# SwiftSketch: A Diffusion Model for Image-to-Vector Sketch Generation

Anonymous Authors

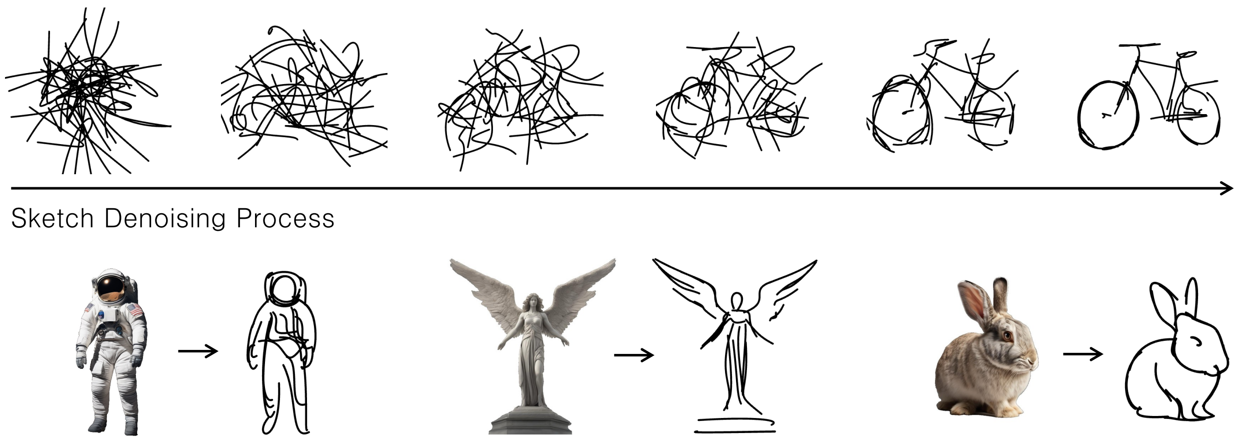


Figure 1. SwiftSketch is a diffusion model that generates vector sketches by denoising a Gaussian in stroke coordinate space (top). It generalizes effectively across diverse classes and takes under a second to produce a single high-quality sketch (bottom). https://swiftsketch.github.io/


## Abstract

*Recent advancements in large vision-language models have enabled highly expressive and diverse vector sketch generation. However, state-of-the-art methods rely on a time-consuming optimization process involving repeated feedback from a pretrained model to determine stroke placement. Consequently, despite producing impressive sketches, these methods are limited in practical applications. In this work, we introduce* SwiftSketch*, a diffusion model for image-conditioned vector sketch generation that can produce high-quality sketches in less than a second. SwiftSketch operates by progressively denoising stroke control points sampled from a Gaussian distribution. Its transformer-decoder architecture is designed to effectively handle the discrete nature of vector representation and capture the inherent global dependencies between strokes. To train SwiftSketch, we construct a* synthetic *dataset of image-sketch pairs, addressing the limitations of existing sketch datasets, which are often created by non-artists and lack professional quality. For generating these synthetic sketches, we introduce ControlSketch, a method that enhances SDS-based techniques by incorporating precise spatial control through a depth-aware ControlNet. We demonstrate that SwiftSketch generalizes across diverse concepts, efficiently producing sketches that combine high fidelity with a natural and visually appealing style.*


## 1. Introduction

In recent years, several works have explored the task of generating sketches from images, tackling both scene-level and object-level sketching [6, 46, 47, 55]. This task involves transforming an input image into a line drawing that captures its key features, such as structure, contours, and overall visual essence. Sketches can be represented as pixels or vector graphics, with the latter often preferred for their resolution independence, enhanced editability, and ability to capture sketches' sequential and abstract nature. Existing vector sketch generation methods often involve training a network to learn the distribution of human-drawn sketches [56]. However, collecting human-drawn sketch datasets is labor-intensive, and crowd-sourced contributors often lack artistic expertise, resulting in datasets that primarily feature amateur-style sketches (Fig. 2, left). On the other hand, sketch datasets created by professional designers or artists are typically limited in scale, comprising only a few hundred samples, and are often restricted to specific domains, such as portraits or product design (Fig. 2, right). Therefore, existing data-driven sketch generation methods are often restricted to specific domains or reflect a non-professional style present in the training data.

With recent advancements in Vision-Language Models (VLMs) [57], new approaches have emerged in the sketch domain, shifting sketch generation from reliance on human-drawn datasets to leveraging the priors of pretrained models [11, 46, 47, 54]. These methods generate professional-looking sketches by optimizing parametric curves to represent an input concept, guided by the pretrained VLM. However, they have a significant drawback: The generation process depends on repeated feedback (backpropagation) from the pretrained model, which is inherently time-consuming – often requiring from several minutes to over an hour to produce a single sketch. This makes these approaches impractical for interactive applications or for tasks that require large-scale sketch data generation.

In this work, we introduce *SwiftSketch*, a diffusion-based object sketching method capable of generating high-quality vector sketches in under a second per sketch. SwiftSketch can generalize across a wide range of concepts and produce sketches with high fidelity to the input image (see Figure 1).

Inspired by recent advancements in diffusion models for non-pixel data [29, 43, 44], we train a diffusion model that learns to map a Gaussian distribution in the space of stroke coordinates to the data distribution (see Figure 1, top). To address the discrete nature of vector graphics and the complex global topological relationships between shapes, we employ a transformer-decoder architecture with self- and cross-attention layers, trained to reconstruct ground truth sketches in both vector and pixel spaces. The image condition is integrated into the generation process through the cross-attention mechanism, where meaningful features are first extracted from the input image using a pretrained CLIP image encoder [36].

With the lack of available professional-quality paired vector sketch datasets, we construct a *synthetic* dataset to train our network. The input images are generated with SDXL [33], and their corresponding vector sketches are produced with a novel optimization-based technique we introduce called *ControlSketch*. ControlSketch enhances the SDS loss [34], commonly used for text-conditioned generation, by integrating a depth ControlNet [58] into the loss, enabling object sketch generation with spatial control. Our dataset comprises over 35,000 high-quality vector sketches across 100 classes and is designed for scalability. We demonstrate SwiftSketch's capability to generate high-quality vector sketches of diverse concepts, balancing fidelity to input images and the abstract appearance of natural sketches.

## 2. Related Work

**Sketch Datasets** Existing sketch datasets are primarily composed of human-drawn sketches, and are designed to accomplish different sketching tasks. Class-conditioned datasets [10, 16] are particularly common, with the largest being the QuickDraw dataset [16], containing 50 million sketches spanning 345 categories. Datasets of image-referenced sketches cover a spectrum of styles, including image trace and contours [1, 10, 25, 50], or more abstract but still fine-grained depictions [14, 40], and very abstract sketches [31]. These large-scale datasets are often created by non-artists. Efforts have been made to collect sketches from professionals [3, 15, 17, 53], but these datasets are often smaller in scale, and are limited to specific domains like portraits [3] or household items [15]. These constraints make them unsuitable for training generative models that can generalize broadly to diverse concepts.

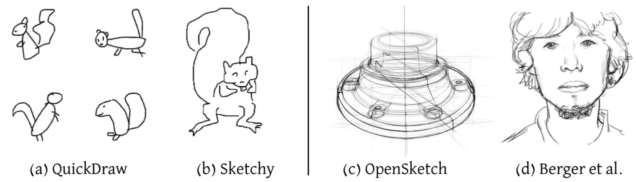


Figure 2. Amateur vs. Professional Sketches. (a) QuickDraw [16] and (b) Sketchy [40] are large-scale datasets, with Sketchy offering more fine-grained sketches, though both exhibit non-professional style. (c) OpenSketch [15] and (d) Berger *et al*. [3] contain professional sketches but are limited in scale and focus on specific domains.

**Data-Driven Sketch Generation** These datasets have facilitated data-driven approaches for various sketch-related tasks [56]. Multiple generative frameworks and architectures have been explored for vector sketch generation, including RNNs [16], BERT [27], Transformers [4, 37], CNNs [8, 23, 42], LSTMs [35, 42], GANs [45], reinforcement learning [30, 60], and diffusion models [49]. However, these methods are fundamentally designed to operate in a class-conditioned manner, restricting their ability to generate sketches to only the classes included in the training data. Additionally, they rely on crowd-sourced datasets which contain non-professional sketches, restricting their ability to handle more complex or artistic styles. On the other hand, existing works for generating more professionally looking sketches are either restricted to specific domains [28] or can only generate sketches in pixel space [6, 25]. Note that image-to-sketch generating can be formulated as a style transfer task, with recent works that employ the text-to-image diffusion priors achieving highly artistic results with high fidelity [12, 18, 48], however, all of these works also operate only in pixel space. In contrast, we focus on vector sketches due to their resolution independence, smooth and clean appearance, control over abstraction, and editable nature.

**VLMs for Vector Sketches** To reduce reliance on existing vector datasets, recent research leverages the rich priors of large pre-trained vision-language models (VLMs) in a zero-shot manner. Early methods [11, 46, 47] utilize CLIP [36] as the backbone for image- and text-conditioned generation. These approaches iteratively optimize a randomly initialized set of strokes using a differentiable renderer [26] to bridge the gap between vector and pixel representations. More recently, text-to-image diffusion models [38] have been employed as backbones, with the SDS loss [34] used to guide the optimization process, achieving superior results [22, 54, 55]. However, the use of the SDS loss has so far been limited to text-conditioned generation. While these approaches yield highly artistic results across diverse concepts, they are computationally expensive, relying on iterative backpropagation.

**Diffusion Models for Non-Pixel Data** Diffusion models have emerged as a powerful generative framework, extending their impact beyond traditional pixel-based data. Recent research demonstrates their versatility across diverse domains, including tasks such as human motion synthesis [43], 3D point cloud generation [21, 29], and object detection reframed as a generative process [7]. Some prior works have explored diffusion models for vector graphics synthesis. VecFusion [44] uses a two-stage diffusion process for vector font generation but its architecture and vector representation are highly complex and specialized for fonts, limiting adaptability to other vector tasks. SketchKnitter [49] and Ashcroft *et al.* [2] generate vector sketches using a diffusion-based model trained on the QuickDraw and Anime-Vec10k dataset, but without conditioning on images or text inputs.

## 3. Preliminaries

**Diffusion Models** Diffusion models [20, 41] are a class of generative models that learn a distribution by gradually denoising a Gaussian. Diffusion models consist of a forward process $q(x_t|x_{t-1})$ that progressively noises data samples $x_0 \sim p_{data}$ at different timesteps $t \in [1, T]$, and a backward or reverse process $p(x_{t-1}|x_t)$ that progressively cleans the noised signal. The reverse process is the generative process and is approximate with a neural network $\epsilon_\theta(x_t, t)$. During training, a noised signal at differnet timesteps is derived from a sample $x_0$ as follows:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ is called the noise scheduler. The common approach for training the model is with the following simplified objective:

$$L_{\text{simple}} = \mathbb{E}_{x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0,\mathbf{I}), t \sim \mathcal{U}(1,T)} \left\| \epsilon - \epsilon_\theta(x_t, t) \right\|^2. \tag{2}$$

At inference, to generate a new sample, the process starts with a Gaussian noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and the denoising network is applied iteratively for $T$ steps, yielding a final sample $x_0$.

**SDS Loss** The Score Distillation Sampling (SDS) loss [34] is used to extract signals from a pretrained text-to-image diffusion model to optimize a parametric representation. For vector graphics, the parameters $\phi$ defining an SVG can be optimized using the SDS loss to represent a desired textual concept. A differentiable rasterizer [26] rasterize $\phi$ into a pixel image $x$, which is then noised to produce $x_t$ at a sampled timestep $t$. This noised image, conditioned on a text prompt $c$, is passed through the pretrained diffusion model, $\epsilon_\theta(x_t, t, c)$. The deviation of the diffusion loss in Eq. (2) is used to approximate the gradients of the initial image synthesis model's parameters, $\phi$, to better align its outputs with the conditioning prompt. Specifically, the gradient of the SDS loss is defined as:

$$\nabla_\phi \mathcal{L}_{SDS} = \left[ w(t)(\epsilon_\theta(x_t, t, y) - \epsilon) \frac{\partial x}{\partial \phi} \right], \tag{3}$$

where $w(t)$ is a constant that depends on $\alpha_t$. This optimization process iteratively adjusts the parametric model.

## 4. Method

Our method consists of three key components: (1) ControlSketch, an optimization-based technique for generating high-quality vector sketches of input objects; (2) a synthetic paired image-sketch dataset, created using ControlSketch; and (3) SwiftSketch, a diffusion model trained on our dataset for efficient sketch generation.

### 4.1. ControlSketch

Given an input image $I$ depicting an object, our goal is to generate a corresponding sketch $S$ that maintains high fidelity to the input while preserving a natural sketch-like appearance. Following common practice in the field, we define $S$ as a set of $n$ strokes $\{s_i\}_{i=1}^n$, where each stroke is a two-dimensional cubic Bézier curve: $s_i = \{p_j^i\}_{j=1}^4 = \{(x_j, y_j)^i\}_{j=1}^4$. We optimize the set of strokes using the standard SDS-based optimization pipeline, as described in Section 3, with two key enhancements: an improved stroke initialization process and the introduction of spatial control. Our process rely on the image's attention map $I_{attn}$, depth map $I_{depth}$, and caption $y$, extracted using DDIM inversion [41], MiDaS [5], and BLIP2 [24] respectively. While previous approaches [47, 54] sample initial stroke locations based on the image's attention map, we observe that this method often results in missing areas in the output sketch, especially when spatial control is applied. To address this, we propose an enhanced initialization method (see Fig. 3, left) that ensures better coverage. We divide the object area into $k = 6$ equal-area regions (Fig. 3c), using a weighted K-Means method that accounts for both attention weights and pixel locations. We distribute $\frac{n}{2}$ points equally across the regions, while the remaining $\frac{n}{2}$ points are allocated proportionally to the average attention value in each region. This means that more points are assigned to regions with higher

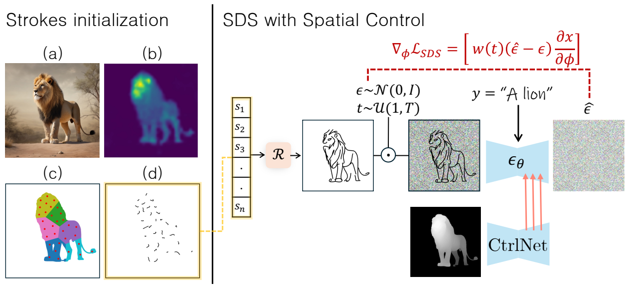


Figure 3. ControlSketch Pipeline. Left: The object area is divided into $k$ regions (c), with $n$ points distributed based on attention values from (b) while ensuring a minimum allocation per region. (d) The initial strokes are derived from these points. Right: The initial strokes are iteratively optimized to form the sketch. At each iteration, the rasterized sketch is noised based on $t$ and $\epsilon$ and fed into a diffusion model with a depth ControlNet conditioned on the image's depth and caption $y$. The predicted noise $\hat{\epsilon}$ is used for the SDS loss.

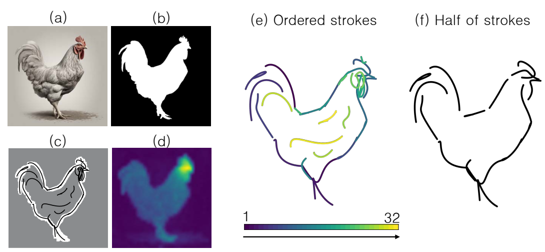


Figure 4. (a) Input image. (b) Object mask. (c) The object's contour is extracted from the mask using morphological operations, and sketch pixels that intersect with the contour are given higher weight. (d) Attention map. (e) We sort the strokes based on a combination of contour intersection count and attention score. (f) A visualization of the first 16 strokes in the ordered sketch, demonstrating the effectiveness of our sorting scheme.

attention. Within each region, the points are evenly spaced to further ensure good coverage. This process determines the location of the initial set of strokes' control points to be optimized, as demonstrated in Figure 3d.

The stroke optimization process is depicted in Figure 3, right. At each optimization step, the rasterized sketch $\mathcal{R}(\{s_i\}_{i=1}^n)$ is noised based on $t$ and $\epsilon$, then fed into a depth ControlNet text-to-image diffusion model [58]. The model predicts the noise $\hat{\epsilon}$ conditioned on the caption $y$ and the depth map $I_{depth}$. We balance the weighting between the spatial and textual conditions to achieve an optimal trade-off between "semantic" fidelity, derived from $y$ (ensuring the sketch is recognizable), and "geometric" fidelity, derived from $I_{depth}$, which governs the accuracy of the spatial structure.

### 4.2. The ControlSketch Dataset

We utilize ControlSketch to generate a paired image-vector sketch dataset. Each data sample comprises the set $\{I, I_{attn}, I_{depth}, I_{mask}, S, c, y\}$, which includes, respectively, the image, its attention map, depth map, and object mask, along with the corresponding vector sketch of the object, class label, and caption. To generate the images, we utilize SDXL [33], along with a prompt template designed to produce images for each desired class $c$ (an example of a generated image for the class "lion" is shown in Figure 3a). We then apply ControlSketch on the masked images to generate the corresponding vector sketches. Additional details are provided in the supplementary. Optimization-based methods, such as ControlSketch, do not impose an inherent stroke ordering. Learning an internal stroke order enables the generation of sketches with varying levels of abstraction by controlling the number of strokes generated. Thus, we propose a heuristic stroke-sorting scheme that prioritizes contour strokes and those depicting salient regions (illustrated in Figure 4). Consequently, each vector sketch $S$ is represented as an ordered sequence of strokes $(s_1, \ldots, s_n)$.

### 4.3. SwiftSketch

We utilize the ControlSketch dataset to train a generative model $M_\theta$ that learns to efficiently produce a vector sketch from an input image $I$. We define $M_\theta$ as a transformer decoder to account for the discrete and long-range dependencies inherent in vector sketches. The training of $M_\theta$ follows the standard conditional diffusion framework, as outlined in Section 3, with task-specific modifications to address the characteristics of vector data and the image-to-sketch task. In our case, the model learns to denoise the set of $(x, y)$ coordinates that define the strokes in the sketch.

The training process is depicted in Figure 5. At each iteration, a pair $(I, S^0)$ is sampled from the dataset, where $S^0 \in R^{2\times 4\times n}$ is the clean sketch in vector representation, and $\mathcal{R}(S^0)$ denotes the corresponding rasterized sketch in pixel space, with $\mathcal{R}$ being a differentiable rasterizer [26]. The image $I$ is processed using a pretrained CLIP ResNet model [36], where features are extracted from its fourth layer, recognized for effectively capturing both geometric and semantic information [47]. These features are then refined through a lightweight CNN to enhance learning and align dimensions for compatibility with $M_\theta$. This process yields the image embedding $I_e$. At each iteration, we sample a timestep $t \sim \mathcal{U}(1, T)$ and noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ to define $S^t$:

$$S^t = \sqrt{\bar{\alpha}_t} S^0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \quad (4)$$

where $\bar{\alpha}_t$ is the noise scheduler as a function of $t$. As illustrated in Figure 5, $S^t$ represents a noised version of $S^0$ in vector space, with the level of noise determined by the timestep $t$. The control points $\{s_1^t, \ldots s_n^t\}$ are fed into

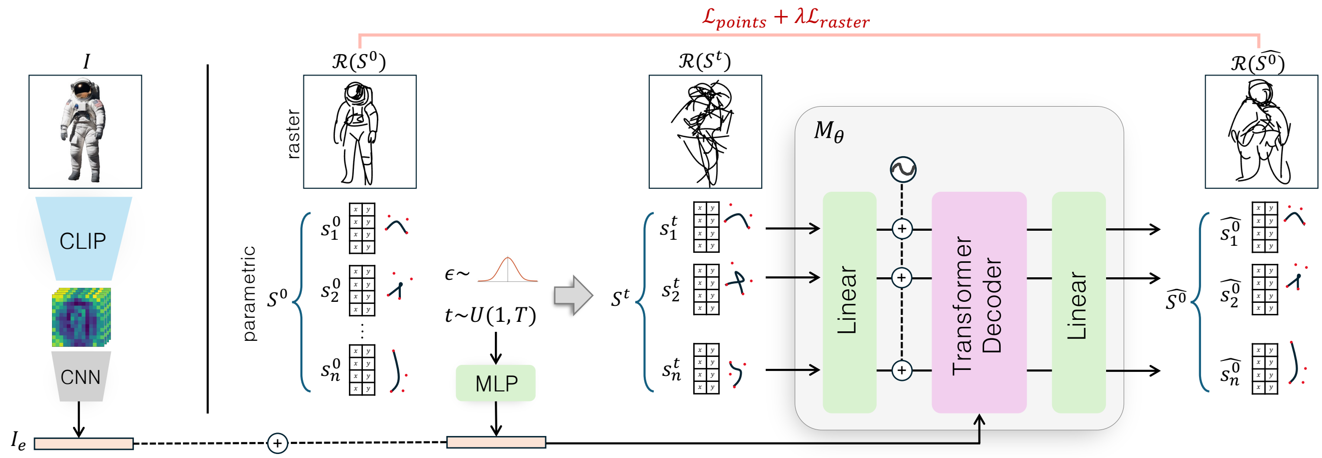


Figure 5. SwiftSketch Training Pipeline. At each training iteration, an image $I$ is passed through a frozen CLIP image encoder, followed by a lightweight CNN, to produce the image embedding $I_e$. The corresponding vector sketch $S^0$ is noised based on the sampled timestep $t$ and noise $\epsilon$, forming $S^t$ (with $\mathcal{R}(S^t)$ illustrating the rasterized noised sketch, which is not used in training). The network $M_\theta$, a transformer decoder, receives the noised signal $S^t$ and is tasked with predicting the clean signal $\hat{S}^0$, conditioned on the image embedding $I_e$ and the timestep $t$ (fed through the cross-attention mechanism). The network is trained with two loss functions: one based on the distance between the control points and the other on the similarity of the rasterized sketches.

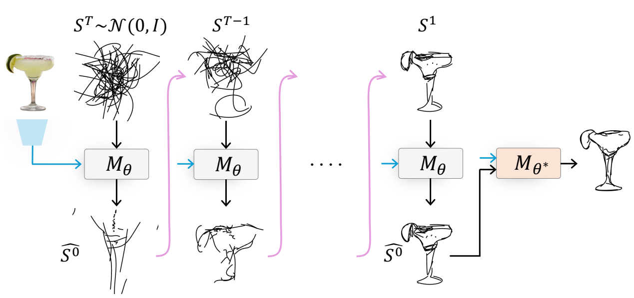


Figure 6. Inference Process. Starting with randomly sampled Gaussian noise $S^T \sim \mathcal{N}(0, \mathbf{I})$, the model $M_\theta$ predicts the clean sketch $\hat{S}^0 = M_\theta(S^t, t, I_e)$ at each step $t$, which is then re-noised to $S^{t-1}$. This iterative process is repeated for $T$ steps and is followed by a final feed-forward pass through a refinement network, $M_{\theta^*}$, which is a trainable copy of $M_\theta$, specifically trained to correct very small residual noise.

the network $M_\theta$, where they are first encoded via a linear layer (depicted in green), and combined with a standard positional embedding before being passed through the transformer decoder (in pink), which consists of 8 layers of cross-attention and self-attention. The encoded timestep $t$ and image features $I_e$ are fed into the transformer through the cross-attention mechanism. The decoder output is projected back to the original points dimension through a linear layer, yielding the prediction $M_\theta(S_t, t, I_e) = \hat{S}^0$.

We train $M_\theta$ with two training objectives $\mathcal{L}_{\text{points}}$ and $\mathcal{L}_{\text{raster}}$, applied on both the vector and raster representation of the sketch:

$$\begin{aligned} \mathcal{L}_{\text{points}} &= \|S^0 - \hat{S}^0\|_1 = \sum_{i=1}^{n} \|s_i^0 - \hat{s_i^0}\|_1, \\ \mathcal{L}_{\text{raster}} &= LPIPS\left(\mathcal{R}(S^0), \mathcal{R}(\hat{S}^0)\right), \end{aligned} \tag{5}$$

where $\mathcal{L}_{\text{points}}$ is defined by the $L_1$ distance between the sorted control points of the ground truth sketch $S^0$ and the predicted sketch $\hat{S}^0$, and $\mathcal{L}_{\text{raster}}$ is the LPIPS distance [59] between the rasterized sketches. $\mathcal{L}_{\text{points}}$ encourages per-stroke precision, while $\mathcal{L}_{\text{raster}}$ encourages the generated sketch to align well with the overall structure of the ground truth sketch. Together, our training loss is: $\mathcal{L} = \mathcal{L}_{\text{points}} + \lambda\mathcal{L}_{\text{raster}}$, with $\lambda = 0.2$.

As is often common, to apply classifier-free guidance [19] at inference, we train $M_\theta$ to learn both the conditioned and the unconditioned distributions by randomly setting $I = \emptyset$ for 10% of the training steps.

The inference process is illustrated in Figure 6. The model, $M_\theta$, generates a new sketch by progressively denoising randomly sampled Gaussian noise, $S^T \sim \mathcal{N}(0, \mathbf{I})$. At each step $t$, $M_\theta$ predicts the clean sketch $\hat{S}^0 = M_\theta(S^t, t, I_e)$, conditioned on the image embedding $I_e$ and time step $t$. The next intermediate sketch, $S^{t-1}$, is derived from $\hat{S}^0$ using Equation (4). This process is repeated for $T$ steps. We observe that the final output sketches from the denoising process may retain slight noise. This is likely because the network prioritizes learning to clean heavily noised signals during training, while small inaccuracies in control point locations have a smaller impact on the loss function, leading to reduced precision at finer timesteps. To address this, we introduce a refinement stage, where

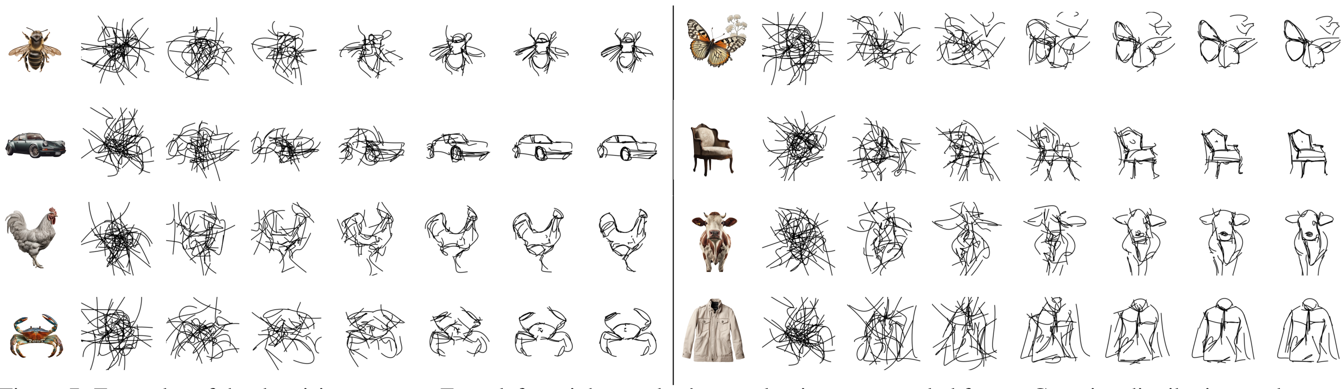

Figure 7. Examples of the denoising process. From left to right: strokes' control points are sampled from a Gaussian distribution, and our network progressively refines the signal to generate a sketch.

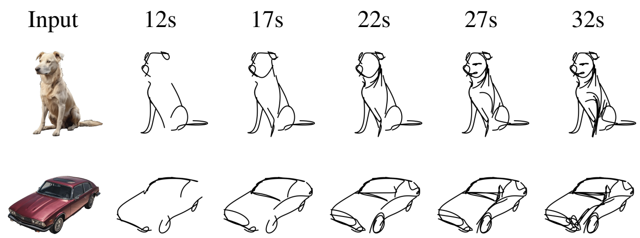


Figure 8. Stroke Order Visualization. Generated sketches are visualized progressively, with the stroke count shown on top. Early strokes capture the object's contour and key features, while later strokes add finer details.

a learned copy of our network, $M_{\theta^*}$, is fine-tuned to perform an additional cleaning step. This refinement network is trained in a manner similar to the original model, with the objective of denoising a slightly noised sketch, conditioned on the same input image features, while the timestep condition is fixed at 0. More details are provided in the supplementary. This refinement stage is inspired by similar strategies employed in the pixel domain [33, 39], where additional processing steps are used to improve the quality and resolution of generated images. As illustrated in Figure 6, after the final denoising step of $M_\theta$ is applied, $\hat{S}^0$ is passed through $M_{\theta^*}$ to perform the additional refinement.

### 4.4. Implementation Details

ControlSketch requires approximately 2,000 steps to converge, taking around 10 minutes on a standard RTX3090 GPU. SwiftSketch is trained with $T = 50$ noising steps, to support fast generation. To encourage the model to focus on fine details, we adjust the noise scheduler to perturb the signal more subtly for small timesteps compared to the cosine noise schedule proposed in [32]. The model is trained on images from 15 classes, with 1,000 samples per class. The training process spans 400K steps, requiring approximately six days on a single A100 GPU. At inference, we use a guidance scale of 2.5. Our synthetic dataset includes an additional 200 test samples for the 15 training classes, as well as 85 additional object categories, each with 200 samples. Additional implementation details, as well as detailed class labels and dataset visualizations are provided in the supplementary material.

## 5. Results

We begin by showcasing SwiftSketch's ability to generate high-quality vector sketches for a diverse set of input images. SwiftSketch successfully generalizes to unseen images within the training categories (Figure 11), creating sketches that depict the input images well while demonstrating a plausible and detailed appearance. On images of unseen categories that pose greater challenges, SwiftSketch effectively captures the essential features of the input images, producing abstract yet faithful representations (Figure 12). Notably, all sketches are provided in vector format, and are generated in just 50 diffusion steps, followed by a single refinement step, with the entire process taking less than one second. In Figure 7, we illustrate the denoising steps of the generation process, starting from a Gaussian distribution and progressively refining towards the data distribution. In Figure 8, we demonstrate the ability of our method to create level-of-abstraction using our ordered stroke technique. We visualize the progressive addition of strokes in the sequence they appear in the output SVG file. Note how the first strokes already convey the intended concept effectively. Additional results of both SwiftSketch and ControlSketch are available in the supplementary.

### 5.1. Comparisons

We evaluate the performance of SwiftSketch and ControlSketch with respect to state-of-the-art methods for image-to-sketch generation, including Photo-Sketching [25], Chan *et al*. [6], InstantStyle [48], and CLIPasso [47]. InstantStyle

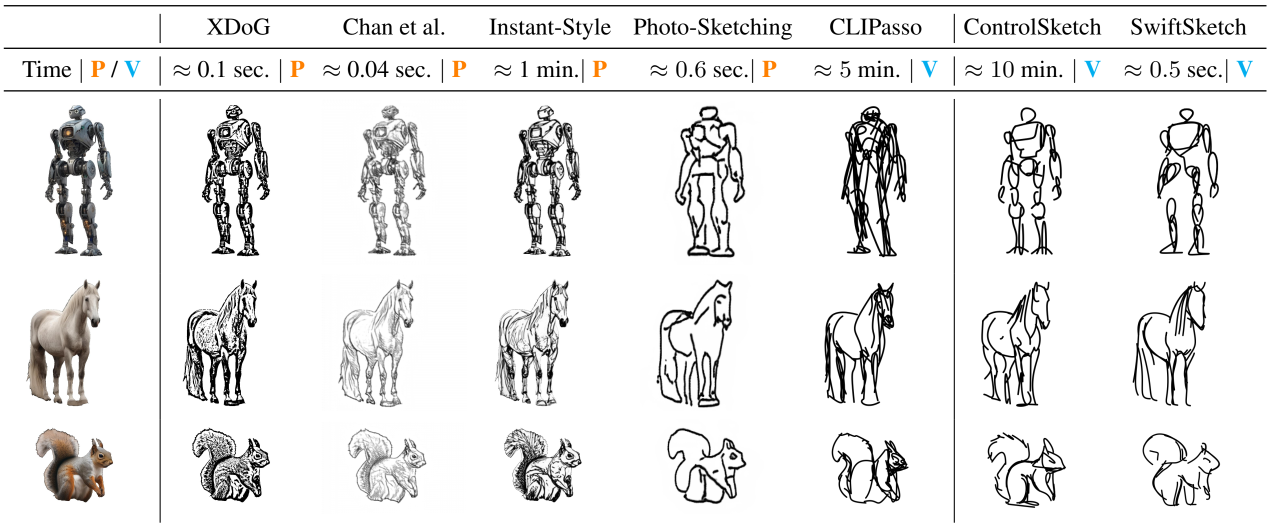


Figure 9. Qualitative Comparison. Input images are shown on the left, with the time required to produce a single sketch and whether the sketches are in Pixel or Vector format indicated at the top. From left to right, the sketches are generated using XDoG [52], PhotoSketching [25], Chan et al. [6] (in anime style), InstantStyle [48], and CLIPasso [47]. On the right are the resulting sketches from our proposed methods, ControlSketch and SwiftSketch.

is applied with a sketch image as the style reference. Figure 9 shows representative results from each method, with XDoG [52], a classic edge detection technique, shown on the left as a baseline. The sketches of Chan *et al.* and InstantStyle are detailed and align well with the overall structure of the input images. However note that they closely follow the edge maps shown on the left. The sketches of Photo-Sketching (fifth column) are more abstract, but can fail to effectively capture the images' content in a natural way. While these approaches are efficient, producing sketches in less than a minute, they focus on generating *raster* sketches. In contrast, our method produces vector sketches, which are resolution-independent, easily editable, and exhibit a smooth, clean style. CLIPasso (sixth column) generates vector sketches that achieve a good balance between fidelity and semantics. However, it is significantly slower, requiring 5 minutes to produce a single sketch, and it may introduce artifacts, such as the noisy overlapping strokes observed in the robot example. ControlSketch (seventh column) produces high-fidelity sketches that remain abstract, smooth, and natural, effectively depicting the input images while avoiding artifacts. However, it is even slower than CLIPasso, as SDS-based methods generally require more time to converge, making it impractical for interactive applications. SwiftSketch, shown in the rightmost column, successfully learns the data distribution from ControlSketch samples, enabling it to produce sketches that approach the quality of optimization-based techniques but in real time. Additional results are available in the supplementary material.

**Quantitative Evaluation** We sample 4,000 images from our dataset (2,000 from our test set of categories seen during training and 2,000 from unseen categories) and additional 2,000 images from the SketchyCOCO [14] dataset to assess generalization on external data. Each set consists of 10 randomly selected categories with 200 images per category. Following common practice in the field, we use the CLIP zero-shot classifier [36] to assess class-level recognition, MS-SSIM [51] for image-sketch fidelity following the settings proposed in CLIPascene [46], and DreamSim [13]. The results are presented in Table 1, where scores for each data type are reported separately, with human sketches from the SketchyCOCO dataset included as a baseline. Chan *et al.* and InstantStyle achieve the highest scores across most metrics due to their highly detailed sketches, which closely resemble the image's edge map. This level of detail ensures that their sketches are both easily recognizable as depicting the correct class (as indicated by the CLIP score) and exhibit high fidelity (as reflected in other measurements). The results show that SwiftSketch generalizes well to test set images from seen categories, as evidenced by its similar scores to ControlSketch (which serves as the ground truth in our case). However, its performances decrease for unseen categories, particularly in class-based recognition. This is especially apparent on the SketchyCOCO dataset, which is highly challenging due to its low-resolution images and difficult lighting conditions. It is important to note that SwiftSketch is trained on only 15 image categories due to limited resources, suggesting that more extensive training could improve its generalization capabilities.

The results demonstrate that ControlSketch produces

Table 1. Quantitative Comparison of Sketch Generation Methods. The scores for Top-1 and Top-3 CLIP recognition accuracy, MS-SSIM, and DreamSim are presented. Results are based on 6,000 random samples: 2,000 from our dataset's test set, 2,000 from unseen categories in our dataset, and 2,000 from the external SketchyCOCO dataset [14]. The best scores in each column are marked in bold.

| | | CLIP Top-1 ↑ | | | CLIP Top-3 ↑ | | | MS-SSIM ↑ | | | DreamSim ↓ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time | Seen | Unseen | Exter. | Seen | Unseen | Exter. | Seen | Unseen | Exter. | Seen | Unseen | Exter. |
| Human [14] | – | – | – | 0.85 | – | – | 0.93 | – | – | 0.16 | – | – | 0.66 |
| Chan *et al*. (Anime) [6] | ≈ 0.04 sec. | 0.94 | **0.99** | 0.87 | **0.99** | **0.99** | 0.96 | 0.79 | 0.77 | 0.45 | **0.32** | **0.43** | **0.45** |
| Chan *et al*. (Contour) [6] | ≈ 0.04 sec. | 0.96 | 0.92 | 0.80 | 0.98 | 0.95 | 0.91 | 0.77 | 0.73 | 0.49 | 0.51 | 0.59 | 0.54 |
| InstantStyle [48] | ≈ 1 min. | 0.97 | **0.99** | – | **0.99** | **0.99** | – | **0.89** | **0.90** | – | 0.36 | 0.44 | – |
| Photo-Sketching [25] | ≈ 0.6 sec. | 0.90 | 0.81 | 0.65 | 0.95 | 0.87 | 0.78 | 0.61 | 0.55 | 0.40 | 0.58 | 0.65 | 0.63 |
| CLIPasso [47] | ≈ 5 min. | **0.98** | 0.97 | 0.88 | **0.99** | **0.99** | 0.95 | 0.65 | 0.60 | 0.52 | 0.48 | 0.57 | 0.57 |
| ControlSketch | ≈ 10 min. | 0.97 | 0.97 | **0.91** | **0.99** | **0.99** | **0.97** | 0.68 | 0.63 | **0.53** | 0.52 | 0.59 | 0.60 |
| SwiftSketch | ≈ 0.5 sec. | 0.95 | 0.70 | 0.56 | 0.98 | 0.82 | 0.70 | 0.62 | 0.56 | 0.47 | 0.53 | 0.66 | 0.64 |

sketches that are both highly recognizable and of high fidelity, outperforming alternative methods, particularly on the SketchyCOCO dataset. To further highlight the advantages of ControlSketch over CLIPasso, we conduct a two-alternative forced-choice (2AFC) perceptual study with 40 participants. Each participant was shown pairs of sketches generated by the two methods (presented in random order) alongside the input image and asked to choose the sketch they perceived to be of higher quality. The study included 24 randomly selected sketches from both our dataset and SketchyCOCO, spanning 24 object classes. Participants rated sketches generated by ControlSketch as higher quality in 89% of cases. Examples of sketches presented in the user study are shown in Figure 13.

## 6. Ablation

We evaluate the contribution of SwiftSketch's main components by systematically removing each one and retraining the network. Specifically, we examine the impact of excluding the LPIPS loss, the L1 loss, and the sorting technique, as well as the effect of incorporating the refinement network. The results are summarized in Table 2, where "Full" represents our complete diffusion pipeline prior to refinement, and "+Refine" denotes the inclusion of the refinement stage. Notably, removing the L1 loss results in a significant drop in performance, highlighting its essential role in the training process. Excluding the LPIPS loss negatively impacts performance, particularly in unseen classes. The metrics indicate comparable performance in the absence of the sorting stage. While the resulting sketches may appear visually similar, the sorting stage is crucial for supporting varying levels of abstraction. Although the network can be trained without this stage and still achieve reasonable results, learning an internal stroke order provides a foundation for training across abstraction levels, where sketches implicitly encode the importance of strokes. The refinement stage enhances recognizability, especially in unseen categories where the output sketches from the diffusion process are noisier. We further illustrate the impact of the refinement network in Figure 14, with additional results provided in the supplementary.

Table 2. Ablation Study. We systematically remove each component in our pipeline and retrain our network. Scores are computed on 4000 sketches in total: 2000 from seen categories and 2000 from unseen categories.

| | CLIP Top-1 ↑ | | CLIP Top-3 ↑ | | MS-SSIM ↑ | | DreamSim ↓ | |
|---|---|---|---|---|---|---|---|---|
| | Seen | Unseen | Seen | Unseen | Seen | Unseen | Seen | Unseen |
| w/o LPIPS | 0.88 | 0.36 | 0.94 | 0.49 | 0.58 | 0.52 | 0.58 | 0.72 |
| w/o L1 | 0.03 | 0.01 | 0.06 | 0.03 | 0.24 | 0.22 | 0.80 | 0.84 |
| w/o Sort | 0.93 | 0.63 | 0.97 | 0.76 | **0.62** | **0.57** | 0.55 | 0.68 |
| Full | 0.94 | 0.61 | 0.97 | 0.73 | **0.62** | **0.57** | 0.54 | 0.68 |
| +Refine | **0.95** | **0.70** | **0.98** | **0.82** | **0.62** | 0.56 | **0.53** | **0.66** |

## 7. Limitations and Future Work

While SwiftSketch can generate vector sketches from images efficiently, it comes with limitations. First, although SwiftSketch performs well on seen categories, as evidenced by our evaluation, its performance decreases for unseen categories. This is particularly apparent in categories that differ significantly from those seen during training (e.g., non-human or non-animal objects). Failure cases often exhibit a noisy appearance or are entirely unrecognizable, such as the carrot in Figure 10. Expanding the number of training categories in future work could enhance the model's generalization. Second, our refinement stage, which is meant to fix the noisy appearance, might over-simplify the sketches, resulting in lost details such as the nose and eyes of the cow in Fig. 10. Lastly, in the scope of this paper, we trained SwiftSketch on sketches with a fixed number of strokes (32). Extending the training to sketches with varying numbers of strokes, spanning multiple levels of abstraction, presents an exciting direction for future research. Our transformer-decoder architecture is inherently suited for such an extension, and our results show that the network can capture essential features from sorted sketches, highlighting its potential to effectively handle more challenging levels of abstraction.

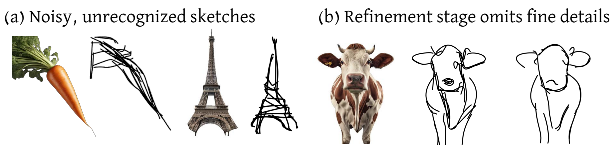


Figure 10. Limitations. (a) Sketches may appear unrecognizable (e.g., carrot) or noisy (e.g., Eiffel Tower). (b) The refinement stage can lead to the loss of fine details, such as the cow's nose and eye.

## 8. Conclusions

We introduced SwiftSketch, a method for object sketching capable of generating plausible *vector* sketches in under a second. SwiftSketch employs a diffusion model with a transformer-decoder architecture, generating sketches by progressively denoising a Gaussian distribution in the space of stroke control points. To address the scarcity of professional-quality paired vector sketch datasets, we constructed a synthetic dataset spanning 100 classes and over 35,000 sketches. This dataset was generated using ControlSketch, an improved SDS-based sketch generation method enhanced with a depth ControlNet for better spatial control. We demonstrated both visually and numerically that ControlSketch produces high-quality, high-fidelity sketches and that SwiftSketch effectively learns the data distribution of ControlSketch, achieving high-quality sketch generation while reducing generation time from approximately 10 minutes to 0.5 seconds. We believe this work represents a meaningful step toward real-time, high-quality vector sketch generation with the potential to enable more interactive processes. Additionally, our extensible dataset construction process will be made publicly available to support future research in this field.

## References

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011. 2

[2] Alexander Ashcroft, Ayan Das, Yulia Gryaditskaya, Zhiyu Qu, and Yi-Zhe Song. Modelling complex vector drawings with stroke-clouds. In *The Twelfth International Conference on Learning Representations*, 2024. 3

[3] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Jeanne Carter, and Jessica K. Hodgins. Style and abstraction in portrait sketching. *ACM Transactions on Graphics (TOG)*, 32:1 – 12, 2013. 2

[4] Kumar Bhunia, Umar Ayan Das, Riaz Muhammad, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. Edinburgh research explorer pixelor: A competitive sketching ai agent. so you think you can sketch? 2020. 2

[5] Reiner Birkl, Diana Wofk, and Matthias Müller. Midas v3.1 – a model zoo for robust monocular relative depth estimation, 2023. 3

[6] Caroline Chan, Frédo Durand, and Phillip Isola. Learning to generate line drawings that convey geometry and semantics. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7905–7915, 2022. 1, 2, 6, 7, 8, 4

[7] Shoufa Chen, Pei Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19773–19786, 2022. 3

[8] Yajing Chen, Shikui Tu, Yuqi Yi, and Lei Xu. Sketch-pix2seq: a model to generate sketches of multiple categories. *ArXiv*, abs/1709.04121, 2017. 2

[9] Chenxwh. BRIA Background Removal v1.4 Model, 2025. 1

[10] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on Graphics (TOG)*, 31:1 – 10, 2012. 2, 1

[11] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *ArXiv*, abs/2106.14843, 2021. 1, 2

[12] Yarden Frenkel, Yael Vinker, Ariel Shamir, and Daniel Cohen-Or. Implicit style-content separation using b-lora, 2024. 2

[13] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. In *Advances in Neural Information Processing Systems*, volume 36, pages 50742–50768, 2023. 7

[14] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. Sketchycoco: Image generation from freehand scene sketches. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5173–5182, 2020. 2, 7, 8, 1

[15] Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia C. Pont, Frédo Durand, and Adrien Bousseau. Opensketch. *ACM Transactions on Graphics (TOG)*, 38:1 – 16, 2019. 2

[16] David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. 2, 1

[17] Yue Han, Jiangning Zhang, Junwei Zhu, Xiangtai Li, Yanhao Ge, Wei Li, Chengjie Wang, Yong Liu, Xiaoming Liu, and Ying Tai. A generalist facex via learning unified facial representation. *ArXiv*, abs/2401.00551, 2023. 2

[18] Amir Hertz, Andrey Voynov, Shlomi Fruchter, and Daniel Cohen-Or. Style aligned image generation via shared attention. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4775–4785, 2024. 2

[19] Jonathan Ho. Classifier-free diffusion guidance. *ArXiv*, abs/2207.12598, 2022. 5

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Sys-*

Figure 11. Sketches generated by SwiftSketch for seen categories, using input images not included in the training data.

Figure 12. Sketches generated by SwiftSketch for unseen categories.

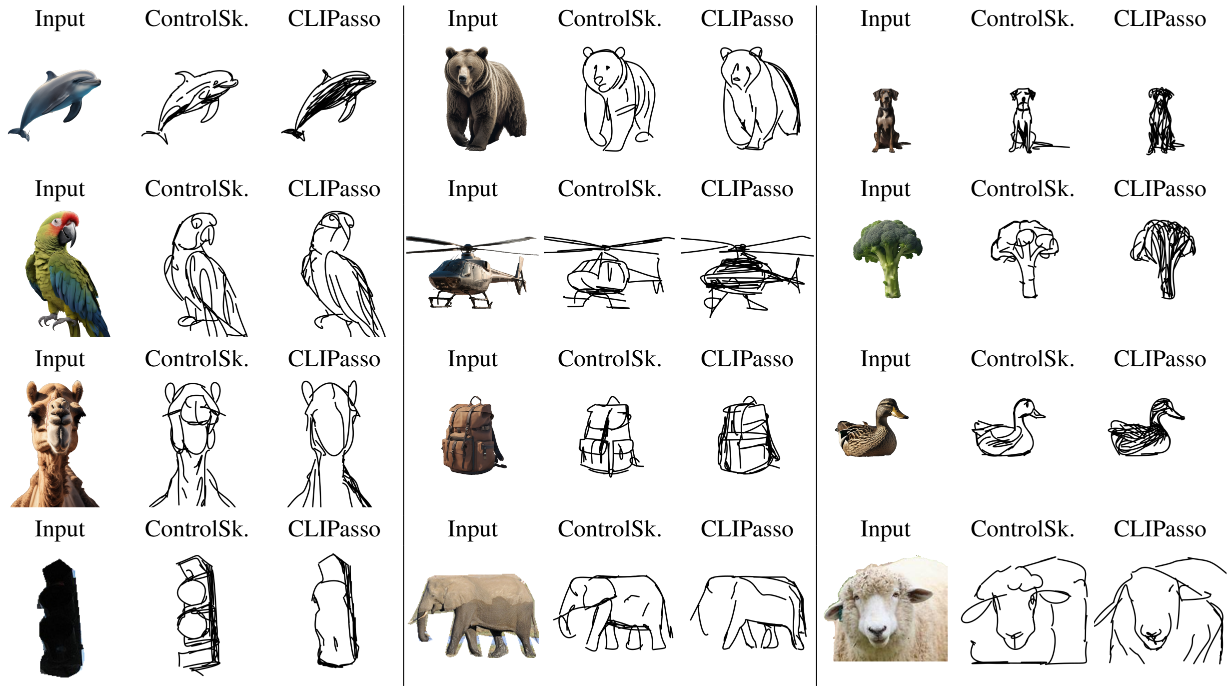


Figure 13. Comparison of ControlSketch with CLIPasso [47]. ControlSketch captures fine details (e.g., camel and bear), avoids artifacts in small objects (e.g., dog and duck), and handles challenging inputs effectively (last row).

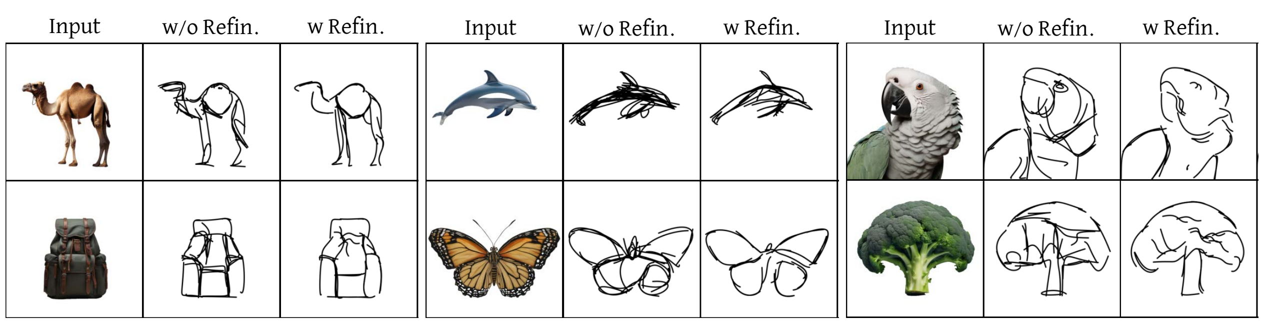


Figure 14. Effect of the refinement network. The output sketches from the diffusion model may contain slight noise, which the refinement network addresses by performing an additional cleaning step. However, this process can sometimes reduce details in the sketch (see Limitations).


*tems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. 3

[21] Zixuan Huang, Mark Boss, Aaryaman Vasishta, James M. Rehg, and Varun Jampani. Spar3d: Stable point-aware reconstruction of 3d objects from single images. 2025. 3

[22] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. *arXiv*, 2022. 3

[23] Moritz Kampelmühler and Axel Pinz. Synthesizing human-like sketches from natural images using a conditional convolutional decoder. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3192–3200, 2020. 2

[24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. 3, 1

[25] Mengtian Li, Zhe L. Lin, Radomír Měch, Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1403–1412, 2019. 2, 6, 7, 8, 4

[26] Tzu-Mao Li, Michal Lukác, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39:1 – 15, 2020. 2, 3, 4

[27] Hangyu Lin, Yanwei Fu, Yu-Gang Jiang, and X. Xue. Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6757–6766, 2020. 2

[28] Difan Liu, Matthew Fisher, Aaron Hertzmann, and Evangelos Kalogerakis. Neural strokes: Stylized line drawing of 3d shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2

[29] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2836–2844, 2021. 2, 3

[30] Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning deep sketch abstraction. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8014–8023, 2018. 2

[31] Kushin Mukherjee, Holly Huey, Xuanchen Lu, Yael Vinker, Rio Aguina-Kang, Ariel Shamir, and Judith E. Fan. Seva: Leveraging sketches to evaluate alignment between human and machine visual abstraction. *ArXiv*, abs/2312.03035, 2023. 2, 1

[32] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *ArXiv*, abs/2102.09672, 2021. 6, 4

[33] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. 2, 4, 6, 1

[34] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *ArXiv*, abs/2209.14988, 2022. 2, 3

[35] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkang Li, and Yi-Zhe Song. Sketchlattice: Latticed representation for sketch manipulation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 933–941, 2021. 2

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 2, 4, 7, 3

[37] Leo Sampaio Ferraz Ribeiro, Tu Bui, John P. Collomosse, and Moacir Antonelli Ponti. Sketchformer: Transformer-based representation for sketched structure. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14141–14150, 2020. 2

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 3

[39] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 6

[40] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Trans. Graph.*, 35(4), July 2016. 2

[41] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 3

[42] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning to sketch with shortcut cycle consistency. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 801–810, 2018. 2

[43] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3

[44] Vikas Thamizharasan, Difan Liu, Shantanu Agarwal, Matthew Fisher, Michael Gharbi, Oliver Wang, Alec Jacobson, and Evangelos Kalogerakis. Vecfusion: Vector font generation with diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7943–7952, June 2024. 2, 3

[45] Varshaneya V, Balasubramanian S, and Vineeth N. Balasubramanian. Teaching gans to sketch in vector format. *Proceedings of the Twelfth Indian Conference on Computer Vision, Graphics and Image Processing*, 2019. 2

[46] Yael Vinker, Yuval Alaluf, Daniel Cohen-Or, and Ariel Shamir. Clipascene: Scene sketching with different types and levels of abstraction. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4123–4133, 2022. 1, 2, 7

[47] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Trans. Graph.*, 41(4), jul 2022. 1, 2, 3, 4, 6, 7, 8, 11

[48] Haofan Wang, Matteo Spinelli, Qixun Wang, Xu Bai, Zekui Qin, and Anthony Chen. Instantstyle: Free lunch towards style-preserving in text-to-image generation. *ArXiv*, abs/2404.02733, 2024. 2, 6, 7, 8, 4

[49] Qiang Wang, Haoge Deng, Yonggang Qi, Da Li, and Yi-Zhe Song. Sketchknitter: Vectorized sketch generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3

[50] Zeyu Wang, Sherry Qiu, Nicole Feng, Holly Rushmeier, Leonard McMillan, and Julie Dorsey. Tracing versus freehand for evaluating computer-generated drawings. *ACM Trans. Graph.*, 40(4), Aug. 2021. 2

[51] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 7

[52] Holger Winnemöller. Xdog: advanced image stylization with extended difference-of-gaussians. In *International Symposium on Non-Photorealistic Animation and Rendering*, 2011. 7

[53] Chufeng Xiao, Wanchao Su, Jing Liao, Zhouhui Lian, Yi-Zhe Song, and Hongbo Fu. Differsketching: How differently do people sketch 3d objects? *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2022)*, 41(4):1–16, 2022. 2

[54] Ximing Xing, Chuan Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *ArXiv*, abs/2306.14685, 2023. 1, 3

[55] Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg gener-

ation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4546–4555, June 2024. 1, 3

[56] Peng Xu, Timothy M. Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey and a toolbox, 2020. 1, 2

[57] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey, 2024. 1

[58] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824, 2023. 2, 4, 1

[59] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[60] Tao Zhou, Chen Fang, Zhaowen Wang, Jimei Yang, Byungmoon Kim, Zhili Chen, Jonathan Brandt, and Demetri Terzopoulos. Learning to doodle with stroke demonstrations and deep q-networks. In *British Machine Vision Conference*, 2018. 2