Volume xx (200y), Number z, pp. 1–12

Line Drawing Vectorization via Coarse-to-Fine Curve Network Optimization

Bin Bao[1] and Hongbo Fu[2]

[1]School of Data and Computer Science, Shandong Women's University
[2]School of Creative Media, City University of Hong Kong

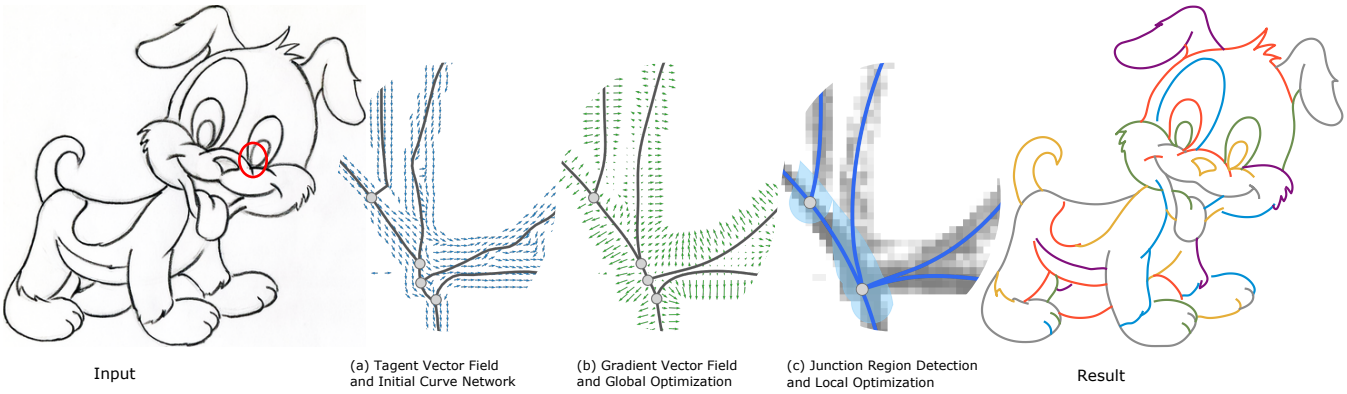

Figure 1: *Given a line drawing image, our method first derives a tangent vector field from the input image and traces it to get an initial curve network (a). It then performs a global optimization that fits the curve network to image centerlines by taking the gradient vector field as an external force (b). Our method finally detects junction regions based on (b) and performs a local optimization in each junction region (c). Our coarse-to-fine optimization framework is able to produce high-quality curves (as shown right) with low computational cost.*

Abstract

Vectorizing line drawings is a fundamental component of the workflow in various applications such as graphic design and computer animation. A practical vectorization tool is desired to produce high-quality curves that are faithful to the original inputs and close to the connectivity of human drawings. The existing line vectorization approaches either suffer from low geometry accuracy or incorrect connectivity for noisy inputs or detailed complex drawings. We propose a novel line drawing vectorization framework based on coarse-to-fine curve network optimization. Our technique starts with an initial curve network generated by an existing tracing method. It then performs a global optimization which fits the curve network to image centerlines. Finally, our method performs a finer optimization in local junction regions to achieve better connectivity and curve geometry around junctions. We qualitatively and quantitatively evaluate our system on line drawings with varying image quality and shape complexity, and show that our technique outperforms existing works in terms of curve quality and computational time.

CCS Concepts

• ***Computing methodologies*** → *Image manipulation; Shape modeling;*

1. Introduction

The vectorization of line drawings is an important component of the workflow in graphic design and computer animation. It serves as a preprocessing step for many applications such as animated construction of line drawings [FZLM11] and creation of curve-based vector graphics [OBW*08, FSH11]. A practical vectorization tool is desired to produce high-quality curves that are faithful to the input line drawings and close to the connectivity of human drawings. The quality of vectorization has a high impact on downstream applications.

Software tools for vectorization of line art include Adobe Live Trace, Inkscape Potrace [Sel03], VectorMagic, and WinTopo

[ZS84], etc. The existing tools are efficient but often fail to deal with detailed complex drawings. The traditional methods based on tracing [BF12] or skeletonization [NHS*13] are not robust to noisy inputs or complex junctions. The modern techniques rely on deep learning or frame fields [BS19]. However, the deep learning approaches [GZH*19, MSSG*21] are not good at geometry modeling, even in areas without ambiguity. Frame fields robustly distinguish directions around junctions. However, extracting curves from a frame field often produces artifacts. The approach of [BS19] traced along the frame field and thus only distinguishes X- and T-junctions. Puhachov et al. [PNCB21] improved curve extraction based on image junction detection, which, however, leads to incorrect connectivity due to lack of detection of Y-junctions. A quick comparison to existing methods is demonstrated in Figure 2.

To effectively and efficiently produce high-quality curves, we propose a novel coarse-to-fine optimization framework. We observe that compared with ambiguous directions in noisy and junction areas, image gradients containing magnitudes are more reliable (see a comparison in Figure 1 (a) and (b)). Thus by taking image gradients as external forces, our optimization algorithm always keeps curves located at image centerlines. Our method starts with a curve network initialized by an existing tracing method [BF12] (Figure 1 (a)). It then performs a global optimization, leading to accurate curve geometry in non-junction regions and coarse refinement of junction positions (Figure 1 (b)). Our method finally detects junction regions based on the coarse solution and perform a local optimization in each junction region, aiming to achieve more continuous connectivity and better curve geometry around junctions (Figure 1 (c)).

Our system is efficient since our optimization solvers perform on sparse curve points rather than dense pixels. In contrast, the frame field-based methods [BS19, PNCB21] computed the frame field by a global optimization on a pixel grid, and deep learning-based methods [GZH*19, MSSG*21] also make computations on image pixels.

To demonstrate the effectiveness of our proposed method, we qualitatively and quantitatively evaluate our system on line drawings with varying image quality and shape complexity. We show that our technique outperforms existing works in terms of curve quality and computational speed. The main contributions of our work are as follows:

- A novel gradient-driven curve network optimization algorithm for line vectorization, which robustly forces a given curve network to fit the centerlines of an input line drawing image.
- A novel line drawing vectorization framework based on coarse-to-fine curve network optimization, which is able to produce high-quality curves with low computational cost.

2. Related Work

Vectorization of Line Drawings. The early studies on line vectorization often suffer from serious artifacts on line art with complex shapes, including the tracing methods [SSC*00, SSTC02] and the skeletonization methods [ZS84, ZY01]. Improved tracing approaches [BF12, CLMP15, NS19] can produce more promising tracing results. However, they still fail on ambiguous junctions due to unreliable tracing guides. Recently, Mo et al. [MSSG*21] proposed a learning-based tracing approach. Their deep learning framework focuses on image coverage and thus has few advantages in capturing geometry and topology. In contrast, our method takes the tracing process as an initial step, and then corrects the geometry and topology by our coarse-to-fine optimization algorithm.

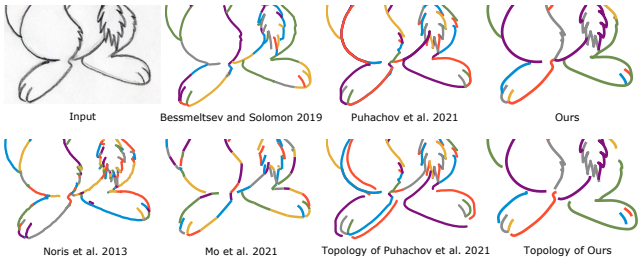

Figure 2: *The traditional method [Noris et al. 2013] is sensitive to noisy inputs. The deep learning approach [Mo et al. 2021] suffers from geometrical artifacts. Frame field-based approaches [Bessmeltsev and Solomon 2019; Puhachov et al. 2021] are more precise but suffer from incorrect connectivity at complex junctions. The topology comparison shows that Puhachov et al.'s results contain long redundant curves (for clarity we draw them with small offsets), though the connectivity of these curves is visually correct. Our method produces higher quality curves in terms of geometry and topology.*

Recently, many learning-based methods have been proposed for this task. However, these works are either not suitable for noisy inputs (e.g., noisy scanned pencil drawings in Figure 10) [KWÖG18, GZH*19, BCY*21, DYH*21], or not good at geometry modeling on detailed complex drawings [EVA*20, MSSG*21].

The recent frame field-based methods target noisy scanned inputs and detailed complex drawings. For example, Bessmeltsev and Solomon [BS19] proposed the frame field to disambiguate directions. However, tracing along the frame field only distinguishes X- and T-junctions. Stanko et al. [SBBB20] explored a parameterization approach, but it tends to over-simplify the shapes. Puhachov et al. [PNCB21] improved curve extraction by detecting junctions, but introduced extra connectivity artifacts due to lack of Y-junction detection. Besides, frame field-based methods are high computational complexity, thus greatly limiting their practical usage. In contrast, our method is more efficient and produces results with better connectivity (see comparisons in Figure 2 and Section 5).

There exist several techniques focusing on topology reconstruction [NHS*13, GZH*19]. We are inspired by these approaches that first built base centerlines and then reconstructed the topology at junctions. Noris et al. [NHS*13] removed the centerlines inside the junction regions and reconnected those outside. Their approach might make the reconstructed curves deviate from the original image lines. Our optimization method avoids such problems by always keeping curves located at image centerlines. Guo et al. [GZH*19] proposed a learning-based method. However, their model is not suitable for noisy inputs as shown in their article. In contrast, our system works well on both clean and noisy drawings.

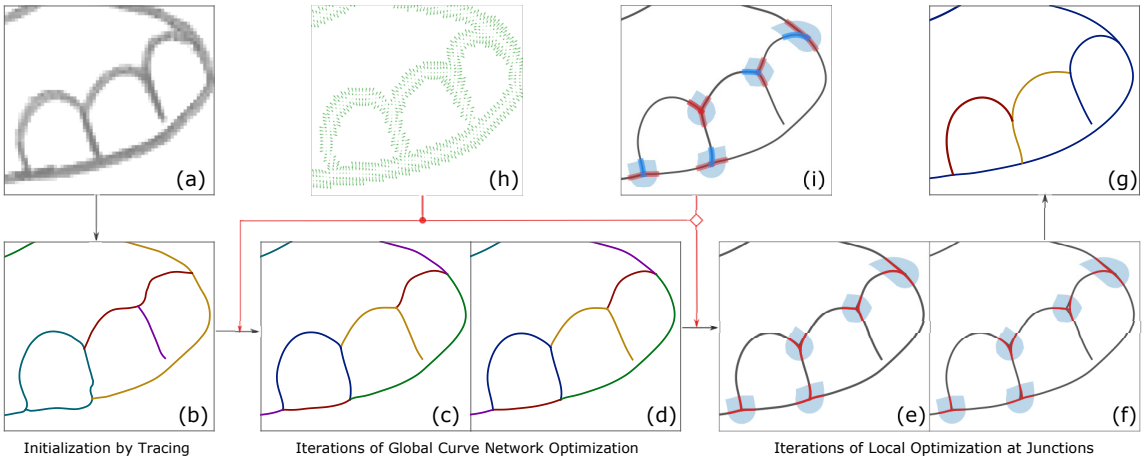

Figure 3: *Workflow of our proposed technique. (a) shows a part of an input line drawing. (b) is the initial curve network generated by an existing tracing approach. We optimize (b) in two steps: (c)-(d) globally optimize the overall geometry and junction positions of the curve network; and (e)-(f) locally optimize the junction positions and curve pieces in junction regions, leading to the final result (g). (h) and (i) serve as the external forces. (h) is the image gradients, and (i) is the connectivity inferred from (d): a curve piece is either continuously connected with another or not (marked in red or blue).*

Sketchy Drawings Vectorization. Vectorization methods for sketchy line drawings [FLB16, PPM18] adopted a region-based manner, and were not suitable for line drawings with many open curves. Besides, these approaches designed for sketchy drawings focus on simplifying sketches and thus often fail to deal with complex shapes, even on clean line drawings.

Sketchy Simplification. Sketchy image simplification aims to transfer sketchy line drawings to clean drawings. The existing solutions for this task include local filter-based methods [BCF*07, DCP17], and deep learning-based approaches [SSISI16, SSII18, XXM*21, MKDM22]. The difference between these methods and ours is that they generate bitmap images while our work produces vector curves. The sketchy image simplification tools can work as a pre-processing step for vectorizing line drawings.

Another related problem is sketchy vector curve simplification, including the clustering methods [BTS05, LRS18], the region-based approach [LWH15] and the global optimization approach [PvMLV*21]. The difference is that these works take vectors as inputs and aim to generate clean curves, while our method produces vectors from the input rasterized bitmap images.

3. Overview

The workflow of our approach is shown in Figure 3. We first adopt a tracing approach to generate an initial curve network, which captures the raw geometry and connectivity (Figure 3 (b) and Section 4.1). We then refine the initial curve network in two optimization steps: (1) globally optimize the overall geometry and junction positions by taking the image gradients to force the curve network to image centerlines (Figure 3 (c)-(d) and Section 4.2); and (2) locally optimize the junction position and curve pieces in each junction region (Figure 3 (e)-(f) and Section 4.3) by taking the connectivity inferred from the result of the global optimization step as soft constraints (Figure 3 (i)). At each step, we formulate an energy minimization problem and employ an iterative solver. Our optimization solvers can converge in several iterations.

4. Methodology

4.1. Initialization by Tracing

To produce an initial curve network, we use a tracing method based on [BF12]. The method of [BF12] distinguishes local ambiguities due to: (1) tracing directions determined by an orientation field and the front-end direction of tracing lines; and (2) cross sections used to distinguish multiple underlying lines which meet at the pixel level due to anti-aliasing. Our method differs from [BF12] by making the following modifications: (1) we compute the orientation field more efficiently by simply adopting a vector field perpendicular to the image gradient; and (2) our tracing algorithm does not deal with connection cases and geometry appearance, since they will be addressed in the subsequent optimization steps. Thus, our tracing process is much faster than [BF12].

4.2. Global Curve Network Optimization

Problem Formulation. We present the initial tracing curve network as a graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ corresponds to the junction points and endpoints, and $\mathcal{E}$ corresponds to the curves. Each edge $e \in \mathcal{E}$ is associated with a curve, denoted as $c_e = \{p_{e,1}, p_{e,2}, \ldots, p_{e,n}\}$, where $p_{e,i}$ is the i th curve point coordinate. Our goal is to optimize the curve network to satisfy the following constraints: (1) each c_e is a smooth curve; (2) each c_e locates at image centerlines; and (3) the connections defined by the graph G keep invariant. We thus define the energy minimization problem as

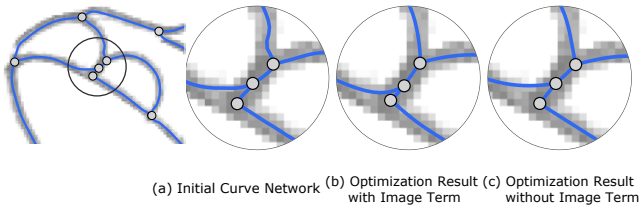

Figure 4: *Comparison of with and without the image term E_{image}. (a) is an initial curve network. (b) is the result with E_{image}, which forces the curve network to the centerlines of input line drawings. (c) shows the result without E_{image}. The comparison shows that E_{image} plays an important role in correcting both curve geometry and junction positions.*

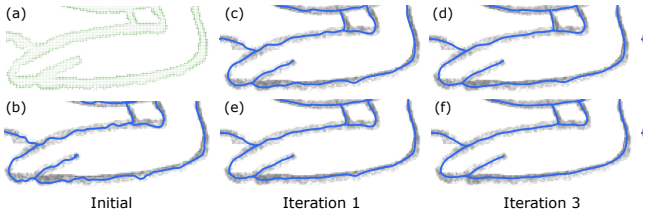

Figure 5: *Results of global curve network optimization for 1 and 3 iterations. (a) visualizes the gradients of the input image and (b) is the initial curve network. (e)-(f) shows the iterations that take the image gradients as external forces, and (c)-(d) shows the iterations without the gradient forces. In comparison, (e)-(f) converges much faster than (c)-(d).*

follows:

$$\begin{cases} \min\limits_{\{c_e\}} \sum\limits_{e\in\mathcal{E}} (E_{smooth}(c_e) + w_d E_{image}(c_e)), \\ s.t. \forall c_e \in \mathcal{N}_e(v)\ endpoints\ are\ equal, \forall v \in \mathcal{V} \end{cases}, \tag{1}$$

where w_d is the balance weight (empirically $w_d = 0.2$), $\mathcal{N}_e(v)$ is the neighboring edges of v, E_{smooth} and E_{image} are defined to satisfy the first and second constraints, respectively, and the equality equations correspond to the third constraint.

The smooth term E_{smooth} encourages the curve to be more continuous while keeping the sections of high curvature by measuring the sum of first- and second-order derivatives over the curve. Specifically, we define E_{smooth} as:

$$\begin{aligned} E_{smooth}(c_e) = w_1 \sum_{i=1}^{|c_e|-1} \|p_{e,i+1} - p_{e,i}\|^2 \\ + w_2 \sum_{i=2}^{|c_e|-1} \|p_{e,i+1} + p_{e,i-1} - 2p_{e,i}\|^2, \end{aligned} \tag{2}$$

where w_1 and w_2 are two balancing weights (empirically $w_1 = 0.8, w_2 = 0.5$).

We introduce the image term E_{image} by observing that image centerlines locate at the minima of image intensity. We thus define the image term as the sum of image intensity at all curve positions:

$$E_{image}(c_e) = \sum_{i=1}^{|c_e|} I(p_{e,i}), \tag{3}$$

where I denotes the input image. Note that E_{image} is the key to correcting curve geometry and junction positions. Figure 4 illustrates the effects by comparing the optimization results with and without the image term (Figure 4 (b) and (c)).

Problem Solver. Computing all points of the curve network at once in Equation 1 is computationally expensive. Independently solving each curve is faster but would make the constraints of Equation 1 unsatisfied. To address this issue, we adopt an iterative approach which alternately updates the coordinates of curves and the junction points. At each iteration, we update each curve by adopting a gradient descent approach and update the junctions as the average of the endpoints of adjacent curves. According to implicit Euler, we define the formula for updating the points of curve c_e as follows:

$$P_e^{t+1} = (\lambda(w_1 L_1 + w_2 L_2) + \mathbb{I})^{-1}(P_e^t - \lambda w_d \nabla I_{\vec{n}}(P_e^t)), \tag{4}$$

with

$$L1 = \begin{bmatrix} 0 & 0 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & 0 & 0 & \end{bmatrix}, L2 = \begin{bmatrix} 0 & 0 & 0 & & & & \\ -2 & 5 & -4 & 1 & & & \\ 1 & -4 & 6 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & -1 & -4 & 5 & -2 \\ & & & & 0 & 0 & 0 \end{bmatrix},$$

where $P_e^t = [p_{e,1}^t, p_{e,2}^t, \ldots, p_{e,n}^t]^T$ are the curve points at the t-th iteration, λ is a fixed step size (empirically set to 2 by default), $\mathbb{I}$ is the identity matrix, $\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]^T$ represents the gradients of the input image I, and $\nabla I_{\vec{n}} = \langle \nabla I, \vec{n} \rangle \vec{n}$ is the projection of ∇I to the curve's normal vectors.

Figure 5 shows the results by Equation 4 for 1 and 3 iterations. The external force $\nabla I_{\vec{n}}$ makes the iterations converge fast (typically 3 iterations by default).

4.3. Local Optimization at Junctions

Based on the optimized curve network generated by Section 4.2, we first detect junction regions and then optimize the junction position and curve pieces inside each junction region. An overview of our algorithm is illustrated in Figure 6.

Junction Region Detection. The junctions of an initial curve network by tracing might deviate from the ground truth, especially in ambiguous areas where many lines meet. We detect the ambiguous regions by clustering junctions of the graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$ defined in Section 4.2, and each cluster defines a new junction region (Figure 6 (h)). Specifically, we perform a typical clustering algorithm [JK05], by taking the distance function defined as:

$$d(v,u) = \begin{cases} 1 & \frac{R(v) \cap R(u)}{max(R(v), R(u))} > 0.3 \\ 0 & otherwise \end{cases}, \tag{5}$$

where $R(v)$ is the polygon region at junction $v \in \mathcal{V}$ within a distance r (empirically $r \in [2, 5]$), as shown in Figure 6 (g).

Optimization in Junction Regions. Given a junction region, we divide the curves that meet in the area into two sets of curve

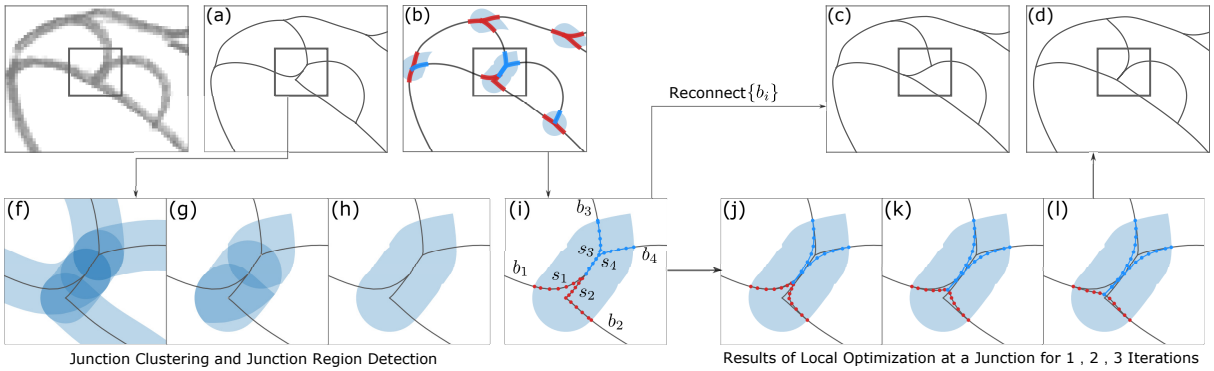

Figure 6: *The workflow of local optimization around junctions. (a) shows the curve network produced by global curve network optimization. We first detect the junction regions (f)-(h) and then estimate the connectivity at junctions (b): a curve piece is either continuously connected with another or not (marked in red or blue). To avoid reconnection artifacts (c), we optimize the curve pieces by taking the connectivity as soft constraints (j)-(l), leading to a correct reconstruction result (d).*

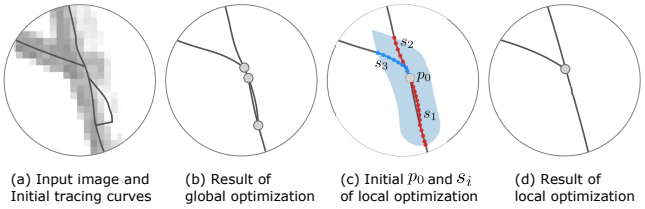

Figure 7: *Illustration of the initial p_0 and s_i, in the case of clustering multiple initial junctions (b) into one junction region (c). (a) shows the input image and initial curves. (d) is the final result.*

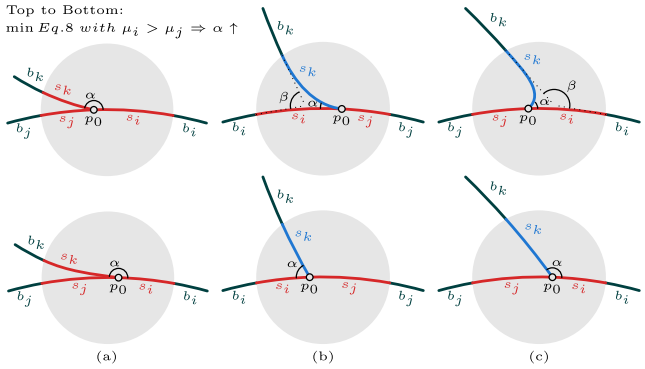

Figure 8: *Top to Bottom: Minimizing Equation 8 with $\mu_i > \mu_j$ makes p_0 move towards s_i, causing α to increase. This changes the shape of s_k in two cases: (a) making s_k smoothly merge to s_i for $(s_k, s_i) \in \mathcal{T}$; and (b)-(c) forcing the other s_k to be nearly straight (suppose s_i is the one that satisfies $\alpha < \beta$).*

pieces. We denote the curve pieces inside and outside the junction region as $\mathcal{S} = \{s_i\}$ and $\mathcal{B} = \{b_i\}$, respectively, where $s_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,n}\}$ and $b_i = \{q_{i,1}, q_{i,2}, \ldots, q_{i,m}\}$ (see an example in Figure 6 (i)). We estimate the connectivity among $\mathcal{S}$ as connection pairs, denoted as $\mathcal{T} = \{(s_i, s_j)\}$. $(s_i, s_j) \in \mathcal{T}$ if the maxima curvature of the Hermit spline of b_i and b_j is less than a threshold. A curve piece s_i is either included in $\mathcal{T}$ or not (marked as red or blue in Figure 6 (b) and (i)). Our goal is to reconstruct the curve pieces inside the junction region ($\mathcal{S}$) according to the estimated connectivity. Directly reconnecting the curve pieces outside ($\mathcal{B}$) might produce curves that deviate from the input image (Figure 6 (c)). To avoid such artifacts, we optimize the junction position and curve pieces by taking the connectivity as a soft constraint and the image term (see Equation 3 in Section 4.2) as another constraint that penalizes deviation (Figure 6 (j)-(l) and (d)).

The goal of the local optimization in a junction region is to optimize the curve pieces $\mathcal{S} = \{s_i\}$ and the junction point p_0 where they meet, by enforcing the following constraints: (1) each s_i locates at image centerlines; (2) each pair (s_i, b_i) is continuously connected; and (3) each pair $(s_i, s_j) \in \mathcal{T}$ is continuously connected, and for the other curve pieces not included $\mathcal{T}$ $((s_o, s_l) \notin \mathcal{T}\ \forall s_l)$, s_o is the linear extension of b_o. Mathematically, we thus define the energy minimization problem as follows:

$$\begin{cases} \min\limits_{\mathcal{S}} E_{continue} + w_t E_{connect} + w_d E_{image} \\ s.t.\ p_{i,0} = p_0 \ \ \forall s_i \in \mathcal{S} \end{cases}, \tag{6}$$

where w_t and w_d are balancing weights (empirically $w_t = 0.8, w_d = 0.2$), E_{image} is defined in Equation 3 to satisfy the first constraint, $E_{continue}$ and $E_{connect}$ are defined to satisfy the second and third constraints, respectively. Initially, we set p_0 as the one closest to the center of the clustered junction region and remove the curve pieces fully inside the region (Figure 7 (b) (c)). As Figure 7 shows, it might lead to topology simplification (see more results in Figure 12 (c) (d) and Figure 14).

To encourage each s_i to be continuous and keep s_i and b_i to be G^1 continuous, we define $E_{continue}$ as:

$$\begin{aligned} E_{continue} = &\sum_{i=1}^{|\mathcal{S}|} \sum_{j=1}^{|s_i|-1} \|p_{i,j+1} - p_{i,j}\|^2 \\ &+ \sum_{i=1}^{|\mathcal{S}|} \|(p_{i,n} - p_{i,n-1}) - (q_{i,1} - p_{i,n})\|^2. \end{aligned} \tag{7}$$

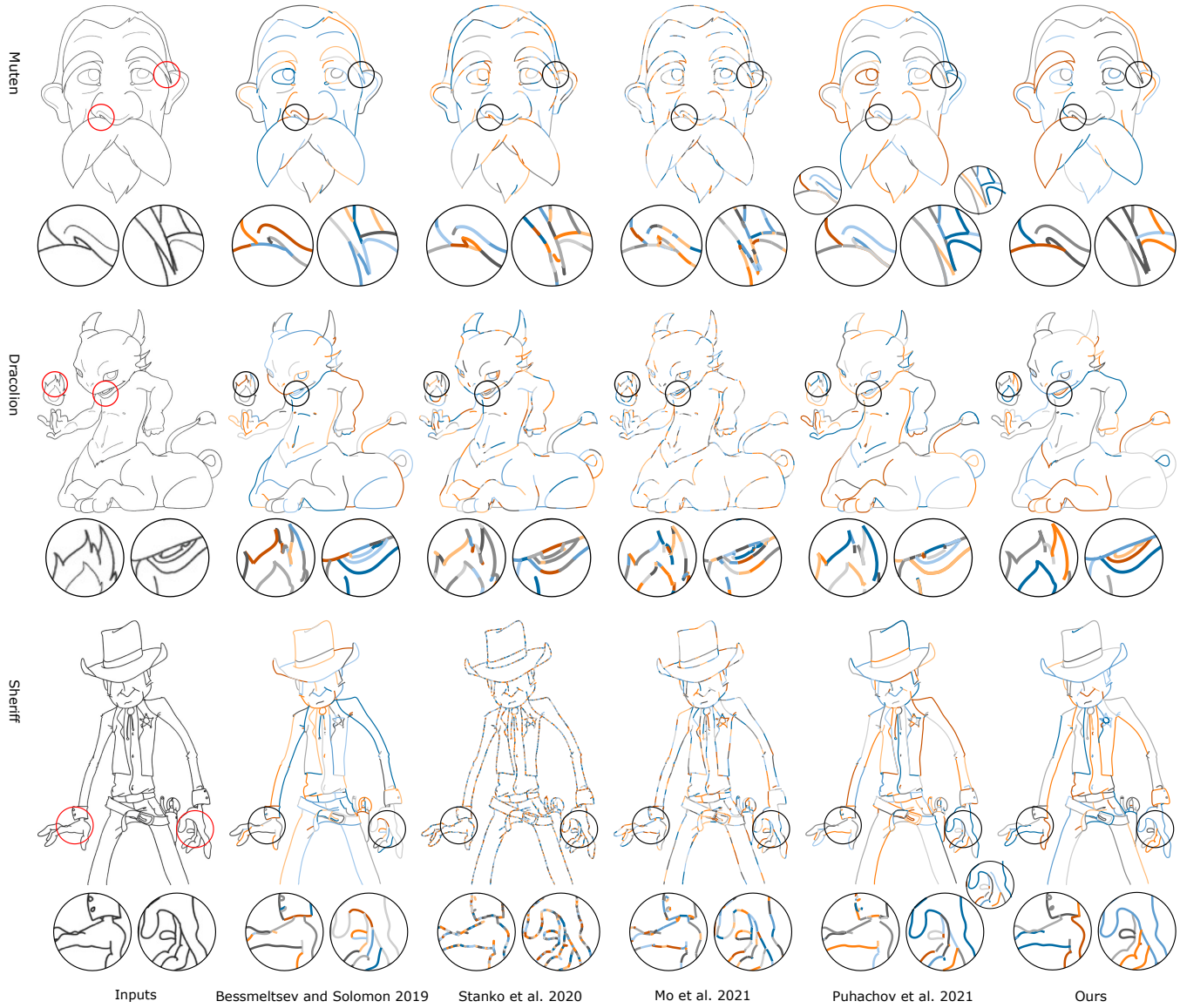

Figure 9: *Comparisons on clean digital inputs. Note that the topology of Puhachov et al.'s results might be incorrect though they look visually correct (as shown in the small circles; for clarity we draw them with small offsets).*

To satisfy the third constraint, each pair $(s_i, s_j) \in \mathcal{T}$ should have the same tangent directions at the connecting point p_0. Thus we formulate the third constraint as $\mu_i(p_0 - p_{i,1}) = \mu_j(p_{j,1} - p_0)$ for each $(s_i, s_j) \in \mathcal{T}$. We set $\mu_i \neq \mu_j$ to control the shape of the other curve pieces $s_k \in \mathcal{S} \backslash \{s_i, s_j\}$. As Figure 8 shows, $\mu_i > \mu_j$ would increase the angle between s_k and s_i, thus either making s_k smoothly merge to s_i (Figure 8 (a)) or forcing s_k to be nearly straight (Figure 8 (b) and (c)). The larger difference between μ_i and μ_j, the more curve shapes would change. We thus define $E_{connect}$ as:

$$E_{connect} = \sum_{(s_i,s_j)\in\mathcal{T}} \|\mu_i(p_0 - p_{i,1}) - \mu_j(p_{j,1} - p_0)\|^2, \quad (8)$$

where μ_i and μ_j are initialized as 1 for each pair $(s_i, s_j) \in \mathcal{T}$, and μ_i is increased for each curve piece $s_k \in \mathcal{S} \backslash \{s_i, s_j\}$:

$$\mu_i = \begin{cases} \mu_i + C(1 - e^{-\frac{\pi - \theta(s_k, s_i)}{\pi}}) & (s_k, s_i) \in \mathcal{T} \\ \mu_i + C(1 - e^{-\kappa(s_k)}) & (s_k, s_i) \notin \mathcal{T}, (s_k, s_j) \notin \mathcal{T}, \\ & \theta(s_k, s_i) < \theta(b_k, b_i) \end{cases}, \quad (9)$$

where $\theta(s_k, s_i)$ is the angle between s_k and s_i, $\theta(b_k, b_i)$ is the angle between b_k and b_i, $\kappa(s_k)$ is the maxima curvature over the curve piece s_k, and C is a constant (set to 10 by default).

Like Equation 1 in Section 4.2, Equation 6 is also solved by an iterative gradient descent approach as illustrated in Figure 6 (j)-(l).

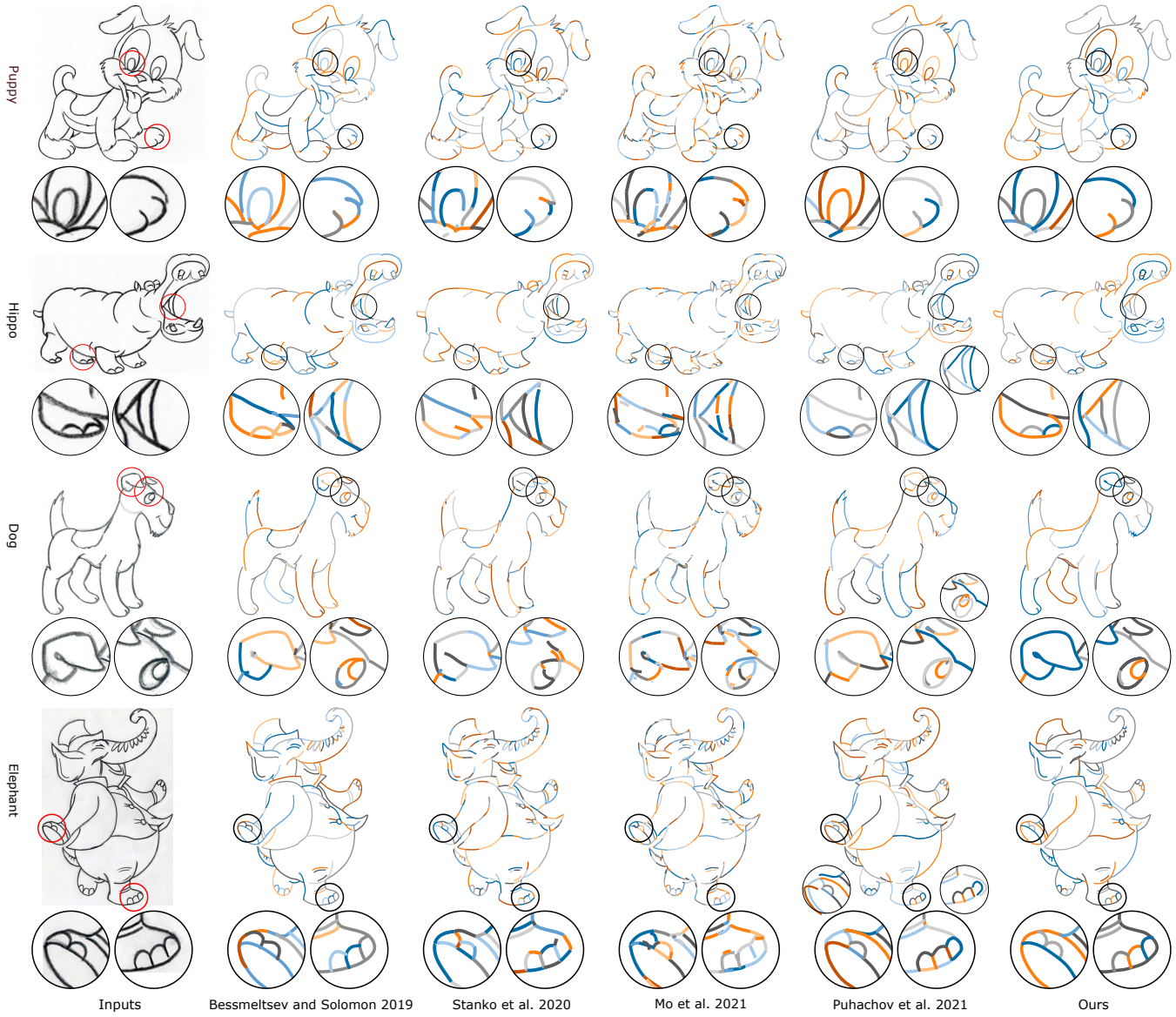

Figure 10: *Comparisons on noisy scanned inputs. Note that the topology of Puhachov et al.'s results might be incorrect though they look visually correct (as shown in the small circles; for clarity we draw them with small offsets).*

5. Results and Discussion

To demonstrate the effectiveness of our proposed method, we qualitatively and quantitatively evaluate our results (Sections 5.1 and 5.2). We show the efficiency of our method in Section 5.3 and show our method as a post-processing step in Section 5.4. We discuss the robustness in Section 5.5 and the limitations in Section 5.6.

5.1. Results and Comparison

We have tested our methods on a variety of line drawings with varying image quality and shape complexity, as shown in Figures 9, 10, 11, and 16.

Comparisons. We compare our method with the recent morden algorithms, including three frame field-based methods [BS19, SBBB20, PNCB21], and one deep learning method [MSSG*21]. We take the results presented in the compared papers if any. Otherwise, we produce the results by the authors' code of the corresponding papers. We ran the code of [SBBB20] with optimized parameters, and for [MSSG*21], we ran their code for several trials and selected visually the best ones. To show the comparison in term of topology, we recolored all the results in the same color palette.

Figures 9 and 10 show that our method produces more continuous curves close to human drawings and is more robust on

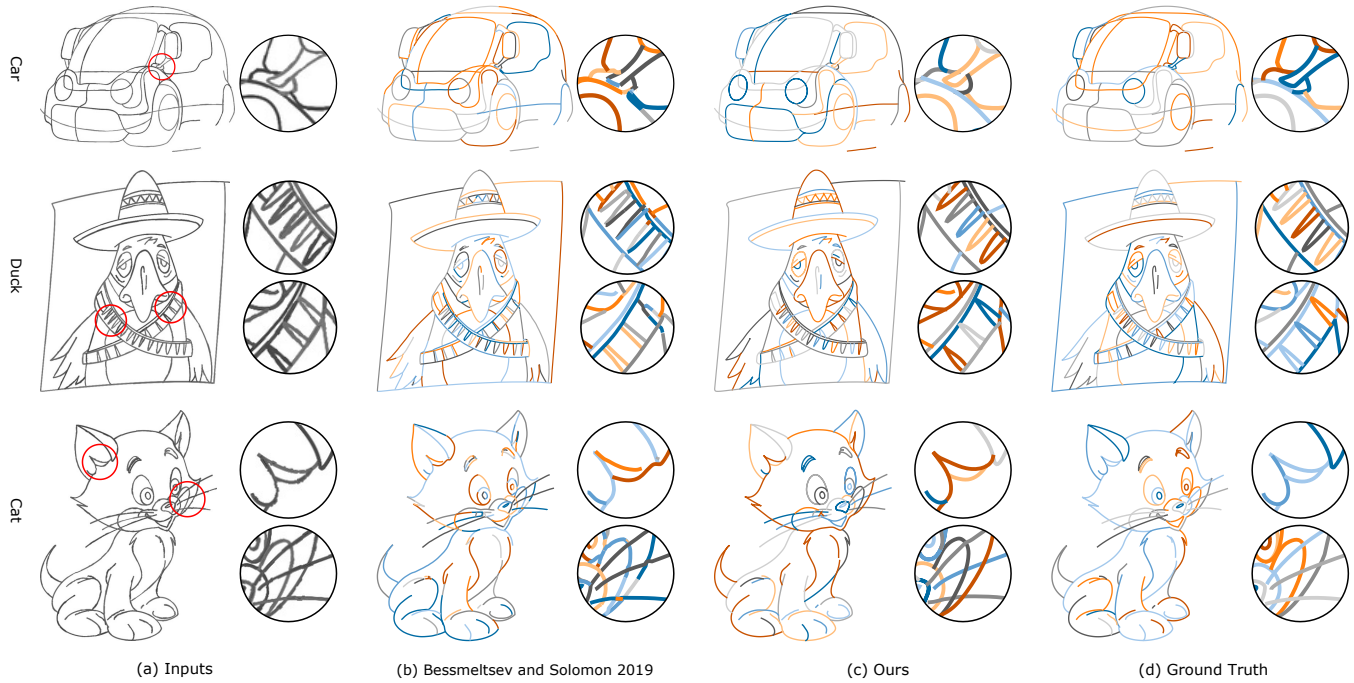

Figure 11: *Comparisons to the ground truth. The inputs (a) are created from the ground truth vector drawings (d). The comparison shows that our results are much closer to the ground truth.*

Examples	Mo et al.			Bessmeltsev and Solomon			Ours			Ground Truth
	Chamfer	F-socre	L	Chamfer	F-socre	L	Chamfer	F-socre	L	L
Car	0.00046	0.740	0.049	0.00038	0.794	0.296	**0.00026**	**0.869**	**0.717**	0.597
Glasses	0.00025	0.861	0.047	**0.00020**	0.898	0.321	**0.00020**	**0.899**	**0.834**	0.667
Duck	0.00036	0.811	0.053	**0.00032**	0.808	0.165	**0.00032**	**0.819**	**0.329**	0.327
Face	0.00036	0.842	0.052	0.00026	0.869	0.178	**0.00019**	**0.904**	**0.450**	0.360
Cat	0.00034	0.789	0.051	0.00032	**0.805**	0.163	**0.00031**	0.804	**0.386**	0.326
Bag	0.00040	0.792	0.072	0.00026	0.833	0.413	**0.00024**	**0.858**	**0.921**	0.802
Product	0.00045	0.746	0.066	0.00032	0.808	0.280	**0.00023**	**0.853**	**0.588**	0.521

Table 1: *Following [YVG20], we evaluate the sketch-to-sketch similarity (compared to ground truth) by computing the Chamfer distance (lower is better) and F-score with distance 0 (higher is better). We measure the quality of vector graphics representation by the average length of vector strokes (see the 'L' columns; closer to the ground truth is better, and longer is preferred with the same similarity).*

fine details and complex junctions where many lines meet. The method of [SBBB20] produces oversimplified curve geometry and misses fine details. The deep learning-based approach [MSSG*21] suffers from small redundant curve segments since it focuses on image coverage rather than geometry modeling. Bessmeltsev and Solomon's method [BS19] produces curves more accurately but often fails at complex junctions (e.g., Sheriff's left hand in Figure 9, Puppy's eyes and Hippo's claws in Figure 10). Puhachov et al.'s results [PNCB21] achieve visually pleasing effects. However, the lack of detection of Y-junctions leads to incorrect connectivity, such as long redundant curves (e.g., marked regions shown in small circles in Figures 9 and 10) and lack of some connections (e.g., the Hippo's crawl in Figure 10).

5.2. Quantitative Evaluation

To quantitatively evaluate our method, we adopt metrics from a recent benchmark [YVG20]. We compute the metrics between the algorithm's vector output and ground truth vector drawings. Note that the dataset of [YVG20] designed for sketchy inputs is not in our scope, and the tested images frequently used in previous works (e.g., inputs in Figures 9 and 10) have no corresponding ground truth. Therefore we generate several image inputs (e.g., examples shown in Figure 11) from vector drawings with various shapes and

Examples	Input Res.	Bessmeltsev and Solomon 2019 time	Stanko et al. 2020 time	Mo et al. 2021 time	Our time
Elephant	500x753	1670s	1061s	18s	3.1s
Puppy	660x624	1180s	2031s	17s	2.4s
Hippo	700x535	1396s	463s	14s	1.8s
Penguin	500x714	1133s	354s	9s	1.6s
Kitten	500x714	2034s	462s	13s	1.9s
Banana Tree	500x714	606s	1079s	21s	2.3s
Muten	1024^2	2172s	2130s	29s	3.7s
Mouse	1024^2	>40min	1531s	31s	3.9s
Dracolion	1024^2	>40min	>40min	38s	5.1s

Table 2: *Our method is computationally more efficient than the frame field-based approaches [BS19, SBBB20] and the deep learning approach [MSSG*21].*

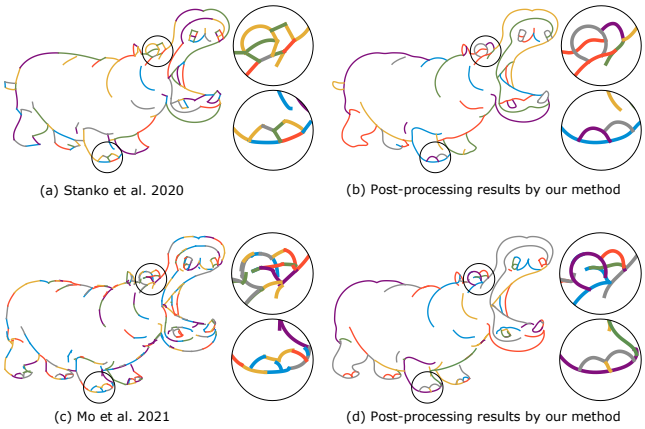

Figure 12: *Using our curve network optimization to post-process the results by Stanko et al. 2020 and Mo et al. 2021, respectively. The top row shows that our algorithm can correct the oversimplified curve geometry of Stanko et al. The bottom row shows that our method can fix the issues of redundant curve segments and undesired line breaks of Mo et al.'s vector output.*

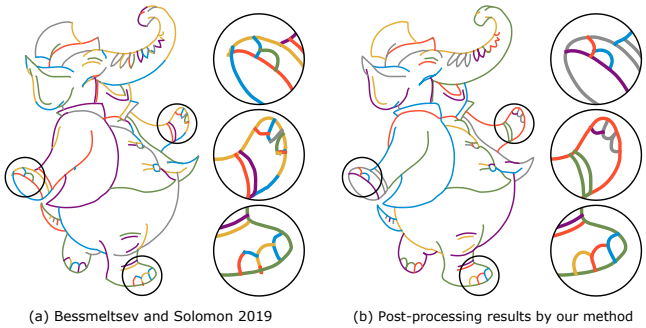

Figure 13: *Using our curve network optimization to post-process the results by Bessmeltseve and Solomon 2019. (a) is their result generated with default parameters. (b) is the result of applying our method to correct the topology artifacts.*

topology connections. To mimic the noisy scanned input images, we render each stroke with varying line width and gray density.

Following [YVG20], we evaluate the sketch-to-sketch similarity between our results and the ground truth by computing Chamfer distance and F-score with distance 0. We also measure the quality of vector graphics representation by the average length of the resulting vector strokes. For comparison, we compute the same metrics on the frame field-based method [BS19] and the deep learning-based method [MSSG*21]. The quantitative evaluation results are shown in Table 1.

The similarity evaluation in Table 1 shows that our results are more precise, since our optimization algorithm always keep the curves to fit the image centerlines. For the vector quality evaluation, our results are much closer to the ground truth and have longer continuous curves (see the visual effects in Figure 11). Our results have a longer curve length than the ground truth because we make connections in all the non-junction areas, while human drawings might make breaks on sharp corners.

5.3. Running Time

We implemented our method in Python. A notebook with Intel(R) Core(TM) i7 2.80GHz and 16GB RAM was used as the testing device. Table 2 presents the running time statistics of our algorithm compared to other approaches. We ran the authors' code of the compared methods [BS19, SBBB20, MSSG*21] on the same testing device. As the table shows, our method is more efficient than the frame field-based methods and the deep learning approach, thanks to the efficient tracing initialization and the fast converged solver for optimization. Note that the deep learning method of Mo et al. [MSSG*21] could be faster on GPUs. It is about half of the running time on CPU in their article, yet slower than ours.

5.4. As Post-processing Step

Our curve network optimization algorithm (Sections 4.2 and 4.3) can also work as a post-processing step for other line drawing vec-

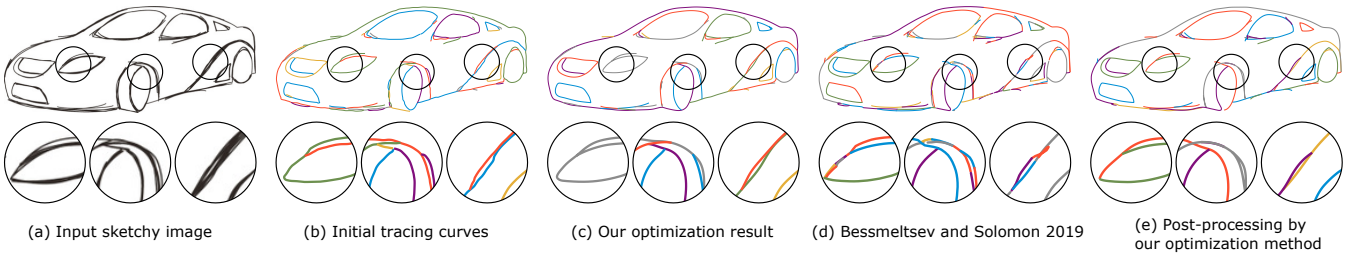

Figure 14: *Results on a sketchy image. (b) and (c) show the results of our initial tracing process and the following optimization algorithm, respectively. In comparison, (d) shows Bessmeltseve and Solomon 2019's result and (e) shows the result of applying our optimization method to (d).*

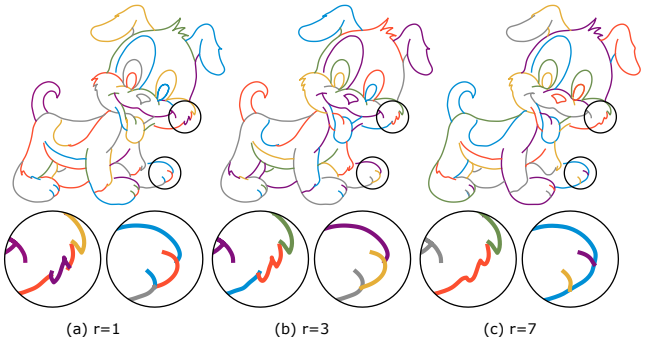

Figure 15: *Effect of changing the parameter r. Increasing r leads to more continuous connectivity, and decreasing r allows capturing more details.*

torization methods. The top row of Figure 12 shows that our technique is able to correct the curve geometry of [SBBB20]. The bottom row shows that our method can fix the issues of redundant curve segments and break lines of [MSSG*21]. Figure 13 shows that our algorithm can correct the topology artifacts of [BS19] at complex junctions.

5.5. Robustness and Paramter Settings

Our method depends on a few key parameters. In our experiments, we use fixed parameter values for the energy weights w_1, w_2 in Equation 2, w_d in Equations 1 and 6, w_t in Equation 6, and the gradient descent step λ in Equation 4. We set one tunable parameter r used for junction clustering in Section 4.3. We selected r within the range $r \in [2,5]$ ($r = 3$ by default). The effect of changing the parameter r is demonstrated in Figure 15.

Figures 12 and 14 show how our optimization framework depends on the intial curve network. Given the initial curves generated by different methods (Figures 12 (a) and (c), Figure 14 (b) and (d)), our optimization method produces similar results (Figures 12 (b) and (d), Figure 14 (c) and (e)). The geometry of our results is not sensitive to the initial curves, while the topology might depend on the initial curve network in some cases such as missing parts.

5.6. Limitations and Future Work

Figure 14 shows our results on rough drawings that contain thick overskeched lines. Our method dose not address the problem of sketchy simplification. However, our optimization algorithm can still improve the initial tracing curves (Figure 14 (b) (c)) and benefit the other vectorization method [BS19] as a post-processing step (Figure 14 (d) (e)).

Although our solution is able to extract correct topology in most cases, our algorithm still fails in some areas. Our method might produce topology that differs from human perception (e.g., Elephant's left front feet in Figure 10). It is mainly because the connectivity is not guaranteed to be correctly estimated (Section 4.3). In future work, an improved connectivity estimation algorithm might be explored to address this issue. As shown in Figure 15, our connectivity estimation depends on a parameter r. A possible future direction is to compute the value of r adaptively rather than select a fixed value for a particular input.

6. Conclusion

We have presented a novel line drawing vectorization framework based on coarse-to-fine curve network optimization. Our optimization algorithm achieves geometry accuracy by keeping curves located at image centerlines. To achieve more continuous connectivity, we perform a global optimization followed by a finer optimization at local junction regions. Since our computation performs on curve points rather than image pixels, our algorithm is efficient. Our system can be immediately useful for downstream applications, such as line art animation and curve-based vector graphics.

References

[BCF*07] BARTOLO A., CAMILLERI K. P., FABRI S. G., BORG J. C., FARRUGIA P. J.: Scribbles to vectors: preparation of scribble drawings for cad interpretation. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling* (2007), pp. 123–130.

[BCY*21] BHUNIA A. K., CHOWDHURY P. N., YANG Y., HOSPEDALES T. M., XIANG T., SONG Y.-Z.: Vectorization and rasterization: Self-supervised learning for sketch and handwriting. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 5668–5677.

[BF12] BAO B., FU H.: Vectorizing line drawings with near-constant line width. *IEEE International Conference on Image Processing (ICIP 2012)* (2012).

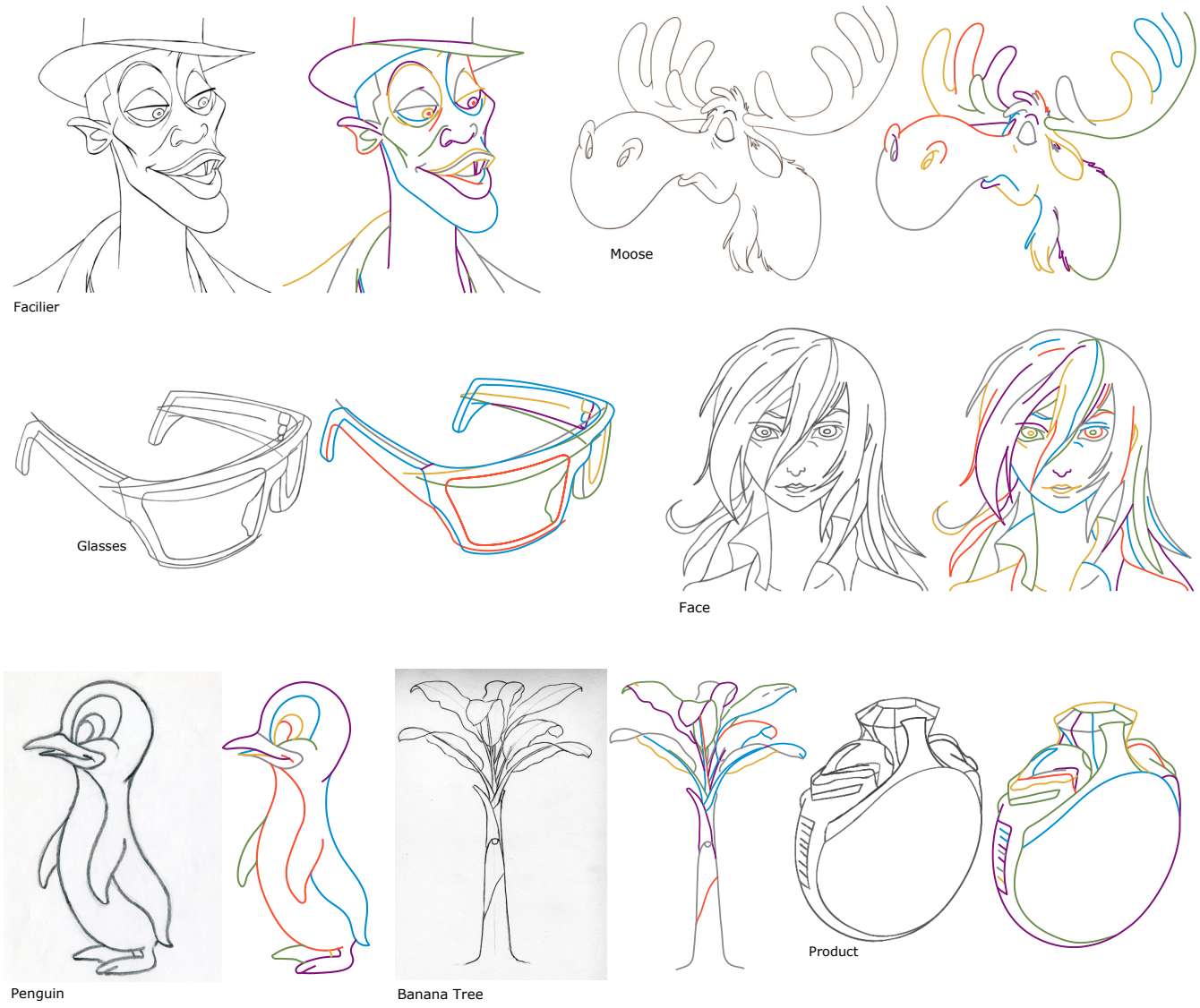

Figure 16: *Gallery of additional results, showing that our method can handle line drawings with varying image quality and shape complexity. Facilier, Moose, Penguin and Banana-Tree are scanned drawings. The others are synthetic images mentioned in Section 5.2.*

[BS19] BESSMELTSEV M., SOLOMON J.: Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics 38*, 1 (2019), 1–12.

[BTS05] BARLA P., THOLLOT J., SILLION F. X.: Geometric clustering for line drawing simplification. In *ACM SIGGRAPH 2005 Sketches on - SIGGRAPH '05* (- 2005), p. nil.

[CLMP15] CHEN J., LEI Q., MIAO Y., PENG Q.: Vectorization of line drawing image based on junction analysis. *Science China Information Sciences 58*, 7 (2015), 1–14.

[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 Workshops* (2017), pp. 2280–2286.

[DYH*21] DAS A., YANG Y., HOSPEDALES T. M., XIANG T., SONG Y.-Z.: Cloud2curve: Generation and vectorization of parametric sketches. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 7084–7093.

[EVA*20] EGIAZARIAN V., VOYNOV O., ARTEMOV A., VOLKHONSKIY D., SAFIN A., TAKTASHEVA M., ZORIN D., BURNAEV E.: Deep vectorization of technical drawings. In *European Conference on Computer Vision* (2020).

[FLB16] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics 35*, 4 (2016), 1–10.

[FSH11] FINCH M., SNYDER J., HOPPE H.: Freeform vector graphics with controlled thin-plate splines. In *Proceedings of the 2011 SIGGRAPH Asia Conference on - SA '11* (- 2011), p. nil.

[FZLM11] FU H., ZHOU S., LIU L., MITRA N. J.: Animated construc-

tion of line drawings. In *Proceedings of the 2011 SIGGRAPH Asia Conference on - SA '11* (- 2011), p. nil.

[GZH*19] GUO Y., ZHANG Z., HAN C., HU W., LI C., WONG T.-T.: Deep line drawing vectorization via line subdivision and topology reconstruction. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 81–90.

[JK05] JON KLEINBERG EVA TARDOS É. T.: *Algorithm Design*. Pearson, 2005.

[KWÖG18] KIM B., WANG O., ÖZTIRELI A. C., GROSS M. H.: Semantic segmentation for line drawing vectorization using neural networks. *Computer Graphics Forum 37* (2018).

[LRS18] LIU C., ROSALES E., SHEFFER A.: Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings. *ACM Transactions on Graphics 37*, 4 (2018), 1–15.

[LWH15] LIU X., WONG T.-T., HENG P.-A.: Closure-aware sketch simplification. *ACM Transactions on Graphics 34*, 6 (2015), 1–10.

[MKDM22] MANDA B., KENDRE P., DEY S., MUTHUGANAPATHY R.: Sketchcleannet - a deep learning approach to the enhancement and correction of query sketches for a 3d cad model retrieval system. *Computers & Graphics 107* (2022), 73–83.

[MSSG*21] MO H., SIMO-SERRA E., GAO C., ZOU C., WANG R.: General virtual sketching framework for vector line art. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021) 40*, 4 (2021), 51:1–51:14.

[NHS*13] NORIS G., HORNUNG A., SUMNER R. W., SIMMONS M., GROSS M.: Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics 32*, 1 (2013), 1–11.

[NS19] NAJGEBAUER P., SCHERER R.: Inertia-based fast vectorization of line drawings. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 203–213.

[OBW*08] ORZAN A., BOUSSEAU A., WINNEMÖLLER H., BARLA P., THOLLOT J., SALESIN D.: Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (TOG) 27*, 3 (2008), 1–8.

[PNCB21] PUHACHOV I., NEVEU W., CHIEN E., BESSMELTSEV M.: Keypoint-driven line drawing vectorization via polyvector flow. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 40*, 6 (dec 2021).

[PPM18] PARAKKAT A. D., PUNDARIKAKSHA U. B., MUTHUGANAPATHY R.: A delaunay triangulation based approach for cleaning rough sketches. *Computers & Graphics 74*, nil (2018), 171–181.

[PvMLV*21] PAGUREK VAN MOSSEL D., LIU C., VINING N., BESSMELTSEV M., SHEFFER A.: Strokestrip: Joint parameterization and fitting of stroke clusters. *ACM Transactions on Graphics 40*, 4 (2021).

[SBBB20] STANKO T., BESSMELTSEV M., BOMMES D., BOUSSEAU A.: Integer-grid sketch simplification and vectorization. *Computer Graphics Forum (Proc. SGP) 39*, 5 (7 2020).

[Sel03] SELINGER P.: Potrace: a polygon-based tracing algorithm. *Potrace (online), http://potrace. sourceforge. net/potrace. pdf (2009-07-01)* (2003).

[SSC*00] SONG J., SU F., CHEN J., TAI C., CAI S.: Line net global vectorization: an algorithm and its performance evaluation. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* (2000), vol. 1, IEEE, pp. 383–388.

[SSII18] SIMO-SERRA E., IIZUKA S., ISHIKAWA H.: Mastering sketching: Adversarial augmentation for structured prediction. *ACM Transactions on Graphics 37*, 1 (2018), 1–13.

[SSISI16] SIMO-SERRA E., IIZUKA S., SASAKI K., ISHIKAWA H.: Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics 35*, 4 (2016), 1–11.

[SSTC02] SONG J., SU F., TAI C., CAI S.: An object-oriented progressive-simplification-based vectorization system for engineering drawings: Model, algorithm, and performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 8 (2002), 1048–1060.

[XXM*21] XU X., XIE M., MIAO P., QU W., XIAO W., ZHANG H., LIU X., WONG T.: Perceptual-aware sketch simplification based on integrated vgg layers. *IEEE Transactions on Visualization and Computer Graphics 27* (2021), 178–189.

[YVG20] YAN C., VANDERHAEGHE D., GINGOLD Y.: A benchmark for rough sketch cleanup. *ACM Transactions on Graphics (TOG) 39*, 6 (2020).

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

[ZY01] ZOU J., YAN H.: Cartoon image vectorization based on shape subdivision. In *Computer Graphics International 2001. Proceedings* (2001), IEEE, pp. 225–231.