Vision, Modeling, and Visualization (2026)
T. Leimkühler, T. Weinkauf and M. Campen (Eds.)

Stroke Vectorization via Involution Prediction: Supplementary Material

Philipp Gerstner, Tanguy Magne and Olga Sorkine-Hornung

ETH Zurich, Zurich, Switzerland

Appendix A: Background - Gumbel-Sinkhorn

This section provides background on the Gumbel-Sinkhorn operator used to train our intersection model.

Doubly stochastic matrix. A doubly stochastic matrix is a square matrix with real, non-negative elements, such that the rows and the columns each sum to 1.

Sinkhorn operator. Sinkhorn's Theorem [SK67] states that for a matrix $A \in \mathbb{R}^{n\times n}$ with $A_{i,j} \geq 0$ satisfying certain technical conditions, there exist positive diagonal matrices D_1 and D_2 such that $S(A) = D_1 A D_2$ is doubly stochastic. Moreover, $S(A)$ is the unique doubly stochastic matrix of the form $D_1 A D_2$.

The matrix $S(A)$ can be approximated numerically via iterative row and column normalization (Sinkhorn iterations), which is differentiable. This converges to $S(A)$ for most non-negative matrices [AZ11]. Let $S^K(A)$ denote the matrix after K iterations. For numerical stability, iterations are performed in log-space [SGG13]. Setting $B = \log(A)$ with $\log(0) = -\infty$ and $\exp(-\infty) = 0$, we initialize $u_i = v_j = 0$ for all i, j, and repeat for $k = 1, \ldots, K$:

$$u_i \leftarrow -\log \sum_j \exp\big(B_{ij} + v_j\big), \tag{1}$$

$$v_j \leftarrow -\log \sum_i \exp\big(B_{ij} + u_i\big). \tag{2}$$

Denoting the complete log-space iteration process as Slog^K_{ij}, the resulting matrix is:

$$S^K_{ij}(A) = \mathrm{Slog}^K_{ij}(\log(A)) = \mathrm{Slog}^K_{ij}(B) = \exp\big(u_i + B_{ij} + v_j\big). \tag{3}$$

Proposition 1 If $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix admitting a Sinkhorn decomposition, then $S(A)$ is also symmetric, i.e., the Sinkhorn operator preserves symmetry.

Proof By Sinkhorn's Theorem, there exist positive diagonal matrices D_1 and D_2 such that $S(A) = D_1 A D_2$ is doubly stochastic. By symmetry and diagonality, we have for the transpose:

$$S(A)^\mathsf{T} = (D_1 A D_2)^\mathsf{T} = D_2^\mathsf{T} A^\mathsf{T} D_1^\mathsf{T} = D_2 A D_1.$$

Both $S(A)$ and $S(A)^\mathsf{T}$ are doubly stochastic, so both $S(A) = D_1 A D_2$ and $S(A)^\mathsf{T} = D_2 A D_1$ are doubly stochastic scalings of A. By uniqueness of the Sinkhorn decomposition, $S(A) = S(A)^\mathsf{T}$. □

Note that finite iterations $S^K(A)$ may not preserve symmetry, but they converge to a symmetric matrix as $K \to \infty$ when starting with a symmetric input.

Optimal permutations. Any doubly stochastic matrix is a convex combination of permutation matrices (the Birkhoff polytope [MOA11]). The Sinkhorn operator thus maps its input to a point on this polytope. For a matrix $A \in \mathbb{R}^{n\times n}$ and temperature $\tau > 0$, Mena et al. [MBLS18] showed that $S_M(A) = S(\exp(A/\tau))$ approximates the permutation matrix P maximizing $\mathrm{trace}(P^\mathsf{T} A)$. The matrix A acts as a log-likelihood table indicating which row/column pairs are likely to be 1 in the permutation. The temperature τ controls proximity to a true permutation matrix, with smaller values increasing convergence but also the risk of vanishing gradients [MBLS18]. To train a model that predicts permutations, we can thus predict A, compute $S_M(A)$ and backpropagate.

Gumbel-Sinkhorn. Mena et al. [MBLS18] propose the Gumbel-Sinkhorn operator S_G, which combines the Sinkhorn operator with Gumbel noise for differentiable sampling from a distribution over permutations:

$$S_G(A) = S\left(\exp\left(\frac{A+\epsilon}{\tau}\right)\right), \tag{4}$$

where ϵ has i.i.d. entries $\epsilon_{i,j} \sim G(0,1)$ (Gumbel distribution). Gumbel noise can be obtained from a uniform distribution as follows [JGP17]:

$$\epsilon_{i,j} = -\log(-\log(\mathrm{u})), \quad \mathrm{u} \sim \mathrm{Uniform}(0,1).$$

Intersection model. As mentioned in the main paper, the forward pass of the intersection model yields a symmetric matrix C, scoring how strongly each half-branch relates to the others. To make this matrix comparable to the ground-truth involution matrix, we apply the Gumbel-Sinkhorn operator [MBLS18]. Since the Sinkhorn operator preserves symmetry, we make the Gumbel noise matrix ϵ symmetric, so that the Gumbel-Sinkhorn operator preserves symmetry as well. By Eqs. 4 and 3, the normalization step becomes:

$$\hat{S} = \mathrm{Slog}^K_{ij}\left(\frac{C+\boldsymbol{g}}{\tau}\right), \quad \begin{cases} \boldsymbol{g}_{i,j} \sim G(0,1) & j \geq i \\ \boldsymbol{g}_{i,j} = \boldsymbol{g}_{j,i} & j < i \end{cases} \tag{5}$$

The resulting matrix $\hat{S}$ is a symmetric doubly stochastic matrix. We use $K = 20$ iterations and $\tau = 1$.

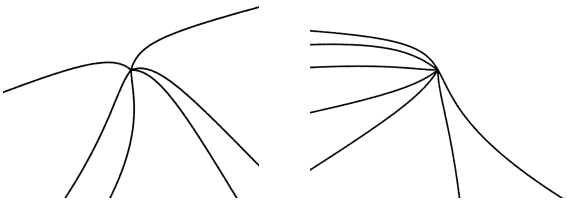

Figure 1: *Sample from our synthetic dataset. Left: branches intersect in a star-like shape. Right: branches intersect in a "half-star".*

Appendix B: Method

Centerline model

Synthetic intersection dataset. To train our centerline model, in addition to the vector inputs taken from existing datasets, we also generate a synthetic dataset of star-like shapes. To that end, a random number k of branches is selected (between 4 and 9). The full-circle interval $[0, 2\pi]$ is then randomly partitioned into k sub-intervals to determine the angular positions of the branches, with some branches kept tangent, i.e., $\pi/2$ apart. Each branch is drawn using a line segment or a Bézier curve, in such a way that no additional intersections occur (see Fig. 1, left). Optionally, only the half-circle interval is partitioned, yielding a "half-star" in which $k-1$ branches meet along the first line (see Fig. 1, right).

Graph extraction

Pseudocode. Algorithm 1 summarizes the whole polyline graph extraction stage of our method (Section 3.2 of the main paper), and Algorithm 2 details the cluster resolution part.

Boundary polygon computation. To compute the boundary polygon used for triangle cluster resolution, we use an algorithm similar to convex hull computation. After picking an extremum as the starting point, we repeatedly choose the most outward point among all unvisited cluster neighbors (rather than among all unvisited points), until we return to the starting point. The algorithm is correct because G has no intersecting edge segments, all node positions are unique and, if (u,v) and (u,w) are connected edges, they are not collinear.

Medial axis computation. When resolving triangle clusters into clean junctions, if the general-case solution fails, we fall back to replacing the cluster by the medial axis of the boundary polygon. An approximate medial axis is obtained via the Voronoi method of Brandt and Algazi [BA92]. We first densify the polygon boundary so that adjacent points are close (maximum distance 0.3 pixels), then compute the Voronoi diagram. Edges lying outside or partly outside the polygon are discarded, while those inside form a medial axis that naturally provides the junction structure. Branches not leading to boundary nodes are pruned, and the remaining Voronoi subgraph is inserted into G, replacing the original cluster interior.

Parameters. The thresholds used in the polyline graph extraction stage fall into two groups. The intensity thresholds (T_1, T_2, and the brightness thresholds used for the dilation and for artifact removal)

Algorithm 1: Polyline graph extraction

input : grayscale centerline image $\hat{I}_C$
output : planar graph G and its branches
$B \leftarrow$ threshold $\hat{I}_C$ with T_1, drop components below T_2, subtract adaptive threshold
$B \leftarrow$ fill holes and remove spikes in B
$S \leftarrow$ skeletonize B
foreach *pixel p of S with more than two neighbors* **do**
  add in B the pixels of a 5×5 mask around p whose intensity exceeds a threshold
end
$B \leftarrow$ fill holes and remove spikes in B
$G \leftarrow$ 8-adjacency graph of B, with a node inserted at every crossing of two diagonal edges
$\mathcal{T} \leftarrow$ clusters of edge-sharing triangles of G
foreach $T \in \mathcal{T}$ **do**
  compute the boundary nodes, boundary polygon and outbound vectors of T
  $G \leftarrow$ `ResolveCluster(G,T) // Algorithm 2`
end
remove spurious short paths and improve sharp tips in G
extract the branches of G as its maximal runs of degree-2 nodes, and smooth each of them
return G and its branches

are applied to the centerline image, which is near-binary thanks to the second term of the centerline loss (Section 3.1 of the main paper). They only have to separate strokes from faint residual response, and have been tuned to the output of the trained centerline model. The geometric thresholds (the 0.25 px candidate grid, the 0.3 px boundary densification and the 6 px spurious-path length) are expressed relative to the 1 px stroke width the centerline model produces, and are thus independent of the input resolution. All values were fixed once and used unchanged for every result reported in this paper.

Appendix C: Quantitative results

Metrics. We give here additional details about the metrics used for the quantitative comparison of vectorization methods. Before computing the metrics, the ground truth and output vector drawings are resized to a 512×512 resolution and normalized into dense polylines: Bézier curves are discretized using 80 samples, and each resulting line segment is resampled with k_s points, where k_s is chosen per segment so that the maximum distance between two consecutive points is 0.1 pixels. We denote by $P = \{p_1, \dots, p_m\}$ and $Q = \{q_1, \dots, q_l\}$ the point sets obtained by concatenating all resampled polyline points of the prediction and the ground truth, respectively.

With this notation, the Chamfer distance is defined as

$$\frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \|p - q\| \; + \; \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \|q - p\|.$$

Let $L(\cdot)$ denote the total arc length of a set of polylines, i.e., the sum of Euclidean distances between consecutive points across all strokes. The total stroke length difference is then $|L(P) - L(Q)|$.

Algorithm 2: RESOLVECLUSTER: cluster resolution

input : graph G, cluster T of degree d, boundary nodes $\mathcal{B}$, boundary polygon P, outbound vectors w
output : G with the interior of T replaced

```
if d = 0 then
    remove T from G                          // isolated noise
else if d = 1 then
    keep the shortest path from the boundary node to the farthest node of T
else if d = 2, with B = {u, v} then
    if ⟨w[u], w[v]⟩ > 0 then
        keep the longest arc of P between u and v   // bend
    else
        keep the shortest interior path between u and v
        // nearly straight line
    end
else
    C ← grid of candidate points inside P at 0.25 px spacing
    for c ∈ C, let vis(c) be the set of b ∈ B such that the segment cb stays in the foreground
    if some c ∈ C has vis(c) = B then
        replace the interior of T by the such c with maximal alignment
    else if some c1, c2 ∈ C have vis(c1) ∪ vis(c2) = B then
        replace the interior of T by the such pair with maximal alignment
    else
        replace the interior of T by the pruned medial axis of P   // always defined
    end
end
return G
```

The *Stroke Density Ratio* (r_{SD}) introduced in the main paper can be written as

$$r_{\text{SD}} = \frac{|\text{strokes}_P| \,/\, L(P)}{|\text{strokes}_Q| \,/\, L(Q)},$$

where $|\text{strokes}_X|$ is the number of strokes. Note that some lines are over-segmented in the ground truth, i.e., several lines may visually represent a single stroke once rasterized. The density $|\text{strokes}_Q|/L(Q)$ may therefore be higher than expected, biasing r_{SD} toward lower values, but this bias is the same for all methods.

Finally, for the intersections metric, which measures the ratio of correctly found intersections in the predicted vector representation, we did not manage to obtain ground truth intersections for six samples, which are thus excluded when computing this metric.

Evaluation results. We present detailed results for the evaluation described in the main paper. Similarly to Yan et al. [YLA*24], we run the various methods on inputs rasterized at three resolutions: 512×512, 768×768 and 1024×1024. To visualize each metric's distribution, the results are presented as box plots (Fig. 2, Fig. 3 and Fig. 4) except for the intersection metric (Table 1). The results exhibit trends similar to those in the main paper: comparable Chamfer distance and length difference to Yan et al. [YLA*24], but better stroke density ratio and intersection detection. Note that the lower intersection detection rate at lower resolution is mostly due to intersections not being visible at that resolution.

Table 1: *Correct intersection rate for the tested methods, at all 3 resolutions.*

Method	512	768	1024
Bessmeltsev and Solomon. [BS19]	15.8%	19.8%	22.1%
Puhachov et al. [PNCB21]	**43.7%**	53.1%	58.2%
Mo et al. [MSSG*21]	7.7%	8.6%	8.9%
Yan et al. [YLA*24]	22.2%	43.0%	50.5%
Ours	42.5%	**61.2%**	**73.7%**

Runtime. We provide more details about the runtime of our method on the benchmark at all three resolutions in Table 2. We report the average, median, minimum, maximum, 90th percentile and 95th percentile times, in seconds. These results confirm that our method is faster on average than other baselines. The only exceptions are extreme samples with many strokes and intersections, which increase the runtime of the graph extraction step, as mentioned in Section 4.3 of the main paper

Appendix D: Qualitative results

We present additional visual comparisons against the methods of Bessmeltsev et al. [BS19] and Yan et al. [YLA*24] in Fig. 5.

Appendix E: Single-line drawings evaluation

We also evaluate our method on the single-line drawing benchmark of Magne and Sorkine-Hornung [MSH25], which contains 56 single-line drawings in vector format. Following their protocol, we measure visual similarity between the input and output by rasterizing both with the same resolution and stroke width, and computing the PSNR (peak signal-to-noise ratio) and SSIM (structural similarity index measure). We also report the relative total stroke length difference and the number of strokes in the vectorized output, which should be 1 for single-line drawings. We compare our method against the same baselines [BS19, PNCB21, MSSG*21, YLA*24] as well as Magne and Sorkine-Hornung. We present the results in Table 3 and visual comparisons in Fig. 6. Our method stands out by reaching the second best scores on all visual metrics and on stroke count, outperformed only by the highly specialized method of Magne and Sorkine-Hornung [MSH25]. It produces far fewer strokes than every other general line drawing vectorization method, with these extra strokes arising where a path stops at an intersection instead of continuing through it (see Fig. 6), which is arguably easier to edit than misrouted branches. Fig. 6 also shows that, while our method generally falls a bit behind that of Magne and Sorkine-Hornung [MSH25] (first two rows), it achieves similar or better results on some samples (last two rows).

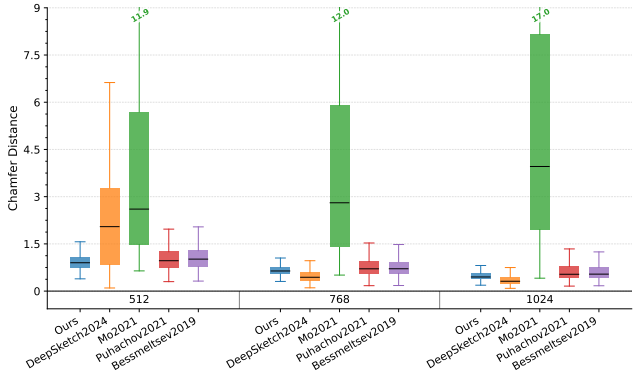

Figure 2: *Distribution of the Chamfer distance between the predicted and the ground truth drawings, for each method and at three input resolutions (*512*,* 768 *and* 1024*). Lower values indicate better geometric accuracy.*

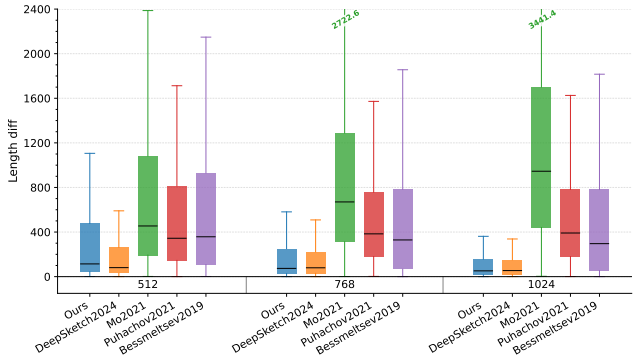

Figure 3: *Distribution of the total stroke length difference between the predicted and the ground truth drawings, for each method and at three input resolutions (*512*,* 768 *and* 1024*). Lower values indicate a closer match in total drawn length.*

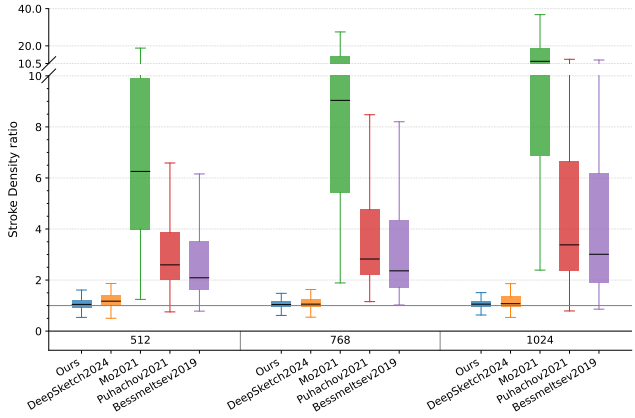

Figure 4: *Distribution of the stroke density ratio r_{SD} for each method and at three input resolutions (*512*,* 768 *and* 1024*). A value close to* 1 *is best, while values above* 1 *indicate over-segmentation.*

Table 2: *Runtime (in second) of our method on the evaluation dataset.*

Resolution	Min	Median	Mean	p90	p95	Max
512	0.2	0.8	2.4 ± 4.7	4.8	9.9	46.8
768	0.5	1.4	3.6 ± 7.1	6.7	14.3	71.6
1024	0.6	1.9	5.0 ± 10.5	8.6	21.4	87.8

References

[AZ11] ADAMS R. P., ZEMEL R. S.: Ranking via sinkhorn propagation, 2011. URL: https://arxiv.org/abs/1106.1925, arXiv:1106.1925. 1

[BA92] BRANDT J. W., ALGAZI V.: Continuous skeleton computation by voronoi diagram. *CVGIP: Image Understanding 55*, 3 (1992), 329–338. doi:10.1016/1049-9660(92)90030-7. 2

[BS19] BESSMELTSEV M., SOLOMON J.: Vectorization of line drawings via polyvector fields. *ACM Trans. Graph. 38*, 1 (Jan. 2019). doi:10.1145/3202661. 3, 5, 6

[GHB*23] GUȚAN O., HEGDE S., BERUMEN E. J., BESSMELTSEV M., CHIEN E.: Singularity-free frame fields for line drawing vectorization. *Computer Graphics Forum 42*, 5 (2023). doi:10.1111/cgf.14901. 6

[JGP17] JANG E., GU S., POOLE B.: Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations* (2017). URL: https://openreview.net/forum?id=rkE3y85ee. 1

[MBLS18] MENA G., BELANGER D., LINDERMAN S., SNOEK J.: Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations* (2018). URL: https://openreview.net/forum?id=Byt3oJ-0W. 1

[MOA11] MARSHALL A. W., OLKIN I., ARNOLD B. C.: *Doubly Stochastic Matrices*. Springer New York, New York, NY, 2011, pp. 29–77. doi:10.1007/978-0-387-68276-1_2. 1

[MSH25] MAGNE T., SORKINE-HORNUNG O.: Single-line drawing vectorization. *Computer Graphics Forum 44*, 7 (2025). doi:10.1111/cgf.70228. 3, 6

[MSSG*21] MO H., SIMO-SERRA E., GAO C., ZOU C., WANG R.: General virtual sketching framework for vector line art. *ACM Trans. Graph. 40*, 4 (July 2021). doi:10.1145/3450626.3459833. 3, 6

[PNCB21] PUHACHOV I., NEVEU W., CHIEN E., BESSMELTSEV M.: Keypoint-driven line drawing vectorization via polyvector flow. *ACM Trans. Graph. 40*, 6 (Dec. 2021). doi:10.1145/3478513.3480529. 3, 6

[SGG13] SHARIFY M., GAUBERT S., GRIGORI L.: Solution of the optimal assignment problem by diagonal scaling algorithms, 2013. URL: https://arxiv.org/abs/1104.3830, arXiv:1104.3830. 1

[SK67] SINKHORN R., KNOPP P.: Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics 21*, 2 (May 1967), 343–348. doi:10.2140/pjm.1967.21.343. 1

[YLA*24] YAN C., LI Y., ANEJA D., FISHER M., SIMO-SERRA E., GINGOLD Y.: Deep sketch vectorization via implicit surface extraction. *ACM Trans. Graph. 43*, 4 (July 2024). doi:10.1145/3658197. 3, 5, 6

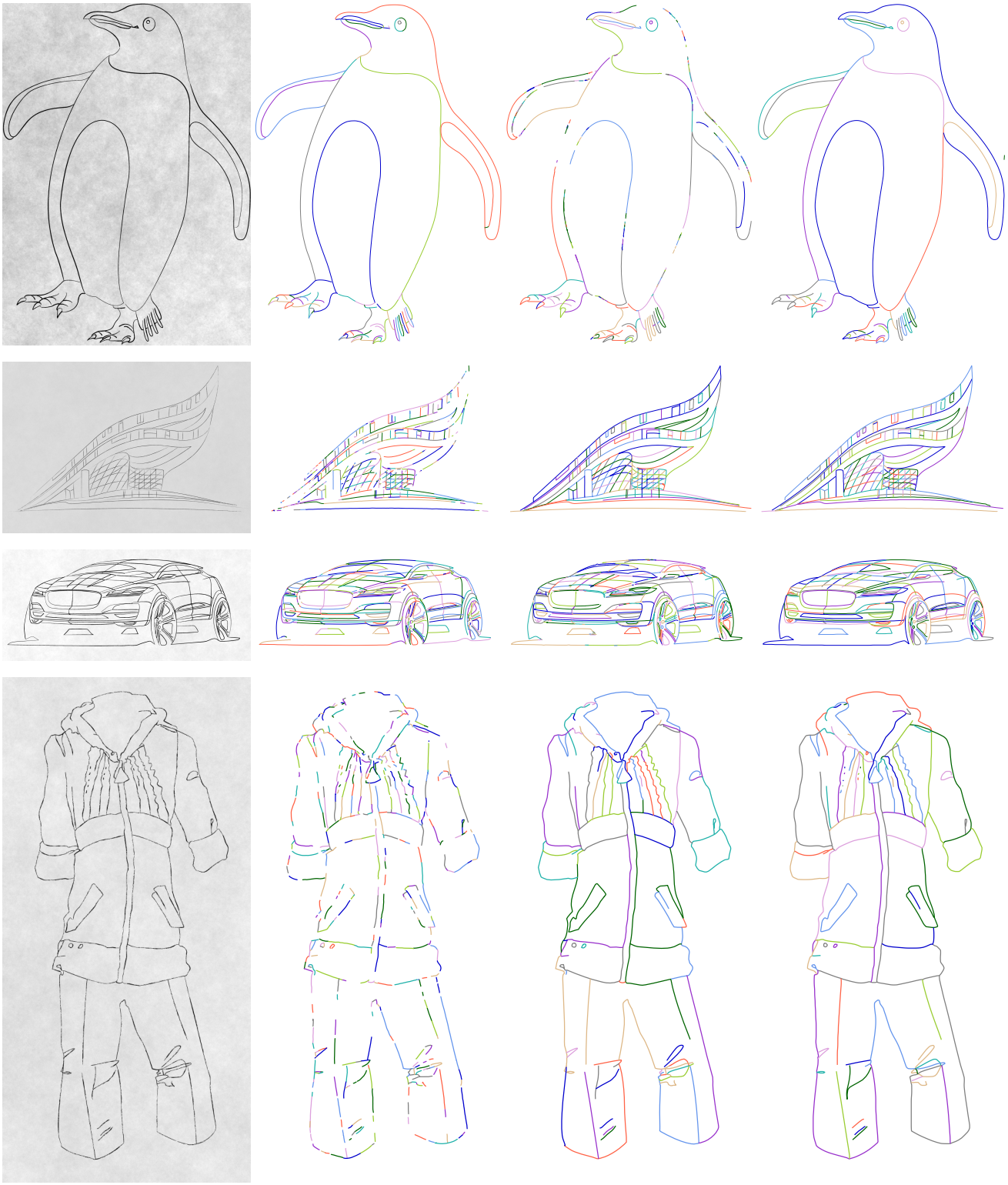

Figure 5: *Qualitative comparison of various vectorization methods. Each stroke is shown in a different color.*

Table 3: *Quantitative comparison of line drawing vectorization performance on the single-line drawing benchmark of Magne and Sorkine-Hornung [MSH25].* PSNR *(peak signal-to-noise ratio) and* SSIM *(structural similarity index measure) capture visual fidelity. The vector output quality is assessed through the* Length ratio*, which reflects how much the total stroke length of the output deviates from the ground truth, and the stroke count, for which the target value is 1.*

	Image		Stroke	
Method	SSIM ↑	PSNR ↑	Length ratio ↓	Count ↓
Bessmeltsev and Solomon [BS19]	0.986	26.6	0.85%	30.9
Puhachov et al. [PNCB21]	0.985	26.2	7.89%	43.0
Mo et al. [MSSG*21]	0.975	23.3	12.78%	55.7
Guțan et al. [GHB*23]	0.984	25.5	0.93%	25.3
Yan et al. [YLA*24]	0.983	26.3	1.04%	10.5
Magne and Sorkine-Hornung [MSH25]	0.987	27.8	0.99%	1.2
Ours	0.986	26.9	7.39%	6.6

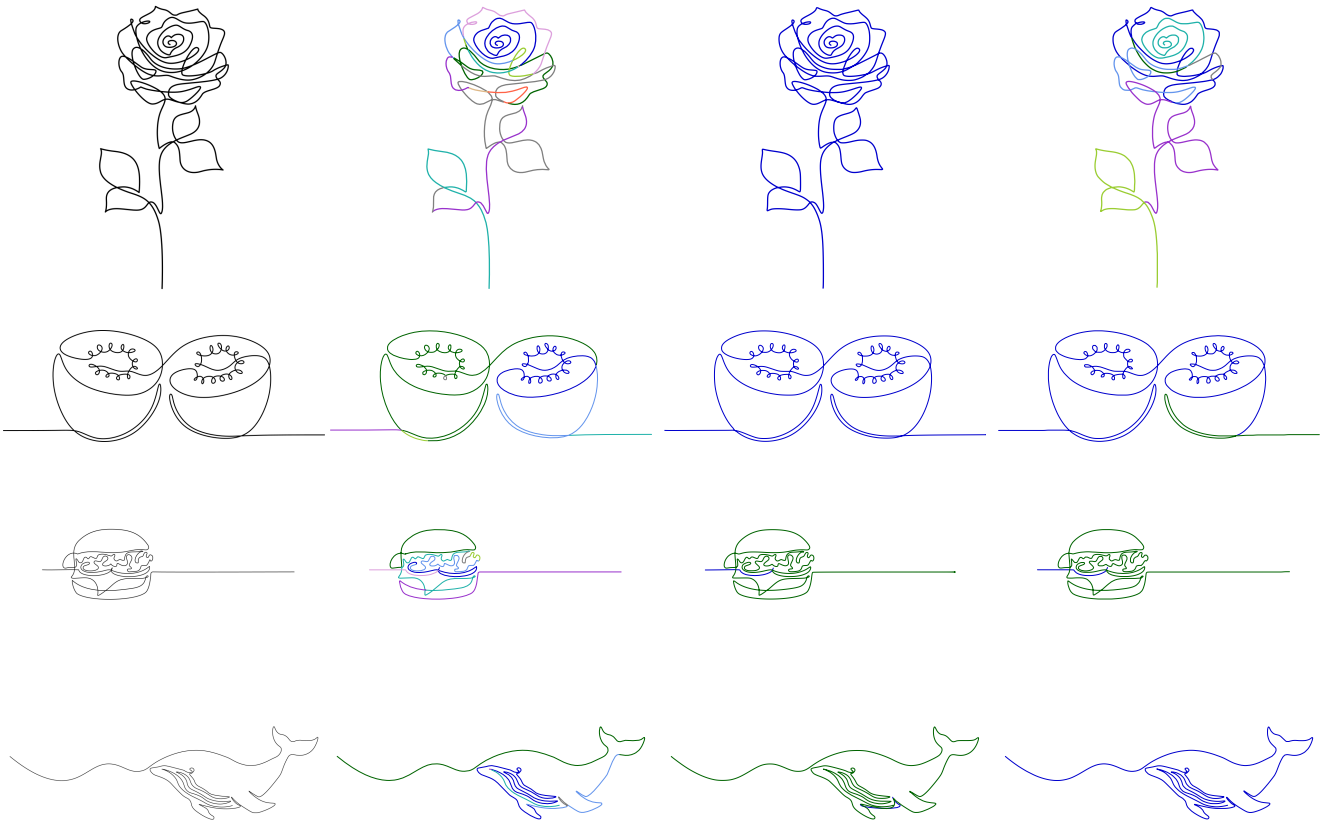

Figure 6: *Qualitative comparison of various vectorization methods on a set of single-line drawings. Each stroke is shown in a different color.*