DOI: 10.1111/cgf.13818
Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

Deep Line Drawing Vectorization via Line Subdivision and Topology Reconstruction

Yi Guo[1,2], Zhuming Zhang[1,2], Chu Han[1,2], Wenbo Hu[1,2], Chengze Li[1,2] and Tien-Tsin Wong[†1,2]

[1]Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[2]The Chinese University of Hong Kong, Hong Kong SAR, China

Abstract

Vectorizing line drawing is necessary for the digital workflows of 2D animation and engineering design. But it is challenging due to the ambiguity of topology, especially at junctions. Existing vectorization methods either suffer from low accuracy or cannot deal with high-resolution images. To deal with a variety of challenging containing different kinds of complex junctions, we propose a two-phase line drawing vectorization method that analyzes the global and local topology. In the first phase, we subdivide the lines into partial curves, and in the second phase, we reconstruct the topology at junctions. With the overall topology estimated in the two phases, we can trace and vectorize the curves. To qualitatively and quantitatively evaluate our method and compare it with the existing methods, we conduct extensive experiments on not only existing datasets but also our newly synthesized dataset which contains different types of complex and ambiguous junctions. Experimental statistics show that our method greatly outperforms existing methods in terms of computational speed and achieves visually better topology reconstruction accuracy.

CCS Concepts

• ***Computing methodologies*** → ***Neural networks; Image manipulation;***

1. Introduction

Vector images are a compact format of digital graphics, which use mathematical primitives to represent lines and curves. As vector graphics are rendered in analytical procedures, they can be easily rasterized in an optimal way with respect to the specific resolution. Also, manipulating vector graphics is flexible and straightforward, as one can modify the parameters and control points pragmatically, or with user interfaces. Comparing to raster image editing, vectors can be modified arbitrarily without losing image quality. As a result, vector graphics have been widely used in many professional contents creation workflows, such as Computer-aided design, UI design, 2D animations, etc.

Despite the benefit of digital editing with vector graphics, many art forms cannot be directly created in vector representation, such as visual arts. Because they require free-form drawing to make the contents more attractive and vivid. What's more, drawing with parametric curves is not intuitive for humans. Therefore, many artists and architects still prefer paper and pencils to make the first version of their work and afterward digitize them into vector graphics. It makes vectorization process to be the most critical step of the digital content creation. And the quality of vectorization determines the faithfulness to the original and the difficulty in further editing.

So it is fundamental to design a vectorization algorithm to convert raster line drawings into vectors accurately. However, the task is non-trivial because of the difficulties in analyzing the topology of raster line drawings due to the ambiguity of line drawings, especially at complex junctions. Figure 1 illustrates two representative ambiguous cases. At junctions, where multiple curves intersect, the connectivity is difficult to predict. It is also challenging to classify the pixels of closing lines.

To handle these cases, Simo-Serra et al. [SSII18] attempted to edit raster line drawings directly, but they still cannot generate vector outputs for finer editing. Traditional vectorization methods estimate the topology of raster images and afterward fit parametric curves by minimizing the reconstruction error. However, most of them fail to decouple complex junctions and line ambiguity. Some recent work like Bessmeltsev et al. [BS19] analyze pixel orientations so that they can detect and process junctions, but they can only distinguish T- and X-junctions. [NHS*13] and [KWÖG18] try to reconstruct the topology by an exhaustive search of all possible connectivities, but obviously, the computational complexity greatly limits their practical uses.

Different from the existing methods, we do not assume any spe-

† Corresponding author: Tien-Tsin Wong (ttwong@cse.cuhk.edu.hk).

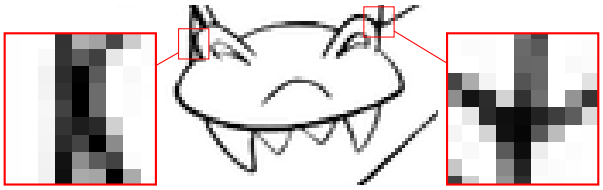


Figure 1: *Two representative types of ambiguity. In the left case, two curves are very close to each other. In the right case, the connectivity is ambiguous.*

cific types of junctions. Instead, we use neural networks to automatically infer the junction connectivity with the highest likelihood. To achieve the goal, we propose to analyze topology from global to local scale, and our method consists of two phases: line subdivision and topology reconstruction. In the line subdivision phase, we oversegment the lines into partial curves via centerline extraction and junction detection. The connectivity at junction locations is not considered during this phase. For this purpose, we use a data-driven method to extract the centerlines with a convolutional neural network (CNN) inspired by [SSII18]. We use a multi-task CNN to simultaneously conduct centerline extraction and junction detection as they are closely related. With the centerlines and junction points, we can simply achieve the line subdivision to break the whole drawing into partial curves.

Afterwards, in the topology reconstruction phase, we estimate the connectivity of partial curves at each junction. To effectively deal with different types of junctions, we tailor another CNN and train the network with a specially designed dataset containing synthesized line drawings with various types of complex and ambiguous junctions. The reconstructed curves will be output from the network model and vectorization is straightforward by using the overall topology and tracing the connected curves.

To qualitatively and quantitatively evaluate our method, we conduct extensive experiments on existing and our newly synthesized datasets. Statistics show that our method greatly outperforms existing methods in terms of computational speed and achieves visually better topology reconstruction accuracy. Our contributions are summarized as follows:

- We propose a two-phase method to vectorize raster line drawing via line subdivision and topology reconstruction.
- Our vectorization method is data-driven by utilizing deep convolutional neural networks. The method is robust enough to process line drawings in various styles or with complex shapes. The input size can be set arbitrarily.
- Our vectorization method is more efficient and achieves visually better topology reconstruction accuracy comparing to existing methods.

2. Related Works

Existing vectorization methods can be roughly classified into two categories: region-based methods and stroke-based methods. Region-based methods vectorize shaded images (e.g. photographs and cartoon images) into various primitives such as linear color gradients [LL06] and PDEs [OBB*13], while stroke-based methods convert line information into parametric representation. Our proposed method belongs to the latter category.

Early stroke-based methods have strong assumptions on the line shape [JV97, HT06, CY98]. In particular, Janssen et al. [JV97] only vectorizes straight line segments [JV97]. Hilaire and Tombre [HT06] use circles and straight lines to represent vectorization results. However, these methods cannot be applied to hand-drawn line drawings whose shapes cannot be simply assumed. To vectorize common hand-drawn line drawings, Chang et al. [CY98] conduct piecewise cubic Bézier curves fitting on the lines. The recent work by Donati et al. [DCP17] proposes an automatic system to vectorize fashion hand-drawn sketches using Pearson's correlation coefficient with multiple Gaussian kernels. But neither Chang et al. [CY98] nor Donati et al. [DCP17] analyze the topology of curves, especially at junctions. So their results are inaccurate when there exists complex topology.

Some recent works analyze the topology of junction areas [NHS*13, FLB16, BS19, KWÖG18]. In particular, Noris et al. [NHS*13] propose a "reverse drawing" procedure to reconstruct all possible drawing states for a junction. The most likely stroke configuration is selected to represent the topology. However, their method is not data-driven and fails in many complex cases. Favreau et al. [FLB16] utilize global information to simplify and vectorize the sketch drawing that contains numerous overdrawn strokes. Unfortunately, their method often leads to oversimplified results, which is significantly deviating from the initially drawn contours. Most recently, Bessmeltsev et al. [BS19] track the orientation of curves with frame field and vectorize the curves by tracing frame fields. However, this method can only tackle T- and X-junctions but fails at more complex junctions. Another method proposed by Kim et al. [KWÖG18] uses a deep neural network to predict the possibility that two pixels occur on the same path, and then semantically segment the line drawings using graph cut. As their computational complexity is related to the number of line pixels, their method cannot deal with high-resolution images. All the above methods suffer from low accuracy or limited scalability. In sharp contrast, our proposed method can accurately analyze the topology of line drawing that contains complex junctions and is computational effective for arbitrary resolution.

Also, there exist some other works and commercial tools related to the conversion from raster images to vector graphics. In particular, commercial tools such as Adobe Live Trace, CorelDRAW, and Inkscape support to vectorize line drawings. Unfortunately, these commercial tools usually require tedious human adjustment to some parameters to achieve the best results. For example, CorelDRAW needs to control "detail" and "smoothness" and it is time-consuming to seek for the best combinations. Simo-Serra et al. [SSISI16] utilize deep neural networks to conduct sketch simplification of raster line drawings and afterward propose an interactive inking approach to convert a rough pencil sketch into a clean line drawing in [SSII18]. Chen et al. [CTYX17] and Ha et al. [HE17] use recurrent neural networks to generate sketch drawings based on human input. The domain-specific methods [WRS*09, CFL13, WSCY15] build up topology by extracting line-network for identifying roads in aerial images, blood vessels in medical images, or galaxy filament in astronomic images. Due

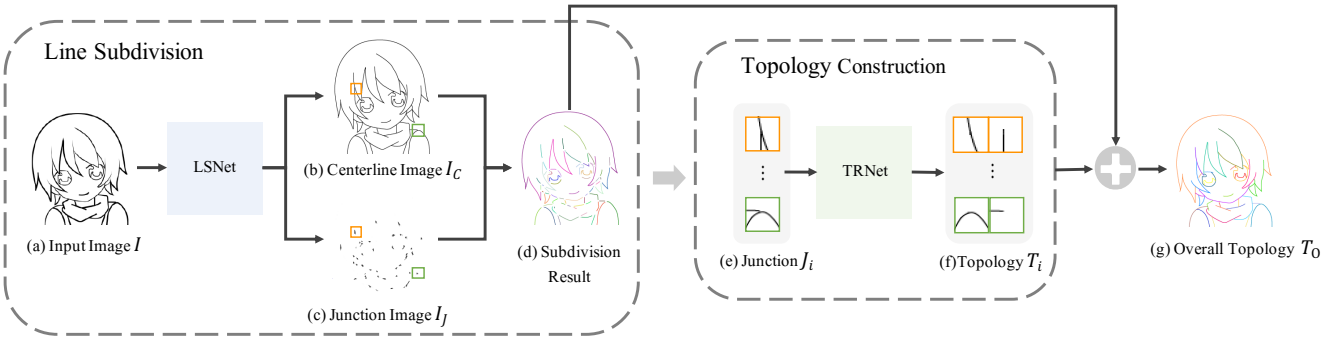


Figure 2: *System overview. Our proposed method analyzes and reconstructs the topology of lines with two phases. The first phase subdivides the lines into partial curves ignoring the junctions. The second phase reconstructs the topology at junctions by predicting the line connectivity.*

to their strong assumptions of the shape priors, their method cannot be applied to line drawings.

3. Approach

3.1. Overview

To better analyze the structure and topology of raster line drawings, we firstly extract the centerline images. Centerline images can compactly represent the essential structure of line drawing images. Given the centerline image and the line topology, accurate vectorization can be straightforwardly conducted. So as overviewed in Figure 2, we propose our line drawing vectorization method consisting of two phases: line subdivision and topology reconstruction. In the first phase, given an input line drawing image I (Figure 2(a)) with variable line width and arbitrary resolution, we generate the centerline image I_C (Figure 2(b)) and the junction image I_J (Figure 2(c)) providing junction candidates. Then, we subdivide the centerline image into partial curves by disjointing all connected curves at the location of junction candidates (Figure 2(d)). In the second phase, we eliminate the ambiguity at junctions by estimating the connectivity of the partial curves and reconstruct the topology T_i (Figure 2(f)) at each junction candidate J_i (Figure 2(e)). With T_i, we can separate distinct connected curves into independent layers. Finally, the overall topology T_O (Figure 2(g)) is summarized with I_C, I_J, and $\{T_i\}_N$. Then based on T_O, we trace and fit the curves with the cubic Bézier curve least-square fitting from [Kha07] to create the final vector output.

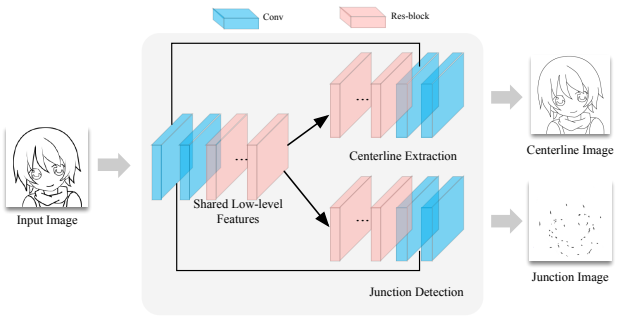


Figure 3: *Structure of Line Subdivision Network (LSNet) designed as a multi-task CNN. Given an input image, it simultaneously generates a centerline image and a junction image.*

3.2. Line Subdivision

In the first phase, we subdivide the lines based on the centerline image I_C (Figure 2(b)) and the junction image I_J (Figure 2(c)), both of which are generated with our Line Subdivision Network (LSNet). As shown in Figure 3, LSNet simultaneously conducts centerline extraction and junction detection in a multi-task manner, as these two sub-tasks are closely related to each other. LSNet is trained with a composite dataset, which will be discussed in detail in Section 3.4.

Centerline Extraction. As stated before, centerline extraction is essential to the estimation of line structure and topology. The traditional method [ZS84] utilizes morphological operators to extract the centerline or skeleton of binary lines, but the results suffer from a certain degree of distortion at junctions or around line ends. The recent work by Simo-Serra et al. [SSII18] uses a data-driven method to normalize all the strokes into constant-width and achieves higher accuracy. Motivated by Simo-Serra's method, given an input image I with variable-width strokes, we also generate constant-width centerline utilizing CNN in a data-driven manner.

Junction Detection and Line Subdivision. As centerline does not infer topology at junctions, junction detection is necessary for topology reconstruction in the next phase. Figure 4 shows positive and negative examples of junction candidates from a successful detection. After the detection, the junction candidates are stored into a junction image I_J which is a subset of the centerline image I_C. Once we acquire the I_C and I_J, we can conduct the line subdivision by simply removing all detected junction candidates from I_C and computing the connected components to acquire partial curves, as illustrated in (Figure 2(d)).

Network Structure. We propose to extract the centerline and junction candidates simultaneously from a single framework, due to the fact that junctions are always located at the intersection of centerlines. In other words, a higher probability of centerline intersection also indicates a higher potential of the existence of a junction. Here, we employ a multi-task deep neural network [Car97] to jointly achieve both tasks. As illustrated in Figure 3, our network starts with a shared path, which consists of two convolutional layers and eight residual blocks. The two task-specific branches are with the same structure, which is composed of eight residual blocks fol-

lowed by two convolutional layers. We use batch normalization and a Rectified Linear Unit (ReLU) activation function right after all the convolutional layers except for the output layer of each branch. To map our output to the range of [-1, 1], we use the *tanh* activation function for each output layer. The size of the convolutional kernels is 3×3 expect for the 1×1 kernel in the output layers. As the network is fully convolutional, the method is capable of handling input images with arbitrary resolutions.

Loss Function. the pair of the training data consists of an input image and two ground truth output images which are the centerline image and the junction image. Data preparation is detailed in Section 3.4. Finally, we train the LSNet in a supervised manner with the combined loss:

$$\mathcal{L}_S = \alpha||I_n - \hat{I}_n||_2^2 + (1-\alpha)||I_J - \hat{I}_J||_2^2, \quad (1)$$

where $||\cdot||_2$ is the L_2-norm. I_n and I_J are the ground truth centerline image and junction image. $\hat{I}_n$ and $\hat{I}_J$ are the two predicted images. α is the weighting factor balancing two tasks with a default value of 0.5.

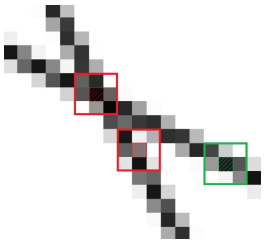


Figure 4: *A local patch of an centerline image. Two red boxes give examples of junction candidates (shadowed with red slash). The green box gives an example of pixels that are not candidates (shadowed with green slash).*

3.3. Topology Reconstruction

After subdividing the centerline images I_C into a set of partial curves (Figure 2(d)), we then analyze the topology at each junction to derive the connectivity of these partial curves. We crop a fixed-size 32×32 patch J_i (Figure 2(e)) from the centerline image I_c (Figure 2(b)) at each junction candidate in I_J (Figure 2(d)). Then the Topology Reconstruction Network (TRNet) is proposed to predict its topology and distinguish each reconstructed curves in a multi-layer form (Figure 2(f)).

Given the input of J_i containing up to K partial curves, we expect the TRNet to predict distinct connected curves into at most K layers. A straightforward idea is to predict K output layers from a single input and minimize the distance to the ground truths. However, this is not appropriate as we cannot assume the orders of these K outputs. Instead, we train the network to focus only on the connectivity of partial curves and ignore their relative orders in the input and output tensors. To do so, we manually assign an order indicator O_i to each input partial curve and ensure the order of curve prediction are still with respect to O_i. This can be achieved by modifying the shape of the input from $32 \times 32 \times 1$ into $32 \times 32 \times (K+1)$. The first layer is the original patch, while each of the rest layers only contains a partial curve, as shown in the network input in Figure 5. An example of order indicator is shown in Figure 6(b). The shape of the output layers is still $32 \times 32 \times K$, representing a maximum of K distinct connected curves, as shown in Figure 6(c).

Since we only care about the topology at the target junction, all unrelated partial curves in J_i should be removed in advance. The removal process consists of two steps, as illustrated in Figure 7. In the first step, we remove all the junctions that we are not interested in (Figure 7(b)). Afterwards, we remove all the isolated curves that are not connected to the target junction (Figure 7(c)).

Network Structure. As illustrated in Figure 5, the TRNet is stacked with two convolutional layers, eight residual blocks, and finally two convolutional layers. Also, there is a skip connection from the network input to the output of the second last convolutional layer to better propagate the orders.

Loss Function. We train the TRNet in a supervised manner with the following loss:

$$\mathcal{L}_T = ||T_i - \hat{T}_i||_2^2, \quad (2)$$

where $||\cdot||_2$ is the L_2-norm. T_i and $\hat{T}_i$ are the ground truth and the predicted topology at the junction J_i respectively.

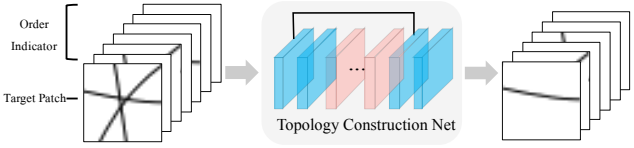


Figure 5: *Structure of Topology Reconstruction Network (TRNet). The $(K+1)$-layer input of TRNet is a junction patch augmented with the order indicator with K layers each of which contains at most one partial curve. The output is K layers each of which contains at most one complete curve.*

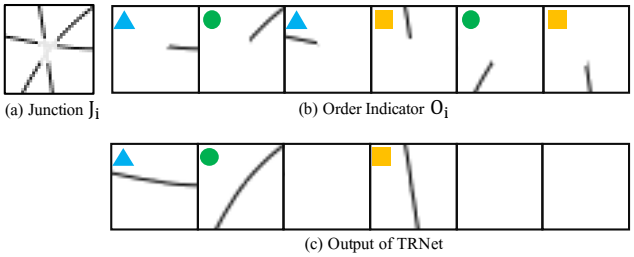


Figure 6: *The layer order indication scheme of TRNet. The output layer order is determined by the input order indicator. Given a junction patch (a), the K-layer order indicator (b) is generated by randomly putting each partial curve in (a) into one layer of (b). The layer order of the network output (c) is uniquely determined by (b).*

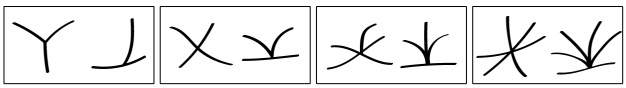
3.4. Dataset Preparation

Although converting a raster image to a vectorized image is an ill-posed problem, the inverse process, i.e., converting a vectorized image to a raster image, is straightforward. So we can create a synthetic dataset by programmatically generating vector drawings and rasterizing them in different rendering configurations.

(a) (b) (c) (d)

Figure 7: *A junction patch cropped from the centerline image I_C may contain pixels that are unrelated to the target junction. Before processed by TRNet, we remove unrelated pixels with two steps. (a) The cropped junction patch; The red pixel indicates the target junction. (b) Step 1: removing unrelated junctions; (c) Step 2: removing partial curves that are not connected to the target junction; (d) Input of TRNet.*

To effectively train the networks and ensure the diversity of the line drawings, we synthesize a dataset that contains curves with various shapes and different kinds of junctions. We generate vector images with cubic Bézier curves. Such curve type is widely used to model smooth paths and is flexible to simulate different shapes of strokes in line drawings. For each vector image, we randomly assign four points to be junctions. For each junction point, we generate a junction with a particular type varying from valence-3 to valence-6. As illustrated in Figure 8, for each type of valence, we freely synthesize the connectivity to maximize the topology diversity of our dataset.



(a) *Valence-3* **(b)** *Valence-4* **(c)** *Valence-5* **(d)** *Valence-6*

Figure 8: *Examples of different types of junctions in our synthesized dataset. Valence-N means that there are N partial curves connected to the target junction.*

Then the training pairs are obtained by rasterizing the above synthesized vector images with a 256×256 canvas. As illustrated in Figure 9, to manually manipulate the width changes along the curve, we randomly select one out of four strategies: gradually growth (Figure 9(a)); first increasing and then decreasing (Figure 9(b)); random width (Figure 9(c)); uniform width (Figure 9(d)). We set a grayness percentage between 0 to 100 for each curve. Then we generate the corresponding centerline image by rasterizing the vector curves into one-pixel-wide, solid black curves (Figure 10(b)). To generate the corresponding junction image, we remove all the unambiguous pixels from the centerline image to preserve ambiguous pixels (Figure 10(c)).

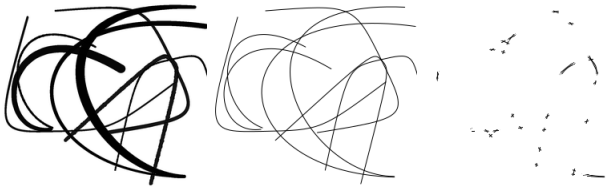
To prepare the training dataset, We generate 20,000 vector images and rasterize them into 20,000 raster training pairs for the Line Subdivision Network (LSNet). Totally 60,000 training pairs for the Topology Reconstruction Network(TRNet) are prepared by cropping local patches as described in Section 3.3.

3.5. Training details

We implement our networks with *TensorFlow* [AAB*15] and separately train LSNet and TRNet on the NVIDIA TITAN V GPU.

(a) (b) (c) (d)

Figure 9: *Four types of lines width mimicking different drawing style. (a) Gradually growth; (b) First increasing and then decreasing; (c) Random width; (d) Uniform width.*



(a) *Input Image* **(b)** *Centerline Image* **(c)** *Junction Image*

Figure 10: *A training pair example of LSNet. (a) Input line drawing with variable line width; (b) One-pixel-wide centerline image; (c) The junction image with junction pixels only.*

We use the Adam optimizer [KB14] with β_1=0.9, β_2=0.999. The learning rate is initially set to $1e^{-4}$, and linearly decreases to $1e^{-6}$. For LSNet, the batch size is set to 3, and the training process takes about 40 hours for totally 80 epochs on a single GPU. For TRNet, the batch size is set to 64, and the training process takes about 3 hours for totally 40 epochs on a single GPU.

4. Experimental Results

To demonstrate the effectiveness of our proposed method, we conduct qualitative and quantitative evaluations on both of the publicly available *Quick, Draw!* dataset [JRK*16] and our newly synthesized dataset. We first separately show the efficacy of each step, including centerline extraction, junction detection, and topology reconstruction. Then, we demonstrate the efficacy of the whole method.

4.1. Centerline Extraction and Junction Detection

We quantitatively evaluate the accuracy of centerline extraction and junction detection on a synthesized testing set that is generated in the same way mentioned in Section 3.4. The testing set contains totally 1000 image pairs and does not overlap with the training set. Each image pair consists of an input image I, a centerline image I_C and a junction image I_J.

We use the weighted absolute difference (WAD) and intersection over union (IoU) metric between the predicted centerline (junction) images and the ground truth to evaluate the framework accuracy. Particularly, WAD only evaluates the non-zero pixels in the ground truth as:

$$WAD = \frac{1}{\sum M} \sum M \odot \left| I - \hat{I} \right|, \tag{3}$$

Where I and $\hat{I}$ are the ground truth and the predicted centerline (junction) images. M is a binary mask whose pixel value equals 1 only if the corresponding pixel in I is non-zero value; $\odot$ is a pixel-wise multiplication operator. The lower WAD value means better accuracy. Besides, IoU measures the overlapping rate of the ground truth and the predicted centerline (junction) images as:

$$IoU = \frac{B \cap \hat{B}}{B \cup \hat{B}} \tag{4}$$

where B and $\hat{B}$ are the binarized ground truth and predicted centerline (junction) image respectively. The threshold for binarization is set to 0.5. The higher IoU value means better accuracy.

Table 1 lists the average WAD and IoU scores over the images of the training and testing set. As scores of the training and testing sets are very similar, we can conclude that our LSNet achieves good generalization ability and is unlikely to overfit.

	Centerline		Junction	
	WAD	IoU	WAD	IoU
Training	7.3012	0.9701	12.0122	0.9089
Testing	7.4410	0.9685	13.5452	0.9002

Table 1: *Quantitative evaluations on LSNet.*

4.2. Topology Reconstruction

TRNet predicts the connectivity of the partial curves at each junction. Examples of input and output of the TRNet are illustrated in Figure 11. We colorize the connected partial curves into different colors. We can see that connectivity of junctions with different valence can be correctly estimated.

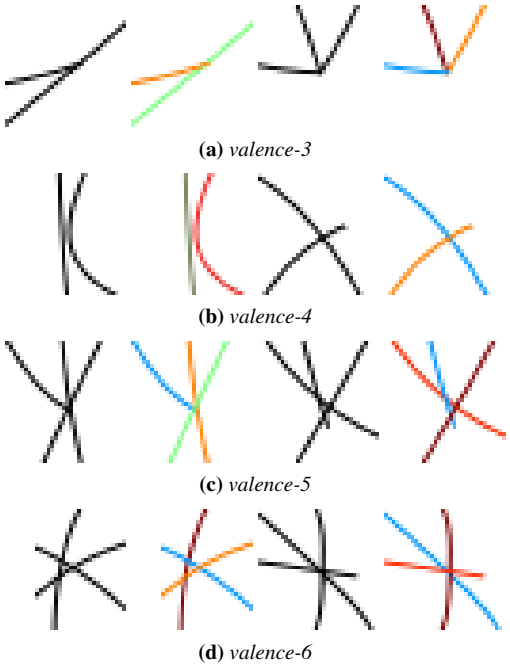
We also quantitatively evaluate the topology construction accuracy for junctions with different valences. The reconstruction is considered successful only if the connection of partial curves is identical to the ground truth. We test 500 junctions for each valence, and the average accuracy is listed in Table 2. We can see that we have achieved high accuracy for Valence-3, 4, 6, and acceptable accuracy for Valence-5. Interestingly, we find that the performance of odd-valence reconstruction is relatively weaker than that of even-valences. This is probably because odd-valence junctions contain more ambiguity in reconstructing joint intersection and termination.

Valence	3	4	5	6
Accuracy	0.976	0.996	0.908	0.970

Table 2: *Topology reconstruction accuracy of TRNet on different junctions types (from Valence-3 to Valence-6).*

4.3. Line Vectorization

We also qualitatively and quantitatively evaluate our vectorization results and compare with those generated by the Favreau's method [FLB16] and Kim's method [KWÖG18]. For Favreau's method, we use the implementation from the author and set all the parameters as default. For Kim's method, we re-train their model with samples from the *Quick, Draw!* dataset in four classes (BASEBALL, CAT, CHANDELIER, and ELEPHANT). We test Favreau's, Kim's, and our methods on two classes (BACKPACK and BICYCLE) of *Quick, Draw!* dataset. For each vector image in the testing set, we raster it into two types of width: 2-pixel width and variable width. Two-pixel width is the default setting in the *Quick, Draw!* dataset, while variable width is to simulate the results drawn by pressure-sensitive pen tablet.



(a) *valence-3*

(b) *valence-4*

(c) *valence-5*

(d) *valence-6*

Figure 11: *Topology reconstruction results of variable junction types. The first and third rows are the input of TRNet, and the second and fourth rows are the output of the TRNet.*

Figure 12 illustrates the vectorization results generates by different methods. Favreau's method generates oversimplified vector images, which obviously deviate from the input image. It also fails to capture non-closure regions. Given the input image with two-pixel-wide curves (Figure 12(c, d, e, and f)), Kim's method generates relatively good results on some simple junctions. However, when there come complicated junctions (e.g., the valence-5 junction in the blow-up in image (e) and valence-6 junction in the blow-up in image (f)), Kim's method fails to get the correct topology. This is because their method can only handle the overlap of two curves. When dealing with variable-width cases, Kim's method may fail to segment the curves (Figure 12(a)) or mistakenly regard a long curve as several short segments (e.g., the bicycle wheel in Figure 12(b)). In sharp contrast, thanks to our two-phase topology analysis, our method can accurately vectorize both the 2-pixel-width and variable-width cases.

We also compare our method with Favreau et al. [FLB16] and Bessmeltsev [BS19] on high-resolution line drawing images. As shown in Figure. 17, Favreau's results get smooth results in sacrifice of accuracy. Bessmeltsev's method can get precise vectorization results but there exist unpleasant line breaks on non-junction areas (e.g., the black box in image *Mouse*). In contrast, our method does not contain line breaking on non-junction regions since we only apply topology reconstruction on the areas. Also, our method outperforms Bessmeltsev's method on complicated junctions with

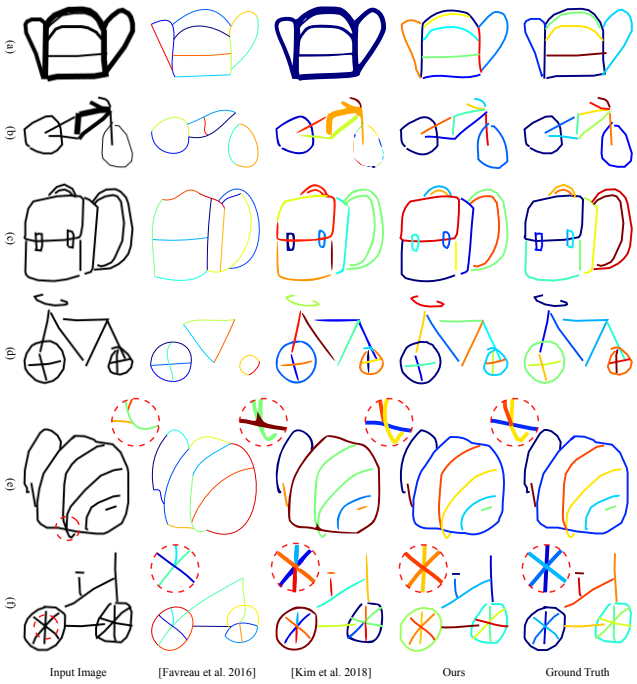


Figure 12: *Vectorization results generated by different methods. The cases in (a) and (b) are of variable width while the other cases are of 2-pixel width. The blowups of the complicated junctions are illustrated in (e) and (f).*

lots of curves (e.g. the black box in image *Dracolion*) and small circles (e.g. the balck box in image *Sheriff*).

We also quantitatively evaluate whether the topology of vectorization results is consistent with the ground truth. Considering that Favreau's and our methods fit splines to each stroke while Kim's method regards each stroke as a region, it's intractable to compare the vectorization results of the three methods directly. So we instead map their line segmentation results back to ground truth lines, such that they can be compared in the same domain. In particular, the ground truth lines are rasterized with a width of 1. Then all pixels of the lines can be classified by a k-nearest-neighbor search from the nearby segmented pixels in the line segmentation results. In our experiment, k is set to 5. With the ground truth and predicted class es, we calculate the stroke IoU (SIoU) as,

$$SIoU = \frac{1}{n} \sum_{i=1,\ldots,n} \max_{j=1,\ldots,m} \frac{P_i \cap \hat{P}_j}{P_i \cup \hat{P}_j}, \tag{5}$$

where P_i and $\hat{P}_j$ are ground truth and predicted strokes. The accuracy of each ground truth stroke is determined by the maximally overlapped predicted stroke. The average SIoU scores over 100 testing images are listed in Table 3. Our method significantly outperforms the two existing methods. Moreover, compared to the two existing methods, our method is more robust to the change of input line width.

4.4. Ablation Study

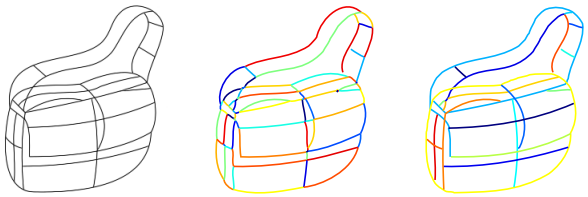
Multi-task Design of Topology Subdivision. Our Topology Subdivision Network is designed to perform centerline extraction and junction detection simultaneously because of their close relation. To further estimate the effectiveness of such multi-task network, we also compare our proposed method with two other possible designs: independent networks and cascade networks. Independent networks tackle these two tasks separately by training two networks with their own training datasets. During testing, the two networks do not rely on each other. On the other side, the cascade network first extracts centerline from the given input and then detects junctions regarding centerline image as input and the two sub-networks are jointly trained. We evaluate the previously mentioned WAD and IoU metrics in equation 3 and 4 of centerline extraction and junction detection on these three designs, and the statistics are listed in Table 4. Our proposed multi-task design outperforms the other two designs probably because our design the two tasks share the same set of local features, making more reliable predictions.

Method	2-pixel width	Variable widths
Favreau [FLB16]	0.4031	0.3721
Kim [KWÖG18]	0.7662	0.2147
Ours	0.8192	0.7576

Table 3: *Average SIoU on Quick, Draw! dataset. Higher is better.*

Design	Centerline		Junction	
	WAD	IoU	WAD	IoU
Independent	8.4399	0.9351	25.7543	0.7423
Cascade	8.4986	0.9343	16.0567	0.8543
Multi-task	7.4410	0.9685	13.5452	0.9002

Table 4: *Comparison of three different LSNet designs in terms of accuracy of centerline extraction and junction detection. For WAD lower is better, while for IoU higher is better. Our multi-task design significantly outperforms the other two designs.*



(a) *Centerline Image* **(b)** *Result w/o TRNet* **(c)** *Result w/ TRNet*

Figure 13: *Vectorization results without and with TRNet. (a) Centerline image generated by LSNet; (b) Vectorization without utilizing topology reconstructed by TRNet; Connectivity at junctions cannot be correctly reconstructed. (c) Vectorization with topology reconstructed by TRNet; The long and smooth curves can be correctly traced due to reconstructed connectivity at junctions.*

Importance of Topology Reconstruction. Our topology reconstruction resolves the ambiguity at junctions. The reconstructed topology is utilized to trace the curves. To demonstrate the importance of topology reconstruction, we disable topology reconstruction and directly conduct curve tracing on centerlines. Given an accurate centerline image illustrated in Figure 13(a), Figure 13(b) is the vectorization result without utilizing the topology reconstructed by TRNet. The vectorization is not satisfying as the connectivity at junctions is not correctly reconstructed, and a complete long curve is regarded as multiple short segments. In sharp contrast, as illus-

trated in Figure 13(c), the long, smooth curves can be correctly traced with the topology reconstructed by our TRNet.

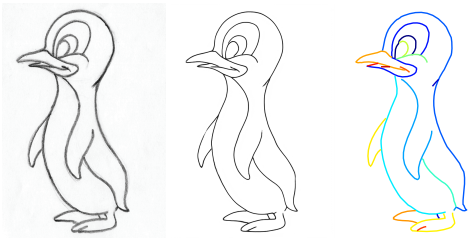
Image size	LSNet	Overall Topology Conduction	Spline Fitting
256×256	0.098	0.118	1.128
512×512	0.312	0.154	1.505
1024×1024	1.03	0.262	2.184

Table 5: *Timing statistics (in seconds).*

4.5. Timing Statistics

We tested our method on a PC with an Intel i7-6700K @ 4.0GHz CPU, 32GB RAM, and a single NVIDIA TITAN V GPU. The running time statistics is composed of two parts. We tested the running time of LSNet, overall topology conduction, and spline fitting for images with three different resolutions. For each resolution, we tested 100 images and calculated the average running time. The results are listed in Table 5. Also, we tested 500 junction patches for the TRNet, and the average running time is 0.021 seconds. As most of the work is done by neural networks, the computational speed is much quicker. At the end of our method, we applied some procedural algorithms to do overall topology conduction and spline fitting. The execution speed is also acceptable.

In sharp contrast, as traditional topology analyzing methods rely on iterative optimization, they are computationally expensive. In particular, both Noris's method [NHS*13] and Bessmeltsev's method [BS19] take dozens of seconds to process a 1024×1024 sketch. Favreau's method [FLB16] even needs several minutes to vectorize a single 1024×1024 image. In the segmentation step, Kim's method [KWÖG18] takes around 5 minutes for an 128×128 image, and it fails to process high-resolution images.

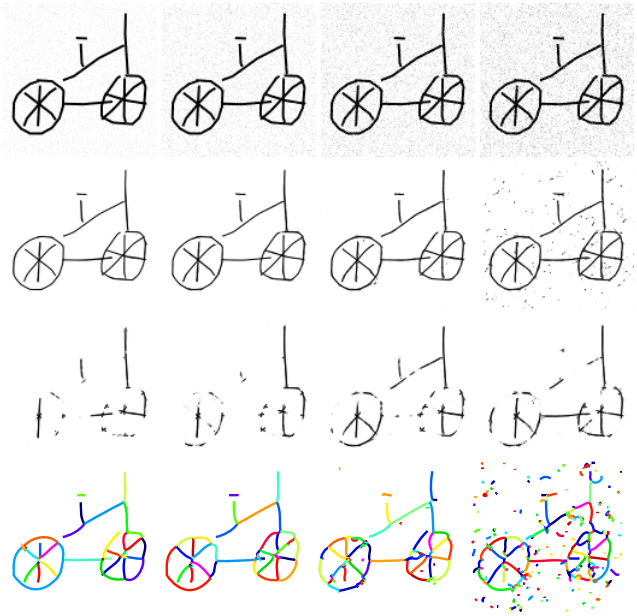


(a) *Original Sketch* **(b)** *Simplification* **(c)** *Vectorization*

Figure 14: *When the input is noisy, we can first conduct line simplification with Simo-Serra's method [SSISI16], and then vectorized the simplified lines with our proposed method. (a) Noisy input line drawing image; (b) Simplified lines by Simo-Serra's method [SSISI16]; (c) Vectorization result with our method.*

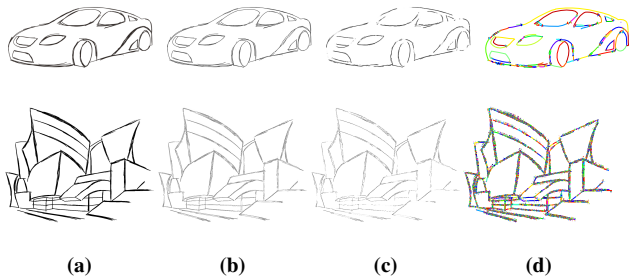
4.6. As Post-processing Step

Our method can also work as a postprocessing step of other raster line drawing processing tools. In particular, a fuzzy raster line drawing can be simplified with Simo-Serra's method [SSISI16] generating a clean line drawing. Then we use our method to vectorize the simplified result. Figure 14 illustrates an example of combining Simo-Serra's method and our method to vectorize a scanned pencil drawing that contains noise. Our vectorization results (Figure 14(c)) are faithful to both the clean line drawings (Figure 14(b)) and the scanned pencil drawings (Figure 14(a)).



(a) $r = 10\%$ **(b)** $r = 20\%$ **(c)** $r = 30\%$ **(d)** $r = 40\%$

Figure 15: *Results of images with Gaussian noise. From the first row to the last row are input images, centerline images, junction images and vectorization results. Note that r represents the ratio of the standard deviation of the Gaussian noise versus that of image values.*



(a) **(b)** **(c)** **(d)**

Figure 16: *Results of sketchy images. From the left to the right are input images(a), centerline images (b), junction images (c) and vectorization results (d).*

4.7. Limitation and Discussion

One of the limitations is that our method is sensitive to image noises. It comes from our assumption that all the input raster images are noise-free so the background of the data is all clean and white. Those abnormal noises will lead to unexpected gradient changes which will greatly harm the accuracy of junction detection of our LSNet. In Figure 15, we show the results on the input images with increasing levels of Gaussian noises. As can be seen in the third row in Figure 15, our proposed LSNet fails to detect junctions even with only 10% Gaussian noise. Therefore, TRNet fails to

reconstruct the topology with inaccurate junction candidates. And it greatly affects our final vectorization results. A workaround is to apply an image smoothing with thresholding to make to background clear before feeding the input image into our network.

We also show the results of the input images with rough and coarse sketches in Figure 16. Such sketchy images usually use multiple lines with severe overlap to describe a single line. However, the goal of our proposed network is to distinguish every single stroke and junction. And our proposed method does not embed any stroke simplification mechanism. That is the reason why lines in our vectorization results of such images are fragile. To obtain a cleaner and better vectorization result, we can apply a post-processing step of stroke simplification that we have mentioned in Section. 4.6.

Currently, our LSNet and TRNet are separately trained, such that the two networks cannot be optimized for each other. It may slightly affect the accuracy of our vectorization results.

5. Conclusion

We propose a two-phase method to accurately vectorize line drawings. In the first phase, we subdivide the lines into partial curves via centerline extraction and junction detection. Then, in the second phase, we reconstruct the topology at each junction. The overall topology generated by the two phases can be directly utilized to trace and vectorize the lines. Thanks to our topology reconstruction, we can get visually better topology reconstruction accuracy comparing to the existing methods. As our core parts are implemented with CNN and can be accelerated by GPU, our method can process high-resolution images in a fast computational speed.

A possible future direction is to realize all the parts of our method with CNN. In this way, all the networks can be jointly optimized, such that the vectorization process will not only be more robust but also achieve faster speed and higher accuracy.

Acknowledgements

This project is supported by the Research Grants Council of the Hong Kong Special Administrative Region, under RGC General Research Fund (Project No. CUHK 14200915 and CUHK 14217516), Guangdong province science and technology plan project (No. 2016A020220013), Shenzhen Science and Technology Program (No. JCYJ20180507182410327 and No. JCYJ20180507182415428).

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: http://tensorflow.org/. 5

[BS19] BESSMELTSEV M., SOLOMON J.: Vectorization of line drawings via polyvector fields. *ACM Trans. Graph. 38* (2019), 9:1–9:12. 1, 2, 6, 8

[Car97] CARUANA R.: Multitask learning. *Machine learning 28*, 1 (1997), 41–75. 3

[CFL13] CHAI D., FORSTNER W., LAFARGE F.: Recovering line-networks in images by junction-point processes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 1894–1901. 2

[CTYX17] CHEN Y., TU S., YI Y., XU L.: Sketch-pix2seq: a model to generate sketches of multiple categories. *arXiv preprint arXiv:1709.04121* (2017). 2

[CY98] CHANG H.-H., YAN H.: Vectorization of hand-drawn image using piecewise cubic bezier curves fitting. *Pattern recognition 31*, 11 (1998), 1747–1755. 2

[DCP17] DONATI L., CESANO S., PRATI A.: An accurate system for fashion hand-drawn sketches vectorization. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2280–2286. 2

[FLB16] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 120. 2, 6, 7, 8

[HE17] HA D., ECK D.: A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477* (2017). 2

[HT06] HILAIRE X., TOMBRE K.: Robust and accurate vectorization of line drawings. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 6 (2006), 890–904. 2

[JRK*16] JONGEJAN J., ROWLEY H., KAWASHIMA T., KIM J., FOX-GIEG N.: The quick, draw!-ai experiment. *Mount View, CA, accessed Feb 17* (2016), 2018. URL: https://quickdraw.withgoogle.com/data. 5

[JV97] JANSSEN R. D., VOSSEPOEL A. M.: Adaptive vectorization of line drawing images. *Computer vision and image understanding 65*, 1 (1997), 38–56. 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 5

[Kha07] KHAN M.: Approximation of data using cubic bezier curve least square fitting. *Nur Online verfügbar: http://130.235 212* (2007). 3

[KWÖG18] KIM B., WANG O., ÖZTIRELI A. C., GROSS M.: Semantic segmentation for line drawing vectorization using neural networks. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 329–338. 1, 2, 6, 7, 8

[LL06] LECOT G., LEVY B.: Ardeco: automatic region detection and conversion. In *17th Eurographics Symposium on Rendering-EGSR'06* (2006), pp. 349–360. 2

[NHS*13] NORIS G., HORNUNG A., SUMNER R. W., SIMMONS M., GROSS M.: Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics (TOG) 32*, 1 (2013), 4. 1, 2, 8

[OBB*13] ORZAN A., BOUSSEAU A., BARLA P., WINNEMÖLLER H., THOLLOT J., SALESIN D.: Diffusion curves: A vector representation for smooth-shaded images. *Communications of the ACM 56*, 7 (2013), 101–108. 2

[SSII18] SIMO-SERRA E., IIZUKA S., ISHIKAWA H.: Real-time data-driven interactive rough sketch inking. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 98. 1, 2, 3

[SSISI16] SIMO-SERRA E., IIZUKA S., SASAKI K., ISHIKAWA H.: Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 121. 2, 8

[WRS*09] WHITED B., ROSSIGNAC J., SLABAUGH G., FANG T., UNAL G.: Pearling: Stroke segmentation with crusted pearl strings. *Pattern Recognition and Image Analysis 19*, 2 (2009), 277–283. 2

[WSCY15] WANG J., SONG J., CHEN M., YANG Z.: Road network extraction: A neural-dynamic framework based on deep learning and a finite state machine. *International Journal of Remote Sensing 36*, 12 (2015), 3144–3169. 2

[ZS84] ZHANG T., SUEN C. Y.: A fast parallel algorithm for thinning digital patterns. *Communications of the ACM 27*, 3 (1984), 236–239. 3

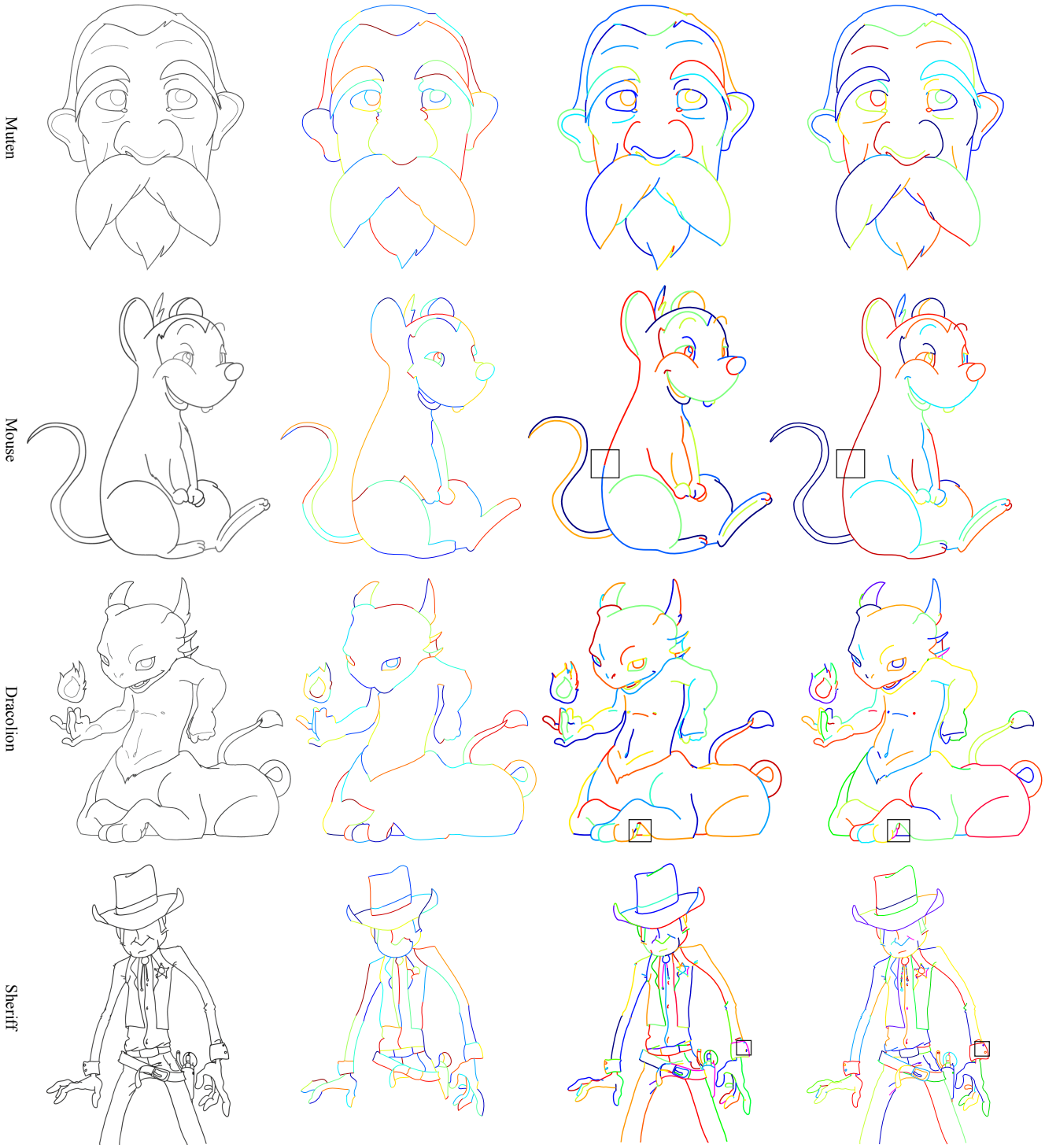


Figure 17: *Vectorization results on high-resolution line drawings. The resolution of all input images is* 1024×1024.