Neural Recognition of Dashed Curves with Gestalt Law of Continuity

Hanyuan Liu[1] Chengze Li[2] Xueting Liu[2] Tien-Tsin Wong[1*]
[1] The Chinese University of Hong Kong [2] Caritas Institute of Higher Education
{liuhy, ttwong}@cse.cuhk.edu.hk {czli, tliu}@cihe.edu.hk

Abstract

Dashed curve is a frequently used curve form and is widely used in various drawing and illustration applications. While humans can intuitively recognize dashed curves from disjoint curve segments based on the law of continuity in Gestalt psychology, it is extremely difficult for computers to model the Gestalt law of continuity and recognize the dashed curves since high-level semantic understanding is needed for this task. The various appearances and styles of the dashed curves posed on a potentially noisy background further complicate the task. In this paper, we propose an innovative Transformer-based framework to recognize dashed curves based on both high-level features and low-level clues. The framework manages to learn the computational analogy of the Gestalt Law in various domains to locate and extract instances of dashed curves in both raster and vector representations. Qualitative and quantitative evaluations demonstrate the efficiency and robustness of our framework over all existing solutions.

1. Introduction

Dashed curve is a frequently used curve form, where the curve is made up of a series of disjoint but continuous curve segments. Dashed curves are widely used in various drawing and illustration applications, such as technical line drawings, graphic designs, fashion designs, paper folding illustrations, etc. While humans can intuitively recognize dashed curves though they are broken into disjoint segments, it is extremely difficult for computers to recognize the broken segments as one semantic curve as high-level semantic understanding is needed for this recognition task. There are three major challenges. Firstly, different dashed curves may be composed of solid curve segments in different lengths, widths, and composition patterns, as shown in Fig. 1a. Secondly, multiple dashed curves might intersect with each other, which complicates recognition of individual dashed curves. Thirdly, the background of the drawings and illustrations might be noisy due to the quality loss in the scanning or photo capturing process, which will further make the identification of the curve segments non-trivial, especially when the curve segments are of short lengths.

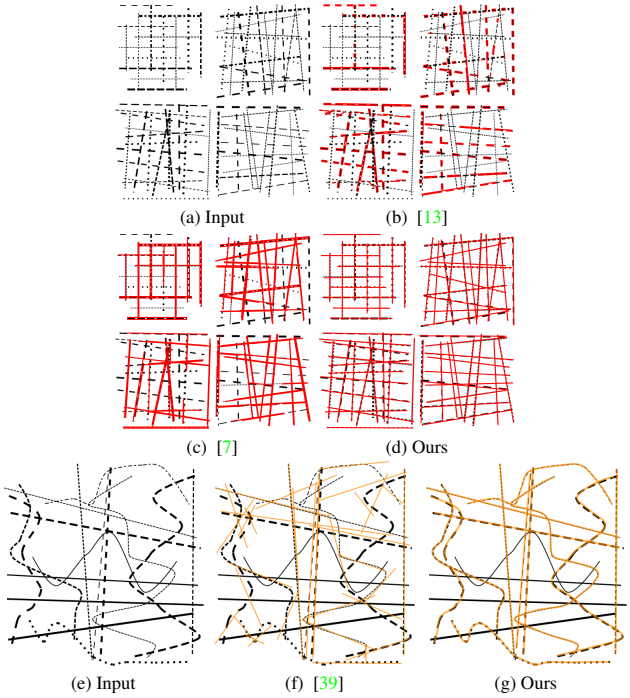

Figure 1. Dashed curve recognition results.

In order to recognize dashed curves from illustrations or graphic designs, several traditional methods have been proposed based on manually designed heuristics [1, 10, 18, 24]. Due to the limitation of human-designed heuristics, these methods are generally restricted to solve only dashed curves of relatively simple and pre-defined shapes and styles and easily get failed when the input becomes complex, as shown in Fig. 1b. A workshop was held in 1995 to encourage research efforts in solving this challenging problem [30]. Unfortunately, it did not gain much success at that time due to the limitations of traditional heuristic methods. Later, methods have been proposed in detecting lines for 3D point

*Corresponding author.

clouds, which share some similarities as the dashed curve recognition task. However, these methods are still based on heuristics and make assumptions on the appearance of the points. Directly applying these methods on dashed illustrations containing dashed curves of different curve styles cannot obtain satisfying results, as shown in Fig. 1c. Recently, deep learning based approaches [27,39] have been proposed in line drawing vectorization and contour detection in natural photographs, but these methods still cannot be directly applied in our task, even if the models are re-trained with dashed illustrations, as shown in Fig. 1f.

We point out that either low-level clues or high-level features alone cannot achieve satisfying dashed curve recognition results. With low-level clues only, such as tangent directions, the computer may easily get confused at junctions or complicated regions. On the other hand, high-level features are too rough to recognize each individual curve, especially when the curves are broken into disjoint short curve segments. Therefore, in this paper, we proposed an innovative approach that combines high-level features and low-level clues to recognize the dash curves and output the precise raster and vector forms of the curves via a learning-based approach.

Our method is designed based on the idea of Gestalt psychology, which suggests that humans can recognize dashed curves from broken curve segments based on the law of continuity [38]. This inspires us to first construct a high-level understanding of the dashed curves and then use this understanding to facilitate the extraction and vectorization of the curves. In particular, we propose to learn the Gestalt law of continuity of a dashed drawing using a deep neural network. This network will aggregate the curve segments into a list of curve descriptors, which contain the high-level embedding of the dashed curves. However, the aggregated curve descriptors only contain high-level rough curve representations, which cannot be directly visualized. We further propose another deep learning network to construct the pixel-level dashed curves from the high-level curve descriptors. These pixel-level dashed curves are raster representations and still lack the notion of continuity. So, we propose the third deep learning network to predict the analytic form of the recognized dashed curves. The three networks are trained together to enable end-to-end training and error propagation. Our method can be applied on dashed illustrations that contain dashed curves of different widths, lengths, shapes, and styles. Satisfying results are obtained in all experiments.

The major contributions of our framework can be concluded as follows:

- We propose an end-to-end framework that can recognize and extract dashed curves based on their semantic meaning with high accuracy and robustness.
- We learn to apply the Gestalt law of grouping and continuity as an explicit constraint in our framework.
- We conduct the learning of the Gestalt law in the feature, raster, and vector domain simultaneously with specially crafted supervision.

2. Related work

2.1. Perceptual Gestalt grouping

Gestalt psychology is a stream of theory on how humans perceive a collection of elements as an entire object [20,28]. The well-known Gestalt principles by Wertheimer [38] reflect how the human visual system group elements into forms. Whenever collections of visual element have one or several common characteristics, they shall get grouped and form a new larger visual object - a *Gestalt*. Psychologists have been trying to simulate the Gestalt principle by finding a computational methodology [9] to predict what humans perceive as Gestalt in images. In this paper, we propose to introduce the Gestalt principles into the field of learning-based computer vision, by guiding the neural networks to learn the grouping and recognition of similar visual elements in both the feature space and the vector space.

2.2. Dashed line and curve detection

The problem of dashed line detection has been raised in the early years of computer graphics research [10]. Correct recognition of dashed lines is essential for parsing various types of media, such as sewing and designing pattern books, nautical charts, and technical line drawings. Early pioneering works relied on predefined features in the vector domain and manually designed heuristics [1,10,23]. These methods usually fail in unexpected cases. In 1995, a dashed line detection contest [30] was held during the First International Workshop on Graphics Recognition to promote the advance in dashed line detection algorithms. Later on, a series of dashed line methods based on Hough transform [18, 22] was proposed, in which the line predictions are gathered from the parameter space. However, due to the limitations of their heuristic designs, these approaches often produce suboptimal results.

In the meanwhile, the vision community has been working on the task of line segment detection from natural images [39]. Recently, the emergence of deep learning has refreshed the state-of-the-art to replace the classical heuristics and Hough-based approaches. These methods either leverage junction analysis [40] to convert junction into line proposals or employ dense prediction [17] to generate a surrogate representation and then extract line segments from the representation. A recent work, LETR [39], uses the Transformer-based object detection technique [4] to achieve end-to-end line segment detection without complicated junction analysis or surrogate maps. Even though these approaches manage to extract straight line segments,

they are unable to understand the *Gestalt Law of continuity and grouping* of dashed line segments and group the segments on the same semantic line. What is more, the existing works are generally designed for straight line segments. To the best of our knowledge, no current work can be generalized to cope with curve segments.

On the other hand, instance segmentation methods such as Mask R-CNN [15] can somehow learn to group dashed line segments via raster mask prediction. However, these methods usually rely on region proposals on the feature domain, which may be interfered by line segments with different semantic meanings, which may share similar visual features as the one to be detected. These methods also achieve low quality in recognizing curved line segments. Curve fitting and vectorization [11, 14, 21, 27, 32] is another research stream that is able to convert complex raster graphics into vector format faithfully. However, most of the curve fitting frameworks only consider solid curves as inputs and will treat disjoint curve segments from one single dashed curve as a list of independent curves instead of a single curve. To handle disjoint curve segments, [12] and [2] tried to simply the disjoint curves in the vector space. However, both methods cannot directly handle dashed curves in raster representations. [11] used a U-Net [31] model to complete dashed curves into solid curves, which could be further vectorized. However, their completion model is not well generalized to handle dashed curves of different appearances and styles. Furthermore, the output of this model is still a raster image, which still needs an extra curve fitting to acquire the analytical vector form. In contrast, our method can directly take dashed curve images as inputs and output detected dashed curves in both raster and vector forms in an end-to-end manner.

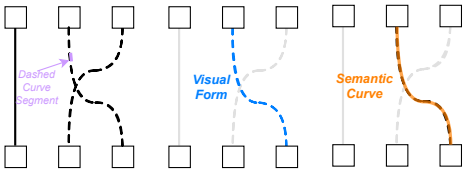

Figure 2. Terminology for dashed curve recognition.

Before we go into the technical details, we would like to define the terminology used in this paper. As shown in Figure 2, each dashed curve is composed of disjointed curve segments, *i.e. dashed curve segment*. Even though these segments do not contact each other physically, we still perceive these segments as an entire group and a mentally continuous element. Here we denote the collection of raster segments from one dashed curve $\mathcal{C}$ as the *Visual Form*, *i.e.* the exact visual stimulus of the dashed curve $\mathcal{C}$. We denote the mentally connected structural line of $\mathcal{C}$ in the analytical vector space as the *Semantic Curve*.

3. Method

3.1. Overview

Our proposed framework aims to achieve the learning of the Gestalt law of continuity with deep learning. As illustrated in Figure 3, the framework contains three modules: the *Curve Feature Aggregator*, the *Visual Form Reconstructor*, and the *Semantic Extractor*. The Curve Feature Aggregator learns to sample and construct a high-level understanding of distinct semantic curves into a list of curve descriptors $\mathcal{D}_i = \{emb_i, P_{i,0}, P_{i,1}\}$, where $\{P_{i,j}\}$ is the endpoint of a certain dashed curve. After that, the high-level semantics of each dashed curve will be input to the Visual Form Reconstructor to guide the virtual continuation of raster dashed curve segments from the same semantic curve to produce the visual form. Moreover, we enable another Semantic Extractor to generate the continuous vector representation of the semantic curves from the curve descriptors. We supervise the module with vector curve regression to improve the overall framework accuracy in terms of visual form recognition and semantic curve fitting. The three submodules are trained together to enable end-to-end training and error propagation. We shall discuss the detailed design of each module in the following sections.

3.2. Curve Feature Aggregator

As mentioned about the difficulties in recognizing visual forms and their underlying semantic strokes and in the spatial domain, we hence construct the features of visual forms with deep neural networks and attempt to learn the feature-level Gestalt law of continuity to distinguish different visual forms in the feature space. Specifically, we aim to design the features with: 1) ability to retain globally (overall trajectory) and locally (local continuity and connectivity) from their compositions; 2) full feature estimation of a visual form from its beginning to its end to minimize ambiguity and information loss. We design our Curve Feature Aggregator module with a vision Transformer architecture [4, 11, 35]. It first converts the input raster image of dashed curves into the deep feature space and partitions distinct semantic curves in the feature domain. After the partitioning, the module outputs a list of curve descriptors $\{CD_i\}$ to represent the features of an individual deemed visual form. The module also aims to regress the deemed visual form endpoints to narrow down the difficulties in visual form recognition and maximize the coverage in feature extraction. The regression of endpoints is also beneficial to the following visual form reconstruction and semantic curve fitting modules.

Given an input image $I \in [0,1]^{w \times h}$, we first obtain an image feature map $X \in \mathbb{R}^{W \times H \times C}$ from a ResNet backbone [16]. To make the Transformer aware of spatial information of the feature map, we concatenate X with a 2D po-

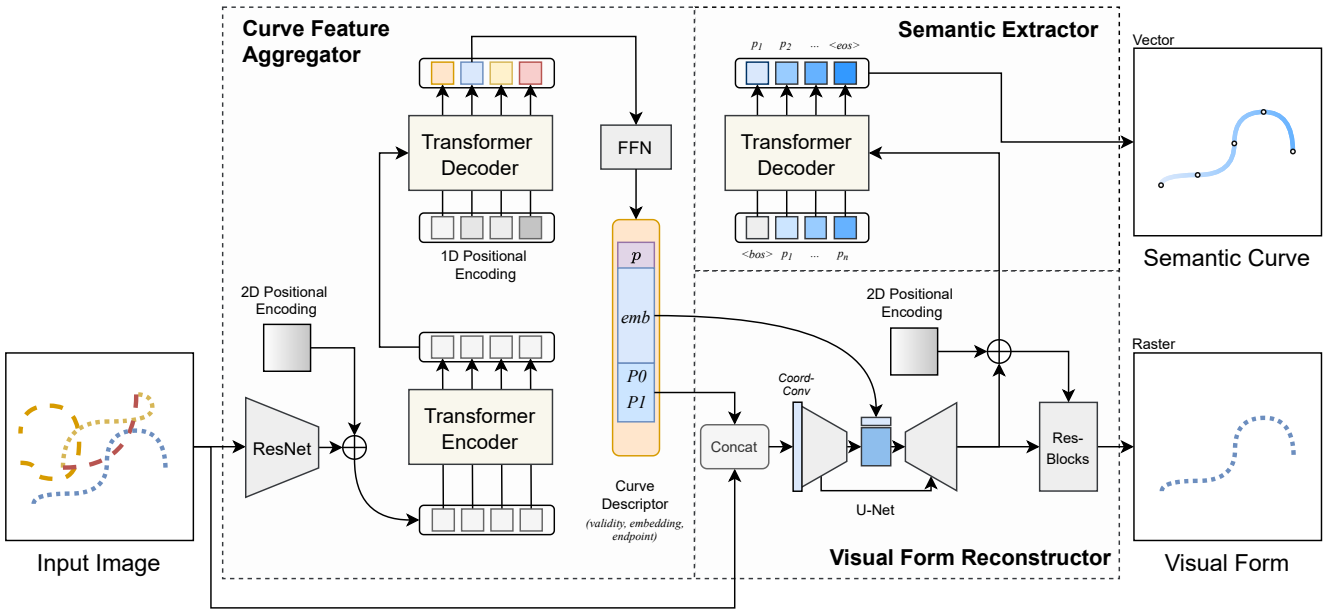

Figure 3. Overview. Given an input image of dashed curves, the Curve Feature Aggregator recognizes dashed curves and converts them into a list of Curve Descriptors; the Visual Form Reconstructor reconstructs the raster-level visual form for dashed curve from the curve descriptor; the Semantic Extractor generate the vectorized semantic curve autoregressively.

sitional encoding $PE \in \mathbb{R}^{W \times H \times d_p}$ and then flatten it into $X \in \mathbb{R}^{WH \times C'}$ where $C' = C + d_p$. We then leverage self-attention from Transformer encoder to get a processed feature map X_f. Given the processed feature map X_f, we design the Transformer decoder to read from the raster image features and output a list of visual form descriptors to represent the semantics of individual visual forms. To achieve so, the decoder decodes X_f with 1D positional encoding PE_{1d} into a list of curve descriptor $\{CD_i\}$ as:

$$TransEmb = \text{TransformerDecoder}(PE_{1d}, X_f), \quad (1)$$

$$\{CD_i\}_N = \text{MLP}(TransEmb) \quad (2)$$

where $TransEmb$ denotes the direct output embeddings from the Transformer decoder.

To enable feature encoding of an uncertain number of visual forms, we design the maximum length of the output sequence to be much larger than the actual number of visual forms in typical cases. We use a classification head [4, 15, 39] to predict if the output curve descriptor is valid and discard invalid features. The maximum output number is set to the size of the input positional embedding $PE_{1d} \in \mathbb{R}^{N \times d_{Trans}}$. On the other hand, to capture complete visual form features, we explicitly use another regression head for each entry of the extracted feature embeddings to predict the location of visual form endpoints. The regression will provide strong evidence to enlighten the feature extraction to be constrained by endpoint locations. However, direct endpoint sequence regression may be burdened by ordering ambiguities that come from the ordering of $(starting\_point, ending\_point)$ within a prediction and also from the ordering of curves across predictions with the same endpoints. The bipartite matching used in common Transformer detection models [4, 39] cannot mitigate this problem, as the Hungarian matching of endpoints cannot distinguish both cases. We opt to leverage the sorting capability of Transformers [37] to make the ordering sorted and supervise the output list of endpoints with the ground truth endpoints sorted in the same manner as [5, 11].

The curve feature loss We employ the curve feature loss to supervise the multitask learning of semantic curve descriptor validation and endpoint regression. The semantic curve description validation objective is a binary cross-entropy classification loss, while the endpoint regression objective is a combination of L1 loss and L2 loss. The overall loss is a weighted combination of two objectives:

$$\mathcal{L}_e\left(P_i, \hat{P}_i\right) = (1-\lambda_e)\left\|P_i - \hat{P}_i\right\|_2^2 + \lambda_e\left\|P_i - \hat{P}_i\right\|_1 \quad (3)$$

$$\mathcal{L}_{confid}(p_i, \hat{p}_i) = -\hat{p}_i \log p_i - (1-\hat{p}_i)\log(1-p_i) \quad (4)$$

$$\mathcal{L}_F = \frac{1}{n_{curve}} \sum_{i=1}^{n_{curve}} (\beta \mathcal{L}_e + \mathcal{L}_{confid}), \quad (5)$$

The target confidence values p_i are binary, where zeros indicate invalid predictions [5, 11]. Note that the transformed embedding emb of curve descriptor CD_i is not supervised with this Curve feature loss. It will be passed to the following visual form and semantic curve reconstruc-

tion stages for finer-grain supervision in raster and vector spaces.

3.3. Visual Form Reconstructor

We propose the Visual Form Reconstructor to learn the Gestalt law of grouping to cluster all segments from the same underlying semantic curve and reconstruct the visual form in the raster space. The module reads the high-level features of visual forms obtained in the Curve Feature Aggregator and gradually refines the features with the assistance of the underlying image details to achieve precise pixel-level segmentation of visual form instances.

We employ the exemplary conditional U-Net [31] architecture to implement the task. The network takes the original raster image of the dashed curves as input and uses one of the curve descriptor entries encoded in the Curve Feature Aggregator as the condition to reconstruct the raster visual form of the given curve descriptor. The U-Net starts from the concatenation of the original image and the endpoint locations. We repeat the 1x4 endpoint vector in the horizontal and vertical direction to form a 4-channel map with the same size as the input image. To make the network aware of the spatial location indicated by the endpoints, we employ CoordConv [25] in the first convolution layer of the U-Net to break the shift-invariance and enforce the network to focus on the endpoint locations. After that, the U-Net takes several levels of convolution and downscaling layers to create image features $DF(\cdot)$. Subsequently, we concatenate the curve embedding emb with $DF(\cdot)$ in the middle of the U-Net to serve as the other condition of reconstruction. The concatenated features will be gradually decoded and refined by a set of upscaling layers incorporating low- and mid-level features propagated from the downscaling U-Net layers. Finally, the last layer of U-Net features will be decoded with a ResNet block to create the final output of the reconstructed raster visual form from the conditions.

The visual form loss This objective is designed to supervise the Visual Form Reconstructor to learn the raster-level Gestalt law of continuity and, in the meantime, supervise the Curve Feature Aggregator to bring more informative high-level features to the raster space. We thereby formulate the visual form loss as below:

$$\mathcal{L}_V(L_{gt}, L_r) = (1-\lambda_v)\|L_{gt} - L_r\|_2^2 + \lambda_v\|L_{gt} - L_r\|_1 \quad (6)$$

3.4. Semantic Extractor

Even if we have constructed the Curve Feature Aggregator and the Visual Form Reconstructor module to learn the Gestalt rules, they are lacking the specific notion of *continuity*. We find that the Visual Form Reconstructor is mainly supervised on the notion of *grouping*, but does not explicitly supervise the semantic curve and which may cause drift on the raster-level prediction of visual forms, as it lacks the understanding of the global semantic curve trajectory. To enhance the learning of the Gestalt law of continuity, we add a Semantic Extractor module to learn the Gestalt Law of continuity and predict the analytic form of semantic curves in the vector domain. The Semantic Extractor is a Transformer decoder, which reads from the mixed representation of features of a given visual form in the late layers of Visual Form Reconstructor and decodes the primitives of a continuous Bézier representation of the semantic curve in an autoregressive manner. That is, we implement a virtual pen moving along the dashed curve and progressively output its vector trajectory with the assistance of visual form and image features. The autoregressive decoding of stokes is proven to be efficient in continuous line drawing such as [3, 8, 27].

The formulation of the Semantic Extractor is defined as:

$$primitive_t, eos = \text{Transformer}(primitive_{0:t-1}, \mathcal{M}) \quad (7)$$

where $primitive_t$ is the t-th Bezier primitive from the semantic curve, eos is the *end of sequence* token, and $\mathcal{M}$ is the feature map generated by the last layer of U-Net from Visual Form Reconstructor. The network progressively generates a sequence of primitives and stops when the prediction of eos is larger than 0.5. Given the fact that $primitive_t$ and $primitive_{t-1}$ are connected, we omit the starting point of each primitive to remove redundancy:

$$primitive_t = (x_{c1}, y_{c1}, x_{c2}, y_{c2}, x_e, y_e)_t \quad (8)$$

We found this can also help enforce the continuity of generated results. For a dashed line $\text{Line}(x_s, y_s, x_e, y_e)$, we represent it as a special case of Bézier curve $\text{CubicBezier}(x_s, y_s, x_s, y_s, x_e, y_e, x_e, y_e)$ by setting control points as its endpoints.

The continuity loss We train the Semantic Extractor module with a continuity loss to supervise the whole output sequence to be close to the ground truth semantic curve vector. The continuity loss shares the similar multitask idea as the Curve feature loss, with a cross-entropy loss to predict and supervise the occurrence of the eos token only at the end of the prediction and a combination of L_1 and L_2 loss to regress the control points of each Bézier primitive. The loss is formulated as:

$$\mathcal{L}_{eos}(eos_i, \widehat{eos}_i) = -\widehat{eos}_i \log eos_i - (1-\widehat{eos}_i)\log(1-eos_i) \quad (9)$$

$$\mathcal{L}_p\left(\theta_i, \hat{\theta}_i\right) = (1-\lambda_p)\left\|\theta_i - \hat{\theta}_i\right\|_2^2 + \lambda_p\left\|\theta_i - \hat{\theta}_i\right\|_1 \quad (10)$$

$$\mathcal{L}_C = \frac{1}{n_{primitive}} \sum_{i=1}^{n_{primitive}} (\beta\mathcal{L}_p + \mathcal{L}_{eos}) \quad (11)$$

3.5. Overall loss function

In summary, our training loss function is made up of three components: (1) a curve feature $\mathcal{L}_F$ for dashed curve

entity recognition and detection, (2) a visual form reconstruction loss $\mathcal{L}_V$ for visual stimulus supervision, and (3) a continuity loss $\mathcal{L}_C$ to extract semantic curves and enforce the continuity constraints. The final loss is formulated as follows:

$$\mathcal{L} = \mathcal{L}_F + \frac{1}{n_{curve}}(\lambda_V \mathcal{L}_V + \lambda_C \mathcal{L}_C) \tag{12}$$

where n_{curve} is the number of the dashed curves on the input image, $\lambda_V = 3.5$ and $\lambda_C = 0.5$. We demonstrate the effectiveness of each loss term in the ablation study (section 4.4).

4. Experiments

4.1. Implementation and training

4.1.1 Framework implementation

We implement our framework in PyTorch 1.8 with its shipped Transformer module. We use six encoder-decoder layers for the Curve Feature Aggregator and eight decoder layers for the Semantic Extractor. Both of the Transformers use eight attention heads. For Curve Feature Aggregator, the 1D sinusoidal positional encoding [35] is used for the parallel decoding scheme [4, 11]. We used a joint training scheme while carefully designing the learning process for each module. Please refer to the supplemental material for the training details, such as hyperparameters and detailed architecture.

Processing input with solid and dashed curves. We point out that ambiguity in understanding leads to a recognition contradiction if solid and dashed curves are recognized simultaneously. A dashed curve can be recognized as one dashed curve or multiple solid curves, which are both semantically correct. So, the framework may encounter recognition ambiguity if solid and dashed curves are trained together. To tackle the ambiguity, we employ a two-step solution:

1. We train our framework only to detect dashed curves and remove the dashed curves with a U-Net based dash curve removing network [11, 31].
2. We then train another *identical* framework (same as the dashed curve recognition framework) only to detect solid curves.

In this manner, we avoid the ambiguity in recognizing dashed curves and solid curves at the same time to improve the overall accuracy and efficiency.

4.1.2 Dataset

We generate a synthetic dataset for training and validation since no existing dashed curve dataset appears in the current literature. For the evaluation of real-world examples, we collect sewing patterns, clothing pattern plans, and graphic designs that contain dashed curves. We annotate each instance of visual forms manually on the dataset. We show some data samples in Figure 4.

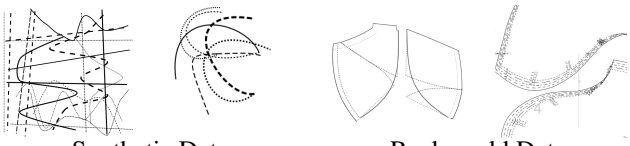

Figure 4. Sample images of our dataset.

We use the Canvas 2D API to create the synthetic dataset of dashed curve drawings and ground truth labels, including the visual form subdivision and the analytical semantic curve representation. We implement the data generator with `node-canvas` [19] and `cairo` [29]. The generator renders random straight and curved lines with the dashed pattern using the `setLineDash()` method from the Canvas 2D API. We chose the exemplary line dash patterns used by TikZ [34] and randomly chose 20 different dash patterns for synthesis. To stabilize training and strive for better generalization ability, we opt to design a progressively growing training pool, with a maximum number of 20,000 data samples. During training, we sequentially sample the data from the pool and drop 1000 new samples to the pool for each epoch. We also apply data augmentation to the training and validation dataset by adding perturbations to the colors and line trajectories. We shall describe the detailed augmentation approach in the supplemental material.

4.2. Evaluation metrics

We refer to the dashed line detection contest [30] to highlight the characteristics of a high-quality dashed curve recognition:

- To detect all the dashed lines/curves from the input images;
- To accurately distinguish the instances of visual forms and estimate their pixel locations;
- To track the dashed lines and curves correctly based on their semantics.

We would like to design our evaluation and comparisons based on the above criteria, but unfortunately, none of the existing methods fulfill all of them. As a result, we have to conduct experiments from different perspectives of the visual raster form and the semantic vector form. Firstly, we present a quantitative comparison on the endpoint prediction accuracy with Line Segment Detector (LSD) [13], probabilistic Hough transform based method (HT) [22], and LETR [39]. These methods can only detect lines in the representation of endpoints and cannot recognize visual form instances. We compute the structure-based average precision (sAP) and structure-based F-score (sF) [17,40] in this

experiment. Secondly, we experiment with the pixel-level prediction of visual form instances against the instance segmentation methods [15, 26]. We evaluate the performance with the average IoU-based average precision (mask AP) on all visual form instances. Finally, to compare with those vector-based line and curve detection methods [11, 27], we evaluate the Chamfer distance (CD), Hausdorff distance (HD), and earth mover's distance (EMD) to measure how well the methods trace the dashed curve trajectory in the vector space. The distances are computed based on the dense sampling of the prediction towards the ground truth semantic curve. We want to emphasize again that only our framework can achieve all these three tasks simultaneously to achieve both high-quality raster visual form extraction and vector semantic curve estimation. For fair comparisons, we have retrained [39], [15], and [26] on our synthetic dataset.

4.3. Results and comparisons

Method	sAP^5	sAP^{10}	sAP^{15}	sF^5	sF^{10}	sF^{15}
LETR* [39]	14.4	42.5	61.3	33.6	60.7	75.0
Ours	23.1	55.4	71.3	43.4	70.6	81.2

Table 1. Quantitative comparison against the SOTA dashed line segment detection, LETR [39]. Note that LETR can only detect straight line segments from input images.

Method	bAP_{50}	bAP_{75}	mAP_{50}	mAP_{75}
Mask R-CNN [15]	71.2	57.6	2.0	1.0
Mask R-CNN† [15]	64.9	54.1	5.8	2.1
Swin* [26]	71.6	36.3	1.5	0.0
Swin*† [26]	64.7	35.5	10.1	3.9
Ours	/	/	42.3	26.7

Table 2. Quantitative comparison on the raster visual form extraction. We treat it as a special case of instance segmentation. * indicates state of the art for instance segmentation method. Methods with † are trained and evaluated on rasterized semantic curves (connected visual forms) as ground-truth masks. Here we denote box AP and mask AP by bAP and mAP respectively.

Method	IoU↑	HD↓	CD↓	EMD↓
DVTD [11]	45.2%	50.9	0.0633	**0.399**
GVS* [27]	52.0%	52.0	0.0303	0.439
Ours	**52.2%**	**50.1**	**0.0157**	0.412

Table 3. Quantitative comparison on tracking quality of dashed curves. * indicates state of the art for line drawing vectorization.

Endpoint estimation We demonstrate the visual comparison of line segment detection methods in Figure 5. We can find that HT and LSD can only group a small portion of dashed lines but fail to distinguish solid lines and dashed lines. LETR [39] is able to detect only dashed lines and fail to predict the endpoint precisely. By contrast, our framework can recognize most of dashed line segments can produce accurate endpoint estimation. We highlight our evaluation statistics against LETR in Table 1.

	no $\mathcal{L}_v$	no $\mathcal{L}_c$	no sorting	full method
mAP_{50} ↑	/	12.1	32.7	**42.3**
mAP_{75} ↑	/	6.6	15.5	**26.7**
IoU ↑	41.3%	/	39.1%	**52.2%**
HD ↓	52.9	/	51.3	**50.1**
CD ↓	0.0972	/	0.0565	**0.0157**
EMD ↓	0.912	/	0.673	**0.412**

Table 4. Ablation study on the loss terms and framework design.

Raster visual form evaluation The quantitative evaluation results are shown in Table 2. We find that the representative instance segmentation methods [15, 26] can locate dashed curves via bounding boxes but fail to extract them from the input due to the lack of topology and continuity understanding of dashed complex curves. By contrast, our method achieves much better performance in terms of visual form reconstruction.

Vector semantic curve tracking We demonstrate the results of semantic curve tracking in Figure 6. For a fair comparison against GVS [27], we use the preprocessing techniques from DVTD [11] to complete the dashed curves into solid curves. Both DVTD and GVS fail to predict meaningful semantics of dashed curves, while our framework can recognize better semantics. The quantitative evaluation results are shown in Table 3. Our framework outperforms DVTD and GVS in terms of rasterized IoU, HD, and CD. We also evaluate the R-GVS [27] that is designed for rough sketches. We find it can directly handle dashed curves but generate unsatisfactory results.

4.4. Ablation study

To verify the effectiveness of our framework design, we conduct ablation studies and demonstrate the quantitative ablation results in Table 4.

Sorting mechanism in training the Curve Feature Aggregator. We explicitly sort the ground truth endpoint targets for curve descriptors in the training of the Curve Feature Aggregator. To validate the sorting mechanism, we train an alternative version of the Curve Feature Aggregator that uses bipartite matching [4] for supervision. We find that the endpoint regression performance degrades in this setting, which is conforming to the conclusions drawn in [5]. Furthermore, the curves that share endpoints could result in confusion for training.

Loss terms. We also conduct ablation studies with different loss terms. We find that the curve feature loss is crit-

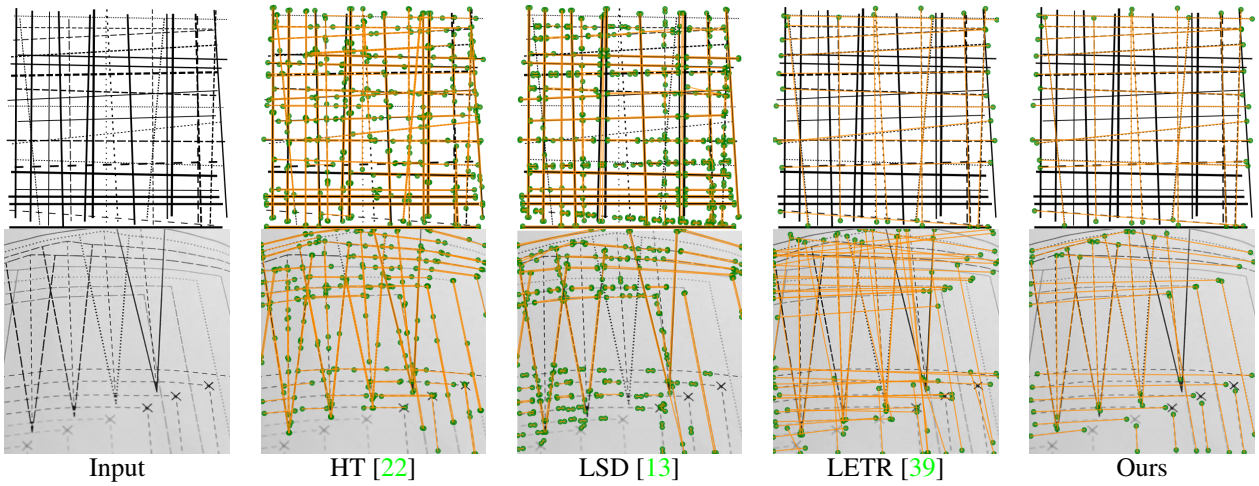

Figure 5. Visual comparison on the extraction of straight visual forms.

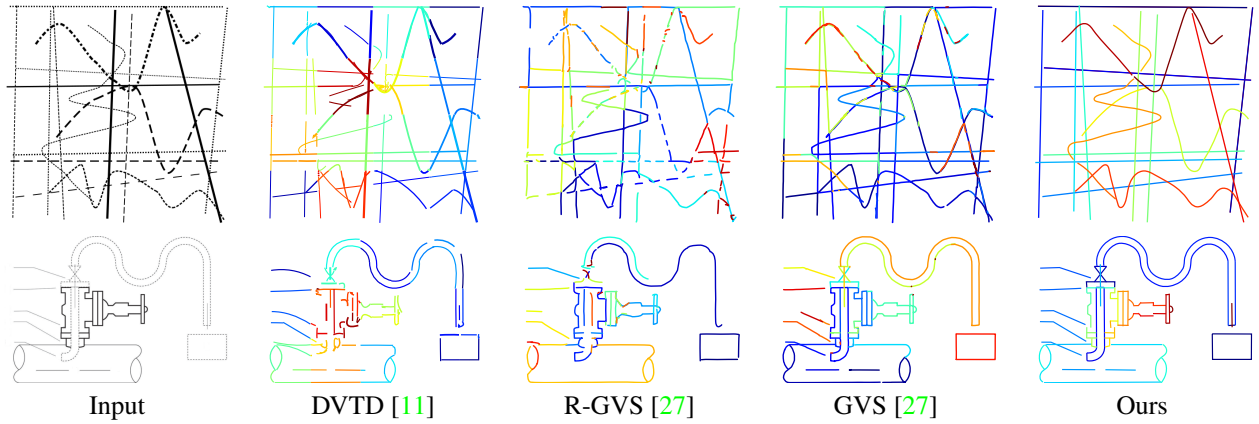

Figure 6. Visual comparison on the recognition and extraction of visual forms with curvature. Curves are visualized with distinct colors.

ical to our framework. Without the curve feature loss our framework will not converge. The related outputs of visual form loss and continuity loss lie in different modalities, so we cannot compare them directly. Compared with the full methods, we can find that both visual form loss and continuity loss can help improve performance.

4.5. Limitations and discussion

The formulation of endpoint regression and curve descriptor limits the scope of our framework to be only within line and curve segment groups, i.e., we can only learn the Gestalt Law on curve-like structures. We shall keep exploring novel solutions to extend the proposed framework to be compatible with the Gestalt Laws on other types of 2D shapes and contents. Additionally, our framework tends to separate double lines, while humans would sometimes treat the double line as a complete entity. It is difficult to imitate this behavior as there is no clear gap between two parallel dash lines and one double line. The recognition ambiguity of double lines remains an open problem. Moreover, our framework strongly relies on the Transformer [35] architecture. The computational cost of our framework is high. We shall investigate the possibility to include efficient attention mechanisms [6, 33, 36] as potential future improvements.

5. Conclusions

In this paper, we present a Transformer-based framework for dashed curve recognition. We employ a universal framework with the gestalt law of grouping and continuity to achieve handling of a vast diversity of discontinued lines, which is proven to be effective by a series of experiments.

Acknowledgement: This project is jointly supported by CUHK Direct Grant for Research (Project No. 4055152) and a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. UGC/FDS11/E02/21).

References

[1] Gady Agam, Huizhu Luo, and Its'hak Dinstein. Morphological approach for dashed lines detection. In Rangachar Kasturi and Karl Tombre, editors, *Graphics Recognition Methods and Applications*, pages 92–105, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. 1, 2

[2] Mikhail Bessmeltsev and Justin Solomon. Vectorization of line drawings via polyvector fields. *ACM Trans. Graph.*, 38(1), Jan. 2019. 3

[3] Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, and Yi-Zhe Song. Vectorization and rasterization: Self-supervised learning for sketch and handwriting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5672–5681, June 2021. 5

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 213–229, Cham, 2020. Springer International Publishing. 2, 3, 4, 6, 7

[5] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. In *Advances in Neural Information Processing Systems*, volume 33, pages 16351–16361. Curran Associates, Inc., 2020. 4, 7

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. 8

[7] Christoph Dalitz, Tilman Schramke, and Manuel Jeltsch. Iterative Hough Transform for Line Detection in 3D Point Clouds. *Image Processing On Line*, 7:184–196, 2017. 1

[8] Ayan Das, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, and Yi-Zhe Song. Cloud2curve: Generation and vectorization of parametric sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7088–7097, June 2021. 5

[9] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. Computational gestalts and perception thresholds. *Journal of Physiology-Paris*, 97(2-3):311–324, 2003. 2

[10] Dov Dori, Liu Wenyin, and Mor Peleg. How to win a dashed line detection contest. In Rangachar Kasturi and Karl Tombre, editors, *Graphics Recognition Methods and Applications*, pages 286–300, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. 1, 2

[11] Vage Egiazarian, Oleg Voynov, Alexey Artemov, Denis Volkhonskiy, Aleksandr Safin, Maria Taktasheva, Denis Zorin, and Evgeny Burnaev. Deep Vectorization of Technical Drawings. In *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 582–598, Cham, 2020. Springer International Publishing. 3, 4, 6, 7, 8

[12] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Fidelity vs. simplicity: A global approach to line drawing vectorization. *ACM Trans. Graph.*, 35(4), July 2016. 3

[13] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010. 1, 6, 8

[14] Yi Guo, Zhuming Zhang, Chu Han, Wenbo Hu, Chengze Li, and Tien-Tsin Wong. Deep Line Drawing Vectorization via Line Subdivision and Topology Reconstruction. *Computer Graphics Forum*, 38(7):81–90, 2019. 3

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 3, 4, 7

[16] 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. 3

[17] Kun Huang, Yifan Wang, Zihan Zhou, Tianjiao Ding, Shenghua Gao, and Yi Ma. Learning to parse wireframes in images of man-made environments. In *CVPR*, June 2018. 2, 6

[18] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988. 1, 2

[19] Automattic Inc. node-canvas. https://github.com/Automattic/node-canvas, 2021. 6

[20] Been Kim, Emily Reif, Martin Wattenberg, Samy Bengio, and Michael C. Mozer. Neural networks trained on natural scenes exhibit gestalt closure. *Computational Brain & Behavior*, 4(3):251–263, Sep 2021. 2

[21] Byungsoo Kim, Oliver Wang, A. Cengiz Öztireli, and Markus Gross. Semantic Segmentation for Line Drawing Vectorization Using Neural Networks. *Computer Graphics Forum*, 37(2):329–338, 2018. 3

[22] Nahum Kiryati, Yuval Eldar, and Alfred M Bruckstein. A probabilistic hough transform. *Pattern recognition*, 24(4):303–316, 1991. 2, 6, 8

[23] Bin Kong, Ihsin T. Phillips, Robert M. Haralick, Arathi Prasad, and Rangachar Kasturi. A benchmark: Performance evaluation of dashed-line detection algorithms. In Rangachar Kasturi and Karl Tombre, editors, *Graphics Recognition Methods and Applications*, pages 270–285, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. 2

[24] CP Lai and R Kasturi. Detection of dashed lines in engineering drawings and maps. In *Proc. First Int. Conf. on Document Analysis and Recognition*, pages 507–515, 1991. 1

[25] 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. 5

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 7

[27] Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. General virtual sketching framework for vector line art. *ACM Trans. Graph.*, 40(4), July 2021. 2, 3, 5, 7, 8

[28] Liangliang Nan, Andrei Sharf, Ke Xie, Tien-Tsin Wong, Oliver Deussen, Daniel Cohen-Or, and Baoquan Chen. Conjoining gestalt rules for abstraction of architectural drawings. *ACM Trans. Graph.*, 30(6):1–10, Dec. 2011. 2

[29] Keith Packard, Carl Worth, and Behdad Esfahbod. Cairo. https://www.cairographics.org/, 2021. 6

[30] Kasturi Rangachar and Prasad Arathi. Dashed-line detection contest. International Workshop on Graphics Recognition, 1995. 1, 2, 6

[31] 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. 3, 5, 6

[32] Peter Selinger. Potrace. http://potrace.sourceforge.net/, 2019. 3

[33] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities, 2020. 8

[34] Till Tantau. Pgf/tikz. https://github.com/pgf-tikz/pgf, 2021. 6

[35] 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. 3, 6, 8

[36] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. 8

[37] Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11080–11090. PMLR, 18–24 Jul 2021. 4

[38] Max Wertheimer. Untersuchungen zur lehre von der gestalt. ii. *Psychologische Forschung*, 4(1):301–350, Jan 1923. 2

[39] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4257–4266, June 2021. 1, 2, 4, 6, 7, 8

[40] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *ICCV 2019*, 2019. 2, 6