac.uk.

real life, such as natural grown organisms with strong symmetry, distorted regular shapes, and man-made artworks, just to name a few. In fact, there is a specific artistic design style called "parametric design" which considers transformations of cross-sectional shapes. Our work is also greatly inspired by this idea.

The input to our system is an image with calibrated camera parameters. During the modeling process, the user sketches a set of parallel cross sections in the image space together with a "main axis" orthogonal to the sections, and also specifies some geometric constraints. The common normal to the sections is estimated by solving an optimization problem under these constraints. The relative positions of these sections can then be determined (up to a scalar) using simple linear equations. Next, we compute appropriate interpolations between adjacent sections to generate the complete shape. For objects consisting of multiple parts, we allow the user to model each part separately and then combine them together with point contact cues.

With our method, novices can model lofted 3D shapes from a single image with the aid of simple sketches requiring minimal effort. The whole modeling process is efficient and intuitive, and no professional skills are required. We demonstrate the wide applicability of our method with various examples.

2 Related work

2.1 Sketch based modeling

A major concern of sketch based modeling is the design of intuitive 2D interactions for generating 3D geometry, especially compared to tradition CAD systems. A comprehensive review can be found in Ref. [1]. We only discuss the most relevant works by classifying them into two categories, single-view methods and multi-view methods, according to the kind of user input.

Using multi-view tools, users can construct and modify 3D surfaces with sketches from different viewpoints. Starting from the seminal work of Teddy [2], contour curves have been widely used in multi-view modeling [3, 4]. Auxiliary lines can also be useful. For example, Ref. [5] used input scaffolds to help infer 3D curves.

Single-view based modeling investigates how to reduce the ambiguity of user input. In Ref. [6] primitives were fitted to the input sketch, while Refs. [7, 8] studied how artists create designs, using profile curves or construction lines to represent 3D shapes. Various geometric constraints were investigated in Ref. [8]; Ref. [9] performed sketch based modeling on RGB-D images. A data-driven approach [10, 11] has also been used to deform stock models to fit the user input.

2.2 Single image based modeling

Since a single image does not provide full 3D information, most single image based modeling methods require user interaction, and can be classified into two categories.

One category allows users to draw construction lines or curves on the picture, providing the computer with information or constraints on the underlying 3D shape. These methods are often correlated with sketch based modeling, and generally require the object being modeled to have rather regular geometry. For example, Ref. [12] proposed a method for modeling cuboids with detected or manually specified corners; Ref. [13] modeled sweep objects and Ref. [14] modeled extruded objects, both requiring manually sketched cross sections, and a "main axis", a line orthogonal to the sections. The requirement for such manual input can be lessened by making use of semantic content in the image, using neural networks, as in Ref. [15]. Our work is closely related to these methods, especially Refs. [13, 14], which focus on intuitive modeling based on user sketching. However, our method is not limited to restricted types of shapes such as swept surfaces or extruded surfaces, but can deal with freeform shapes with more complicated geometry by benefiting from the flexibility and generality of lofting surfaces.

The other category relies strongly on existing template shapes: Ref. [16] used a small collection of models to guide the reconstruction process, while Refs. [17, 18] both deformed a candidate model to fit the 2D target shape under the guidance of part correspondences.

Other methods restrict this problem to a specific class of objects with strong in-class similarities: Refs. [19–21] considered architecture, Ref. [22] focused on garments, while Ref. [23] investigated tree modeling from a single image.

2.3 Lofting surfaces

Lofting is a popular technique for constructing freeform surfaces and has been widely studied in computer-aided design. In this approach, the user defines a curve network first, and a lofting surface is automatically constructed to interpolate this network. Early work [24] focused on finding flexible representations of freeform shapes. New types of patches such as those in Ref. [25] were later proposed, and subdivision surfaces [26–29] became a useful tool for this task. In recent years, the idea of using curve networks has also inspired sketch based modeling work such as Refs. [5, 8]. While lofting surfaces have been widely used in computer-aided design, they typically involve the specification of a rather complex curve system, which requires substantial user effort and professional modeling skills. We are the first to apply this technique for intuitive shape modeling from a single image.

3 Modeling a single lofting surface

3.1 Preliminaries

In most CAD systems, in order to create a lofting surface, the user needs to draw a set of parallel curves representing the surface cross sections, and optional guide curves connecting them. The lofting algorithm then automatically generates a 3D surface passing through all these inputs. As shown in Fig. 1, our definition of a lofting surface is adopted from the one used in existing CAD systems. Note that in our approach, the guide curves are replaced either by detected object silhouettes in the image, or manually assigned point correspondences between adjacent cross sections. Since sketching of the lofting shape is performed in 2D, we further define a "main axis" of the shape which is orthogonal to all cross sections, providing additional geometric cues for shape generation in 3D.

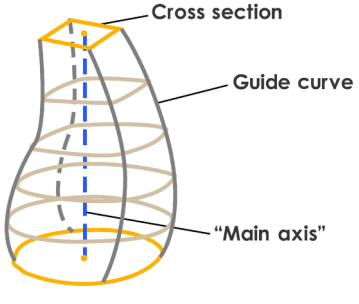


Fig. 1 A lofting surface. Differing from the concept of lofting surfaces used in CAD systems, we additionally define a "main axis" orthogonal to the cross sections.

To model a single lofting shape in a 2D image, the user needs to sketch the parallel cross sections, and the main axis which intersects each section. The user is allowed to specify geometric constraints linking adjacent sections, providing additional cues about the 3D shape (see Section 3.2). Our algorithm then estimates the sections' orientation and relative positions in 3D (see Section 3.3), and constructs the 3D geometry of the surface by interpolating between the sections (see Section 3.4). The point correspondences between adjacent sections can help to guide the interpolation process, making the generated shape more accurate and appealing.

3.2 User interface

Figure 2 shows a typical user interaction session. Given an input image, as in Fig. 2(a)), the user can optionally choose to use either the default camera parameters set in our system, or to input their own calibrated camera parameters. The user first sketches a set of curves γ_i on the image, representing the parallel cross sections Γ_i with different shapes in 3D (see Fig. 2(b)). Each curve is re-sampled as an ordered sequence of points. Then the user draws the

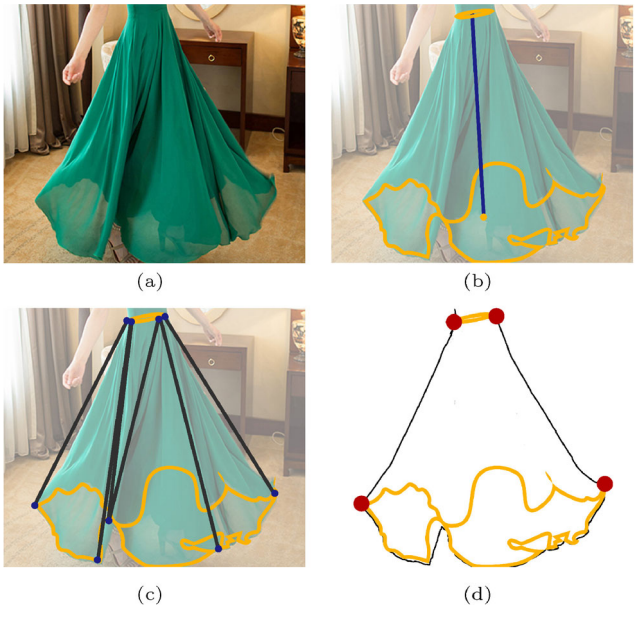


Fig. 2 User input. (a) Original image. (b) User sketched curves representing cross sections (shown in yellow) and the main axis (shown in blue). (c) Point correspondences between sections, with the related points shown in blue and their connections shown in dark gray. (d) Manually assigned boundary points (in red) show where the input curves meet the shape silhouette in the image.

"main axis" in some direction by multiple mouse clicks, each of which represents an intersection point with one of the cross sections. Geometric constraints and point correspondences between adjacent curves can be specified by selecting curve points and drawing lines to connect them (see Fig. 2(c)). Finally, the user assigns two boundary points on each curve to denote the location where the curve meets the object boundary on the image (see Fig. 2(d)). Note that this step can also be automated, but the result may be susceptible to edge detection errors.

3.3 Finding the section normals

3.3.1 Outline

Given a set of parallel sections projected into the 2D image, we use a similar method to the one in Ref. [14] to compute their common normal vector in 3D. This solves an optimization problem subject to three types of geometric constraints defined by the user, including orthogonality, parallelism, and coplanarity (see Fig. 3).

Given a normal vector $\boldsymbol{n}$, the fitness function is defined as a weighted sum of three terms involving individual constraints:

$$f(\boldsymbol{n}) = w_{\mathrm{o}} \sum_{i=1}^{n_{\mathrm{o}}} f_{\mathrm{o}}^{i}(\boldsymbol{n}) + w_{\mathrm{p}} \sum_{i=1}^{n_{\mathrm{p}}} f_{\mathrm{p}}^{i}(\boldsymbol{n}) + w_{\mathrm{c}} \sum_{i=1}^{n_{\mathrm{c}}} f_{\mathrm{c}}^{i}(\boldsymbol{n})$$

where f_{o}^{i}, f_{p}^{i}, and f_{c}^{i} denote orthogonality, parallelism, coplanarity constraints respectively, and n_{o}, n_{p}, n_{c} are the numbers of individual constraints of each kind. In our implementation we set $w_{\mathrm{o}} = w_{\mathrm{p}} = w_{\mathrm{c}} = 1.0$.

3.3.2 Orthogonality

The orthogonality constraint restricts the spatial relation between two line segments lying on the same section. As three perpendicular lines form an xyz-coordinate system projected into the image space, one can solve for their positions in 3D up to a scalar [13]. The orthogonality relationship defined in the carrying plane of a section can greatly reduce the ambiguity when inferring shapes from a single image. Furthermore, lofted shapes often have one or more rectangular cross sections, where the orthogonality constraint can help to accurately preserve their prominent features.

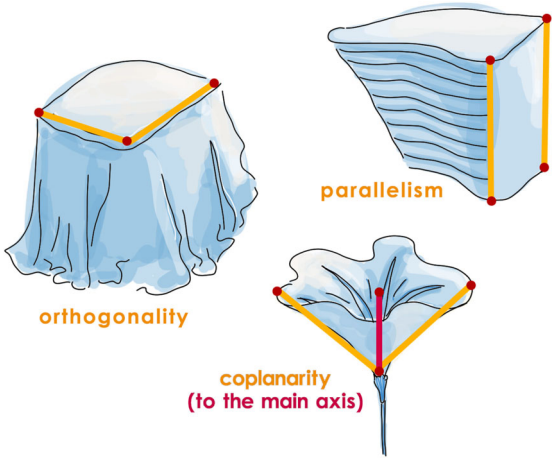


Fig. 3 User specified geometric constraints.

Suppose l_1, l_2 are two orthogonal line segments projected into image space, and $l_1(\boldsymbol{n})$, $l_2(\boldsymbol{n})$ are their 3D counterparts. Then the orthogonality term is defined as

$$f_{\mathrm{o}}(\boldsymbol{n}, l_1, l_2) = \exp\left(-\frac{(\theta(l_1(\boldsymbol{n}), l_2(\boldsymbol{n})) - \pi/2)^2}{\sigma^2}\right)$$

where $\theta(\cdot,\cdot)$ is the angle between two line segments in 3D. In our implementation we use $\sigma = 0.1$.

3.3.3 Parallelism

Similarly to orthogonality, the term for two parallel line segments is defined as

$$f_{\mathrm{p}}(\boldsymbol{n}, l_1, l_2) = \exp\left(-\frac{\theta(l_1(\boldsymbol{n}), l_2(\boldsymbol{n}))^2}{\sigma^2}\right)$$

To simplify user interaction, we only consider the situation in which one of the line segments is the main axis in our implementation.

3.3.4 Coplanarity

While parallelism is a special case of coplanarity, it is non-trivial to clearly define the general form of coplanar relationships in a 2D image. Therefore, we simply consider the situation where a line is both coplanar to the main axis and parallel to the viewing plane. This can be easily determined and gives a hint to the relative sizes of different sections.

Suppose l is a line segment on the image with the above properties, and $l(\boldsymbol{n})$ denotes its 3D counterpart. We expect the angle between l and the main axis L to be constant under perspective projection. Hence, the coplanarity term for l may be defined as

$$f_{\mathrm{c}}(\boldsymbol{n}, l) = \exp\left(-\frac{(\theta(l(\boldsymbol{n}), L(\boldsymbol{n})) - \theta(l, L))^2}{\sigma^2}\right)$$

3.3.5 Optimization for $\boldsymbol{n}$

The overall fitness function is highly non-linear, so optimization can be easily trapped in a local maximum. Therefore, we simply sample a set of normal vectors uniformly on a Gaussian sphere, and select the one that maximizes the function as

the optimal solution. In practice, the result works well enough to meet our requirement for efficient modeling.

Although theoretically $\boldsymbol{n}$ can be anywhere on the hemisphere (since one plane has two opposite normal vectors), the 2D main axis limits the solution space to a semi-circle defined by the intersection of the hemisphere and the 3D plane formed by L on the image plane and the camera origin [14]. This greatly reduces our sampling space.

If the geometric constraints are too few, the optimization method may result in more than one solution, for the same reason as the back projection problem with three orthogonal lines. In this case, we divide the semicircle for normal sampling into two quarters and run the optimization on both, providing two possible solutions for the user to choose from.

3.4 Interpolation between adjacent sections

In order to model a complete shape that fits the image silhouettes from the user specified cross sections, we simply compute a set of intermediate sections between the known ones. The interpolation process is carried out in the following two steps.

3.4.1 Shape transformation

In the first stage, we use 2D shape blending methods [30, 31] to compute natural transitions between adjacent section curves. Due to the unclear definition of "natural", there are usually multiple solutions to this problem, and the one obtained using the above algorithms may not be desirable. Therefore, we take into account the manually specified point correspondences between adjacent curves. The specified points divide each curve into several segments and we apply the method in Ref. [30] to each segment separately.

3.4.2 Image fitting

In the second stage, we adjust the sizes of the intermediate sections obtained above to match the shape silhouette in 2D (see Fig. 4). For each input 2D section curve γ_i, two manually determined points match the shape silhouette on the image, while for an interpolated curve $\bar{\gamma}_j$, we hope to find two boundary points $\bar{p}_j^0$, $\bar{p}_j^1$ automatically in the image space. For this purpose, we first compute two approximate points $\hat{p}_j^0$, $\hat{p}_j^1$ for each $\bar{\gamma}_j$ in 2D, by linear interpolation of boundary points on the input curves. Then we cast a 2D ray through $\hat{p}_j^0$, $\hat{p}_j^1$, seeking intersections with the detected 2D shape silhouette for use as the boundary points $\bar{p}_j^0$, $\bar{p}_j^1$ for this curve.

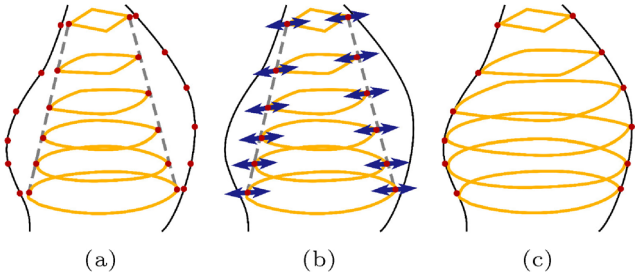


Fig. 4 Adjusting the intermediate curves to fit the silhouette. (a) We first compute two approximate boundary points for each curve by linear interpolation and use them to find the real boundary points. (b) We expand each curve on its underlying plane in 3D. (c) The resulting shape; sections' sizes match the silhouette.

Having found the boundary points, their positions in 3D, denoted by $\bar{P}_j^0$, $\bar{P}_j^1$, can be uniquely determined on $\bar{\Gamma}_j$'s underlying plane (see Fig. 5(a)). We then adjust the translation and scaling of the curve $\bar{\Gamma}_j$ in 3D to ensure that it touches $\bar{K}_j^0$ and $\bar{K}_j^1$, the two lines passing through $\bar{P}_j^0$, $\bar{P}_j^1$ and perpendicular to $\bar{P}_j^0\bar{P}_j^1$, as shown in Fig. 5(b).

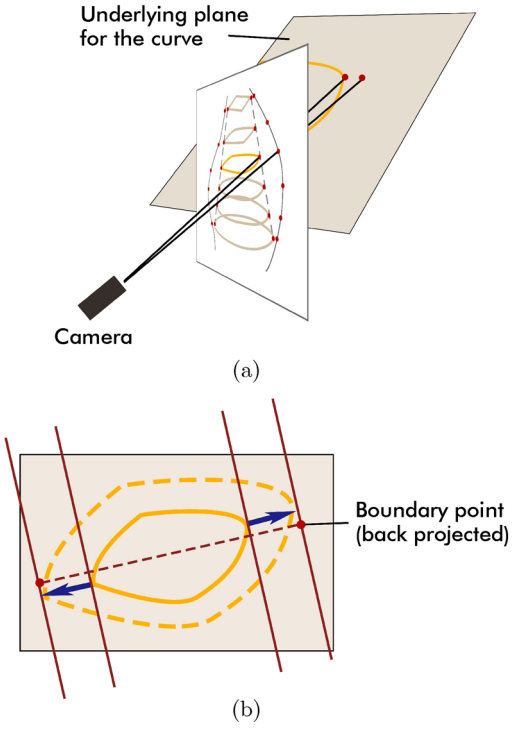


Fig. 5 Transforming a section curve in 3D to fit the shape silhouette in 2D. (a) The positions of the boundary points in 3D are uniquely determined by intersecting the camera ray with the underlying plane of the curve (light brown). (b) A shape-preserving scaling of the curve on its underlying plane is guided by the back projected boundary points. The four red solid lines are orthogonal to the red dotted lines.

4 Modeling multiple parts

4.1 Basics

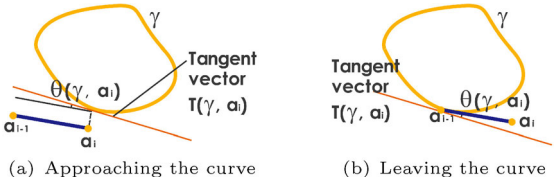
In addition to modeling a single lofting surface, we allow the user to model objects composed of several lofting surfaces, each of which can be modeled using the method described above. The main problem is now estimating relative depths of different parts. In our method, we let the user specify a junction between two parts, i.e., the point where the two parts overlap, following Ref. [7]. Since occlusions often exist in images where an object consists of multiple parts, we allow the user to modify the detected 2D shape boundaries, in order to resolve partially occluded boundaries.

4.2 Tracing existing curves

In many cases, different parts of an object may have cross sections sharing the same outline, or part of the outline. To better stitch different parts, we allow the user to trace existing curves when modeling new parts, with a snapping feature akin to the magnetic lasso tool in PhotoShop. The normal vector of the new section is taken to be the same as the original one.

In the curve tracing process, the user first selects an existing curve γ and then starts sketching. During the process, let $\{a_0, \cdots, a_{i-1}\}$ be the points already drawn on the canvas, where a_i is the most recent point. To decide whether a_i should be snapped to the curve, we consider the distance $d(\gamma, a_i)$ between a_i and γ measured in pixels, and the angle $\theta(\gamma, a_i)$ (we assume smoothness for the new curve), as shown in Fig. 6.

Inspired by the hysteresis effect in Ref. [32], we need to handle two cases, approaching the traced curve, or leaving the curve, using different thresholds. In our implementation, $d(\gamma, a_i) < 30$, $\theta(\gamma, a_i) < 20$ for the former case, while $d(\gamma, a_i) < 5$ or $d(\gamma, a_i) < 30$, $\theta(\gamma, a_i) < 10$ for the latter.



(a) Approaching the curve (b) Leaving the curve

Fig. 6 Two different cases for tracing existing curves when modeling new parts.

5 Results and discussion

5.1 Results

We have tested our work on a variety of images including different types of objects, ranging from outdoor natural objects to indoor man-made objects. A wide range of freeform objects can be modeled using our method with just a few user interactions (see Figs. 7 and 8). The modeling process is simple and intuitive, each model being created within a couple of minutes.

Figure 7 shows some single lofted shapes generated with our system. The shapes of these objects are very irregular, but their cross sections are easy to sketch from the image. Correspondences are assigned according to the folds and textures on the surface.

Figure 8 shows results with multiple parts. Figures 8(a) and 8(b) show cases where the outlines of two cross sections in different parts overlap: the seams of the dress in orange in Fig. 8(a), and the bottoms of the two mountains in Fig. 8(b). After selecting an existing curve, the user can easily draw a new one with the same normal vector using our snapping tool. Objects in Figs. 8(c) and 8(d) consist of two parts sharing a common cross section: the top surfaces of each object. After modeling one part, the user can select and add an existing curve directly to another part, which helps to determine the orientation of the other part at the same time. In Fig. 8(e), a junction line is added between the receptacles of the two morning glories, giving them the same depth.

Table 1 summarizes the modeling time taken as well as the numbers of input curves, correspondences, and constraints for each example; these models were made by a novice user with minimal training.

We have also used synthetic images to quantitatively evaluate the modeling accuracy of our method. Using SolidWorks CAD software, we first modeled ground truth lofted surfaces with shape complexity comparable to that found in real images. We then rendered the ground truth surfaces to generate 2D synthetic images. Finally, the user carefully sketched over the synthetic images (e.g., drawing section curves by tracing the surface boundary), and generated 3D shapes using our system. As shown in Fig. 9, the differences between the aligned ground-truth shape and the interactively modeled shape is very small (a few percent of the diagonal length of the shape bounding box).

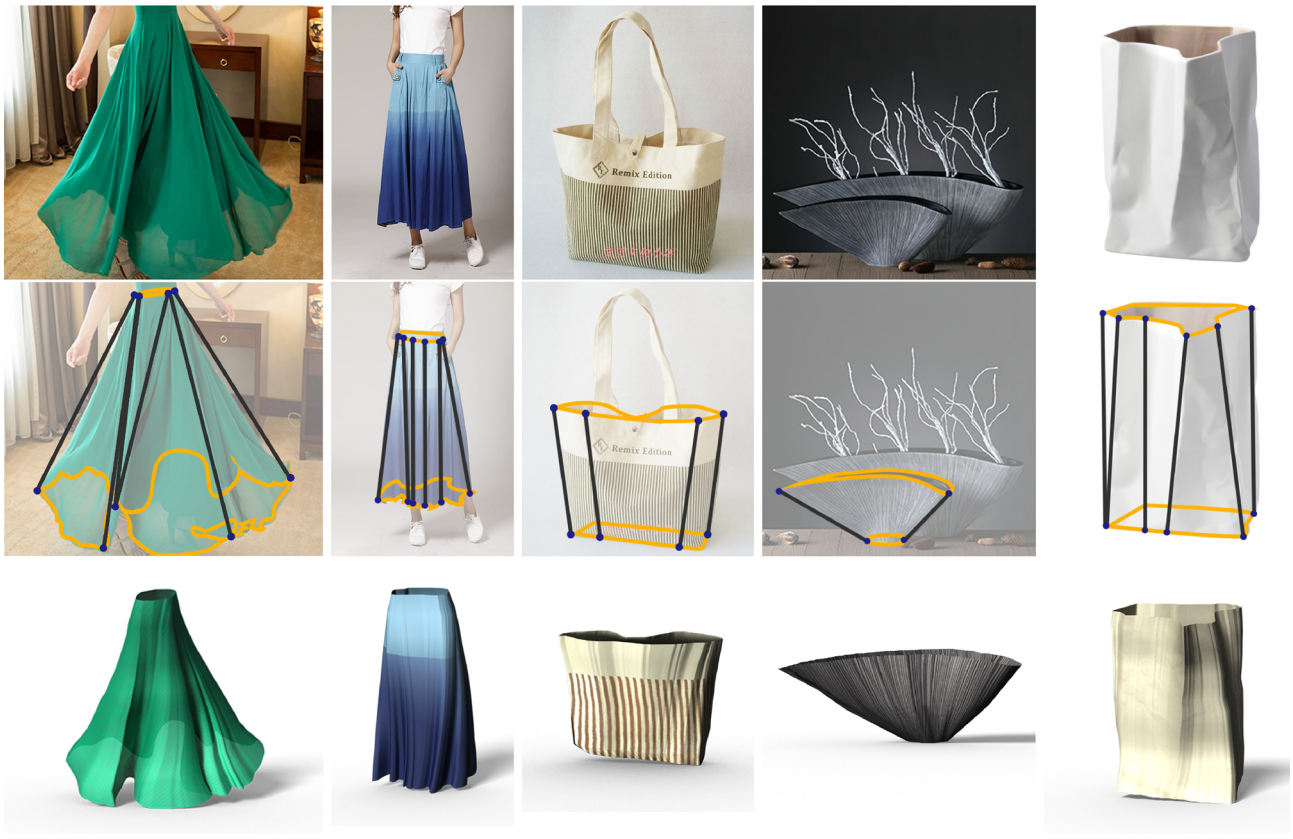


Fig. 7 Modeling results using our method. Top: input images. Middle: cross sections sketched by the user and points in correspondence. Bottom: reconstructed 3D models.

Table 1 Modeling time and numbers of input curves, correspondences, and constraints for each example

	Modeling single surface					Modeling multiple marts				
Example	Green dress	Blue dress	Bag	Vase	Paper bag	Dress	Mountains	Table cloth	Table	Flowers
Time (s)	73	93	77	50	79	224	315	246	174	186
Curve	2	2	2	2	2	3+3	5+4	2+2	3+2	3+3
Correspondence	6	6	4	2	6	10+4	17+11	10+13	6+2	5+5
Constraint	3	1	2	2	2	5	8	4	1	1+1

Note that in our current approach, textures are manually assigned. To do so automatically, we could adapt the illumination estimation and appearance completion method in Ref. [18] by relaxing the global symmetry constraints to partial symmetry [33, 34].

5.2 Limitations

Our tool makes several assumptions that may not hold in certain cases. Firstly, each part of the object must be simply connected and have no more complex topology than an annulus. Secondly, section orientation estimation depends on manually assigned geometric constraints. This means that the underlying shape of the target object should be easily inferred from the image without causing any ambiguity for the user.

6 Conclusions

In this paper, we have presented a novel interactive technique for modeling lofted shapes from a single image. Unlike existing modeling methods, we allow the user to model freeform shapes with complicated geometry with simple interactions, where only a few sketches and relationship assignments are required. The results show that a great variety of natural or man-made objects can be easily modeled with our system, and the modeling accuracy is satisfactory.

A possible extension of our work is to apply it in novel image editing applications that cannot be done using previous approaches, as the modeled complete geometry enables novel view synthesis, relighting, etc. We may also adopt shape editing

Fig. 8 Modeling lofted shapes with multiple parts. Left: input images. Middle: input curves representing the outlines of cross sections. Right: reconstructed 3D models.

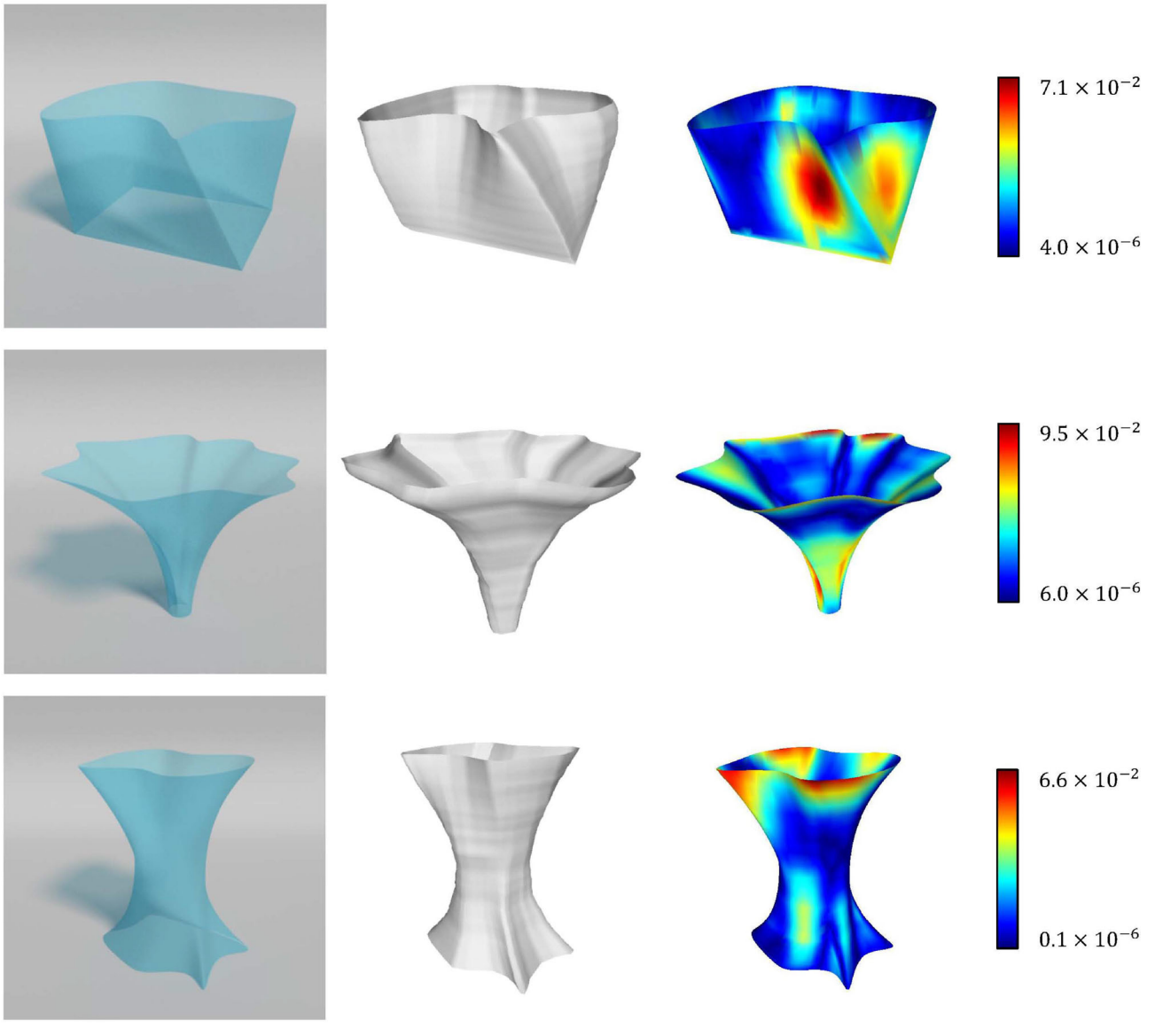


Fig. 9 Quantitative evaluation of our modeling results. Left: rendered synthetic images of the ground truth shapes. Middle: reconstructed shapes using our method. Right: visualization of the geometric errors between the shapes.

and physical simulation techniques to deform the shape in 3D and therefore create realistic 2D editing on the projected image. Another future project worth considering is to combine our interactive method with single-view reconstruction in computer vision. If reliable depth or surface normal information can be computationally recovered, we may use it to simplify user interaction and guide shape interpolation. Further, by introducing a more complex section curve structure, we may construct shapes with more complicated topology or geometry at the cost of additional user interaction.

References

[1] Ding, C.; Liu, L. G. A survey of sketch based modeling systems. *Frontiers of Computer Science* Vol. 10, No. 6, 985–999, 2016.

[2] Igarashi, T.; Matsuoka, S.; Tanaka, H. Teddy: A sketching interface for 3D freeform design. In: Proceedings of the ACM SIGGRAPH courses, Article No. 21, 2007.

[3] Schmidt, R.; Wyvill, B.; Sousa, M. C.; Jorge, J. A. ShapeShop: Sketch-based solid modeling with Blob Trees. In: Proceedings of the ACM SIGGRAPH Courses, Article No. 14, 2006.

[4] Li, C.-J.; Pan, H.; Liu, Y.; Tong, X.; Sheffer, A.; Wang, W. BendSketch: Modeling freeform surfaces through 2D sketching. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 125, 2017.

[5] Schmidt, R.; Khan, A.; Singh, K.; Kurtenbach, G. Analytic drawing of 3D scaffolds. *ACM Transactions on Graphics* Vol. 28, No. 5, Article No. 149, 2009.

[6] Shtof, A.; Agathos, A.; Gingold, Y.; Shamir, A.; Cohen-Or, D. Geosemantic snapping for sketch-based modeling. *Computer Graphics Forum* Vol. 32, No. 2pt2, 245–253, 2013.

[7] Andre, A.; Saito, S. Single-view sketch based modeling. In: Proceedings of the 8th Eurographics Symposium on Sketch-Based Interfaces and Modeling, 133–140, 2011.

[8] Xu, B.-X.; Chang, W.; Sheffer, A.; Bousseau, A.; McCrae, J.; Singh, K. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 131, 2014.

[9] Li, Y.; Luo, X.; Zheng, Y.; Xu, P.; Fu, H. SweepCanvas: Sketch-based 3D prototyping on an RGB-D image. In: Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, 387–399, 2017.

[10] Xu, K.; Chen, K.; Fu, H; Sun, W.-L.; Hu, S.-M. Sketch2Scene: Sketch-based co-retrieval and co-placement of 3D models. *ACM Transactions on Graphics* Vol. 32, No. 4, Article No. 123, 2013.

[11] Xie, X. H.; Xu, K.; Mitra, N. J.; Cohen-Or, D.; Gong, W. Y.; Su, Q.; Chen, B. Sketch-to-design: Context-based part assembly. *Computer Graphics Forum* Vol. 32, No. 8, 233–245, 2013.

[12] Zheng, Y.; Chen, X.; Cheng, M.-M.; Zhou, K.; Hu, S.-M.; Mitra, N. J. Interactive images: Cuboid proxies for smart image manipulation. *ACM Transactions on Graphics* Vol. 31, No. 4, Article No. 99, 2012.

[13] Chen, T.; Zhu, Z.; Shamir, A.; Hu, S.-M.; Cohen-Or, D. 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics* Vol. 32, No. 6, Article No. 195, 2013.

[14] Cao, Y.-P.; Ju, T.; Fu, Z.; Hu, S.-M. Interactive image-guided modeling of extruded shapes. *Computer Graphics Forum* Vol. 33, No. 7, 101–110, 2014.

[15] Xin, C.; Li, Y. W.; Luo, X.; Shao, T. J.; Yu, J. Y.; Zhou, K.; Zheng, Y. AutoSweep: Recovering 3D editable objects from a single photograph. *IEEE Transactions on Visualization and Computer Graphics* DOI: 10.1109/TVCG.2018.2871190, 2018.

[16] Huang, Q.-X.; Wang, H.; Koltun, V. Single-view reconstruction via joint analysis of image and shape collections. *ACM Transactions on Graphics* Vol. 34, No. 4, Article No. 87, 2015.

[17] Xu, K.; Zheng, H. L.; Zhang, H.; Cohen-Or, D.; Liu, L. G.; Xiong, Y. S. Photo-inspired model-driven 3D object modeling. *ACM Transactions on Graphics* Vol. 30, No. 4, Article No. 80, 2011.

[18] Kholgade, N.; Simon, T.; Efros, A.; Sheikh, Y. 3D object manipulation in a single photograph using stock 3D models. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 127, 2014.

[19] Debevec, P. E.; Taylor, C. J.; Malik, J. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In: Proceedings of the SIGGRAPH 96 Conference, 1996. Available at https://www.pauldebevec.com/Research/debevec-siggraph96-paper.pdf.

[20] Jiang, N. J.; Tan, P.; Cheong, L. F. Symmetric architecture modeling with a single image. *ACM Transactions on Graphics* Vol. 28, No. 5, Article No. 113, 2009.

[21] Nishida, G.; Bousseau, A.; Aliaga, D. G. Procedural modeling of a building from a single image. *Computer Graphics Forum* Vol. 37, No. 2, 415–429, 2018.

[22] Zhou, B.; Chen, X. W.; Fu, Q.; Guo, K.; Tan, P. Garment modeling from a single image. *Computer Graphics Forum* Vol. 32, No. 7, 85–91, 2013.

[23] Tan, P.; Fang, T.; Xiao, J.; Zhao, P.; Quan, L. Single image tree modeling. *ACM Transactions on Graphics* Vol. 27, No. 5, Article No. 108, 2008.

[24] Chiyokura, H.; Kimura, F. Design of solids with free-form surfaces. *ACM SIGGRAPH Computer Graphics* Vol. 17, No. 3, 289–298, 1983.

[25] Hermann, T. G2 interpolation of free form curve networks by biquintic Gregory patches. *Computer Aided Geometric Design* Vol. 13, No. 9, 873–893, 1996.

[26] Catmull, E.; Clark, J. Recursively generated B-spline surfaces on arbitrary topological meshes. In: *Seminal Graphics*. ACM, 183–188, 1998.

[27] Nasri, A. Curve interpolation in recursively generated B-spline surfaces over arbitrary topology. *Computer Aided Geometric Design* Vol. 14, No. 1, 13–30, 1997.

[28] Nasri, A. H. Interpolating an unlimited number of curves meeting at extraordinary points on subdivision surfaces. *Computer Graphics Forum* Vol. 22, No. 1, 87–97, 2003.

[29] Schaefer, S.; Warren, J. D.; Zorin, D. Lofting curve networks using subdivision surfaces. In: Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing, 103–114, 2004.

[30] Sederberg, T. W.; Greenwood, E. A physically based approach to 2-D shape blending. *ACM SIGGRAPH Computer Graphics* Vol. 26, No. 2, 25–34, 1992.

[31] Sederberg, T. W.; Gao, P.; Wang, G.; Mu, H. 2-D shape blending: An intrinsic solution to the vertex path problem. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, 15–18, 1993.

[32] Watanabe, N.; Washida, M.; Igarashi, T. Bubble clusters: An interface for manipulating spatial aggregation of graphical objects. In: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, 173–182, 2007.

[33] Mitra, N. J.; Guibas, L. J.; Pauly, M. Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics* Vol. 25, No. 3, 560–568, 2006.

[34] Shi, Z. Y.; Alliez, P.; Desbrun, M.; Bao, H. J.; Huang, J. Symmetry and orbit detection via lie-algebra voting. *Computer Graphics Forum* Vol. 35, No. 5, 217–227, 2016.