Computers & Graphics 132 (2025) 104365

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on CAD/Graphics 2025

LineDrawer: Stroke-level process reconstruction of complex line art based on human perception

Zhengyu Huang [a], Zhongyue Guan [a], Zeyu Wang [a,b,*]

[a] The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, 511453, China
[b] The Hong Kong University of Science and Technology, 999077, Hong Kong, China

ARTICLE INFO

Keywords:
Line art
Stroke-level reconstruction
Drawing process inversion
Deep learning

ABSTRACT

Line art is a fundamental yet powerful form of artistic expression. In this paper, we introduce a novel task aimed at enhancing novice understanding of reproducibility in line drawings: reconstructing the stroke-by-stroke drawing process from complex line art. This task poses substantial challenges, as it requires resolving stroke ambiguity, variations in stroke thickness, and stroke overlapping. To address these issues, we propose a hierarchical framework that emulates human drawing behavior, comprising three stages: (1) high-level generation of global semantic stroke order, (2) mid-level optimization of human drawing mechanics, and (3) low-level perceptual stroke rendering. Drawing inspiration from the human tendency to conceptualize the overall structure before refining local details, we first extract keyframes of the drawing sequence that guide global ordering using a diffusion-based model. Simultaneously, based on the assumption that humans can infer strokes from any cue point in a line drawing, we train a stroke renderer to extract variable-width sub-strokes at the pixel level. Lastly, we formulate a set of equations to model human drawing dynamics, enabling more detailed inference of stroke composition and sequencing within the identified keyframes. This framework effectively integrates high-level semantic understanding with low-level stroke reconstruction, facilitating stroke-level process recovery in complex line drawings. Extensive experiments and user studies demonstrate that our method produces relatively natural and coherent drawing process animations for high-quality line art.

1. Introduction

Artistic line drawing, or line art, is a fundamental form of visual expression in which monochromatic strokes — varying in thickness, curvature, and density — delineate contours, structures, and spatial relationships within a composition. As a highly abstracted representation of visual information, the creation of line art inherently reflects human cognitive strategies for decomposing complex subjects into hierarchical geometric primitives. This intrinsic connection positions the reconstruction of the line art drawing process as an interdisciplinary challenge at the intersection of computer graphics and the study of perceptual mechanisms in artistic cognition. In this paper, we focus on "complex" line arts, defined as high-resolution images characterized by intricate stroke topology and a clean background, without the influence of real-world lighting or shading. A typical example is the line art found in digital illustrations. Despite their apparent simplicity, such drawings are difficult for novices to reproduce. Even when tasked with replicating a given drawing in this style, novices often struggle to determine where to begin, highlighting the need for tools that can reveal and guide the underlying drawing process.

Recent advancements in artificial intelligence have shown promising results in reconstructing the drawing process for paintings from real-world images using stroke-based rendering approaches [1–4] or pixel-based simulation techniques [5–8]. When the input is a line drawing, prior work has primarily focused on stroke ordering [9] after a vectorization process [10–13]. However, these methods encounter critical limitations when applied to "complex line art" due to three intrinsic challenges: (1) The explicit nature of strokes, such as sharp tails and non-uniform widths, necessitates pixel-precise decomposition rather than probabilistic stroke approximation using parameterized brushes [1–3] or fixed-width vector representations [10,12,13]. (2) The presence of overlapping strokes introduces exponential combinatorial ambiguities in topological ordering [9,14]. (3) The absence of color and texture information heightens reliance on geometric reasoning to infer plausible temporal sequences from static inputs [4,8], a challenge that existing methods struggle to address effectively.

Motivated by these gaps, we introduce LineDrawer, a framework that directly infers stroke sequences from raster images of complex line art. In this context, a stroke refers to a continuous line with variable

* Corresponding author.
E-mail address: zeyuwang@ust.hk (Z. Wang).

https://doi.org/10.1016/j.cag.2025.104365
Received 27 June 2025; Received in revised form 27 July 2025; Accepted 29 July 2025
Available online 22 August 2025

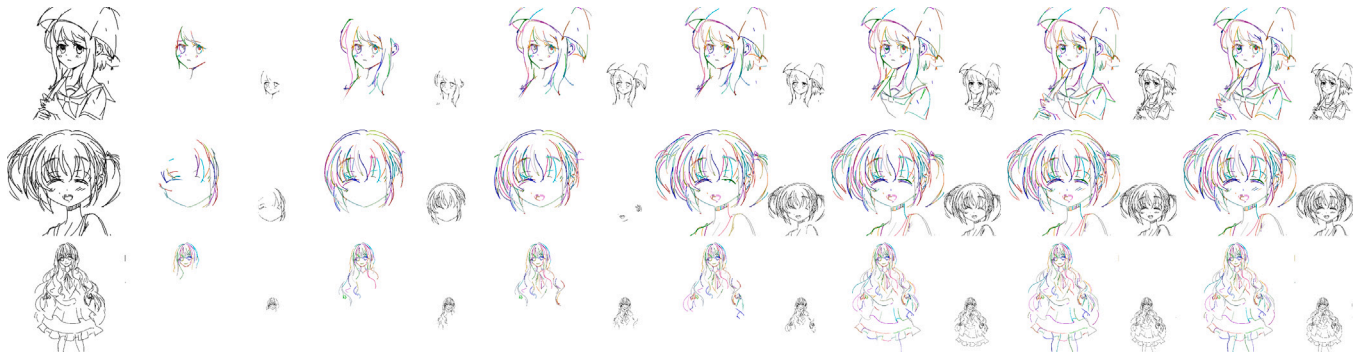


Fig. 1. Line art drawing process reconstruction. Given a line art, our method reconstructs the stroke-by-stroke drawing process. The first column of each line shows the input drawing, followed by keyframes showing the reconstructed sequence of strokes in color. The varying colors indicate the segmentation mask corresponding to each individual stroke. The global reference in the lower right corner of each keyframe is a raster image generated from a diffusion model and does not contain any stroke information. Visually, these global references match the corresponding keyframes in our stroke-by-stroke reconstruction process.

thickness drawn by a pen. Inspired by the natural drawing behavior of artists, we propose a three-level hierarchical approach that models (1) high-level semantic intent, (2) mid-level drawing heuristics, and (3) low-level stroke perception. In the first stage, we employ a pre-trained diffusion model, PaintsUndo [5], to generate a semantic-level drawing sequence, which serves as implicit guidance for global stroke ordering. In the second stage, a neural network trained to simulate human perceptual inference maps cue regions to variable-width local sub-strokes. Finally, the third stage introduces a habitual function that captures human drawing mechanics to merge local sub-strokes and optimize a globally coherent stroke sequence.

In summary, our contributions are as follows:

- We introduce a novel sub-stroke perceptual rendering approach that extracts corresponding strokes from complex line art using a localized cue-aware mechanism, simulating human cognitive perception of strokes.
- We propose a human mechanical optimization strategy to reconstruct the stroke-level drawing process for complex line arts, effectively connect both high-level semantics and low-level stroke details, and overcome the hallucination in the original PaintsUndo [5].
- Through experiments and user studies, we demonstrate that our method generates faithful, artist-like drawing sequences for complex line arts, surpassing existing approaches in accuracy and coherence.

2. Related work

2.1. Drawing process reconstruction

"Drawing process" is a general concept involving various media and different drawing stages. As for process reconstruction of stylized drawings, we refer readers to [1–4,6–8] for a more thorough review. Here we mainly discuss the more related literature on line drawings, commonly involved in the draft stage of all kinds of drawings using a pen or a stylus.

Early work can be traced back to a simpler problem setting - how to generate a reasonable drawing order given a sequence of strokes [9,15]. Fu et al. [9] design the computational procedures to mimic the key principles of drawing order from drawing cognition. Liu et al. [15] further divide the drawing process into different phases and propose entropy-based stroke selection and ordering. Their limitation lies in the poor generalization ability due to the heuristic algorithm and long calculation time, given complex line drawings with several hundred strokes as input.

A later stream of work attempts to sequentially construct stroke-based drawings of common objects [16–19]. Ha et al. [16] model sketch drawings using recurrent neural networks, which enables interesting applications such as ending prediction of incomplete drawings and conditional reconstruction. However, an obvious visual bias can be observed comparing the user input drawing and the reconstructed one. Based on the sequential generative model by [16], Song et al. [17] propose a stroke-level photo-to-sketch synthesis model. Their methods can only generate abstract symbolic drawings requiring separate training for distinct object categories, which are also limited by the complexity of the drawing and the generalization ability. Methods from Huang et al. [18,19] can generate complete portraits that partially match incomplete sketches through either a two-stage [18] or stroke-by-stroke [19] drawing process. However, their temporal sequence of generated results during the drawing process exhibits inconsistency.

Recently, PaintsUndo [5] attempts to model the drawing process of anime drawings. Different from previous work targeting at placing strokes by a single media, PaintsUndo mimics all kinds of human behaviors during the drawing process, including sketching, inking, coloring, shading, transforming, layer operation, and other decision-making processes. We extract the sketching stage prediction from PaintsUndo as the global reference for our process reconstruction pipeline. However, it cannot generate the drawing process stroke by stroke, and the generated frames are often inconsistent, caused by hallucination. Thus, we applied a global optimization to eliminate these negative effects. Fig. 2 gives 15 frames of global reference for each line art corresponding to Fig. 1, where we successfully removed the hallucination frames.

2.2. Stroke extraction

Previous stroke segmentation methods [20,21] can effectively segment symbolic drawings with few strokes in QuickDraw [22]. Kim et al. [20] employ a deep neural network to detect pixel-pair similarity and segment strokes from the original raster line drawing by labeling all foreground pixels based on this similarity. The processing time of this method increases exponentially with the number of foreground pixels, which makes it impractical to scale up to high-resolution line drawings. Ito et al. [21] further enhance the efficiency of the aforementioned method, though this improvement comes at the cost of reduced segmentation accuracy. Despite these efforts, handling more complex topologies remains challenging for them when dealing with high-resolution drawings. Due to the small kernel sizes in their proposed networks, they struggle with complex drawings containing significantly more strokes. Our method addresses this limitation by employing an active window size of 512×512, enabling the segmentation of longer strokes across images of arbitrary size.

Another workflow that can extract paths of strokes from a given line drawing is often known as vectorization. Noris et al. [23] extract stroke topology and centerlines using image gradients alone, which provides insufficient local information, resulting in unreliable vector

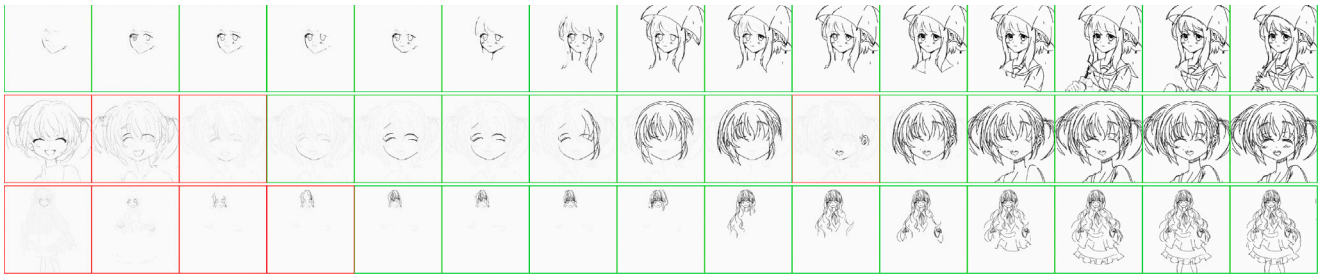


Fig. 2. The hallucination of PaintsUndo [5] may not occur (Row 1), may occur in intermediate frames (Row 2), or only in the first few frames (Row 3). Our method discriminates the hallucination frames (red box) and uses the rest of the keyframes as global references.

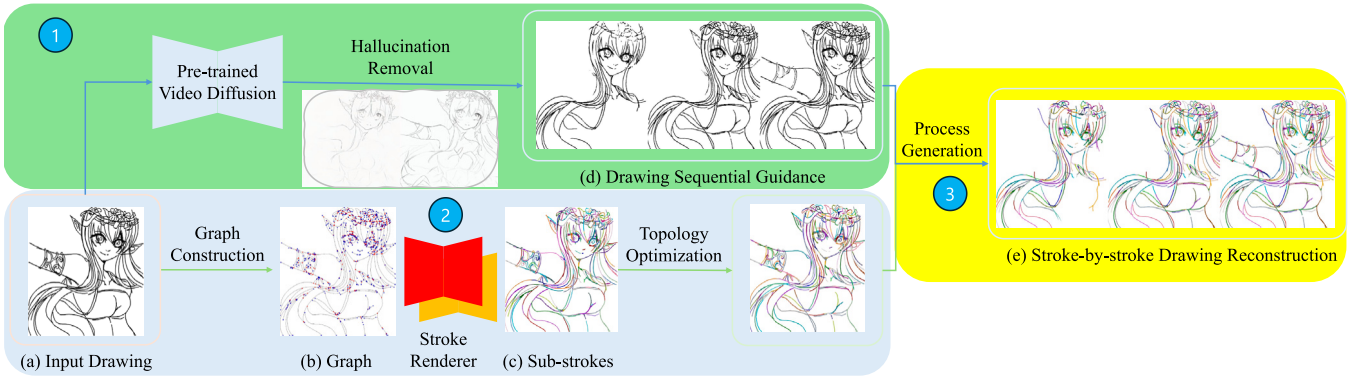


Fig. 3. Given an input line art (a), our method consists of three intermediate modules that simulate human drawing behavior: (1) a diffusion prior used to obtain a global reference sequence (d) for guidance; (2) a stroke renderer trained to generate sub-strokes (c); and (3) human mechanical optimization, encompassing topology optimization, hallucination removal, and process generation.

strokes. Guo et al. [24] propose a two-phase approach for local and global topology reconstruction. Bessmeltsev and Solomon [10] effectively vectorize simple line drawings using a graph-based approach with PolyVector fields. Yet, their method relies on pre-defined X and T-junctions, limiting its ability to handle more complex topology structures. Mo et al. [11] introduce a dynamic window to iteratively search and vectorize undrawn areas. Due to their window searching strategy, their method tends to generate short, consecutive strokes and may miss details in highly complex drawings. Puhachov et al. [12] use keypoint detection to guide graph construction and then optimize it with PolyVector Field. However, its success depends heavily on keypoint detection accuracy. Yan et al. [13] adopt a pairwise training strategy, mapping one line drawing to one single ground truth vector drawing. We doubt the existence of only one "correct" topology in ambiguous cases, as there may be several reasonable solutions. In contrast, we adopt a cue-wise stroke extraction strategy to encourage reasonable stroke orientation given a query region, therefore not limiting to finding the only "correct" stroke segmentation when facing ambiguity. In this case, our method can handle line drawings with complex topologies more flexibly.

3. Method

3.1. Problem formulation

Given a drawing process consisting of n strokes $P = \{s_1, s_2, \dots, s_k, \dots, s_n\}$, its final drawing result is $I = \text{Draw}(P)$, where s_k is the kth stroke on the rendered image I, function $\text{Draw}(\cdot)$ generates the stroke-by-stroke drawing process P. Our goal is to find the inverse function Draw^{-1} that predicts the drawing process P', where $P' = \text{Draw}^{-1}(I)$, $P' = \{s'_1, s'_2, \dots, s'_n\}$. To minimize the difference between the original drawing process P and the predicted P', we have the objective function $\mathcal{F}$:

$$\mathcal{F} = \min \mathcal{L}(P, P') \tag{1}$$

Based on the general drawing principles - from macro to detail, we divide the factors of the loss function $\mathcal{L}$ into the following two: (1) overall stroke order, and (2) shape of each single stroke. (1) and (2) correspond to the global and local information of the overall process reconstruction, respectively. Consequently, we decompose the problem into two relatively independent sub-problems, resulting in a smaller solution space.

Sub-problem (1): stroke order prediction. Given the disordered strokes, we introduce a reference sequence of the drawing process that imitates how an artist draws. A reference sequence here is a series of snapshots arranged chronologically during the drawing process.

Denote the timestamp of the drawing process as $t \in \{0, 1, \dots, N\}$, the global reference at moment t as R_t, and the newly generated stroke at moment t as s_t. The global reference at moment t should be a sum of the generated strokes from time step 0 to t, that is $R_t = \Sigma_{k=0}^{t} \text{Draw}(s_k)$. The global reference between moment i and moment j should satisfy $R_j - R_i = \Sigma_{k=i}^{j} \text{Draw}(s_k)$. Therefore, the objective function of the sub-problem (1) is simplified as

$$\mathcal{F}_1 = \mathcal{L}_g(R_j - R_i, \sum_{k=i}^{j} \text{Draw}(s_k)) \tag{2}$$

where $\mathcal{L}_g$ is the general loss function for sub-problem (1).

The introduction of global reference sequences simplifies the original long-term prediction problem into a short-term layer matching problem between several drawing sequences.

Sub-problem (2): stroke rendering. Given the stroke order in the generated drawing process P', we can focus on rendering each single stroke s_k by minimizing the loss $\mathcal{F}_2$ between the ground-truth strokes s_k and predicted strokes s'_k:

$$\mathcal{F}_2 = \sum_{k \in \{0,1,\dots,N\}} \mathcal{L}_c(s_k, s'_k) \tag{3}$$

where $\mathcal{L}_c$ is the loss function for local optimization. Since sub-problems 1 and 2 interact with each other, we introduce an human drawing

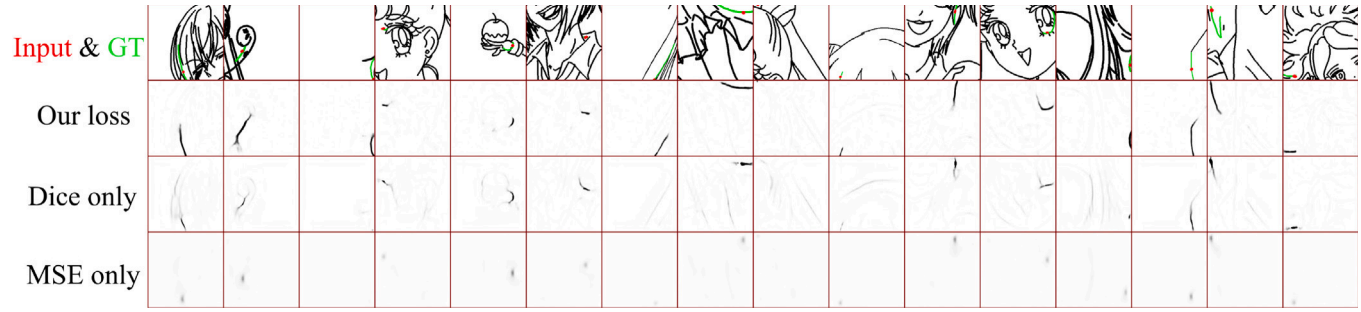


Fig. 4. Ablation study of our proposed loss function. The results demonstrate that using our loss produces outputs most similar to the ground truth stroke (green) when provided with the same input (red point and black line drawing). By the 1500th training iteration, $\mathrm{CNN_V}$ optimized with our loss function begins to converge, whereas Dice fails to capture the stroke shape, and MSE tends to output only the background image.

mechanical optimization, which is further decomposed into drawing process generation and a topology optimization.

3.2. Overview

Our pipeline is shown in Fig. 3. We generate local strokes and the global reference sequence of the input line drawings separately, and then further combine the quality of the drawings through human mechanical optimization.

3.3. Local stroke rendering network

Since the drawing process prediction is the inverse process of drawing creation, we start from sub-problem (2) single-stroke generation. To solve Eq. (3), we construct the graph $G(V, E)$ based on a skeleton map of the input line art, where V includes the stroke endpoints and their intersections, E includes all paths between two adjacent vertices. For each path $e \in E$, there is a geometrically corresponding stroke s. And it is allowed that more than one e pairs an identical stroke. In this case, the part corresponding to e is considered as a sub-stroke. Thus, we transform the problem to train a sub-stroke renderer CNN such that $\mathcal{F}_2 = \Sigma\mathcal{L}(\mathrm{CNN}_E(e), s)$ Since there may be short paths whose $\mathrm{len}(e.\mathrm{path}) \leq V_{th}$ threshold or generating pseudo-paths on the intricate line arts of the strokes, we further relax the correspondence from e to the associated point $v_e = \mathrm{Sampling}(e)$ and use this to train a sub-stroke renderer CNN such that predicted strokes $s' = \mathrm{CNN}_E(e)$ to make the results more stable. If v is belong to stroke s_k, then there exists a sub-stroke $s_{sv} = \mathrm{CNN}_V(v) \in s_k$, and $s_k = \cup s_{sv}$. So our optimization objective is

$$\mathcal{F}_2 = \min \mathcal{L}_c s_k, \mathrm{Merge}_{e\in E}(\mathrm{CNN_V}(\mathrm{Sampling}(e)) \quad \mathrm{if}(\mathrm{len}(e.path) \leq V_{th})\,\mathrm{else}\,:\,\mathrm{CNN_E}(e)) \tag{4}$$

Network architecture. Our model comprises two identical encoder–decoder pairs based on a fully convolutional network (CNN) architecture, as depicted in Fig. 3. The first CNN, referred to as CNN_V, takes as input a point positioned on a stroke and outputs its corresponding stroke. Following CNN_V, the second CNN, denoted as CNN_E, is designed to render the full strokes from edge e.

Inspired by ResNet [25] and Smart Inker [26], each CNN in our framework comprises 24 layers, primarily using Conv-BatchNorm-ReLU blocks. The network downsamples the input three times using convolutions and restores it to its original size through sub-pixel convolutions, allowing the input image size to be any integer multiple of 8. To avoid the error-prone nature of zero-padding, the first layer employs a 9×9 pixel kernel with 4×4 reflective padding. Both our CNN_V and CNN_E share the same architecture as Table A.1 shown in Appendix A.

Loss function. To specify the loss function L_c in Eq. (3), we must account for the imbalance between the output rendering strokes and the background, where negative samples (background) typically dominate.

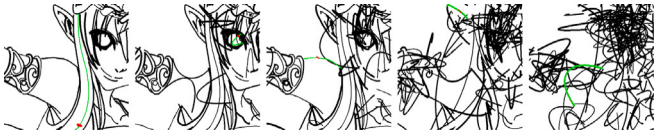


Fig. 5. Data augmentation. We show the stroke replacement with different probabilities. Either the original stroke or the replaced stroke can be selected. As the replacement probability p increases, the line art gradually loses the original information. In addition, the shape of the red input point has a certain randomness in the drawing.

To address this, we adopt a loss function that combines Dice loss [27] with Mean Squared Error (MSE) loss L_2:

$$\mathcal{L}_c = \alpha \cdot L_2(s_i, \mathcal{F}_\theta(I_c, x_i)) + (1-\alpha)\mathcal{L}_{Dice}(s_i, \mathcal{F}_\theta(I_c, x_i)) \tag{5}$$

where $\alpha = 0.09$. The notations are consistent with those in Eq. (3). MSE is a natural choice as it encourages the strokes to maintain their original shapes, while Dice loss is advantageous due to its robustness against class imbalance. It effectively handles the disparity in the number of positive and negative samples, allowing the model to focus more on accurately rendering the smaller strokes. Otherwise, due to the imbalance of positive and negative samples, training with MSE or DICE loss alone tends to be unstable, ultimately leading to the generation of pure white images regardless of the input prompts. We illustrate the effectiveness of our proposed loss function in Fig. 4. Given the imbalanced ratio of positive to negative labels in the output, relying solely on MSE disproportionately emphasizes negative samples, resulting in outputs consisting exclusively of background images.

Data synthesis. To validate the stroke order and learn the stroke rendering of hand-drawn high-quality line arts, we collected data as described in Section 4. To stabilize training and avoid overfitting, we add random stroke replacement in addition to random rotation, inversion, and scaling, which is used to correspond to various complex topologies. As shown in Fig. 5, we replace the original strokes with a certain probability p to the arcs with the same starting and ending points, which can increase the diversity of strokes and enable us to get the stroke renderer in a limited number of samples and realize the few-shot learning of $\mathrm{CNN_V}$ and $\mathrm{CNN_E}$.

Two-stage training. As Fig. 6 shows, in Stage I, we randomly crop a patch from the original line art as the active window, select a stroke from it, and randomly generate a small dot (red). We then generate a mask for this small red dot and concatenate it with the cropped original line art to create a two-channel image as input. Due to the local connectivity of CNN, $\mathrm{CNN_V}$ with one point input has difficulty in corresponding to a long stroke (Fig. 7).

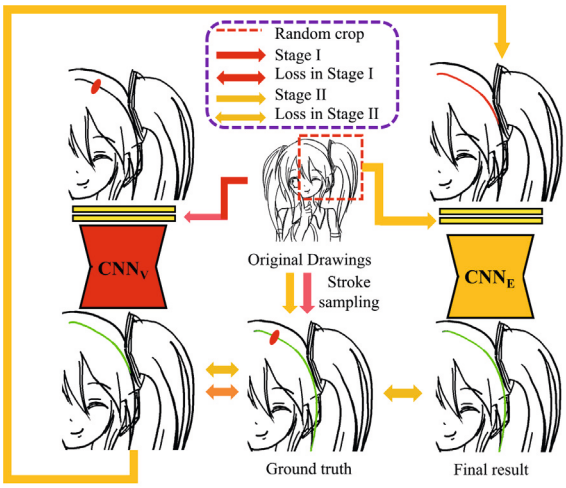


Fig. 6. Two-stage stroke renderer training. Our input for both CNNs is a 2-channel image. Once a stroke (green) is selected, we only need to sample a random query region (red) on this stroke for CNN_V in Stage I, and its output is the input of CNN_E. Note that before Stage II starts, CNN_V copies the parameters to CNN_E as initialization, and that both CNNs are trained at Stage II.

After training, we obtain two CNNs as shown in Fig. 6. They are used for different types of inputs, and we show this separately in Fig. 7: CNN_V can render strokes for the intersection part of strokes, but the results are usually localized; CNN_E is effective for rendering long strokes. In the practical inference, we chose an active window of 512×512 to improve the computing speed. To render strokes that may exceed the size of the window, we split the input (e.path) of CNN_E into K line segments (according to len(e.path)), with partial sharing between the successive segments, and merge the final result into a stroke.

3.4. High-level stroke order generation

Typically, the best way to obtain a global painting process is to record the entire process and then generate snapshots. However, such data is not easy to collect and obtain, so we used the method PaintsUndo [5] to predict a reasonable sequence of painting processes from the diffusion model. Since we have already obtained a sequence global reference $\{R_i\}$ of the drawing process from the time i from 0 to T time, we convert the objective of Eq. (2) into

$$\mathcal{F}_1 = \sum_{i=0}^{T} \mathcal{L}_g(R_i, \sum_{k=0}^{i} \text{Draw}(s_k)) \tag{6}$$

This means that each sub-stroke should belong to the frame in which it first appeared. In the actual implementation, all the sub-strokes go frame by frame along the chronological order to find the best match. For R_i and a certain sub-stroke s_k at time i, our determination function $\mathcal{D}$ is

$$\mathcal{D} = \frac{\text{Area}(R_i \bigcap \text{Draw}(s_k))}{\text{Area}(\text{Draw}(s_k))} \tag{7}$$

where function $\text{Area}(\cdot)$ calculates the pixel-level area of its input raster image, and we determine which frame a stroke should belong to by setting the hyperparameter M_{th}. This method eliminates the hallucinations from the reference.

Hallucination removal. Our stroke-level rendering module operates as a self-contained system that functions without requiring any global reference image. This architecture endows our method with intrinsic hallucination detection capabilities: whenever PaintsUndo's prediction sequence exhibits either (Rule 1) dissimilarity to the original sketch in overlapping regions or (Rule 2) temporal discontinuity with the previous frame, we categorize such a frame as a hallucination.

To simplify and speed up the calculation, we use Eq. (7) to pre-calculate the neighboring frames as well as the current frame between the input frames to determine whether the current frame is a hallucination image or not, which implements the hallucination removal in Figs. 2 and 3. In this step, we use a relatively strict condition for distinction and the corresponding rules are:

- Rule 1. Hyperparameter $M_{th} = 0.6$.
- Rule 2. The number of pixels in the previous frame is greater than the next (continuity law)

3.5. Human mechanical optimization

So far, we have obtained global references for high-level semantics and sub-strokes for low-level semantics. Our next task is to perform an optimization that mimics the artist's behavior and synthesizes a meaningful order. In the context of line art, the primary factor influencing stroke order is user habit, which is unique to each user. This assertion is corroborated by the findings of Qiu et al. [14], which demonstrate that even when stroke orders from professional artists, 35.94% of participants perceive them as computer-generated rather than hand-drawn. Nevertheless, the remaining 64.06% demonstrate the existence of similarities. Consequently, by observing the artists' drawing behavior during the data collection process, we simplified the primary factors to two: the smoothness of the line strokes $\mathcal{S}$ and the mechanics of the drawing $\mathcal{M}$.

Mechanics of the drawing. Artists use strategies that are more comfortable for themselves, including drawing outlines before details, fewer hand movements when drawing the same object, from left to right (if right-handed), and top to bottom (for gravity). We apply these rules in inter-frame stroke ordering with $\mathcal{M}$. For a stroke s_t, the stroke s_{t+1} is defined as its neighbor (undrawn). The stroke priority is thereby determined as $\text{SP}(s_t, s_{t+1}) = S(s_t, s_t{+}1){+}M(s_{t+1}{-}s_t)$, where S mentioned above denotes the possibility of merging, and $\mathcal{M}$ become the cost function for the movement of the strokes. The inter-frame optimization is based on the stroke selection of the graph. We adopt a simple BFS (Breadth-First Search) when building the graph from the skeleton map (Fig. 3) as an implementation, selecting the stroke s_{t+1} with minimal SP as the next stroke in a junction (other stroke are delayed), and update the current position for next selection, in line with human drawing habits. The validity of this ordering was verified in our user study (Experiment 2 in Section 4). To enhance the efficiency of the sorting process, the strokes are merged before BFS with the following topology criteria.

Topology optimization for stroke smoothness. Users will prefer smoother strokes for aesthetics and merging sub-strokes. The two mergeable neighbor sub-strokes should have some overlap and no increase in topological complexity after merging. We first define a stroke as follows: given a line art consists of $P = \{s_1, s_2, \ldots, s_k\}$, for a graph $G_{s_k}(V, E)$ built from a skeleton of a stroke s_k, if there are only two vertices with $\text{len}(G_{s_k}.V) = 2$ and one edge with $\text{len}(G_{s_k}.E) = 1$, we consider it as the simplest stroke; if $\text{len}(G_{s_k}.V) > 2$ but it can be drawn in a single stroke and it does not intersect with the other simplest strokes, we call it a single connected component stroke. We consider that a line art can be decomposed into the simplest and single connected component strokes. Thus, our objective function $\mathcal{F}_t$ for topology optimization is

$$\mathcal{F}_t = \min \sum_{s \in P} (\text{len}(G_s.E) + \text{len}(G_s.V) + \frac{1}{\sum_{e \in E} \text{len}(G_s.E.path)}) \tag{8}$$

To restore the original topology from complex line drawings, a topological simplification strategy is used—as shown in Fig. 8, we use a

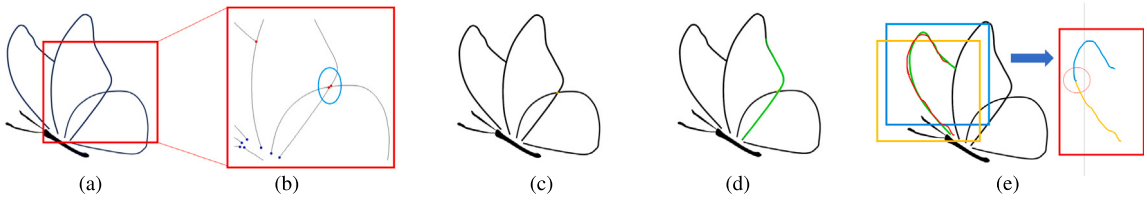


Fig. 7. CNN_V vs. CNN_E for single stroke rendering of a line art (a). With the same short edge input (b) as a cue region, CNN_E may not work (c). While CNN_V (d) can better determine the direction of strokes. On the other hand (e), CNN_E can get a complete stroke (green) when the input is a long edge path. As the input exceeds the size of our active window, we adaptively split the input (red segment) into two parts (orange and blue), whose results would be merged automatically as the complete green stroke. Note that the split segments have a small overlap in the red circle.

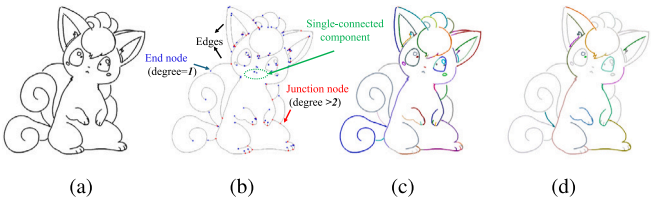


Fig. 8. Topology optimization. Given an input line art (a), sub-strokes (c) are generated from the graphs (b) and subsequently merged (highlighted in (d)) when they meet the criterion Eq. (9).

topological criterion to try to merge sub-strokes in the domain. The criterion of the merge test for any sub-stroke s_i and s_j sharing a junction node is:

$$\begin{cases} \text{len}(\cup(G_{s_i} \cdot G_{s_j}).V) <= \max(\text{len}(G_{s_i}.V), \text{len}(G_{s_j}.V)) \\ \text{len}(\cup(G_{s_i} \cdot G_{s_j}).E) <= \max(\text{len}(G_{s_i}.V), \text{len}(G_{s_j}.E)) \end{cases} \tag{9}$$

When the merge is successful, the path is automatically increased, so the third term of Eq. (8) is implicit in the merge operation. Among the merged strokes, we give a higher stroke priority to strokes. At this point, we have completed the rendering of all the single strokes in the line drawing. Determined the shape of the strokes $s \in \{s_1, s_2, \ldots\}$ in the sub-problem 1.

4. Experiments

4.1. Dataset and settings

Dataset. To validate our method, we require high-quality line art along with its associated drawing processes as ground truth. While the QuickDraw [22] and Benchmark [28] datasets offer a variety of vector sketches, they do not include information on the drawing process. Other datasets, such as OpenSketch [29], SpeedTracer [30], and DifferSketching [31], provide sketches with both stroke paths and timestamps, but these primarily depict industrial components or common objects, which do not align with our research focus. Thus, we collect 182 pieces of clean and high-quality anime line arts from experienced artists, each annotated with stroke paths and timestamps for each point along the paths. Due to individual differences in brush strokes, we selected 2 artists with similar brush strokes by observation and took 13 of their drawings as our training data to maintain consistent stroke rendering in network learning, and the rest 169 as the test set T169. As our model is trained on stroke-level data, the output for each training step is a single stroke. And 13 artworks contain 2920 artist-drawn strokes. Through augmentations such as random curve-to-arc replacement in Fig. 5, translation, rotation, and cropping, we generate tens of thousands of variations. Combined with Dropout layers in our network architecture and specialized loss functions, this approach effectively prevents overfitting during training. To verify that our dataset is more challenging (including varying-width strokes and complex topology), we chose the same dataset of Deep Sketch Vectorization (DSV) [13] as a control test set T369 (369 line drawings with uniform and thin strokes).

Implementation details. All experiments were conducted on a Windows platform using an NVIDIA RTX 4090 GPU. In Eq. (4), we set $V_{th} = 20$ pixels. For training the stroke renderer, both input patches and output images were set to a resolution of 512 × 512 pixels. In Stage I, training was carried out over 15,100 iterations with a batch size of 16, followed by a Stage II refinement phase comprising 500 iterations, maintaining the same batch size. Before Stage II, parameters learned by CNN_V in Stage I were transferred to CNN_E as the initialization for joint training. As the training data were randomly cropped from original line drawings, the term "iterations" more accurately describes the training process than "epochs". We used the Adam optimizer [32] with an initial learning rate of 0.001. A learning rate decay schedule was applied every 1000 iterations with a decay factor of $\gamma = 0.5$. For data augmentation, the stroke replacement probability was set to $p = 0.5$ in Stage I and $p = 0.2$ in Stage II. For optimization, we used $M_{th} = 0.5$ in Eq. (7) for the global reference generated by the pre-trained diffusion model and $M_{th} = 0.8$ for ground truth frames in the following evaluation, and $M_{th} = 0.6$ in hallucination removal, as we mentioned.

4.2. Evaluation

Benchmark. Our method is the first to address image-to-drawing-process inversion for complex line arts (**raster images**), generating a stroke-by-stroke process. We select two relevant tasks with state-of-the-art baselines that can yield similar results: **vectorization** — virtual sketching (VS) [11], PolyVectorization (PV) [12], Keypoint-Driven PolyVector-Flow Vectorization (KPV) [33], DSV [13] and **image-to-painting-process generation** — PaintTransformer (PT) [2] and Stylized Neural Painting (SNP) [3] to evaluate reconstruction accuracy and image quality. The inputs for both relevant tasks and our approach are the same, raster images in 1024 × 1024. Because Fu et al. [9] attempted to solve an NP-hard problem when finding the Hamiltonian paths, this heuristic-based stroke-ordering method for **vector strokes input** is too time-consuming with hundreds of strokes and unsuitable for a fair comparison. Semantic-based image-to-painting methods [4,8] do not work because they cannot obtain semantic layers in line arts (reason (c) in Section 1).

As such, we evaluate our method in two dimensions: the reconstruction quality of the results, and the human-likeness of the drawing process. We compare our method with different baselines as described below.

Reconstruction quality. The objective of this evaluation is to ascertain the degree of pixel-level similarity between the target line drawing image and the rendered output. We use Chamfer Distance (CD) and Intersection over Union (IoU) between the final reconstructed result and ground truth to measure whether the method can recover the input drawing. Fréchet inception distance (FID), compared with ground truth, peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM), are used to measure the image generation quality. As shown

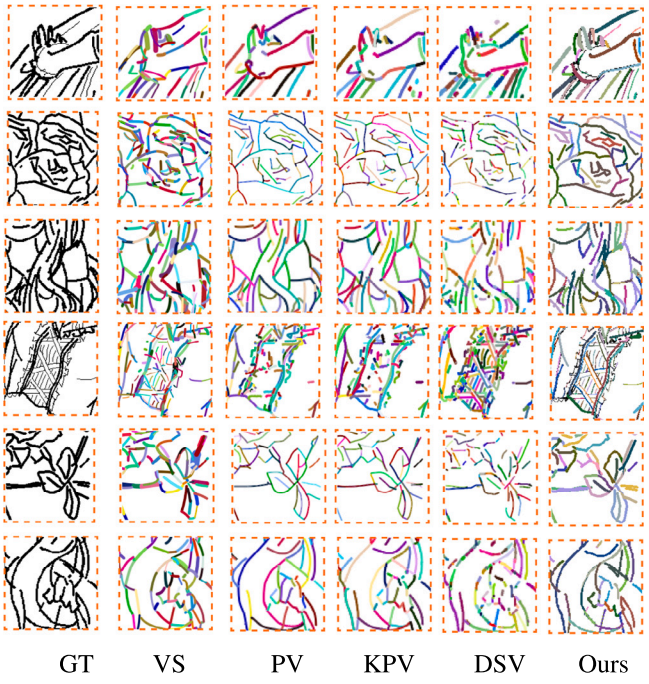


Fig. 9. Reconstruction details comparison. Our method reconstructs line arts with complex topologies more accurately. As our extraction method allows for variable width (Row 4) and differentiation of overlapping strokes (the two strokes in the upper right of Row 5), our reconstruction results are numerically superior to the SOTA vectorization method in Table 2.

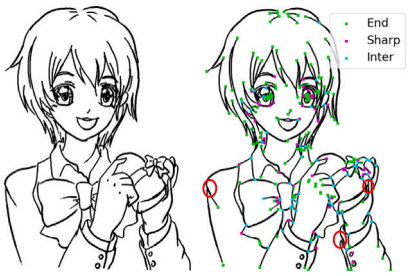


Fig. 10. Cause of KPV's low SEC performance. For complex line drawings, undetected junctions during KPV's preprocessing stage (indicated by red circles) cause cascading failures in vectorization optimization, resulting in suboptimal SVG outputs and low SEC.

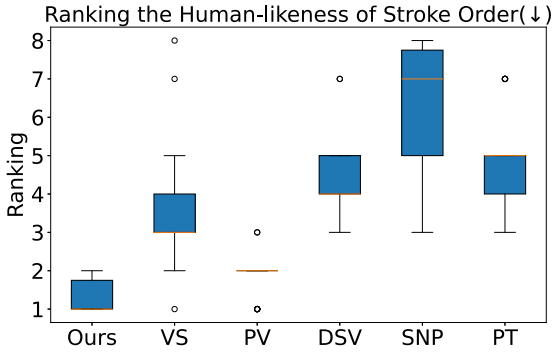


Fig. 11. Global human-likeness ranking.

in Table 2, our method can best recover the original input line art with a maximum IoU and minimum CD while maintaining the highest image quality output in both T169 and T369. All methods perform worse on T169 than on T369, suggesting that our dataset is indeed more challenging. As Fig. 9 shows, our stroke rendering results in complex topologies recovering the original line art better than other vectorization methods, which is also consistent with the previous quantitative results. The probable reason that KPV's SSIM is the highest comes from its low success rates — 59.17% for T169 and 82.00% for T369 (Table 3) — meaning that it can only handle relatively simple line drawings. The qualitative comparison results with image-to-painting-process methods (SNP, PT) and vectorization methods (DSV, PV, KPV, VS) are shown in Fig. 14. This indicates that quantitative performance and quantitative evaluation results are visually consistent.

As presented quantitatively through Successful Examples Count (SEC) metrics in Table 3, we attribute distinct failure mechanisms for each baseline method:

1. For DSV cases exhibiting low SEC, the predominant cause stems from unhandled stroke width variations encountered during processing of intricate line drawings—a condition that critically destabilizes its second-stage fine-tuning module per algorithmic specifications outlined originally by Yan et al. [13].

2. Regarding KPV's suboptimal performance, systematic failures originate primarily in the preprocessing phase, where erroneous junction detections occur within high-complexity regions (e.g., red circles in Fig. 10).

Global human-likeness. To further substantiate that our proposed method (global PaintsUndo [5] reference images + local reordering) in Fig. 14 generates globally human-like strokes, we conducted a rigorous user study structured as follows: We recruited 30 participants to evaluate stroke generation quality through comparative analysis of animation sequences produced by six competing methods (ours, PT, SNP, VS, KPV, PV, DSV). Participants ranked outputs based on perceptual resemblance to human-drawn strokes using a curated set of six complex line drawings exhibiting structural complexity (referenced in supplementary videos Comparison 1–3). Key findings demonstrate the efficacy of our approach:

1. As quantified in Fig. 11, our method achieved superior ranking results against baseline approaches.

2. Statistical significance was confirmed through: the p-value of ANOVA (Analysis of Variance, 3.49×10^{-49}) and the one of Kruskal–Wallis test (7.29×10^{-29}), rejecting null hypotheses at $\alpha = 0.05$ threshold. Moreover, a pairwise two-tailed t-test (Table 1) revealed significant inter-method disparities, conclusively establishing optimal performance for complex line arts.

Computational time. As shown in Table 3, our run times are competitive, albeit slower than those of VS, PT, and SNR. However, our reconstruction quality is superior. Meanwhile, we maintain a rendering time below 0.3 s per stroke, allowing users to observe real-time, stroke-by-stroke rendering visualizations while awaiting results without noticeable delays.

Generalizability. As Fig. 12 shown, Beyond anime characters, our approach can be applied to various objects, such as (a) machinery (e.g., a motorcycle), (b) a botanical drawing (a flower), (c) complex architectural structures, and (d) a scene containing botanicals and an animal.

Quality of the drawing process. Since our approach follows a human mechanical optimization strategy, three main factors influence whether the generated stroke order is human-like or not: (1) how well the generated drawing process matches the global reference, (2) how well the local drawing process follows common human drawing rules, and (3) how well the quality of each single stroke. We decompose the evaluation of these factors into the following experiments.

Experiment 1: To measure the similarity between our generated stroke order and the global reference (main factor 1), we calculated precision and recall between rendered images of the stroke sequence generated by us and the reference at each timestep, called per-frame precision (recall); As Fig. 15 shows, the per-frame precision between

Table 1
Pairwise two-sided t-test results (p-values) for global human-likeness ranking.

	Ours	VS	PV	DSV	SNP	PT
Ours	–	2.69×10^{-12}	3.80×10^{-05}	2.83×10^{-21}	4.83×10^{-26}	5.26×10^{-23}
VS	2.69×10^{-12}	–	1.88×10^{-08}	3.48×10^{-02}	6.04×10^{-11}	1.31×10^{-04}
PV	3.80×10^{-05}	1.88×10^{-08}	–	1.12×10^{-16}	2.86×10^{-23}	3.87×10^{-19}
DSV	2.83×10^{-21}	3.48×10^{-02}	1.12×10^{-16}	–	3.73×10^{-09}	2.04×10^{-02}
SNP	4.83×10^{-26}	6.04×10^{-11}	2.86×10^{-23}	3.73×10^{-09}	–	1.84×10^{-05}
PT	5.26×10^{-23}	1.31×10^{-04}	3.87×10^{-19}	2.04×10^{-02}	1.84×10^{-05}	–

Table 2
Quantitative comparisons. Results are presented in the $X|Y$ format, where X corresponds to the T169 dataset (Ours) and Y to the T369 dataset [28]. (CD: 10^{-6}, IoU/SSIM: 10^{-2}).

Method	CD ↓	IoU ↑	PSNR ↑	SSIM ↑	FID ↓
SNP	63.69 | 17.75	21.23 | 2.31	11.51 | 16.38	65.77 | 3.28	459.10 | 454.07
PT	37.62 | 15.24	7.86 | 2.31	12.31 | 15.99	1.44 | 1.67	158.68 | 249.14
PV	18.48 | 5.10	73.34 | 64.19	15.22 | 21.59	74.62 | 86.78	21.14 | 14.85
VS	19.27 | 5.62	72.22 | 64.72	15.29 | 21.68	74.60 | 86.76	17.77 | 16.53
KPV	29.96 | 4.39	51.34 | 69.31	14.45 | 22.70	**77.41** | **87.12**	56.51 | 16.32
DSV	37.03 | 4.93	49.25 | 65.65	13.97 | 21.84	74.32 | 86.69	59.97 | 14.83
Ours	**0.87** | **0.22**	**98.79** | **98.69**	**28.08** | **46.06**	74.69 | 86.82	**2.28** | **1.59**

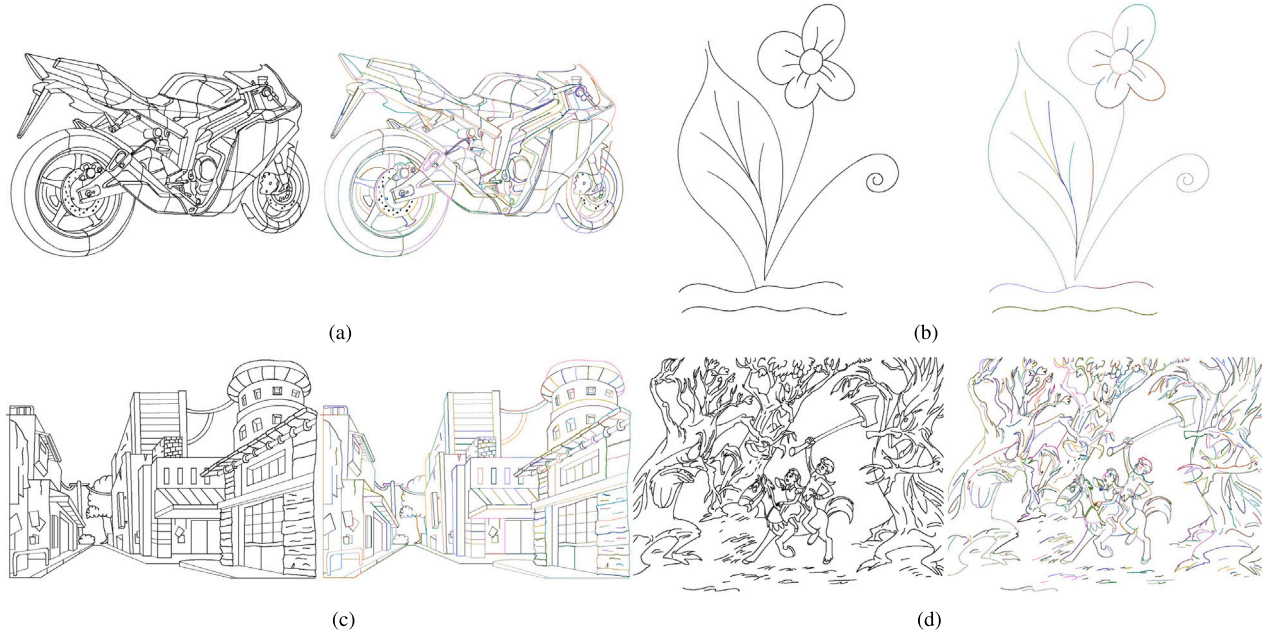


Fig. 12. Generalizability of our method across more diverse categories. The input line drawings on the left of (a) ~ (d) are sourced from T369. Our method can be applied to (a) a motorcycle, (b) a flower, (c) architecture, and (d) a complex scene with an animal.

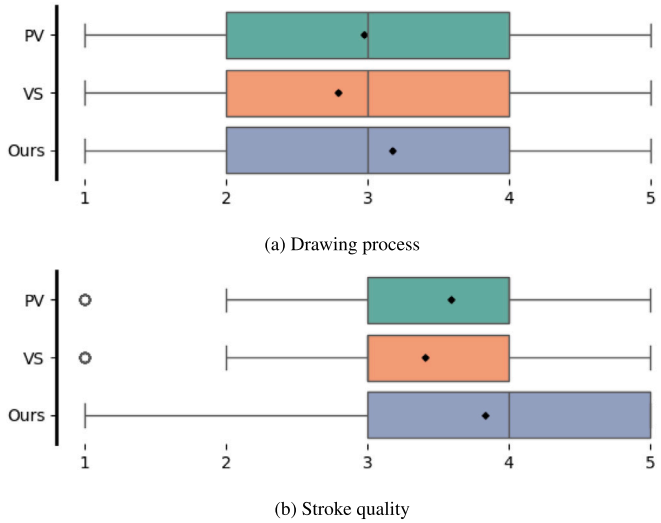
Table 3
Computation time and successful examples count (SEC).

Method	T169		T369	
	SEC	Time (s)	SEC	Time (s)
SNR	169	59.89	369	63.08
PT	169	14.33	369	11.29
PV	169	331.59	369	68.04
VS	169	55.48	369	20.62
KPV	100	717.05	302	252.86
DSV	153	179.94	363	163.51
Ours	169	50.53	369	35.83

our process images and the final reference is always maintained at a high level. And our per-frame recall increases gradually to 98.53% during the drawing process, which indicates the stroke process generated by our method can well mimic the reference drawing process.

Experiment 2: We set the global reference the same and render the sequence of our segmented strokes (Ours), vectorized strokes by VS [11], and PV [12] for comparison of the generated local stroke order, and the humanlikeness of every single stroke. We select 9 drawings from the training data and render 27 stroke sequences with three different sets of strokes. We invite 36 participants to rate each sequence on a likely scale from 1 to 5. The participants were students of higher education, and 34 of them were novices in drawing. They were presented with each stroke sequence in GIF format and asked to evaluate the segmentation results by rating the following statements on a scale of 1 to 5, where 1 indicates “strongly disagree” and 5 indicates “strongly agree”.

- I think that the drawing process depicted in the GIF above exhibits a human-like quality. (Main factor 2)
- I think that each single stroke depicted in the GIF above is of high quality. (Main factor 3)



(a) Drawing process

(b) Stroke quality

Fig. 13. Distribution of user ratings for the drawing process and stroke quality.

Table 4
ANOVA and Kruskal–Wallis test for the Ours, PV, and VS groups.

	ANOVA	Kruskal–Wallis
Drawing process	5.42×10^{-4}	4.29×10^{-4}
Stroke quality	2.89×10^{-6}	6.37×10^{-7}

Table 5
The p-values of two-tailed t-test between results by ours and PolyVectorization (PV) [12], and by ours and Virtual Sketching (VS) [11].

	Ours-PV	Ours-VS
Drawing process	4.48×10^{-2}	1.04×10^{-4}
Stroke quality	3.04×10^{-3}	9.47×10^{-7}

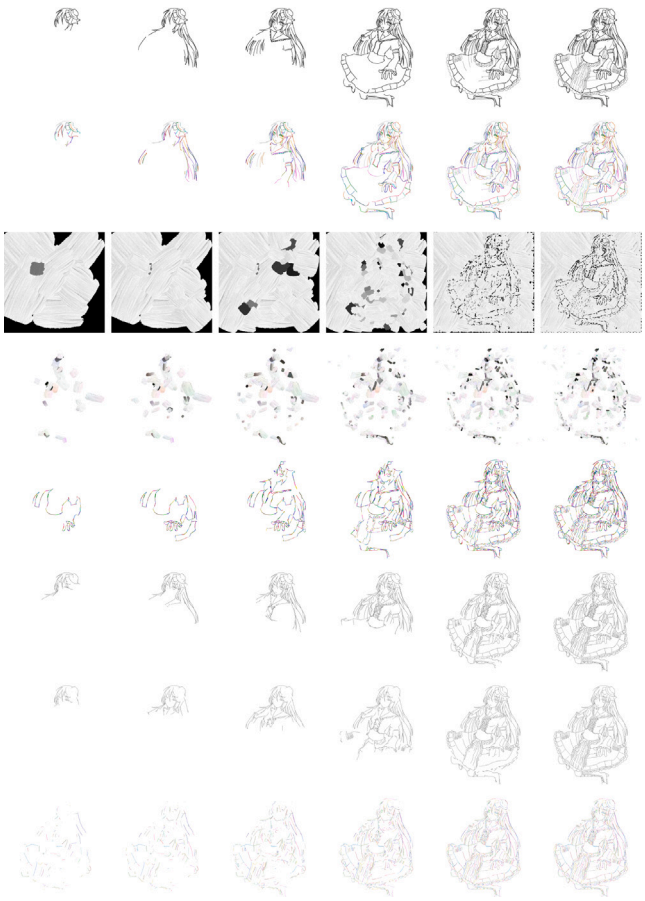
Note that our results here only adopted a 1-frame (the input line drawing) global reference. This means these stroke orders are following our inter-frame stroke ordering. This user experiment demonstrates the validity of our inter-frame stroke ordering, while confirmation of the validity of the global optimization is provided by Experiment 1 in the main text, which provides a qualitative and quantitative evaluation.

Results. The average scores in terms of local stroke order are 2.98, 2.73, and 3.17 for PV, VS, and Ours, respectively. PV, VS, and Ours score 3.59, 3.41, and 3.84, respectively, for the quality of the single stroke.

The distribution of user ratings is visualized in Fig. 13. The stroke sequence generated by our approach achieves a slightly higher average score in the evaluation of the drawing process, suggesting superior local process reconstruction. Additionally, participants rated the quality of our strokes as the highest on average among the three methods, highlighting the capability of our approach to reconstruct individual strokes.

We conducted a hypothesis test between the distributions of user ratings, and the p-values are less than 0.005 between each pair. The p-values of ANOVA and Kruskal–Wallis test are reported in Table 4, and these results support a significant difference between the three data groups. To determine which specific group pairs showed statistically significant differences in means, pairwise two-tailed hypothesis tests were performed. The p-values two-sided two-sample t-test are reported in Table 5. This implies a significant difference in scores between our method and the other method, and the null hypothesis that both average scores are the same can be rejected with a probability of $(1-p)$.

Participants recognize that the local stroke order generated by our method is more reasonable, while our rendered strokes are of higher quality. The results indicate that our inter-frame sorting algorithm and rendered strokes outperform the other two vectorization methods.



(a) t = 0.05 (b) t = 0.10 (c) t = 0.20 (d) t = 0.40 (e) t = 0.80 (f) t = 1.00

Fig. 14. Qualitative comparisons for global drawing process. Our method allows a good reconstruction from different moments (t=0.05, 0.1, 0.2, 0.4, 0.8, 1.0) at the stroke level when the corresponding ground truth is used as a reference, and for the other methods, we show their original order at the corresponding moments. From top to bottom, they are ground truth (also our global reference), ours, PT, SNP, VS, KPV, PV, DSV.

Limitations and future work. While user studies indicate our stroke order appears more human-like, and statistical analysis supports this observation, this does not imply universal adoption, particularly among experts. These individuals possess deeply personalized experiences and refined drawing techniques developed over years of practice. Both PaintsUndo and our inter-frame ordering may not meet these artists' expectations. Consequently, developing customized stroke ordering methods that adapt to experts' individual habits based on our existing approach represents a promising future research direction. Another limitation is that our current method is only suitable for clean line arts without shading effects: when the stroke overlap becomes excessive, resulting in shaded regions (such as solid circles), the skeleton extraction algorithm we employ fails to derive reasonable cues in these areas. This often leads to reconstruction failures. Therefore, achieving reasonable guidance generation for regions with higher ambiguity caused by shading will constitute another direction for future work.

5. Conclusion

Our method successfully inverts the stroke-by-stroke drawing process from a complex line art. Our stroke renderer and human mechanical optimization guarantee the high quality of a single stroke,

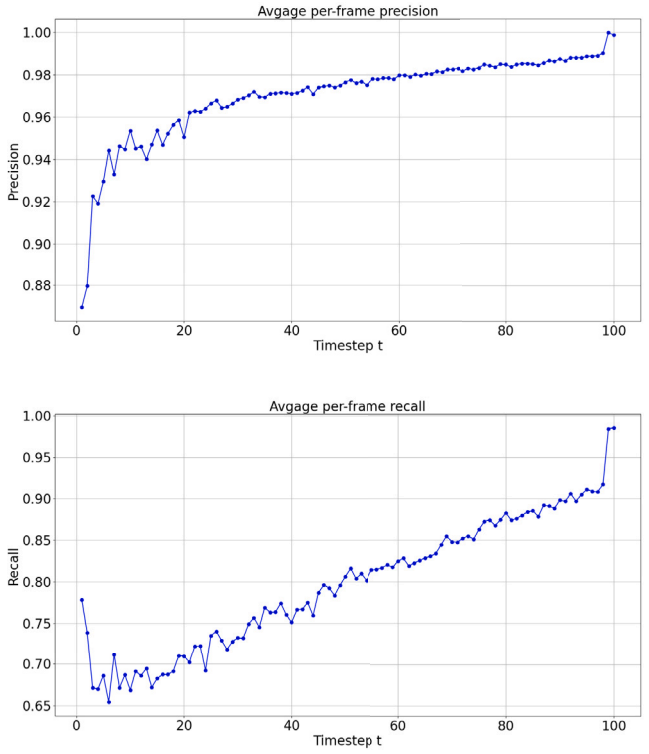


Fig. 15. Per-frame precision and recall along the normalized timestep (t=1%–100%).

while global optimization is capable of reconstructing the keyframes of the drawing process with a relatively reasonable stroke order. Our method yields higher quality reconstructed strokes than the traditional SOTA vectorization method, and overcomes the problem of PaintsUndo having no stroke-by-stroke order and the hallucination frame generation. In addition, a user study has shown that our inter-frame stroke order mechanism has some soundness. As a limitation, our method may not deal with highly overlapped strokes (such as shading) well, which leaves it as future work. Because the evaluation of "human-like" and "reasonable" stroke ordering is highly individualized, another future work could also investigate how to generate individually tailored stroke orders that can be better applied to teaching users how to draw.

CRediT authorship contribution statement

Zhengyu Huang: Writing – review & editing, Writing – original draft, Visualization, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Zhongyue Guan:** Validation. **Zeyu Wang:** Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by Guangzhou Industrial Information and Intelligent Key Laboratory Project #2024A03J0628 and Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things #2023B1212010007. We thank the reviewers for their helpful suggestions and all the participants who contributed to our studies.

Table A.1
Network architecture. CBR is a Conv-BatchNorm-ReLU block. PCB is a PixelShuffle-Conv2d-BatchNorm block with a fixed upscale_factor=2. KS is the kernel size. The probability for Dropout layer is 20%.

Layer	KS	Stride	Padding	Output channel size
Input				$2 \times H \times W$
CBR	(9,9)	2	4	$128 \times \frac{H}{2} \times \frac{W}{2}$
CBR	(3,3)	1	1	$128 \times \frac{H}{2} \times \frac{W}{2}$
CBR	(3,3)	1	1	$128 \times \frac{H}{2} \times \frac{W}{2}$
CBR	(3,3)	1	1	$128 \times \frac{H}{2} \times \frac{W}{2}$
CBR	(3,3)	1	1	$128 \times \frac{H}{2} \times \frac{W}{2}$
CBR	(3,3)	2	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	2	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$512 \times \frac{H}{8} \times \frac{W}{8}$
CBR	(3,3)	1	1	$512 \times \frac{H}{8} \times \frac{W}{8}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
CBR	(3,3)	1	1	$256 \times \frac{H}{4} \times \frac{W}{4}$
Dropout				
PCB	(3,3)	1	1	$64 \times \frac{H}{4} \times \frac{W}{4}$
PCB	(3,3)	1	1	$16 \times \frac{H}{2} \times \frac{W}{2}$
PCB	(3,3)	1	1	$4 \times H \times W$
Sigmoid				$1 \times H \times W$

Appendix A. Our network architecture

See Table A.1.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cag.2025.104365.

Data availability

Data will be made available on request.

References

[1] Huang Z, Heng W, Zhou S. Learning to paint with model-based deep reinforcement learning. In: Proceedings of the IEEE international conference on computer vision. 2019.
[2] Liu S, Lin T, He D, Li F, Deng R, Li X, et al. Paint transformer: Feed forward neural painting with stroke prediction. In: Proceedings of the IEEE international conference on computer vision. 2021.
[3] Zou Z, Shi T, Qiu S, Yuan Y, Shi Z. Stylized neural painting. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 15689–98.
[4] de Guevara ML, Fisher M, Hertzmann A. Segmentation-based parametric painting. 2023, arXiv:2311.14271.
[5] Paints-Undo Team. Paints-undo GitHub page. 2024.
[6] Zhao A, Balakrishnan G, Lewis KM, Durand F, Guttag JV, Dalca AV. Painting many pasts: Synthesizing time lapse videos of paintings. In: 2020 IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 8432–42. http://dx.doi.org/10.1109/CVPR42600.2020.00846.
[7] Blattmann A, Dockhorn T, Kulal S, Mendelevitch D, Kilian M, Lorenz D, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. 2023, arXiv:2311.15127.

[8] Chen B, Wang Y, Curless B, Kemelmacher-Shlizerman I, Seitz SM. Inverse painting: Reconstructing the painting process. 2024, arXiv:2409.20556.
[9] Fu H, Zhou S, Liu L, Mitra NJ. Animated construction of line drawings. ACM Trans Graph 2011;30(6):1–10. http://dx.doi.org/10.1145/2070781.2024167.
[10] Bessmeltsev M, Solomon J. Vectorization of line drawings via polyvector fields. ACM Trans Graph 2019;38(1). http://dx.doi.org/10.1145/3202661.
[11] Mo H, Simo-Serra E, Gao C, Zou C, Wang R. General virtual sketching framework for vector line art. ACM Trans Graph 2021;40(4):51:1–51:14.
[12] Puhachov I, Neveu W, Chien E, Bessmeltsev M. Keypoint-driven line drawing vectorization via PolyVector flow. ACM Trans Graph 2021;40(6). http://dx.doi.org/10.1145/3478513.3480529.
[13] Yan C, Li Y, Aneja D, Fisher M, Simo-Serra E, Gingold Y. Deep sketch vectorization via implicit surface extraction. ACM Trans Graph 2024;43(4). http://dx.doi.org/10.1145/3658197.
[14] Qiu S, Wang Z, McMillan L, Rushmeier HE, Dorsey J. Is drawing order important? In: Babaei V, Skouras M, editors. 44th Annual conference of the European association for computer graphics, eurographics 2023 - short papers. Eurographics Association; 2023, p. 37–40. http://dx.doi.org/10.2312/EGS.20231009.
[15] Liu J, Fu H, Tai C-L. Dynamic sketching: simulating the process of observational drawing. In: Proceedings of the workshop on computational aesthetics. New York, NY, USA: Association for Computing Machinery; 2014, p. 15–22. http://dx.doi.org/10.1145/2630099.2630103.
[16] Ha D, Eck D. A neural representation of sketch drawings. In: International conference on learning representations. 2018, URL https://openreview.net/forum?id=Hy6GHpkCW.
[17] Song J, Pang K, Song Y-Z, Xiang T, Hospedales T. Learning to sketch with shortcut cycle consistency. In: 2018 IEEE/CVF conference on computer vision and pattern recognition. Institute of Electrical and Electronics Engineers; 2018, p. 801–10. http://dx.doi.org/10.1109/CVPR.2018.00090, Computer Vision and Pattern Recognition 2018, CVPR 2018 ; Conference date: 18-06-2018 Through 22-06-2018, http://cvpr2018.thecvf.com/.
[18] Huang Z, Peng Y, Hibino T, Zhao C, Xie H, Fukusato T, et al. DualFace: Two-stage drawing guidance for freehand portrait sketching. Comput Vis Media 2022;8(1):63–77. http://dx.doi.org/10.1007/S41095-021-0227-7.
[19] Huang Z, Xie H, Fukusato T, Miyata K. AniFaceDrawing: Anime portrait exploration during your sketching. In: Brunvand E, Sheffer A, Wimmer M, editors. ACM SIGGRAPH 2023 conference proceedings. ACM; 2023, p. 14:1–14:11. http://dx.doi.org/10.1145/3588432.3591548.
[20] Kim B, Wang O, Öztireli AC, Gross MH. Semantic segmentation for line drawing vectorization using neural networks. Comput Graph Forum 2018;37(2):329–38. http://dx.doi.org/10.1111/CGF.13365.
[21] Ito S, Takagi N, Sawai K, Masuta H, Motoyoshi T. Fast semantic segmentation for vectorization of line drawings based on deep neural networks. In: International conference on machine learning and cybernetics. IEEE; 2022, p. 231–6. http://dx.doi.org/10.1109/ICMLC56445.2022.9941326.
[22] Google Creative Lab. Quick, draw! dataset. 2017, https://github.com/googlecreativelab/quickdraw-dataset. (Accessed: 11 October 2024).
[23] Noris G, Hornung A, Sumner RW, Simmons M, Gross M. Topology-driven vectorization of clean line drawings. ACM Trans Graph 2013;32(1). http://dx.doi.org/10.1145/2421636.2421640.
[24] Guo Y, Zhang Z, Han C, Hu W, Li C, Wong T. Deep line drawing vectorization via line subdivision and topology reconstruction. Comput Graph Forum 2019;38(7):81–90. http://dx.doi.org/10.1111/CGF.13818.
[25] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition. IEEE Computer Society; 2016, p. 770–8. http://dx.doi.org/10.1109/CVPR.2016.90.
[26] Simo-Serra E, Iizuka S, Ishikawa H. Real-time data-driven interactive rough sketch inking. ACM Trans Graph 2018;37(4). http://dx.doi.org/10.1145/3197517.3201370.
[27] Milletari F, Navab N, Ahmadi S. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In: Fourth international conference on 3D vision, 3DV 2016, stanford, CA, USA, October 25-28, 2016. IEEE Computer Society; 2016, p. 565–71. http://dx.doi.org/10.1109/3DV.2016.79.
[28] Yan C, Vanderhaeghe D, Gingold Y. A benchmark for rough sketch cleanup. ACM Trans Graph 2020;39(6). http://dx.doi.org/10.1145/3414685.3417784.
[29] Gryaditskaya Y, Sypesteyn M, Hoftijzer JW, Pont S, Durand F, Bousseau A. OpenSketch: a richly-annotated dataset of product design sketches. ACM Trans Graph 2019;38(6). http://dx.doi.org/10.1145/3355089.3356533.
[30] Wang Z, Qiu S, Feng N, Rushmeier H, McMillan L, Dorsey J. Tracing versus freehand for evaluating computer-generated drawings. ACM Trans Graph 2021;40(4). http://dx.doi.org/10.1145/3450626.3459819.
[31] Xiao C, Su W, Liao J, Lian Z, Song Y-Z, Fu H. DifferSketching: How differently do people sketch 3D objects? ACM Trans Graph 2022;41(6). http://dx.doi.org/10.1145/3550454.3555493.
[32] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y, editors. 3rd International conference on learning representations, ICLR 2015, san diego, CA, USA, May 7-9, 2015, conference track proceedings. 2015.
[33] Puhachov I, Neveu W, Chien E, Bessmeltsev M. Keypoint-driven line drawing vectorization via PolyVector flow. ACM Trans Graph 2021;40(6). http://dx.doi.org/10.1145/3478513.3480529.