Pacific Graphics (2022)
N. Umetani, E. Vouga, and C. Wojtan (Editors)

Learning a Style Space for Interactive Line Drawing Synthesis from Animated 3D Models (Supplementary Material)

Zeyu Wang[1], Tuanfeng Y. Wang[2] and Julie Dorsey[1]

[1]Yale University [2]Adobe Research

1. Overview

In the supplementary material, we describe the details of input drawing embedding at run time, dataset preparation, network architecture, and training strategy. We provide a systematic evaluation with various test cases and show that our approach learns a powerful latent style space for effective line drawing animation synthesis given different types of user input. We then validate our system design choices via ablation studies.

2. Input Drawing Embedding

At run time, we allow the user to create a line drawing at a keyframe in the animation. We need to obtain a style code from the input drawing that can faithfully reconstruct itself via our generator. The source of the input drawing can be 1) a NPR rendering of the keyframe with a set of user-specified parameters; 2) manual editing based on 1), e.g., removing or adding strokes; and 3) drawing from scratch over a reference image of the underlying geometry. Due to the huge space of drawing variation and limited capacity of NPR drawing generation, our StyleNet (E_S) only works well with input from 1). To enable an interactive editing workflow with full user control, during test time, we adopt an optimization-based approach to embed an input drawing I_a at keyframe a into the latent style space. Specifically, we compute geometry feature g when a keyframe is selected. We freeze the weights of our generator G and optimize a latent code z^* so that the generated image, $I'_a = G(g;z)$, is similar to the input drawing. Since the gradient of our binary drawing can be unstable for optimization, we adopt a pyramid structure with different levels of blurring, i.e.,

$$z^* = \arg\min_z \sum_{k=1,33,65} \left\| GauSm_k(I_a) - GauSm_k(I'_a) \right\|_1, \quad (1)$$

where k is the kernel size of the Gaussian smoothing operator in pixels. We initialize the optimization with the projection of StyleNet, i.e., $z_0 = E_S(I_a, M_a)$.

3. Implementation Details

3.1. Training Data

We evaluate our approach on five animation sequences: `Mouse` [ZZCB21] (480 frames), `Lilly` [3DP22] (1,200 frames), `Human` [Ado22] (3,000 frames), `Michelle` [Ado22] (200 frames), and `Vegas` [Ado22] (123 frames). For `Human`, we use the first 2,000 frames for network training and the rest of the sequence is only used for generalization test on unseen frames. For each scenario, we use NPR [DFRS03] to build a synthetic dataset for network training. For each frame, we use three NPR methods, i.e., suggestive contours, ridges and valleys, and apparent ridges, and sample four thresholds for each. We combine outputs from these methods and generate 64 ($4 \times 4 \times 4$) NPR line drawings for each frame and 4 additional Canny edge maps [Can86] generated with different thresholds. For each frame, it takes about 15 seconds on average to generate all 68 line drawings with a resolution of $W, H = 512, 512$. We apply commonly used techniques for 2D data augmentation, including translation, rotation, scaling, and flipping. We show samples training data for `Human` in Figure 1. Figure 2 shows an example of the input channels that represent the underlying geometry at each frame with a set of geometric properties.

Figure 1: *Sample training data for the `Human` character. Our synthetic dataset consist of line drawings generated with different non-photorealistic rendering methods with varying parameters.*

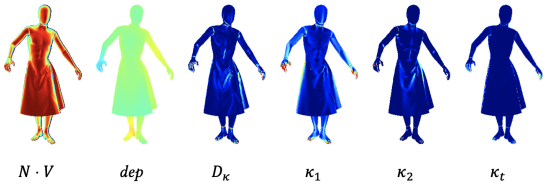

Figure 2: *Input to GeoNet E_G. We concatenated six geometric properties, i.e., shading, depth map, and four types of surface curvature. Blue represents low values and red represents high values.*

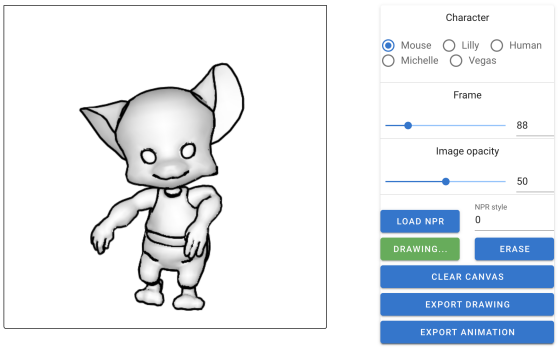

Figure 3: *User interface for drawing editing and interactive animation authoring. It allows the user to load an NPR line drawing, edit an existing drawing, or draw from scratch. We show a shading image of the geometry at the target frame in the background with adjustable opacity as a reference. In the animation mode, user can play the sequence, add, edit, and delete keyframes.*

3.2. Data Collection for Evaluation

We collect line drawings from three sources to evaluate our system.

1. Similar to the synthetic dataset generation pipeline, we generate unseen NPR line drawings for selected frames from the animation sequence with random sampled NPR parameters. We denote these drawings as NPR.
2. We allow users to edit an NPR line drawing by adding new strokes or erase existing ones. We build a user interface for this purpose as shown in Figure 3. It allows users to select a frame and load an NPR line drawing from the training set or from NPR. We show a shading image of the geometry at the selected frame in the background with adjustable opacity. Users can draw and erase strokes on the left canvas. We denote these drawings as NPR w/ edits.
3. More experienced users can directly create line drawings from scratch by looking at the reference shading image. We denote these drawings as Freehand.

3.3. Network Architecture and Training Strategy

Our GeoNet, E_G, is a neural feature extractor built with convolutional layers followed by Parametric Rectifying Linear Unit (PRelu) activation [HZRS15] and batch normalization. Starting from the multi-channel geometric signal map $M_a \in \mathbb{R}^{512\times512\times6}$, our network gradually decreases the dimension of the output to 256, 128, 64, and 32, while increases the number of channels gradually to 32, 64, 256, and 512 after each layer. This results in a 2D geometric feature, $g \in \mathbb{R}^{32\times32\times512}$. Our StyleNet, E_S, adopts a similar architecture but maps the input line drawing $I_a \in \mathbb{R}^{512\times512\times1}$ into a 1D style code, $z \in \mathbb{R}^{1\times1\times2048}$ with five more layers. The architecture of our generator, G, follows StyleGAN2 [KLA*20] including bilinear upsampling, equalized learning rate, noise injection at every layer, variance adjustment of residual blocks and leaky ReLU. The final output of this pipeline is a line drawing image $I'_a \in \mathbb{R}^{512\times512\times1}$. We trained our network with a learning rate of 0.02 using four NVIDIA V100 GPUs with a batch size of 4. It takes about 48 hours on average to converge in our experiments.

During the drawing embedding step, we implement the optimization using the *optim* package from PyTorch [Fac20]. We choose LBFGS algorithm [LN89] as our optimizer with a learning rate of 0.1. The optimization takes place on a single NVIDIA V100 GPU for 100 steps for all the cases discussed in this paper. This optimization takes about 30 seconds for 50 steps on average.

4. Evaluation

We evaluate our system with respect to line drawing synthesis, style interpolation, latent code embedding, and interactive editing. Experiments show that our approach outperforms vanilla style transfer and style interpolation baselines. Our approach produces plausible output that follows the properties of user input and transitions naturally in the animation. Although our network is trained in a case-specific manner, we show that it is easy to generalize to new cases with quick finetuning. Users of our system think the synthesized line drawings are consistent with their style and are useful for efficient and controllable animation authoring.

Latent style space embedding. We first evaluate the latent space embedding for a given line drawing. As discussed in the paper, the style code directly predicted from an input drawing may not be accurate enough for an edited NPR drawing or one drawn from scratch. In Figure 4, we see that the drawing generated from the learned latent style code z is similar to the target input in general but missing quite some details especially for NPR w/ edits and Freehand cases. With latent code optimization, the generated drawing can recover strokes missing from the initial projection. Our pyramid strategy facilitates gradient propagation and leads to better reconstruction compared to the one without the pyramid as shown by the $L_1 + \text{VGG}$ error. In case b), the pyramid strategy helps recover strokes that are missing in the optimization without the pyramid.

Disentanglement between style and geometry. Since we cast the style as an intrinsic property that depicts the relationship between stroke placement and geometric features, same drawing style at different frames in a sequence should be mapped to the same place in the latent style space. In other words, no matter which frame the style is learned from, the latent style code should all produce the same drawing for the target frame. Our NPR dataset provides consistent style across frames in a sequence. In Figure 5, given a target line drawing I_a at a certain frame a, we randomly pick two other frames b and c from the same sequence. The corresponding drawings with the same style are denoted as I_b and I_c. We show the style code extracted from I_b and I_c can be used to faithfully reconstruct the target drawing I_a.

Style interpolation. Our latent style space is learned from a set of separated NPR line drawings. The dataset itself cannot provide supervision on smooth style transition due to the nature that some NPR parameters are not interpolatable. Instead, our interpolation loss and strokeness loss provide self-supervision for this purpose. Here, we evaluate the performance of our method on interpolating between two projected latent style codes. Starting from style interpolation at a fixed frame, we learn the geometric features g and keep it unchanged during the interpolation. We embed the source

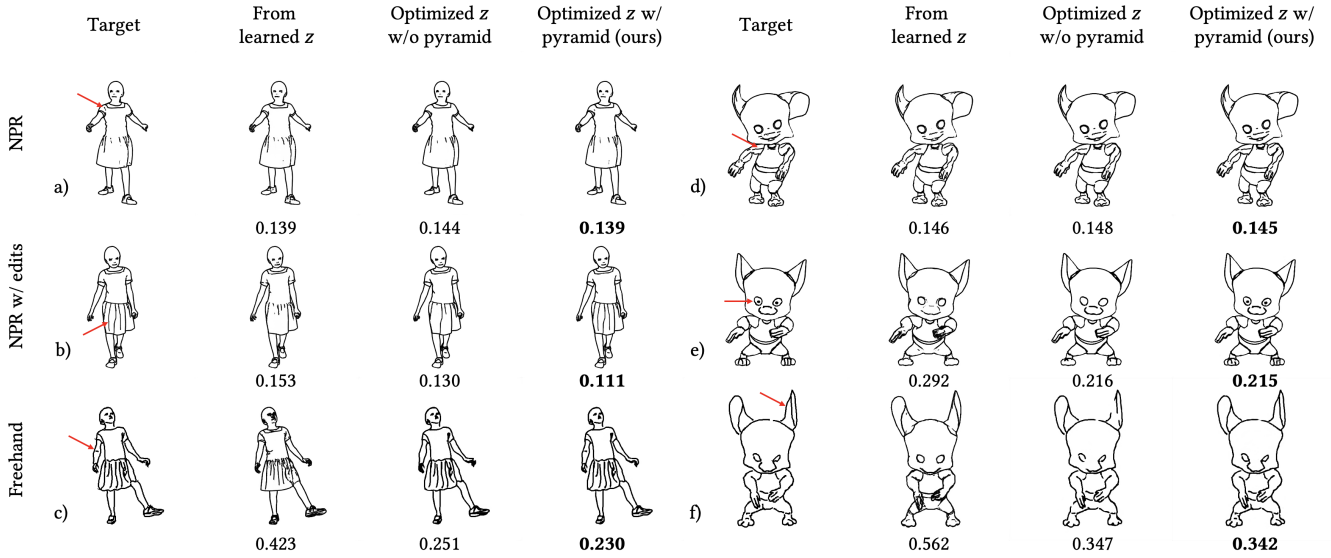

Figure 4: *Evaluation of latent style code embedding. We test our style embedding approach on different types of line drawings, i.e., NPR, NPR w/ edits, and Freehand. We show generated drawings from 1) learned latent style code z, 2) optimized z without the pyramid strategy, and 3) optimized z with the pyramid strategy (ours). Our approach achieves best reconstruction from the latent embedding. Red arrows highlight challenging areas. Ours achieves the lowest $L_1 + VGG$ error as shown by the values below each entry.*

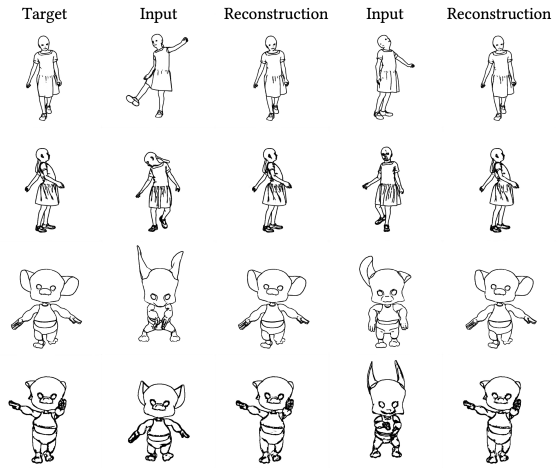

Figure 5: *Disentanglement between geometry and style. For each target drawing (1^{st} col.), we learn the latent style code z from one rendered with the same NPR parameters but at a different frame (2^{nd} and 4^{th} cols.) The drawing reconstruction from the learned style codes is almost identical at the target frame (3^{rd} and 5^{th} cols.)*

and target line drawings into latent space to obtain the corresponding latent style codes z_{source} and z_{target}. We perform linear interpolation between z_{source} and z_{target} to generate the transitioning line drawings accordingly, as shown in Figure 6.

An alternative baseline method for this task is to adopt a vanilla StyleGAN [KLA*20]. In the even rows of Figure 6, we train a vanilla StyleGAN with the same NPR dataset and perform the drawing embedding with our pyramid-based optimization. Instead of performing interpolation in the style space, we directly perform interpolation in the W space for the vanilla StyleGAN to generate corresponding output. The optimization-based embedding performs reasonable for reconstructing target drawings in the W space [KLA*20]. However, without an explicit disentanglement of style and geometry, the interpolation results using the vanilla StyleGAN have heavy artifacts.

Table 1: *Statistics of style interpolation on static frame in Figure 6. We calculate sparsity loss, interpolation loss, and the strokeness loss along the interpolated sequence. We show our method outperforms the baseline approach on style interpolation.*

		Lilly			Mouse		
		L_{sparsity}	L_{interp}	L_{stroke}	L_{sparsity}	L_{interp}	L_{stroke}
NPR	Ours	**0.069**	**0.044**	**0.025**	**0.098**	**0.073**	**0.042**
	Baseline	0.082	0.207	0.113	0.130	0.237	0.181
NPR w/ edits	Ours	**0.054**	**0.047**	**0.020**	**0.075**	**0.097**	**0.034**
	Baseline	0.059	0.275	0.147	0.099	0.190	0.148
Freehand	Ours	0.073	**0.096**	**0.047**	**0.072**	**0.126**	**0.052**
	Baseline	**0.072**	0.283	0.148	0.090	0.244	0.168

In this experiment, we test line drawings from NPR, NPR w/ edits, and Freehand. We show that our method learns a robust latent style space for all the three types of drawings. In Table 1, we report the sparsity loss, interpolation loss, and the strokeness loss along the interpolated sequence. We show that our method achieves lower loss which agree with the qualitative comparison.

Next, we elaborate on the experiment setup for an animation sequence rather than a static frame. The geometric features g are

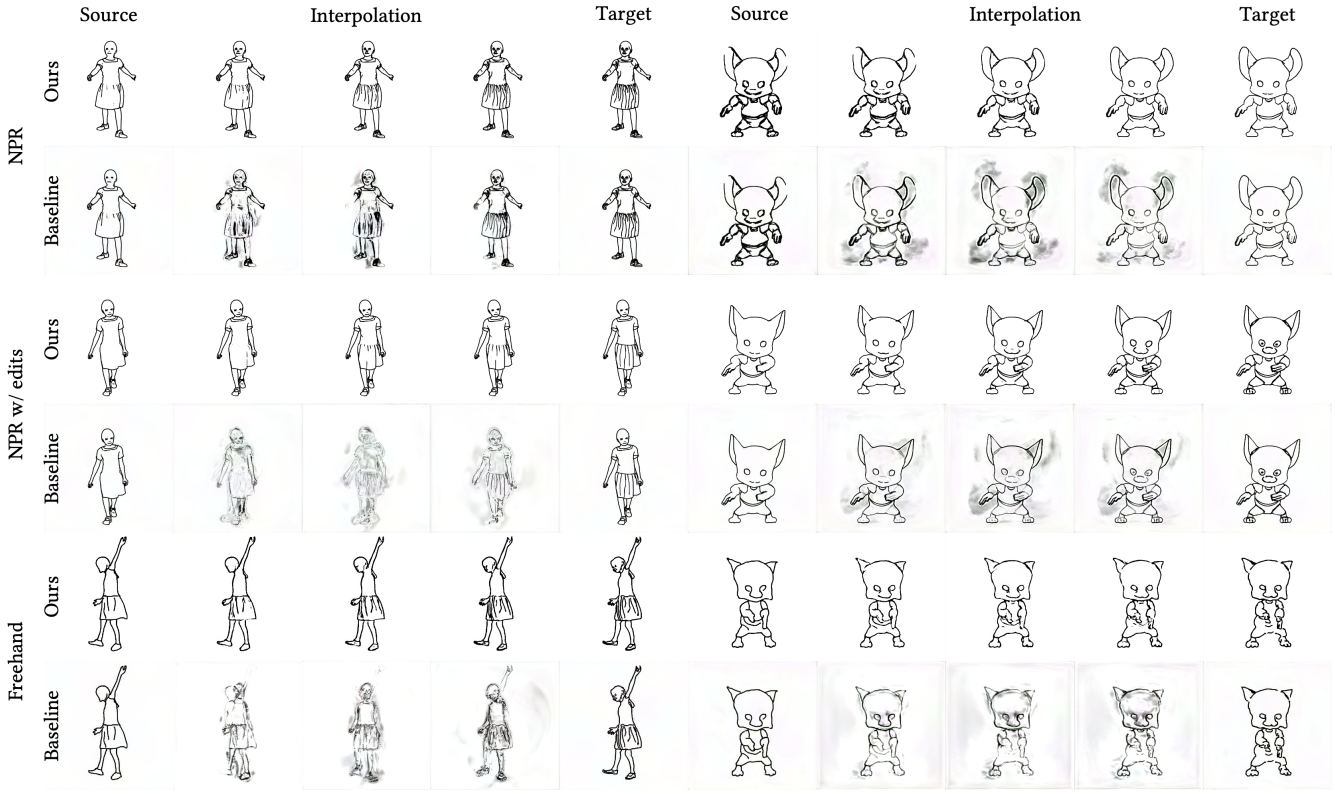

Figure 6: *Style interpolation at a static frame. For each case, the source line drawing (1^{st} or 6^{th} column) and the target (5^{th} or 10^{th} column) are given and embedded into the style space. Linearly interpolated style codes are used to generate transitioning line drawings (2^{nd}–4^{th} or 7^{th}–9^{th} columns). We compare our approach (odd rows) with a baseline method (even rows), vanilla StyleGAN trained with the same dataset. Our approach outperforms the baseline method due to geometry/style disentanglement and drawing-specific supervision during training.*

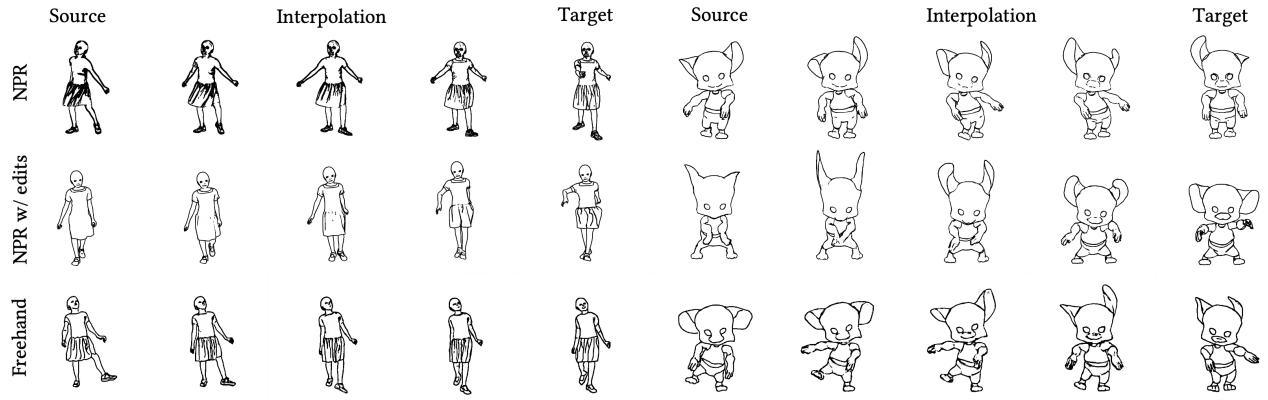

Figure 7: *Style interpolation between dynamic frames. We evaluate our system on animation sequences with dynamic frames using the same experiment setup from Figure 6.*

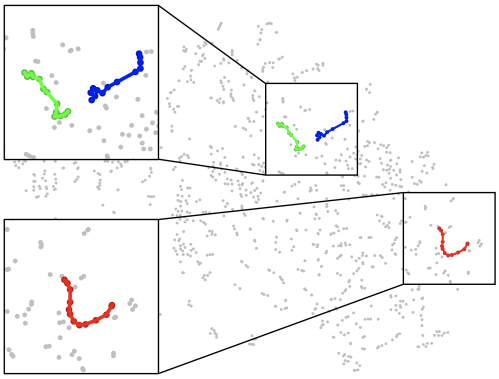

Figure 8: *t-SNE visualization of style interpolation between dynamic frames. For the* `Lilly` *dataset, we calculate the VGG features for* 10*k randomly sampled NPR line drawings in the training dataset and embed them into a 2D space via t-SNE [VdMH08]. We then calculate the VGG features for each frame in the three clips shown in Figure 7. We visualize the three sequences (NPR: red, NPR w/ edits: green, Freehand: blue) with the same 2D embedding. The trajectories indicate that the predicted style interpolation between frames is perceptually smooth and generalized well to unseen styles (NPR w/ edits and Freehand).*

learned from the underlying geometry at each frame accordingly, while the style code z is still interpolated linearly between the source and target line drawings. As shown in Figure 7, our learned latent space supports different types of line drawings for interpolation between the frames. The generated line drawings have a smoothly transitioning style along the sequence. We also visualize the drawing sequence as a trajectory in the 2D embedding space as shown in Figure 8. Specifically, for the test animation, we randomly sample 10k NPR line drawings from our dataset and compute the 512-dimensional VGG feature for each drawing. We apply t-distributed stochastic neighbor embedding (t-SNE) [VdMH08] for the VGG features. For the sequences in Figure 7, we embed the VGG feature for each entry into the same t-SNE domain. We see that the line drawing animation with linear interpolated style forms a smooth trajectory in the perceptual embedding space. This backs up our observation that the transition between source and target line drawings is smooth and natural.

5. Ablation Study

We use an ablation study to validate our design choices. Starting from a source edited NPR line drawing, we perform style interpolation on a static frame towards a target edited NPR line drawing. Figure 9 shows the effect of the loss terms used in network training. We see that the three losses proposed to self-supervise drawing interpolation, i.e., sparsity loss, interpolation loss, and strokeness loss, are functioning as expected as the toy examples discussed in the paper. We observe a smoother transition from the source to the target in our full pipeline. We also observe that those losses provide supervision over the gaps between samples in the NPR dataset, which helps the method generalize to unseen input line drawings.

We adopt the same style interpolation setup to evaluate the effect of latent space dimension. We train the same network but reduce the dimension of the latent style space from 2048 to 1024 to 512. After training with the same number of epochs, we compare their style interpolation performance in Figure 10. We observe that the quality of line drawing synthesis is improved along with the increase of the latent space dimension, where the model size and training time for each epoch are increased as well. Therefore, our choice of a 2048-dimensional latent style space is a reasonable tradeoff between training efficiency and network performance.

References

[3DP22] 3DPEOPLE: 3D People for Your Visualizations and Animations. https://3dpeople.com/, 2022. 1

[Ado22] ADOBE: Mixamo. https://www.mixamo.com, 2022. 1

[Can86] CANNY J.: A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6 (Nov 1986), 679–698. 1

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive Contour Software: rtsc. https://rtsc.cs.princeton.edu, 2003. 1

[Fac20] FACEBOOK: torch.optim. https://pytorch.org/docs/stable/optim.html, 2020. 2

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1026–1034. 2

[KLA*20] KARRAS T., LAINE S., AITTALA M., HELLSTEN J., LEHTINEN J., AILA T.: Analyzing and Improving the Image Quality of StyleGAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2020), pp. 8110–8119. 2, 3

[LN89] LIU D. C., NOCEDAL J.: On the Limited Memory BFGS method for Large Scale Optimization. *Mathematical Programming 45*, 1 (1989), 503–528. 2

[VdMH08] VAN DER MAATEN L., HINTON G.: Visualizing Data Using t-SNE. *Journal of Machine Learning Research 9*, 11 (2008). 5

[ZZCB21] ZHENG M., ZHOU Y., CEYLAN D., BARBIC J.: A Deep Emulator for Secondary Motion of 3D Characters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2021), pp. 5932–5940. 1

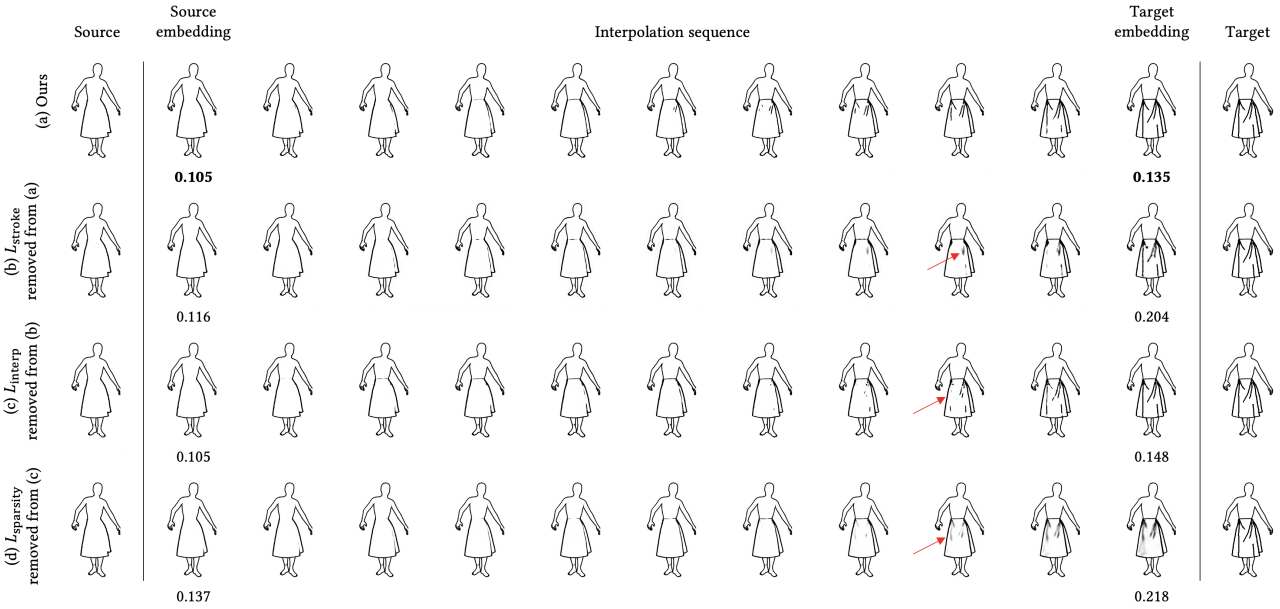

Figure 9: *Ablation study of loss terms. Given the same source and target line drawings, we first embed the drawings to obtain their latent style codes. We linearly interpolated the style codes to generate the in-between drawings. From top to bottom, we first show our method (a), and gradually remove strokeness loss (b), interpolation loss (c), and sparsity loss (d). Reconstruction error is reported for the embedding of source and target drawings. Red arrows highlight the artifacts in the interpolation. We show that, with all the loss terms, our approach performs the best for style interpolation and reconstruction of unseen input line drawings.*

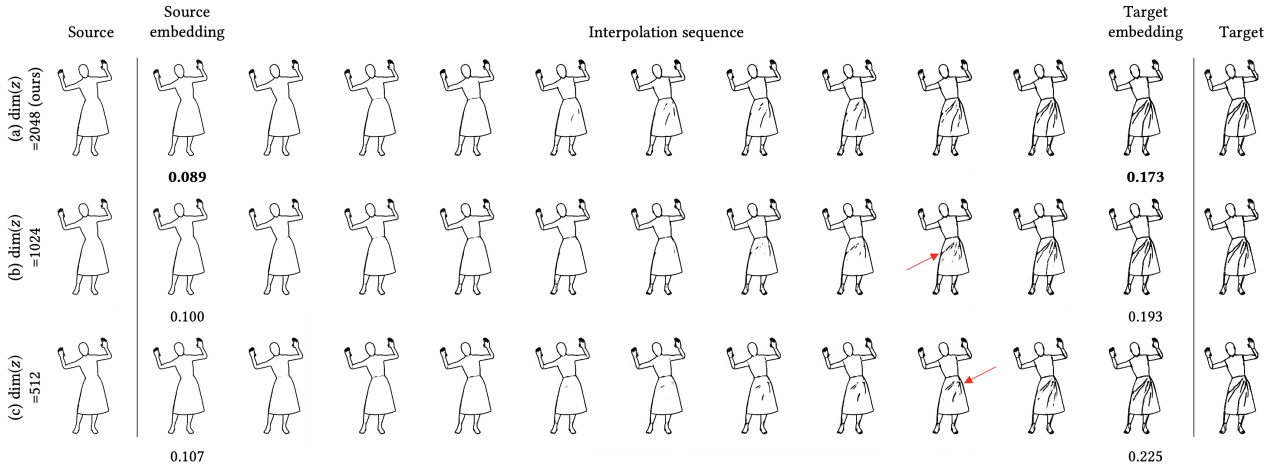

Figure 10: *Ablation study of latent space dimension. We train the same network with a latent space dimension gradually decreasing from 2048 to 1024 to 512. With an experiment setup similar to Figure 9, we show that a 2048-dimensional latent space (ours) is more capable of learning diverse drawing styles. Decreasing latent space dimension worsens the performance in drawing embedding and style interpolation.*