Tune-It: Optimizing Wire Reconfiguration for Sculpture Manufacturing: Supplemental Material

QIBING WU*, Shandong University, China
ZHIHAO ZHANG*, Shandong University, China
XIN YAN, Shandong University, China
FANCHAO ZHONG, Shandong University, China
YUEZE ZHU, University College London, United Kingdom
XURONG LU, Shandong University, China
RUNZE XUE, Shandong University, China
RUI LI, Shandong University, China
CHANGHE TU, Shandong University, China
HAISEN ZHAO†, Shandong University, China

Wire sculptures are important in both industrial applications and daily life. We introduce a novel fabrication strategy for wire sculptures with complex geometries by tuning the target shape to a collision-free shape for the wire-bending machine and then bending it back to the target by a human. The key challenge lies in tuning the least number of bending points, which is formulated as an "Optimizing Wire Reconfiguration" problem. We first fit the input target wire with consecutive line segments and circular segments to ensure the bending manufacturing constraints for each segment, then generate tuned wire through a bilevel optimization. This involves selecting the bending points at the upper level with a beam search strategy and determining the specifically tuned angles at the lower level. We perform a thorough physical evaluation using a DIY wire-bending machine. The results show the effectiveness of our proposed approach in realizing a wide range of intricate and complex wire sculptures.

CCS Concepts: • **Computing methodologies** → **Shape modeling**; *Graphics systems and interfaces.*

Additional Key Words and Phrases: Wire sculpture, wire fabrication, wire reconfiguration, computational fabrication

ACM Reference Format:
Qibing Wu, Zhihao Zhang, Xin Yan, Fanchao Zhong, Yueze Zhu, Xurong Lu, Runze Xue, Rui Li, Changhe Tu, and Haisen Zhao. 2024. Tune-It: Optimizing Wire Reconfiguration for Sculpture Manufacturing: Supplemental Material. *ACM Trans. Graph.* 1, 1 (September 2024), 10 pages. https://doi.org/10.1145/3680528.3687588

*Equal contribution
†Corresponding author

Authors' addresses: Qibing Wu, qibingwu0228@gmail.com, Shandong University, Qingdao, China; Zhihao Zhang, zhangsfsd1@gmail.com, Shandong University, Qingdao, China; Xin Yan, io.yanxin@gmail.com, Shandong University, Qingdao, China; Fanchao Zhong, fanchaoz98@gmail.com, Shandong University, Qingdao, China; Yueze Zhu, yueze.zhu@outlook.com, University College London, London, United Kingdom; Xurong Lu, lxr20030328@foxmail.com, Shandong University, Qingdao, China; Runze Xue, runzexue@outlook.com, Shandong University, Qingdao, China; Rui Li, ruizai6@hotmail.com, Shandong University, Qingdao, China; Changhe Tu, chtu@sdu.edu.cn, Shandong University, Qingdao, China; Haisen Zhao, haisenzhao@sdu.edu.cn, Shandong University, Qingdao, China.

0730-0301/2024/9-ART $15.00
https://doi.org/10.1145/3680528.3687588

A STRIKE BENDING AND INTERPOLATED BENDING

Two strategies, strike bending and interpolated bending, are employed when manufacturing a circular segment.

The circular segment produced by interpolated bending is faster than strike bending but has a limited minimum radius R_{min}. If the radius of circular segment is less than R_{min}, the bending angle becomes very large. At this point, when the feeder continues to drive the wire forward, the bending pin blocks the wire forward with significant resistance, resulting in bending failure. Therefore, in this case, strike bending is more advantageous to form circular segments. Although strike bending uses multiple incremental flexion bending operations that require additional time, it can avoid the limitation of R_{min}.

B MANUFACTURING CONSTRAINTS DETAILS

This section elucidates the details of the three manufacturing constraints.

Minimal length constraint. For each bending operation, the wire must contact the bending pin in order to be bent. This requirement is defined as the Minimal length constraint. The distance between the bending pin and the wire outlet, L_{min}, is the minimal length and is a constant constraint. For our DIY machine L_{min} has been measured to be 7 mm.

Bending angle range constraint. For our DIY bending machine, the bending angle is constrained by the rotation limit of the bending pin, which is synchronized with the rotation of the bending head. Due to the spring-back effect of metal materials, the maximum bending angle is smaller than the maximum rotation angle of the bending pin. The maximum bending angle, α_{max}, defines the range of the bending angle as [-α_{max}, α_{max}]. For our case, α_{max} has been measured to be 110°.

G1 line segment constraint. Our shape consists of line segments and circular segments, with an angle α^l between them. When employing interpolated bending to form a circular segment, we need to rotate the bending pin to a specific angle α^c and fix it in place. Subsequently, we feed the wire to the corresponding length. However, due to the distance between the outlet and the bending pin, before forming a circular segment, the wire must be fed to make contact with the bending pin. Therefore, there must be a line segment with a length of at least L_{min} before each circular segments. Furthermore, the line segment preceding each circular segment must be G1 continuous so that circular segment can be manufactured. This occurs because the shape formation position of circular segment is at the outlet as the bending pin rotates against the wire. When the line segment before circular segment already contacts the bending pin, α^l cannot be changed anymore. Thus, a circular segment is bending fabricable only when it has a G1 continuous line segment in preceding.

C SEGMENT FITTING

We use three different fitting strategies to fit segments.

C.1 Line segment fitting (LSF)

In the fitting process, line segment $\bar{s}$ is fitted by connecting the first point p_a and the last point p_{b+1} of all the merged bending segments $\{s_a, s_{a+1}, ..., s_b\}$.

C.2 Circular segment fitting without constraints (CSF)

In the fitting process, circular segment $\check{s}$ is obtained using least squares fitting from the points $\{p_a, p_{a+1}, ..., p_b, p_{b+1}\}$ of all the merged bending segments $\{s_a, s_{a+1}, ..., s_b\}$. Note that p_a and p_{b+1} must be in the fitted circular segment. The specific process is as follows.

We first fit the points $\{p_a, p_{a+1}, ..., p_b, p_{b+1}\}$ to a plane using CGAL [Fabri and Pion 2009]. Next, project all points and the plane onto a two-dimensional plane for circular fitting. We compute a series of radii and center of the circles, which are formed by each point p_k in $\{p_{a+1}, ..., p_b\}$, in conjunction with p_a and p_{b+1}. Then a binary search iteration is used to find the fitted circular segment with the minimal error, defined as the sum of the Euclidean distance from $\{p_{a+1}, ..., p_b\}$ to the fitted circle. Finally, the two-dimensional circular segment is projected back into the three-dimensional space to obtain a well-fitted circular segment $\check{s}$.

C.3 Circular segment fitting with constraints (CSFWC)

Due to the G1 line segment constraint outlined in Appendix B, the fitted circular segment in Subsection C.2 is not bending fabricable. To satisfy the manufacturing constraint, we re-fit circular segment $\dot{s}_i$ after the graph cut to generate a pair of $(\bar{s}'_i, \check{s}'_i)$, and they are two fitted segments of $\dot{s}_i$, which ensures the G1 line segment constraint.

Similarly to Subsection C.2, we use the least squares fitting to fit the points $\{p_a, p_{a+1}, ..., p_b, p_{b+1}\}$ of $\dot{s}_i$. Only when calculating the radius and center of a circle are different, where we will ensure that there is a G1 continuous line segment before circular segment. For each p_i in $\{p_{a+1}, ..., p_b\}$ combined with p_a and p_{b+1}, we fit a $(\bar{s}'_i, \check{s}'_i)$. The direction of $\bar{s}'_i$ coincides with the tangential direction at the starting point p_t of $\check{s}'_i$. The key to solving the problem is to find the intersection point between a circle centered on p_a with a radius of L_{min} and a circle that satisfies G1 continuity and passes through the points p_i and p_{b+1}. One of the intersection points is the p_t.

In addition, we will also fit $\dot{s}_i$ to a line segment $\bar{s}'_i$ by connecting the first point and the last point of $\dot{s}_i$.

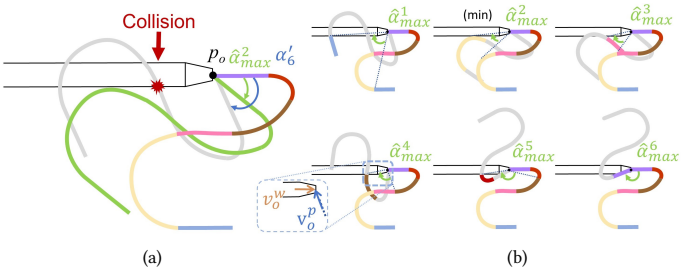

Fig. 1. Demonstration of FAO. (a) The colored shape is the state before bending α'_6, the gray shape is the state after bending α'_6, and the green shape is the state where bending angle $\tilde{\alpha}^6_{max}$ ($\widehat{\alpha}^2_{max}$) without collision. $\alpha'_6 \notin [0, \tilde{\alpha}^6_{max}]$, collision will occur. (b) indicates the computed $\{\widehat{\alpha}^1_{max}, \widehat{\alpha}^2_{max}, \widehat{\alpha}^3_{max}, \widehat{\alpha}^4_{max}, \widehat{\alpha}^5_{max}, \widehat{\alpha}^6_{max}\}$ (Green arrow). The minimum $\widehat{\alpha}^i_{max}$ is $\widehat{\alpha}^2_{max}$.

D FEASIBLE ANGLE OPERATOR

As shown in Figure 1, the feasible range of p'_i includes all the rotating angles of α'_i that do not lead to any collision between the preceding segments $\{s^*_0, s^*_1, ..., s^*_{i-1}\}$ of p'_i and the wire-bending machine, where each bending segment s^*_j produces a restrained feasible range to α'_i for collision-free. Moreover, we observe that $\{s^*_0, s^*_1, ..., s^*_{i-1}\}$ shares the same rotation transform as the bending pin, whose rotation realizes the bend angle α'_i of p'_i. Hence, the feasible range of p'_i is indeed the intersection of the feasible ranges of each bending segment s^*_j, whose maximal feasible angle is:

$$\widehat{\alpha}^j_{max} = \min\{\angle(v^p_o, v^w_o) - \arcsin(\frac{0.5d}{\overline{v^p_o}}), p \in s^*_j\} \tag{1}$$

where d is the diameter of the wire tube, p is any points of s^*_j, v^p_o is a vector from p to p_o, $v^p_o = p - p_o$, p_o is the pin's rotation center that aligns with the outlet center, v^w_o is a direction vector of the wire outlet, $\angle(v^p_o, v^w_o)$ is the angle between vectors, $\overline{v^p_o}$ is the length of v^p_o. Finally, the feasible range of p'_i is $[0, \tilde{\alpha}^i_{max}]$, $\tilde{\alpha}^i_{max} = \min\{\widehat{\alpha}^1_{max}, \widehat{\alpha}^2_{max}, ..., \widehat{\alpha}^{i-1}_{max}, \widehat{\alpha}^i_{max}\}$.

E CRO OPTIMIZATION

For CRO in bilevel tuned wire optimization, we attempt to use nonlinear continuous optimization methods to tune the bending angles $\{\alpha^*_1, ..., \alpha^*_i\}$ of tuned points in $\{p'_1, p'_2, ..., p'_i\}$ with ipopt [Waechter et al. 2002]. Here is the formulation:

$$\begin{aligned} &\min_{\{\overline{\alpha}^*_1, \overline{\alpha}^*_2, ..., \overline{\alpha}^*_k\}} (\alpha^*_i - \alpha'_i)^2 \\ &s.t. \quad \widehat{\alpha}^i_{max} = \min\{\angle(v^p_o, v^w_o) - \arcsin(\frac{0.5d}{\overline{v^p_o}}), p \in s^*_i\} \\ &\qquad \alpha^*_i > \widehat{\alpha}^i_{max}, j = 0...i \end{aligned} \tag{2}$$

Fig. 2. Comparison of the number of tuned points of our beam search method (left figure of each model) and the greedy-based method (right figure of each model). Our method can obviously reduce the manual operation cost. Our algorithm takes 22.11 seconds on average, and the greedy-based method takes 13.57 seconds on average.

where i is the last angle when calling CRO, $\widehat{\alpha}^{i}_{max}$ is calculated in the feasible angle operator (FAO), and $\{\overline{\alpha}^{*}_{1}, \overline{\alpha}^{*}_{2}, ..., \overline{\alpha}^{*}_{k}\}$ are tuned points in the current state.

The primary issue with this process is its slowness, 40% - 60% slower than the heuristic-based searching strategy described in the main text. The reason for this is as follows: This optimization process requires multiple iterations to achieve the optimal goal. During each iteration, multiple derivatives must be computed for each point in that state. We must also consider that all preceding segments can be bent without collisions with the changed tuned points values. In other words, we need to consider all the historical states before the current collision. After a certain number of iterations, a set of feasible solutions may be found, but optimization continues in an attempt to further minimize the optimization objective. This results in many iterations being unnecessary. But limiting the number of iterations to reduce time may lead to a wrong solution. Based on the reason, we use the heuristic-based searching strategy to tune these tuned points.

However, we must acknowledge that although this method is relatively slow in current testing, it excels over the method in our main text in certain extreme cases where there are numerous tuned points and almost all of tuned points need to be changed simultaneously to avoid collisions. Nevertheless, such situations are rare in wire art.

F GREEDY VS. BEAM SEARCH

Being the initial algorithm for bending a single wire that incorporates the two-stages-bending strategy, no current methods address the exact OWR problem that we do. We take the greedy-based method proposed as a benchmark for comparison against our beam search strategy. As indicated in Figure 2, the beam search strategy results in fewer tuned points, albeit with a marginally increased execution time, thus demonstrating its efficiency.

G OVERALL ALGORITHM LOOP

In candidate nodes generation of beam search, if all nodes in $\mathcal{T}$ are abandoned, it means that a collision-free solution cannot be found at the moment. To solve this situation, we need to start from fitting again.

We divide the initial input, a series of line segments, into smaller segments. As the number of initial segments increases, the number

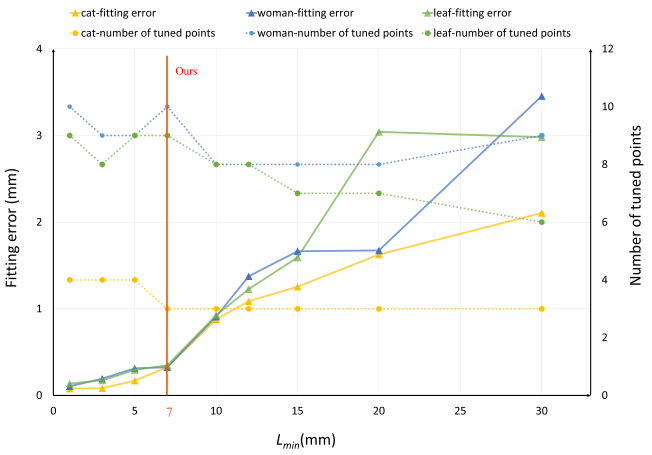

Fig. 3. Illustration the impact of minimum manufacturing length L_{min} on fitting errors and the number of tuned points. As L_{min} increases, the fitting error continues to increase, and the overall number of tuned points decreases. Due to the requirement for a specific distance between the bending pin and the wire outlet of the machine, we ultimately select a relatively optimal value 7 mm to L_{min}.

of candidate fabricable segments also increases. When performing the beam search again, there may be an increase in the bending points that become tuned points, thus having a greater chance of finding collision-free solutions. If not found, repeat the loop until a solution is found.

H SETTING TUNED SCORE FOR BENDING POINTS

We give each bending point p'_i a weight called tuned score to help determine tuned points from $\{p'_1, ..., p'_m\}$ in the subsequent process. Inspired by [Lira et al. 2018], we first traverse each bending segment s'_i, performing the forward-and-backward searching to connect adjacent bending segments, getting the longest non-collision part $\{s^{b\prime}_1, ..., s^{b\prime}_m\}$ of $\mathcal{W}'$, and recording bending points p'_i at both ends. The p'_i is considered to be what causes the collision. Then we count the occurrences of each p'_i appearing at the ends and denote this count as $\mathcal{L}(i)$. The higher the value of $\mathcal{L}(i)$, the more likely p'_i is designated as a tuned point. The formulation is as follows:

$$\mathcal{L}(i) = \sum_{j \in m} T(i, s^{b\prime}_j), \quad T(i, s^{b\prime}_j) = \begin{cases} 1, if\ p'_i\ is\ end\ of\ s^{b\prime}_j \\ 0, otherwise \end{cases} \tag{3}$$

I OPTIMIZING WIRE RECONFIGURATION RESULTS

We perform various experiments to evaluate our algorithm. Different hyperparameters and setting are assessed below.

Fitting error threshold. We use the fitting error threshold ϵ as the termination condition in the candidate fabricable segments generation step. Consequently, it is ensured that the fitting errors of the results during the graph cut remain below the ϵ. However, the fitting error tends to increase due to the re-fitting phase, which will ignore ϵ to meet manufacturing constraints. As shown in Figure 13 (paper), there exists a non-linear correlation among various ϵ and their corresponding fitting errors across eight different shapes, with the minimum error not being consistent across all shapes. Typically,

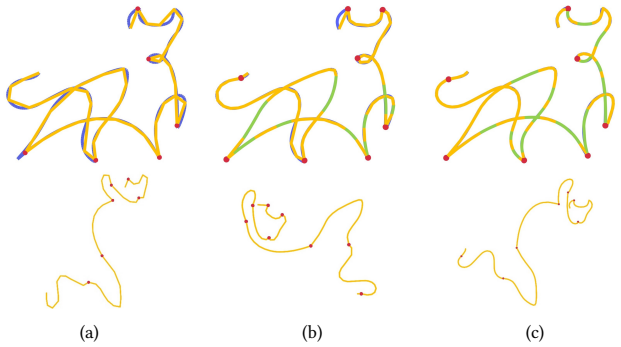

Fig. 4. Illustration of the wire size experiment. The blue shape is the input model and the green-yellow shape (line segment: Green, circular segment: Yellow) is the fitted model. The red dots are the tuned points. The bounding box dimensions of the small bull model (a) are 118.90 mm by 102.34 mm. The bounding box dimensions of the medium-sized bull model (b) are 297.25 mm by 255.84 mm. The bounding box dimensions of the big bull model (c) are 594.494 mm by 511.68 mm.

an increase in ϵ initially decreases the fitting error and then begins to increase. The reason is that small ϵ leads to many short segments that require extensive refitting, increasing the error. In contrast, a large ϵ inherently has a significant error. We choose ϵ to be 0.6 mm based on the results of all examples.

Two types of fitting segments. In our final results (Figure 9 in paper), two types of fitting segments are utilized to fit the input wire $\mathcal{W}$, with line segments generated by flexion bending and circular segments generated by interpolated bending and strike bending. Our technique for the OWR problem is general enough to employ either line segments or circular segments to fit $\mathcal{W}$ separately. In Figure 12 (paper), we compare the fitting errors of the combined strategy (c) with pure line segments (a) and pure circular segments (b). It is observed that the combined strategy results in significantly lower average fitting errors compared to the individual line segment and circular segment strategy, as illustrated in 0.682 mm vs. 0.908 mm vs. 0.499 mm, while the number of tuned points is almost the same (7 vs. 8 vs. 7).

Minimum segment length. As illustrated in Figure 3, an increase in the minimum length of line segment L_{min} leads to a rise in the fitting error and a decrease in the number of tuned point. The quantity of tuned points does not show significant variations as L_{min} values ranging from 1 mm to 30 mm. A substantial disparity in fitting errors is observed as a result of varying L_{min}. Setting L_{min} to approximately 10 mm results in a significant escalation in the fitting error across all three shapes. Despite the fact that the fitting errors are remarkably minor when L_{min} is less than 5 mm, our DIY wire-bending machine constraints necessitate setting L_{min} at 7 mm in our experiments.

Wire size. The wire size is an important hyperparameter in our algorithm. Figure 4 illustrates the segment-bendable wire $\mathcal{W}'$ and the tuned wire $\mathcal{W}^*$ of the *bull* model with varying wire sizes, while adhering to default manufacturing constraints (such as G1 line segment constraint) et al. With the wire size increasing, it is observed that the average fitting error decreases (0.900 mm vs. 0.816 mm vs. 0.499 mm), the number of bending segments increases (48 vs. 80 vs. 107), the number of tuned points almost the same (6 vs. 8 vs. 7). Furthermore, our algorithm will output more circular segments with a larger wire size while the same size one does not have circular segment.

Bending order. A wire $\mathcal{W}$ can be produced starting from either endpoint in the forward or reverse bending orders. As shown in Figure 5, there is a slight variation in the number of tuned points produced when tuned wire is created from both orders of the *Bird* and *Cat* models. Typically, when the initial bending sequence of a shape bends inward continuously, it tends to lead to more collisions, resulting in more tuned points. Conversely, if the bending sequence expands outward, it generally reduces the likelihood of collisions, such as the spiral shape, if we start the bending from the center outward.

Scalability. For geometrical structures in Figure 9 (paper), our algorithm effectively produces the tuned wire with an appropriate count of tuned points. As we explore more complex geometric structures, our approach still shows its ability of scalability. In Figure 15 (paper), we apply the space-filling curve result developed by [Zhao et al. 2016] to illustrate the scalability of our algorithm in managing highly complex structures. The *Three-People* curve is 35669.2 mm long and has a bounding box that measures 592.165 mm by 487.429 mm. With such a complex input, our algorithm takes about 61 hours to generate 160 tuned points. Most of the algorithm's time is spent on CRO, and as the number of tuned points increases, each subsequent CRO process may take longer. In addition, the complex geometry and large number of bending points greatly increase the number of CRO.

3d Shapes. Our algorithm is also applicable to 3D wires. The primary distinction between 2D and 3D wires lies in the way bending angles are formulated. In 2D wires, each bending angle is adjusted solely within the 2D plane. In contrast, 3D wires require the inclusion of an axial angle to denote the orientation of the bending plane relative to the horizontal plane. Consequently, each bending angle in the 3D wires has two degrees of freedom to adjust. Therefore, the operators for resolving collisions become somewhat more intricate due to the consideration of the rotation of the bending head and the collision between the wire and the bending head. In Figure 6, we show the computational results of several 3D wires. For the *Bike*,

Fig. 5. Illustration of reversing the sequence of wire. After the fitting is completed, we reverse the fitted elements sequences, getting different number of tuned points. We select less one for final fabrication.

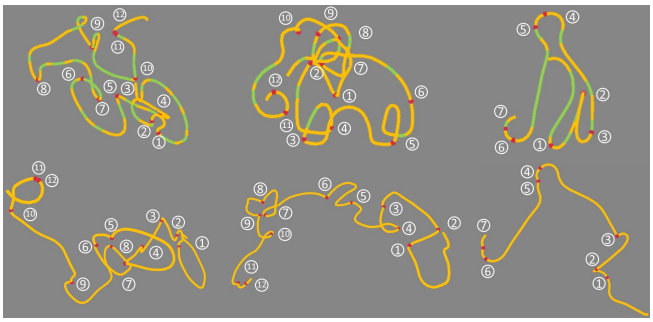

Fig. 6. 3D results gallery generated by our algorithm. The models are arranged in the order of *Bike*, *Elephant*, *Human*. Top photos are fiiting result (line segment: Green, circular segment: Yellow), and the bottom photos are tuned wires.

Elephant and *human* models [Liu et al. 2017], our algorithm takes approximately 2.9 hours on average to produce 10 tuned points on average.

Failure case. The algorithm may fail when there are several consecutive circular segments. As shown in Figure 16 (paper), the input W' has two consecutive circular segments with rounded center angles exceeding π. No amount of change in the angle of the tuned point p_1^* can generate W^* while satisfying the constraints.

J EXPERIMENTAL ENVIRONMENT

We constructed a desktop-level wire-bending machine based on [Dejan 2018]. However, due to the extensive use of 3D printed parts in the tutorial, which are characterized by low precision and susceptibility to wear, the bending machine exhibits significant errors in the length of the wire feed and the angle of bending, reaching 30% – 50%. Therefore, we made several improvements to alleviate this problem, the machine modified shown in Figure 4 (paper). Specifically, we used precision machined iron parts for the straighteners, feeder, and bending head instead of 3D printed components. Moreover, the feeder rotates under the drive of the motor and feeds the wire through friction, so we had performed a knurling treatment on it. In addition, we replaced the wooden boards with aluminum plates for the pedestal to avoid board deformation. These enhancements significantly improved the accuracy of our machine compared to [Dejan 2018]. The bending machine control program is implemented using Arduino, with the Arduino MEGA2560 R3 development board facilitating the entire bending process. Our machine is equipped with three stepper motors: two 57 stepper motors control the feeder feed and the z-axis bending separately, while a 42 stepper motor governs the bending head. All motors are driven by the DM542C driver, with a motor subdivision of 16. The stride angle of the 57 stepper motor is 1.8 °and that of the 42 stepper motor is 0.9 °. In addition, a servo controls the up and down movement of the bending pin. The bending machine operates at a motor speed of 600 pulse signals per second. We use a 14 mm diameter copper tube for the main axle, and the wire used is aluminum, with a diameter of 1.5 mm and a Vickers Hardness of 20 HV.

K PHYSICAL EVALUATION OF THE BENDING.

Although we have implemented numerous enhancements to our machine, it still exhibits several issues that result in errors during the bending process, necessitating further consideration.

- *The gravity of wire*: During the bending process, the wire will inevitably sag due to its own gravity. As the length of the bent wire increases, the influence of gravity becomes increasingly apparent.
- *The spring-back effect of the wire*: The spring-back effect of metal materials has always been a complex problem and it is imperative to compensate for it. A data-driven methodology is adopted that gives the machine an input angle and measures the output angle. Subsequently, we perform a linear interpolation and inversion on it, obtaining a mapping of desired angles to the compensated input angles, including the lookup table for line segments and for circular segments, as shown in Appendix L.
- *The rotation of the wire*: Due to the hollow tube of the axis in our machine, the two ends of the wire are not fixed between the feeder and the wire outlet. When the feeder rotates to feed the wire, the wire will unconsciously rotate, resulting in inaccurate bending angles. For this reason, the error in the 3D shape is relatively large.
- *The limitations of machine*: Compared to industrial-grade bending machines, the motor used in our machine may experience step loss, the straightener used cannot fully straighten the wire, the number of gear teeth cannot accurately correspond to each angle, and the limitation of motor drive subdivision makes it difficult to precisely convert angles into steps, etc, resulting bending errors.

We will consider machine bias in our algorithms in the future.

L LOOKUP TABLE OF MACHINE BENDING ANGLE

The spring-back effect of metal materials has always been a complex problem in industrial production. To mitigate this issue, we adopted a data-driven methodology to compensate for the rebound. For line segments, we provide the machine with a command for the desired angle α^d, and after the machine bends the wire, we measure the actual bending angle α^c, as shown in Table 1. When fabricating a circular segment using interpolated bending, the rotation angle α^d of the bending pin is calculated using the radius of the desired circular segment R^d, as shown in Table 2.

For our DIY machine, the rotation of the bending pin is driven by a motor with pulse signals PS, which rotates the bending head via a gear mechanism. The gear tooth ratio is 1:2 and the motor subdivision is 16, meaning the bending pin rotates 360°with 12800 motor steps. Hence, PS is equal to $\alpha^d * 35.5556$, which is the data input to the machine.

Finally, we perform linear interpolation and inversion on the two table, obtaining the mapping of desired angles and radius to the compensated input angles.

Table 1. This table lists the line segments lookup table of the input pulse signals PS (desired bending angles) to the compensated bending angles. Each row indicates (**The input pulse signals** PS), (**The compensated bending angles** α^c).

PS **(step)**	α^c (°)	PS **(step)**	α^c (°)	PS **(step)**	α^c (°)
350	1.0	1500	26.8	2700	71.3
400	2.1	1600	28.8	2800	75.2
500	3.3	1700	33.2	2900	78.0
600	4.2	1800	36.3	3000	81.5
700	5.0	1900	38.8	3100	86.2
800	7.8	2000	44.0	3200	90.0
900	8.7	2100	47.3	3300	93.4
1000	11.3	2200	49.6	3400	96.7
1100	14.2	2300	54.3	3500	100.8
1200	17.5	2400	58.8	3600	106.1
1300	20.0	2500	61.5	3700	109.9
1400	24.4	2600	64.4	3800	113.0

Table 2. This table lists the circular segments lookup table of the input pulse signals PS (desired radius R^d of circular segment) to the compensated radius R^c of circular segment. Each row indicates (**The input pulse signals** PS), (**The compensated radius** R^c).

PS **(step)**	R^c **(mm)**	PS **(step)**	R^c **(mm)**
400	800	700	83
450	350	750	72
500	260	800	64
520	247	850	45
550	213	900	34
570	175	950	27
600	124	1000	22
650	106	1100	15

M TERMINOLOGY LIST

Table 3. This table lists all parameters used in our algorithm. Each row indicates the parameter (**Param**), the meaning of the parameter (**Significance**), and the related section where the parameter appears for the first time (**Usage**).

Param	**Significance**	**Usage**
R_{min}	the minimum radius of interpolated bending	Section 3.1
L_{in}	the incremental feeding distance of strike bending	Section 3.1
L_{min}	the distance between the bending pin and the wire outlet, i.e. the minimum length of line segment	Section 3.1
α_{max}	the maximum bending angle	Section 3.1
$\mathcal{W}$	the input wire	Section 3.2
s_i	bending segment of $\mathcal{W}$	Section 3.2
p_i	end point of s_{i-1} and s_i, bending point of $\mathcal{W}$	Section 3.2
α_i	bending angle between s_{i-1} and s_i	Section 3.2
$\mathcal{W}'$	the segment-bendable wire	Section 3.2
$\mathcal{W}^*$	the collision-free tuned wire	Section 3.2
p'_i	end point of s'_{i-1} and s'_i, bending point of $\mathcal{W}'$	Section 3.2
α'_i	bending angle of $\mathcal{W}'$	Section 3.2
p^*_i	bending point of $\mathcal{W}^*$	Section 3.2
α^*_i	bending angle of $\mathcal{W}^*$	Section 3.2
s^*_i	bending segment of $\mathcal{W}^*$	Section 3.2
$\mathcal{M}$	wire-bending machine	Section 3.2
$\overline{s^*_i}$	the length of line segment s^*_i	Section 3.2
$R(s^*_i)$	the radius of circular segment s^*_i	Section 3.2
$\hat{s}_i$	a candidate fabricable segment	Section 4
$\bar{s}_i$	a candidate fabricable line segment	Section 4
$\check{s}_i$	a candidate fabricable circular segment	Section 4
$E(\bar{s})$ or $E(\check{s})$	the fitting error, the maximum Euclidean distance from the fitted bending segment to the traversed bending segments	Section 4
ϵ	the fitting error threshold	Section 4
l_i	the i-th candidate fabricable segment	Section 4
$d(s_j, l_i)$	the fitting error distance between s_j and l_i	Section 4
λ_1	a scaling coefficient of $d(s_j, l_i)$	Section 4
$L(l)$	label term	Section 4
λ_2	the penalty coefficient of number of labels	Section 4
$\bar{s}'_i$	a fitted line segment of $\hat{s}_i$	Section 4
$(\bar{s}'_i, \check{s}'_i)$	two fitted segments of $\hat{s}_i$ which ensures the G1 line segment constraint	Section 4
s'_i	bending segment of $\mathcal{W}'$	Section 4
$\tilde{\alpha}^i_{max}$	the maximal feasible bending angle of p'_i	Section 5.1
$\widehat{\alpha}^i_{max}$	the maximal feasible ranges of each bending segment s^*_j	Section 5.1
d	the diameter of the wire tube	Section 5.1
v^p_o	the vector from p to p_o, $v^p_o = p - p_o$	Section 5.1
p_o	the pin's rotation center that aligns with the outlet center	Section 5.1
v^w_o	the direction vector of the wire outlet	Section 5.1
$\overline{v^p_o}$	the length of v^p_o	Section 5.1
$\mathcal{T}$	the beam search tree	Section 5.2
ω	the probability of setting p'_i to tuned point and generating a node of $\mathcal{T}$	Section 5.2
$\mathcal{L}(p^*_j)$	the tuned score of tuned points	Section 5.2
W_{beam}	the number of candidate solutions in a beam search,i.e. beam width	Section 5.2

Table 4. This table lists all the terms used in this paper. Each row indicates the meaning of the terminology(term) and the related section where the term used for the first time (Usage).

Term	Usage
wire sculpture: the creation of sculpture out of wire.	Abstract
wire-bending machine: a machine that can bend wire into different shapes.	Abstract
bending point: the point between different segments.	Abstract
Optimal-Wire-Reconfiguration (OWR): reconfigure wire to make it collision-free with minimal adjustments.	Abstract
line segment: straight-line segment.	Abstract
circular segment: segment of a circular arc.	Abstract
tuned wire: a deformed wire by tuning bending points to avoid collision with the wire-bending machine.	Abstract
collision-free constraint: the wire does not collision with the wire-bending machine during bending process.	Section 1
Machine-And-Then-Human-Bending: a two-stages-bending strategy with the machine bending first, followed by human.	Section 1
machine-bending stage: the tuned wire bent by wire-bending machine.	Section 1
human-bending stage: human bend the tuned wire to the desired shape.	Section 1
tuned point: a bending point with tuned angle that its α^* is not equal to α'.	Section 1
bending angle: angle of bending point.	Section 1
optimal reconfiguration planning (ORP): ORP searches for the least number of reconfiguration steps to transform between configurations.	Section 1
bending segment: basic bending unit of wire.	Section 1
fabricable bending segment: bending segment that meets manufacturing constraints.	Section 1
"decompose-then-assemble" strategy: a strategy that decomposes the wire into a series of collision-free subwires to manufacture first and then manual assembles them.	Section 2
bending pin: component used to complete bending.	Section 3.1
flexion bending: a bending strategy to bend a line segment.	Section 3.1
interpolated bending: a bending strategy to bend a circular segment with the constrain R_{min}.	Section 3.1
strike bending: a bending strategy to bend a circular segment without the constrain R_{min}.	Section 3.1
manufacturing constraints: minimal length constraint, bending angle range constraint, G1 line segment constraint.	Section 3.1
bending fabricable: can be fabricated by machines.	Section 3.1
segment-bendable wire: a wire whose segments are bending fabricable.	Section 3.2
constant point: a bending point that its $\alpha^* = \alpha'$.	Section 3.2
forward-and-backward traverse procedure: traverse in both backward and forward directions.	Section 4
candidate fabricable segment: the segment gotten after forward-and-backward traverse.	Section 4
none-overlapping fabricable segment: the segment without overlap obtained after graph-cut.	Section 4
feasible bending angle range / feasible range: The feasible range of p'_i includes all the rotating angles of α'_i that do not lead to any collision between the preceding segments $\{s_0^*, s_1^*, ..., s_{i-1}^*\}$ of p'_i and the wire-bending machine.	Section 5.1
Feasible Angle Operator (FAO): FAO is used to calculate the feasible bending angle range.	Section 5.1
Collision Resolving Operator (CRO): CRO aims to resolve the collisions by tuning the bending angles of tuned points.	Section 5.1

N PSEUDOCODE

N.1 Pseudocode of segment-bendable wire generation

Algorithm 1 Segment-bendable Wire Generation

Input: $S = \{s_0, s_1, ..., s_n\}$ $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ $P = \{p_0, p_1, p_2, ..., p_{n+1}\}$
Output: Segment-bendable wire $S' = \{s'_0, s'_1, ..., s'_m\}$
Output: Bending points $\{p'_1, ..., p'_m\}$

//Candidate fabricable segments
for each s_i in S **do**
  //Find longest line segment $\bar{s}$
  while $i + k_f <$ n and $i - k_f > 0$ and $E_{\bar{s}_i} < \epsilon$ **do**
    LSF(i-k_f,i+k_f) k_f++;
  $L_{\bar{s}_i} = k_f$ - k_b
  //Find longest circular segment $\check{s}$
  while $i + k_f <$ n and $i - k_f > 0$ and $E_{\check{s}_i} < \epsilon$ **do**
    CSF(i-k_f,i+k_f) k_f++;
  $L_{\check{s}_i} = k_f$ - k_b
  if $L_{\check{s}_i} < L_{\bar{s}_i}$ **then** $\dot{s}_i$ is $\check{s}_i$
  else $\dot{s}_i$ is $\bar{s}_i$

//remove duplicated Candidate fabricable segments
if $\dot{s}_i = \dot{s}_j$ **then** remove $\dot{s}_j$
Get $\{\dot{s}_0, \dot{s}_1, ..., \dot{s}_d\}$

//Graph cut for none-overlapping fabricable segments
$\{\dot{s}_0, \dot{s}_1, ..., \dot{s}_k\}$ = Graph-cut($\{\dot{s}_0, \dot{s}_1, ..., \dot{s}_d\}$)

//meet the manufacturing constraints
for each $\dot{s}_i$ in $\{\dot{s}_0, \dot{s}_1, ..., \dot{s}_k\}$ **do**
  if $L_{\dot{s}_i} < L_{min}$ **then**
    Merge $L_{\dot{s}_i}$ with $L_{\dot{s}_{i-1}}$, compute E_b
    Merge $L_{\dot{s}_i}$ with $L_{\dot{s}_{i+1}}$, compute E_f
    if $E_b < E_f$ **then** Merge $L_{\dot{s}_i}$ with $L_{\dot{s}_{i-1}}$
    else Merge $L_{\dot{s}_i}$ with $L_{\dot{s}_{i+1}}$
  if $\dot{s}_i$ is $\check{s}_i$ **then**
    $E_{\check{s}_i} = LSF(\check{s}_i)$ //fitting line segment
    $E_{\check{s}_i} = CSFWC(\bar{s}_i)$ //fitting circular segment with constrains
    if $E_{\check{s}_i} < E_{\bar{s}_i}$ **then** $\dot{s}_i$ is $\check{s}_i$
    else $\dot{s}_i$ is $\bar{s}_i$
  $s'_i = \dot{s}_i$
return $\{s'_0, ..., s'_m\}$ $\{p'_1, ..., p'_m\}$

function GRAPH-CUT($\{\dot{s}_0, \dot{s}_1, ..., \dot{s}_d\}$)

$$\mathcal{E} = \sum_{s_j \in S_w} D(s_j, l_i) + \sum_{(s_j, s_{j+1}) \in S_w} S(s_j, s_{j+1}, l) + L(l) \tag{4}$$

function LSF(a, b)
  $line(a, b+1)$ is connecting p_a and $p_b + 1$
  for j in (a,b) **do**
    $E_{\bar{s}_i}$ =maxEuclidean distance of p_i to $line(a, b)$
function CSF(a, b)
  fitting a plane using $\{p_a, p_{a+1}, ..., p_{b+1}\}$
  using least squares fitting to find $cir(a, b)$
  $cir(a, b)$ is a circular that p_a and $p_b + 1$ are the two ends.
  for j in (a,b) **do**
    $E_{\check{s}_i}$ =maxEuclidean distance of p_i to $cir(a, b)$
function CSFWC(a, b)
  fitting a plane using $\{p_a, p_{a+1}, ..., p_{b+1}\}$
  using least squares fitting to find $(\bar{s}'_i, \check{s}'_i)$
  $(\bar{s}'_i, \check{s}'_i)$ are a line segment following a G1 continuous circular segment.
  for j in (a,b) **do**
    $E_{\check{s}_i}$ =maxEuclidean distance of p_i to $(\bar{s}'_i, \check{s}'_i)$

N.2 Pseudocode of CRO

Algorithm 2 Collision Resolving Operator

Input: $S^* = \{s_0^*, s_1^*, ..., s_i^*\}$
Input: $A^* = \{\alpha_1^*, \alpha_2^*, ..., \alpha_{i+1}^*\}$
Input: $P^* = \{p_1^*, p_2^*, ..., p_{i+1}^*\}$
Output: Angles with collision-free $\hat{A}^* = \{\alpha_1^*, \alpha_2^*, ..., \alpha_{i+1}^*\}$

function CRO(S^*, A^*, P^*)
  // number of tuned points
  $n_t \leftarrow 0$
  // set of tuned points; initialize $\hat{A}^*$
  $P_t \leftarrow \emptyset$; $\hat{A}^* \leftarrow \emptyset$
  for each p_i^* in P^* **do**
    if p_i^* is tuned point **then**
      $P_t \cup \{p_i^*\}$
      n_t+ = 1
  for int $d \leftarrow 1$; $d \leq n_t$; d++ **do**
    $\hat{A}^* \leftarrow A^*$
    // tune a subset of A^*, d is the size of the subset
    return *sub_cro*$(d, S^*, \hat{A}^*, P_t)$
  return $\emptyset$

function SUB_CRO($depth, S^*, \hat{A}^*, \hat{P}_t$)
  if $d = 1$ **then**
    for each $\hat{p}_i^*$ in $\hat{P}_t$ **do**
      for $\alpha_t \leftarrow 1$; $\alpha_t \leq \alpha_{max}$; α_t++ **do**
        $\hat{\alpha}_i^*$ += α_t
        if check_collision_free($S^*, \hat{A}^*, \hat{P}_t$) **then**
          return $\hat{A}^*$
        α_i^* -= 2 * α_t
        if check_collision_free($S^*, \hat{A}^*, \hat{P}_t$) **then**
          return $\hat{A}^*$
    return $\emptyset$
  else
    for each $\hat{p}_i^*$ in $\hat{P}_t$ **do**
      for $\alpha_t \leftarrow 1$; $\alpha_t \leq \alpha_{max}$; α_t++ **do**
        $\hat{\alpha}_i^*$ += α_t
        $\hat{P}_t \leftarrow \hat{P}_t - \{\hat{p}_i^*\}$
        if *sub_cro*$(depth - 1, S^*, \hat{A}^*, \hat{P}_t) \neq \emptyset$ **then**
          return $\hat{A}^*$
        α_i^* -= 2 * α_t
        if *sub_cro*$(depth - 1, S^*, \hat{A}^*, \hat{P}_t) \neq \emptyset$ **then**
          return $\hat{A}^*$
    return $\emptyset$

N.3 Pseudocode of Beam search

Algorithm 3 Beam Search

Input: $S' = \{s_0', s_1', ..., s_m'\}$; $A' = \{\alpha_1', \alpha_2', ..., \alpha_m'\}$; $P' = \{p_1', p_2', ..., p_m'\}$; $\mathcal{L}'$ = $\{\mathcal{L}_1', \mathcal{L}_2', ..., \mathcal{L}_m'\}$; The beam search width W_b;
Output: Angles with collision-free $\hat{A}^* = \{\alpha_1^*, \alpha_2^*, ..., \alpha_m^*\}$

A nodes vector C of the current depth of beam search
A nodes vector B of the previous depth of beam search
$\mathbb{P} \leftarrow \emptyset$; $\mathbb{C} \leftarrow \emptyset$; $\hat{A}^* \leftarrow \emptyset$
$S^* \leftarrow s_0'$; $A^* \leftarrow \emptyset$; $P^* \leftarrow \emptyset$;
if check_collision_free(S^*, A^*, P^*) **then**
  node $n \leftarrow \{S^*, A^*, P^*\}$
  $\mathbb{P} \leftarrow \mathbb{P} \cup n$
else
  return $\emptyset$
$i \leftarrow 1$ //iteration of beam search, the height of search tree
while $\mathbb{P} \neq \emptyset$ **do**
  for each node n of $\mathbb{P}$ **do**
    get S^*, A^*, P^* of n;
    $S^* \leftarrow S^* \cup s_i'$, $A^* \cup \alpha_i'$, $P_{temp}^* \leftarrow P^*$
    if check_collision_free(S^*, A^*, P^*) **then**
      $\omega \leftarrow rand()mod100 \div 100$
      if probability ω < 0.3 **then**
        //tuned point situation
        set p_i^* as possible tuned point;
        $P_{temp}^* \leftarrow P_{temp}^* \cup p_i^*$
        node $n' \leftarrow \{S^*, A^*, P_{temp}^*\}$
        $\mathbb{C} \leftarrow \mathbb{P} \cup n'$
      //constant point situation
      set p_i^* as constant point;
      $P_{temp}^* \leftarrow P_{temp}^* \cup p_i^*$
      node $n' \leftarrow \{S^*, A^*, P_{temp}^*\}$
      $\mathbb{C} \leftarrow \mathbb{P} \cup n'$
    else
      $A_{temp}^* \leftarrow \emptyset$
      //tuned point situation
      set p_i^* as tuned point;
      $P_{temp}^* \leftarrow P_{temp}^* \cup p_i^*$
      $A_{temp}^* \leftarrow$ CRO (S^*, A^*, P^*)
      if $A^* \neq \emptyset$ **then**
        node $n' \leftarrow \{S^*, A_{temp}^*, P_{temp}^*\}$
        $\mathbb{C} \leftarrow \mathbb{P} \cup n'$
      //constant point situation
      set p_i^* as constant point;
      $P_{temp}^* \leftarrow P_{temp}^* \cup p_i^*$
      $A_{temp}^* \leftarrow$ CRO (S^*, A^*, P^*)
      if $A^* \neq \emptyset$ **then**
        node $n' \leftarrow \{S^*, A_{temp}^*, P_{temp}^*\}$
        $\mathbb{C} \leftarrow \mathbb{P} \cup n'$
  if $\mathbb{C} = \emptyset$ **then**
    return $\emptyset$
  for each node n of $\mathbb{C}$ **do**
    scoring and sorting $\mathbb{C}$
  $\mathbb{P} \leftarrow$ the first W_b nodes of $\mathbb{C}$, $\mathbb{C} \leftarrow \emptyset$
  $i \leftarrow i + 1$
  if i = m **then**
    $\hat{A}^* \leftarrow$ the first A^* of node of $\mathbb{P}$
    return $\hat{A}^*$

O DISCUSSION ON BEAM SEARCH

Our beam search algorithm is not exhaustive but guarantees finding a feasible solution. Beam search prunes many paths and only keeps a limited number of best paths at each level, making it non-exhaustive, which makes it significantly faster than exhaustive search algorithms. We employ beam search to discover solutions with the minimal number of tuned points. By adjusting the beam width, we aim to approach the optimal solution, although we cannot guarantee that it is optimal. When beam search fails to find a solution (Figure 16 in paper), we restart fitting and searching with smaller segments to ensure eventual discovery of a solution. Splitting the circular segments of Figure 16 into shorter segments results in more points becoming tuned point candidates. In the worst-case scenario, all points are tuned points, indicating segments fully bent by human hands. Therefore, our algorithm guarantees finding a feasible solution.

P MULTI-VIEW EXAMPLE TEST

Multi-view wire art can be interpreted in various ways depending on the viewer's perspective. Previous works, [Hsiao et al. 2018] and [Tojo et al. 2024], can generate multi-view wires with high geometric complexity and artistic appeal. We tested five examples, and the results demonstrate that our algorithm effectively addresses the challenges of multi-view wire manufacturing, as shown in Figure 7.

Q COMPLEX RESULTS

We conducted experiments on thirteen more complex 2D shapes, characterized by increased self-intersections and an enhanced artistic quality, typically achievable only through the artist's skill. The results demonstrate that our algorithm can effectively handle these intricate forms, as shown in Figure 8.

REFERENCES

Dejan. 2018. Arduino 3D Wire Bending Machine. (2018). https://howtomechatronics.com/projects/arduino-3d-wire-bending-machine/?utm_content=cmp-true

Andreas Fabri and Sylvain Pion. 2009. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 538–539.

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

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

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

Kenji Tojo, Ariel Shamir, Bernd Bickel, and Nobuyuki Umetani. 2024. Fabricable 3D Wire Art. In *ACM SIGGRAPH 2024 Conference Proceedings (SIGGRAPH '24)*. https://doi.org/10.1145/3641519.3657453

Andreas Waechter, Carl Laird, et al. 2002. Ipopt: Interior Point OPTimizer. https://coin-or.github.io/Ipopt/. (2002). Accessed: 2024-05-14.

Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2016. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

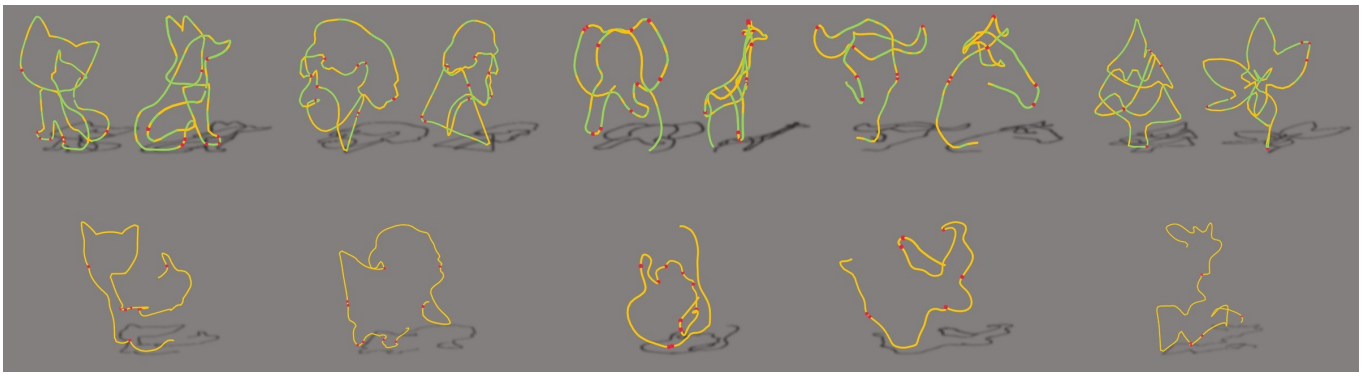

Fig. 7. Results of multi-view wire arts generated by [Tojo et al. 2024]. Top-left: View 1. Top-right: View 2. Bottom: tuned wire. The models are arranged in the order of *Cat-dog*, *Einstein-Newton*, *Elephant-giraffe*, *Bull-horse*, *Tree-flower*.

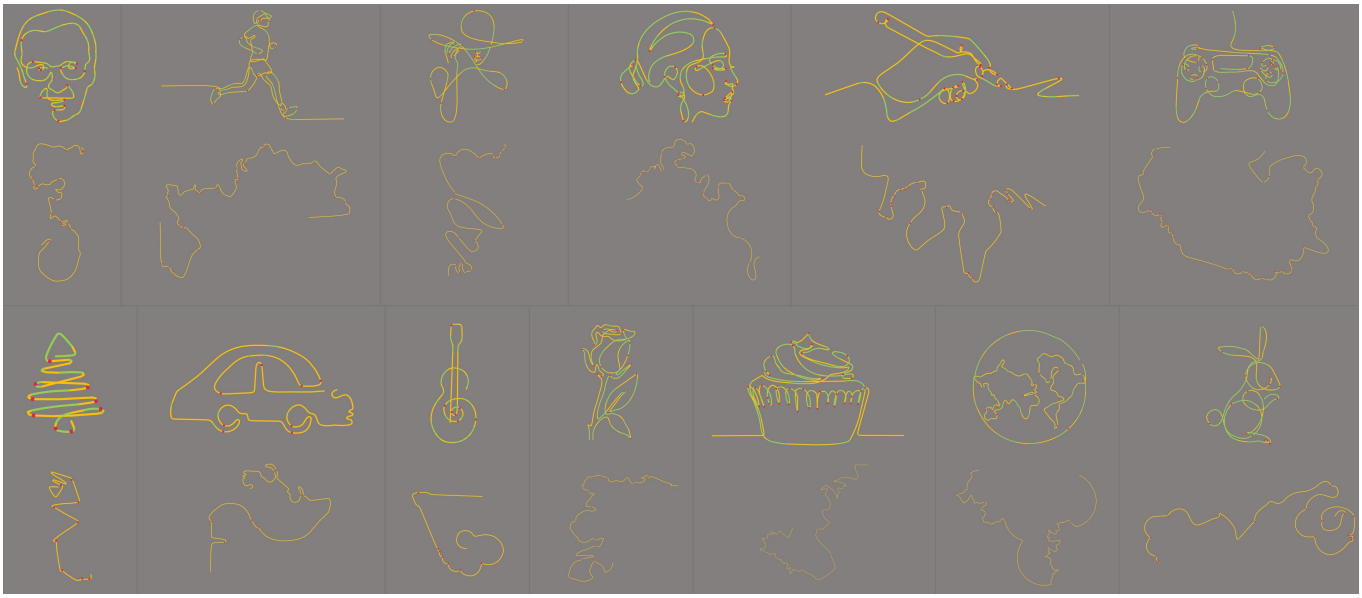

Fig. 8. Thirteen results of 2d wire art examples. The models are arranged in the order of *Man*, *Runner*, *Hat woman*, *Side face woman*, *Writing*, *Game player*, *Tree*, *Car*, *Guitar*, *Flower*, *Cake*, *World map*, *Rabbit*. We show both the fitting result (line segment: Green, circular segment: Yellow) and the tuned wire.