학 사 학 위 논 문
Bachelor's Thesis

A Computational Approach for Spider Web Inspired Fabrication

2018

Abileva, Yekaterina
한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

학 사 학 위 논 문

2018

한 국 과 학 기 술 원

전산학부

Abileva, Yekaterina

위 논문은 한국과학기술원 학사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2018년 6월 11일

심사위원장 Sue Bok Moon (인)

심 사 위 원 Jean-Charles Bazin (인)

심 사 위 원 Andrea Bianchi (인)

A Computational Approach for Spider Web Inspired Fabrication

Yekaterina Abileva

Advisor: Sue Bok Moon

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science

Daejeon, Korea
June 11, 2018

Approved by

Sue Bok Moon
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

BCS 20140940

Yekaterina Abileva. A Computational Approach for Spider Web Inspired Fabrication. School of Computing . 2018. 23+ii pages. Advisor: Sue Bok Moon. (Text in English)

Abstract

Some species of spiders can build intriguing and sophisticated web structures using just a single continuous thread. Inspired by this spiders' ability, we investigate how to automatically fabricate planar objects from a single thread in such a way that it looks like an input image provided by the user. Creating objects with threads is commonly referred as string art in the context of art pieces creation. This is typically a manual, tedious work reserved for skilled artists. To allow casual users for creating their personalized thread objects from their own images, we propose a fully automatic system composed of hardware and software components. This requires to solve several challenges, such as the actual fabrication and the optimization of the thread connections. We experimentally demonstrate that our system can create visually appealing results and be applied for various kinds of input image contents and types, such as human face photos, paintings, animals, buildings, logo drawings and text.

Keywords Computer-aided design, Fabrication, Fabrication and Material Design, Entertainment

Contents

List of Figures

Chapter 1. Introduction

Some species of spiders can build intriguing and sophisticated web structures using just a single continuous thread. For example Miagrammopes spiders in the family of Uloboridae can build webs consisting of as few as one sticky thread [15]. Beyond the intriguing visual aspect of such webs, this type of web construction can be explained by biological evolution: it has been shown that single-thread traps catch significantly more preys per thread than do multiple-thread traps [15]. Inspired by this spider's ability of building sophisticated web structures using just a single thread, we investigate how to automatically fabricate planar objects from a single continuous thread (no cut, no glue) in such a way that it looks like an input image provided by the user. For example, given a photo of a face, the goal is to create a web-like object made of a single thread and resembling the input face (see Figure 1.1) .

As a first step towards this challenging goal, in this paper we focus on planar object fabrication, just like spider webs. In the planar configuration, the fabrication is performed by arranging a single thread between pins located on the boundary of a planar support, such as a circle. The input is an image provided by the user, such as a photo, drawing or painting, see fig:teaser. We then run the proposed thread connection optimization algorithm to find in which order the thread must be connected to the pins so that the simulation result of the optimized thread connection looks like the input image, see rendered image in fig:teaser. We then fabricate the spider web-like thread object by physically connecting the thread to the pins using our custom-made machine, see fig:teaser. The output is the fabricated version of the input image (see photo of the fabricated object in fig:teaser.

Our work is along the lines of several recent works on novel, unconventional, inspiring computational fabrication of objects and art creation, such as knitting [18], thermoforming [25], stippled prints with drone [8] [9], perforated lampshades [33], printing with magnetic flakes [23], spray painting [17], airbrush system [27], balloons [28], inflatable structures [29] and kites [24]. We propose a computational approach that automatically optimizes the thread connection and manufactures the object with a custom-made machine.

Creating objects with threads is commonly referred as string art (or spirelli) in the context of art pieces creation. Creating string art pieces is typically a manual, tedious work. While casual people can manually design and fabricate simple basic shapes, complex patterns can only be crafted by skilled artists, such as the artist Petros Vrellis[1]. Designing the thread pattern is a particularly challenging task because it requires to decide how to connect the thread, i.e. connect which pin to which pin and which order. Moreover the manual fabrication itself can take up to numerous hours or even days depending on the complexity of the pattern and the number of thread connections. In contrast, our system works in a fully automatic manner, from thread connection optimization to fabrication, and thus can be used by casual users with no prior experience in string art.

Our paper and the recent string art work of the artist Petros Vrellis[1] have one goal in common: optimization of the thread connections using algorithm. Both works have been conducted independently and we were not aware of Vrellis' work when starting this project. In addition, his exact approach is not described[2]. Moreover, in his work, the pins are connected by hand, which might take numerous hours or even days. In contrast, we provide a full description of our thread optimization algorithm and also

[1] http://artof01.com/vrellis/

[2] We contacted him by email but did not get information about his exact approach unfortunately.

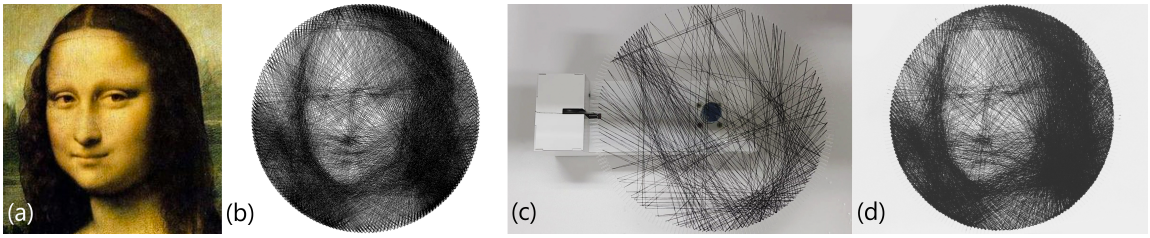


Figure 1.1: Inspired by some spiders species that can build intriguing web structures from a single continuous thread, we propose an approach that can automatically fabricate planar objects from one single thread in such a way that it looks like an input photo provided by the user. (a): input photo, (b): simulation result obtained by the proposed thread connection optimization, (c): our custom-made fabrication machine to connect the thread according to this optimized set connections, (d): photo of the fabricated object. Both the thread connection optimization and the actual fabrication run in a fully automatic manner. The fabricated object has a diameter of 36cm, is made of one continuous thread (no cut, no glue) and the total thread length is around 200 meters.

present a custom-made fabrication machine that can automatically connect the thread. We also describe our machine in detail and provide all the hardware specifications and component models. In addition, we present and compare different strategies for the thread connection optimization.

We propose a fully automatic system composed of hardware and software components. This requires to solve several challenges, such as the manufacturing itself and the optimization of the thread connections. For example a naive approach for the thread optimization is intractable due to combinatorial explosion. If the fabrication support has K pins on the boundary, there are K pin possible choices for each thread connection. For an object composed of N thread connections (i.e. N apparent "lines"), this leads to a total of K^N possible combinations. Assuming typical values of $K = 200$ pins and $N = 1000$ lines, and as low as just 1ms per combination evaluation, this would still require in the order of 10^{12} years! Additional challenges include the mathematical formulation of the cost to minimize for the thread connection optimization, as well as taking into account the physical/mechanical constraints of the machine when optimizing the thread connections.

Our two main contributions are as follows:

- The first full description of an automatic fabrication system using one single continuous thread (no cut, no glue). We describe both the software (e.g. thread optimization) and hardware (e.g. our custom-made fabrication machine) components in detail.
- We introduce and compare different thread optimization methods, including physical constraint-aware algorithms to deal with the mechanical constraints of the machine.

We experimentally demonstrate that our system can create visually appealing results and be applied for various kinds of image contents and types, such as face photo, paintings, animals, buildings, logo drawings and text. In summary, our system allows casual users for creating their personalized recreational string art from their own photos in a fully automatic manner.

Chapter 2. Related Works and Contributions of this Dissertation

Our work is related to several categories of work such as fabrication, computational art creation and image generation. We now discuss the most related methods in these areas.

Fabrication Several exciting works recently investigated computational fabrication, for example knitting [18], thermoforming [25], creation of perforated lampshades [33], balloons [28], inflatable structures [29], tile decors [**?**] and kite [24]. Our work belongs to this line of works on computational fabrication and we present a fully automatic system that can fabricate spider web-inspired thread patterns.

Computational art creation and image generation Computational methods have also been proposed for the design and generation of recreational arts and images in various contexts such as pixel art [13], emerging images [19], camouflage images [19], shadow art [20], collage art [7] [10], text art [32] [16], spray painting [17], and airbrush [27], among many others. While some of these references are relevant to our goal, there are two main differences with our work. First, we consider a single continuous thread, which requires a dedicated thread connection optimization algorithm. Second, since we aim to fabricate the object, we need to design a specific custom-made machine and also take its physical constraints into account for the thread optimization.

Dithering and halftoning Generation of images with specific patterns has been studied in the contexts of dithering, halftoning, or stippling. For example halftoning is the process of generating a pattern of binary pixels that creates the illusion of a continuous-tone image [22]. Closely related is stippling where dots are drawn, both digitally [5][11] or physically [8][9], in such a way that it resembles a target input photo. Another interesting work is the generation of QR code in such a way that it looks similar to an input image [**?**]. These works consider a discrete set of dots or pixels that can be individually controlled, optimized and drawn. In contrast, we consider one single continuous physical thread, which requires dedicated methods on the software (optimization) and hardware (fabrication) sides.

Chapter 3. Proposed Approach

In our setup, we consider a circular support, a set of N pins (typically 200) uniformly distributed on the boundary of this support, as well as one continuous thread of a single color. An example is shown in Figure 1.1. To automatically fabricate the spider web-like objects resembling the user-provided target photo, we need to achieve two main goals. The first one is to compute the thread connection optimization (i.e. connect from which pin to which pin and in which order) so that the resemblance to the input target photo is maximized. For this we will introduce and compare two different strategies for the thread optimization. We refer to this as the software component (Section 3.1). The second goal is the actual fabrication of the thread object, that is physically connecting the thread to the pins, based on the optimized thread combination. We refer to this as the hardware component and we will present our custom-made SpiderPrinter machine (Section 3.2).

3.1 Hardware

SpiderPrinter We introduce our fabrication machine, named SpiderPrinter, capable of fabricating thread objects automatically. SpiderPrinter is a custom-made turn-table machine that dispenses a single thread around a circular transparent plate (Figure 3.1(a)). The plate contains pins protruding from its circumference, with every two adjacent pins forming a V cut (Figure 3.1(b)). SpiderPrinter operates through the synchronized motions of two motors: first, a servomotor guides the thread to the top or bottom side of the plate, and second, a stepper motor rotates the plate, effectively pulling the thread from one point of the circumference to another, forming a line. Our proposed fabrication is composed of a sequence of lines placed alternatively on the top or bottom side of the plate. Because the plate is transparent, the lines on both layers equally contribute to the final result. Compared, for example to winding the thread around a pin, the main advantage of our two-layer approach is its simplicity and efficiency. For example winding the thread around a pin would result in longer fabrication times, due to the more complex motion path for the motors, and would require more complex mechanisms (higher degrees of freedom) as well as challenging control. In contrast, our approach simply operates a sequence of in-plane rotation and up/down motion. An additional advantage compared to winding is that it ensures that two consecutive lines always share exactly the same point on the circumference (i.e. the bottom of the V cut), hence increasing the precision and look of the final output. In our design the plate is directly attached to the stepper motor's shaft with an adapter (Figure 3.1(a)), so that a thread on the bottom layer cannot possibly pass through the plate's center. We will show that this physical constraint can simply be enforced in the thread connection optimization.

SpiderPrinter uses a circular 2T acrylic plate with a diameter of 36 cm, that contains 200 equidistant pins protruding from its circumference (length: 12 mm, thickness: 1.5 mm), with every two adjacent pins forming a V cut (Figure 3.1(b)). The distance between two pins is 2 mm (1.8°), which is large enough to comfortably allow the passage of a small PolyLactic Acid (PLA) 3D printed arm (thickness: 1 mm), attached to the servomotor. The thread we used is a spun polyester thread 60S/3 (approximately 0.14 mm thick). To operate the machine, the user simply takes a new acrylic plate and screws it to the adapters on the top of the stepper motor's shaft. After inserting a new spool of thread on the back of the machine, the user manually inserts the thread to a small hole in the servomotor's arm (Figure 3.1(c)).

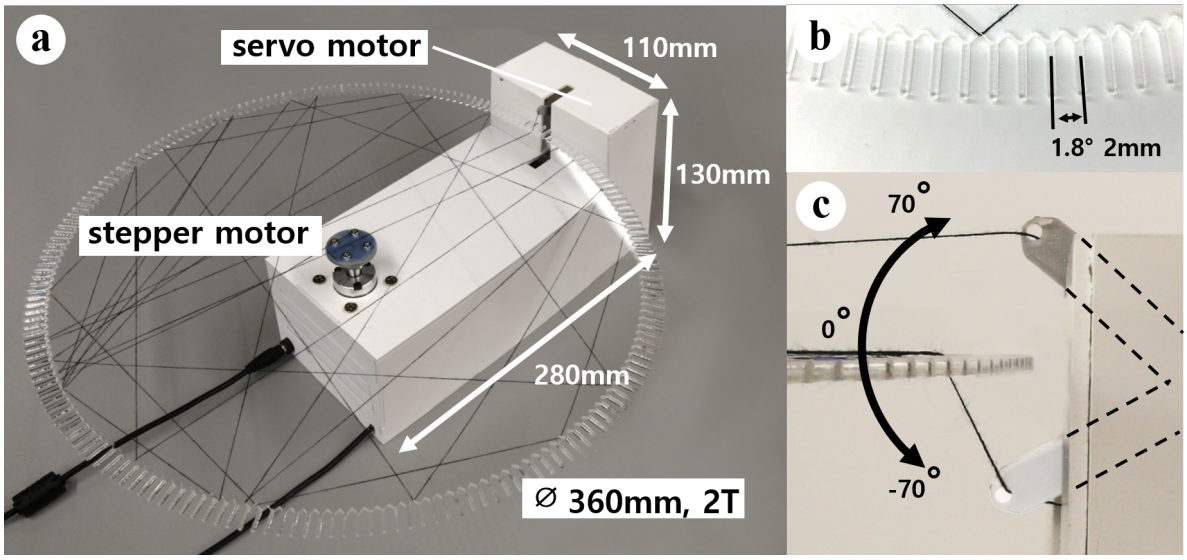


Figure 3.1: Overview of the SpiderPrinter (a), top view of the acrylic plate's pins used for the fabrication (b), and side view of the servo-arm motion from the top to the bottom layer of the acrylic plate (c).

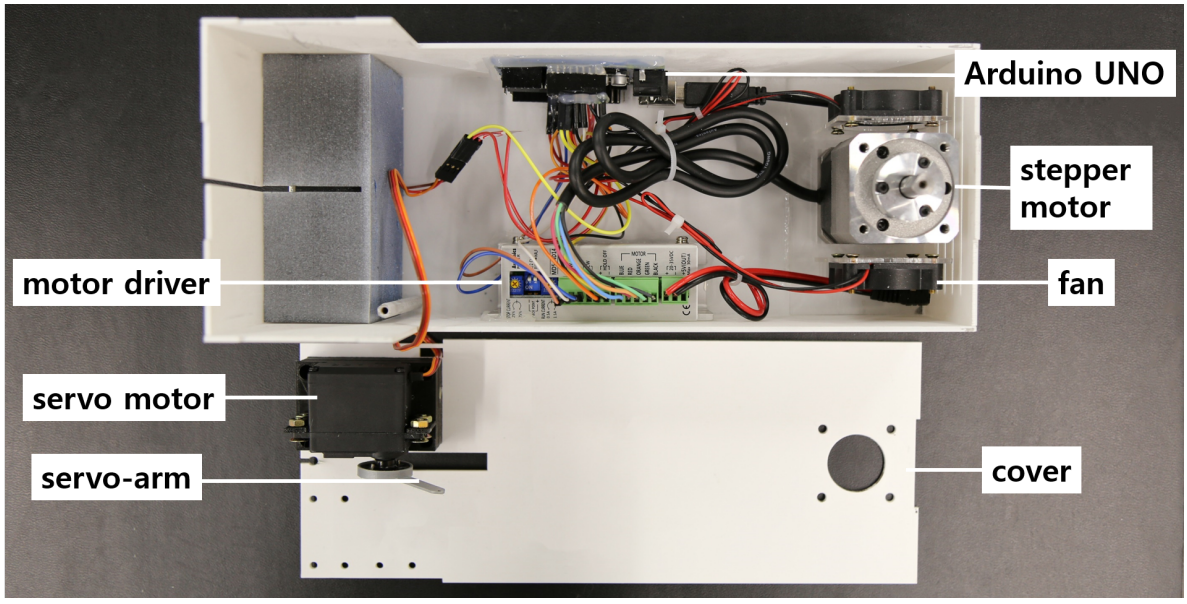


Figure 3.2: Inside view of our SpiderPrinter.

Then she manually pulls the wire and attach it to an arbitrary pin. Finally she can launch the fabrication by just clicking a "start" button in a custom Java program that controls the motors, and the fabrication will run automatically until completion.

Hardware specifications A detailed view of the SpiderPrinter hardware is presented in Figure 3.2. SpiderPrinter consists of a 280 x 110 x 130 mm acrylic box, enclosing a structure built by gluing together multiple 2 mm laser-cut acrylic sheets. The structure holds together several electro-mechanical components at fixed distances: a stepper motor, a micro-stepper driver, an analog servo motor, two cooling fans, a thread dispenser, and an Arduino UNO controlling board connected through USB to a PC. The acrylic plate is screwed on a 3D printed 30mm radius adapter made of PolyLactic Acid (PLA), which is fixed on the top of the shaft of a 5-phase stepping motor (A15K-S545-G10 with 0.75A/Phase) capable of 15 kgf-cm torque. The micro-stepper controller (MD5-ND14) drives the stepper motor with a 0.072° accuracy per step. Two small fans (SZH-GNP511, 5V at 0.14A, 6000 RPM) are attached next to the base of the stepper motor to provide cooling and ensure an operational temperature between 30 and 40°C. A 25 mm long PLA arm is attached to the shaft of a servo motor (FS5103B, 4.8V, 3kg.cm) and it is used to guide the thread through the V holes of the acrylic plate. The servo motor is mounted so that its shaft is coplanar with the plate and orthogonal to the arm's motion. The thread spool is placed on the back of the device and two small bearings ensure its fluid motion. The overall system is powered by a 24V power adapter connected to a wall socket.

3.2 Software

In this section, we will present our thread connection optimization. Before going into the details of the algorithm, let's first clarify the terminology. In the following, we use the term pins as a general term to refer to the elements where the thread is attached, such as nails hammered on a board or in our case the bottom of the V cuts in the plate (Figure 3.1(b)). We call line a part of the thread connected between two pins (see Figure 3.3(d)). We interchangeably use the terms thread combination, arrangement, connection, and occasionally line depending on the context. The user-provided image is called input image or target image.

We now present how to compute the thread arrangement, i.e. connect the thread from which pin to which pin and in which order. We will introduce two different strategies for the thread optimization and compare them. We consider black thread and white background. Therefore the absence of thread in a part of the image is seen as white. An important and interesting observation is that the perceived color tone depends on the arrangement of the threads, i.e. if many threads pass trough a certain part, then this part will be perceived as darker. This means that, even if the thread has a single black color, humans can perceive variations of gray depending on the thread arrangement. Therefore by optimizing the thread arrangement, it is possible to control the perceived color tone.

3.3 Preprocessing

In a pre-processing step, we convert the input image to a canonical form with a virtual circular layout (see Figure 3.3). This virtual circular layout represents the physical circular support and contains the same number K of virtual pins as on the physical circle. First we convert the input color image to grayscale, crop it to a square and resize it to 2000×2000 pixels. Then we set the virtual circle at the

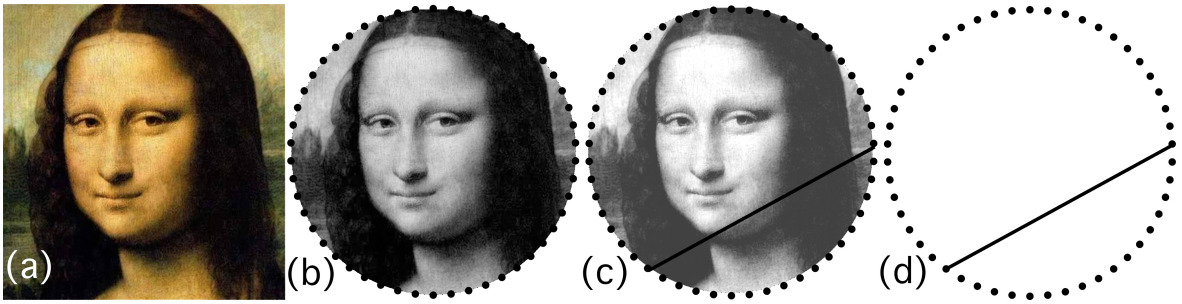


Figure 3.3: Preprocessing and thread connections. Given an input image (a), we convert it to a canonical virtual circular layout (b). A selected line on this virtual circle (c) is used to render the simulation result and corresponds to one line on the physical circle (d).

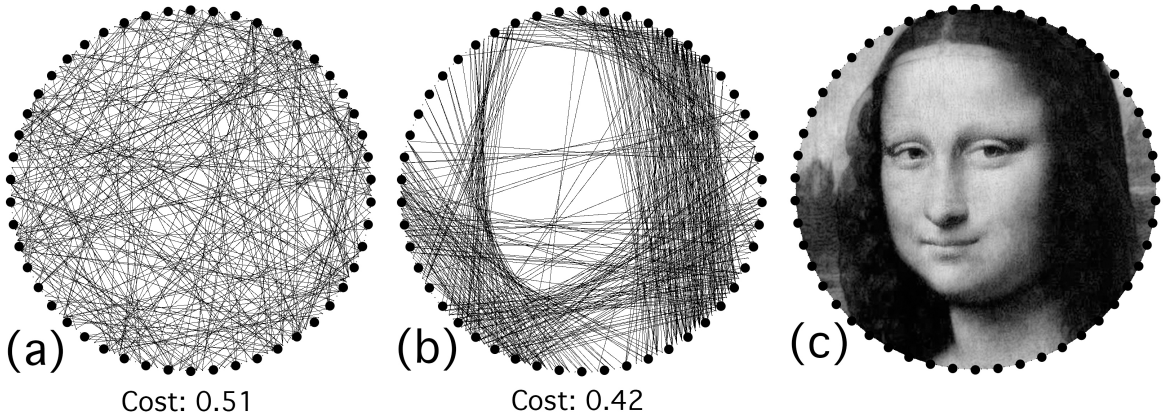


Figure 3.4: Examples of tone cost values for two rendered images (a,b) with the same number of lines compared to the target image (c).

center of the image, as shown in Figure 3.3(a). The visual content inside the circle is considered. Now the goal is to find how to connect the thread to the virtual pins.

3.4 General algorithm

We now introduce our general bottom-up optimization algorithm that iteratively adds lines (i.e. thread connection) one by one in such a way that the "resemblance" to the target photo gets increased (the resemblance will be formally defined later). It is general in the sense that it is not specifically tuned to a particular resemblance cost definition, and thus different cost definitions can be used. Given a set of lines, we can render an image composed of these lines, and we now want to compute a cost that compares this rendered image I_r with the target (input) image I_t. It is bottom-up because we start with an empty result (i.e. without any thread) and iteratively add thread connections. While the proposed algorithm takes local decision, experiments will show that it can provide visually satisfying results in a short amount of time (in a couple of seconds or minutes, depending on the number of lines).

The proposed thread optimization algorithm is shown in Algorithm 1. The input is the user-provided target image I_t. We first set the maximal number N of lines and the number K of pins. Typical values are $N = 3000$ lines and $K = 200$ pins, and all the results shown in this paper have been created with these values. The pins are uniformly distributed on the virtual circle boundary and can be represented

by an index from 1 to K. The output is the list of connections $\mathcal{L}$ containing the optimized pin-to-pin connections. For initialization (Line 2), $\mathcal{L}$ is set to the pin of index 1, without lack of generality. At each iteration (the outer for loop at Line 4), we search for the best line locally, i.e. the i-th iteration optimizes the i-th line of the object. At each of these iterations, we test the thread connections between the previous pin PrevPin to all the K pins (the inner for loop at Line 8), evaluate their cost (Line 10), and finally select the pin leading to the best cost (Line 11). The selected best next pin is then added to the list of pin connections (Line 21). The cost is obtained at Line 10 and can be computed in different ways, as will be discussed in Section 3.5. The function CheckAllowedConnection() at Line 9 tests if a pin connection is allowed. For example, if a connection between two pins is already in the list of selected lines, then this connection is not allowed. More examples of use cases will be provided below. Finally the procedure stops when a stopping criteria is reached. The general Algorithm 1 stops when the maximum number of lines has been reached or when no allowed combinations are found. Additional stopping criteria will also be discussed below.

ALGORITHM 1: Thread Combination Optimization

1: **Input:** target image I_t, maximum number of lines N, the number of pins K
2: **Initialization**: the combination list $\mathcal{L} = [1]$, i.e. the first pin (without lack of generality)
3: *//Find the next best line iteratively*
4: **for** i=1:N **do**
5: BestCost=∞, BestNextPin=$\emptyset$
6: PrevPin=$\mathcal{L}(i)$, i.e. the latest added pin
7: *//Test all the pin candidates*
8: **for** NextPin=1:K **do**
9: **if** CheckAllowedConnection(PrevPin,NextPin)==True **then**
10: cost=ComputeCost($\mathcal{L}$,PrevPin,NextPin)
11: **if** cost¡BestCost **then**
12: cost=BestCost
13: BestNextPin=NextPin
14: **end if**
15: **end if**
16: **end for**
17: **if** BestNextPin==$\emptyset$ *//i.e. no combination allowed* **then**
18: stop
19: **end if**
20: *//Update*
21: $\mathcal{L} = [\mathcal{L}$, BestNextPin] *//add the best pin to the combination list*
22: **end for**
23: **Return:** $\mathcal{L}$ (*i.e., the computed combination list*)

3.5 Cost

At Line 10 of Algorithm 1, the function ComputeScore evaluates the resemblance cost obtained by the set of optimized lines $\mathcal{L}$. In this section, we present how we computed this cost. Overall, the goal is to compute the resemblance between the target (input) photo I_t and the resulting rendered image I_r composed of the set of optimized lines $\mathcal{L}$. In the following, we present two kinds of cost we experimented

with: local scale and global scale.

Global scale Given a set of lines $\mathcal{L}$, we render an image composed of these lines. Considering black thread and white background, we draw the lines by line rasterization [31] in black over a white background image. We obtain the rendered image I_r. We define the global-scale cost as the tone cost between the rendered image I_r and the target (input) image I_t, where tone cost is computed as the pixelwise mean squared error (MSE) between the Gaussian-blurred versions of the two images:

$$C(\mathrm{I}_r, \mathrm{I}_t) = \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} \left(g(\mathrm{I}_r)_{x,y} - g(\mathrm{I}_t)_{x,y}\right)^2}{W \times H} \tag{3.1}$$

where $g(\mathrm{I})$ is the Gaussian-blurred version of the image I, $g(\mathrm{I})_{x,y}$ is the value of the blurred image $g(\mathrm{I})$ at the position (x, y), and H,W refer to the height and width of the image. In our implementation, we consider grayscale version of the images with values scaled between 0 and 1, and we employed a Gaussian kernel of size 15×15. The range of the tone cost is $[0, 1]$, with lower value indicates higher tone similarity. We use the difference of Gaussian-filtered images instead of the difference of images to model the human visual perception along distance [21][26] [3][1]. We call this cost the global scale cost because it considers information of the whole image: i.e. when we test a pin connection, we render the whole image composed of all the lines selected so far plus the currently tested pin, and we compare the whole rendered image to the whole input image by the tone cost of Eq. 3.1. We select the pin providing the lowest cost, then we add it to the list of lines, and we go for the next iteration/line.

In practice, while some implementation tricks can speed up the execution[1], this global scale defined at Eq. 3.1 is computationally expensive. Indeed, at each iteration of the for loop at Line 8 (pin evaluation), it requires to create the rendered image I_r (i.e. draw all the lines selected so fare, plus the current pin), apply a gaussian filter on the rendered image I_r, and compute the MSE of the tone cost. Overall, it is a computationally expensive process, around 8 seconds per line for a typical number of $K = 200$ pins, that is around 7 hours in total for typical 3000 line connections (see details in experiments section). In the following, we will propose a faster alternative cost that runs around 70 times faster (around 0.12 second per line, so around 6 minutes in total).

Local scale For the local scale cost, we use local information of the image. Contrary to the global scale approach, we start by rendering I_r using only the lines $\mathcal{L}$ selected so far, i.e. without the current line (PrevPin,NextPin), and its gaussian-fitlered version $g(\mathrm{I}_r)$. Therefore this can be done *outside* the for loop of the pin evaluation (Line 8), and thus is faster. Next, instead of computing the total MSE cost over the whole image (Eq. 3.1), we compute the local signed pixelwise tone difference:

$$D(\mathrm{I}_r, \mathrm{I}_t)_{x,y} = g(\mathrm{I}_r)_{x,y} - g(\mathrm{I}_t)_{x,y} \tag{3.2}$$

The output tone difference matrix D has the same size as the input image. Since it contains the signed difference, the range of every element of D is $[-1, 1]$. A positive value means that $g(\mathrm{I}_r)_{x,y} > g(\mathrm{I}_t)_{x,y}$, i.e. $g(\mathrm{I}_r)_{x,y}$ is brighter than $g(\mathrm{I}_t)_{x,y}$, and so should be darker to resemble the input image more. Reciprocally, a negative value means that $g(\mathrm{I}_r)_{x,y} < g(\mathrm{I}_t)_{x,y}$, i.e. $g(\mathrm{I}_r)_{x,y}$ is darker than $g(\mathrm{I}_t)_{x,y}$, and so should be brighter to resemble the input image more. Because our method can only add lines, we can only make an image location darker, not brighter.

[1] In practice, the image I_r does not have to be rendered from scratch when testing a new combination, i.e. at each iteration of the for loop at Line 8. Indeed, it is possible to just draw the current line (PrevPin,NextPin) on top of the image composed of the lines $\mathcal{L}$ obtained so far at the previous iteration i (Line 4). Alternative ways are also possible, such as "undrawing" the previously tested line and drawing the current line, which avoids creating a duplicate of the image.

Based on the above explanation, our aim is to find a line that will make the part darker. For this, we test all the pin combinations at Line 8 and select the one whose brightness is too bright compared to the input target photo so that we can make it darker. Concretely, to compute the brightness of a line, we obtain the coordinates of the pixels along this line by line rasterization [31], and compute the cost based on the values of these pixels in the difference matrix D. We compute the line cost as the average values (divided by the number of rasterized line pixels) to avoid indirectly favoring shorter segments). Finally we select the line with the highest cost, i.e. the line with the largest brightness difference.

In terms of implementation, as discussed above, the rendered image I_r and its gaussian-filtered version $g(\mathrm{I}_r)$ are composed of the lines obtained so far, at the previous iteration. Therefore the tone difference at Eq. 3.2 can be computed once per iteration i, i.e. outside the for loop of the pin tests (Line 8), and thus runs much faster than the global scale computation. The cost of a line is thus reduced to a sum of values along the line, which is computationally cheap.

We call this cost the local scale cost because we use the local information along the line. We select the pin providing the lowest line cost, then we add it to the list of lines, and we go for the next iteration/line.

3.6 Physical constraints-aware optimization

The thread optimization described so far allows any pins to be virtually connected to any pins. However our fabrication machine has physical constraints that we need to take into account during the thread optimization. First, as discussed in Section 3.1, we privileged designing a machine that does not wind around a pin in order to avoid complicated hardware mechanisms and control strategies. Our proposed alternative strategy is that the machine connects the thread by alternating top and bottom connections, i.e. the bottom layer contains the even connections and the top layer the odd connections. Since our plate is transparent, bottom and top layers are seamlessly combined and equally contribute to the final perceived result. An important practical aspect is that the threads on the bottom layer cannot pass through the plate's center because the plate's center is attached to the stepper motor's shaft with an adapter (see Figure 3.1(a)). Therefore the thread optimization must be adapted to deal with this physical constraint. It can be implemented by a general function that checks if a connection is allowed: the function CheckAllowedConnection(P_j,P_k) at Line 9 returns True if the combination between the pins of indices i and j is allowed, and returns False otherwise. For example, if the line defined by the pins P_j and P_k passes within 3cm of the plate's center (approximately the motor's shaft radius) and is on the bottom layer, then the function returns False and this connection will be skipped.

A second physical constraint is that two pins too close to each other should not be connected. In order to ensure that our machine can pull a thread between any two pins and hold it firmly in place, we experimentally found that the minimum distance between two consecutive pins should be no smaller than 5degrees. The function CheckAllowedConnection() can simply deal with this case by returning False when the distance between two pins is shorter than a threshold set to 5°.

Chapter 4. Results

4.1 Experiments

4.1.1 Implementation and hardware details

Our algorithm has been implemented in Python (non-optimized code) and runs on a laptop equipped with an Intel i7-4790K 4.0GHz CPU and 32GB RAM. By default, we are using a circular layout and 200 pins. Given the input photo provided by the user, the program automatically computes and returns the optimized pin combination (i.e. connecting which pin to which pin, and in which order) as well as a visualization of this pin combination (i.e. the rendered image I_r). The duration of the pin optimization varies greatly with the choice of the cost function. It typically takes around 5-8 hours for the global cost and just 4-6 minutes for the local cost, depending on the complexity of the input photo and the number of pins. The fabrication takes between 1 and 3 hours, depending on the number of lines, with our current machine. If fabrication time is an issue, the fabrication can be drastically sped up with faster motors and encoders. The total cost of the machine is around 120 USD: Arduino 15 USD, Motor driver 30 USD, Motor 50 USD, servo 20 USD, plastic cables, cover and paint 5 USD. The fabrication cost of one thread object is composed of the supporting acrylic plate and the thread, which is less than 10 USD in total.

4.1.2 Results

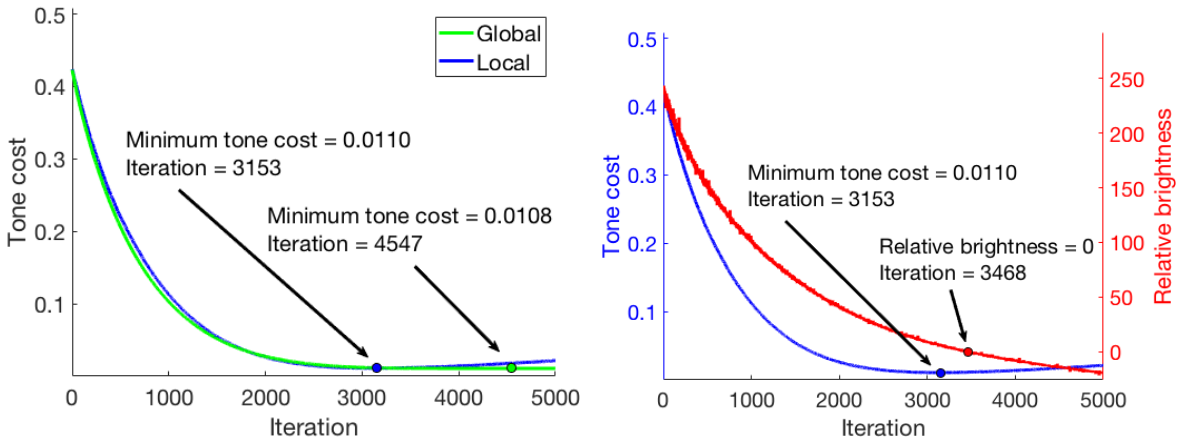
Figure 1.1 shows a representative result obtained by our approach. Given a photo of the Mona Lisa painting, our optimization algorithm computed a set of thread connections that we then fabricated with our machine. The photo of the fabricated object is visually satisfying, resembles the input image and contains recognizable face features of Mona Lisa.

In Section 3.2, we proposed two optimization costs: global and local cost. Before showing more fabrication results, we will compare and evaluate these two costs, pick up the one leading to the most satisfying results, and then show additional fabrication results obtained with this cost.

Quantitative evaluation

We will now quantitatively evaluate and compare the global and local costs. While the local cost is used to select the line, we can still compute the target global tone cost (after the line is selected). Therefore we will first compare the two costs in terms of global tone cost. Given a photo of the Mona Lisa painting, Figure 4.1(a) shows the evolution of the global tone cost where the connections are selected by the global and local cost. The local cost reaches a minimum of 0.0110 of tone cost after 3153 iterations and the global cost a minimum of 0.0108 of tone cost after 4547 iterations. It is interesting to see that while the local cost "indirectly" minimizes the global tone cost, its global tone cost is only about 2% more than the "direct" global cost method. Moreover, the pin optimization took around 6 hours for the global cost, but just around 4 minutes for the local cost. For completeness, Figure 4.1(b) shows the evolution of the brightness difference cost of the selected lines. Figure 4.2 shows the evolution of the thread result along the iterations of the algorithm for the local cost corresponding to Figure 4.1(a).

The visual results corresponding to the minimum costs of Figure 4.1(a) are shown in Figure 4.3.



(a) (b)

Figure 4.1: Visualization of the global tone cost evolution obtained by the local and global approaches. Iteration on the x-axis equivalently refers to the number of lines.

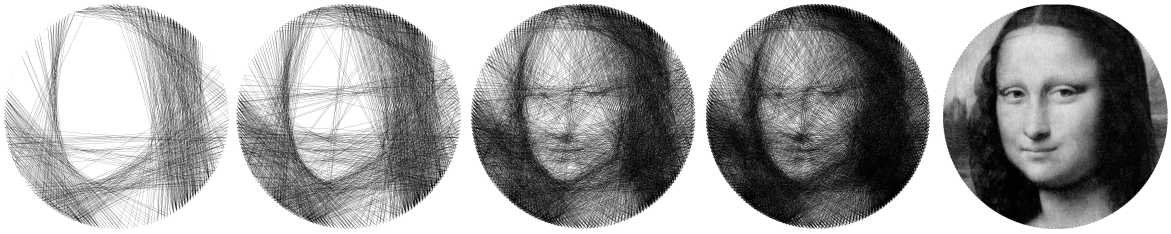


Figure 4.2: Evolution of the thread result at every 750 iterations of the algorithm for local cost, corresponding to the evolution of Figure 4.1(a). The target image is shown on the right.

We measure and compare the PSNR on these images. The PSNR values of the images produced by the local (Figure 4.3(b)) and global (Figure 4.3(c)) costs are respectively 7.51 and 7.64. The relative PSNR difference is just 1.7%, therefore we can say that the results obtained by the global and local costs are similar in terms of PSNR value.

Qualitative evaluation

In addition to quantitative evaluation, we also conducted a qualitative user study to compare the local and global costs, as well as to measure the overall visual quality of our results. We recruited 12 participants for the user study (3 females, 9 males, between 19 and 31 year old). None of the participants was involved in our work. The participants were shown the input photo as well as the thread result computed by either the global or local cost. For both costs, we showed the result when the minimum global tone cost is obtained (see Figure 4.1 and as discussed in Section 4.1.2). We asked the participants to evaluate the satisfaction statement "I am satisfied with the overall visual quality of the thread result" on a 5-point Likert scale: strongly disagree (1), disagree (2), neutral (3), agree (4) and strongly agree (5). We also asked them to evaluate the resemblance statement "The thread result resembles the input photo" on the same 5-point Likert scale. We conducted the study on all the input images of Figure 4.5. For each participant, the input images and the thread results obtained by both costs were shown in a random order. Since the perception of the thread results depends on the distance and eyesight of the participants, they were allowed to wear their prescription glasses, if any, and also adjust the distance to the thread results at the beginning of the user study. The distance adjusted by each participant was

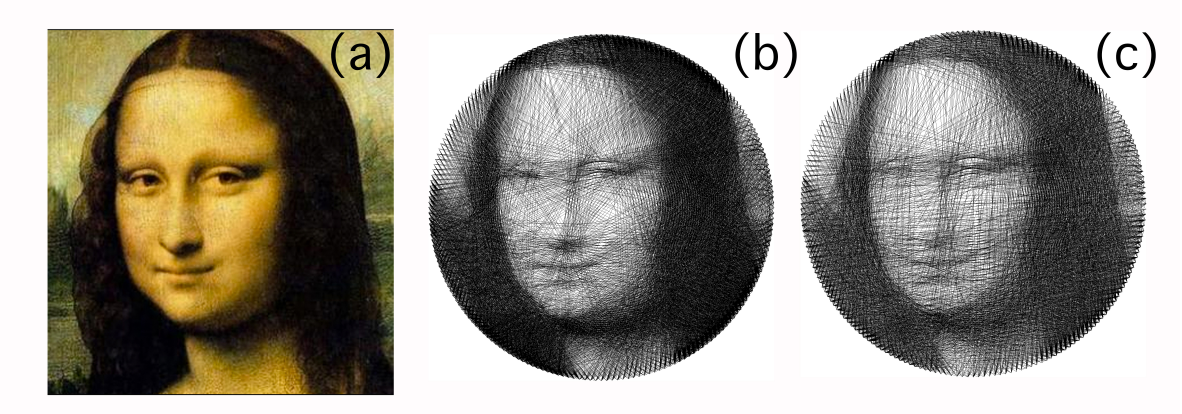


Figure 4.3: Given a photo of the Mona Lisa painting (a), visualization of the results obtained by the local (b) and global (c) costs, corresponding to the cost evaluation of Figure 4.1(a).

then fixed for the rest of the user study. Then, in practice, for each participant, we obtain a rate for each input image for the thread result obtained by the global and local cost.

The results of the user study are shown in Figure 4.4. In terms of satisfaction (Figure 4.4(a)), the ratings for the global and local costs are respectively as follows: 57% (global) vs. 68% (local) of the responses were 4 (agree) or higher, 86% vs. 85% were 3 (neutral) or higher. 14% (global) vs. 15% (local) of the responses were negative (disagree or less). On the whole, these ratings suggest that, first, both costs produce satisfying fabrication results overall and second, both costs lead to a similar level of satisfaction. Detailed statistical analysis will be provided below. To further evaluate the results, Figure 4.4(c) shows the evaluation score for each of the input images. It shows that the score range can be large for some input images (e.g. Van Gogh or Baby) and short for others (e.g. Mona Lisa or Mary). Globally, 8 out of the 8 input images for both global and local costs have a median score equal to or higher than 3 (neutral).

User study results for the resemblance to the input photo are available in Figure 4.4(b) and Figure 4.4(d). The ratings for the global and local costs are respectively as follows: 65% (global) vs. 68% (local) of the responses were 4 (agree) or higher, 88% vs. 88% were 3 (neutral) or higher. On the whole, these ratings suggest that both costs provide a satisfying level of resemblance to the input photo and second, both costs lead to a similar level of resemblance perception. Figure 4.4(d) shows the evaluation score for each of the input images. It shows that 8 out of the 8 input images for both global and local costs have a median score higher than 3 (neutral) or higher.

For a detailed statistical analysis, the user study results were analyzed with a repeated measure two-way analysis of variance (ANOVA), with a post-hoc pairwise comparison with Bonferroni correction (alpha 0.05). We also performed direct comparison using a two-tail t-test with alpha level of 0.05. Our results reveal that there was a statistical difference with the "thread" variable for similarity ($F(6,154)=45.0$, p¡0.01), and satisfaction ($F(6,154)=32.2$, p¡0.01). No statistical difference was found for the optimization method (global vs local) nor a significant interaction between variables was found. Post-hoc comparisons revealed that satisfaction and perceived similarity create three distinguished groups (500 threads, 1000 threads, all others) with score increasing with the number of threads. In general, perceived quality of the image increased with the number of threads, with no perceived differences above 1000 threads. To double check these results we also performed a t-test for satisfaction using as source material the users' satisfaction score for images generated using the number of of threads that minimized the cost of both

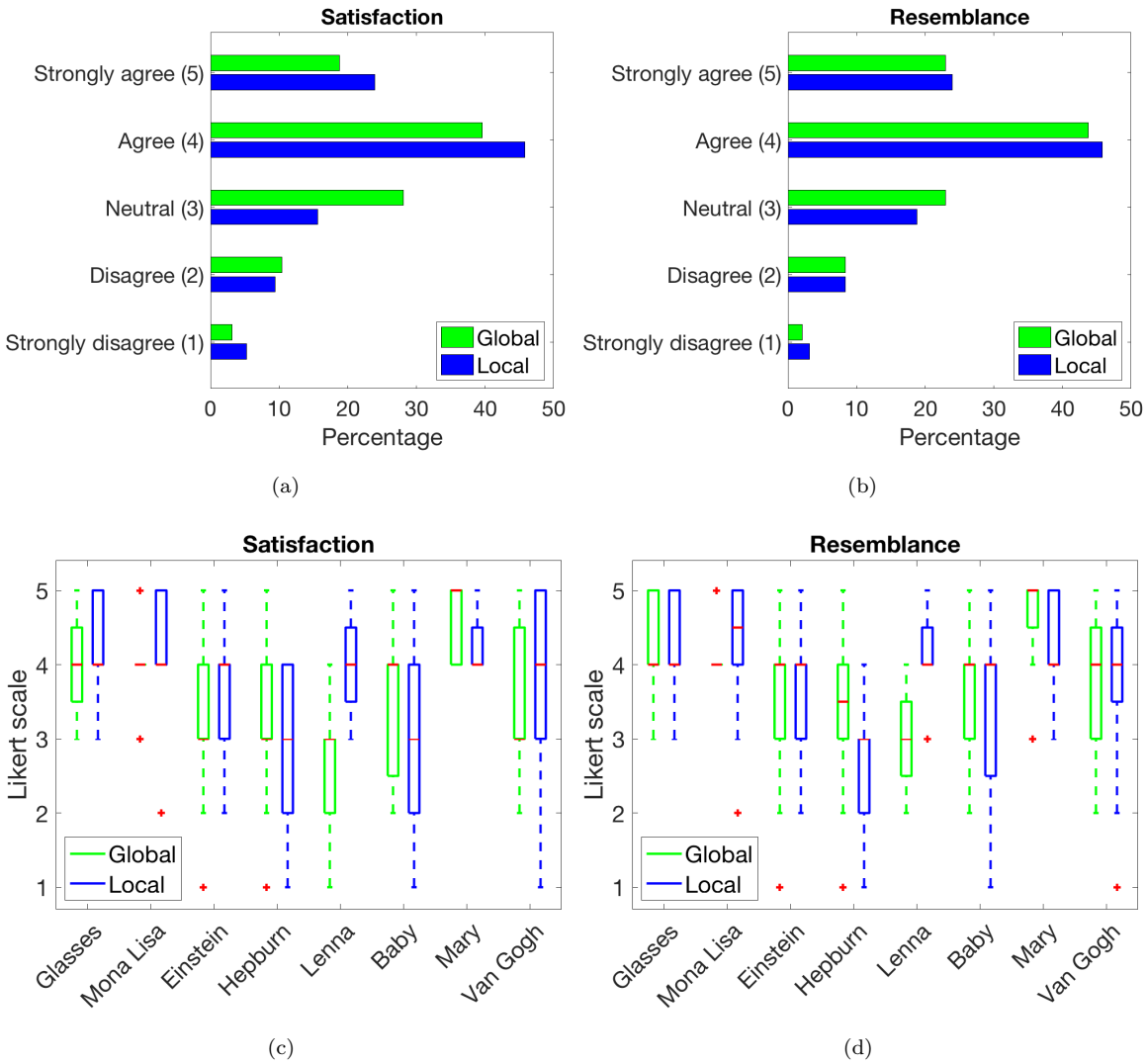


Figure 4.4: Plot of the user study results for the global and local strategies regarding satisfaction (left) and resemblance to the input photo (right), overall (top) and per input photo (bottom). The result per input photo is displayed by box plot where the median is drawn in red.

methods. The results revealed again no statistical differences, meaning that the two optimization methods (global vs local) are perceived as equivalent from the users. Finally, it is important to note that perceived satisfaction is highly correlated with the degree of similarly of the image with the source image (r=0.92, n=168, pj0.01).

In conclusion, given the cost values, the PSNR values, the optimization duration, the user study evaluation and its statistical analysis, we conclude that the local cost is a preferred option. Therefore in the following, we will show and further evaluate results obtained by the local cost.

Additional results

Figure 4.5 shows several representative results for various kinds of face appearance, in terms of input materials (photos vs. paintings with different artistic styles, e.g. expressionism for Van Gogh and mannerism for El Greco), head orientations (frontal vs. three quarter views), facial hair (none vs. beard), facial expressions (neutral vs. extreme with tongue stuck out), age (baby vs. adults), genders (male vs.

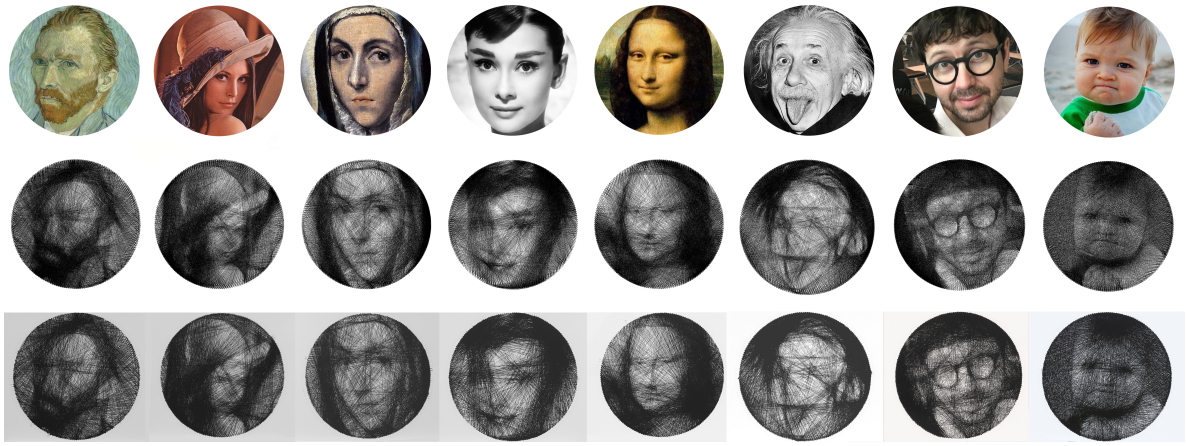


Figure 4.5: Representative results obtained by the proposed approach. Top: input image, Middle: rendering of the thread optimization, Bottom: photo of the fabricated object. From left to right: Van Gogh (self-portrait), Lenna (Lena Söderberg)[3], The Virgin Mary (by El Greco), Audrey Hepburn, Mona Lisa (by Leonardo da Vinci), Albert Einstein, man with glasses, and the success baby meme[4]. We referred to them as `Van Gogh`, `Lenna`, `Mary`, `Hepburn`, `Mona Lisa`, `Einstein`, `glasses` and `baby`.

female), and accessories (with/without glasses and hat).

Beyond faces Our approach can be used beyond human faces. For example, Figure 4.6 shows additional results for different kinds of input: the Eiffel Tower, tiger head and a flower.

Text Our approach can also be applied for text. For this, given an input text, we create an image version of the text. Figure 4.7 shows application of our approach for the fabrication of kids room name plate "Kate". Note how sharp the letter edges and boundaries are, despite using a single continuous thread.

Customized size Our approach can be applied on customized sizes. We used a default diameter of 30cm, see for example the results of Figure 4.5 and Figure 4.6. Our approach allows to modify the diameter by simply moving the location of the motor shaft. Figure 4.8 shows a representative fabrication result of earrings with a diameter of 5cm by our machine.

Multi-view Our approach can be extended for multi-view optimization in 3D where the sides of a 3D cube can show different images depending on the observed viewpoint, see Figure 4.9. In the original 2D approach (Algorithm 1), the input is one image and the pins are in a 2D plane. For the 3D multi-view extension, we consider two input images and the pins are located on the edges of a 3D cube. We assign these two input images on two non-facing sides of the cube, e.g. one image to the top side and one image to the left side. Assuming orthographic projection, we render the observation from one view by projecting all the threads along this viewing direction (i.e. normal direction of the cube side of the current view), which provides the render image. The multi-view cost is computed as the sum of the costs of the two views. We optimize the thread connections in 3D by measuring the cost of each pin combination (i.e. including pins at different depths) and pick up the one leading to the best cost. A result obtained by this approach is shown in the cube example of Figure 4.9. The cube is composed of 356 pins and its size is $10 \times 10 \times 10$cm. For this specific example, the thread was manually connected, as

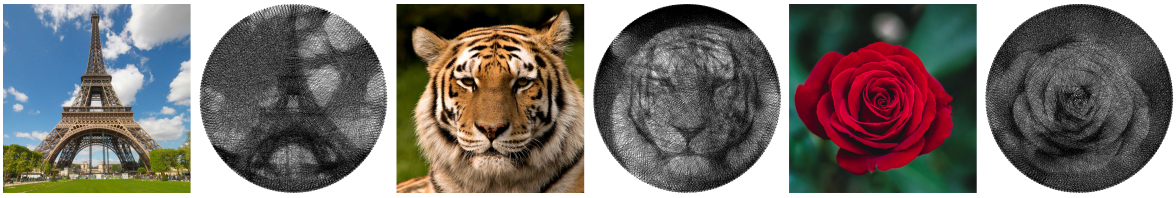


Figure 4.6: Our method can be applied on various image contents (animals, flowers, paintings, etc)

Figure 4.7: Application from text input for the fabrication of kids room name plate, here "Kate".

our current SpiderPrinter is designed for 2D patterns. The thread optimization took around 5 minutes and the manual fabrication with 420 lines took around 4 hours. When the cube is seen from a random direction, no image is visible, but a circle can be seen from the left view and a heart from the top view.

Colors Finally we also experimented with colors. Figure 4.10 shows an example of the ACM SIGGRAPH logo composed of two dominant colors blue and red. We used two long threads of these two colors. Our current SpiderPrinter prototype is designed for single thread, i.e. it is not designed to automatically switch between blue and red threads. That is why we printed color layer by layer, i.e. red thread layer and then blue thread layer. For this, first, we automatically extract the two dominant color clusters from the input image by K-means algorithm. We then decompose the input image into two images: one for the blue cluster and one for the red cluster. Then we compute the thread connection for

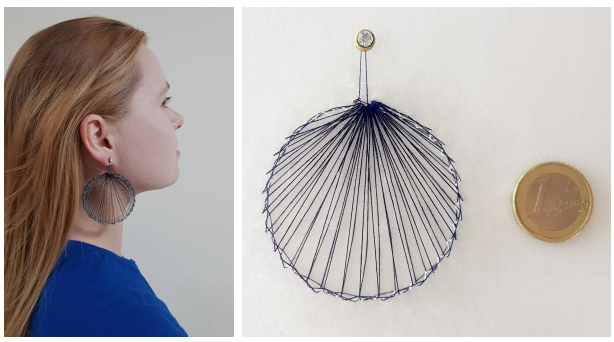


Figure 4.8: Fabrication of earrings with a radius of 5cm.

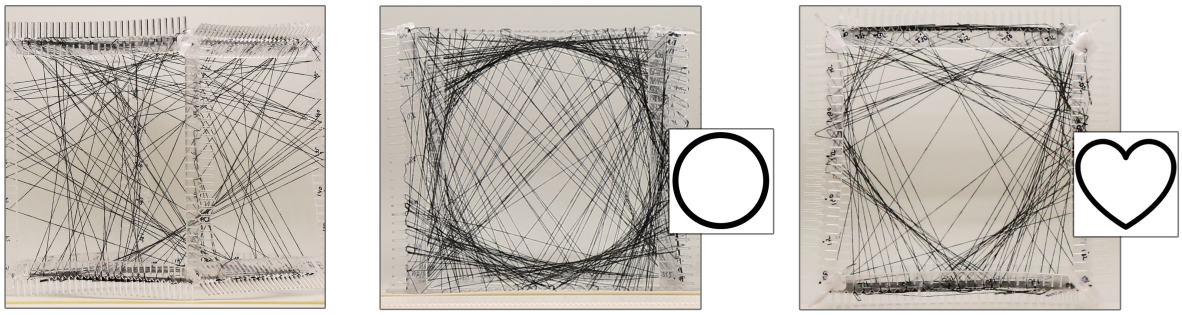


Figure 4.9: Extension to multi-view thread optimization in 3D where the sides of a cube shows different images. From left to right: photo of a tilted view of the cube with no apparent image, the left side of the cube shows a circle, and the top side shows a heart.

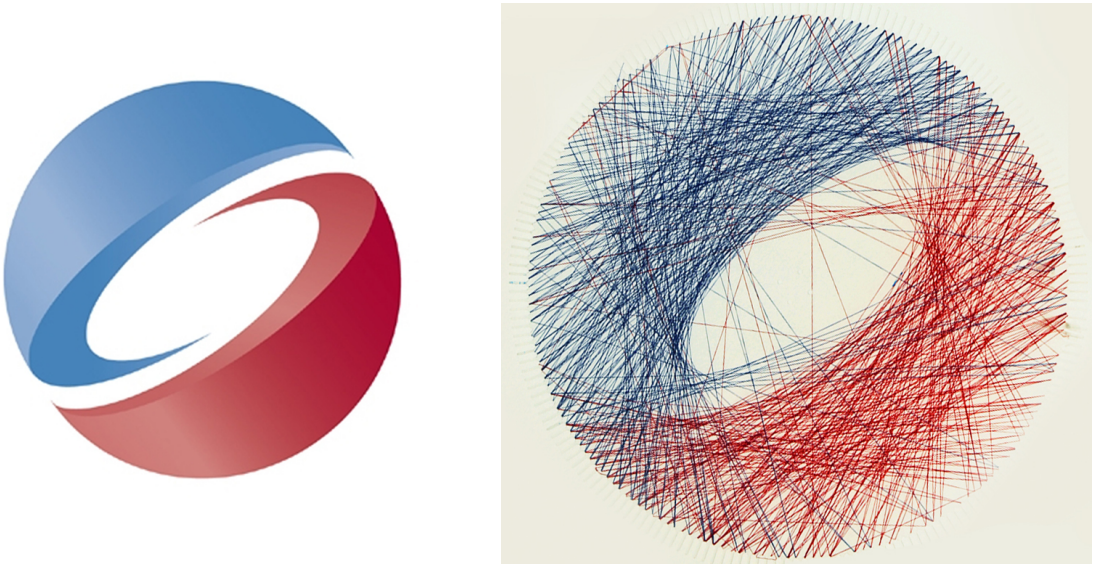
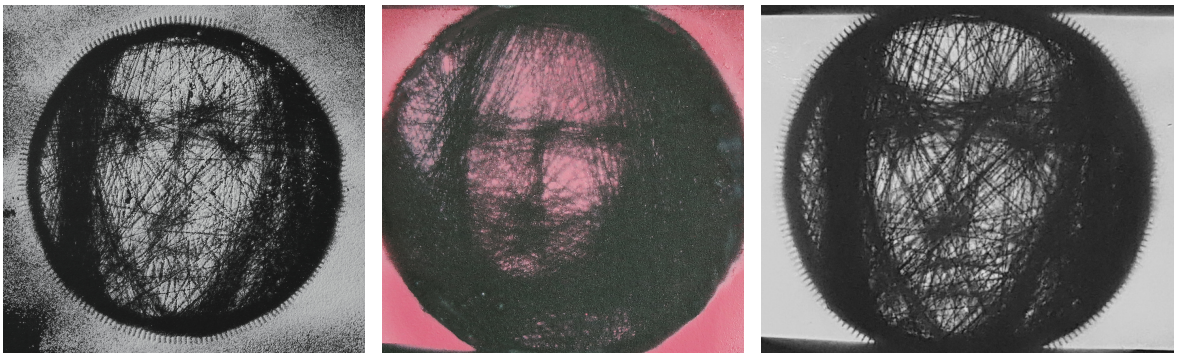


Figure 4.10: Fabrication result of the ACM SIGGRAPH logo with two colors. Left: input logo image, Right: photo of the fabricated object.

each of these two images independently. Lastly, for the fabrication, the machine connects the thread of the blue cluster, we manually change the blue thread spool to the red thread spool, and then the machine connects the thread of the red cluster. A photo of the final fabrication result is shown in Figure 4.10.

Stencil Our thread fabrication can also be used as a stencil for personalized decoration. For example Figure 4.11(a) was obtained by manually sprinkling sugar for personalized cake decoration, and Figure 4.11(c) was obtained by spray painting.

Bio-medical applications In collaboration with a research group in biology at *anonymous*, we also investigated the use of our thread optimization approach for bio-medical applications. For this, we asked an experienced biologist to manufacture 5 meters of bio-compatible threads following the method of Do et al. [6]. These bio-compatible threads are composed of chitosan and heparin, which are abundant in crustacean shell and animal cells, respectively. They can be easily produced by extracting a string-shaped polyelectrolyte complex at the interface of cationic chitosan and anionic heparin solutions [6]. Part of the manufacturing process is shown in Figure 4.12(a). Such bio-threads provide both flexibility and



(a) (b) (c)

Figure 4.11: Applications of our thread fabrication as a stencil with sugar for personalized cake decoration (a) and with spray painting (b).

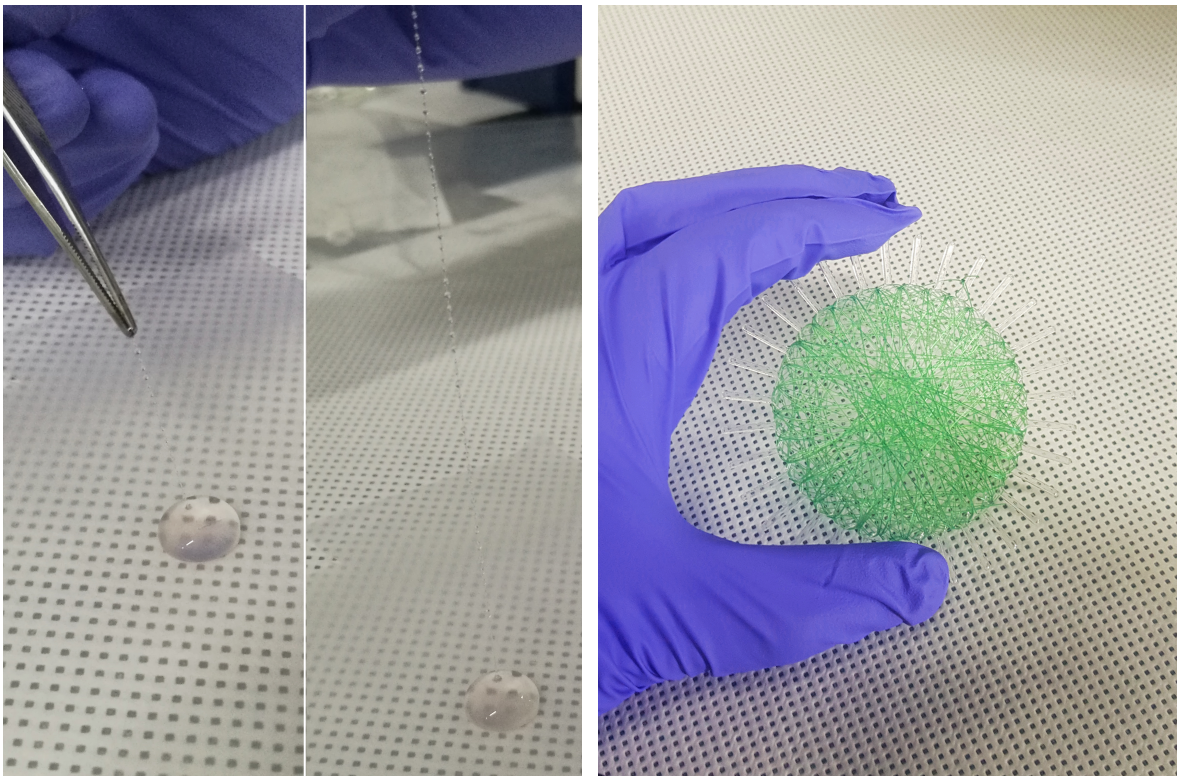
high mechanical strength. In addition, they are biologically safe because they are made from "natural" sources. Upon implanted in bodies, they can be naturally degraded by chemical/enzymatical reactions. Therefore these threads have shown to be useful for biomedical applications, such as medical sutures and wound dressings. The material cost is inexpensive, about 2 USD per meter. As a proof of concept, Figure 4.12(b) shows a photo of a pattern fabricated by our approach with bio-compatible thread. We showed our results and approach to three experienced biologists holding PhD and with 10+ years of experience. They commented our thread optimization and fabrication machine offer promising directions for the use of bio-threads in various medical applications such as patient-personalized pattern fabrication for bio-compatible functional scaffolds and new stem cell culture platforms.

4.1.3 Limitations and future work

In Section 3.2, we introduced a simple-yet-efficient thread optimization algorithm that provides visually satisfying results. In future work, we plan to investigate advanced optimization algorithms to further improve the quality of the results. In particular, we believe that reinforcement learning [30] has a great potential since it aims to maximize the cumulative long-term reward rather than an immediate short-term reward. The action performed by the agent is the selection of the next pin.

Our SpiderPrinter hardware presented in Section 3.1 can be extended in different directions. First, our current version allows to fabricate objects of different sizes, for example 30cm in Figure 1.1 and 5cm in Figure 4.8. However, smaller sizes pose the challenge of smaller distances between pins and thus need an accurate motor. This can be solved by selecting appropriate motors depending on the use case, which might increase the machine cost.

Our current machine allows the fabrication using a thread of a single color at a time. Concretely, for multiple colors, it requires the user to stop the process and manually substitute the thread spool, as discussed for the two-color fabrication of Figure 4.10. Future work will investigate the usage of hardware mechanism for fabrication of multi-color objects, such as multiple arms, automatic spool switching mechanism, or automatic colorization of the thread using ink or spray when the thread is passing by the servomotor's arm. Finally, an exciting direction for future work is to investigate an hardware mechanism to handle and rotate 3D surfaces to automatically fabricate 3D thread objects.



(a) (b)

Figure 4.12: (a) manufacturing the bio-compatible thread [6]. (b) photo of the fabricated pattern obtained by the proposed approach using a bio-compatible thread [6]. We colorized the bio-thread on purpose for a better visualization.

Chapter 5. Conclusion

Inspired by how some species of spiders can build intriguing and sophisticated web structures using a single continuous thread, we have presented a computational fabrication system to fabricate planar objects from a single thread in such a way that it looks like an input image. Our work is along the lines of several recent works on novel, unconventional, inspiring computational fabrication of objects and art creation. Our two key contributions are a simple-but-effective thread optimization algorithm and a custom-made fabrication machine that connects the thread pins. Our system runs in a fully automatic manner: the user provides an image, the thread connections are computed by our optimization algorithm and then the thread object is fabricated by our machine. We showed fabrication results obtained from different kinds of image contents and types, such as face photos, painting, animals, buildings, objects and logos. We also extended our approach to multi-view 3D fabrication as well as two-color fabrication. We also investigated bio-medical applications with experienced biologists. Experiments with quantitative and qualitative evaluations have successfully demonstrated the quality of our results. show that our approach can provide visually satisfying results on different inputs. Finally we hope that our work can motivate researchers that nature (in our case Miagrammopes spider webs) can be an exciting source of inspiration for computational fabrication.

Bibliography

[1] Jianghao Chang, Benoît Alain, and Victor Ostromoukhov. 2009. *Structure-aware error diffusion.* TOG (SIGGRAPH Asia) 28, 5 (2009), 162:1–162:8.

[2] Weikai Chen, Yuexin Ma, Sylvain Lefebvre, Shiqing Xin, Jonàs Martínez, and Wenping Wang. 2017. *Fabricable tile decors.* TOG (SIGGRAPH Asia) 36, 6 (2017), 175:1–175:15.

[3] Hung-Kuo Chu, Chia-Sheng Chang, Ruen-Rone Lee, and Niloy J. Mitra. 2013. *Halftone QR codes.* TOG (SIGGRAPH Asia) 32, 6 (2013), 217:1–217:8.

[4] Hung-Kuo Chu, Wei-Hsin Hsu, Niloy J. Mitra, Daniel Cohen-Or, Tien-Tsin Wong, and Tong-Yee Lee. 2010. *Camouflage images.* ACM Transactions on Graphics (SIGGRAPH) 29, 4 (2010), 51:1–51:8.

[5] Oliver Deussen, Marc Spicker, and Qian Zheng. 2017. *Weighted linde-buzo-gray stippling.* TOG (SIGGRAPH Asia) 36, 6 (2017), 233:1–233:12.

[6] Minjae Do, Byung Gee Im, Joseph P. Park, Jae Hyung Jang, and Haeshin Lee. 2017. *Functional Polysaccharide Sutures Prepared by Wet Fusion of Interfacial Polyelectrolyte Complexation Fibers.* Advanced Functional Materials 27, 42 (2017).

[7] Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 2007. *3D collage: expressive non-realistic modeling.* In Proceedings of the 5th International Symposium on Non-Photorealistic Animation and Rendering. 7–14.

[8] Brendan Galea, Ehsan Kia, Nicholas Aird, and Paul G. Kry. 2016. *Stippling with Aerial Robots. In Computational Aesthetics / Expressive.* 125–134.

[9] Brendan Galea and Paul G. Kry. 2017. *Tethered flight control of a small quadrotor robot for stippling.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[10] Hua Huang, Lei Zhang, and Hong-Chao Zhang. 2011. *Arcimboldo-like collage using internet images.* TOG (SIGGRAPH Asia) 30, 6 (2011), 155:1–155:8.

[11] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. 2006.

[12] . ACM Transactions on Graphics (SIGGRAPH) 25, 3 (2006), 509–518.

[13] Johannes Kopf and Dani Lischinski. 2011.

[14] . ACM Transactions on Graphics (SIGGRAPH) 30, 4 (2011), 99:1–99:8.

[15] Y.D. Lubin and S. Dorugl. 1982. *Effectiveness of single-thread webs as insect traps: sticky trap models.* Bulletin - British Arachnological Society 5, 9 (1982), 399–407.

[16] Ron Maharik, Mikhail Bessmeltsev, Alla Sheffer, Ariel Shamir, and Nathan Carr. 2011. *Digital micrography.* ACM Transactions on Graphics (SIGGRAPH) 30, 4 (2011), 100:1– 100:12.

[17] Tobias Martin, Nobuyuki Umetani, and Bernd Bickel. 2015. *OmniAD: data-driven omni-directional aerodynamics*. ACM Transactions on Graphics (SIGGRAPH) 34, 4 (2015), 113:1–113:12.

[18] James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica K. Hodgins. 2016.*A compiler for 3D machine knitting*. ACM Transactions on Graphics (SIGGRAPH) 35, 4 (2016), 49:1–49:11.

[19] Niloy J. Mitra, Hung-Kuo Chu, Tong-Yee Lee, Lior Wolf, Hezy Yeshurun, and Daniel Cohen-Or. 2009. *Emerging images*. TOG (SIGGRAPH Asia) 28, 5 (2009), 163:1–163:8.

[20] Niloy J. Mitra and Mark Pauly. 2009. *Shadow art*. TOG (SIGGRAPH Asia) 28, 5 (2009), 156:1–156:7.

[21] Wai-Man Pang, Yingge Qu, Tien-Tsin Wong, Daniel Cohen-Or, and Pheng-Ann Heng. 2008. *Structure-aware halftoning*. ACM Transactions on Graphics (SIGGRAPH) 27, 3 (2008), 89:1–89:8.

[22] Thrasyvoulos N. Pappas and David L. Neuhoff. 1999. *Least-squares model-based halftoning*. IEEE Trans. Image Processing 8, 8 (1999), 1102–1116.

[23] Thiago Pereira, Carolina L. A. Paes Leme, Steve Marschner, and Szymon Rusinkiewicz. 2017. *Printing anisotropic appearance with magnetic flakes*. ACM Transactions on Graphics 36, 4 (2017), 123:1–123:10.

[24] Romain Prévost, Alec Jacobson, Wojciech Jarosz, and Olga Sorkine-Hornung. 2016. *Large-Scale Painting of Photographs by Interactive Optimization*. Computers and Graphics 55 (2016), 108–117.

[25] Christian Schüller, Daniele Panozzo, Anselm Grundhöfer, Henning Zimmer, Evgeni Sorkine, and Olga Sorkine-Hornung. 2016. *Computational thermoforming*. ACM Transactions on Graphics (SIGGRAPH) 35, 4 (2016), 43:1–43:9.

[26] Jianhong (Jackie) Shen. 2009. *Least-Squares Halftoning via Human Vision System and Markov Gradient Descent (LS-MGD): Algorithm and Analysis*. SIAM Rev. 51, 3 (2009), 567–589.

[27] Roy Shilkrot, Pattie Maes, Joseph A. Paradiso, and Amit Zoran. 2015. *Augmented Airbrush for Computer Aided Painting (CAP)*. ACM Transactions on Graphics 34, 2 (2015), 19:1–19:11.

[28] Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. 2012. *Computational Design of Rubber Balloons*. Comput. Graph. Forum 31, 2 (2012), 835–844.

[29] Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. 2014. *Designing inflatable structures*. ACM Transactions on Graphics (SIGGRAPH) 33, 4 (2014), 63:1–63:10.

[30] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

[31] Xiaolin Wu. 1991. *An efficient antialiasing technique*. In SIGGRAPH. 143–152.

[32] Xuemiao Xu, Linling Zhang, and Tien-Tsin Wong. 2010. *Structure-based ASCII art*. ACM Transactions on Graphics (SIGGRAPH) 29, 4 (2010), 52:1–52:10.

[33] Haisen Zhao, Lin Lu, Yuan Wei, Dani Lischinski, Andrei Sharf, Daniel Cohen-Or, and Baoquan Chen. 2016. *Printed Perforated Lampshades for Continuous Projective Images.* ACM Transactions on Graphics 35, 5 (2016), 154:1–154:11.