The Visual Computer (2018) 34:1633–1644
https://doi.org/10.1007/s00371-018-1549-z

ORIGINAL ARTICLE

An improved topology extraction approach for vectorization of sketchy line drawings

Jiazhou Chen[1] · Mengqi Du[1] · Xujia Qin[1] · Yongwei Miao[2]

Published online: 10 May 2018

Abstract
Vectorization converts raster scans of line drawings into vector graphics; it breaks the barrier between line drawing generation and postprocessing. Prior work on line drawing vectorization considerably succeeded in revealing artists' drawing intention driven by structural topologies. However, none of them is able to extract simplified topologies for sketchy line drawings consisted by many unwanted lines. In this paper, we propose an improved topology extraction approach based on artists' sketching customs. Redundant regions and open curves are discriminated from artists' deliberate ones and further removed progressively through an iterative optimization mechanism. We demonstrate that our improved topology benefits our vectorization method as well as existing topology-driven ones and allows them to vectorize rough sketchy line drawings robustly and efficiently.

Keywords Line drawing · Sketch · Vectorization · Topology extraction

1 Introduction

Line is an essential element of object shape information; thus, line drawings have become a popular tool for presenting and communicating ideas in both scientific and design domains. Line drawings have two major representations: vector graphics and raster image, either of which dominates on the generation and usage of line drawings simultaneously. The vector graphics provide many practical facilities such as resolution independence, editing convenience and storage economy, but are difficult to create as they require tuning control points and degree parameters. On the contrary, most artists still prefer to freehand sketching on papers to quickly present their ideas, but lines rasterized in scanned bitmaps can be neither edited conveniently nor stored economically.

The goal of line drawing vectorization is to automatically convert line drawing raster bitmaps into vector graphics; it shortens the aforementioned gap between the generation and usage of line drawings. The key is to find a set of parametric curves that integrally preserve the main structure of the input line drawing. It is particularly challenging to automatically clean up all redundant lines introduced by artists' fast and casual sketching without destroying structural details. A redundant stroke/region could be longer/larger than a deliberate stroke/region; thus, it is already difficult to distinguish them. Traditionally, artists solve this challenge by carefully re-drawing representative lines on the top of the sketching draft, which is tedious and time-consuming.

Lines are mixed together in the input raster bitmap; neither separating nor grouping them is possible [2,16]. On the other hand, image simplification methods also fail to preserve the main structures of the input line drawings [23], since stroke thickness variation may lead to line connectivity breaking. Among existing vectorization methods in the literature, topology-driven ones have made a big progress on preserving the main structure while removing unwanted lines. Noris et al. employ a gradient-based pixel clustering and the minimum spanning tree to extract the topology of line drawings, and analyze the junction continuities by a reverse drawing procedure [18]. However, this method can only process clean line drawings, as the topology has not been simplified well. Favreau et al. extract skeletons by morphological dilation and thinning to construct an initialized curve

✉ Jiazhou Chen
cjz@zjut.edu.cn

✉ Xujia Qin
qxj@zjut.edu.cn

[1] College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China

[2] School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China

network, and explicitly balance fidelity to the input bitmap with simplicity of output [10]. Their hypergraph exploration can minimize the number of curves and their degrees in the output vector graphics, but can not clean up redundant regions or strokes. In summary, the state-of-the-art methods make use of the drawing topology, but are limited to non-qualified topology extraction, and thus fail to vectorize rough sketchy line drawings.

In this paper, we propose an improved topology extraction method for vectorization of sketchy line drawings. We first over-segment the input image using the trapped-ball algorithm. Then an iterative region optimization is employed to remove all redundant regions introduced by artists' inaccurate sketching and preserve all deliberate regions even if they are smaller than redundant ones. Thirdly, a topological graph is initialized by a skeleton extraction, which not only guarantees the curve continuity but also finds all junctions and endpoints. This topology graph is further simplified by an iterative open curve removal process. In the end, we propose an efficient vectorization method that makes use of our improved topology to robustly vectorize rough sketchy line drawings. We demonstrate the improvement of our topology extraction on various types of sketchy line drawings. Our contributions can be summarized as followed:

- We propose an improved topology extraction approach that captures the main structure of sketchy line drawings without the residual of unwanted sketching lines.
- We introduce a line drawing vectorization method that is simple but efficient, only takes several seconds for very rough line drawings.

2 Related work

Line drawing vectorization Line drawing vectorization stems from converting scanned engineering drawings to electric diagrams; thus, early vectorization approaches focus on geometric primitive recognition, such as straight lines [6,7], ellipses [24], Bézier curves [28] and even B-splines [5]. Recognition-based methods are capable of producing compact parametric curves through fitting algorithms, but the vectorization robustness is barely guaranteed. For instance, if lines in the input image have thickness variations, special treatments have to be involved to against over-fitting errors, like the layer separation in the work of Halaire and Tombre [12,13].

Recognition-based vectorization methods are limited to particular geometric primitives. To vectorize freeform lines, tracking-based methods first calculate a tangential field for the input image and then trace streamlines using line integral convolution along this field. A number of existing direction field estimation methods can be found, like Gabor filter [4], tensor field [15] and non-oriented gradient field [8]. Central streamlines can be directly traced out starting from a seed point at the line center [1]; or as midpoints of two contour streamlines that are traced out starting from seed points at the line boundary [17]. For both cases, special correction mechanisms have to be applied to avoid the accumulating errors in the tracking stage, such as Kalman filter by Bartolo et al. [3], 2D quadratic fitting and Runge–Kutta algorithm by Chen et al. [9]. And Bao and Fu developed a tracing-based line drawing vectorization method that uses cross sections to accurately trace lines with near-constant width [1].

Stroke simplification Artists' line drawings commonly consist of rough strokes mixed with many redundant ones, rather than clean strokes. The target of line drawing simplification is to replace redundant strokes with a set of clean ones that fully represent the original content. These representative strokes can be found by either selecting from input ones or grouping to have new ones. Stroke selection can be achieved by a stroke removal process starting from the strokes with low priorities, which are measured by stroke properties, such as length or density. In particular, Preim and Strothotte directly limit the line number when generating simplified drawings of 3D scenes [19], Wilson and Ma compute a complexity map to control the line density in pen-and-ink illustrations [27], and Grabli et al. introduce an image-space density computation to measure the significance of 3D lines and use it to remove strokes with low density progressively until the reminder satisfies the user [11].

However, the representative strokes can be hardly selected from many short ones, which are drawn with an intention of creating a long line. In this case, grouping them is a more reasonable way than selection. For this sake, Barla et al. group lines in the original drawing into θ-groups based on geometric properties including proximity, continuation and coverage and finally create a single line for each group [2]. And Liu et al. point out that regions formed by lines can also highly impact the stroke grouping [16]. They thus make use of the Gestalt principle of closure in line drawings to determine the grouping standard on a semantic level. Once all strokes are grouped using an iterative cyclic refinement, each group is then replaced by a smooth curve for the final simplification. And Chen et al. reconstruct 2D curves from sketched strokes through a non-oriented gradient field estimation that is robust to different drawing styles [8].

Topology extraction Aforementioned simplification methods require vector strokes as input; they are only suitable for digitally created line drawings, not for paper-based line drawings that are still preferred by most artists. To vectorize the line drawings in raster images, recent methods extract a structural topology to drive the vectorization process. Noris et al. employ a gradient-based pixel clustering technique to compute the underlying topological skeleton, and use it to solve junction ambiguities in the vectorization stage by a reverse

drawing procedure [18]. However, this method can only deal with clean line drawings.

2D skeleton, one-pixel-width medial axis of each line drawing, is the key element of structural topology [20,22]. However, skeleton extraction methods employed by existing line drawing vectorization approaches are noise sensitive and structure breaking. These issues are not evident for vectorization of clean line drawings [18,31], but tend to introduce large inaccuracy for vectorization of sketchy line drawings. To process rough line drawings, Favreau et al. extract a curve network with one-pixel-width skeleton as a drawing topology and use it to balance between fidelity to the input bitmap and simplicity of the output vector graphics [10]. However, the morphological dilation and thinning in their topology extraction inherit the inaccuracy of artists' sketchy drawing, as they can not clean up redundant regions or open curves. And the subsequent hypergraph exploration in Favreau et al.'s method can only reduce the number of Bézier curves and their degrees, not the topology itself.

3 Overview

Figure 1 illustrates the overview of our approach. It takes a bitmap of sketchy line drawing as input and segments this bitmap into separated regions according to pixel color difference between strokes and background. Secondly, unwanted regions introduced by artists' scratchy sketching are identified and removed. Thirdly, an initial topology is estimated through a combination of skeleton extraction and the detection of endpoints and junctions. This topology is optimized by removing unexpected open curves and used to drive the final vectorization method to convert the input bitmap into high-quality vector graphics.

The core of our approach is an iterative topology simplification mechanism that removes all unexpected regions and open curves while preserving the user-intended drawing structure, from Fig. 1c–e. We estimate and simplify a topological graph based on skeleton extraction. It measures the area, flatness, outer width and independence for background regions, and parallelism and relative distance for open curves. These measurements are combined as a unified framework and updated dynamically in an iterative manner. Our approach can better identify the user-intended drawing structure from unexpected regions and curves comparing with existing methods and helps our vectorization method achieve high-quality vector graphics for sketchy line drawings robustly and efficiently.

4 Topology extraction and simplification

Our method starts with segmenting the input image to separated regions. Since strokes are roughly drawn by artists, some of them do not form completely closed regions. Inspired by Zhang et al. [30], we employ the trapped-ball segmentation method on the binary form of the input image. It produces regions without leakages. These regions have two types: regions covered by strokes, noted as R_s and many separated background regions, noted as $R_i, i \in [1, N]$, where N is their total number. All background regions R_i are dilated without crossing stroke regions, to cover all remaining unsegmented pixels [10]. Note that the trapped-ball method over-segments most narrow background regions due to its leakage-free nature, we will explain how to remove them in the next subsection.

4.1 Region removal

Figures 1b and 2b show our segmentation results. Due to the roughness of artists' sketches, many unexpected background regions are produced around the repeated strokes, and they are extremely difficult to distinguish from regions with artists' intention. Based on our observation on many sketches and deep discussion with artists, we summarized four hypotheses to differentiate them:

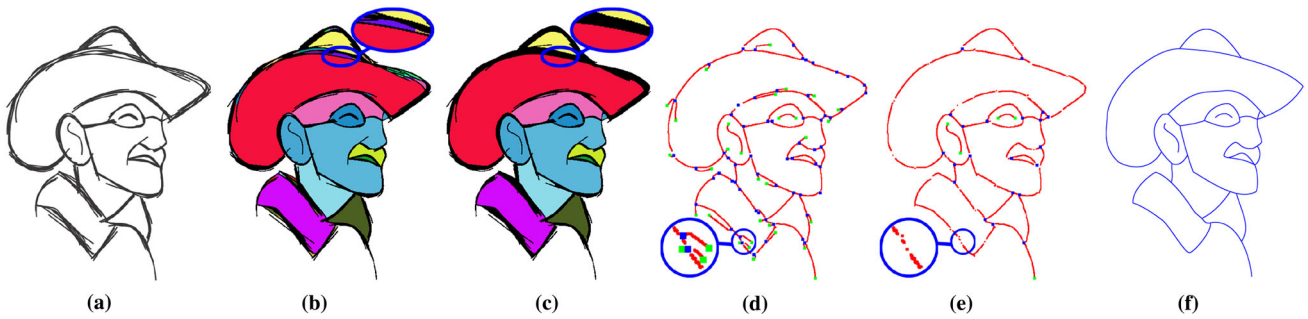


Fig. 1 Overview of our approach. **a** Input bitmap, **b**, over-segmentation, **c** region removal, **d** initial topology, **e** open curve removal and **f** our vectorization

Area Unexpected regions usually have small area, as they are generated by artists' repeated strokes in short distances;

Flatness Unexpected regions normally have flat shapes, not round ones, as they are formed by adjacent strokes with similar orientations;

Outer width Unexpected regions often have many adjacent strokes due to artists' sketching redundancy, these adjacent strokes make the boundaries of the unexpected regions have larger width than intended regions;

Independence Unexpected regions always appear jointly while intended regions are more likely independent, due to the repeatability of sketchy strokes.

The flatness, the area and the outer width hypotheses are constraints about every single region, while the independence hypothesis concerns the relationships between adjacent regions and their connected strokes. These four hypotheses are all reasonable, but none of them is always right in all cases. We first use the first third hypotheses as a joint optimization function instead of individual criteria:

$$\Phi_i = F_i \cdot A_i / W_i \tag{1}$$

where F_i, A_i, W_i are the flatness degree, the area and the average outer width of the ith background region. In our implementation, we traverse the boundary of each background region R_i as a point sequence $\{s_{i,j}\}$, $j \in [1, m_i]$ in a clockwise order, where m_i is the point count of this sequence. We then compute their normalized tangents $t_{i,j} = \mathbf{normalize}(s_{i,j+1} - s_{i,j-1})$, and rotate them 90° clockwise to get inward normals $n_{i,j} = \mathbf{rotate}(t_{i,j})$. As illustrated in Fig. 2c, we shot a ray from each boundary point $s_{i,j}$ along its inward normal $n_{i,j}$. This ray hits the other side of the region boundary, and we note the length of this ray as the inner length $l_{i,j}^{\text{inner}}$ for the point $s_{i,j}$. Similarly, we also shot a ray along the outward normal $-n_{i,j}$, note the outer length when this ray hits another region as $l_{i,j}^{\text{outer}}$ for the point $s_{i,j}$.

We compute the right part of Eq. 1 with the inner and outer length for each boundary point. The area of the ith background region is calculated as the average of inner length:

$$A_i = \sum_{j=1}^{m_i} l_{i,j}^{\text{inner}} / m_i \tag{2}$$

The flatness of the ith background region is calculated as the standard deviation of inner length:

$$F_i = \left(\sum_{j=1}^{m_i} (l_{i,j}^{\text{inner}} - A_i)^2 / m_i \right)^{1/2} \tag{3}$$

And the outer width of the ith background region is calculated as the average of outer length:

$$W_i = \sum_{j=1}^{m_i} l_{i,j}^{\text{outer}} / m_i \tag{4}$$

The background region R_i with small Φ_i is regarded as an unexpected one, as it is very flat, surrounded by many strokes or has a small area. We remove background region R_i and merge all its pixels into stroke region R_s if its Φ_i is less than the region threshold τ_r. We take τ_r as $0.4 - 0.9$ according to the stroke number and complexity. However, we found some of unexpected background regions were always left in our experiments. Their Φ_i are not small enough to be removed, as they have large areas, round shapes or few strokes embraced. And increasing the threshold τ_r can remove more unexpected background regions, but will lead to remove intended ones in the meantime.

Based on the independence hypothesis that has not been used in Eq. 1 yet, we propose an iterative optimization that updates Φ_i dynamically. In each iteration, we recompute W_i for all background regions left from the previous iteration, update Φ_i and remove regions whose updated Φ_i are less than τ_i. We iterate this removal and updating iteration until no more regions are removed. W_i becomes larger and Φ_i gets smaller if surrounding regions have been removed in

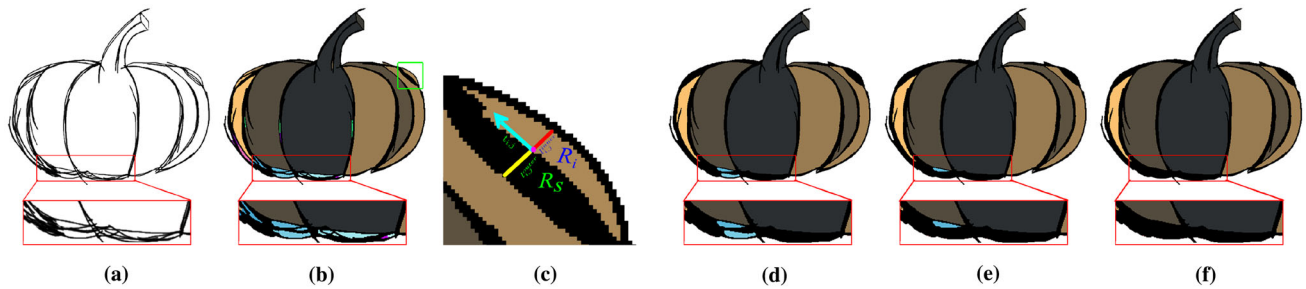


Fig. 2 Illustration of our iterative region removal process. **a** Input bitmap, **b** initial segmentation, **c** inner and outer length, **d** iteration #1, **e** iteration #2 and **f** iteration #3

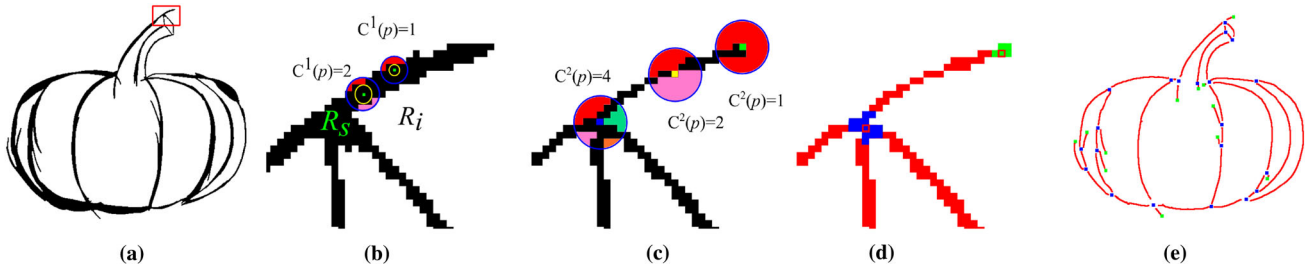


Fig. 3 Illustration of skeleton extraction and topology analysis. **a** Merged regions, **b** skeleton extraction, **c** skeleton points, **d** junctions and endpoints and **e** topological graph

the previous iteration; thus, our iteration optimization can successfully remove most of unexpected background regions without losing intended ones. Figure 2d–f shows examples where our optimization stops in 3 iterations. And 1–4 region removal iterations are required in most of our experiments.

4.2 Skeleton extraction

After the background region removal in the previous subsection, we are now ready to analyze the topology of line drawing. The first step is to extract the skeleton of the stroke regions. The main challenges of this step have threefold: one is the robustness against the stroke thickness variations, the second is the continuity of the skeleton, and the last is the centeredness of the skeleton. Many 2D skeleton extraction methods are proposed in the literature, but none of them solves these three challenges simultaneously to our knowledge. For instance, the iterative dilation employed by Favreau et al. removes all open curves that have to be added back in the postprocessing [10], and the minimum spanning tree used by Noris et al. requires a gradient field estimation which is inaccurate for strokes with varying thickness [18].

We propose an improved skeleton extraction based on the pearling method introduced by Whited et al. [26]. It can guarantee the continuity and centeredness of the skeleton and is insensitive to the stroke thickness variations. Note there are might several disconnected stroke regions, we explain here how to extract the skeleton for one connected stroke region R_s without loss of generality.

For each pixel p in the stroke region R_s, we put a circle centered at p with a growing radius. The growth will stop when this circle meets the boundary of the stroke region ∂R_s, we note this radius as $r(p)$. We then create an inspective circle at p with a radius $r(p)+2$ and collect all non-stroke pixels in this inspective circle. If the pixel p locates at the center of one stroke, its inspective circle will cover non-stroke pixels on both side of the stroke. If the pixel p does not locate at the center of one stroke, its inspective circle will only cover non-stroke pixels on one side of the stroke. Figure 3b illustrates a skeleton point on the left and a non-skeleton point on the right, and the yellow circle shows the maximal growing circle while the blue circle shows the inspective circle. Thirdly, we divide non-stroke pixels in every inspective circle into connective clusters by an 8-pixel-neighborhood connection algorithm. As illustrated in Fig. 3b, the left point has two connective clusters(red and pink) in its inspective circle, while the right point has only one connective cluster (red) in its inspective circle. Finally, we identify a skeleton point from the stroke region R_s by checking whether the number of its connective clusters $C^1(p)$ is larger than one or not:

$$label(p) = \begin{cases} \text{skeleton} & \text{if } C^1(p) > 1 \\ \text{non-skeleton} & \text{if } C^1(p) <= 1 \end{cases}$$

The set of skeleton points will form a fully connected skeleton network whose width is less than 2 pixels, as shown in Fig. 3c. To construct the whole topology, we have to detect feature points as well, such as junctions and endpoints. We apply the aforementioned neighborhood pixel clustering again for each skeleton points. In order to guarantee to cover both sides of the skeleton points, we put a circle at each skeleton point with a radius one pixel larger than the skeleton width (equals to 3 constantly), and check the number of connective clusters for non-skeleton points in this circle $C^2(p)$. We combine this number with $C^1(p)$ to identify junctions and endpoints as:

$$label(p) = \begin{cases} \text{endpoint} & \text{if } C^1(p) > 1 \,\&\&\, C^2(p) < 2 \\ \text{skeleton} & \text{if } C^1(p) > 1 \,\&\&\, C^2(p) = 2 \\ \text{junction} & \text{if } C^1(p) > 1 \,\&\&\, C^2(p) > 2 \\ \text{non-skeleton} & \text{if } C^1(p) <= 1 \end{cases}$$

Figure 3d shows an example of our junction and endpoint identification, green pixels are endpoints, and blue pixels are junctions. These pixels, noted as q_j, are not isolated points, but small regions contain several pixels due to our non-skeleton clustering method. Therefore, we first cluster them using a simple 8-pixel-neighborhood connection algorithm and compute an average position $\bar{q} = \sum_{q_j \in c_i} q_j / |c_i|$ for each cluster c_i. Secondly, we select the pixel in this cluster that is closest to each average position $v_i = \arg\min\{q_j \in c_i \big| ||q_j - \bar{q}||\}$ to obtain the representative point for this

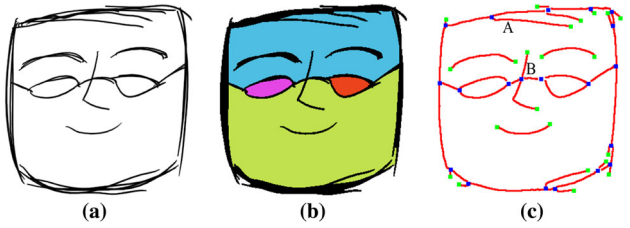


Fig. 4 An example of unexpected open curves. **a** Input bitmap, **b** regions after removal and **c** extracted skeleton

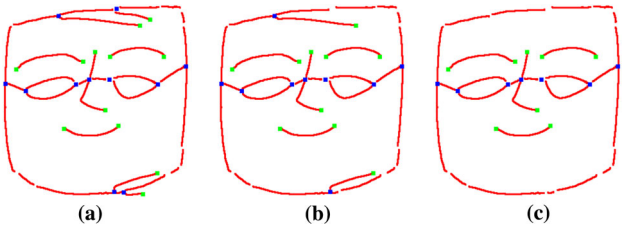


Fig. 5 Illustration of our iterative optimization for open curve removal. **a** Iteration #1, **b** iteration #2 and **c** iteration #3

endpoint or junction region. Points with red frames are the selected endpoint and junction in Fig. 3d. Our careful junction and endpoint selection improve the robustness of topology optimization and the accuracy of line vectorization in the subsequent processes.

Based on the accurate feature point positions, we narrow the skeleton to one pixel width using the depth-first traversal and construct a topological graph $G = \langle V, E \rangle$, where $V = \{v_i\}$ represents the set of all junctions and endpoints, and $E = \{e_i\}$ represents the set of all edges consisted of skeleton points among v_i. Figure 3e illustrates an example of our topological graph.

4.3 Open curve removal

Though we have removed unexpected background regions, the extracted topological graph still contains many unexpected open curves. Open curves are defined as edges connecting to endpoints in the topological graph G, and other edges that only connect to junctions on both sides are called closed curves. Many prior works decide the reservation of the open curves according to their length. However, length hypothesis is not always true. Figure 4 illustrates such a counterexample; the long open curve 'A' should be removed, while the short open curve 'B' should be preserved. In this paper, we improve the open curve selection by parallelism and relative length measurements instead of the absolute length. Firstly, open curves that are parallel to their adjacent closed curves are more likely to be produced by artist's casual scribbling and should be removed. We use the Hausdorff distance to depict this parallelism hypothesis:

$$\text{Hausdorff}(e_i, e_j) = \max_s \min_t |p_i(s) - p_j(t)| \tag{5}$$

where $p_i(s)$ and $p_j(t)$ are parameterized points in edges e_i and e_j separately, and e_i and e_j include all edges in the topological graph G. Equation 5 first calculates the minimal Euclidean distance from point $p_i(s)$ in the edge e_i to all points in the edge e_j, and then takes the maximal distance for all $p_i(s)$ as the Hausdorff distance. Note that $\text{Hausdorff}(e_i, e_j) \neq \text{Hausdorff}(e_j, e_i)$ as the maximum and minimum computations are not interchangeable.

Every open curve connects at least one closed curve, and open curves that are connected to long closed curves are more likely to be unexpected than ones connected to short closed curves. We thus improve Eq. 5 using a relative distance measurement that divides Hausdorff distance by the length of the connected closed curve: $\text{Hausdorff}(e_i, e_j)/\text{length}(e_j)$. And the reservation of the open curve e_i is defined as the minimum among all of its relative Hausdorff distances with adjacent closed curves:

$$S(e_i) = \min_j \frac{\text{Hausdorff}(e_i, e_j)}{\text{length}(e_j)} \tag{6}$$

We remove the open curves whose $S(e_i)$ is less than a curve threshold τ_p. τ_p is set in the range of $[0.1, 0.5]$ in our experiments. The junction point degenerates to a common skeleton point once its connected open curve is removed, and the other two curves connected to this junction point will be merged as a new curve. Therefore, both $V = \{v_i\}$ and $E = \{e_i\}$ are upgraded after unexpected open curve removal, and we recompute the open curve reservation formula and remove more unexpected open curves in an iterative manner. Merged closed curves are longer than any of the two original ones, the reservation values $S(e_i)$ of their adjacent remaining open curves become smaller. We keep the threshold τ_p constant, and remove open curves in each iteration if their reservation values $S(e_i)$ become less than τ_p. The iteration will terminate when no more open curves are removed. Our iterative strategy removes most of unexpected open curves, even if they are too long to be identified by the simple length judgement. Figure 5 shows all iterations of the open curve removal for the input shown in Fig. 4. Normally, about 1-4 iterations are sufficient in all our experiments.

5 Vectorization

The open curve removal process simplifies the topological graph to a new one $G' = \langle V', E' \rangle$, in which all unexpected open curves, junctions and endpoints are removed. We can now convert the remaining curves $E' = \{e'_i\}$ into parameterized curves. We first detect turning points in each e'_i by 2D curve curvatures. The curvature for each point p_j is com-

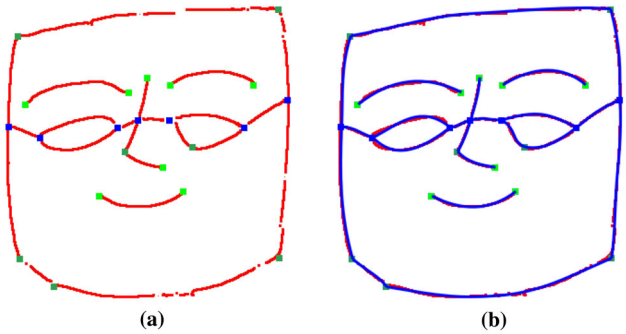


Fig. 6 Illustration of our curve fitting result. **a** Turning point in dark green and **b** Bézier curves in blue

puted by $\sigma_j = (p_{j-\delta} - p_j) \cdot (p_{j+\delta} - p_j)/||p_{j-\delta} - p_j|| \cdot ||p_{j-\delta} - p_j||$, where δ is a sampling interval parameter, set to 5 in all our experiments. Figure 6a illustrates all detected turning points in dark green.

Turning points separate each edge e_i' into more segments, and we fit each segment by a cubic Bézier curve $f(s) = \sum_{i=0}^{3} f_i \cdot s^i$ separately. We first reformulate it by $f(t) = \sum_{i=0}^{3} f_i \cdot t^i(1-t)^{3-i}$, where f_0 and f_3 can be directly decided by the positions of its segment terminals. Secondly, we sample curve points p_j within this segment, and compute f_1 and f_2 by solving a least square equation. Coefficient parameters f_i are computed independently for horizontal and vertical coordinates.

We employ the Hausdorff distance between the original line segment and the fitted Bézier curve to calculate the fitting error. If the fitting error is larger than a threshold τ_f, we divide this segment into two parts with equal lengths, and fit them again. This fitting-and-dividing iteration will stop when the errors of all segments are less than τ_f. We take τ_f as 4 pixels in all our experiments. Finally, we average tangential directions on both sides of dividing terminals to preserve the G^1 continuity of each edge. To preserve the G^1 continuity at the junctions, we estimate a continuity level between arbitrary two connected branches using the angle of their tangential direction difference at the junction. If this angle is larger than 140° and other angles at this junction, we take these two branches as one single edge in the fitting step to enforce their continuity at the junction. Note that for the sake of curve simplicity, we always fit each segment by lower-degree curves (straight lines and quadric Bézier curves) before the cubic Bézier fitting, and accept them according to their fitting errors. Figure 6 shows our vectorization result on the top of the topological graph, and demonstrates the accuracy of our fitting method.

6 Results and discussion

Topology comparison Topology optimization is the core of our work. Figure 7 compares our topology with skeletons extracted by Favreau et al.'s method [10] and the simplified line drawing bitmap by Simo-Serra et al. [23]. Figure 7b, c shows skeleton results produced by the software released on authors' homepage. Figure 7b uses a minimal region size 7, and can not successfully remove all unexpected regions, like the two in the green rectangle. Figure 7c uses a minimal region size 8. It removes the smaller unexpected region in the green rectangle, but also removes the necklace region in the pink rectangle.

Favreau et al. discard all open curves in their region-based skeleton extraction and add them back as pixels at a distance greater than the closest skeleton point. However, single distance thresholding fails to preserve the wanted open curves if long unexpected open curves exist. In the green ellipses of Fig. 7b, unexpected open curves on the top of the head have not been removed, while wanted open curves of eyebrows are removed. Favreau et al.'s hypergraph exploration can simplify final vector curves, but hardly correct topological errors in the initial skeleton.

Figure 7d, e shows line drawing image simplification results proposed by Edger et al. [23]. They are produced by authors' web-based application using simplification degrees 300 and 400 separately. Figure 7d does not preserve the necklace detail, and Fig. 7d, e generates many broken curves, which will be difficult to connect in the vectorization stage.

Our method takes flatness, area, outer width, independence hypotheses for the region removal and parallelism, relative distance hypotheses for the open curve removal, optimizes the topology graph in an iterative manner and thus removes all unexpected regions and open curves while preserving most structural details. Figure 7f illustrates our improved skeleton and final vectorization. Note that gaps in Fig. 7f are produced by open curve removal, they do not exist in the topological graph, and thus naturally filled up by our vectorization method, as shown in Fig. 7g.

Vectorization comparison Figure 8 compares our method with most existing line drawing vectorization methods. Figure 8b shows results produced by Adobe Illustrator CC, which contain many unexpected curves. Noris et al. focus on the clean line drawings, fail to simplify the sketchy line drawings, shown in Fig. 8c. Favreau et al. improve the simplification by skeleton extraction, but they still produce errors when input line drawings are rough, as pointed out by red ellipses in Fig. 8d. Figure 8e shows Simo-Serra et al.'s image simplification results; they neither simplify the input into single curves in many places nor produce vectorization outputs. Figure 8f shows Liu et al.'s stroke simplification results that are close to the intention of the artist, but their method requires stroke information, cannot apply to the rasterized line drawing images. Our method can achieve competitive vectorization results without stroke information; our improved topology optimization has removed all unexpected

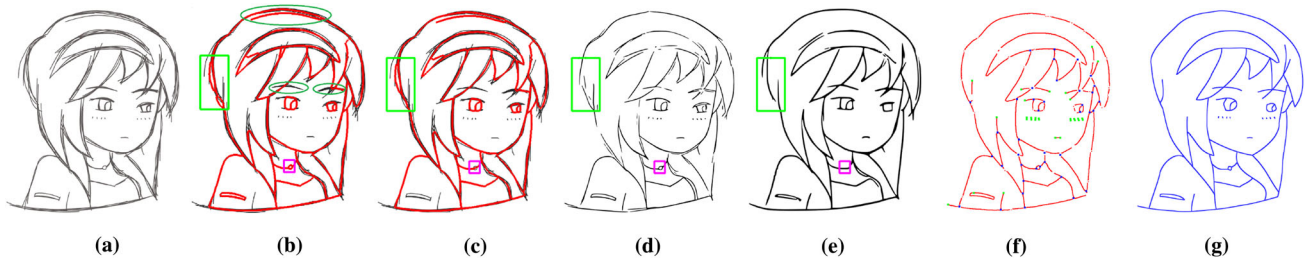


Fig. 7 Topology comparison with state-of-the-art methods Favreau et al. [10] and Simo-Serra et al. [23]. **a** Input bitmap, **b** Favreau et al. Min region size = 7, **c** Favreau et al. Min region size = 8, **d** Simo-Serra et al. Simp. degree = 300, **e** Simo-Serra et al. Simp. degree = 400, **f** our topology graph and **g** our vectorization

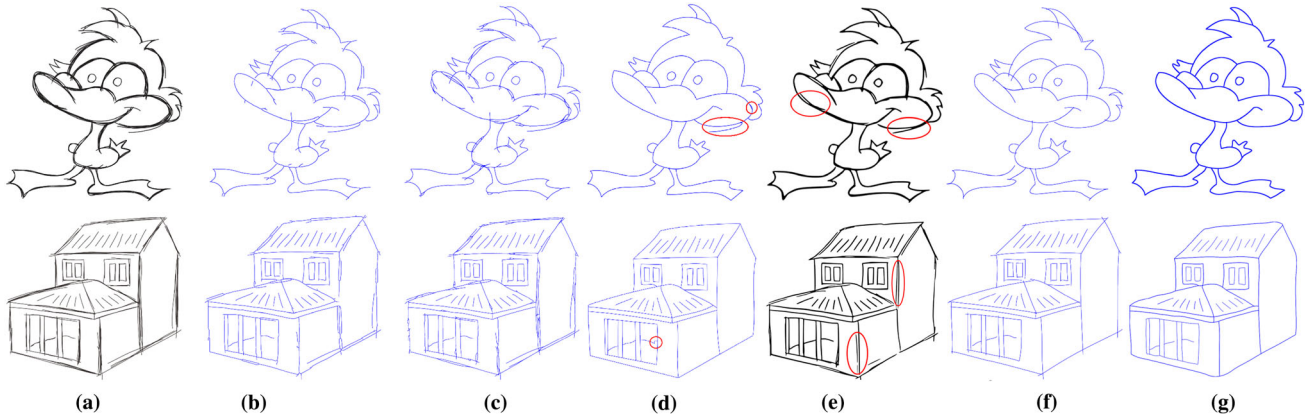


Fig. 8 Vectorization comparison with existing vectorization methods. **a** Input bitmap, **b** adobe illustrator CC, **c** Noris et al., **d** Favreau et al., **e** Simo-Serra et al., **f** Liu et al. and **g** our result

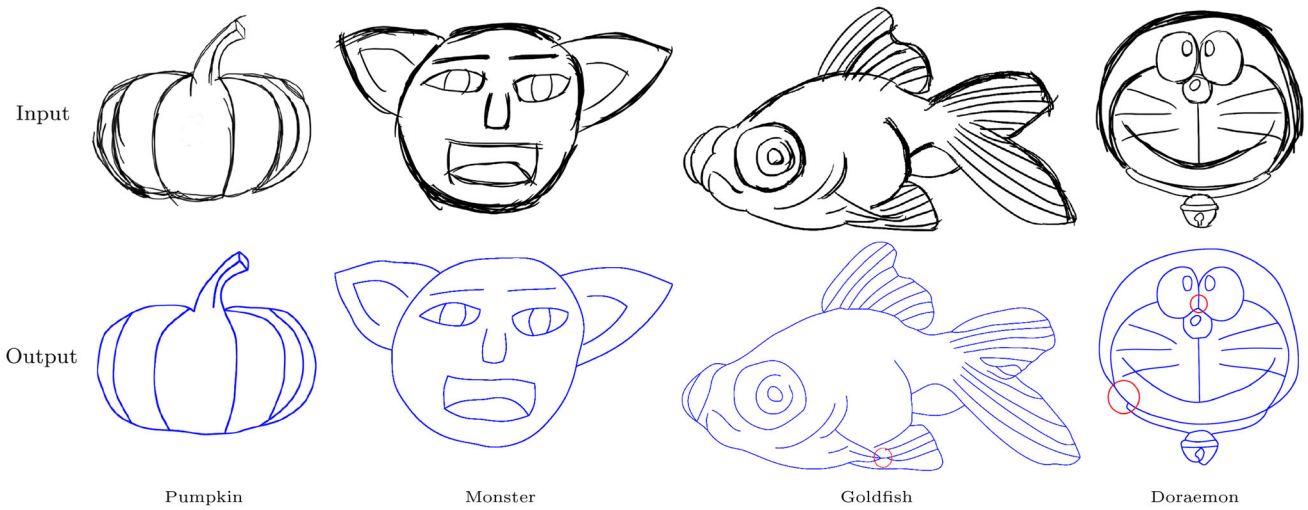


Fig. 9 More results using our vectorization method and some failure cases in red circles

open curves and regions even they are long or large, although illustrated in Fig. 8g.

Figure 9 shows more results of our vectorization method, the roughness of line drawings demonstrates the validity for very rough sketchy line drawings. And Fig. 10 illustrates the robustness of our method for sketchy line drawings with varying thickness. Figure 10a is drawn using strokes with constant large thickness, and Fig. 10c is drawn using strokes with varying thickness; their results are both good and similar.

Parameters Our method has two major parameters to tune: τ_r and τ_p. They control the simplification level of regions

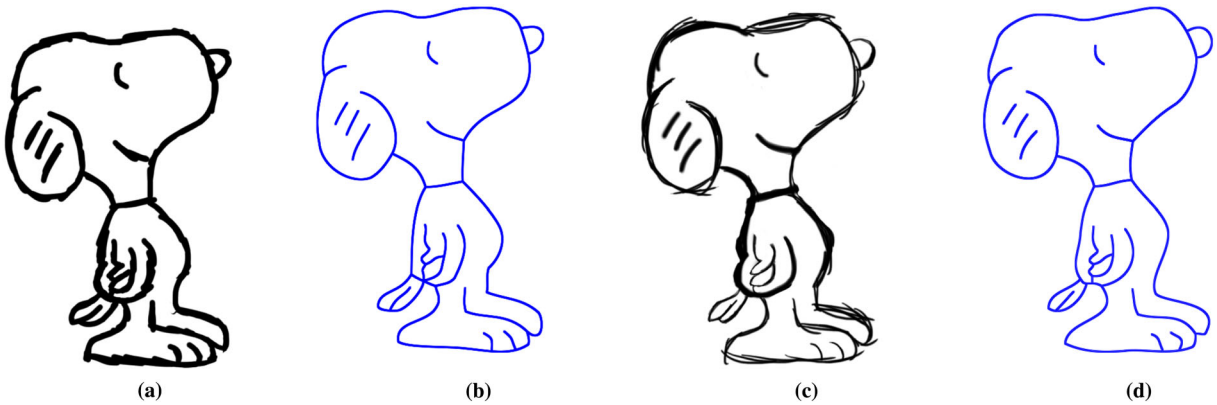


Fig. 10 Our results for line drawings drawn by strokes with large and varying thickness. **a** Snoopy bitmap 1 with large thickness, **b** our result for large thickness, **c** Snoopy bitmap 2 with varying thickness and **d** our result for varying thickness

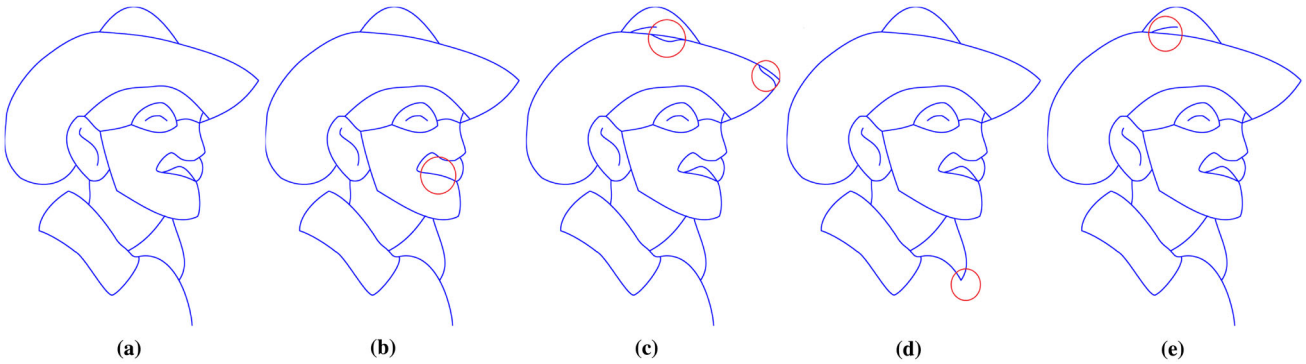


Fig. 11 Our vectorization results with different parameters τ_r and τ_p. **a** $\tau_r = 0.5, \tau_p = 0.2$, **b** $\tau_r = 0.55, \tau_p = 0.2$, **c** $\tau_r = 0.45, \tau_p = 0.2$, **d** $\tau_r = 0.5, \tau_p = 0.85$ and **e** $\tau_r = 0.5, \tau_p = 0.15$

Table 1 Statistics of topology count, times and parameters

Figures	Topology count		Times (s)				Parameters	
	Regions	Open curves	Region opti.	Open curve opti.	Vectorization	Total	τ_r	τ_p
Duck	33 → 14	32 → 2	0.22	0.25	1.39	4.1	0.4	0.3
Gentleman	83 → 12	34 → 1	0.35	0.18	0.85	2.6	0.5	0.2
Pumpkin	28 → 7	15 → 2	0.1	0.1	0.2	1	0.5	0.1
Face	67 → 4	14 → 2	0.5	0.17	0.51	2.6	0.5	0.2
House	51 → 22	26 → 2	0.58	0.47	1.11	5.1	0.5	0.35
Girl	173 → 19	43 → 9	1.06	0.34	1.21	5.0	0.7	0.3
Doraemon	100 → 15	14 → 6	0.49	0.25	2.23	5.8	0.5	0.1
Goldfish	95 → 20	36 → 14	0.94	0.47	1.65	5.7	0.9	0.1
Monster	75 → 11	35 → 0	0.54	0.33	0.88	4.9	0.6	0.5
Lantern	49 → 32	25 → 3	0.66	0.25	0.89	1.8	0.4	0.1
Snoopy 1	12 → 5	25 → 9	0.3	0.22	0.72	1.3	0.4	0.1
Snoopy 2	34 → 5	64 → 9	0.49	0.31	0.69	1.6	0.5	0.1

and open curves separately. Figure 11 shows examples using different τ_r and τ_p values for the Gentleman line drawing. The best result is achieved when $\tau_r = 0.5$ and $\tau_p = 0.2$. Increasing τ_r to 0.55 results in removing the wanted regions, and decreasing τ_r to 0.45 leads to leave unexpected regions. Similarly, increasing τ_p to 0.85 results in removing wanted open curves, and decreasing τ_p to 0.15 leads to leave unexpected open curves. Table 1 shows all parameters we use in this paper, which takes τ_r in range $[0.4, 0.9]$ and τ_p in range $[0.1, 0.5]$. Note that these parameters only impact a

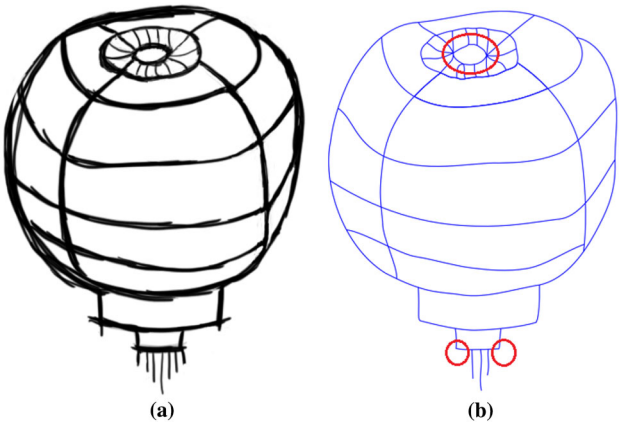


Fig. 12 A failure case. **a** Input bitmap and **b** our result

few regions and open curves, since our iterative optimization solves most issues.

Performance Our method is fully automatic; all experiments are conducted on laptop computer with 2.60 GHz CPU, 8GB system memory. The second and third rows in Table 1 show the quantitative changes of regions and open curves. Though iterative optimizations are involved for both region and open curve removal, a few of iterations are required and only part of the topological graph has to be recomputed in each iteration. As shown in the middle rows in Table 1, two iteration processes only take about 1 second, and the whole computation can be done within 6 seconds, which is faster than most existing line drawing vectorization methods. Our current program is fully implemented on CPU, and it is also possible to reach interactive or even realtime performance using GPU acceleration in the future.

Limitations Our topology simplification relies on the image segmentation in the initialization stage. The trapped-ball algorithm we have employed is insensitive to small gaps between adjacent regions, but fails to connect large gaps. And it is also problematic for separating strokes when they are closed, as the trapped-ball algorithm tends to merge them, like the red circles in Fig. 9. Our assumptions for region and open curve removal priority are violated in some special cases. Figure 12 illustrates such a failure case, where close short strokes are either removed or connected improperly. Taking machine learning techniques may be a good solution to better handle these issues, like Simo-Serra et al.'s work [21,23].

Our method only has two parameters, which are intuitive enough for the user to understand. However, the user still has to tune it manually to achieve ideal results. Since the open curve removal is optimized after the region removal, the region threshold parameter τ_r also impacts the open curve optimization. Therefore, τ_r and the curve threshold parameter τ_p are not fully independent, τ_p has to be re-tuned once τ_r is adjusted sometimes.

7 Conclusion

In this paper, we present an improved topology extraction approach for vectorization of sketchy line drawings. The key idea is a unified optimization mechanism to remove unexpected regions and open curves. The input bitmap is first divided into separated regions by a gap-insensitive image segmentation, and unexpected regions are removed iteratively according to artists' drawing customs. We then apply a skeleton extraction to construct an initial topological graph. Our skeleton extraction automatically identifies endpoints and junctions without discarding any open curve. Our iterative optimization mechanism contributes again here to remove unexpected open curves to further optimize the topology. Comparisons with state-of-the-art methods demonstrate our improved topology extraction method can vectorize very rough sketchy line drawing robustly and efficiently.

Our method lacks of high-level understanding of the line drawing. We plan to employ the hypergraph exploration method proposed by Favreau et al. to improve the curve simplicity in the future [10]. And we also believe that employing machine learning techniques can help our method simplify the structural topology on a semantic level. Our idea of iterative topology simplification has great potential for other contents, such as image vectorization [29], videos [25] and 3D models [14] as well. These new contents may raise new challenges; we hope more researchers can investigate this research direction together in the future.

Compliance with ethical standards

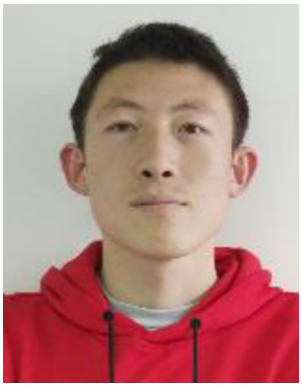
Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bao, B., Fu, H.: Vectorizing line drawings with near-constant line width. In: Proceedings of the 19th IEEE International Conference on Image Processing, pp. 805–808. Orlando, Florida, USA (2012). https://doi.org/10.1109/ICIP.2012.6466982
2. Barla, P., Thollot, J., Sillion, F.X.: Geometric clustering for line drawing simplification. In: Proceedings of the 16th Eurographics Conference on Rendering Techniques, pp. 183–192. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2005). https://doi.org/10.2312/EGWR/EGSR05/183-192
3. Bartolo, A., Camilleri, K.P., Fabri, S.G., Borg, J.C.: Line tracking algorithm for scribbled drawings. In: Proceedings of the 3rd International Symposium on Communications, Control and Signal Processing, pp. 554–559. IEEE (2008). https://doi.org/10.1109/ISCCSP.2008.4537287
4. Bartolo, A., Camilleri, K.P., Fabri, S.G., Borg, J.C., Farrugia, P.J.: Scribbles to vectors: preparation of scribble drawings for CAD interpretation. In: Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling, pp. 123–130. ACM,

New York, NY, USA (2007). https://doi.org/10.1145/1384429.1384456
5. Bo, P., Luo, G., Wang, K.: A graph-based method for fitting planar b-spline curves with intersections. J. Comput. Des. Eng. **3**(1), 14–23 (2016). https://doi.org/10.1016/j.jcde.2015.05.001
6. Bonnici, A., Camilleri, K.: A circle-based vectorization algorithm for drawings with shadows. In: Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling, pp. 69–77. ACM, New York, NY, USA (2013). https://doi.org/10.1145/2487381.2487386
7. Bonnici, A., Camilleri, K.P.: Scribble vectorization using concentric sampling circles. In: Proceedings of the 3rd International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 89–94 (2009). https://doi.org/10.1109/ADVCOMP.2009.20
8. Chen, J., Guennebaud, G., Barla, P., Granier, X.: Non-oriented MLS gradient fields. Comput. Graph. Forum **32**(8), 98–109 (2013). https://doi.org/10.1111/cgf.12164
9. Chen, J., Lei, Q., Miao, Y., Peng, Q.: Vectorization of line drawing image based on junction analysis. Sci. China Inf. Sci. **58**(7), 1–14 (2015). https://doi.org/10.1007/s11432-014-5246-x
10. Favreau, J.D., Lafarge, F., Bousseau, A.: Fidelity versus simplicity: a global approach to line drawing vectorization. ACM Trans. Graph. **35**(4), 120:1–120:10 (2016). https://doi.org/10.1145/2897824.2925946
11. Grabli, S., Durand, F., Sillion, F.X.: Density measure for line-drawing simplification. In: Proceedings of the 12th Pacific Conference on Computer Graphics and Applications, pp. 309–318 (2004). https://doi.org/10.1109/PCCGA.2004.1348362
12. Hilaire, X., Tombre, K.: Improving the accuracy of skeleton-based vectorization. In: Proceedings of the Fourth International Workshop on Graphics Recognition Algorithms and Applications. Lecture Notes in Computer Science, vol. 2390, pp. 273–288. Springer, Berlin (2002). https://doi.org/10.1007/3-540-45868-9_24
13. Hilaire, X., Tombre, K.: Robust and accurate vectorization of line drawings. IEEE Trans. Pattern Anal. Mach. Intell. **28**(6), 890–904 (2006). https://doi.org/10.1109/TPAMI.2006.127
14. Huang, H., Wu, S., Cohenor, D., Gong, M., Zhang, H., Li, G., Chen, B.: L1-medial skeleton of point cloud. ACM Trans. Graph. **32**(4), 65 (2013). https://doi.org/10.1145/2461912.2461913
15. Kyprianidis, J.E., Kang, H.: Image and video abstraction by coherence-enhancing filtering. Comput. Graph. Forum **30**(2), 593–602 (2011). https://doi.org/10.1111/j.1467-8659.2011.01882.x
16. Liu, X., Wong, T.T., Heng, P.A.: Closure-aware sketch simplification. ACM Trans. Graph. **34**(6), 168:1–168:10 (2015). https://doi.org/10.1145/2816795.2818067
17. Nieuwenhuizen, P.R., Kiewiet, O., Bronsvoort, W.F.: An integrated line tracking and vectorization algorithm. Comput. Graph. Forum **13**(3), 349–359 (1994). https://doi.org/10.1111/1467-8659.1330349
18. Noris, G., Hornung, A., Sumner, R.W., Simmons, M., Gross, M.: Topology-driven vectorization of clean line drawings. ACM Trans. Graph. **32**(1), 4:1–4:11 (2013). https://doi.org/10.1145/2421636.2421640
19. Preim, B., Strothotte, T.: Tuning rendered line-drawings. In: Proceedings of Winter School of Computer Graphics, vol. 3, no. 1–2, pp. 228–238 (1995)
20. Saha, P.K., Borgefors, G., di Baja, G.S.: A survey on skeletonization algorithms and their applications. Pattern Recogn. Lett. **76**, 3–12 (2016). https://doi.org/10.1016/j.patrec.2015.04.006
21. Sasaki, K., Iizuka, S., Simo-Serra, E., Ishikawa, H.: Joint gap detection and inpainting of line drawings. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2017). https://doi.org/10.1109/CVPR.2017.611
22. Silver, D., Cornea, N.D., Min, P.: Curve-skeleton properties, applications, and algorithms. IEEE Trans. Vis. Comput. Graph. **13**, 530–548 (2007). https://doi.org/10.1109/TVCG.2007.1002
23. Simo-Serra, E., Iizuka, S., Sasaki, K., Ishikawa, H.: Learning to simplify: fully convolutional networks for rough sketch cleanup. ACM Trans. Graph. **35**(4), 121:1–121:11 (2016). https://doi.org/10.1145/2897824.2925972
24. Sun, J., Liang, L., Wen, F., Shum, H.Y.: Image vectorization using optimized gradient meshes. ACM Trans. Graph. **26**(3), 11–18 (2007). https://doi.org/10.1145/1276377.1276391
25. Wang, C., Zhu, J., Guo, Y., Wang, W.: Video vectorization via tetrahedral remeshing. IEEE Trans. Image Process. **26**(4), 1833–1844 (2017). https://doi.org/10.1109/TIP.2017.2666742
26. Whited, B., Rossignac, J., Slabaugh, G., Fang, T., Unal, G.: Pearling: stroke segmentation with crusted pearl strings. IEEE Pattern Recognit. Image Anal. **19**(2), 277–283 (2009). https://doi.org/10.1134/S1054661809020102
27. Wilson, B., Ma, K.L.: Rendering complexity in computer-generated pen-and-ink illustrations. In: Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, NPAR '04, pp. 129–137. ACM, New York, NY, USA (2004). https://doi.org/10.1145/987657.987674
28. Xia, T., Liao, B., Yu, Y.: Patch-based image vectorization with automatic curvilinear feature alignment. In: ACM SIGGRAPH Asia 2009 Papers, SIGGRAPH Asia '09, pp. 115:1–115:10. ACM, New York, NY, USA (2009). https://doi.org/10.1145/1661412.1618461
29. Xie, G., Sun, X., Tong, X., Nowrouzezahrai, D.: Hierarchical diffusion curves for accurate automatic image vectorization. ACM Trans. Graph. **33**(6), 230:1–230:11 (2014). https://doi.org/10.1145/2661229.2661275
30. Zhang, S.H., Chen, T., Zhang, Y.F., Hu, S.M., Martin, R.R.: Vectorizing cartoon animations. IEEE Trans. Vis. Comput. Graph. **15**(4), 618–629 (2009). https://doi.org/10.1109/TVCG.2009.9
31. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Commun. ACM **27**(3), 236–239 (1984). https://doi.org/10.1145/357994.358023

Jiazhou Chen received his double Ph.D. degrees in INRIA Bordeaux Sud-Ouest, France, and the State Key Laboratory of CAD and CG at Zhejiang University, China, in 2012. He is now an assistant professor in Zhejiang University of Technology. His research interests include computer graphics and visual media computing.



Mengqi Du is a Master's degree candidate in the College of Computer Science and Technology, Zhejiang University of Technology, China. His research interests include visual media computing and digital arts.

Xujia Qin born in 1968 is currently a professor at Zhejiang University of Technology, China. He received his PhD degree from Dalian University of Technology, China, in 2001. He worked as a postdoctor in State Key Laboratory of CAD and CG, Zhejiang University, China, in 2001–2003. His research interests include computer graphics and geometry modeling.

Yongwei Miao received his Ph.D. degree in computer graphics from the State Key Laboratory of CAD and CG at Zhejiang University in March 2007. From February 2008 to February 2009, he worked as a visiting scholar in the University of Zürich, Switzerland. From November 2011 to May 2012, he worked as a visiting scholar in the University of Maryland, USA. Dr. Miao is a professor in the College of Computer Science and Technology, Zhejiang University of Technology, China. His research interests include computer graphics, digital geometry processing, visual media computing, 3D reconstruction and computer vision.