SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2015, Vol. 58 072101:1–072101:14
doi: 10.1007/s11432-014-5246-x

Vectorization of line drawing image based on junction analysis

CHEN JiaZhou[1*], LEI Qi[2], MIAO YongWei[1] & PENG QunSheng[2]

[1]*College of Computer Science, Zhejiang University of Technology, Hangzhou* 310023*, China;*
[2]*State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou* 310058*, China*

Received February 7, 2015; accepted April 22, 2015 ; published online May 14, 2015

Abstract Converting a scanned or shot line drawing image into a vector graph can facilitate further edit and reuse, making it a hot research topic in computer animation and image processing. Besides avoiding noise influence, its main challenge is to preserve the topological structures of the original line drawings, such as line junctions, in the procedure of obtaining a smooth vector graph from a rough line drawing. In this paper, we propose a vectorization method of line drawings based on junction analysis, which retains the original structure unlike done by existing methods. We first combine central line tracking and contour tracking, which allows us to detect the encounter of line junctions when tracing a single path. Then, a junction analysis approach based on intensity polar mapping is proposed to compute the number and orientations of junction branches. Finally, we make use of bending degrees of contour paths to compute the smoothness between adjacent branches, which allows us to obtain the topological structures corresponding to the respective ones in the input image. We also introduce a correction mechanism for line tracking based on a quadratic surface fitting, which avoids accumulating errors of traditional line tracking and improves the robustness for vectorizing rough line drawings. We demonstrate the validity of our method through comparisons with existing methods, and a large amount of experiments on both professional and amateurish line drawing images.

Keywords vectorization, line drawing, line tracking, junction analysis, vector graph

Citation Chen J Z, Lei Q, Miao Y W, et al. Vectorization of line drawing image based on junction analysis. Sci China Inf Sci, 2015, 58: 072101(14), doi: 10.1007/s11432-014-5246-x

1 Introduction

Line is ubiquitous in art. It is not only an effective tool for depicting shapes by painters, but also an essential element for conveying ideas by animation designers. Although digital painting devices, such as capacitance pens, touch screens and electronic tablet, are popular in recent times, most artists still use pen-and-paper rubbing, especially for line drawing. However, lines on real papers usually take a lot of repetitive drawing to finish, and barely support further editing or reuse. Fortunately, the computer technology provides the possibility to increase the efficiency of line drawing creation, and facilitates unlimited editing and reuse. To achieve this goal, we have to scan the line drawing paper into a computer, and convert it to a vector graph using the computing power of this computer, which is also known as vectorization. Besides further editing and reuse, vectorization of line drawing images can also largely

* Corresponding author (email: cjz@zjut.edu.cn)

reduce their storage size, which is critical for computer animation. It is also a requisite step for many sketch-based applications in computer graphics, such as image/object searching [1,2], image vectorization [3] and storytelling [4] and animations [5,6].

The vectorization of line drawing is to convert its representation form a bitmap to a vector graph, that preserves all lines in the original image, including their position and topological structures. Such an automatic vectorization task seems easily feasible for our human eyes, but introduces big challenges for computers, the reasons of which can be concluded into three parts:

Firstly, lines in paper-based line drawing images are not as clear as we expected. They are probably of different width, with gaps and messy strokes. Extracting smooth vector graph from a coarse line drawing image tends to introduce disconnection and lose details. Noises contained in the input image and errors introduced in the scanning procedure further increase the difficulty of line drawing vectorization. Erosion operation in morphological methods can shrink lines in the scanned image into one pixel width, but it costs a lot of computation and always shortens lines [7]; recognition-based methods are much robust and efficient, but only suitable for simple geometry primitives, such as straight lines and arcs [8–12].

Secondly, lines in a line drawing image may have a very complex topological structure. For instance, a single line may contain featured points like endpoints and L-shape corners; the intersection of different lines may produce other junction types, such as T-shape, Y-shape or X-shape junctions. Existing methods can only detect the position of feature points and junction points, together with the number of adjacent branches, but do not precisely analyze the topological relationship between adjacent line branches, thus fails to distinguish T-shape and Y-shape junctions for instance [13–17].

Thirdly, the final vectorizing result is not unique. One single line drawing can be represented by different valid vector graphs. Strictly speaking, the quality of vectorization could only be fairly evaluated by the original designer, which is hardly possible for some antique line drawing work. Therefore, existing vectorization methods are actually based on the following two uniqueness assumptions: valid vectorization should preserve the topological structure of the original line drawing [18], and the result vector graph should locate at the middle position of the original line drawings.

To overcome the aforementioned issues, we first propose a fitting-based correction mechanism for line tracking in this paper, to alleviate the accumulating error in traditional line tracking, and further improve the smoothness and accuracy of the resulting vector graph. Secondly, we introduce a junction analysis approach for line drawing images. Our analysis approach can not only detect the position of junctions and the number and orientation of their branches, but also determine the smoothness between adjacent branches of each junction, which ensures the preservation of topological structures. A large number of experiments on both professional and amateurish line drawing images demonstrate the robustness of our approach, which reveals its competence for rough line drawing images and reconstructing structure-preserving vector graphs.

2 Related work

Existing vectorization methods basically deal with two kinds of images: natural images and line drawing ones. Vectorization of natural images usually first segment the input image based on color similarity, and then represent each segmented region using uniform colors or transitional colors [19,20]. Meanwhile, vectorization of line drawing images consists of extracting lines from the input image and converting them to vector graphs. Both kinds of vectorization take bitmap images as input, but they differ from the content and goal of vectorization. In this paper, we focus on the vectorization of line drawing images due to the limited spaces, and classify them to three categories: recognition-based methods, tracking-based methods and topology-driven ones.

2.1 Recognition-based methods

The work of vectorizing line drawing images stems from converting scanned paper-based engineering drawings to electronic diagrams. Since these engineering drawings usually consist of regular shapes, early

vectorization methods mainly focus on recognizing these geometric primitives from the input image. For instance, Bonnici and Camilleri [11,12] employed multiple concentric sampling circles to extract straight lines and compute their location parameters; Chiang et al. [21] proposed a new approach based on maximal inscribing circles, that obtains positions, orientations and width of straight lines; Halaire and Tombre [8,10] separated the input binary image into layers of homogeneous thickness and extracted skeletons in each layer for improved accuracy in the primitive recognition.

Although recognition-based methods are very accurate, they can only deal with simple primitives, such as straight lines and arcs. To avoid this limitation, fitting-based methods are employed to recognize more complicated primitives, like ellipses [19] and Bézier curves [22]. However, they are still limited to particular primitives, thus can not recognize free-form line drawings.

2.2 Tracking-based methods

To vectorize any line drawings with arbitrary complexities, tracking-based methods extracted all points located in the center of lines from the input image, and connected adjacent points to form a path, which is considered as a vectorization result. To find these central points in thick lines, Liu et al. [23] and Rajan et al. [24] employed an erosion operations from morphology methods, which can shrink the line width to one pixel; Zou and Yan [13,14] segmented the regions covered by the line drawings into a set of non-overlapping triangles, and connected the centroid of adjacent triangles to get vectorization result; Liu and Dori [15,16] proposed a sparse pixel vectorization method, that selectively connected the midpoints of horizontal and vertical runs; Whited et al. [17] introduced a pearling method that computed the centerlines of the strokes, bifurcations, and the thickness function along each stroke. Similarly, Yang et al. [25–28] makes use of scan line analysis, Lu et al. [29] matches contour lines, and Wong et al. [27] slides windows in adaptive segmentation to get line skeleton. Although these methods can vectorize any line drawings with arbitrary complexity and thickness, they usually cost a lot of computation, and fail to get correct results around junctions or rough line drawings.

Another type of tracking-based methods is based on the convolution of vector field. Convolution-based methods first calculate a tangential vector for each pixel in the image, and then obtain streamlines using a line integral convolution along tangential directions. For instance, Bartolo et al. [30] first employed Gabor filter to get directions of each pixel in the input image, and then traced central paths through a Kalman filter; instead of directly tracing central points, Nieuwenhuizen et al. [31] traced the contour lines on both sides of the line drawings, while taking the midpoints as tracking paths.

Comparing with the first type of tracking-based methods, convolution-based methods cost much less computation, but tend to produce accumulating errors in the procedure of vector field convolution. To overcome this issue, Kyprianidis and Kang [32] used Ronge-Kutta integration instead of Euler integration; Bartolo et al. [33] introduced a correction mechanism in convolution procedure to make sure that convoluted points place in the center of line drawings; Wang et al. [34] proposed a gradient-guided warping technique to correct line centers. In this paper, we present an improved convolution-based approach, that introduces a quadratic surface fitting to better reduce the accumulating errors, and improves the final vectorization accuracy.

2.3 Topology-driven methods

In the vectorization process, one of the main requirements is to maintain the same topological structure as the one in the original image [18]. Among these topological structures, junction is the most important one [35]. Recognition-based methods can easily find junctions by elongating adjacent geometry primitives [12]. Based on this observation, Hilaire et al. [8] and Pham et al. [35] further employed an optimization solver to identify the junction types, such as T-shape, Y-shape and X-shape. But, recognition-based methods are still limited to simple geometry primitives like straight lines or arcs. On the other side, tracking-based methods usually need a unique direction to move forwards or backwards during tracking, which is not the case when more than three branches exist at junctions. The directional singularity property near junction points makes tracking-based methods incapable for detecting

junction positions, branch numbers and other important information of topological structures. Although postprocessing like principal component analysis can be employed to analyze the branch numbers, such approaches heavily rely on the precise position of the junction center, which is very difficult to determine when the branch number is unknown.

To overcome the aforementioned structure-preserving issues in recognition-based and tracking-based methods, Noris et al. [18] proposed a topology-driven approach. The topological structure, depicted as a minimum spanning tree, is first extracted using a gradient-based pixel clustering technique, following a reconstruction of all line drawings by a centerline extraction procedure. To avoid the directional singularities around the junctions, this approach identifies an ambiguous region for each junction, removes all drawing information inside this region, and applies "reverse drawing" using drawing information only outside this ambiguous region to preserve topological structures.

However, although line drawings in ambiguous regions contain directional singularities, their contours can produce stable cues to determine smoothness between adjacent branches. Based on this observation, we introduce an improved junction analysis approach in this paper, that is able to obtain a smooth and structure-preserving vectorization results. To extend our processing object from clean line drawing images as in [18] to rough ones, we further present a tracking correction mechanism based on quadratic surface fitting.

3 Line tracking

Line tracking is a procedure of finding a trajectory starting from an initialized point whilst guided by a vector field. We first compute a tangential field from the input line drawing image; and then starting from a central point, we do line integral convolution to get the vectorization result. To get smooth but accurate line tracking result, we employ color tensor to construct tangential field, and introduce a correction mechanism for central position to reduce accumulating errors in the procedure of line tracking.

3.1 Tangential field computation based on color tensor

Firstly, we use Sobel filter to compute both horizontal and vertical color first-order directional derivatives of the input images:

$$I_x = (R_x, G_x, B_x), \quad I_y = (R_y, G_y, B_y). \tag{1}$$

Then, a color gradient tensor could be constructed using these two directional derivatives:

$$T = \begin{pmatrix} I_x \cdot I_x & I_x \cdot I_y \\ I_y \cdot I_x & I_y \cdot I_y \end{pmatrix}. \tag{2}$$

Here, "$\cdot$" represents inner product between vectors. It is worth mentioning that compared with converting the input image into a grayscale image, our color gradient tensor preserves much more color information, thus can avoid missing lines at the places where the luminance of the background is close to the line drawings.

Thirdly, we apply a Gaussian smoothing on this tensor field to reduce the affects of image noises and the line roughness, and thus improve the smoothness of target tangential field.

$$T_S = T \otimes G(\sigma). \tag{3}$$

Here $\otimes$ represents a 2D convolution operator and $G(\sigma)$ denotes 2D Gauss function with standard deviation of σ, where $\sigma = 1$ in this paper. As shown in Figure 1, Sobel filter is very sensitive to luminance changes of line drawings, even subtle ones. This tends to reduce the smoothness of tensor field in the center of line drawings, thus introduce distraction for line tracking (b). With a larger standard deviation σ, the smoothness of the tensor field can be improved, but the accuracy in the center of line drawings is affected due to the over-smoothing. The major cause is that those aforementioned operators are designed to detect edges where color is rapidly changing, rather than luminance valleys in the center of line drawings.

Instead, our tensor field smoothing method transfers tensors on edges to luminance valleys with lower gradient magnitude, thus obtains a smooth tensor field in all line drawing regions (d).

Finally, a classic eigenvector analysis on tensor matrix gives the maximum/minimum eigenvalue $\lambda_{\max}$ ($\lambda_{\min}$). We take the normalized eigenvector corresponding to the maximum eigenvalue as the gradient direction g for each pixel, and the normalized eigenvector corresponding to the minimum eigenvalue as the tangential direction t. It is important to emphasize here that these two eigenvectors are in the gradient and tangential directions.

3.2 Line tracking with correction mechanisms

Given a tangential direction $t(x)$ for each pixel x with a starting point x_0, we can get a trajectory by a discrete line integral convolution:

$$x_{k+1} = x_k + s \cdot t(x_k). \tag{4}$$

Here, $s := \text{sign}\langle t(x_{k-1}),\ t(x_k)\rangle$, $\langle\cdot,\cdot\rangle$ is inner product between two vectors. Sign s coincides with the previous convolution direction, which can guarantee tracking always moving forward, thus avoids inverse tracking. This is also the reason why tangential vectors are drawn as line segments rather than arrows, as shown in Figure 1.

The convolution formula (4) may produce large errors, which mainly come from two parts. One is sampling errors: tangential direction only defines on integer pixels (as shown in Figure 1), while the coordinates of x_k may be non-integer. To improve the accuracy, we employ bilinear interpolation to calculate tangential directions for any non-integer position, denoted by $t'(x_k)$. The other one is accumulating errors: every tracking step takes previous steps as a starting point, whose errors may be enlarged after many times of convolution, as shown in Figure 2(a). In this paper, we employ a Runge-Kutta method instead of the classic Euler integral:

$$x_{k+1} = x_k + s \cdot t'\left(x_k + \frac{t'(x_k)}{2}\right). \tag{5}$$

Although the improvement on integral methods reduces the accumulating errors, it can not guarantee the tracking trajectory always being the center of line drawings, as shown in Figure 2(b). Considering this, we propose a correction mechanism for line tracking, that is based on a quadratic surface fitting of line's cross section to pull the diverged path to the center of line drawings. We first build a local Frenet frame for the current tracked position using the gradient $\bar{x}$ and tangent $\bar{y}$; and then sample in grid $(2n+1) \times (sm+1)$ luminance under this frame (we use $b = 6, m = 4$ in this paper); thirdly, a 2D quadratic function $f(\bar{x}, \bar{y}) = a_0\bar{x}^2 + 2a_1\bar{x}\bar{y} + a_2\bar{y}^2 + a_2\bar{x} + a_4\bar{y} + a_5$ is fitted using luminance values on these samples; finally, we project x to the central axis of $f(\bar{x}, 0)$: $(-a_3/2a_0, 0)$ to correct the line tracking. It is worth emphasizing that we track both the center and contours on both sides of line drawings. The aforementioned tracking method can be adapted for contour tracking easily. The only adjustment is that we sample gradient magnitude computed by Sobel filter, instead of luminance. The gradient magnitude reaches maximum on contours and approaches minimum in the center of line drawings, thus can naturally correct line tracking under our pipeline.

Figure 2 (c) and (d) illustrates the comparison between our methods with existing midpoint-based methods. Midpoint-based methods first get contour points on both sides of line drawings by thresholding, and then take the midpoint of these two points as a tracking point. Midpoint-based produce a jagging issue. It can be solved by a smoothing postprocessing, but will introduce a shrinking problem. Our correction mechanism, based on quadratic surface fitting, takes into consideration both current tracking position and local neighborhood line information, thus produces precise and smooth correcting results.

4 Junction analysis

A junction is a point where lines intersect. The topological structure around a junction is the central position of the junction, the number and orientations of junction branches, as well as the smoothness

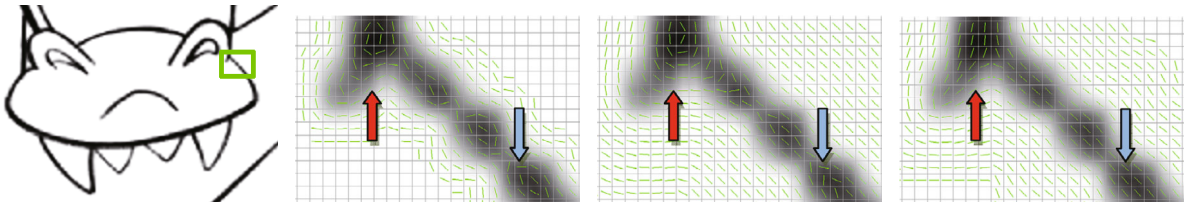

Figure 1 (a) Input image; (b) Sobel filter; (c) first-order Gaussian derivative; (d) our method. Vector field computed using Sobel filter has a poor smoothness (b); the one computed using first-order Gaussian derivative (with a standard variance 1) is smoother, but over-smoothes at junctions (c); our tensor-based method overcomes these shortcomings (d). Note that we use green bar to illustrate the normalized vector of each pixel.

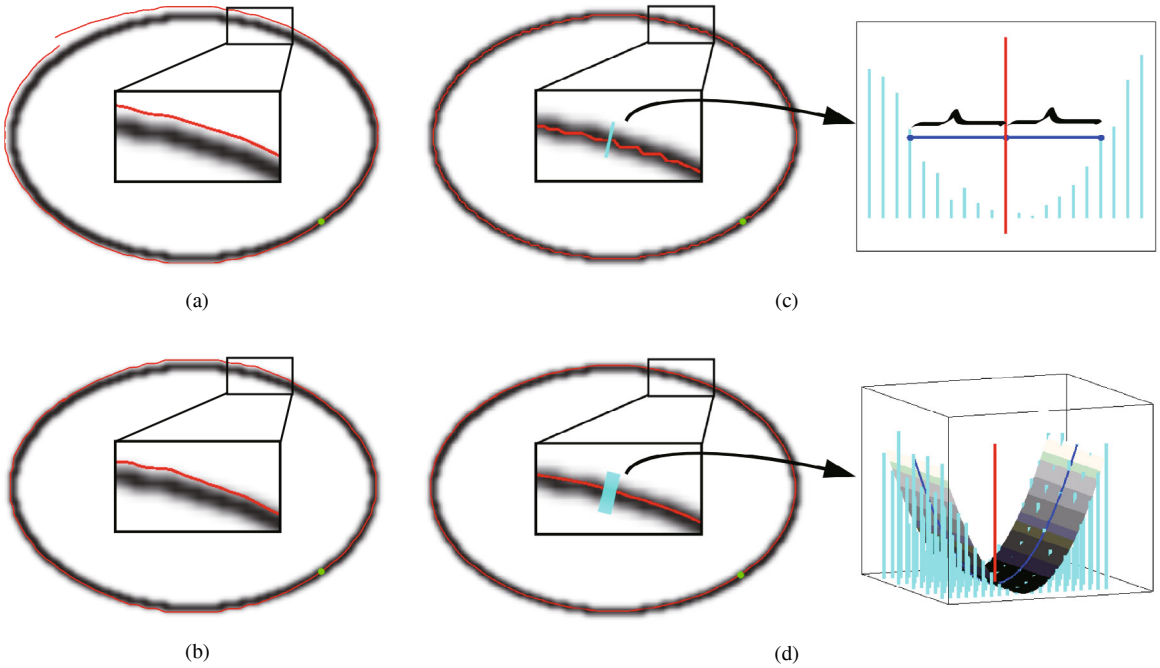

Figure 2 (a) Euler method; (b) Runge-Kutta method; (c) central point correction; (d) our fitting-based correction. To illustrate our improvement on line tracking accuracy, we take an elliptic without junctions as an input image. The tracking starts from the green point and ends until both sides meet. In (c) and (d), heights of the blue lines indicate luminance in local sampling neighborhood, red lines in the middle represents the correcting tracking position (under Frenet frame).

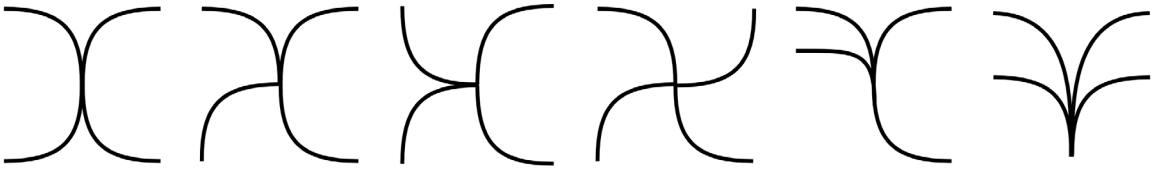

Figure 3 Examples for junctions with four branches. Different topological connection (i.e. smoothness) brings a big challenge for the vectorization task around junctions.

between among adjacent branches (i.e. whether they are connected without a sharp turn). Since a junction with two branches can be easily detected by a corner detection technique, we focus on junctions with three or more branches in this paper. For example, a junction with three branches mainly includes Y-shape, γ-shape and T-shape; while a junction with four branches includes even more topological structures, as shown in Figure 3.

Traditional line tracking methods usually enter a junction region from one of its branches, and leave this region without knowing the total branch number connected to this junction. This phenomenon leads

to the loss of lines and misinterpretation of junctions as endpoints, which directly reduces the precision of vectorization results. Besides, since the vector field that heavily affects line tracking has directional singularities in junction region, the selected leaving branch is indeterminate. Under this circumstance, an inevitable minor error from the line tracking may produce a tremendous difference in the vectorization result. This vastly increases the sensitivity of these methods, as shown in Figure 4(a). Different from central line tracking, our contour line tracking is able to provide a stable tracking even in junction region due to the fact that contours never go close to the center of directional singularities, as shown in Figure 4(b).

Based on aforementioned observations, we introduce an improved junction analysis in this paper. We first trace both midpoints and contour points simultaneously, and detect a junction encounter efficiently according to contour line tracking; secondly, based on luminance polar mapping, we compute the accurate center position of this junction and its branch numbers; and finally, we determine the smoothness between adjacent branches based on the sharpness of tracked contours. Consequently, a vectorization result with a consistent topological structure is obtained.

4.1 Judgement on junction point

According to the line tracking algorithm presented in Section 3, given a tangential vector field, central line path $\{x_k^C\}$ and contour paths on both sides $\{x_k^L\}$ and $\{x_k^R\}$ could be tracked from a starting point. The divergence of tracking directions on left and right contours indicates that a junction point is approaching:

$$\theta = \arccos \frac{\langle x_k^L - x_{k-1}^L,\ x_k^R - x_{k-1}^R \rangle}{\|x_k^L - x_{k-1}^L\|\ \|x_k^R - x_{k-1}^R\|}. \tag{6}$$

Here, $\langle \cdot, \cdot \rangle$ is an inner product of vectors. When $\theta > 30^\circ$ or $\|x_k^L - x_k^R\|$ is suddenly 4 pixels bigger than the line width, we could conclude the encounter of a junction, as shown in Figure 5(a).

It is worth mentioning that, since three line tracking processes perform separately and move one pixel in every tracking step, their respective trajectories may lose synchronization in some cases. Take an arc as an example, line tracking on outer side contour of the arc is moving slower than the inner side of the arc. To solve this issue, we project the midpoint, left and right contour points to a straight line perpendicular to the tracking path.

4.2 Branch numbers and orientations

When a junction point is encountered, its exact position and number of line branches should be calculated before further line tracking. In this paper, we propose a luminance polar mapping to estimate both the number and orientation simultaneously. Firstly, we keep tracing the central path $3L$ pixels's length without using the correction mechanism (here we take $L = 6$), and sample even-spaced L points on this path as potential junction centers $\{C_k\}$, as in Figure 5(b). To find the most reasonable one among those potential junction centers, we take c_k as center and r as radius to draw a circle as potential junction region. In this region, we map the orthogonal coordinate to polar coordinate, and compute luminance averages under this polar coordinate $[0, 2\pi] \times [0, 255]$ to acquire a luminance curve, as the amaranth curve shown in Figure 5(c).

We compute a luminance polar curve for each potential junction center, and smoothen this curve to remove noises, thereby calculate all local minimum values of this curve. We denote the average of these minimum values for each potential junction center as $\{\bar{h}_k\}$. In fact, each local minimum indicates one branch direction whose luminance average is very small in the polar coordinate. Therefore, the number of local minimum values is equal to the number of branches for the corresponding potential junction center. To select the most reasonable junction center, we first select potential centers with maximal branch numbers, which helps to reject centers far away from the real center. Subsequently, we take the one with a smallest $\bar{h}_k$ as the final junction center among potential junction ones, as shown in Figure 5(d).

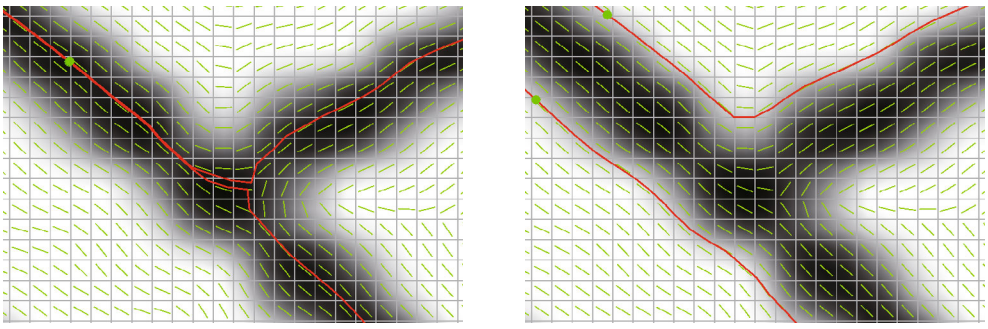

Figure 4 Comparison the stability between midpoint tracking and contour tracking. (a) Starting from two midpoints whose distance is only 0.1 pixel separately, two tracked lines diverge each other around junctions, it shows central line tracking is very sensitive; while contour tracking is rather stable and accurate as in (b).

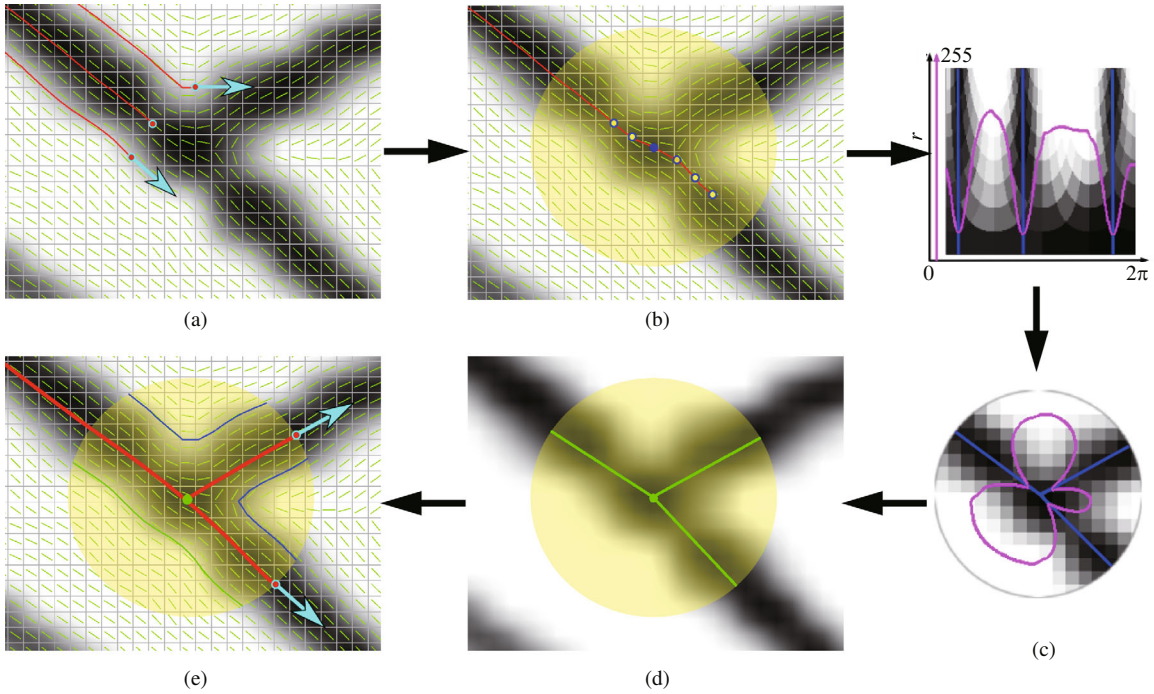

Figure 5 Pipeline of junction analysis: a junction is detected by directional difference through contour tracking on both sides (a); then we take even-spaced samples on the central tracking path to get potential junction centers (b); a luminance polar mapping technique (c) is employed to analyze both branch number and their orientations simultaneously (d); finally, topological structure is determined by the smoothness of contour paths (e).

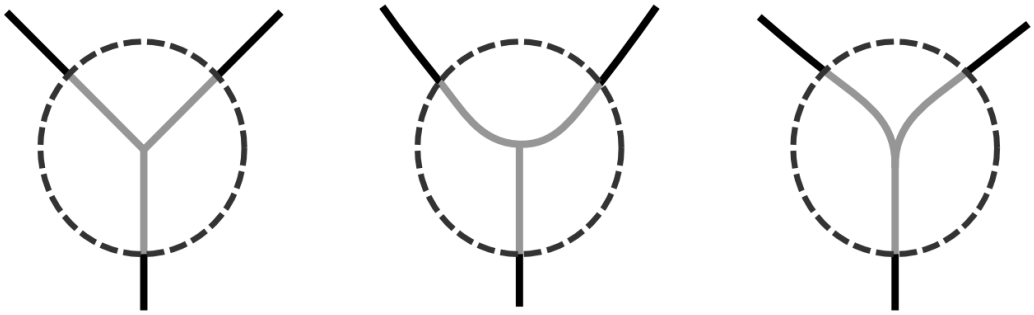

Figure 6 Three line drawings share the same junction center, branch number and orientations (outside the gray dash circles), but differ in the smoothness between branches.

4.3 Smoothness between adjacent branches

The smoothness between adjacent branches defines whether the transition between them is smooth or not. It is one of the most important topological structures in line drawings, and not included in what we have represented in the last subsection (i.e. junction center and branch number), as shown in Figure 6.

For arbitrary adjacent two branches, we parameterize them as $x_1(t_1)$ and $x_2(t_2)$. In this paper, we use first-order geometrical continuity (G^1 continuity) to describe the smoothness between adjacent branches, which means these two branches share the same normalized tangential vector at the joint point:

$$\lim_{t_1\to 1}\frac{x_1'(t_1)}{|x_1'(t_1)|}=\lim_{t_2\to 0}\frac{x_2'(t_2)}{|x_2'(t_2)|}. \tag{7}$$

Unfortunately, it is always difficult to compute the tangential vector of line branches at junction region from the original image. According to mathematical definition 7, it is defined as the tangential vector as the branch approaches the junction center, i.e. $x_1'(t_1)$ when $t_1 \to 1$ and $x_2'(t_2)$ when $t_2 \to 0$. However, the problem is, due to the influence of directional singularity and image noise near junctions; the closer it is to junction center, less accurate the midpoint tracking path is, and less stable the tracking direction will be, as in shown in Figure 4(a).

Contrary to central line tracking, contour tracking shows quite high stability in junction area, as shown in Figure 4(b). In this paper, we propose a branch smoothness computation method that is based on contour trajectories. Firstly, through contour tracking, we obtain the contour trajectory between any adjacent two branches $x_1(t)$ and $x_2(t)$ (already computed when detecting the encounter of junction regions); and then we take L contour points $\{x_k^{12}\}$ $(0 \leqslant k \leqslant L)$ which are the closest to the junction center. The local bending angle is computed using the following formula:

$$\beta_k^{12}=\arccos\frac{\langle x_{k-a}^{12}-x_k^{12},\ x_{k+a}^{12}-x_k^{12}\rangle}{\|x_{k-a}^{12}-x_k^{12}\|\ \|x_{k+a}^{12}-x_k^{12}\|}. \tag{8}$$

Here a represents the interval between three sampled points for the computation of sharpness β_k. The smaller the interval a, the more precise the value of β_k, while it is more sensitive to the noise of trajectory. In contrast, the larger a, the more noise-resistent β_k, but its precision decreases. Thus, we set $a = 4$ in this paper. The bending angle of the whole contour trajectory is computed as $\beta^{12} = \min_{a\leqslant k\leqslant N-a}\beta_k^{12}$. If this angle is larger than 150°, two branches corresponding to this contour trajectory are considered to be G^1 continuous. If a branch is not G^1 continuous with any of its adjacent branches, we employ the spline curvature given in [18] to calculate its G^1 continuity with nonadjacent branches.

For two line branches that satisfy G^1 continuity, we use third-order Hermit spline to connect them in the junction region. After connecting all the branches that are G^1 continuous, some branches may be still disconnected with any other branches. For this sake, we first use second-order spline function to interpolate this branch and the corresponding junction center, and then prolong/truncate this spline to its first encounter of other branches (including their connecting splines), as shown in Figure 7.

5 Comparisons and discussions

5.1 Comparisons

Considering the directional singularity of central line skeleton in junction region, Noris et al. [18] completely ignored the line information in this region. The authors connect junction branches outside of the junction regions using a spline function, and use its curvature to judge whether they are G^1 continuous or not. Although this method retains the smoothness between adjacent branches, it tends to get a vectorization result inconsistent with the topological structure of the original line drawing image. On the contrary, we make full use of the stable contour tracking in the junction region, which provides us with accurate hints to determine the correct G^1 continuity between branches.

Figure 8 shows results from Wintopo software, the method introduced by Noris et al. [18] and the ground truth provided by artists. WinTopo retains the overall appearance of line drawings, but each

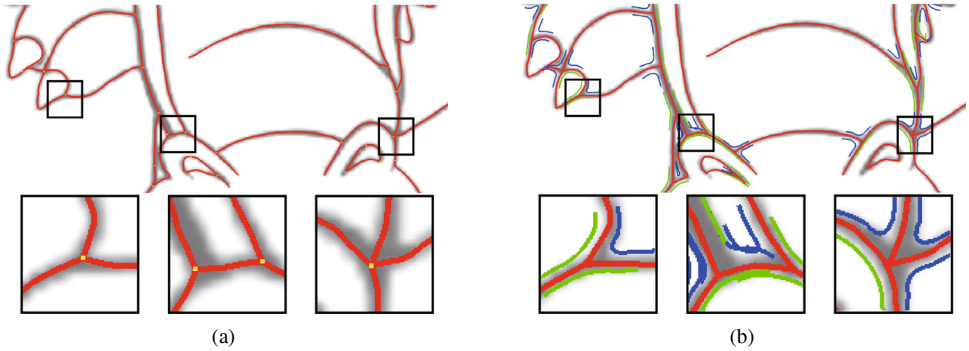

Figure 7 Intermediate results for junction analysis. (a) Junction centers and branch numbers; (b) smoothness between adjacent branches. Our method reconstructs lines in junction regions according to the smoothness of contours (b), it improves the one only using branch numbers (a). Note that green curves are G^1 continuous, and blue curves are not G^1 continuous.

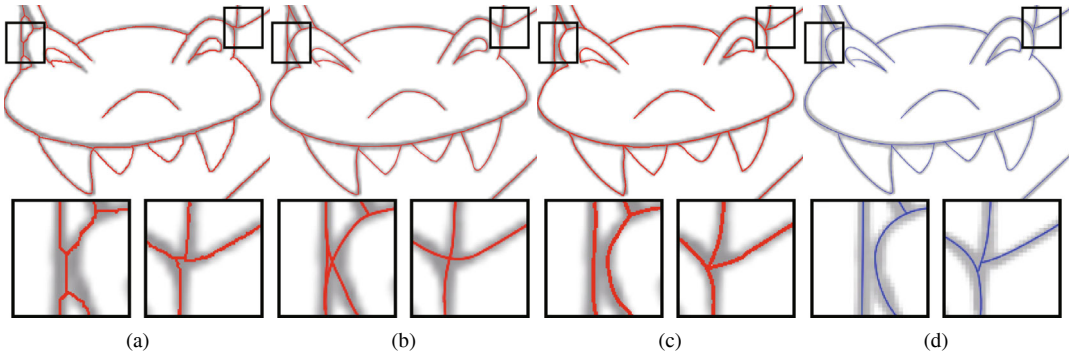

Figure 8 Comparison of results from (a) WinTopo software, (b) Noris et al. [18], (c) our method and (d) ground truth, hand-drawn by artists. It shows our method performs better to preserve topological structures of the original line drawing image.

junction could only be attached to two branches, and lines tend to be twisted. The vectorization results generated by Noris et al. have smoother lines, but lose the topological structure around junctions on both top left and top right corner. Our method produces smooth vectorization results preserving the original topological structures. This figure also gives the manual vectorized lines from artists, which can provide a visual assessment of different vectorization methods.

Figure 9 exhibits a further comparison among various vectorization methods on line drawing images. Remarkably, before using the smoothness between branches, our method demonstrates a better performance in keeping sharp features (pointed with green arrows). In comparison, by using the smoothness, the method introduced by Noris et al. [18] over-smoothed some junction points, enlarging the topological structure difference between their results and the original image (pointed with red arrows). That is due to the fact that Noris et al. have totally ignored the line information inside junction regions.

Figure 10 shows our results for rough line drawing images. Common vectorization softwares and the approach like [18] are incompatible to such input images. Although existing methods like the one present by Bartolo et al. [33] could vectorize rough line drawing images, but these methods have many limitations like losing junction points and important details, as shown in Figure 10(b). Figure 11 shows a few more results produced by our method.

5.2 Discussions and limitations

Our method inherits the high efficiency of classic line tracking methods, it only takes 0.2 to 3 s to vectorize a line drawing image, depending on the image size and the amount of lines, which is much faster

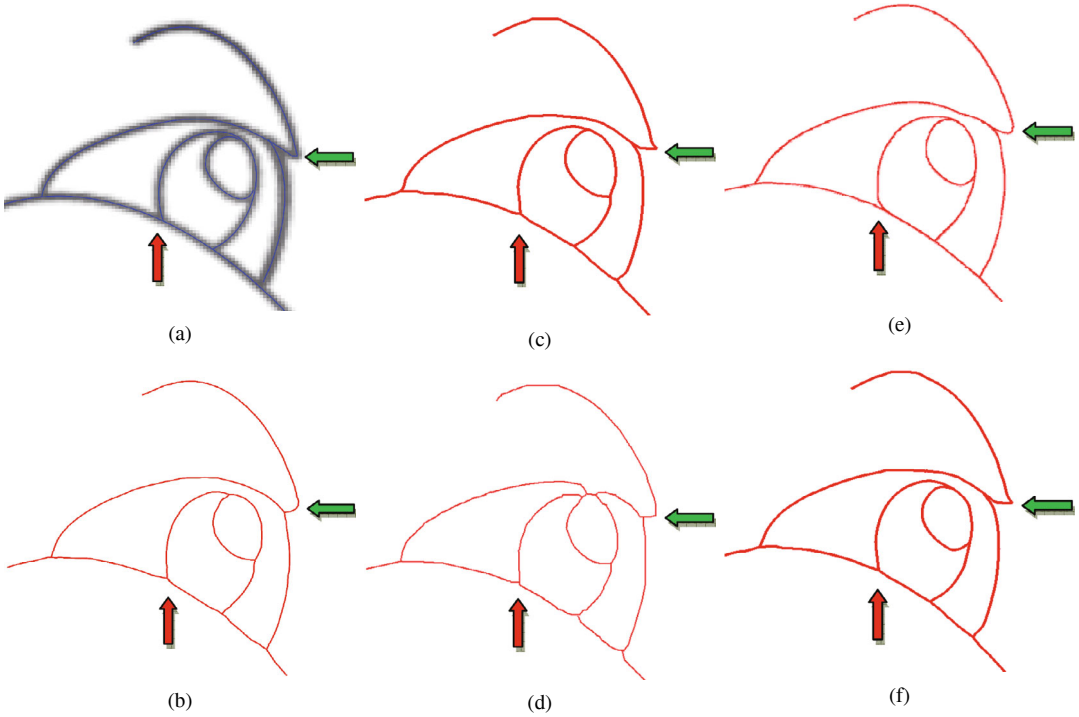

Figure 9 (a) Artist interaction and input image; (b) results with Wintopo software; (c) Ref. [18] without smoothness; (d) Ref. [18] with smoothness; (e) our method without smoothness; (f) our method with smoothness. Comparable results based on different methods. The method introduced by Noris et al. [18] destroys the junction structure after vectorization due to an incorrect smoothness determination, see red arrows in (c) and (d). While our method preserves both topological structures and sharp features, see green arrows in (e) and (f).

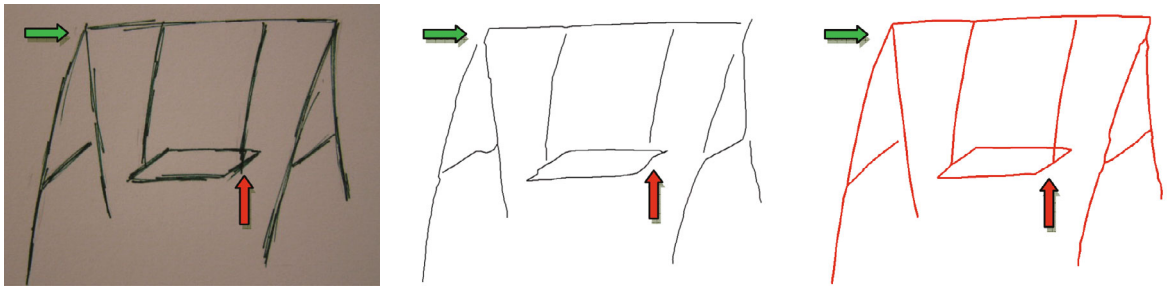

Figure 10 Comparison of (a) rough line drawing with (b) method by Bartolo et al. [33] and (c) our proposed method. Although the method introduced by Bartolo et al. [33] preserves the overall appearance of the input line drawing, it lacks details and is incapable of reconstructing topological structures at junctions. Note that Wintopo software and Noris et al.'s method [18] are unfit for those rough line drawings as well. Meanwhile, our method works for rough line drawings well, with an adjusted parameter of $\sigma = 3.0$ in formula (3), with a bigger sampling range for quadratic surface fitting.

than that proposed by Noris et al. [18]. It is noteworthy that the tensor-based vector field computation (Subsection 3.1), quadratic surface fitting (Subsection 3.2) and luminance polar mapping (Subsection 4.3) do not rely on line tracking process and can be computed in parallel. We thus make use of GPU to accelerate, pre-compute these three steps before line tracking, and store their values in texture buffers to support further access. The parallel computation on GPU only takes 0.07–0.5 s. This data is recorded from a workstation with NVIDIA GeForce 8800GT Intel® Core™2 DuoE6550 @2.33 GHz.

Our method still has two major limitations. Although our method is completely automatic, several important parameters have to be adjusted to avoid unexpected results, such as the standard variance σ in formula 3, the angle threshold to determinate branch smoothness and sampling range for quadratic surface fitting.

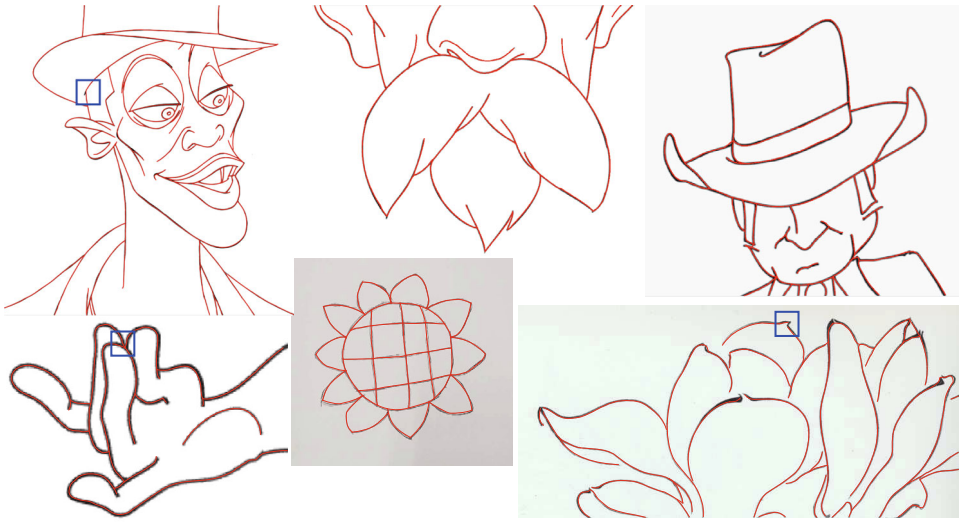

Figure 11 More vectorization results.

(a) (b) (c)

Figure 12 Three cases (blue rectangles in Figure 11) where user interactions are needed. (a) Merging interaction; (b) adding interaction; (c) splitting interaction.

The other limitation of our method arises in the case when multiple junctions are too close to each other. For instance, because of two very close junctions, our method fails to get the accurate position around the green arrow in Figure 9 and does not succeed to separate three fingers in Figure 11. Therefore, we have developed an interactive tool, for junction moving, splitting and merging, which allows users to manually modify and automatically obtain vectorized results. All results shown in Figures 1–10 are automatically generated without user interaction, while some manual interactions are required to vectorize complex line drawings in Figure 11, as illustrated in Figure 12.

6 Conclusion and future work

In this paper, we present a vectorization approach of line drawing images based on junction analysis. It first introduces a correction mechanism for line tracking based on quadratic surface fitting, which allows us to reduce accumulating errors in the process. Using this improved line tracking, we propose a junction analysis combining central line tracking and contour tracking. It computes the position of junction centers and the branch number simultaneously with a luminance polar mapping, to obtain smooth and structure-preserving vectorization. A plenty of experiments and comparisons demonstrate the validity and efficiency of our approach.

Since our method has solved the junction topological structure problem that is essential for the vectorization of line drawing images, our future work will focus on extending it to the vectorization of common images and line drawing animations, besides overcoming the limitations of our core algorithms. Preliminary experiments indicate that our method enhances the vectorization quality of common images, but needs better color processing. In addition, temporal coherence between video frames should be paid more attention for the vectorization of line drawing animations.

Acknowledgements

This research was supported by National Natural Science Foundation of China (Grant Nos. 61303138, 61272309) and Open Project of the State Key Lab of CAD&CG, Zhejiang University (Grant No. A1428).

References

1 Chen T, Cheng M M, Tan P, et al. Sketch2Photo: Internet image montage. ACM Trans Graph, 2009, 28: 1–10
2 Xu K, Chen K, Fu H B, et al. Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. ACM Trans Graph, 2013, 32: 1–12
3 Lai Y K, Hu S M, Martin R R, et al. Automatic and topology-preserving gradient mesh generation for image vectorization. ACM Trans Graph, 2009, 28: 1–8
4 Lee B S, Kazi R H, Smith G. SketchStory: telling more engaging stories with data through freeform sketching. IEEE Trans Vis Comput Graph, 2013, 19: 2416–2425
5 Yoo I, Vank J, Nizovtseva. Sketching human character animations by composing sequences from large motion database. Vis Comput, 2014, 30: 213–227
6 Fu H B, Zhou S Z, Liu L G, et al. Animated construction of line drawings. ACM Trans Graph, 2011, 30: 1–10
7 Zhang T, Suen C Y. A fast parallel algorithm for thinning digital patterns. Commun ACM, 1984, 27: 236–239
8 Hilaire X, Tombre K. Improving the accuracy of skeleton-based vectorization. Graphics recognition algorithms and applications. Berlin/Heidelberg: Springer, 2002. 273–288
9 Sezgin T M, Davis R. Handling overtraced strokes in hand-drawn sketches. In: Proceedings of Making Pen-Based Interaction Intelligent and Natural. New York: AAAI Fall Symposium, 2004. 1–4
10 Hilaire X, Tombre K. Robust and accurate vectorization of line drawings. IEEE Trans Patt Anal Mach Intell, 2006, 28: 890–904
11 Bonnici A, Camilleri K P. Scribble vectorization using concentric sampling circles. In: Proceedings of International Conference on Advanced Engineering Computing and Applications in Sciences. Los Alamitos: IEEE Computer Society, 2009. 89–94
12 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. New York: ACM, 2013. 69–77
13 Zou J J, Yan H. Line image vectorization based on shape partitioning and merging. In: Proceedings of International Conference on Pattern Recognit, Barcelona, 2000. 994–997
14 Zou J J, Yan H. Cartoon image vectorization based on shape subdivision. In: Proceedings of Computer Graphics International. Los Alamitos: IEEE Computer Society, 2001. 225–231
15 Wenyin L, Dori D. Sparse pixel tracking: a fast vectorization algorithm applied to engineering drawings. In: Proceedings of International Conference on Pattern Recognit, Vienna, 1996. 808–812
16 Dori D, Liu W. Sparse pixel vectorization: an algorithm and its performance evaluation. IEEE Trans Patt Anal Mach Intell, 1999, 21: 202–215
17 Whited B, Rossignac J, Slabaugh G, et al. Pearling: stroke segmentation with crusted pearl strings. IEEE Trans Patt Anal Mach Intell, 2009, 19: 277–283
18 Noris G, Hornung A, Sumner R W, et al. Topology-driven vectorization of clean line drawings. ACM Trans Graph, 2013, 32: 1–11
19 Sun J, Liang L, Wen F, et al. Image vectorization using optimized gradient meshes. ACM Trans Graph, 2007, 26: 1–11
20 Xia T, Liao B, Yu Y. Patch-based image vectorization with automatic curvilinear feature alignment. ACM Trans Graph, 2009, 28: 1–10
21 Chiang J Y, Tue S, Leu Y. A new algorithm for line image vectorization. Patt Recog, 1998, 31: 1541–1549
22 Chang H H, Yan H. Vectorization of hand-drawn image using piecewise cubic Bézier curves fitting. Patt Recog, 1998, 31: 1747–1755
23 Liu K, Huang Y S, Suen C Y. Identification of fork points on the skeletons of handwritten Chinese characters. IEEE Trans Patt Anal Mach Intell, 1999, 21: 1095–1100
24 Rajan P, Hammond T. From paper to machine: extracting strokes from images for use in sketch recognition. In: Proceedings of the 5th Eurographics conference on Sketch-Based Interfaces and Modeling, Annecy, 2008. 41–48
25 Yang Y, Zhu C, Sun Q. Vectorization of linear features on color scanning topographic maps (in Chinese). J Comput Aid Des Comput Graph, 2009, 21: 533–541
26 Song J, Su F, Cai S. The segmentation of text from line based on the line recognition in scanning engineering drawings (in Chinese). J Nanjing Univ, 2001, 37: 535–541
27 Wong J, Guo C. An improved image template thinning algorithm (in Chinese). J Image Graph, 2004, 9: 297–301
28 Liu R, Li Y. Study of auto-vectorization based on scan-thinning algorithm (in Chinese). Acta Geodaetica et Cartographica Sinica, 2012, 41: 309–314

29 Lu Z, Zhang Q. Line contour tracking in engineering drawing vectorization (in Chinese). J Image Graph, 1997, 2: 878–882
30 Bartolo A, Camilleri K P, Fabri S G, et al. Scribbles to vectors: preparation of scribble drawings for CAD interpretation. In: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling. New York: ACM, 2007. 123–130
31 Nieuwenhuizen P R, Kiewiet O, Bronsvoort W F. An integrated line tracking and vectorization algorithm. Comput Graph Forum, 1994, 13: 349–359
32 Kyprianidis J E, Kang H. Image and video abstraction by coherence-enhancing filtering. Comput Graph Forum, 2011, 30: 593–602
33 Bartolo A, Camilleri K P, Fabri S G, et al. Line tracking algorithm for scribbled drawings. In: Proceedings of International Symposium on Communications, Control and Signal Processing, St Julians, 2008. 554–559
34 Wang S D, Ma Z Y, Liu X H, et al. Coherence-enhancing line drawing for color images. Sci China Inf Sci, 2013, 56: 110903
35 Pham T A, Delalandre M, Barrat S, et al. Accurate junction detection and characterization in line-drawing images. Patt Recog, 2014, 47: 282–295