2D Gaussian Splatting for Bézier Spline Line Art Vectorization (Supplementary Document)

TIANHAO CHEN, DisneyResearch|Studios, Switzerland and ETH Zürich, Switzerland
CLARA FERNANDEZ, DisneyResearch|Studios, Switzerland
MARTEINN OSKARSSON, Walt Disney Animation Studios, Canada
CHUCK TAPPAN, Walt Disney Animation Studios, United States of America
OLGA SORKINE-HORNUNG, ETH Zürich, Switzerland
MARKUS GROSS, DisneyResearch|Studios, Switzerland and ETH Zürich, Switzerland
CHRISTOPHER SCHROERS, DisneyResearch|Studios, Switzerland
ABDELAZIZ DJELOUAH, DisneyResearch|Studios, Switzerland

Overview

In this supplementary document we present material that would otherwise crowd the main text: extra implementation and metric definitions, *additional* qualitative figures, and *full* quantitative tables over multiple input resolutions on the same Deep Sketch benchmark [Yan et al. 2024] and metrics reported in the main Experiments section (the main paper focuses on representative qualitative figures and the 512 px tables there).

A Additional Method and Implementation Details

Loss Details. The self-intersection loss $\mathcal{L}_{\text{int}}(\mathcal{P})$, penalizes the presence of self-intersections for each Bézier segment [Hirschorn et al. 2024].

$$\begin{aligned}&\mathcal{L}_{\text{intersect}}(\mathcal{P})\\&=\frac{1}{n}\sum_{k=0}^{n-1}\text{AND}(\text{XOR}(O(\mathbf{P}_{3k},\mathbf{P}_{3k+1},\mathbf{P}_{3k+2}),O(\mathbf{P}_{3k},\mathbf{P}_{3k+1},\mathbf{P}_{3k+3})),\\&\qquad\text{XOR}(O(\mathbf{P}_{3k+2},\mathbf{P}_{3k+3},\mathbf{P}_{3k}),O(\mathbf{P}_{3k+2},\mathbf{P}_{3k+3},\mathbf{P}_{3k+1}))),\end{aligned}\tag{1}$$

where $O(\mathbf{A},\mathbf{B},\mathbf{C})$ computes the orientation of points $\mathbf{A},\mathbf{B},\mathbf{C}$, defined as

$$O(\mathbf{A},\mathbf{B},\mathbf{C})=\text{sigmoid}\left((\mathbf{B}_y-\mathbf{A}_y)(\mathbf{C}_x-\mathbf{A}_x)-(\mathbf{B}_x-\mathbf{A}_x)(\mathbf{C}_y-\mathbf{A}_y)\right).\tag{2}$$

Implementation Details. For solid line art, we optimize for 200 iterations using the reconstruction loss only, with $\lambda_{\text{LPIPS}}=2\times10^{-3}$. For textured line art, we optimize for 1,000 iterations using both the reconstruction loss and the geometric loss, with $\lambda_{\text{LPIPS}}=2\times10^{-3}$, $\lambda_{\text{cont}}=0.1$, $\lambda_{\text{int}}=0.1$, and $\lambda_{\text{curv}}=1\times10^{-4}$. The maximum curvature threshold is set to $\kappa_{\max}=1\times10^{4}$.

During optimization, stroke simplification and densification are applied every 100 iterations with thresholds $o_{\min}=0.01$, $\delta_{\max}=1\times10^{-5}$, and $d_{\min}=5$ pixels.

Authors' Contact Information: DisneyResearch|Studios, Stampfenbachstrasse 48, Zürich, Switzerland; ETH Zürich, Rämistrasse 101, Zürich, Switzerland. Corresponding authors: Tianhao Chen, chentia@student.ethz.ch; Clara Fernandez, clara.fernandez@disneyresearch.com; Abdelaziz Djelouah, aziz.djelouah@disneyresearch.com;

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
SIGGRAPH Conference Papers '26, Los Angeles, CA, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811211

Chamfer Distance. Given two sets of points $\mathcal{P}$ and $\hat{\mathcal{P}}$ sampled from the vector ground truth and the vectorization result, respectively, we compute the Chamfer Distance between

$$d_{\text{Chamfer}}=\frac{1}{2|\mathcal{P}|}\sum_{\mathbf{p}\in\mathcal{P}}\min_{\hat{\mathbf{p}}\in\hat{\mathcal{P}}}\|\mathbf{p}-\hat{\mathbf{p}}\|_2+\frac{1}{2|\hat{\mathcal{P}}|}\sum_{\hat{\mathbf{p}}\in\hat{\mathcal{P}}}\min_{\mathbf{p}\in\mathcal{P}}\|\hat{\mathbf{p}}-\mathbf{p}\|_2.\tag{3}$$

Here we sample 200 points uniformly along each stroke. We also compute the relative stroke length difference between the ground truth and the result, which is the ratio between the absolute stroke length difference and the ground truth stroke length.

B Additional Results

The following pages extend the qualitative and quantitative results shown in the main paper. We first provide additional side-by-side comparisons on the clean and textured Deep Sketch-style benchmarks (Figures 1 and 2), followed by extended brush-stroke texture evaluations (Figures 3 and 4), comparisons against DiffVG as differentiable renderer and extra video results on the dataset of Mo et al. [2024] (Figure 5). The main paper shows representative comparison figures and 512 px benchmark tables; here we include supplementary examples and full per-resolution tables (Tables 1 and 2).

Full benchmark tables (clean and textured). We evaluate on the benchmark of Yan et al. [2024], rasterizing vector ground truth at several resolutions: 256, 384, 512, 768, or 1024 pixels (matching the row groups in the tables), using the same clean and textured protocols as in the main paper. For each method and resolution we report: *success rate*, the fraction of benchmark scenes for which the method's pipeline completed without numerical failure and produced a valid output for scoring; Chamfer distance and stroke length difference (see Section A); mean stroke count; and reconstruction quality via LPIPS, PSNR, and SSIM between the rendered vectorization and the raster input.

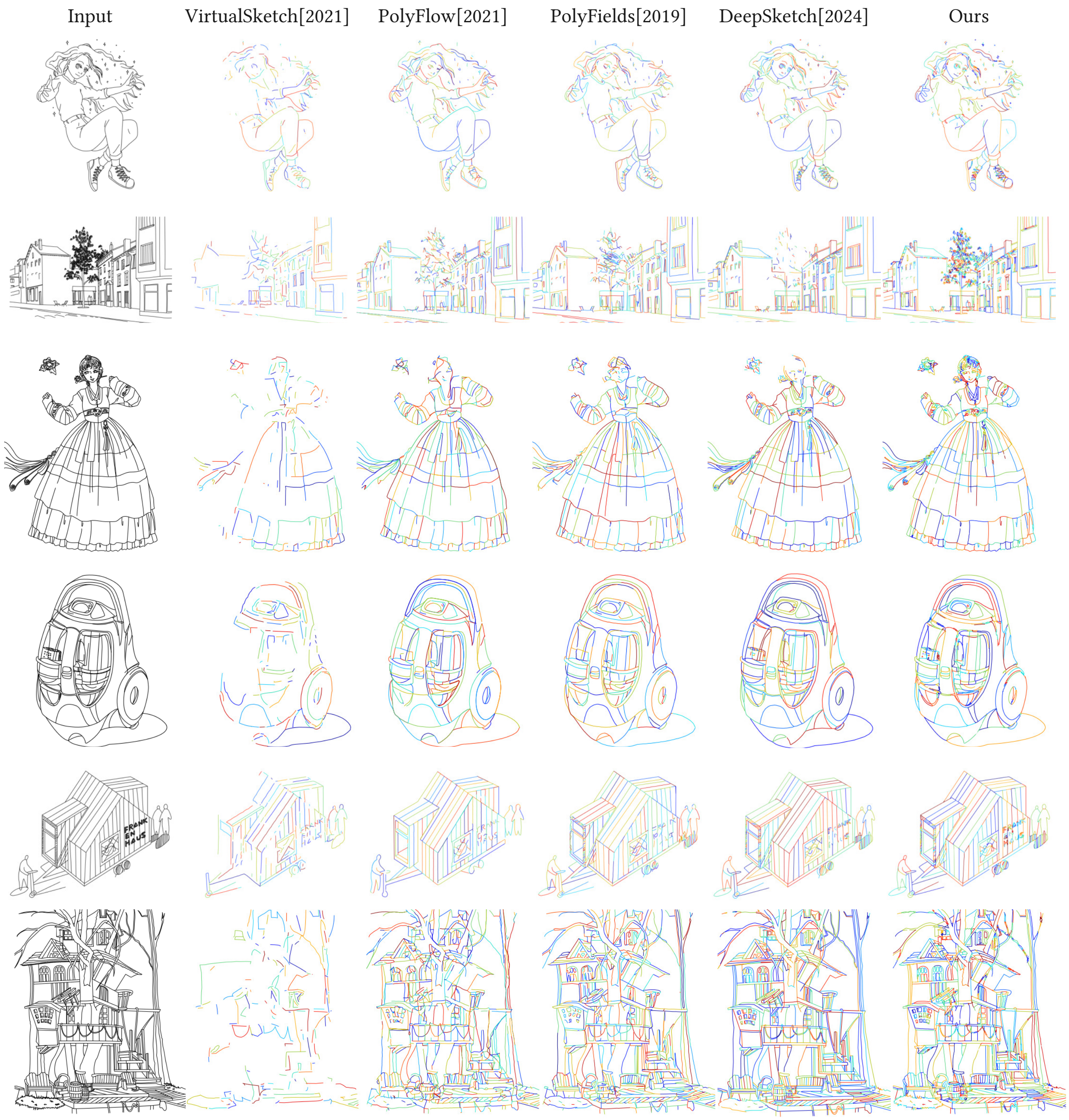

Fig. 1. Additional qualitative comparisons on *clean* line art (black strokes, uniform width) from the same benchmark family as in the main paper. Each row shows the raster input, several baselines (VirtualSketch, PolyFlow, PolyFields, DeepSketch), and our result. These examples supplement the main qualitative comparison figure and highlight cases where our reconstruction remains faithful to thin structures and stroke topology while staying compatible with downstream editing.

References

Mikhail Bessmeltsev and Justin Solomon. 2019. Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 1–12.

Or Hirschorn, Amir Jevnisek, and Shai Avidan. 2024. Optimize & reduce: a top-down approach for image vectorization. In *Proceedings of the AAAI Conference on Artificial*

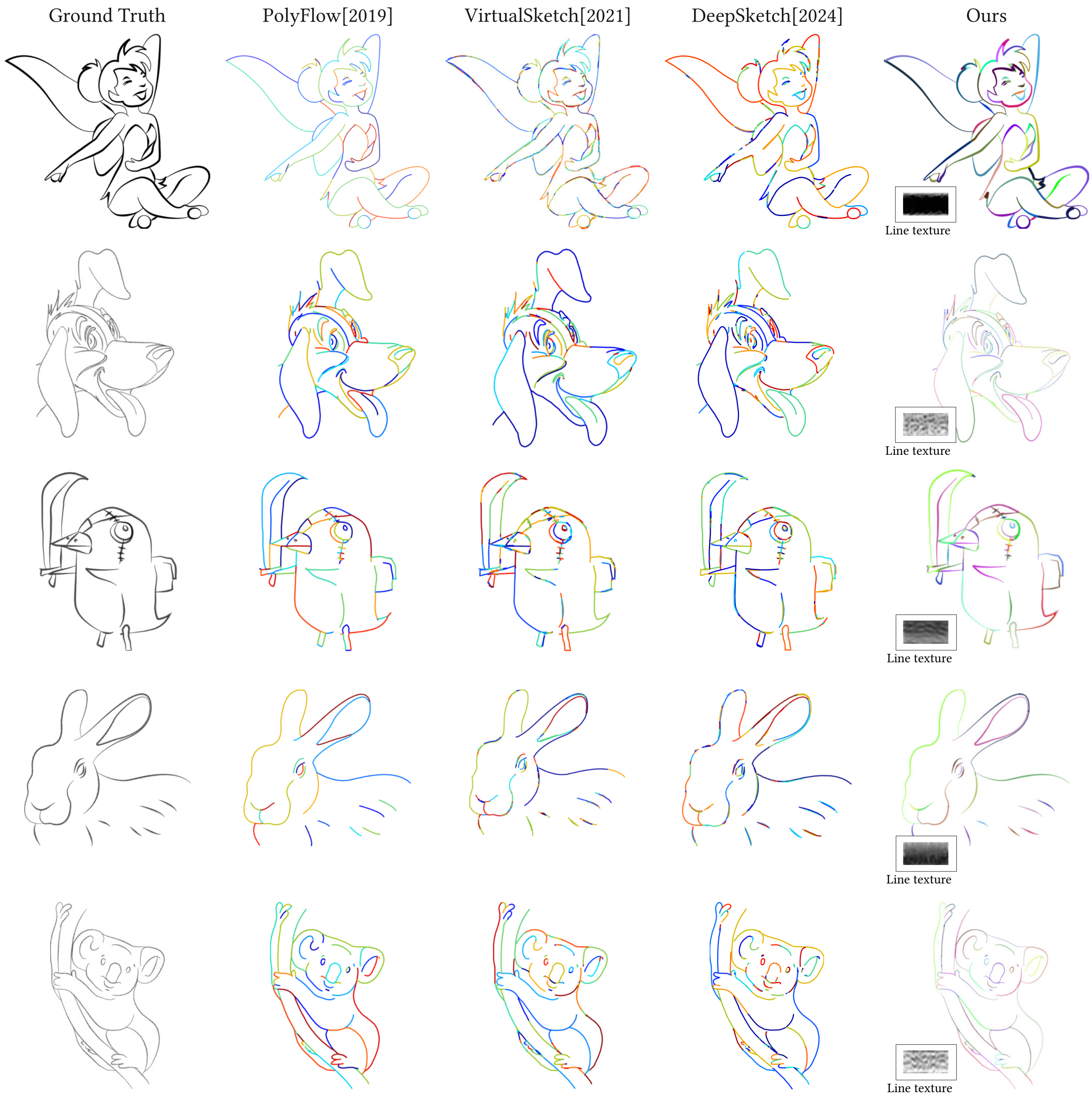

Fig. 2. Additional qualitative comparisons on *textured* line art with varying stroke width and brush appearance, supplementing the main textured comparison figure. Strokes are shown in distinct colors. Our vectorization tends to assign colors more consistently with semantic parts and avoids merging unrelated regions compared to other methods. We also show the estimated brush texture.

Intelligence, Vol. 38. 2148–2156.

Haoran Mo, Chengying Gao, and Ruomei Wang. 2024. Joint stroke tracing and correspondence for 2d animation. *ACM Transactions on Graphics* 43, 3 (2024), 1–17.

Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. 2021. General virtual sketching framework for vector line art. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

Resolution	Metric	VirtualSketch[2021]	PolyFlow[2021]	PolyFields[2019]	DeepSketch[2024]	Ours
256px	Success Rate ↑	100%	99.73%	100%	100%	100%
	Chamfer Dist. ↓	4427.54	8.78	7.02	5.62	2.91
	Str. Len. Diff. ↓	92.57%	17.49%	19.04%	8.53%	4.94%
	Str. Count	14.70	132.51	139.08	89.87	114.59
	LPIPS $\times 10^{-3}$ ↓	436.31	56.13	47.52	20.22	14.84
	PSNR ↑	23.82	30.32	30.36	33.31	37.30
	SSIM $\times 10^{-2}$ ↑	73.64	92.26	92.17	96.10	98.39
384px	Success Rate ↑	100%	98.64%	100%	100%	100%
	Chamfer Dist. ↓	945.04	4.23	3.82	3.65	2.65
	Str. Len. Diff. ↓	67.97%	7.62%	8.08%	5.37%	2.48%
	Str. Count	40.50	138.37	141.19	97.60	88.25
	LPIPS $\times 10^{-3}$ ↓	283.33	33.52	26.94	13.60	8.12
	PSNR ↑	24.93	30.81	30.98	32.73	38.24
	SSIM $\times 10^{-2}$ ↑	80.56	96.22	96.01	97.63	99.25
512px	Success Rate ↑	100%	97.56%	100%	100%	100%
	Chamfer Dist. ↓	149.29	1.75	1.55	1.45	0.94
	Str. Len. Diff. ↓	34.06%	10.59%	4.58%	3.14%	1.40%
	Str. Count	84.86	141.37	143.84	94.58	113.65
	LPIPS $\times 10^{-3}$ ↓	124.33	21.93	20.58	10.07	5.15
	PSNR ↑	25.87	31.28	30.92	32.44	39.70
	SSIM $\times 10^{-2}$ ↑	89.96	97.90	97.28	98.37	99.62
768px	Success Rate ↑	100%	95.93%	97.29%	100%	100%
	Chamfer Dist. ↓	10.25	1.36	1.19	1.26	1.15
	Str. Len. Diff. ↓	9.86%	12.04%	2.70%	2.17%	1.23%
	Str. Count	125.35	194.91	148.68	95.43	108.02
	LPIPS $\times 10^{-3}$ ↓	36.50	14.81	13.76	6.67	3.26
	PSNR ↑	27.04	31.19	30.75	32.43	38.98
	SSIM $\times 10^{-2}$ ↑	96.36	98.64	98.39	99.06	99.75
1024px	Success Rate ↑	100%	92.95%	94.85%	100%	100%
	Chamfer Dist. ↓	18.53	1.34	0.87	1.06	0.89
	Str. Len. Diff. ↓	8.27%	14.19%	1.64%	1.69%	0.84%
	Str. Count	123.43	212.84	132.62	95.65	108.40
	LPIPS $\times 10^{-3}$ ↓	35.69	24.94	9.87	5.34	1.80
	PSNR ↑	27.04	28.87	31.08	32.49	40.13
	SSIM $\times 10^{-2}$ ↑	97.36	98.07	98.97	99.34	99.87

Table 1. Quantitative comparison on the *clean* benchmark dataset [Yan et al. 2024] over input resolutions (metrics summarized in the preceding paragraph).

Ivan Puhachov, William Neveu, Edward Chien, and Mikhail Bessmeltsev. 2021. Keypoint-driven line drawing vectorization via PolyVector flow. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–17.

Chuan Yan, Yong Li, Deepali Aneja, Matthew Fisher, Edgar Simo-Serra, and Yotam Gingold. 2024. Deep sketch vectorization via implicit surface extraction. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–13.

Resolution	Metric	VirtualSketch[2021]	PolyFlow[2021]	PolyFields[2019]	DeepSketch[2024]	Ours
256px	Success Rate ↑	100%	98.91%	99.73%	100%	100%
	Chamfer Dist. ↓	422.24	17.62	16.36	6.52	4.20
	Str. Len. Diff. ↓	20.20%	16.28%	18.46%	10.95%	10.89%
	Str. Count	156.09	121.75	89.44	88.72	56.96
	LPIPS $\times 10^{-3}$ ↓	240.64	91.01	193.93	77.74	22.85
	PSNR ↑	16.72	18.09	17.47	19.50	26.84
	SSIM $\times 10^{-2}$ ↑	72.34	87.79	77.99	91.19	98.23
384px	Success Rate ↑	100%	98.64%	99.73%	100%	100%
	Chamfer Dist. ↓	299.50	14.87	12.37	2.96	3.14
	Str. Len. Diff. ↓	15.79%	13.46%	15.11%	6.89%	6.81%
	Str. Count	234.65	161.87	119.09	97.28	72.99
	LPIPS $\times 10^{-3}$ ↓	170.09	130.06	129.98	116.00	21.49
	PSNR ↑	18.33	18.17	19.09	18.72	29.16
	SSIM $\times 10^{-2}$ ↑	78.74	86.34	82.58	88.04	98.75
512px	Success Rate ↑	100%	96.21%	99.19%	100%	100%
	Chamfer Dist. ↓	622.12	4.52	4.47	2.05	1.13
	Str. Len. Diff. ↓	14.51%	11.60%	11.06%	4.68%	5.08%
	Str. Count	103.32	201.29	156.92	108.02	121.85
	LPIPS $\times 10^{-3}$ ↓	159.04	117.06	119.12	109.36	18.43
	PSNR ↑	18.40	18.99	19.08	19.44	29.02
	SSIM $\times 10^{-2}$ ↑	86.23	88.94	88.73	89.96	98.77
768px	Success Rate ↑	100%	94.04%	97.02%	100%	100%
	Chamfer Dist. ↓	302.50	6.69	6.97	3.83	1.21
	Str. Len. Diff. ↓	16.88%	11.77%	10.08%	3.11%	3.12%
	Str. Count	420.15	240.21	192.41	97.60	101.78
	LPIPS $\times 10^{-3}$ ↓	103.86	92.16	59.86	83.97	18.96
	PSNR ↑	21.98	24.33	23.39	21.98	32.54
	SSIM $\times 10^{-2}$ ↑	89.94	92.72	92.39	93.20	99.35
1024px	Success Rate ↑	100%	91.87%	94.31%	100%	100%
	Chamfer Dist. ↓	555.44	4.89	4.78	1.96	1.02
	Str. Len. Diff. ↓	22.28%	12.25%	10.10%	2.15%	2.45%
	Str. Count	516.03	291.46	238.16	105.54	110.58
	LPIPS $\times 10^{-3}$ ↓	100.19	64.08	39.96	40.06	17.22
	PSNR ↑	23.93	24.41	27.03	26.22	33.93
	SSIM $\times 10^{-2}$ ↑	94.65	95.56	97.24	97.31	99.49

Table 2. Quantitative comparison on the *textured* benchmark dataset [Yan et al. 2024] over input resolutions (metrics summarized in the preceding paragraph).

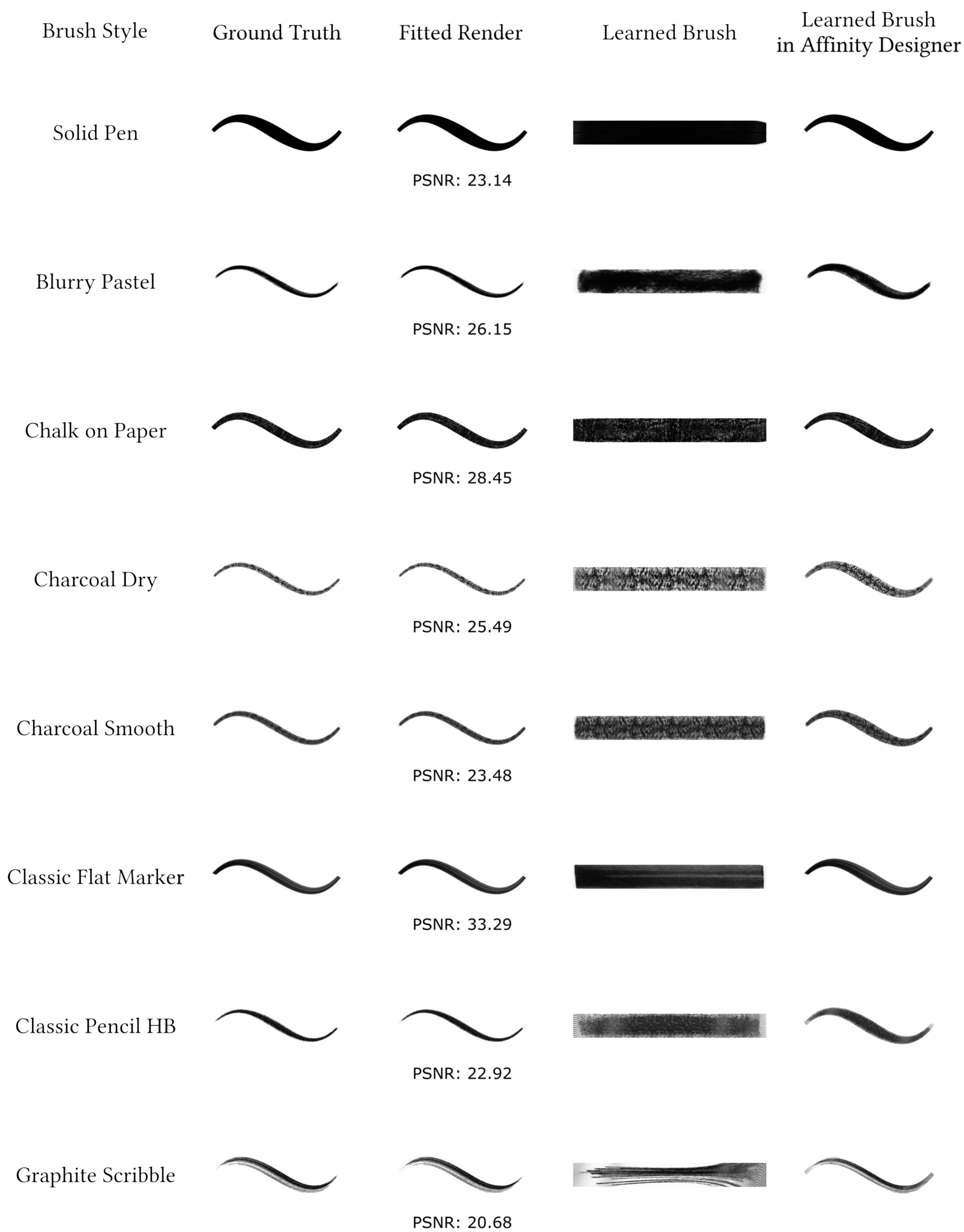

Fig. 3. Brush appearance and fitting fidelity. On each row we show the original brush, the rendering after fitting, the learned brush texture and then the application of the brush in DCC software.

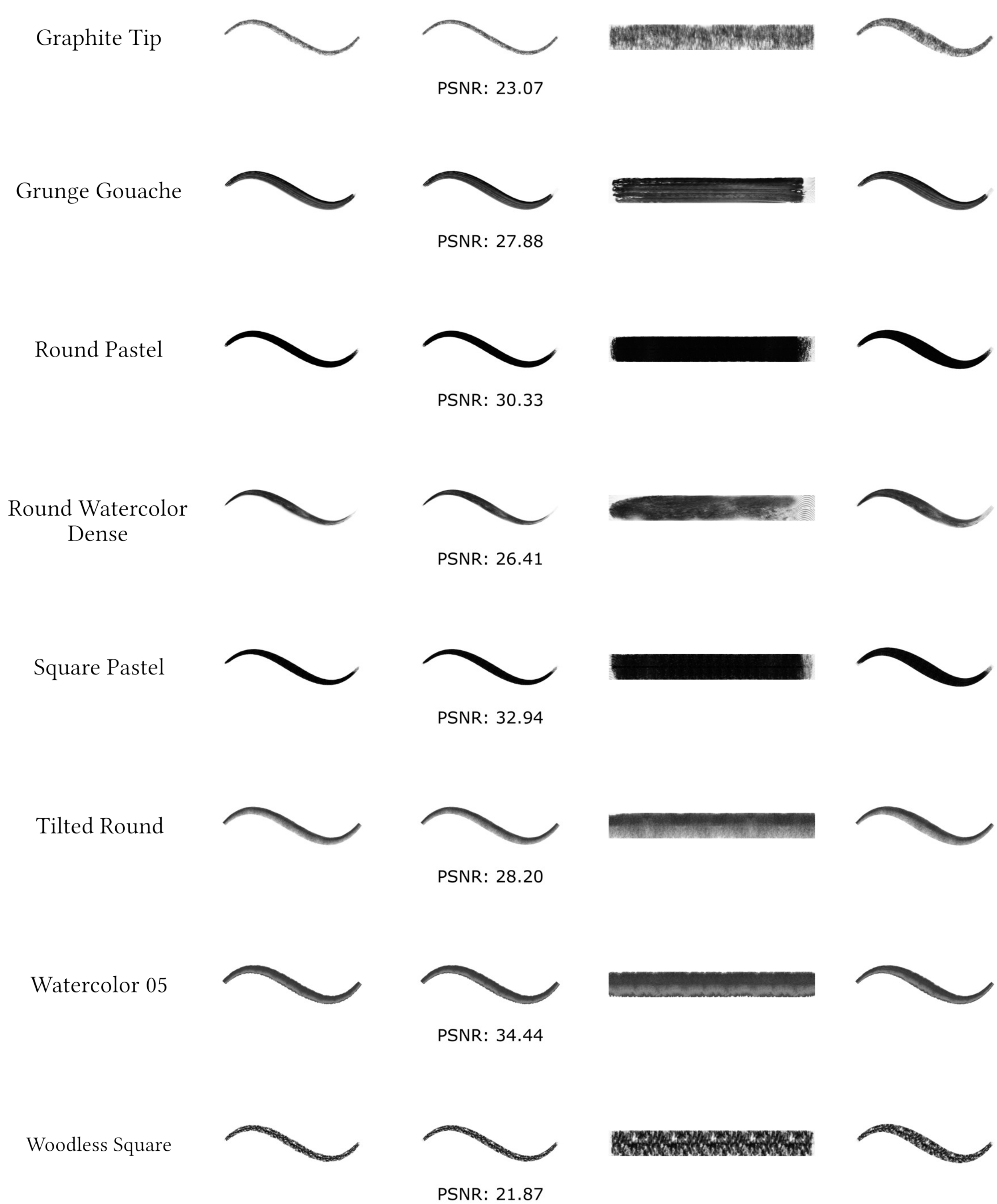

Fig. 4. Brush appearance and fitting fidelity (additional results). On each row we show the original brush, the rendering after fitting, the learned brush texture and then the application of the brush in DCC software..

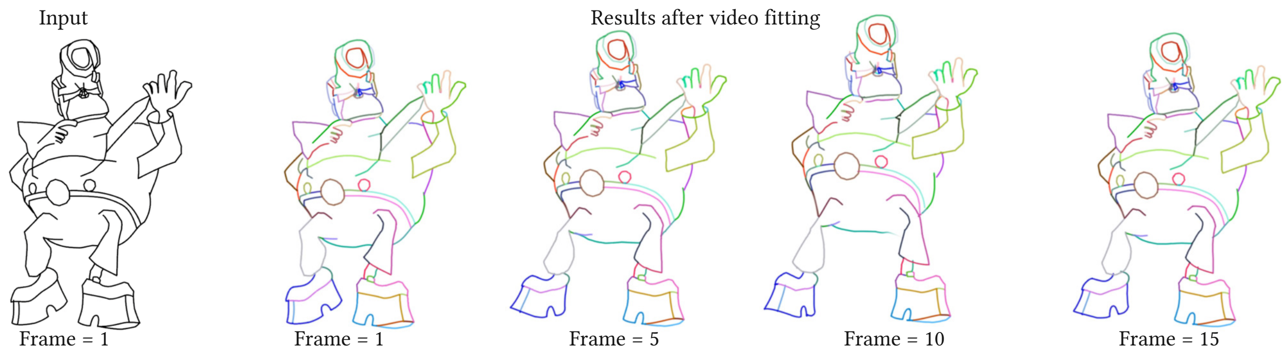

Fig. 5. Additional video vectorization results using the dataset and setup of Mo et al. [2024]. We jointly optimize Bézier strokes across all frames with our differentiable 2D Gaussian renderer; stroke colors indicate persistent stroke identities as they are tracked over time. The figure includes multiple sequences or timesteps so that temporal coherence and reconstruction quality can be compared to per-frame fitting (discussed in the main paper's video figure).

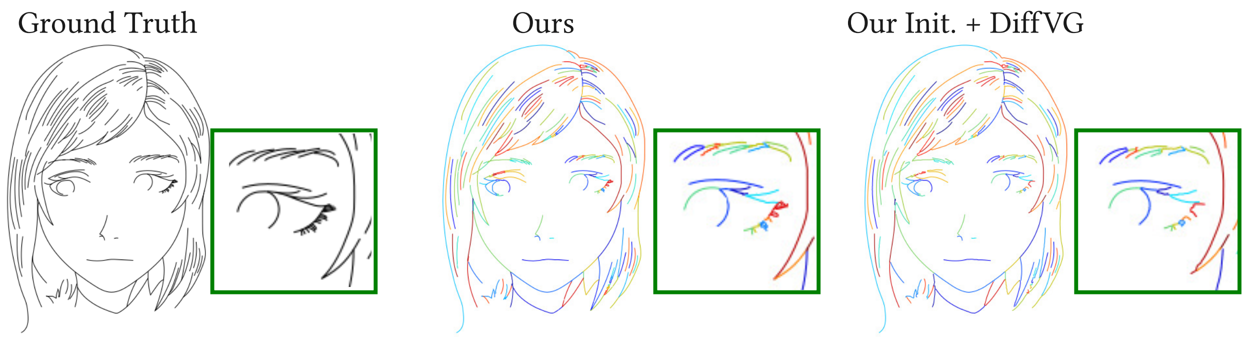

Fig. 6. Qualitative comparisons between using our differentiable renderer and DiffVG.

Fig. 7. Starting from a randomly initialized set of strokes (1024), we compare using our differentiable renderer and DiffVG as back end.