Orientation Field Guided Line Abstraction for 3D Printing

Zhonggui Chen [1,4], Wen Chen [2,3], Jianzhi Guo [2,3], Juan Cao* [2,3,4], Yongjie Jessica Zhang [4]

[1] Department of Computer Science, Xiamen University, Xiamen 361000, China
[2] School of Mathematical Sciences, Xiamen University, Xiamen 361000, China
[3] Fujian Provincial Key Laboratory of Mathematical Modeling and High-Performance Scientific Computation, Xiamen University, Xiamen 361005, China
[4] Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

This paper focuses on printing a 2D color image by using a desktop fused deposition modeling (FDM) 3D printer. A fully automatic framework is presented to convert a 2D image into a non-photorealistic line drawing which is suitable for 3D printing. Firstly, an image is partitioned into a moderate number of regions, and contours of these regions are extracted to deliver the high level abstraction of the image. The contour lines are further refined by taking into consideration the restrictions of 3D printing. Next, an orientation field based on the contours and feature lines at a finer level is computed to guide the placement of streamlines. The distances between streamlines are carefully controlled such that the density respects the pixel intensity values. Finally, the resulting streamlines are converted into printing paths and printed by using filaments with specified colors. Experimental results show the feasibility and efficacy of our method on portraying a given image by using a few of 3D printable non-intersecting lines while preserving features and tone variation in the image.

1 Introduction

3D printing is a form of additive manufacturing, where a three dimensional digital model is turned into a solid object by laying down successive layers of material. 3D printing has a myriad applications in diverse industries, and studies of 3D printing focus on various aspects for different application backgrounds. In the computer graphics community, how to convert a given 3D object to meet different fabrication requirements or how to achieve high-quality and cost-effective fabrication has been the focus of many recent attempts. Instead of addressing the issue of how to print a 3D model, in this paper we focus on printing 2D color images with a 3D printer.

Among existing 3D printing technologies, fused deposition modeling (FDM) is the most simple-to-use and environment-friendly one, and FDM based printers designed for home use are also very economical and affordable. It has become the most widely used 3D printing technology nowadays. Hence, we restrict ourselves to low-end FDM printers in this paper. FDM technology based printers build objects layer by layer from the very bottom up. In each layer, the print head moves along a prescribed path, meanwhile, the thermoplastic filament is heated and extruded throughout a nozzle onto the base or the previous layer. Inspired by this line drawing manner, here we aim at printing a 2D color images using a 3D printer by converting 2D images into a set of 3D printable lines. Essentially, our goal is to render images in a non-photorealistic line drawing style which can be directly printed by a 3D printer.

There has been a lot of efforts in computer graphics on generating non-photorealistic images with line drawing style and impressive results have been achieved [10]. However, those results usually violate the requirements of direct 3D printing indeed. As a matter of fact, a wide variety of line drawing methods adopt feature edge extraction techniques to capture and convey shape features of the real scene. Represented by a sequence of pixels, the resulting zigzag lines may be very short, intensively locate at some regions and intersect with each other. While in 3D printing, fairly smooth and long lines are desired to avoid discretization artifacts in the final printing results. In particular, sharp turns/corners/endpoints lead to deceleration of the print head and leaking of filament from the nozzle. Intersections cause the extruder to hit and drag across the already printed parts. Closely located line segments either prevent the filament extruding from the nozzle or end up with filament bunching up at corresponding regions. In traditional line drawing methods, lines can be rendered with different colors to better convey the original scene. On the contrary, printing in multiple colors is non-trivial in FDM based 3D printing as most FDM 3D printers work with a single color. Some research focus on generating continuous line illustration, i.e., portraying the scene with a single continuous line [17, 18]. However, it is difficult to trace out the feature lines and convey the tone variation of the 2D images by using a unicursal curve in general. In a recent literature [4], semi-continuous line illustration was generated for 3D printing. Relying on manual segmentation results, the line drawing results are able to preserve the contours of the image while features inside each sub-region are ignored.

In this paper, we focus on printing color images using FDM based 3D printers in line drawing style. Several existing techniques are adapted to our framework to achieve this goal. The specific contributions of this paper are as follows:

(1) We develop a fully automatic non-photorealistic line drawing algorithm for converting a 2D image into a 3D printable line illustration. After specifying the physical printing size, line width and colors, our method automatically generates visually pleasant 3D printing results.

(2) We construct printable contours of the input image, which portray the most important features of the image, even with the strict limitations of printing size and resolution of the 3D printers.

(3) We generate intersection-free and reasonably spaced streamlines, which are suitable for direct 3D printing. In addition, streamline orientations are well controlled to better convey features at the finer level. Their distribution also approximates tone variations of the input image precisely.

(4) Our method is capable of obtaining multi-color outputs using 3D printers with a single extruder, without resorting to costly color 3D printers.

* Corresponding author. Email address: Juancao@xmu.edu.cn (Juan Cao)
Part of this work was done while Zhonggui Chen and Juan Cao were visiting the Department of Mechanical Engineering, Carnegie Mellon University.

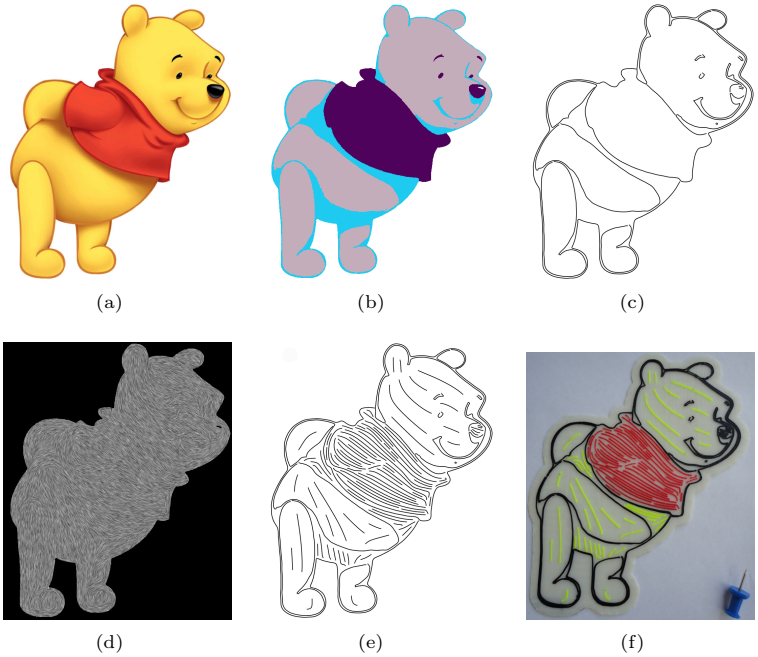

Figure 1: Overall process. (a) Input image; (b) image segmentation; (c) contour extraction; (d) orientation field generation; (e) streamline placement; and (f) 3D printing.

The remainder of this paper is organized as follows. Related work relevant to our research is presented in Section 2. Details of our line drawing abstraction algorithm are provided in Section 3. Results of our algorithm are shown in Section 4, while section 5 presents the concluding remarks and possible extension of this work.

2 Related Work

Line drawing is a two-dimensional visual art style featuring of portraying scenes only with lines. Non-photorealistic rendering (NPR) in line drawing style frequently appears in animations, architecture illustrations, and advertisements. There exist numerous investigations on generating a line drawing from an image in the field of computer graphics. As the contours or feature lines convey the most important shape information of an image, most technologies for line drawing are based on the edge detection method, and the generated lines may vary in width, density and color as necessary to portray the features and the overall shading of the scene. For detailed discussions on existing line drawing methods, we refer the readers to a recent survey paper [10]. In addition, some researches address the problem of continuous line illustration, i.e., portraying a scene with a single line. Due to its restricted nature, the complexity of the problem increases dramatically [2, 17, 18], and the proposed methods are either time-consuming or rely on user interactions. Image processing techniques for generating mazes from images [19, 16] share some similarities with line drawing methods which trace contours and approximate the tone of the source image. However, all of these techniques, ignore the concerns specific to 3D printing (e.g., line width and spacing). Hence, their results are not applicable for direct printing in general.

With the fast development of modern fabrication technologies, direct drawing of lines in 3D space by using a 3D extruder becomes possible. Fabricating 3D shapes using multiple wires has led to growing interests in the 3D printing community. While wireframe fabrication remains challenging, as various geometric and physical constraints should be considered during the entire drawing process, e.g., the already-fabricated parts should always keep stable and not be collided by the moving extruder. To speed-up the prototyping process, a regular wireframe is fabricated directly in 3D space using a standard FDM 3D printer, instead of printing the original solid layer-by-layer [11]. Various 5DOF or 6DOF 3D fabrication systems were also designed to print geometrically complex wireframe shapes [12, 7]. Huang *et al.* proposed to generate a feasible fabrication sequence of struts for creating general frame shapes using a 6DOF robotic fabrication system [8]. A mixed reality system was developed to provide intuitive guidance for easy 3D drawing using a 3D extruder pen [20].

Instead of drawing lines in 3D space, in this paper we address the more fundamental question of how to depict a color image in 2D domain by using FDM 3D printers. A space filling curve named connected Fermat spiral was introduced as a fundamental 2D fill pattern for layered fabrication in 3D printing [21]. Printing paths constructed by connected Fermat spirals possess appealing properties, e.g., smoothness and intersection-free, making them suitable for 3D printing. 3D printing techniques were applied to food printing [22], where an arbitrary input image is automatically converted into optimized printable paths with prominent features, disregarding the tone variation. A dithering technique for 3D printers with at least two extruders was presented for producing continuous tone variation on models using interleaved color patterns [13]. In the semi-automatic continuous line drawing method [4], lines were generated by solving several traveling salesman problems (TSP) on each manually segmented region. In contrast to the dual-color mixing method [13], where at least two nozzles were needed to convey an impression of tone variation, it generated line segments with distributions consistent with the tone variation. However, this method is computationally expensive due to the inherent complexity of the TSP. In addition, the line orientations were also random in the results, as a TSP solver computes the shortest path without taking into account the orientation information. In this paper, we develop a fully automatic and efficient line drawing algorithm to convert a 2D image into a 3D printable line illustration, where the line orientations are more precisely controlled, with features and tone variation preserved.

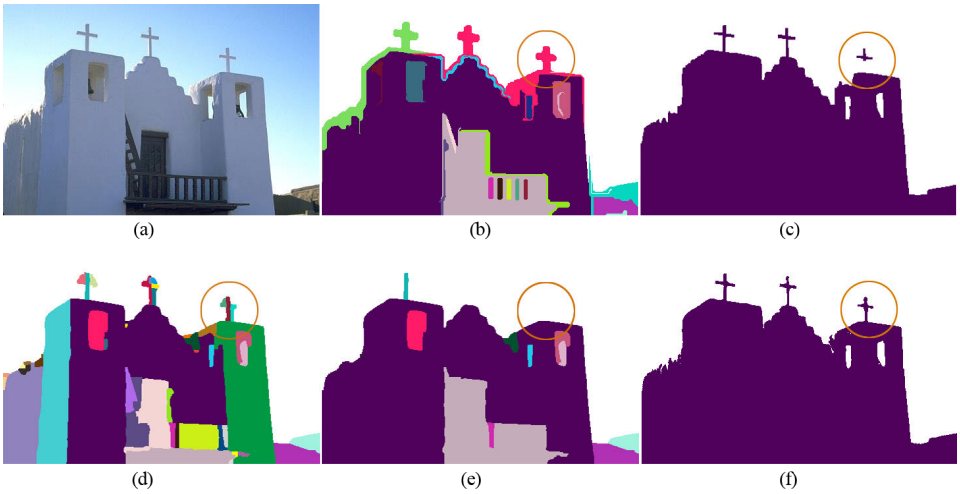

Figure 2: Comparison of image segmentation methods, where different regions are filled with different colors. (a) Input image; (b) segmentation result from the graph-based method [5]; (c) segmentation result from the graph cut based method [14]; (d) and (e) segmentation results from the hierarchical image segmentation method [1], by thresholding ultrametric contour map at levels 0.1 and 0.3, respectively; and (f) result from the active contour based method in [6].

3 Algorithm

3.1 Algorithm Overview

Our framework takes an arbitrary image as input. To obtain color prints while keeping the important features and tone variations in the image, we need to overcome several obstacles. The first issue is color printing, as we use the 3D printers with only single/double extruders in this paper. We firstly adapt the segmentation technique to partition the image into a reasonable number of regions, with each of them printed individually in its own specified color, see Figure 1(b). Secondly, we need to preserve the most important features of the image. To achieve this goal, we further optimize the contours of each region to make them suitable for 3D printing, see Figure 1(c). Thirdly, to better capture the features inside each region, an orientation field is created as a guidance for later streamline generation, see Figure 1(d). Finally, we trace the streamlines, and distances between them are adaptively controlled such that their distribution is both consistent with the image tone variation and suitable for 3D printing, see Figure 1(e). The algorithm pipeline is given in Figure 1 and details of each step will be described in the remainder of this section.

3.2 Image Segmentation

Multi-color printing is the first obstacle for us to address, as the single-extruder or dual-extruder printers are capable of extruding only one or two colors at a time. Here we segment the original color image into different segments, and then print each of them individually using filament with a similar color. Image segmentation techniques have been widely studied as it is fundamental to the fields of image analysis, image understanding and image pattern recognition. Numerous methods/algorithms have been proposed in different application backgrounds, such as edge detection methods, dual clustering methods, and region growing methods. However, neither a single theory/method nor universal segmentation framework exists that can be adapted to all images or applications. For a comprehensive survey of image segmentation techniques, please refer to paper [23]. For the purpose of this paper, we need a segmentation method which is capable of automatically generate a moderate number of sub-regions, corresponding to meaningful objects/components and having appropriate sizes and small color variations. These sub-regions do not need to be simply-connected, however, small pieces should be avoided as they are difficult or even unable to be printed due to the resolution limit of the printers.

There exist many image segmentation algorithms which offer a basic fit for our application. Here, we compare the performances of four representatives from the state-of-the-art segmentation methods, including the active contour based method [6], graph-based method [5], graph cut based method [14], and hierarchical image segmentation method [1]. The active contour based method [6], which performs best for our application, is adopted in this paper. Owing to the combination of active contours with a statistical framework, the active contour based method can precisely segment an image into a moderate number of meaningful parts with few small pieces. As shown in Figure 2, the active contour based method yields less sub-regions than the method in [5], while provides a more precise segmentation than the method in [14], see the circled regions in Figure 2(b), (c), and (f). By thresholding ultrametric contour map at different levels, the method in [1] is capable of generating hierarchical segmentations. However, it may lead to too many segments at a fine level or fail to capture some visually important objects at a coarse level, see Figure 2(d) and (e).

3.3 Printable Contour Generation

Restricted to the resolution and print size of 3D printers, images fabricated by 3D printers in the line drawing manner have low resolutions. That is, only a limit number of lines can be used to represent the original image in 3D printing. To better convey the shapes and objects in the original image, we include the contours of the images into the final printing line set. Based on the segmentation result, the single-pixel-width boundary of each sub-region is traced out. These contours are represented as sequences of the boundary pixels. Those zigzag contours, consisting of many short branches and sharp turns, see Figure 3(c), are not suitable for printing. We should also bear in mind that lines to be printed by a 3D printer have a physical width. Closely located segments of the contours may cause filament overlapping in the final printing. Hence, the layout of the contours should be further optimized to reduce the short branches/sharp turns and amplify the distance between closely located segments.

Let us first introduce some notations to assist the discussion. A contour is a sequence of pixels or points, denoted by $T = \{v_0, v_1, \cdots, v_n\}$, where v_i is called a vertex of T. $d(v_i, v_j) = \|v_i - v_j\|$ is the ordinary straight-line distance between vertices

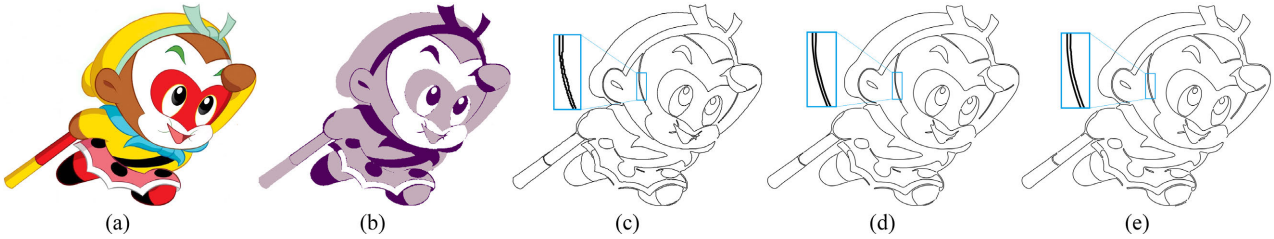

Figure 3: Printable contour generation. (a) Input image; (b) segmentation result; (c) initial contours; (d) contour smoothing; and (e) final contours obtained by pulling apart the segments close to each other.

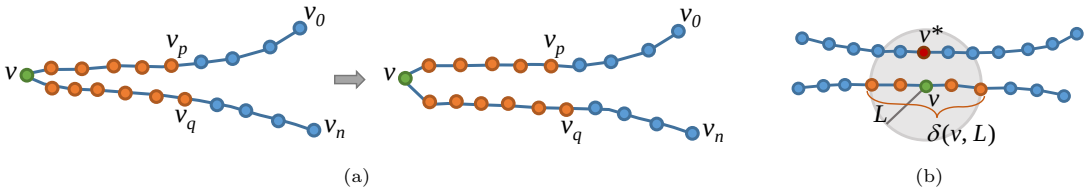

Figure 4: Pulling apart closely located segments. (a) Step 1: pulling apart v-region, where v is a feature point, and the segments between v_p and v_q are defined as V-region; and (b) step 2: pulling apart parallel lines, where v^* is the closest vertex of v that is not in $\delta(v, L)$.

v_i and v_j. For two vertices $v_s, v_t \in T$ with $s < t$, the distance from vertex v_i to segments between v_s and v_t on the contour T, denoted by $d(v_i, \widehat{v_s v_t})$, is the distance between vertex v_i and its closest point on segments between v_s and v_t. For any vertex $v_i \in T$, $\delta(v_i, L) = \{v_p, v_{p+1}, \cdots, v_i, \cdots, v_{q-1}, v_q\} \subseteq T$ is the vertex sequence satisfying $d(v_i, v_j) < L$, where $j = p, \cdots, q$. That is, $\delta(v_i, L)$ is the set of neighboring vertices of v_i that are within distance L, see Figure 4(b). In addition, we assume that the pre-specified line width is L.

First, we remove the contours with total length less than L from the contour set. Then, the fairness of the remainder contours is improved by applying the Laplacian filter, where vertices with turn angle smaller than $\pi/6$ are marked as feature points and they are fixed during the faring process to prevent shrinkage at sharp features. A faring result is shown in Figure 3(d). Finally, we increase the space between closely located segments by applying the following two steps:

Step 1. Starting from each feature vertex $v \in T$, find the last vertices v_p, v_q $(p < q)$ backward and forward along T satisfying $d(v_p, \widehat{v v_q}), d(v_q, \widehat{v_p v}) < L$. Segments between v_p and v_q are defined as the V-shape region of v on T, see Figure 4(a) for instance. Then, each vertex v_s between v_p and v is moved a distance $(L - d(v_s, \widehat{v_p v}))/2$ along the normal direction of the line. Vertices between v and v_q are moved in the same manner. Intuitively, a V shape is adjusted to a U shape, see Figure 4(a) as an example.

Step 2. For any remaining vertex v on the contours, we check if it is too close to other lines. We find its closest vertex v^* which is not in $\delta(v, L)$, as shown in Figure 4(b). If $d(v, v^*) < L$, then both v^* and v are moved backward at a distance $(L - d(v^*, v)))/2$. This step is applied repeatedly until no vertex out of the V-shape regions needs to be moved. An example of optimized contours is shown in Figure 3(e).

3.4 Orientation Field Design

With printable contours generated, our next step is to fill each region with curved lines such that they convey the features and tone variations of the original image. To control line orientations, we design a vector field to guide line placement. Note that, we cannot directly use the gradient field of the image since the gradient may go to zero at some regions of an image, especially for cartoon images, as shown in Figure 5. Thus, we use the local contour orientation to derive the orientation vectors at pixels with a vanished gradient.

The contours obtained in the previous section share overall features with the original image. However, detailed features located inside each region are missing. Therefore, constructing vector field simply based on contours would ignore textures in the image. To better convey the original image, we take features in each region into consideration. In particular, we use the method in [15] to find the feature edges at a finer level and derive vector fields from both the contours and feature edges. The final vector field used for line placement is a linear combination of these two vector fields. Although the feature edges extracted using the method in [15] usually contain the object contours in the image, here we consider the contours separately to emphasize their influence on the final orientation field.

Let S_c and S_e be the set of line segments of the obtained contours and feature edges, respectively. For each segment s, we denote its length and orientation angle by l_s and θ_s, respectively. A tensor with orientation θ is defined by a symmetric matrix:

$$A(\theta) = \begin{pmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{pmatrix},$$

whose two unit eigenvectors give two orthogonal orientation vectors. Then the tensor at any point p on the image is defined by:

$$T(p) = \omega_c \sum_{s \in S_c} \frac{l_s A(\theta_s)}{1 + d(p, s)^{3/2}} + \omega_e \sum_{s \in S_e} \frac{l_s A(\theta_s)}{1 + d(p, s)^{3/2}}, \quad (1)$$

where $d(p, s)$ is the distance from p to the line segment s, and w_c and w_e are user-specified weights ranging in $[0, 1]$. Intuitively, the longer segment has a stronger influence on the field. Meanwhile, the influence of a segment on the field at a point declines as the distance between the segment and point increases. Figure 1(d) and Figure 5(b) show two orientation fields computed by using Eq. 1 with $\omega_c = 0.2$ and $\omega_e = 0.8$, and they are visualized by using the line integral method in [3].

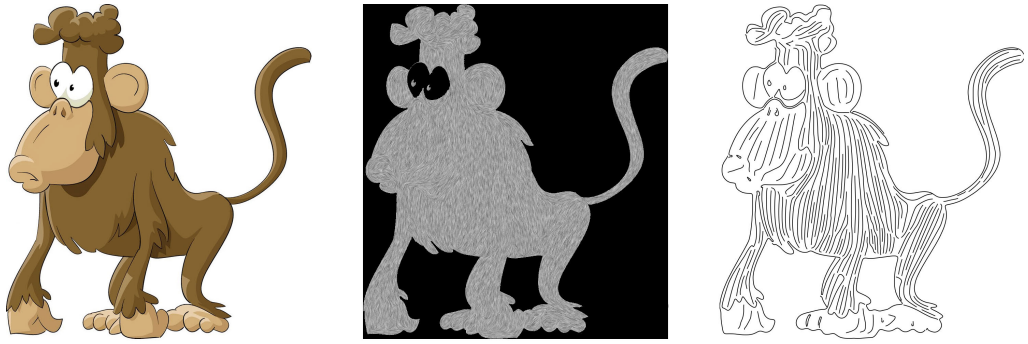

Figure 5: Orientation field design. (a) Input image; (b) vector field generated from our method; and (c) line drawing result.

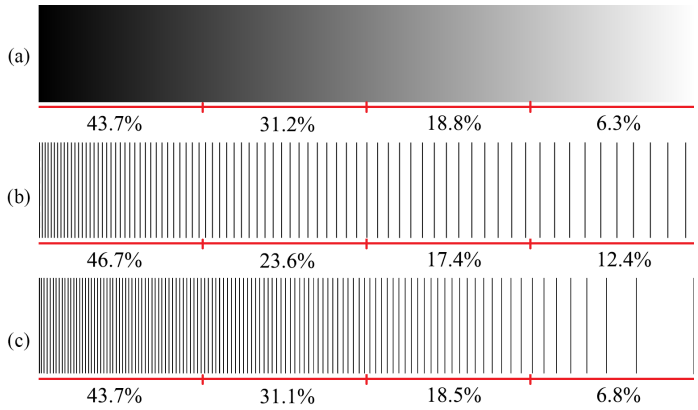

Figure 6: Line placement results with different spacing functions. (a) Input color ramp; (b) line placement by using the linear spacing function, and (c) line placement result from our method.

3.5 Streamline Placement

Displaying streamlines is a widely used technique for vector field visualization [9]. Our goal is to depict intensity variations in an image by using long and adaptively spaced streamlines. Here we study how to control the local density of streamlines, which is expressed by the distance between two adjacent streamlines.

Our method starts from a random seed point in the field, and then a streamline grows beyond that point backward and forward by using the Runge-Kutta integrator with a fixed step size. The growing process stops when the line goes too close to existing streamlines or it reaches the boundary of a region. The new random seed point is selected if its distance to the existing streamlines is larger than a specific value. The algorithm stops when there is no valid seed point. The remaining problem is to control the separating distance between streamlines. We can simply compute the distance between streamlines by a linear function with respect to pixel's luminance values, as did in [18]:

$$d(p) = (d_{\max} - d_{\min})I(p) + d_{\min},$$

where $d_{\max}$ and $d_{\min}$ are the maximum and minimum distance between streamlines, respectively, and $I(p)$ is the normalized image luminance at point p. In a color image, the luminance value is usually computed by a weighted combination of three color channels. We adopt the implementation of color inversion in OpenCV, where each pixel's luminance value is determined by $0.299 \times Red + 0.587 \times Green + 0.114 \times Blue$. $d_{\min}$ can be set as the width of printing path. That is, printed lines will only touch each other in the black regions. Figure 6(b) shows a line placement result by using the above linear spacing function. The percentages in each quarter represent the ink density in the image, and the line density in the line drawings. To better convey the luminance variations in the image, the area covered by lines in the domain should be proportional to the darkness of the image. Thus, we have

$$d(p) = \frac{L}{1 - I(p)}, \tag{2}$$

where L is the width of the printing path. When $I(p)$ goes to 1, $d(p)$ goes to infinity. In other words, no line appears in white regions. As shown in Figure 6(c), our new spacing function returns a more precise matching of the reference percentages.

4 Experimental Results

We show line drawing results on a variety of input images. Our line drawing algorithm was implemented in C++ on an Intel Core TM i5 3.2 GHz CPU with 12 GB RAM. The 3D printing experiments were conducted on a Makerbot replicator 2X 3D desktop printer with Makerware 2.0. Printing results are based on the default setting, where the extrusion width is set at 0.4mm and layer thickness at 0.2mm in all experiments of this paper. The parameter L (i.e., line width) in our algorithm is set to be the extrusion width of our 3D printer by default. A supportive raft is generated by Makerware to provide a base for drawing, therefore we obtain a stable one-piece result. G-code is used to transform the lines to print paths for 3D printer.

We compare our line drawing method with three existing methods, including the Canny edge detector, the EdgeDrawing method [15] and the TSP-based method [4], as shown in Figure 7. To obtain a quality 3D printing result, lines served as the printing paths are preferred to be long and continuous with less sharp turns. From Figure 7 we can observe that, the result

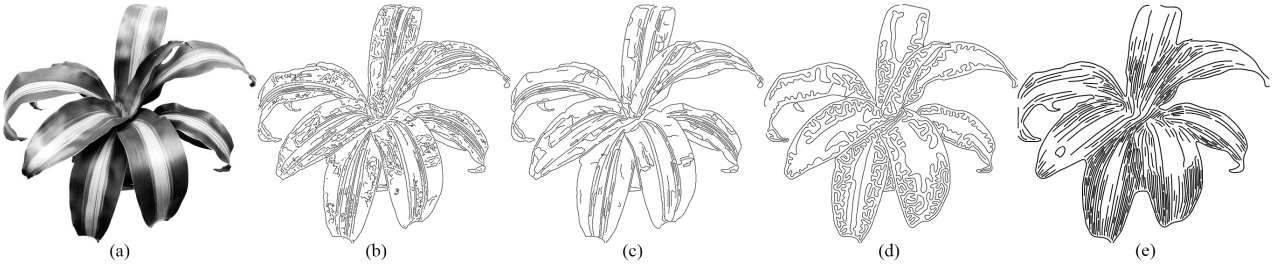

Figure 7: Comparisons with other line drawing methods. (a) Input image; (b) result from the Canny edge detector; (c) result from the EdgeDrawing method [15]; (d) result from the TSP-based method [4]; and (e) our result.

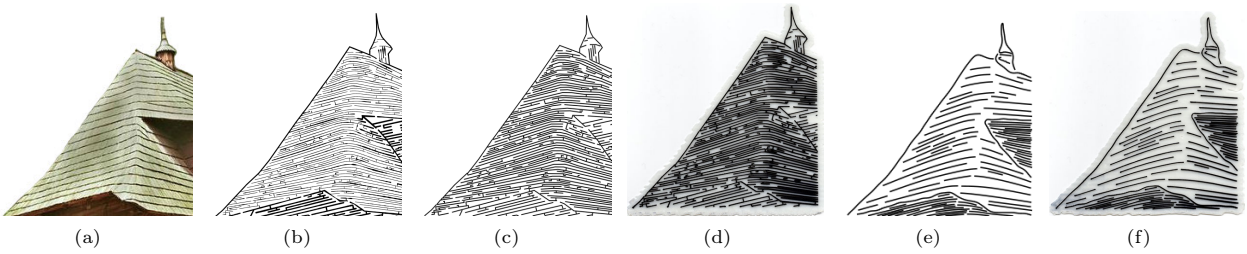

Figure 8: Comparison with the maze generation method [19]. (a) Input image; (b) maze patterns from [19]; (c) lines extracted from (b); (d) 3D printing result of (c); (e) line drawing result from our method; and (f) 3D printing result of (e).

from the Canny edge detector suffers from some obvious flaws, such as containing too many short and zigzag segments. The EdgeDrawing method is capable of capturing the most important features while introducing less short lines, as compared to the Canny edge detector. However, it failed to portray the tone variation in the image. The TSP-based method [4] generates long and fairness lines with a distribution adapting to the tone variation, but the line segments are randomly oriented. Our method shares the advantages of the TSP-based method, i.e., generating relatively long and continuous lines with distribution adapting to color intensity. Moreover, our method has superiority in controlling the line orientations.

We also compare our method with the maze generation method [19]. The comparison results have been shown in Figure 8. Here we only print a part of the original image (Figure 12 in [19]), due to the printing size limit of the 3D printer. Note that, the maze generation method [19] conveys the tone of the original image by controlling the widths of lines as well as their spacing, as shown in Figure 8(b). To take them as printing paths, we have to first convert them to line segments with a single width. As shown in Figure 8(c) and (d), the tone variation of the original image is not well preserved after unifying the line width.

Several monochromatic and color printing results as well as the original images are shown in Figure 9. For color printing, black filaments are used for printing contour lines. Other lines inside each sub-region are printed by using filaments with the average color of these lines. As we print all the results using a 3D printer with just one or two extruders, we manually switch colors by pulling off and re-feeding the filament to the extruder.

A straightforward application of our algorithm is to fabricate stamps from given images, of which two examples are shown in Figure 10. Table 1 reports the statistics of our method on generating 2D line drawings and 3D printing results. The time for color printing includes the time for changing the filaments with different colors.

5 Conclusion

This paper presents a fully automatic algorithm for the creation of 2D non-photorealistic line drawing which is suitable for direct 3D printing. An image is partitioned into several regions whose contours considered as the major features are optimized for 3D printing. To better capture the texture properties of the scene, an orientation filed is derived from the contours and extracted features to guide the placement of streamlines. The distance between streamlines is carefully controlled such that their distribution approximates the image intensity. Our approach generates line drawings whose directions are determined by both the contours and the fine features of the image, as opposed to [4], where the line orientations are random.

Our algorithm fails to handle images containing too many small details in areas with sharp variation of intensity, mainly due

Table 1: Statistics of our method on generating 2D line drawings and 3D prints.

Figure	Image resolution	Cluster number	Orientation field construction (sec)	Streamline placement (sec)	Pringting size (mm)	Layer number	Printing time (min)
1	680×825	4	19.6	25.1	105×130.5	2	45
8(f)	292×240	3	0.6	0.5	132×138.8	2	28
9(a)	642×567	2	7	7.2	135×118.6	2	27
9(b)	900×900	4	24.9	40.6	115.7×120	2	60
9(c)	427×550	4	24.4	21.4	86.2×110.5	2	65
9(d)	714×1000	4	115.9	109.4	84.8×117.4	2	79
9(e)	811×478	3	42.9	19.4	135×78.9	3	60
10(a)	428×500	2	3.6	41.2	42.8×50	25	35
10(b)	610×365	3	4.3	20.2	70×41.2	40	100

Figure 9: Monochromatic (a) and color (b-e) printing results with original images.

Figure 10: Photographs of 3D fabricated stamps. (a) A 3D printed stamp of 25 layers and a stamped napkin; (b) a 3D printed stamp of 40 layers and a stamp on a dough.

to the limitation of resolution and print size in 3D printing. One possible solution to improve the printing results is to print the complicated images as separate components and then assemble them. The printing process with multiple colors is also laborious in this paper, as the low-end printers work with one or two extruders, and transitions for multiple colors can only be achieved manually. For future work, we would like to extend our method to convert a given 3D model to wireframes that are suitable for direct 3D fabrication using robotic fabrication systems.

Acknowledgement

The research of Zhonggui Chen and Juan Cao was supported by the National Natural Science Foundation of China (No. 61472332, 61572020, 61100105), the Natural Science Foundation of Fujian Province of China (No. 2015J01273), and the grant of China Scholarship Council. The research of Yongjie Jessica Zhang was supported in part by the PECASE Award N00014-16-1-2254 and NSF CAREER Award OCI-1149591.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, May 2011.

[2] R. Bosch and A. Herman. Continuous line drawings via the traveling salesman problem. *Operations Research Letters*, 32(4):302–303, 2004.

[3] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 263–270, New York, NY, USA, 1993. ACM.

[4] Z. Chen, Z. Shen, J. Guo, J. Cao, and X. Zeng. Line drawing for 3D printing. *Computers & Graphics*, 66(Supplement C):85 – 92, 2017. Shape Modeling International 2017.

[5] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[6] G. Gao, C. Wen, H. Wang, and L. Xu. Fast multiregion image segmentation using statistical active contours. *IEEE Signal Processing Letters*, 24(4):417–421, 2017.

[7] N. Hack and W. V. Lauer. Mesh-mould: Robotically fabricated spatial meshes as reinforced concrete formwork. *Architectural Design*, 84(3):44–53, 2014.

[8] Y. Huang, J. Zhang, X. Hu, G. Song, Z. Liu, L. Yu, and L. Liu. Framefab: Robotic fabrication of frame shapes. *ACM Transactions on Graphics (TOG)*, 35(6):224, 2016.

[9] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing'97*, pages 43–55. Springer, 1997.

[10] Y. Liu, Y. U. Minjing, F. U. Qiufang, W. Chen, Y. Liu, and L. Xie. Cognitive mechanism related to line drawings and its applications in intelligent process of visual media: a survey. *Frontiers of Computer Science*, 10(2):216–232, 2016.

[11] S. Mueller, S. Im, S. Gurevich, A. Teibrich, L. Pfisterer, F. Guimbretière, and P. Baudisch. Wireprint: 3D printed previews for fast prototyping. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 273–280. ACM, 2014.

[12] H. Peng, R. Wu, S. Marschner, and F. Guimbretière. On-the-fly print: Incremental printing while modelling. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 887–896. ACM, 2016.

[13] T. Reiner, N. Carr, R. Mech, O. Stava, C. Dachsbacher, and G. Miller. Dual-color mixing for fused deposition modeling printers. *Computer Graphics Forum*, 33(2):479–486, 2014.

[14] M. B. Salah, A. Mitiche, and I. B. Ayed. Multiregion image segmentation by parametric kernel graph cuts. *IEEE Transactions on Image Processing*, 20(2):545–557, 2011.

[15] C. Topal and C. Akinlar. Edge drawing: A combined real-time edge and segment detector. *Journal of Visual Communication and Image Representation*, 23:862 – 872, 2012.

[16] L. Wan, X. Liu, T.-T. Wong, and C.-S. Leung. Evolving mazes from images. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):287–297, 2010.

[17] F. J. Wong and S. Takahashi. A graph-based approach to continuous line illustrations with variable levels of detail. *Computer Graphics Forum*, 30(7):1931–1939, 2011.

[18] F. J. Wong and S. Takahashi. Abstracting images into continuous-line artistic styles. *The Visual Computer*, 29(6):729–738, 2013.

[19] J. Xu and C. S. Kaplan. Image-guided maze construction. *ACM Transactions on Graphics (TOG)*, 26(3):29, 2007.

[20] Y.-T. Yue, X. Zhang, Y. Yang, G. Ren, Y.-K. Choi, and W. Wang. Wiredraw: 3D wire sculpturing guided with mixed reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3693–3704. ACM, 2017.

[21] H. Zhao, L. Lu, Y. Wei, D. Lischinski, A. Sharf, D. Cohen-Or, and B. Chen. Printed perforated lampshades for continuous projective images. *ACM Transactions on Graphics (TOG)*, 35(5):154, 2016.

[22] H. Zhao, J. Wang, X. Ren, J. Li, Y.-L. Yang, and X. Jin. Personalized food printing for portrait images. *Computers & Graphics*, 70:188 – 197, 2018. CAD/Graphics 2017.

[23] H. Zhu, F. Meng, J. Cai, and S. Lu. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, 34:12–27, 2016.