Supplementary Material
Neural Recognition of Dashed Curves with Gestalt Law of Continuity

A. Training setup

During the training, we use three separated Adam optimizers for Curve Feature Aggregator, Visual Form Reconstructor, and Semantic Extractor. We set the initial learning rate of each optimizer to be 0.0001. We use the StepLR scheduler from the PyTorch library to decay the learning rate of Curve Feature Aggregator and Semantic Extractor by 0.95 every 30 epochs. The learning rate of Visual Form Reconstructor is decayed by 0.9 every 50 epochs. We trained our models using 4 NVIDIA Titan RTX GPUs with gradient checkpointing [1] and automatic mixed precision.

Given the fact that input images contain different numbers of dashed curves, our framework is slightly different with a classical auto-encoder: the output curve descriptor list may contain invalid embeddings that do not contribute to the reconstruction. Even though the ground truths are well-formed to provide supervision, the unbalanced throughput between Curve Feature Aggregator and Visual Form Reconstructor/Semantic Extractor makes the initial training unstable. To mitigate this issue, we split the modeling training into three stages: we first warm up the Curve Feature Aggregator with only the endpoint regression supervision and curve descriptor validity classification supervision until the average classification accuracy reaches a certain threshold $\theta_w = 0.7$; we then add Visual Form Reconstructor and Semantic Extractor into training while fixing the weights of the Curve Feature Aggregator for $T = 50$ epochs; finally, we jointly train all three modules until convergence. In the training stages involving Visual Form Reconstructor and Semantic Extractor, we run the `forward` and `backward` process for each dashed curve separately and leverage the gradient accumulation trick to reduce GPU memory usage.

B. Data augmentation

To improve the training efficiency and generalization power, we conduct the following augmentations:

- Addition of Gaussian noise and Perlin noise
- Randomly remove some groups of curves from the data pair
- Random color jittering
- Random horizontal/vertical flip (the order of ground truth groups will be changed accordingly, and the control points of curves will be changed accordingly also)
- Random 2D translation (the control points of curves will be changed accordingly)

Note that we opt not to use random cropping and resizing since they could lead to corrupted topology (discontinuation of curves) and aliasing issues.

C. Network architecture

C.1. Curve Feature Aggregator

We use 12 residual blocks with instance normalization for the ResNet-backbone [3] in Curve Feature Aggregator. As for the Transformer part, we use 6 Transformer encoder blocks and 6 Transformer decoder blocks with layer normalization and 8 heads of multi-head attention [7]. We set the dimension of the Transformer $d_{transformer} = 128$, the dimension of the feed-forward network $d_{feedforward} = 256$. We use a 3-layer FFN with a hidden dimension of 256 to process the Transformer outputs.

We use the standard U-Net [5] architecture with 5 downsampling layers and 5 upsampling layers for Visual Form Reconstructor. We replace the first convolution layer with the CoordConv layer [4] and use instance normalization after the U-Net convolutions. A residual block without normalization layer and a 1x1 convolution layer is used for converting the feature map from the U-Net output to the raster visual form output.

C.2. Semantic Extractor

The feature map from Visual Form Reconstructor is converted by a Transformer network in an auto-regressive manner. We use 8 Transformer decoder blocks with layer normalization and 8 heads of multi-head attention for the Semantic Extractor. We set the dimension of the Transformer $d_{transformer} = 64$, the dimension of the inner feed-forward network $d_{feedforward} = 128$. We use two 3-layer FFN with the hidden dimensionality of 128 to perform the conversion between the Transformer features and the primitives.

D. Hyperparameters

The hyperparameters used in each loss function are listed as follows.

The curve feature loss

$$\mathcal{L}_e\left(P_i, \hat{P}_i\right) = (1-\lambda_e)\left\|P_i - \hat{P}_i\right\|_2^2 + \lambda_e\left\|P_i - \hat{P}_i\right\|_1 \tag{1}$$

$$\mathcal{L}_{confid}(p_i, \hat{p}_i) = -\hat{p}_i \log p_i - (1-\hat{p}_i)\log(1-p_i) \tag{2}$$

$$\mathcal{L}_F = \frac{1}{n_{curve}} \sum_{i=1}^{n_{curve}} (\beta \mathcal{L}_e + \mathcal{L}_{confid}) \tag{3}$$

$\lambda_e = 0.5$, $\beta = \sqrt{\max(imageWidth, imageHeight)}$, n_{curve} is the total number of dashed curves in the input image.

The visual form loss

$$\mathcal{L}_V(L_{gt}, L_r) = (1-\lambda_v)\|L_{gt} - L_r\|_2^2 + \lambda_v\|L_{gt} - L_r\|_1 \tag{4}$$

$\lambda_v = 0.3$

The continuity loss

$$\mathcal{L}_{eos}(eos_i, \widehat{eos}_i) = -\widehat{eos}_i \log eos_i - (1-\widehat{eos}_i)\log(1-eos_i) \tag{5}$$

$$\mathcal{L}_{primitive}\left(\theta_i, \hat{\theta}_i\right) = (1-\lambda_p)\left\|\theta_i - \hat{\theta}_i\right\|_2^2 + \lambda_p\left\|\theta_i - \hat{\theta}_i\right\|_1 \tag{6}$$

$$\mathcal{L}_C = \frac{1}{n_{primitive}} \sum_{i=1}^{n_{primitive}} (\beta \mathcal{L}_{primitive} + \mathcal{L}_{eos}) \tag{7}$$

$\lambda_p = 0.5$, $\beta = \sqrt{\max(imageWidth, imageHeight)}$, $n_{primitive}$ is the total number of primitives belonging to the semantic curve $\mathcal{SC}_k$.

The overall loss

$$\mathcal{L} = \mathcal{L}_F + \frac{1}{n_{curve}}(\lambda_V \mathcal{L}_V + \lambda_C \mathcal{L}_C) \tag{8}$$

$\lambda_V = 3.5$, $\lambda_C = 0.5$, n_{curve} is the total number of dashed curves in the input image.

E. Application: Dashed curve style editing

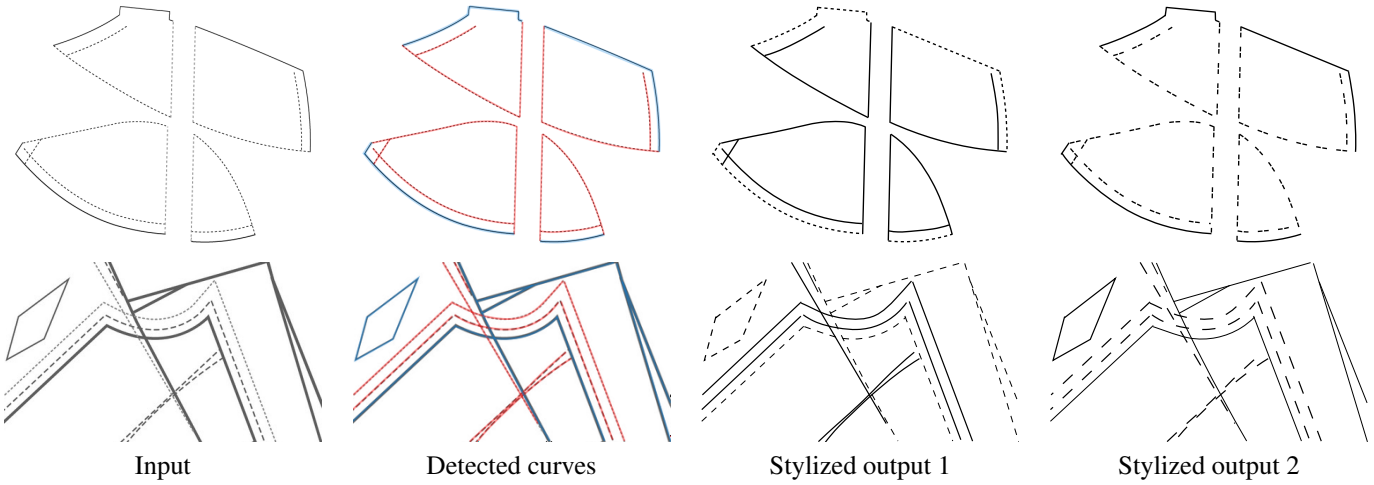


Figure 1. Style alternation based on semantic curve understanding.

With post-processing, our framework can be used as a dashed curve style editor for technical line drawing and sewing design patterns. We demonstrate two use cases of our framework for: inverting the solid and dashed curves and modifying dash style in Figure 1.

F. Experimental results with noisy inputs

F.1. Random degradation

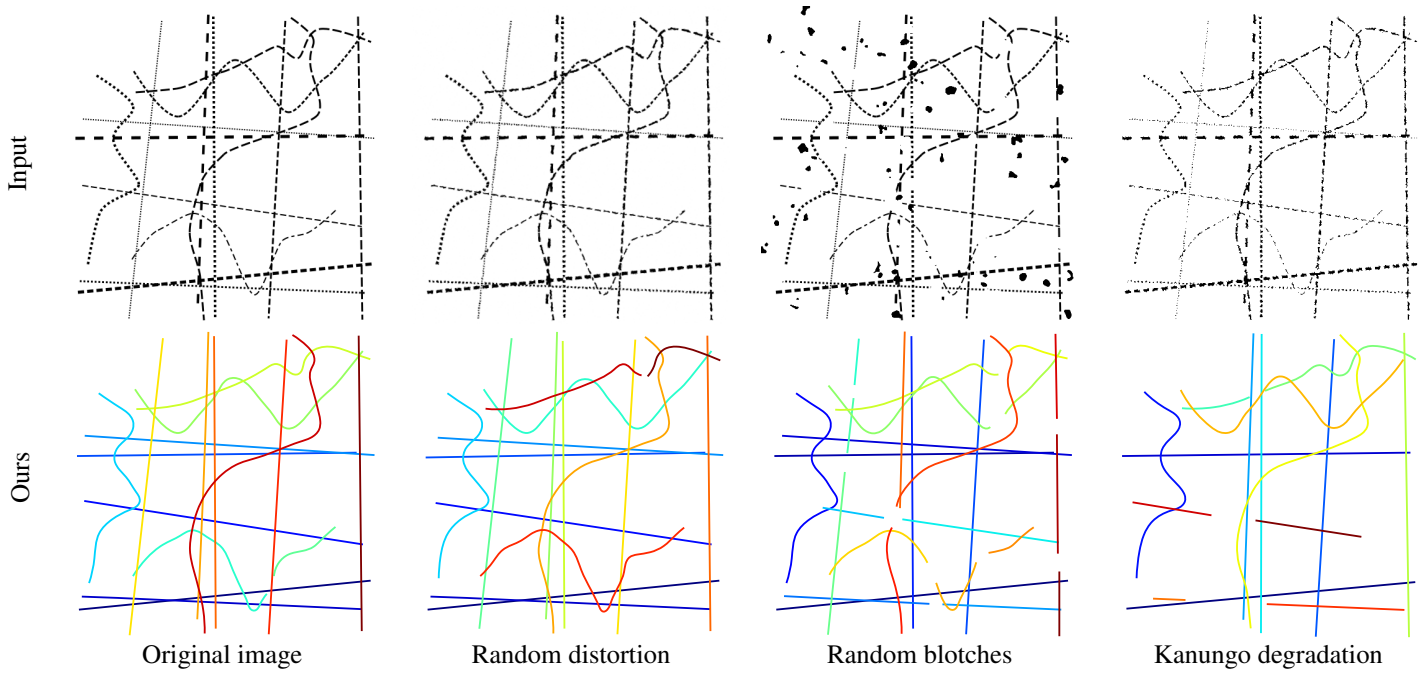


Figure 2. Experimental results with random degradation.

F.2. Addition of Gaussian noise

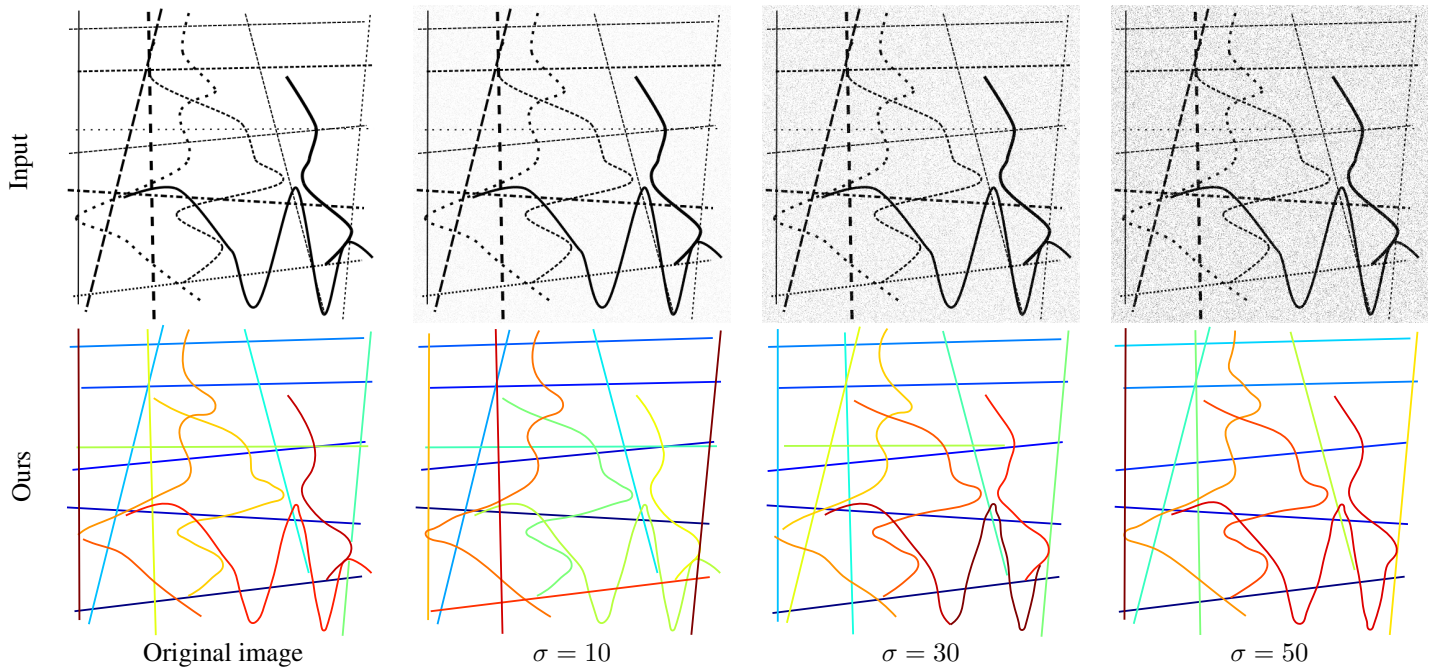


Figure 3. Experimental results with addition of Gaussian noise.

F.3. Failed cases

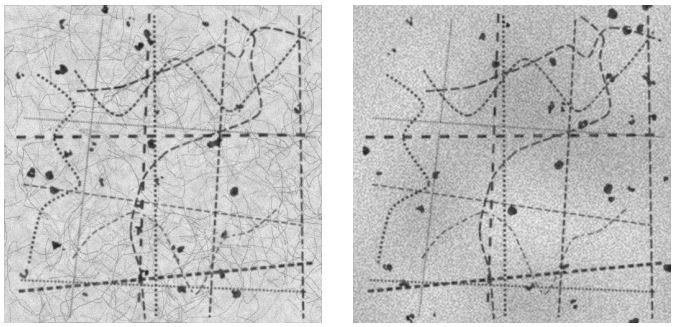


Figure 4. Failed cases.

Our current framework fails to generalize on inputs that are degraded with unseen types of noise, *e.g.* fibrous noise (4a) and multi-scale noise (4b). Training with related data augmentations might improve the generalization ability.

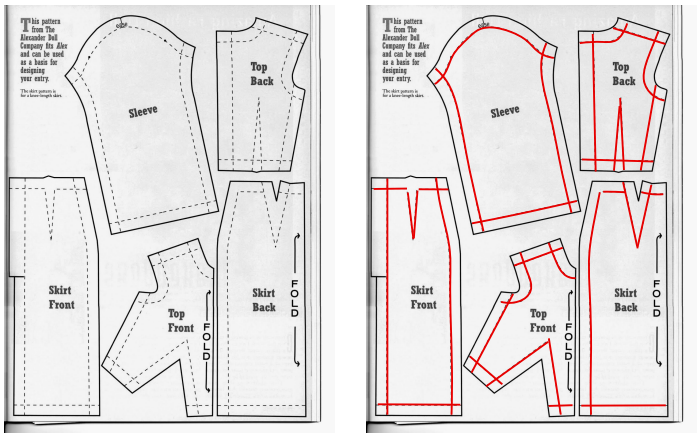
G. Dashed curve recognition on real-world inputs

Our framework can benefit downstream tasks which require the correct semantics of dashed curves. We show examples in the following subsections.

G.1. Stitch line parsing from sewing patterns

In sewing patterns, designers use dashed curves to indicate where the fabric pieces will be stitched together. Our framework can parse stitch lines from sewing patterns and thereby providing correct semantics and labels for garment stitches. In

that case, our framework can serve as a component of sewing pattern vectorization toolkit [2] and provide better topology semantics and stitch labels for garment generation from sewing pattern images [6].

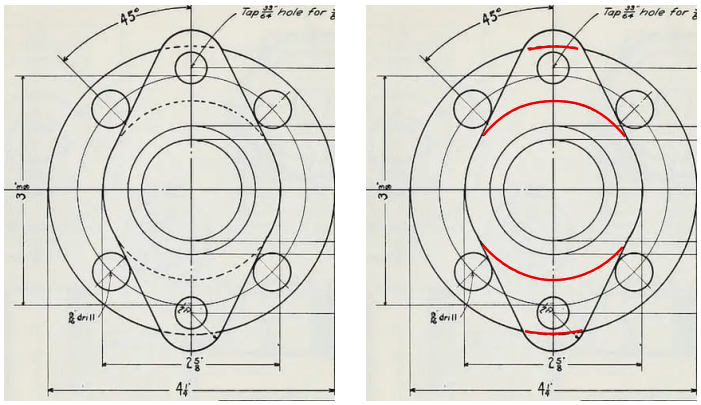


(a) Input (b) Detected stitch lines

Figure 5. Stitch line parsing from sewing pattern images.

G.2. Hidden edge detection from technical drawings

In technical line drawings, it is standard practice to use dashed lines or dashed curves to represent any edges of an object that is hidden from the current view. In this way, dashed curves carry depth information of certain edges of an object. Either vectorizing a dashed curve as separate curve fragments or completing a dashed curve in raster space would lead to missing curve semantics or the loss of depth information respectively. Our framework can help recognize hidden edges and vectorize them with intact curve semantics in technical drawings. With correct semantics of hidden edges, our framework can facilitate 3D reconstruction of objects from single 2D technical drawings [8].

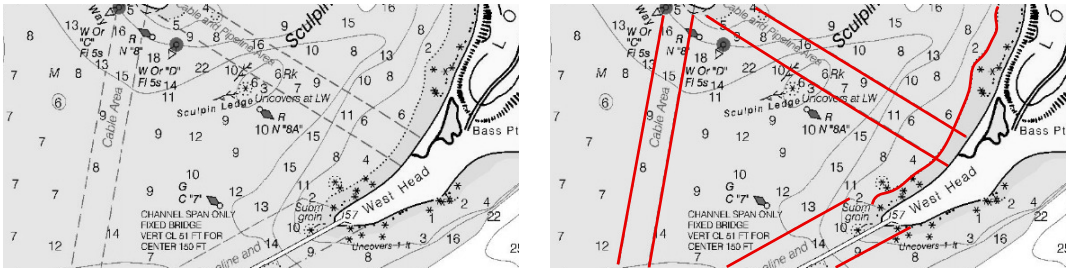


(a) Input (b) Detected hidden edges

Figure 6. Hidden edge detection from technical drawings.

G.3. Digitization of nautical charts

Dashed curves mark channel sides in nautical charts. Our framework can recognize these marks from nautical chart with continuous semantics and facilitate the digitization of nautical charts.



(a) Input (b) Detected dashed curves

Figure 7. Detected dashed curves on a nautical chart. The recognition results are obtained from a fine-tuned version of our framework.

References

[1] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. 1

[2] Joost De Cock. Freesewing. https://freesewing.org/, 2016. 5

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society. 1

[4] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, 2018. 1

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing. 1

[6] Yu Shen, Junbang Liang, and Ming C Lin. Gan-based garment generation using sewing pattern images. In *European Conference on Computer Vision*, pages 225–247. Springer, 2020. 5

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1

[8] Yingze Wang, Yu Chen, Jianzhuang Liu, and Xiaoou Tang. 3d reconstruction of curved objects from single 2d line drawings. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1834–1841. IEEE, 2009. 5