WireRoom: Model-guided Explorative Design of Abstract Wire Art

ZHIJIN YANG, Shenzhen University
PENGFEI XU*, Shenzhen University
HONGBO FU, City University of Hong Kong
HUI HUANG, Shenzhen University

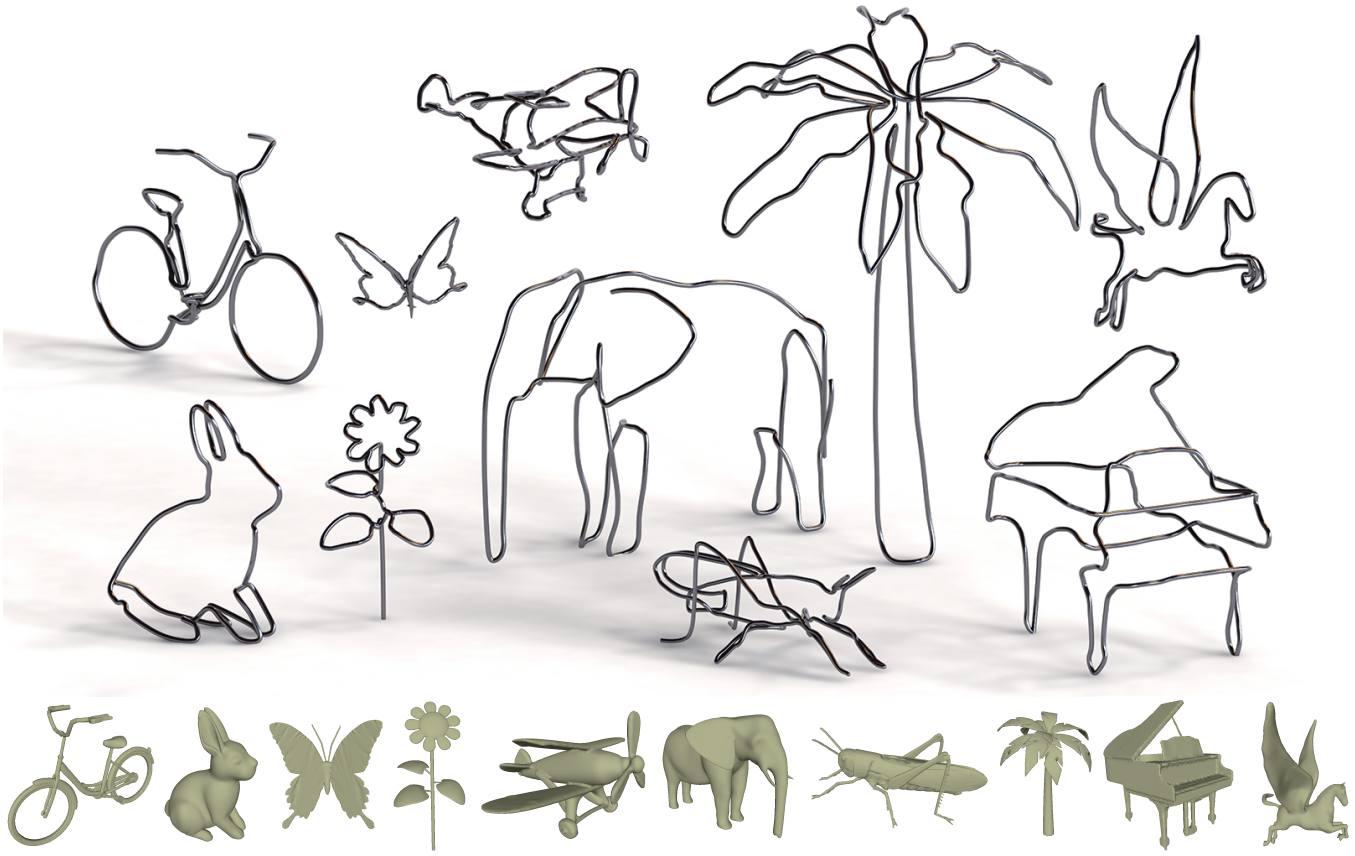

Fig. 1. The wire shapes interactively designed with our *WireRoom* framework. All these wire shapes are with a single-wire structure. Bottom: the input models used for guiding the creation of these wire shapes.

*Corresponding author: Pengfei Xu (xupengfei.cg@gmail.com)

Authors' addresses: Zhijin Yang, yangzhijin1998@gmail.com, College of Computer Science & Software Engineering, Shenzhen University; Pengfei Xu, xupengfei.cg@gmail.com, College of Computer Science & Software Engineering, Shenzhen University; Hongbo Fu, hongbofu@cityu.edu.hk, School of Creative Media, City University of Hong Kong; Hui Huang, hhzhiyan@gmail.com, College of Computer Science & Software Engineering, Shenzhen University.

0730-0301/2021/8-ART128 $15.00
https://doi.org/10.1145/3450626.3459796

We present *WireRoom*, a computational framework for the intelligent design of abstract 3D wire art to depict a given 3D model. Our algorithm generates a set of 3D wire shapes from the 3D model with informative, visually pleasing, and concise structures. It is achieved by solving a dynamic travelling salesman problem on the surface of the 3D model with a multi-path expansion approach. We introduce a novel explorative computational design procedure by taking the generated wire shapes as candidates, avoiding manual design of the wire shape structure. We compare our algorithm with a baseline method and conduct a user study to investigate the usability of the framework and the quality of the produced wire shapes. The results of the comparison and user study confirm that our framework is effective for producing informative, visually pleasing, and concise wire shapes.

CCS Concepts: • **Computing methodologies** → **Shape modeling**; • **Applied computing** → **Fine arts**.

Additional Key Words and Phrases: abstract wire art, wire modeling, computational design

ACM Reference Format:
Zhijin Yang, Pengfei Xu, Hongbo Fu, and Hui Huang. 2021. *WireRoom*: Model-guided Explorative Design of Abstract Wire Art. *ACM Trans. Graph.* 40, 4, Article 128 (August 2021), 13 pages. https://doi.org/10.1145/3450626.3459796

1 INTRODUCTION

Wire art is a special art form which adopts physical wires (e.g., metal wires) to represent 2D and 3D shapes, and is adopted for various purposes, e.g, decorative design. While the use of wires as primitive elements is a common nature of wire art, as shown in Figure 2, wire artworks created by different artists often exhibit different styles. For example, artists may use wires extensively to approximate the surface of a 3D model (e.g., the Jackrabbit created by Ruth Jensen[1] in Figure 2 (left)), or use wires moderately to abstractly depict 3D and 2D shapes (e.g., the Deer created by Bud Bullivant[2] in Figure 2 (middle) and the Cat created by Maria Szabo[3] in Figure 2 (right)). Manual design and creation of these artworks require extensive effort of artists. A valuable question is can we assist wire art design with computational frameworks?

A few attempts have been made to address this question. For example, Zehnder et al. [2016] presented a framework for designing curve networks with a given curve pattern to approximate a 3D surface (e.g, similar to the Jackrabbit in Figure 2 (left)). Iarussi et al. [2015] proposed a framework to assist the design and fabrication of 2D wire art (e.g., similar to the Cat in Figure 2 (right)). While for designing a concise 3D wire artwork abstractly depicting a 3D content, e.g, the Deer created by Bud Bullivant, it still heavily depends on an artist's creativity and professional skills.

Developing a computational framework for assisting the creation of desired abstract 3D wire artwork is challenging for the following reasons. Given a 3D content, e.g, a 3D model, it is nontrivial to generate a concise 3D wire with a simple structure, e.g., with a single-wire structure similar to the Deer in Figure 2 (middle). It might be easy to produce scattered 3D curves to depict a given 3D content with existing techniques, e.g., using line drawing rendering techniques [DeCarlo et al. 2003; Haralick 1983; Judd et al. 2007] to extract 3D curves from a given 3D model. However, it is unknown how to form a concise connected structure with these 3D curves. Simply connecting these curves might result in a 3D curve network with a complex structure (see Figure 12). On the other hand, since a 3D wire artwork is an extremely abstracted representation of a 3D content, its shape is highly unconstrained and different people may have their own preferences during the creation (see the user-created wire shapes in the supplemental materials). It is challenging to generate desired wire artworks which satisfy the personal demand of users with a moderate amount of user intervention.

To address these problems, we present *WireRoom*, a computational framework for the design of abstract 3D wire art with a single-wire structure, as shown in Figure 1. Our contributions are two-fold: first, we propose an algorithm to automatically generate a set of 3D wire shapes from a 3D model with informative, visually pleasing, and concise structures. We formulate the wire shape generation procedure as a dynamic travelling salesman problem (DTSP) [Punnen 2007] on the surface. To avoid heavy computation in finding the optimal solution and provide an adequate number of 3D wire shapes with satisfactory structures, we adopt a greedy strategy to find multiple path branches with an evaluation criteria which favors informative and concise paths. Second, we introduce a novel explorative computational design procedure. This procedure avoids manual design of the wire shape structure, the most sophisticated step when creating a wire artwork, and provides sufficient wire candidates to satisfy the personal preferences of different users. Users may select the most satisfied wire shape and perform subsequent interactive editing to create a desired wire artwork with our addition, erasure, and manipulation tools.

We have compared our automatic wire generation algorithm with a baseline solution and the results confirm the robustness of our algorithm. We conducted a user study to investigate the usability of our framework and the quality of the created wire shapes. Both the statistic data and user feedback suggest that our framework is effective for wire shape creation and the quality of wire shapes generated with our algorithm is comparable with or even superior to the manually designed ones by the user study participants. We also introduce a wire deformation procedure and provide a printed fabrication plan to ease manual realization of wire shapes with metal wires. A simple user test shows that such instructions reduce the user effort for the fabrication.

Fig. 2. Wire artworks often exhibit different styles. Left: the Jackrabbit by ©Ruth Jensen. Middle: the Deer by ©Bud Bullivant. Right: the Cat by ©Maria Szabo.

2 RELATED WORK

Our work is not the first attempt to create wires or curves to depict a 3D content. Most of the existing works have focused on interactive creation of 3D curves. They may take 2D input with additional spatial constraints (e.g., 3D planes [Dorsey et al. 2007; Xu et al. 2018], 3D surfaces [Bae et al. 2008], 3D scaffolds [Schmidt et al. 2009], relations among curves [Xu et al. 2014]), 3D input with spatial positioning devices (e.g., hand-held controller [Rosales et al. 2019], mobile [Kwan and Fu 2019; Ye et al. 2021]), or 2D and 3D input combined in VR [Drey et al. 2020] and AR [Arora et al. 2018] configurations. For a more thorough review of such techniques, please refer to [Bhattacharjee and Chaudhuri 2020]. With these techniques, it is possible to create any desired 3D wire shapes. However, the interactive creation process heavily depends on the creative design and manual operations of users, making such tools not very friendly for ordinary users, particularly those with little training in art and

[1]https://www.flickr.com/photos/sparkflight/
[2]http://WiredbyBud.etsy.com/
[3]http://turanianwalk.etsy.com/

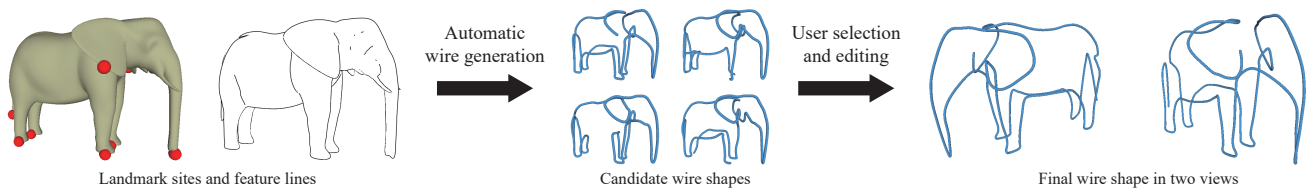

Fig. 3. The workflow of our framework. Given an input model, we first extract the landmark sites and the feature lines (left). Then our automatic wire generation algorithm returns a set of wire shapes (middle). The user may select the most desired one and perform subsequent edits to obtain the final wire shape (right, displayed in two views).

design. In contrast, our framework is able to first provide a set of wire shape candidates from a reference 3D model, and need only a moderate amount of additional editing to obtain a desired wire shape.

Generating 3D curves or wires from a model is an interesting problem and a number of techniques have been proposed for various applications. Some works (e.g., ridges and valley [Haralick 1983], suggestive contour [DeCarlo et al. 2003], apparent ridges [Judd et al. 2007]) extract scattered curves on a 3D model surface to achieve non-photorealistic rendering effects. The curves extracted by these techniques can successfully convey the content of a given model in an abstract manner. But it is unclear how to form a connected curve with a concise structure. Instead of extracting scattered curve segments, more works have attempted to form curve networks on a 3D surface. The structure of the formed curve networks can be determined by exemplars [Chen et al. 2016; Dumas et al. 2015; Zehnder et al. 2016], or according to fabrication means (e.g., robotic arm [Huang et al. 2016; Ma et al. 2020], 5DOF wireframe printer [Wu et al. 2016], standard FDM 3D printer [Mueller et al. 2014], manual weaving [Tao et al. 2017; Vekhter et al. 2019]). In addition, some works decompose a 3D surface into patches with curve networks as micro structures [Garg et al. 2014; Malomo et al. 2018]. The curves or wires in these techniques are employed extensively to form relatively dense structures, which are distinct from abstract wire structures produced by our framework.

Several works have attempted to extract abstract curve networks or wire shapes from 3D shapes. The extracted abstract curve networks can be considered as an alternative representation of 3D models. For example, De Goes et al. [2011] presented a method to extract abstract curve networks from 3D shapes by combining shape segmentation and approximation. Gori et al. [2017] introduced an algorithm for extracting descriptive compact curve networks from free-form man-made shapes. Instead of being special representations of 3D shapes, the extracted abstract curve networks can also be utilized for shape processing, e.g. shape abstraction [Mehra et al. 2009] or manipulation [Gal et al. 2009]. Some abstract wire shapes are extracted or created with the consideration of physical fabrication. For example, Miguel et al. [2016] proposed a method for designing abstract wire shapes composed of interlocking planar wires. Liu et al. [2017b] presented WireFab, a framework for modeling and fabricating 3D content in mixed forms including abstract wires. Wang et al. [2019] presented a framework to help the design and fabrication of abstract wire shapes by 3D printing a model with grooves on the surface. Although these works can help to extract abstract wire shapes from 3D shapes, none of them guarantees to produce a single-wire structure.

The most related works to ours might be [Iarussi et al. 2015] and [Lira et al. 2018]. They decompose existing 2D [Iarussi et al. 2015] or 3D [Lira et al. 2018] wire shapes into several Eulerian paths for easier fabrication. Our problem is distinct from theirs since we do not require an existing wire shape as input. Therefore we cannot formulate our problem as a Eulerian path finding procedure. Instead, we establish feasible paths via solving a DTSP [Punnen 2007]. But it is still possible to borrow their formulation by introducing a graph composed of feature curves extracted by existing techniques (e.g., line drawing rendering techniques). However, it is not clear which feature curves should be included in this graph. The Eulerian path travelling a large set of feature curves would result in complex structures (please refer to Section 6.1 for a more detailed comparison).

Another relevant topic is 3D reconstruction of wire shapes. Liu et al. [2017a] presented a method for reconstructing 3D wire shapes from multi-view images. Their follow-up work [Liu et al. 2018] solves a similar problem, but with a sequence of depth images as input. Wang et al. [2020] presented a method for 3D reconstruction of a wire shape from an RGB video. The camera motion associated to the video is estimated simultaneously. Hsiao et al. [2018] introduced a framework for creating multi-view wire shapes, which exhibit multiple contents in different views. Instead of reconstructing wire shapes in digital form, Yue et al. [2017] tried to facilitate the construction of wire shapes in real environments with the guidance of a mixed reality system. These works focus on the conversion of wire shapes from reality to virtual, or vice versa. In contrast, our framework aims to assist the design of novel wire shapes given a 3D surface model.

3 OVERVIEW

Our framework allows a novice user to design an abstract 3D wire artwork from a surface model. As illustrated in Figure 3, the typical workflow is as follows. The user first loads a surface model as guidance for the wire artwork design. He/she may select a camera view to reveal a desired shape of the model. The view-dependent features in this view will be captured and utilized in our algorithm. After running our automatic wire generation algorithm, a set of wire shapes will be returned to user for selection. After picking the most desired one, the user may perform subsequent edits on the wire shape if necessary. In the following, we will first describe the

automatic wire generation algorithm (Section 4), and then describe the interaction interface of the framework (Section 5).

4 AUTOMATIC WIRE GENERATION

4.1 Algorithm

Observations. Our algorithm takes a 3D surface model in the form of triangle mesh as input and automatically generates a set of 3D wire shapes. When devising our algorithm, we observe that a good abstract wire shape usually fulfills the following requirements.

First, the wire shape should be informative. It should capture sufficient features of a target shape so that people can easily recognize the underlying shape expressed by the wire shape. Figure 2 (middle) and (right) show two abstract wire artworks created by artists. Both of them are composed of curves resembling the feature lines of the shapes being depicted, e.g., the contours of deer and cat. Several online courses[4][5] teaching children wire art making also suggest that the creation should start with drawing such feature lines (e.g., contours) of a target shape. Motivated by this observation, we focus on extracting wires from the surface of the input model. The extracted wires should capture the feature lines (i.e., ridges and valleys [Haralick 1983], suggestive contours [DeCarlo et al. 2003], and apparent ridges [Judd et al. 2007] in our implementation) of the model in a specified view.

Second, the wire shape should be concise. A wire shape with massive curves may faithfully retain the content of the input model, but will also cause visual clutter and lose the beauty of abstract art. This is also confirmed by the abstract wire shapes in Figure 2: the curves in those abstract wire shapes do not resemble the feature lines of the target shapes repetitively. A feature line is usually conveyed by a single segment of curve. This observation motivates us to devise a strategy to avoid repetitive exhibition of feature lines when extracting wires.

Third, the wire shape is preferable to have a simple structure, e.g., made of only one single wire. This is not only a requirement of abstract representation, but also reduces the work of wire connection and thus eases the fabrication process. This single-wire structure requirement coincides with the travelling salesman problem, i.e., finding a single-line path travelling given sites.

Problem formulation. Base on the above observations and discussions, we formulate the wire extraction process as a dynamic travelling salesman problem (DTSP) with dynamic distance measurement. To achieve this, we first extract a set of landmark sites on the surface, and find the optimal path travelling all of them along the edges of the input triangle mesh. The landmark sites determine which parts of the shape will be captured by the path. They are selected as the extreme points [Au et al. 2011; Zhang et al. 2008] of the model (see Figure 3). We adopt the algorithm described in [Au et al. 2011] for automatic extraction of landmark sites in our implementation. Users may also directly add or remove the landmark sites for special design purposes.

Defining the optimal path as the shortest path passing all the landmark sites is reasonable since it coincides with the demand of single-wire structure of wire shape. However, directly solving the TSP would result in a path consisting a sequence of geodesic paths between pairs of the landmark sites. As illustrated in Figure 6, such geodesic paths often fail to capture desired feature lines of a target model and therefore damage the recognizability of the generated wire shape. In addition, the geodesic paths may gather on the surface of the model, leading to a redundant structure. To produce informative but not redundant wire shapes, we introduce feature attraction weights and path repulsion weights to warp the triangle surface implicitly (Figure 4) so that the triangle edges near the feature lines are shortened and the edges near an existing path are lengthened. When finding a geodesic path between two landmarks on the warped triangle surface, the path is naturally attracted by the feature lines and repulsed by the existing paths.

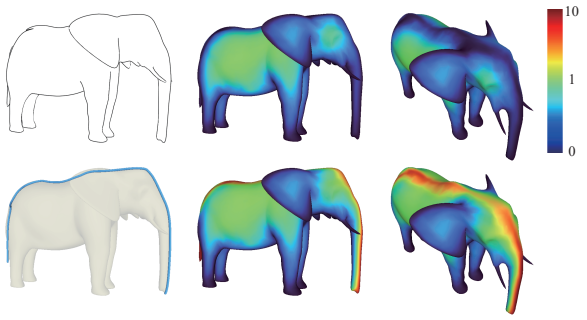

Fig. 4. The visualization of the weights for implicitly warping the surface model. Top: the visualization of the warping effect with the feature attraction weights. Bottom: the visualization of the warping effect with both the feature attraction weights and the path repulsion weights. Left: the feature lines (top) and an existing path (bottom) for computing the weights. Middle and right: the visualization of the warping effects in two views.

Feature attraction weights. To encourage feature coverage, we introduce feature attraction weights to warp the surface so that the triangle edges near the features are shortened and therefore the optimal path is attracted to the features. The weights are computed based on the 3D feature lines produced by the line drawing rendering techniques [DeCarlo et al. 2003; Haralick 1983; Judd et al. 2007]. After extracting these 3D feature lines on the surface of the model, we regard them as the sources to build a distance field on the surface using the heat method [Crane et al. 2017]. To define the feature attraction weights with this distance field, we first introduce the following modified logistic function:

$$L(x) = \frac{a}{1 + \exp\left(-bx/\bar{l}\right)} + (1 - a), \tag{1}$$

where $\bar{l}$ is the average length of the surface edges. a is the range parameter, which controls the range of this function, i.e., $L(x) \in [1 - a/2, 1)$ when $x \in [0, +\infty)$. b is the steepness parameter, which controls the rate of change of $L(x)$. Then the feature attraction weight w_i for each vertex v_i of the surface mesh with the distance value x_i is defined as $w_i = L(x_i)$. We define the weight function in the form of a modified logistic function due to its s-shaped characteristic: the function value changes fast within a certain range of x and stabilizes to 1 out of this range. This characteristic naturally constrains the warping effect within the neighboring region of the

[4]https://www.incredibleart.org/lessons/high/kevan-wiresculpture.htm
[5]http://marshallfredericks.org/wp-content/uploads/2014/01/Wire-Sculpture.pdf

feature lines. In our experiments, the input triangle meshes typically have around 8,000 vertices and we set the range parameter $a = 1.8$ and the steepness parameter $b = 0.5$ (i.e., $w_i \in [0.1, 1)$, and the effect of these weights is mainly within the 5-ring neighbors of the features lines) in our implementation. Then the adjusted edge length l'_{ij} in the graph is computed as $l'_{ij} = \frac{w_i + w_j}{2} l_{ij}$. Figure 4 (top) visualizes this warping effect with the feature attraction weights on a surface model. The edges around the feature lines are more shortened compared with those in the non-feature areas. The effects of parameters a and b for computing the feature attraction weights can be observed in Figure 6.

Path repulsion weights. The feature attraction weights guarantee that the optimal path captures the features of the input model. However, the optimal path may go through the same features multiple times and fail to reach a satisfactory level of total feature coverage. To address this, we introduce path repulsion weights to warp the surface so that the triangle edges near the existing path are lengthened. In this way, different parts of the optimal path will spread out, thus reducing the redundancy of the path and increasing feature coverage (see Figure 8 for the effect of the weights) . During the optimal path computation, each time when it reaches a landmark site, the path repulsion weights are updated. Specifically, we first build a distance field by setting the existing path as sources. Then the path repulsion weight w_i^* for each vertex v_i with distance value x_i^* is defined by $w_i^* = L(x_i^*)$. We also define this weight function in the form of the modified logistic function under the same consideration as in defining the feature attraction weights. Here we set the range parameter $a^* = 1.98$ and the steepness parameter $b^* = 0.5$ (i.e., $w_i^* \in [0.01, 1)$) in our implementation. This weight is used for lengthening the edges. The adjusted edge length l'_{ij} is computed as $l'_{ij} = \frac{w_i + w_j}{w_i^* + w_j^*} l_{ij}$ (see Figure 4 (bottom) for the warping effect visualization with respect to an existing path (in blue)). Here the lower bound of w^* is smaller than w, i.e, $\inf\{w\} = 10 \inf\{w^*\}$, ensuring that the expanded path will be repelled by existing paths at the feature region. The effect of parameter a^* and b^* can be observed in Figure 8.

Note that, the default parameter values were used for generating all the wire shapes in the paper, except for those in Section 4.2. a and a^* determine the ranges of the feature attraction weights and the path repulsion weights, respectively. Their values do not depend on input models. b and b^* determine the effective area of the weights. The appropriate values of these two parameters are affected by the triangle resolutions of the input models. In this paper, the input models for all experiments are with around 8,000 vertices. We set b and b^* with a default value of 0.5 to restrict the effective area within the 5-ring neighbors of feature lines or existing paths.

Multi-path expansion for near-optimal paths. The classic TSP is already NP-hard. After introducing the feature attraction and path repulsion weights, the distances between pairs of landmark sites will dynamically change when the path reaches a new site, making this DTSP more difficult to solve. On the other hand, to generate an adequate number of wire shapes to satisfy different preferences, we need not only one optimal path, but a set of near-optimal paths. Therefore, we adopt a greedy strategy with simultaneous multiple paths expansion (see Figure 5). Specifically, we keep track of a set of

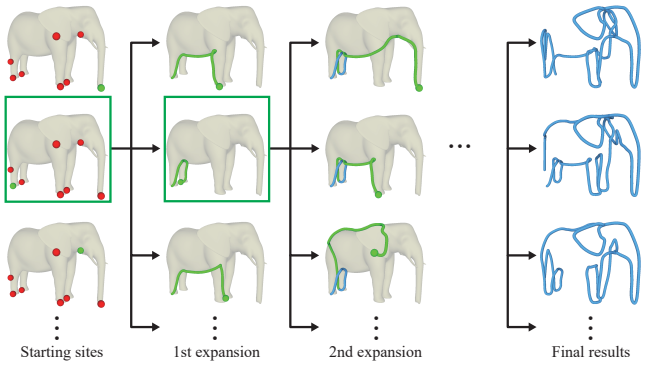

Fig. 5. The multi-path expansion procedure for near-optimal paths. For clear illustration, only one branch is shown in each path expansion. The green rectangles highlight the branches for path expansion.

paths $\mathcal{P}$ and let them expand to the unvisited sites simultaneously. In the following, we consider a path $p \in \mathcal{P}$ is a set of consecutive edges on the surface, i.e., $p = \{e_i\}$. Initially, every site is considered as a starting site of a path p and is included in p as a special edge element. The paths yielded by all sites are included in $\mathcal{P}$. In each step, a path $p \in \mathcal{P}$ can potentially expand to K nearest unvisited sites and yield K expanded paths $\{\Delta p_k\}$. Δp_k is the geodesic path between the current and a next site of path p. In our implementation, we set $K = 5$. To avoid exponential increase of the tracked paths, only a subset of the expanded paths will be retained to yield a new set of tracked paths $\mathcal{P}'$. The selection is based on their feature coverage score, defined by:

$$c = \frac{\sum_{e \in \Delta p} l'(e)}{\sum_{e \in \Delta p} l(e)}, \tag{2}$$

where $l'(e)$ is the edge length on the warped surface, $l(e)$ is the edge length on the original surface. A small value of c_k indicates that Δp_k passes the regions with uncovered features, and therefore Δp_k is likely to be retained. We rank all the expanded paths according to their c_k and keep the top M to yield the new tracked path set $\mathcal{P}'$. We set $M = 10$ in our implementation. This path expansion continues until all sites are visited. In the final round of expansion, all the generated paths are complete and they can be ranked in a global view.

The ranking of all generated paths is based on a global feature conformity error E_c and a path redundancy error E_r as defined by the following equation:

$$E(p) = E_c(p) + \lambda E_r(p), \tag{3}$$

where $\lambda = 0.1$ in our implementation. Figure 9 shows the results when λ has different values. The global feature conformity error E_c measures the distribution similarity between the path and the feature lines of the model. We first compute two Gaussian fields $\mathbf{F}_p$ and $\mathbf{F}_f$ on the surface with a Gaussian density function $g(x, \mu, \sigma)$, where x is the geodesic distance from a surface point to the path or feature lines. $\mu = 0$, and $\sigma = 4\bar{l}$ in our implementation. Then the global feature conformity error E_c is defined as:

$$E_c(p) = \left\| \mathbf{F}_p - \mathbf{F}_f \right\|. \tag{4}$$

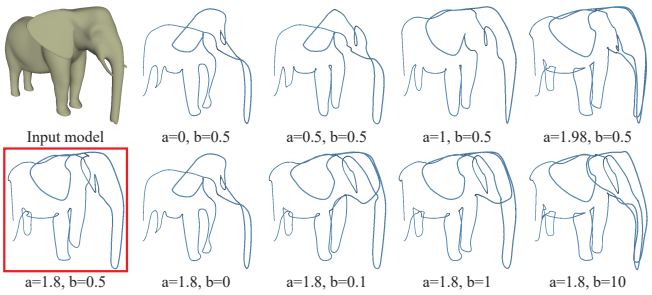

Fig. 6. The wire shapes generated with different combinations of a and b. The red rectangle highlights the wire shape generated with the default parameters. When $a = 0$, i.e., the first wire shape in the top row, the generated path approximates to the geodesic path on the unwarped surface.

We simply adopt the L_2 norm to measure the difference between these two fields, instead of more sophisticated measurement like the earth mover's distance [Solomon et al. 2014], due to the high efficiency of L_2 and comparable performance. The path redundancy error E_r measures how repetitive the path is when expressing the features of the model. We first compute a set of Gaussian fields $\mathbf{F}_{\Delta p}$, where $\{\Delta p\}$ are the expanded paths during the expansion of path p, i.e, $\Delta p \subset p$. Then E_r is computed as:

$$E_r(p) = \left\| \mathbf{F}_p - \sum_{\Delta p \subset p} \mathbf{F}_{\Delta p} \right\|. \tag{5}$$

Then the top M paths with the smallest $E(p)$ will be returned as the candidate wires.

Spline curve fitting. The returned paths are composed of triangle edges and may have jagged shapes. We adopt C^2 interpolating splines [Yuksel 2020] with a Bézier interpolation function to obtain a more visually pleasing wire. This method guarantees the fitted curve is C^2 continuous. During curve fitting, we first select the vertices on the path with high curvature values (larger than $1/\bar{l}$ in our implementation) as control points. After selecting these control points, we then evenly select vertices (with interval value $5\bar{l}$ in our implementation) on the path as the remaining control points.

Tree-like wire generation. Our algorithm generates wire shapes consisting of one single wire. It can be easily extended to generate wire shapes with a tree-like structure. This is achieved by simply allowing the path expansion to occur in the previously visited landmark sites, instead of the current one. We can also control the number of this middle part path expansion in the algorithm to generate wire shapes in desired structures. For example, if we allow the middle part expansion to occur once, the resulting wire shape will have two branches. Please see Figure 7 for illustration.

Fig. 7. Two wire shapes with a tree-like structure. The red rectangles highlight the middle part expansions.

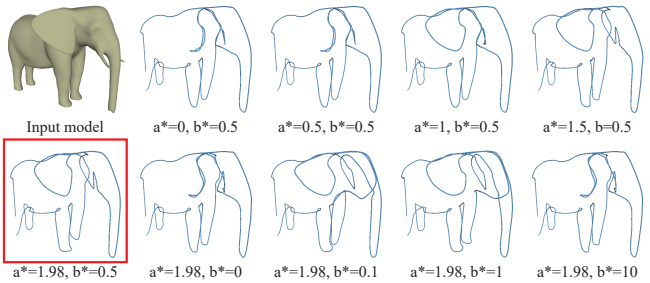

Fig. 8. The wire shapes generated with different combinations of a^* and b^*. The red rectangle highlights the wire shape generated with the default parameters.

4.2 Effects of Parameters

In general, the parameters in the aforementioned algorithm control the feature coverage and structure conciseness of the generated wire shapes. In this section, we investigate and discuss the effects of parameters under non-default values.

Effects of feature attraction weights. These weights control how the path covers the feature lines on the model. According to Equation (1), $w_i \in [1 - a/2, 1)$. To ensure that w_i is positive, the reasonable range of a is $[0, 2)$. A small a will reduce the edge shortening effect at the feature region. When $a = 0$, the generated path approximates to the geodesic path (slightly different due to the influence of the path repulsion weights) on the unwarped surface. b controls the effective area of the edge shortening effect, and its range is $[0, +\infty)$. A small b (close to 0) will span the effective area to the whole model, and a large b (larger than 10) will shrink the effective area around the feature lines. Both cases will invalidate the edge shortening effect. Figure 6 illustrates the impact of a and b under different values. Note that a^* and b^* remain the default values in these results. For clear comparison, the order for visiting the landmark sites is fixed in this experiment.

Effects of path repulsion weights. These weights control the redundancy of the generated paths. A small a^* will reduce the edge lengthening effect at the path region. The effect of b^* is similar to that of b. Figure 8 illustrates the effects of a^* and b^* under different values. Note that a and b remain the default values in these results. For clear comparison, the order for visiting the landmark sites is fixed in this experiment.

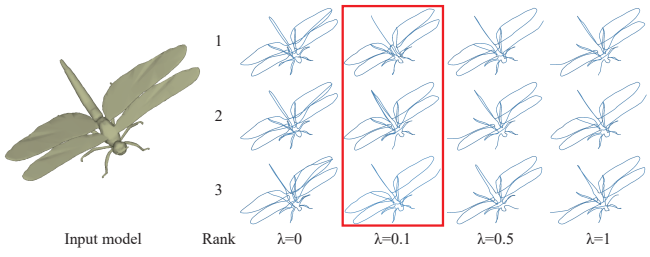

Fig. 9. The top-ranked wire shapes with different values of λ. The red rectangle highlights the wire shapes generated with the default parameter.

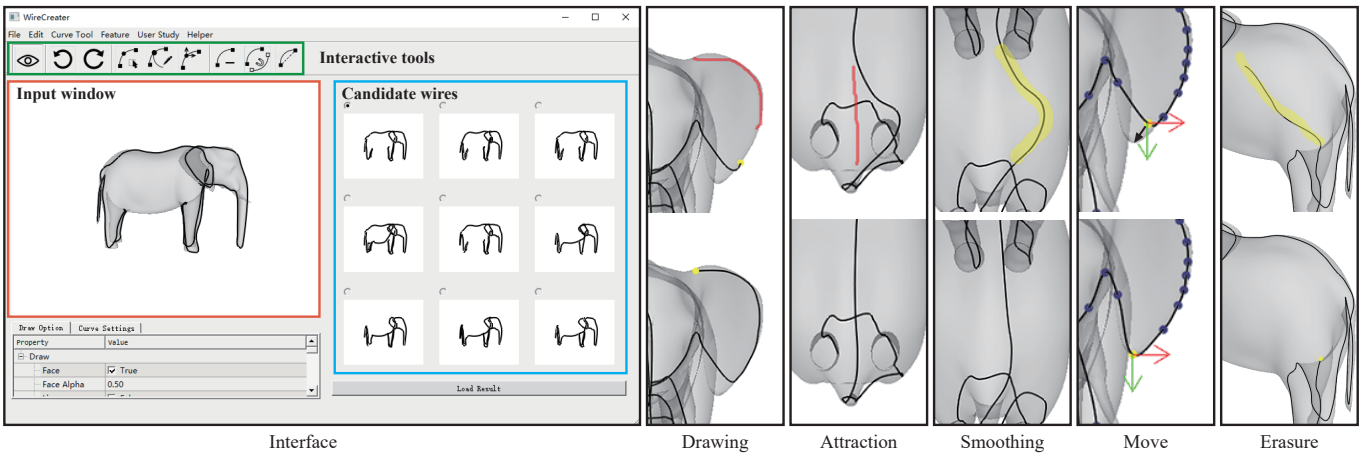

Fig. 10. Left: the main interface of our framework. Right: the interactive tools provided by our framework. Please refer to the accompanying video for interactive sessions.

Effects of E_c and E_r. The global feature conformity error E_c measures the distribution similarity between the path and the feature lines of the model. The error will be large if a large number of features are uncovered, or a large portion of the part is at the non-feature region. The path redundancy error E_r measures how repetitive the path is when expressing the features of the model. The error will be large if the path has a lot of self-overlap. λ controls the relative contribution of these two errors. Figure 9 illustrates the effect of λ.

5 USER INTERFACE

Figure 10 shows the prototype of our framework. The left window displays a 3D surface model and a selected wire shape. The 3D surface model is rendered in a semi-transparent mode to avoid visual obstruction of the wire shape. The interactive editing is also performed in this window. The user can rotate the camera with an Arcball-like interface. The right part contains several small candidate windows, which display the automatically generated wire shapes in the same view as the left window. To select a desired wire shape, the user may click the corresponding candidate window and click the confirmation button. Then the selected wire shape will be displayed in the left window for subsequent edits. We provide the following wire addition, manipulation, and erasure tools to the user (see Figure 10).

Addition. We provide a free drawing tool for wire addition. The user may draw a free-form curve on the surface of the model with this tool. After finishing the drawing, this curve will automatically connect to a pre-selected control point of the existing wire. The connection is realized by a geodesic path and the endpoint with a shorter path is selected for connection. This tool also enables a path expansion effect by simply drawing a point.

Manipulation. We provide three tools for wire manipulation. The first manipulation tool is an attraction tool. The user may draw a curve close to the existing wire. The nearby segment of the wire will be attracted to the curve without breaking its structure. The second tool is a smoothing tool. The user may paint on a segment of the existing wire with a smoothing brush, and the painted segment will then be smoothed. The third tool is a direct manipulation tool. The user may directly move the selected control points for wire manipulation.

Erasure. We provide an erasure tool for wire erasure. The user may paint on a segment of the existing wire with an erasure brush to remove this segment. To avoid destructing the wire structure, we allow the wire to be erased from the endpoints only.

Curve specification before wire generation. Typically, the above interaction tools are used after getting an automatically generated wire. Alternatively, the user may also specify curves on the surface of the model with these tools before the wire generation. The specified curves will be treated as special landmark sites in the path expansion procedure, i.e., each curve should be passed via the endpoints. In this way, the user may control the local details of the wire without concerning the structure of an entire wire shape. Figure 11 shows the effect of specifying curves before wire generation.

Fabrication plan for manual realization. The wire shape produced by our framework can be processed with the method in [Lira et al. 2018] to obtain segments fabricable by a wire bending machine. However, this procedure usually breaks the integral structure of our single-line wire shape. On the other hand, manually bending the wires to reproduce a wire shape is not easy especially for novice users due to the lack of spatial and structural cues [Yue et al. 2017].

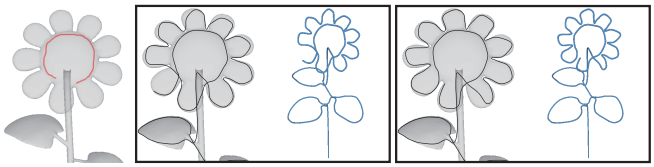

Fig. 11. The user may specify curves before the wire generation. Left: the specified curve. Middle and right: two wire shapes with this curve specification.

To address this problem, we generate a fabrication plan for manual realization of wire shape. Given a digital wire shape, we first decompose it into several segments, each of which is near-planar. This is achieved by a simple hierarchical clustering of the points on the wire. Then we project each segment on its associated plane, while keeping the segments connected. This results in a new wire shape with piecewise planar structure. Each planar part of this new wire shape can be printed to guide manual realization. The user can bend the segments consecutively with one single wire, and fine-tune the relative position of neighbored segments to obtain the fabricated wire shape. Figure 16 shows three fabricated wire shapes. A fabrication plan can be found in the supplemental materials.

6 EVALUATION

We have tested our framework extensively with various 3D models in different categories. Figure 1, 15, and 18 show the representative results generated by our framework. To evaluate our framework thoroughly, we have conducted the following experiments. First, we compare our wire generation algorithm with a baseline method, i.e., creating wire shapes by connecting the curves generated by the line drawing rendering techniques. We adopt the simulated annealing algorithm [Iarussi et al. 2015; Kirkpatrick et al. 1983; Lira et al. 2018] to solve the curve connection problem. Second, we conducted a user study to evaluate the effectiveness of our framework. The study was composed of two parts. In Study I, a number of participants were invited to create wire shapes with our framework. It was designed to investigate the usability of our framework, with and without the automatic wire generation module. In Study II, the wire shapes created with and without the automatic wire generation module were evaluated by another group of participants.

6.1 Baseline Comparison

To the best of our knowledge, there is no similar computational framework for abstract wire shape design from a 3D model. Existing works [De Goes et al. 2011; Gori et al. 2017; Mehra et al. 2009] are able to produce abstract curve networks from 3D models, but their results have distinct abstraction styles compared with our single-wire shapes. A straightforward attempt to generate single-wire shapes is to connect the 3D curves generated by the line drawing rendering techniques. However, it is unclear how to form a reasonable connection between the scattered curves. Lira et al. [2018] and Iarussi et al. [2015] provided solutions for finding Eulerian paths in 2D and 3D curve networks, respectively. We extend their algorithm to solve the curve connection problem. Specifically, we first extract 3D curves by using ridges and valleys, suggestive contours, and apparent ridges. The long curves are retained and short curves are discarded as outliers. To achieve a similar setting in [Lira et al. 2018] and [Iarussi et al. 2015], we consider the curves as edges and their endpoints as junctions. In addition, we redefine the join operator such that it can be applied to any two junctions and introduce a connection error by the geodesic distance between these junctions. We then apply the simulated annealing algorithm [Kirkpatrick et al. 1983] to obtain a final graph. In this graph, if two curves share a junction, the corresponding endpoints are connected by a geodesic path on the surface model.

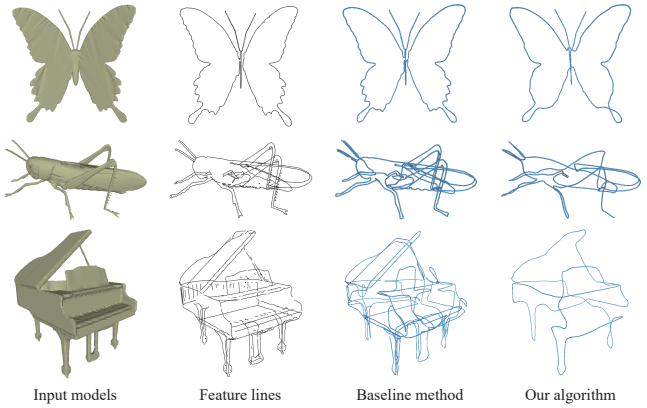

Fig. 12. The comparison between our algorithm and the baseline method. The baseline method is able to generate an acceptable result for the Butterfly, but fails to produce satisfactory wire shapes for the Locust and the Piano. Our algorithm is able to generate wire shapes with concise structures for these three models. Note that the results generated by our algorithm were selected from the candidates and no wire editing was performed after the selection.

Figure 12 shows several comparison results. Note that the results generated by our algorithm were selected from the candidates and no wire editing was performed after the selection. For the simple case, e.g., the Butterfly, this baseline method is able to generate an acceptable wire shape, though the shape of the result is relatively fixed. For more complex cases, e.g., the Locust and the Piano, it is notable that the wire shapes generated by the baseline method exhibit over-complicated structures. This is due to the following reasons. First, the shapes of the input curves are fixed. Connecting them might introduce some redundant paths. In addition, the connection paths may also gather on the surface, leading to visual clutter. Second, although the short curves have been filtered out, the input curves may still be messy. Keeping all curves in the final wire shape results in a complicated structure. Our algorithm takes the feature lines as soft constraints, thus avoiding introducing redundant paths. Our approach adopts feature coverage to satisfy the feature constraints, thus preventing messy curves. For both the simple and complex cases, our algorithm is able to generate wire shapes with concise structures.

6.2 Study I: Usability Study

This study aims to investigate the usability of our framework. We did not include other techniques in this study since there was no similar computational framework for creating such abstract wire shapes. To better examine our framework, we tested it in two different configurations, i.e., with and without the automatic wire generation module. In the following, we denote these two configurations as **WireRec** (short for wire recommendation) and **NonWireRec**, respectively. In both configurations, the participants were allowed to use the interactive tools provided by our framework for wire shape creation.

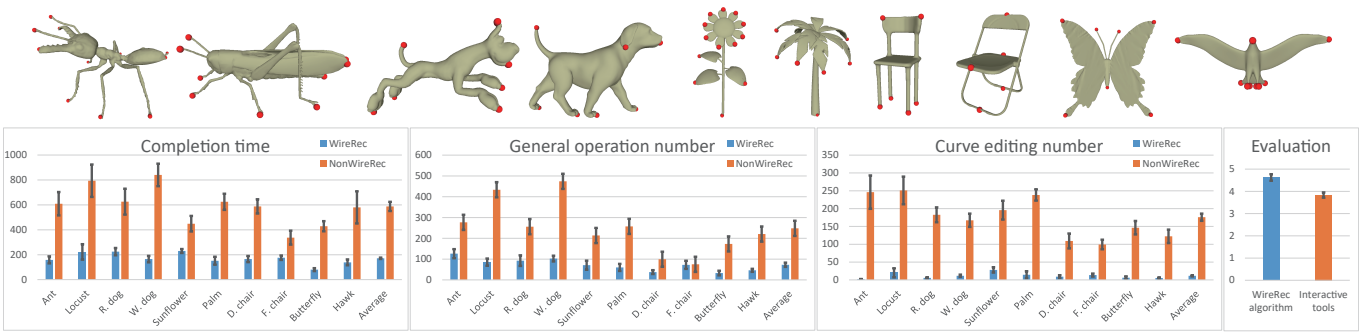

Fig. 13. Top: 10 models with different complexities were tested in Study I. These models were differentiated in 5 pairs according to the structural similarities or categories. Red spheres represent the automatically detected landmark sites. Bottom: average completion time, average general operation number, and average curve editing number for each model in the two configurations; the user evaluation of the automatic wire generation algorithm and the interactive tools. The unpaired t-test confirmed that there exist significant differences between **WireRec** and **NonWireRec** in the completion time, the general operation number, and the curve editing number for almost all models. The error bar represents the standard error of the mean.

Participants. We recruited 12 university students to participate in this study: 10 males and 2 females, aged 21 to 26. 6 of them were familiar with 3D modeling tools, such as Autodesk 3ds Max. 4 of them had good drawing skills. One participant had tried the creation or design of wire shape. All the participants got paid after this study to compensate their time.

Apparatus. The study was conducted on a PC with Intel Xeon 2.1GHz processor and 32GB RAM. A Wacom Cintiq 22HDT interactive display with both touch and pen input was connected to the PC as the input and output device. A mouse was also provided. The participants were recommended with pen input for easier curve drawing. They were allowed to switch between mouse and pen inputs freely. The touch input was disabled to avoid misoperation.

Task. The task was free creation of wire shapes with given 3D models as guidance. We provided 10 models with different complexities (see Figure 13). The models were differentiated in 5 pairs, i.e, Ant and Locust, Running dog and Walking dog, Sun flower and Palm, Dining chair and Folding chair, Butterfly and Hawk. The pairing was according to the similarities or categories of the models. Each participant was asked to create one wire shape for each model, in either **WireRec** or **NonWireRec**. The configurations for the models in a pair were different. Therefore each participant created 10 wire shapes, 5 in **WireRec** configuration and 5 in **NonWireRec** configuration. The overall configuration of models was counterbalanced across all participants, so that the wire shapes created in the two configurations were equal for each model. In total there were 5 (model pairs) × 2 (configurations) × 12 (participants) = 120 trials, each of which produced one wire shape.

This task design was based on the following considerations. First, the wire shape creation was an open-ended task and the planning was critical. It was not wise to let a participant create the wire shapes from the same model twice, since the second creation would be greatly affected by the first one. Second, the pairing of models guaranteed that the wire shapes created by a participant in two configurations had similar complexities, and thus reduced the influence of the configuration setting on a participant.

Before the study, the participants were introduced to our framework, including the automatic wire generation module and the interactive tools. After they understood all the functions of our framework, they could try to create a wire shape from a simple Dolphin model, in both **WireRec** and **NonWireRec** configurations. This tutorial session lasted approximately 15 minutes. During the study, the participants were asked to create the wire shapes consecutively and a short break was also allowed. The orders of the configurations and the models were both counterbalanced across all participants. The whole study lasted around 1.5 hours on average for each participant.

Performance measures. During the study, the following information was recorded: the completion time of individual trials, the number of curve editing operations (curve addition, manipulation, and erasure), and the number of general operations (camera manipulation, redo, and undo). The created wire shapes were stored for further evaluation. We also collected the feedback from the participants, including rating the interactive tools and automatic wire generation module.

Results. Figure 13 plots the statistics of the core information recorded in this study. It was not surprising that in **WireRec** configuration, the participants spent less completion time (171.4s vs. 587.8s on average), performed fewer general operations (72.8 vs. 248.0 on average) and fewer curve editing operations (11.7 vs. 175.9 on average) to create a wire shape. The unpaired t-test also confirmed that there existed significant differences between **WireRec** and **NonWireRec** configurations in the completion time, the number of general operations, and the number of curve editing operations for almost all models. Please see the supplemental materials for the p-values of the unpaired t-tests.

In general, the completion time and the operation number were proportional to the complexity of the model in **NonWireRec** configuration. For example, the participants spent less time and performed fewer operations when creating wire shapes from the Sunflower, Butterfly, and Folding chair models. For Sunflower and Butterfly, the curves could be added in a small number of views and the planning of the curves was simple. For the Folding chair, the curves could be

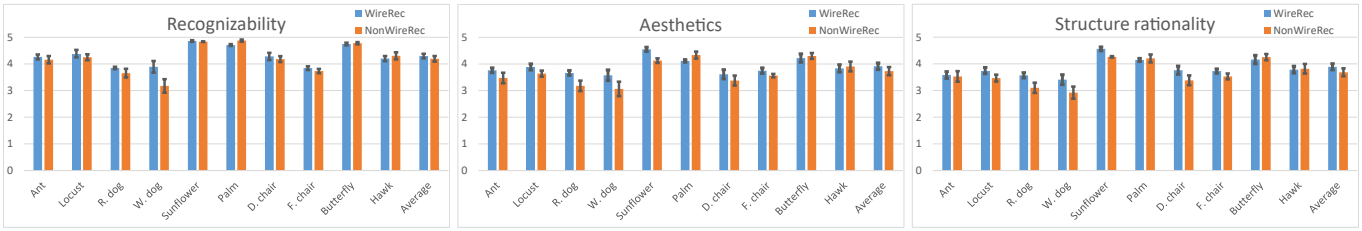

Fig. 14. Statistics of Study II. The wire shapes created in **WireRec** generally were comparable to or better than those created in **NonWireRec** in recognizability, aesthetics, and structure rationality. The error bar represents the standard error of the mean.

added effectively via the path expansion effect of our wire addition tool. Some of these facts were also true for the Ant and the Running dog, which had similar structures with the Locust and the Walking dog, respectively, but needed less completion time and fewer operations.

The completion time and the operation number for creating a wire shape in **WireRec** configuration were relatively stable across different models. In this configuration, the participants followed the typical workflow of our framework and performed moderate interventions (Figure 13). The participants were in general satisfied with the default landmark sites and did not edit them, except for the Folding chair. In addition, they needed to perform only a few curve editing operations after selecting a generated wire shape. In 16 of 60 trials that were in **WireRec** configuration, the participants accepted the wire shapes generated by the algorithm without further editing. However, it was notable that the completion time and operation number for the Sunflower had the largest values, though this model was not complex at all. It is possibly because the participants tended to edit the curves at the petal region so that they could perfectly match the contour, since they reflected the most prominent features of the model. In contrast, for the other models, the local shapes of the curves were less important compared with their overall structures. This observation also reminded us to include a curve correction function [Su et al. 2014] in the future.

We asked the participants to rate the interactive tools and the automatic wire generation module with a discrete scale from 1 (negative) to 5 (positive) after they finished the task. The rating was according to the ease of use for the interactive tools, and the satisfaction of the generated wire shapes for the automatic wire generation module. The rating scores (see Figure 13) showed that the majority of the participants were quite satisfied with our algorithm (4.63 on average), though the interactive tools could be improved (3.83 on average). All the participants commented that "*the algorithm is indispensable since it saved a lot of time*". A participant with good drawing skills commented that "*the returned wires are in general satisfactory, but I still need to fine-tune the curves to achieve a more perfect result*". Several participants with 3D modeling experiences commented that "*I feel the interactive tools in general are intuitive to use*"; "*when I drew the ears of the Walking dog, the curve was projected on the head if it slightly exceeded the contour. I expect to improve the system so that it can determine the correct surface for projection*".

6.3 Study II: Visual Evaluation

This study aimed to visually evaluate the wire shapes created in **WireRec** and **NonWireRec** configurations in Study I and find out whether our framework was comparable with the manual design.

Participants. We invited another group of participants to help this study. In total 40 university students participated: 36 males and 4 females, aged 22 to 34. 11 of them reported that they had art design background.

Task. The task was to rate the wire shapes after observing them. The participants were asked to rate wire shapes in three aspects, i.e., recognizability, aesthetics, and wire structure rationality. Here the wire structure rationality was based on the conciseness and feature coverage. The score was from 1 to 5, where 1 means negative, 3 neural, and 5 positive. The wire shapes to be rated were displayed via a 3D viewer. The participants could rotate the camera to find the best view to observe the wire shapes. We found that it was challenging for a participant to rate the wire shapes in a stable standard if only one wire shape was displayed at a time. Displaying multiple shapes to be rated would help the participants to give more consistent scores. Nevertheless, displaying too many shapes might introduce visual clutter and confuse the participants. We thus decided to display 6 wire shapes at a time to achieve a balance. These 6 wire shapes were created from the same model, with half created in **WireRec** and the other half in **NonWireRec** configuration. Each participant was asked to rate 6 wire shapes per model and 60 (instead of 120 to avoid fatigue) wire shapes in total. The combination of the wire shapes for display were counterbalanced so that each wire shape was rated by 20 participants.

Results. Figure 14 plots the statistics of the scores. Each score is the average score of 6 wire shapes created in **WireRec** or **NonWireRec**. For most of models, the wire shapes created in **WireRec** received higher scores than those created in **NonWireRec**, except for the Palm, the Butterfly, and the Hawk. The unpaired t-test on the wire shapes created in two configurations confirmed that there existed significant difference in these cases: the aesthetics score ($p = 0.008$) and the structure rationality score ($p = 0.005$) for the Sunflower. This indicated that our algorithm was able to generate wire shapes comparable to or even better than the manual designs of the users.

Most recognizability scores of the models in both configurations were larger than 4, except for the Running dog, the Walking dog, and the Folding chair. This fact confirmed that the abstract wire shapes were able to express desired content in most cases. For the

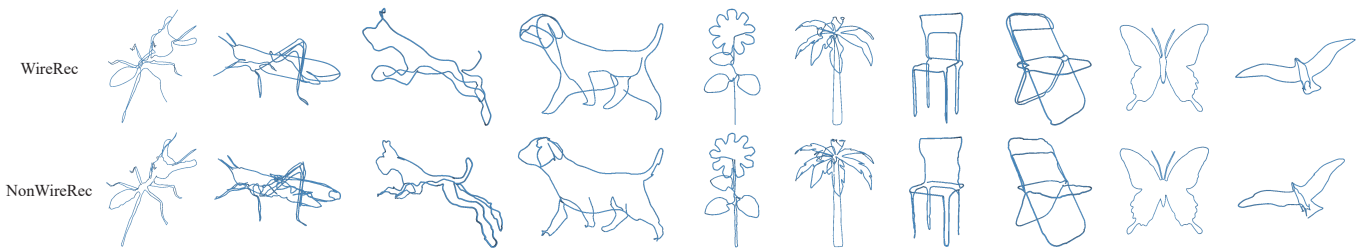

Fig. 15. The selected wire shapes from the creations in Study I. These wire shapes received the highest scores in two configurations for each model in Study II. Top: the wire shapes created in **WireRec**. Bottom: the wire shapes created in **NonWireRec**. The participants might perform wire editing after selecting the most desired wire shape from the candidates in **WireRec**. More wire shapes created by the participants can be found in the supplemental materials.

Running dog and Walking dog, although the corresponding wire shapes might be recognized as animals, it indeed needed some efforts to determine their exact contents. For the Folding chair, the wire shapes were not able to capture the distinctive features of this model, e.g., the seat surface, making observers difficult to associate the wire shapes with the folding chairs. Almost all recognizability scores of the wire shapes created in **WireRec** and **NonWireRec** were undifferentiated, except for the Walking dog. We found that the wire shapes of this model created in **NonWireRec** were in relatively low qualities, therefore damaging their recognzabilities. Please see the 3D wire shapes in the supplemental materials for more details.

The wire shapes created in **WireRec** generally were more visually pleasing (the scores were larger than 3.5 for all models). This was mainly because these wire shapes generally had more concise structures and smoother curves. Some participants in Study I tended to draw all details of the wire shapes in **NonWireRec**. The curves drawn by the participants with inadequate drawing skills might be jagged even after the spline fitting. For the Palm, the Butterfly, and the Hawk, the wire shapes in **NonWireRec** received higher scores. This was due to the following reasons. First, these models had simple structures and the participants could pay more attention on drawing the curves. Second, more details could be captured by the drawings, e.g., the leaves of the Palm, the wings of the Butterfly, and the claws of the Hawk.

The structure of the wire shapes created in **WireRec** were more reasonable for the majority of the models (the scores were larger than 3.5 for all models except for the Walking dog). For the Palm, the Butterfly, and The Hawk, their wire shapes created in **NonWireRec** configuration were considered to have better structures due to the following reasons. First, as discussed before, these wire shapes had better feature coverage since the drawings captured more details of the models, e.g., the leaves of the Palm. Second, some wire shapes indeed had more concise structures, e.g., several participants in Study I only drew the outline of the Butterfly, resulting in a recognizable wire shape with simple structure. However, our algorithm was actually able to generate a similar wire shape, but some participants in Study I just ignored it and selected others (please see the wire shapes created in Study I in the supplemental materials). The wire shapes of the Walking dog created in **WireRec** received 3.41, mainly due to the low feature coverage at the head part of the Walking dog.

It was also notable that the professional skills and the aesthetic perceptions of the participants greatly affected the quality of the created wire shapes in both configurations (See the supplemental materials for more detail). Figure 15 shows 2 wire shapes with the highest scores in two configurations for each model. These results again suggested that our algorithm was able to produce comparable or even better wire shapes.

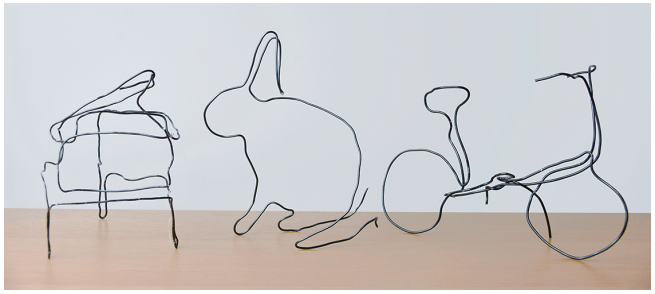

Fig. 16. The metal wire shapes fabricated with our fabrication plans.

6.4 Fabrication

We invited two volunteers to fabricate given digital wire shapes with our fabrication plans (see Figure 16). Neither of them had tried the fabrication of wire shapes. The fabrication process lasted approximately 15 to 45 minutes, depending on the complexities of the wire shapes. They confirmed the effectiveness of the fabrication plan when fabricating simple wire shapes, e.g., the Bunny and the Bicycle. The printed plan provided cues for bending the wires on a plane. However, they also commented that it was quite challenging to fabricate a complex wire shape, e.g., the Piano in Figure 1, 12, since it had a large number of planar segments, and it was confusing to fine-tune the relative positions of the connected segments. This fact reminded us that it would be interesting to design a folding algorithm so that the folded wire shape can be fabricated by a wire bending machine, and the desired wire shape could be obtained by deforming the folded wire shape. Please see the fabrication plan of the Bunny (Figure 1) in the supplemental materials.

6.5 Limitations

Our framework does not always produce wire shapes with high recognizability (e.g., the Running dog, the Walking dog, and the Folding chair in Study I). Our multi-path expansion approach tends to return wires with smooth curves, therefore discarding the local

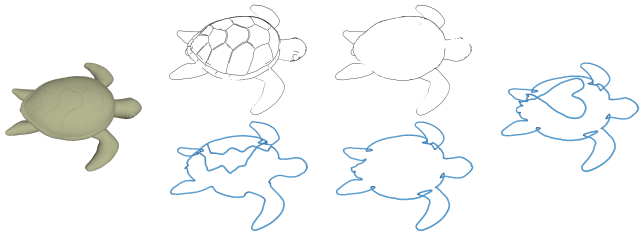

Fig. 17. Our algorithm is not effective for a model with rich surface details (second column). Smoothing the model can alleviate this problem, but the produced wire shape would fail to capture the original surface details (third column). Such details can only be specified via user interventions (fourth column).

details of a 3D model (e.g., the Palm in Figure 15). Our framework is not effective for capturing planar features, therefore losing some prominent characteristic of a model (e.g, the Folding chair in Figure 15). Similar to the wire artworks created by artists, the wire shapes created with our framework suffer from the limitation of narrow visual angle. Although it might be solvable by adding more feature lines and landmark sites, the produced wire shapes would contain too many curves, leading to visual clutter. Although we have shown that our framework works well for various 3D models, it cannot be applied to models with rich surface details, since these features may confuse the multi-path expansion approach (Figure 17). This problem could be alleviated by smoothing such models in a pre-processing step to remove the redundant features. However, the surface details of the models will be neglected in the produced wire shapes. To express such surface details, it is inevitable to add curves via user interventions (Figure 17).

7 CONCLUSION AND FUTURE WORK

We have presented *WireRoom*, a computational framework for the design of abstract 3D wire art to depict a given 3D model. Our algorithm automatically generates a set of 3D wire shapes from the 3D model with informative, visually pleasing, and concise structures. We formulate the wire generation problem as a DTSP with a dynamically changed distance measurement on the surface of the 3D model, and propose a multi-path expansion approach to obtain a set of near-optimal paths. We introduce a novel explorative computational design procedure, which avoids manual design of wire shape structures. We believe that our framework can greatly reduce the effort for abstract wire art design, and assist novice users in creating desired abstract wire shapes.

As future work, we plan to further improve our framework according to the feedbacks from the user study participants, e.g., including the curve correction function in the framework, improving the curve projection algorithm to achieve intelligent projection effect. It would be interesting to further improve the interactive tools by leveraging cognitive or perceptual information from users, making the framework adaptive to different individuals. Another future work is to design a wire folding algorithm to make wire shapes produced by our framework fabricable by wire bending machines. This will further increase the practical usage of the wire shapes with a single-wire structure.

ACKNOWLEDGMENTS

We thank the reviewers for their constructive comments, the user study participants for their time. We thank Ruth Jensen, Maria Szabo, and Bud Bullivant for allowing us to use the images of their wire artworks. This project was partially supported by NSFC (62072316), the Research Grants Council of HKSAR (CityU 11212119), the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, GD Talent Program (2019JC05X328), GD Science and Technology Program (2020A0505100064, 2015A030312015), DEGP Key Project (2018KZDXM058), National Engineering Laboratory for Big Data System Computing Technology, and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

REFERENCES

Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. 2018. Symbiosissketch: combining 2d & 3d sketching for designing detailed 3d objects in situ. In *CHI*. 1–15.

Oscar Kin-Chung Au, Youyi Zheng, Menglin Chen, Pengfei Xu, and Chiew-Lan Tai. 2011. Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization and Computer Graphics* 18, 7 (2011), 1125–1134.

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *UIST*. 151–160.

Sukanya Bhattacharjee and Parag Chaudhuri. 2020. A survey on sketch based content creation: from the desktop to virtual and augmented reality. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 757–780.

Weikai Chen, Xiaolong Zhang, Shiqing Xin, Yang Xia, Sylvain Lefebvre, and Wenping Wang. 2016. Synthesis of filigrees for digital fabrication. *ACM Trans. on Graphics* 35, 4 (2016), 1–13.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2017. The heat method for distance computation. *Commun. ACM* 60, 11 (2017), 90–99.

Fernando De Goes, Siome Goldenstein, Mathieu Desbrun, and Luiz Velho. 2011. Exoskeleton: curve network abstraction for 3d shapes. *Computers & Graphics* 35, 1 (2011), 112–121.

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive contours for conveying shape. *ACM Trans. on Graphics* 22, 3 (2003), 848–855.

Julie Dorsey, Songhua Xu, Gabe Smedresman, Holly Rushmeier, and Leonard McMillan. 2007. The mental canvas: a tool for conceptual architectural design and analysis. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*. 201–210.

Tobias Drey, Jan Gugenheimer, Julian Karlbauer, Maximilian Milo, and Enrico Rukzio. 2020. VRSketchIn: exploring the design space of pen and tablet interaction for 3d sketching in virtual reality. In *CHI*. 1–14.

Jérémie Dumas, An Lu, Sylvain Lefebvre, Jun Wu, and Christian Dick. 2015. By-example synthesis of structurally sound patterns. *ACM Trans. on Graphics* 34, 4 (2015), 1–12.

Ran Gal, Olga Sorkine, Niloy J Mitra, and Daniel Cohen-Or. 2009. iWIRES: an analyze-and-edit approach to shape manipulation. *ACM Trans. on Graphics* 28, 3 (2009), 1–10.

Akash Garg, Andrew O Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire mesh design. *ACM Transactions on Graphics* 33, 4 (2014).

Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. 2017. Flowrep: descriptive curve networks for free-form design shapes. *ACM Trans. on Graphics* 36, 4 (2017), 1–14.

Robert M Haralick. 1983. Ridges and valleys on digital images. *Computer vision, graphics, and image processing* 22, 1 (1983), 28–38.

Kai-Wen Hsiao, Jia-Bin Huang, and Hung-Kuo Chu. 2018. Multi-view wire art. *ACM Trans. on Graphics* 37, 6 (2018), 242–1.

Yijiang Huang, Juyong Zhang, Xin Hu, Guoxian Song, Zhongyuan Liu, Lei Yu, and Ligang Liu. 2016. Framefab: robotic fabrication of frame shapes. *ACM Trans. on Graphics* 35, 6 (2016), 1–11.

Emmanuel Iarussi, Wilmot Li, and Adrien Bousseau. 2015. WrapIt: computer-assisted crafting of wire wrapped jewelry. *ACM Trans. on Graphics* 34, 6 (2015), 1–8.

Tilke Judd, Frédo Durand, and Edward Adelson. 2007. Apparent ridges for line drawing. *ACM Trans. on Graphics* 26, 3 (2007), 19–es.

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.

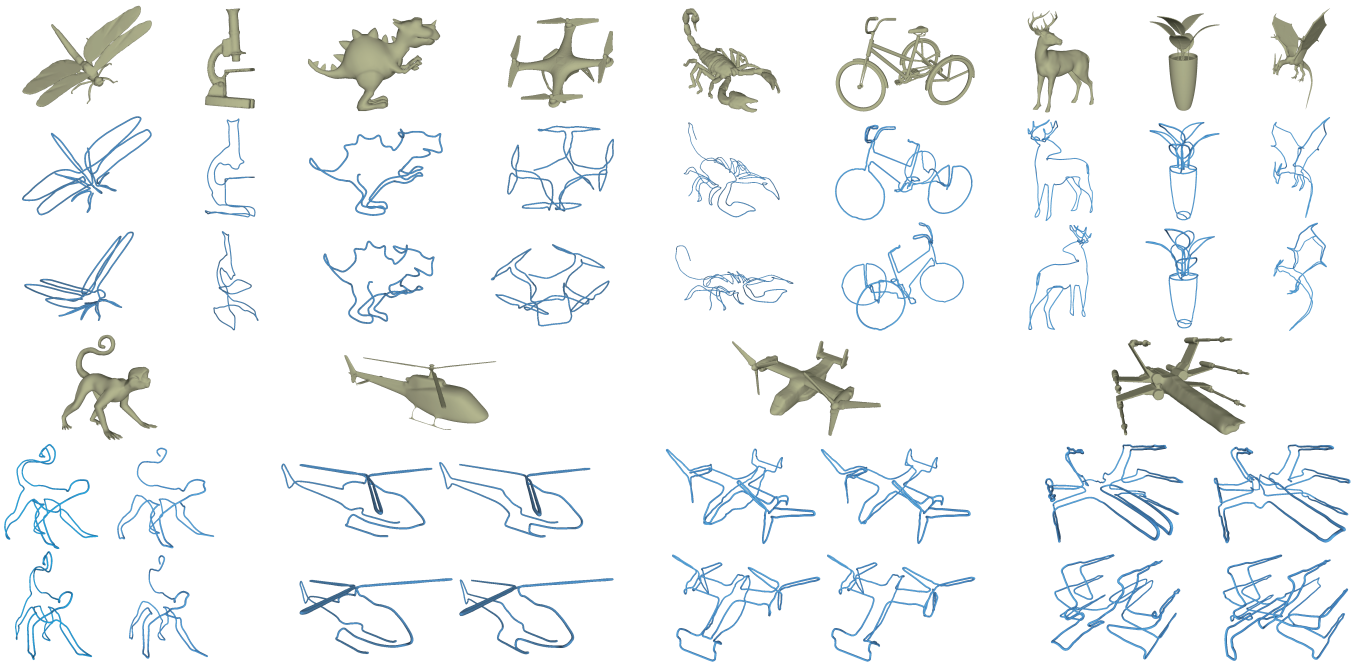

Fig. 18. Gallery of 3D wire shapes produced with our framework. Each wire shape is rendered in two views. Bottom: two wire shapes are produced for each model. All these wire shapes are with a single-wire structure.

Kin Chung Kwan and Hongbo Fu. 2019. Mobi3dsketch: 3D sketching in mobile AR. In *CHI*. 1–11.

Wallace Lira, Chi-Wing Fu, and Hao Zhang. 2018. Fabricable Eulerian wires for 3D shape abstraction. *ACM Trans. on Graphics* 37, 6 (2018), 1–13.

Lingjie Liu, Duygu Ceylan, Cheng Lin, Wenping Wang, and Niloy J Mitra. 2017a. Image-based reconstruction of wire art. *ACM Trans. on Graphics* 36, 4 (2017), 1–11.

Lingjie Liu, Nenglun Chen, Duygu Ceylan, Christian Theobalt, Wenping Wang, and Niloy J Mitra. 2018. CurveFusion: reconstructing thin structures from RGBD sequences. *ACM Trans. on Graphics* 37, 6 (2018), 1–12.

Min Liu, Yunbo Zhang, Jing Bai, Yuanzhi Cao, Jeffrey M Alperovich, and Karthik Ramani. 2017b. WireFab: mix-dimensional modeling and fabrication for 3D mesh models. In *CHI*. 965–976.

Zhao Ma, Alex Walzer, Christian Schumacher, Romana Rust, Fabio Gramazio, Matthias Kohler, and Moritz Bächer. 2020. Designing robotically-constructed metal frame structures. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 411–422.

Luigi Malomo, Jesús Pérez, Emmanuel Iarussi, Nico Pietroni, Eder Miguel, Paolo Cignoni, and Bernd Bickel. 2018. FlexMaps: computational design of flat flexible shells for shaping 3D objects. *ACM Trans. on Graphics* 37, 6 (2018), 1–14.

Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J Mitra. 2009. Abstraction of man-made shapes. *ACM Trans. on Graphics* 28, 5 (2009), 1–10.

Eder Miguel, Mathias Lepoutre, and Bernd Bickel. 2016. Computational design of stable planar-rod structures. *ACM Trans. on Graphics* 35, 4 (2016), 1–11.

Stefanie Mueller, Sangha Im, Serafima Gurevich, Alexander Teibrich, Lisa Pfisterer, François Guimbretière, and Patrick Baudisch. 2014. WirePrint: 3D printed previews for fast prototyping. In *UIST*. 273–280.

Abraham P Punnen. 2007. The traveling salesman problem: applications, formulations and variations. In *The traveling salesman problem and its variations*. Springer, 1–28.

Enrique Rosales, Jafet Rodriguez, and ALLA SHEFFER. 2019. SurfaceBrush: from virtual reality drawings to manifold surfaces. *ACM Trans. on Graphics* 38, 4 (2019), 1–15.

Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009. Analytic drawing of 3D scaffolds. *ACM Trans. on Graphics* 28, 5 (2009), 1–10.

Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. 2014. Earth mover's distances on discrete surfaces. *ACM Trans. on Graphics* 33, 4 (2014), 1–12.

Qingkun Su, Wing Ho Andy Li, Jue Wang, and Hongbo Fu. 2014. EZ-sketching: three-level optimization for error-tolerant image tracing. *ACM Trans. on Graphics* 33, 4 (2014), 1–9.

Ye Tao, Guanyun Wang, Caowei Zhang, Nannan Lu, Xiaolian Zhang, Cheng Yao, and Fangtian Ying. 2017. Weavemesh: a low-fidelity and low-cost prototyping approach for 3d models created by flexible assembly. In *CHI*. 509–518.

Josh Vekhter, Jiacheng Zhuo, Luisa F Gil Fandino, Qixing Huang, and Etienne Vouga. 2019. Weaving geodesic foliations. *ACM Trans. on Graphics* 38, 4 (2019), 1–22.

Peng Wang, Lingjie Liu, Nenglun Chen, Hung-Kuo Chu, Christian Theobalt, and Wenping Wang. 2020. Vid2Curve: simultaneous camera motion estimation and thin structure reconstruction from an RGB video. *ACM Trans. on Graphics* 39, 4 (2020), 132–1.

Yinan Wang, Xi Yang, Tsukasa Fukusato, and Takeo Igarashi. 2019. Computational design and fabrication of 3D wire bending art. In *SIGGRAPH Asia 2019 Posters*. 1–2.

Rundong Wu, Huaishu Peng, François Guimbretière, and Steve Marschner. 2016. Printing arbitrary meshes with a 5DOF wireframe printer. *ACM Trans. on Graphics* 35, 4 (2016), 1–9.

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2form: 3d curve networks from 2d sketches via selective regularization. *ACM Transactions on Graphics* 33, 4 (2014).

Pengfei Xu, Hongbo Fu, Youyi Zheng, Karan Singh, Hui Huang, and Chiew-Lan Tai. 2018. Model-guided 3D sketching. *IEEE Transactions on Visualization and Computer Graphics* 25, 10 (2018), 2927–2939.

Hui Ye, Kin Chung Kwan, and Hongbo Fu. 2021. 3D curve creation on and around physical objects with mobile AR. *IEEE Transactions on Visualization and Computer Graphics* (2021).

Ya-Ting Yue, Xiaolong Zhang, Yongliang Yang, Gang Ren, Yi-King Choi, and Wenping Wang. 2017. Wiredraw: 3d wire sculpturing guided with mixed reality. In *CHI*. 3693–3704.

Cem Yuksel. 2020. A class of C^2 interpolating splines. *ACM Trans. on Graphics* 39, 5 (2020), 1–14.

Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2016. Designing structurally-sound ornamental curve networks. *ACM Trans. on Graphics* 35, 4 (2016), 1–10.

Hao Zhang, Alla Sheffer, Daniel Cohen-Or, Quan Zhou, Oliver Van Kaick, and Andrea Tagliasacchi. 2008. Deformation-driven shape correspondence. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1431–1439.