ControlSVG: Exploring Controllable SVG Generation with Autoregressive Models

Qirui Li[1] **Teng Hu**[1] **Ran Yi**[1*] **Paul L. Rosin**[2] **Yu-Kun Lai**[2]
[1]Shanghai Jiao Tong University
[2]Cardiff University

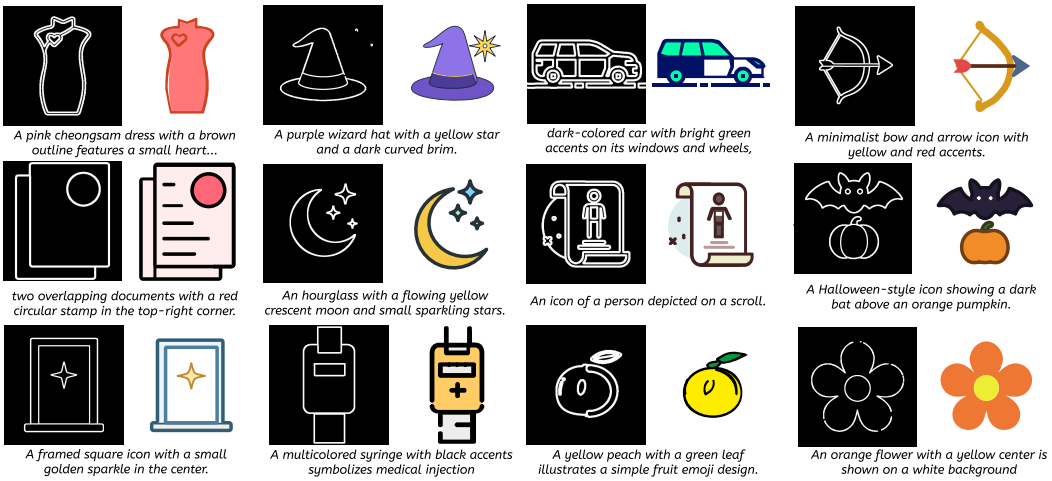

Figure 1: Our **ControlSVG** generates high-quality SVG with complex lines, while also achieving outstanding condition following and text alignment.

Abstract

Autoregressive (AR) models have reformulated Scalable Vector Graphics (SVG) generation as next-token prediction, demonstrating remarkable potential in text-to-SVG tasks. However, controllable SVG generation guided by spatial signals (e.g., sketches or scribbles) remains largely unexplored within AR models. While a natural approach, inspired by controllable image generation, is to adapt methods like Condition Prefilling or Conditional Decoding, they fall short in SVG generation. To address these challenges, we introduce **ControlSVG**, a versatile and effective framework tailored for integrating precise spatial controls into autoregressive SVG generation. Firstly, we propose a Visual-Feedback Conditioning mechanism that constructs a "See-and-Draw" loop. This design mitigates cross-modal drift and ensures long-range consistency. Secondly, we explore spatially aware learning objectives that explicitly bridge discretized SVG tokens with their absolute 2D spatial coordinates, enhancing robustness against coordinate deviations during sequential generation. Extensive experiments on Canny-Edge-to-SVG generation demonstrate that ControlSVG outperforms existing baselines, achieving superior spatial alignment and high-quality SVG. We further provide preliminary validations on real human-drawn scribbles and extend the framework to scribble-to-SVG and image-to-SVG settings, suggesting its potential applicability beyond Canny-based control. code is avaliable at `https://github.com/IApple233/Controlsvg`

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

The emergence of Large Language Models has revolutionized Scalable Vector Graphics (SVG) generation. By treating SVG code as a specialized language, recent works [1–4] have successfully extended the text-to-code paradigm to vector graphics. By discretizing commands, coordinates, and attributes into special SVG tokens, these models can generate complex SVG structures from text prompts.

However, text-driven SVG generation still struggles with fine-grained spatial control. Relying solely on textual prompts inherently limits the ability to specify precise layouts. In practical design scenarios, users often start from much sparser structural guidance, including Canny edges, sketches, or scribbles. A straightforward solution is to adopt a composite Condition-to-Image-to-SVG pipeline, but this approach often introduces vectorization artifacts and struggles to produce high-quality SVGs. These limitations motivate us to develop a native controllable SVG generation framework that directly maps structural conditions to high-quality vector graphics.

To achieve such control, a natural solution is to adapt established paradigms from autoregressive controllable image generation, specifically **Condition Prefilling** [5, 6] and **Conditional Decoding** [7]; however, directly adapting these encounters fundamental obstacles. For example: **Condition Prefilling**, which prefills visual condition image tokens to the sequence, suffers from a severe modality gap: to align abstract vector commands with pixel-based conditions, the model must implicitly simulate the rendering process. This heavy cross-modal burden frequently leads to "semantic drift", where the generated tokens are progressively detached from the visual guidance in long sequences. In parallel, **Conditional Decoding** relies on a structured 2D spatial alignment between the condition and the output. This assumption holds in the pixel-level image domain where spatial correspondence is intrinsic, but collapses for parameterized SVGs, as the output is a serialized 1D sequence of variable-length tokens, lacking a fixed one-to-one correspondence with the condition image, thereby invalidating the efficacy of conditional decoding.

To address these challenges, we propose **ControlSVG**, a controllable autoregressive framework for conditional SVG generation. **First, to resolve the modality misalignment** between serialized outputs and 2D conditions, we propose a **Dynamic Visual Feedback** mechanism. Instead of relying on static prefixes, we establish a "See-and-Draw" loop: intermediate SVG sequences are rasterized on-the-fly and re-injected as visual prompts. By unifying the partial output and the target condition within the same image domain, we enable the model to perform direct spatial comparisons, thereby mitigating cross-modal drift and ensuring long-range consistency. **Second, we identify a conflict in the learning objective.** Standard Cross-Entropy loss treats coordinate tokens as strictly discrete classes, where minor numerical deviations incur disproportionately high penalties. To mitigate this, we introduce a **Spatially Aware Learning Objective** that accounts for the geometric proximity of tokens. This softens the training target based on spatial distances, prioritizing geometric validity over strict token exactness and improving robustness to quantization errors.

Extensive experiments substantiate the efficacy of our framework. We benchmark ControlSVG against a comprehensive spectrum of baselines, including composite raster-to-vector pipelines, representative open-source text-to-SVG models, and large-scale proprietary foundation models such as GPT-5 and Gemini3 Pro.

Our main contributions are summarized as follows:

1. We propose **ControlSVG**, the first autoregressive framework that enables precise spatial control over SVG generation using sketches, scribbles, and images.
2. We introduce a **Dynamic Visual Feedback** mechanism based on intermediate rendering, which bridges the modality gap between 1D vector commands and 2D visual conditions.
3. We design a **Spatially Aware Learning Objective** tailored for discretized coordinates, which resolves the conflict between discrete token supervision and continuous geometric fidelity, significantly enhancing training stability.
4. Extensive experiments show that ControlSVG surpasses state-of-the-art baselines in generation quality and spatial alignment. Furthermore, preliminary experiments on real human-drawn scribbles, together with exploratory extensions to *scribble-to-SVG* and *image-to-SVG* tasks, suggest the potential versatility of ControlSVG beyond Canny-based control.

2 Related Work

2.1 Autoregressive Models for SVG Generation

Early attempts to generate SVGs directly utilized architectures such as RNNs, VAEs, and Transformers to compress SVG commands into latent representations. For instance, DeepSVG [8] parameterizes SVGs using a hierarchical dual-transformer architecture but often struggles with geometric consistency. Recently, the advent of Large Language Models (LLMs) has unleashed the potential of generating SVGs via XML code synthesis. However, the constrained context length of standard LLM-based methods poses significant challenges when handling complex SVGs.

To bridge this gap, subsequent works have integrated LLMs with diverse structural priors and visual feedback mechanisms. Chat2SVG [9] introduced a hybrid framework that combines the semantic reasoning of LLMs with the refinement capabilities of diffusion models, utilizing dual-stage optimization to enhance geometric complexity. LLM4SVG [2] further empowered LLMs by introducing learnable semantic tokens and a modular architecture to encode SVG components, effectively reducing semantic ambiguities and predicting high-precision coordinates. In the realm of multimodal integration, StarVector [3] incorporates a vision encoder to achieve end-to-end Image-to-SVG generation, significantly improving structural complexity. Similarly, OmniSVG [1] advances this by employing a unified Vision-Language Model to process interleaved multi-modal instructions. However, these methods primarily focus on text-driven synthesis or full-image reconstruction, lacking the capability to support fine-grained spatial conditions such as sketches and scribbles.

In this paper, we address this limitation by proposing **ControlSVG**. Our framework effectively bridges the modality gap between abstract vector commands and 2D spatial conditions, enabling precise generation strictly guided by diverse fine-grained control signals.

2.2 Controllable Image Generation

In the field of text-to-image (T2I) generation, relying solely on textual prompts is often insufficient for conveying specific artistic styles or precise geometric constraints. To address this, researchers have proposed various methods to capture visual concepts from reference images that are difficult to describe via text, a task commonly referred to as controllable generation. Representative works such as ControlNet [10] and T2I-Adapter [11] introduce spatial conditioning, such as edge maps, segmentation masks, and depth maps, to provide fine-grained control over the diffusion-based generation process. Subsequently, frameworks like UniControl [12] and Uni-ControlNet [13] have further expanded this domain by optimizing condition encoder architectures and training strategies.

Despite the remarkable success of diffusion models, controllable generation within autoregressive (AR) frameworks remains relatively under-explored. ControlVAR and CAR [5, 14] represent early attempts that utilize a next-scale prediction paradigm to jointly model control signals and images, successfully extending the visual AR framework to spatial guidance tasks. Recently, research has pivoted toward token-based AR control. For instance, ControlAR [7] introduces a versatile and efficient paradigm by injecting spatial conditions directly into the autoregressive decoding process. To further enhance structural consistency and fine-grained editing, EditAR [6] leverages a unified AR framework for diverse conditional generation tasks, including image manipulation and multi-task synthesis. Unlike the iterative denoising process inherent in diffusion models, these AR-based methods treat controllable generation as a conditional sequence modeling task. Our objective is to fully harness the sequential modeling capabilities of AR models to establish a robust paradigm for controllable SVG generation, ensuring long-range structural coherence.

3 ControlSVG

3.1 Problem Formulation

We propose **ControlSVG**, an autoregressive (AR) framework designed to generate Scalable Vector Graphics (SVG) that strictly adhere to multi-modal conditional guidance, including both textual and spatial conditions such as sketches and scribbles. Unlike existing pixel-based controllable models, ControlSVG reformulates SVG generation as a sequence modeling task over a discrete token space.

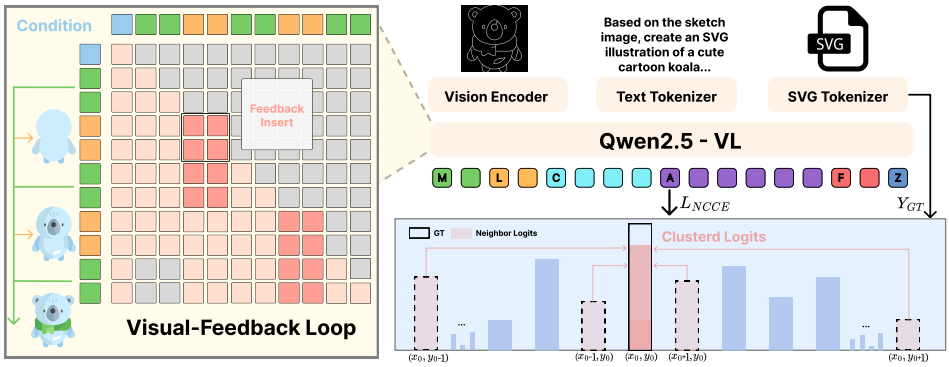

Figure 2: Overview of the ControlSVG framework. Our method introduces Dynamic Visual Feedback (DVF) to establish a "See-and-Draw" loop where intermediate sequences are rasterized and injected as an additional condition. Additionally, a Spatially Aware Learning Objective is proposed to optimize coordinate tokens over a spatial manifold.

Formally, let $S = \{s_1, s_2, \ldots, s_T\}$ denote a sequence of discrete SVG tokens, encompassing command types and quantized coordinates. Given a visual condition I_c and a text prompt T as inputs, our objective is to learn the conditional distribution:

$$P(S|I_c, T) = \prod_{t=1}^{T} P(s_t|s_{<t}, I_c, T). \tag{1}$$

By integrating structural priors from I_c with semantic instructions from T, the model autoregressively predicts SVG elements while maintaining high fidelity to the input constraints.

3.2 Dynamic Visual Feedback

Standard AR models for SVG generation typically rely on a static condition prefix. However, as the sequence length increases, the correlation between abstract coordinate tokens $s_{<t}$ and the initial visual condition I_c tends to attenuate. This often leads to "*blind generation*", where the model loses track of the cumulative canvas state and accumulates spatial errors over long-range sequences. To overcome this, we introduce **Dynamic Visual Feedback (DVF)**, an automatic mechanism that transforms the generation process into a "See-and-Draw" loop – intermediate SVG sequences are rasterized and re-injected as visual tokens. Instead of treating SVG generation as a purely text-like sequence task, DVF interleaves the sequence with updated visual snapshots of the canvas.

The Visual-Feedback Loop. Suppose an SVG is composed of N paths, where P_i denotes the token sequence of the i-th path. After predicting each path (or a predefined group of paths), we employ a rasterizer $\mathcal{R}$ to render the currently generated SVG sequence into a raster image $I_{pred}^{(i)}$:

$$I_{pred}^{(i)} = \mathcal{R}(P_1, P_2, \ldots, P_i). \tag{2}$$

This rendered image $I_{pred}^{(i)}$ captures the explicit geometric state of the canvas. We then utilize a frozen vision encoder from the backbone to project this image $I_{pred}^{(i)}$ into a set of visual feedback tokens v_i.

Interleaved Autoregressive Modeling. Unlike traditional methods that treat visual features as external latent variables, DVF inserts these visual feedback tokens v_i directly back into the generative sequence. This extends the input into an interleaved format: $[I_c, T, P_1, P_2, \ldots, P_i, v_i]$ when predicting $(i+1)$-th path. Consequently, the conditional probability for the $(i+1)$-th path is reformulated as:

$$P(P_{i+1} \mid P_{\leq i}, v_i, I_c, T) = \prod_{j \in P_{i+1}} P(s_j \mid s_{<j}, v_i, I_c, T), \tag{3}$$

where v_i denotes the most recent visual feedback token. Crucially, when predicting path P_{i+1}, only the latest feedback token v_i is utilized as the supplementary condition, while previous visual feedback tokens $v_{<i}$ are excluded from the active context.

This design provides two distinct advantages: (1) Computational Efficiency: By introducing a visual token of one single image (v_i) per prediction step, we maintain a constant marginal overhead.

This prevents the computational explosion and memory strain that would occur if multiple high-dimensional visual tokens were accumulated throughout the long-range generation process. (2) RoPE (Rotary Positional Embedding) Continuity and Information Fidelity: Since v_i is rendered from the current state of all preceding paths $\{P_1, \ldots, P_i\}$, it provides an accurate representation of the canvas, which already contains the information in historical feedback tokens $v_{<i}$. More importantly, the structural coherence of SVG in Transformer relies on the Rotary Positional Embeddings ($RoPE$) [15]. The relative positions of $[P_k, P_{k+1}]$ are continuous, while using $v_{<i}$ as an additional condition like $[P_k, v_k, P_{k+1}]$ will create gaps, disrupting the model's perception of relative positions. By keeping only the most proximal feedback token v_i, we minimize positional displacement and preserve the logical continuity between consecutive paths.

By explicitly feeding the embedding of rendered result v_i back into the model, this strategy ensures that the sequence remains structurally coherent while providing precise visual guidance. This allows the model to perceive the existing SVG paths on the canvas and dynamically adjust the coordinates of the subsequent path, ensuring strict alignment with the global condition I_c and maintaining long-range geometric consistency. While injecting visual feedback tokens increases the context length, the computational overhead is well-constrained. A detailed analysis of the inference time cost is provided in the supplementary material.

3.3 Spatially Aware Learning Objective

SVG Tokenization. ControlSVG leverages the tokenizer architecture of OmniSVG as its backbone. To facilitate autoregressive modeling, each SVG path is decomposed into a sequence of discrete tokens. Specifically, an SVG script is represented as a combination of M paths, where each path consists of: (1) **Path Coordinates**: Each point is defined by a command identifier $U \in \{\text{M, L, C, A, Z}\}$ and a coordinate pair (x, y). To fully leverage the pretrained weights of OmniSVG, we strictly follow its tokenization design and fix the canvas size to 200×200. We linearize the 2D coordinates into a 1D vocabulary using a mapping function $\langle x, y \rangle \mapsto x \times w + y$, where w denotes the canvas width. (2) **Fill Attributes**: These are represented as quantized hex color values. Through this discretization, SVG generation is reformulated as a next-token prediction task.

Limitation of Standard Cross-Entropy Loss. Standard Large Language Models (LLMs) typically optimize discrete sequences using the Cross-Entropy (CE) loss:

$$\mathcal{L}_{CE} = -\log P(s_t = y | s_{<t}), \tag{4}$$

where y is the ground-truth token. While CE loss is effective for categorical semantic tokens (e.g., words), it is suboptimal for SVG coordinates that reside in a continuous spatial metric space. In a discretized canvas, coordinate tokens $c_1 = \langle x, y \rangle$ and $c_2 = \langle x, y + 1 \rangle$ are visually near-identical. However, standard CE treats them as mutually exclusive classes, penalizing a near-miss prediction as severely as a distant coordinate. This lack of spatial awareness results in a non-smooth optimization target, where the model struggles to generalize across spatially related coordinates. To address this, we propose two strategies to inject spatial priors into the learning objective and to tolerate a close prediction:

Gaussian Soft-Targets (GST). To model spatial uncertainty, we replace the one-hot ground truth with a smoothed distribution. We define a target distribution $Q(k)$ over the coordinate vocabulary. Let $c_k, c_y \in \mathbb{R}^2$ denote the 2D coordinates represented by token k and the ground-truth token y, respectively. The probability mass decays with their Euclidean distance:

$$Q(k) \propto \exp\left(-\frac{\|c_k - c_y\|_2^2}{2\sigma^2}\right). \tag{5}$$

The model is then optimized by minimizing the Kullback-Leibler (KL) divergence between the predicted distribution P and the target spatial distribution Q:

$$\mathcal{L}_{KL} = D_{KL}(Q \parallel P) = \sum_k Q(k) \log \frac{Q(k)}{P(k)}. \tag{6}$$

While this approach encourages the model to assign higher probabilities to neighboring regions than distant regions, we discuss its specific empirical trade-offs in the experiments section.

Neighborhood Cluster Cross-Entropy (NCCE). To maintain the sharpness of the prediction while introducing spatial tolerance, we propose NCCE. Instead of forcing the model to pick a single

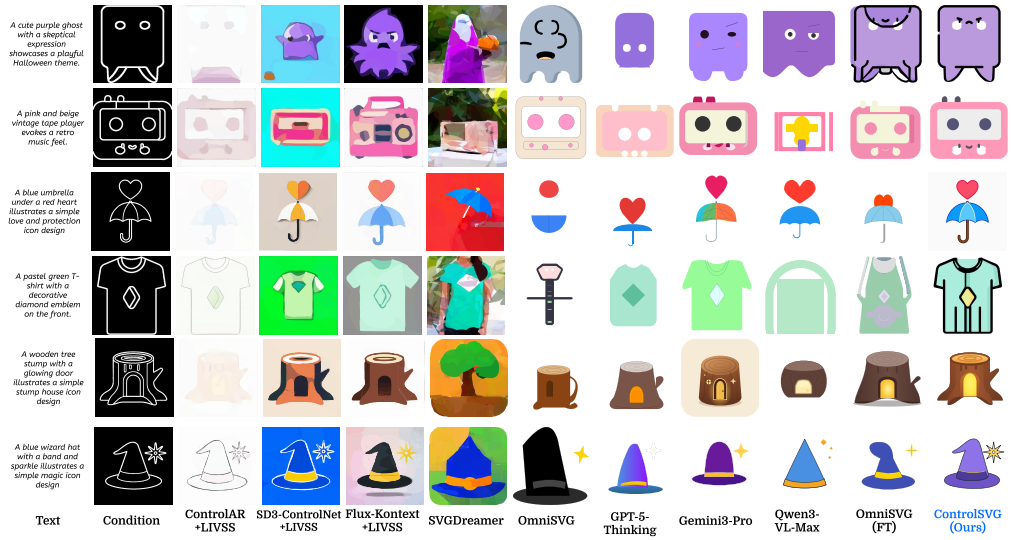

Figure 3: Qualitative comparison of existing controllable image generation, Text-to-SVG methods, VLLMs methods and our *ControlSVG*. Our method generates SVG with high quality and good alignment with the visual condition.

correct token, we redefine the optimization target from a singular token to a spatial manifold. By introducing a spatial window $\mathcal{N}(y)$ centered at the ground-truth coordinate y, we encourage the model to concentrate its probability mass within a geometrically meaningful neighborhood. Any prediction falling within this neighborhood window is considered geographically valid. Specifically, we aggregate the logits of all tokens within the neighborhood $\mathcal{N}(y)$ to form a cluster-level probability. The probability that the model predicts a point within the valid neighborhood is formulated as:

$$P(\mathcal{N}(y)) = \frac{\sum_{k \in \mathcal{N}(y)} \exp(\hat{Y}_k)}{\sum_{i=1}^{V} \exp(\hat{Y}_i)}, \tag{7}$$

where $\hat{Y}_k$ is the logit for token k, and V is the vocabulary size. The NCCE loss is then defined as:

$$\mathcal{L}_{\text{NCCE}} = -\log P(\mathcal{N}(y)) + L_{CE}. \tag{8}$$

To preserve exact-token discrimination while introducing spatial tolerance, we combine the standard coordinate CE loss with NCCE. This hybrid objective encourages the model to place high probability mass near the ground-truth coordinate while still maintaining a peak at the exact target token.

Analysis and Final Objective. While both strategies aim to incorporate spatial priors, they exhibit distinct optimization characteristics. Gaussian Soft-Targets (GST) explicitly models distance-based uncertainty; however, minimizing the KL divergence against a dense, smoothed distribution can inadvertently dilute the supervisory signal. This often leads to "optimization softening," where the model struggles to converge to high-confidence predictions. In contrast, Neighborhood Cluster Cross-Entropy (NCCE) retains the gradient sharpness of standard CE while effectively creating a "tolerance zone." It allows model to learn distinct decision boundaries without being penalized for negligible spatial shifts. **Consequently, we adopt NCCE as the primary learning objective for ControlSVG.** We provide a detailed ablation study in Section 4.3, which confirms that NCCE yields superior convergence rates and higher geometric fidelity compared to GST.

4 Experiments

4.1 Experiment Setup

Datasets We evaluate ControlSVG on MMSVGBench, a comprehensive dataset derived from the Image-to-SVG tasks in OmniSVG [1]. We utilize BLIP-2 [16] to generate descriptive captions for each image, which serve as textual guidance. Regarding spatial constraints, we follow the methodology of ControlNet [10] to extract Canny edges as our primary conditioning signal. Our evaluation focuses on two distinct categories: *Icon* and *Illustration*.

Dataset	Type	Methods	Image Quality			Condition Following		
			FID↓	Aesth.↑	CLIP↑	SSIM↑	Dino Sim.↑	Dist. Trans.↓
MMSVG-Icon	Cond-to-Img+Vec	SD3-ControlNet+LIVSS	190.1	4.616	0.301	0.709	0.754	31.984
		ControlAR+LIVSS	175.5	4.401	0.276	0.566	0.783	39.507
		Flux-Kontext+LIVSS	183.4	4.796	**0.303**	0.567	0.829	12.502
	Text-to-SVG	SVGDreamer	179.2	4.616	0.279	–	–	–
		OmniSVG (text2svg)	130.2	4.606	0.229	–	–	–
	MLLM	GPT-5-Thinking	178.2	4.640	0.279	0.729	0.822	19.465
		Gemini3-Pro	123.4	4.809	0.300	0.708	0.843	12.946
		Qwen-VL-Max	150.1	4.529	0.279	0.754	0.751	24.275
	Cond-to-SVG	OmniSVG (FT)	126.7	4.771	0.281	0.790	0.849	14.300
		ControlSVG (Ours)	**123.1**	**4.843**	0.284	**0.859**	**0.870**	**9.445**
Illustration	Cond-to-Img+Vec	SD3-ControlNet+LIVSS	186.4	4.622	0.311	0.667	0.738	26.772
		ControlAR+LIVSS	174.6	4.058	0.247	0.555	0.739	50.549
		Flux-Kontext+LIVSS	186.1	4.679	0.311	0.556	0.790	16.857
	Text-to-SVG	SVGDreamer	168.4	4.734	0.283	–	–	–
		OmniSVG (text2svg)	117.7	4.644	0.214	–	–	–
	MLLM	GPT-5-Thinking	109.8	4.575	0.299	0.726	0.774	28.669
		Gemini3-Pro	105.9	**4.807**	**0.322**	0.718	**0.845**	17.258
		Qwen-VL-Max	129.3	4.557	0.294	0.692	0.758	29.118
	Cond-to-SVG	OmniSVG (FT)	109.7	4.673	0.264	0.755	0.764	16.470
		ControlSVG (Ours)	**93.7**	4.744	0.292	**0.792**	0.838	**15.458**

Table 1: Quantitative comparison results across image quality and condition following metrics on MMSVGBench. **Bolding** indicates the best performance among SVG-based generation methods. Underlining indicates the second-best performance in each column.

Baselines As there is currently no prior work dedicated specifically to condion-to-SVG generation, we establish a competitive baseline by selecting leading methods from three domains:

(1) Condition-to-Image + Vectorization: We evaluate controllable diffusion-based and autoregressive models, including SD3-ControlNet [10], ControlAR [7], and Flux-Kontext [17]. Since our task is SVG generation while these models output raster images, instead of utilizing AR-based image-to-SVG models, which often suffer from instability and structural distortion, we employ LIVSS [18], an optimization-based method, to vectorize their outputs into SVGs. This design provides a competitive raster-to-vector baseline: compared with generative image-to-SVG models such as OmniSVG, LIVSS showed better reconstruction fidelity and fewer invalid or distorted paths.

(2) Text-to-SVG Generation: We include specialized vector generation models such as SVGDreamer [19], and OmniSVG [1]. Crucially, as these methods lack an interface for spatial conditioning, they are provided only with text prompts. Consequently, their performance is assessed solely on aesthetic and generative quality metrics.

(3) Multi-modal Large Language Models (MLLMs): We compare our framework with leading SOTA MLLMs, including GPT-5-thinking [20], Gemini-3-Pro [21], and Qwen3-VL-Max [22]. These models are prompted to directly synthesize SVG code based on the provided visual conditions and textual instructions.

Evaluation Metrics We utilize two sets of metrics targeting visual quality and spatial fidelity: 1) For visual quality, we employ the Fréchet Inception Distance (FID) [23], alongside the Aesthetic Score [24] and CLIP Score [25] for assessing artistic appeal and semantic prompt alignment, respectively; 2) Meanwhile, to quantify spatial condition following, we calculate SSIM [26] for structural similarity and the cosine similarity of features extracted via DINOv2 [27], which captures high-level perceptual correspondences that are robust to stylistic variations while remaining sensitive to geometric alignment. Finally, we incorporate Distance Transform (DT) [28], a metric that evaluates geometric fidelity by calculating the spatial displacement between sampled points on the generated vector paths and the corresponding structural elements in the condition.

4.2 Comparison with Existing Methods

Qualitative Results. As illustrated in Figure 3, qualitative results highlight the limitations of existing paradigms. Specifically, composite pipelines of controllable image generation and vectorization methods suffer from vectorization artifacts and a lack of stylized simplicity of professional SVG icons.

Text-to-SVG methods, lacking condition guidance, often struggle to adhere to spatial constraints or maintain graphical integrity. While MLLMs (e.g., GPT-5, Gemini-3-Pro) can generate relatively complete and aesthetic SVG code, their capacity for strict condition-following remains insufficient for precise coordinate alignment. In contrast, ControlSVG eliminates vectorization-induced fidelity loss by precisely restoring complex topologies. By synthesizing clean, semantically aligned icons with harmonious aesthetics, our method demonstrates superior performance in high-quality controllable SVG synthesis.

Quantitative Results. As demonstrated in Table 1, our proposed ControlSVG achieves leading performance for controllable SVG generation. On the MMSVG-Icon dataset, it leads in overall performance. For the more complex MMSVG-Illustration task, our method attains the best FID and SSIM, trailing only Gemini3-Pro in aesthetic and CLIP scores while maintaining significantly higher structural precision. Compared to vectorized Cond-to-Img baselines, our native approach avoids the inevitable distortion and loss of condition-following capability caused by path constraints during pixel-to-vector conversion. Furthermore, ControlSVG outperforms its Text-to-SVG baseline in aesthetics, proving that structural guidance enhances both rendering quality and semantic understanding. Notably, our method exhibits a decisive advantage over MLLMs in structural fidelity; while MLLMs struggle with discrete coordinate precision and global spatial reasoning in code generation, our specialized control architecture ensures robust geometric correspondence and precise spatial anchoring.

User Study. We compared ControlSVG against three representative competitors in user study, i.e., Flux-Kontext, GPT-5-Thinking, and Gemini3-Pro. 13 CV/CG professionals were recruited, with each participant evaluating 30 sets of comparisons, consisting of 15 icons and 15 illustrations. Each set featured two randomly arranged side-by-side SVGs, one generated by ControlSVG and the other by a baseline. Participants were asked to perform a pairwise comparison (A, B, or Tie), prioritizing "condition-following" as the primary metric, with "overall visual aesthetics" as the secondary factor. The user voting results are shown in Fig. 4, demonstrating the superior quality of ControlSVG results compared with baselines.

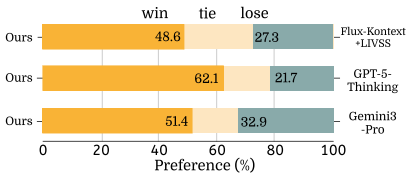

Figure 4: User Study. The results generated by ControlSVG received more preference from the participants compared to other methods.

4.3 Ablation Study

Effectiveness of Core Components. We conduct a comprehensive ablation study to investigate the individual contributions of Dynamic Visual Feedback (DVF), Gaussian Soft-Targets (GST), and the Neighborhood Cluster Cross-Entropy (NCCE), with results detailed in Table 2 and Fig. 5. Removing DVF (*w/o DVF*) leads to a degradation in overall image quality and structural fidelity, resulting in misaligned paths. While the introduction of GST enhances global layout coherence, its performance in condition following remains inferior to NCCE, and certain structural defects in image quality still persist. We attribute this to the fact that the soft targets employed in GST expand the model's exploration range, which, while beneficial for diversity, make the model struggle to converge to high-confidence predictions. NCCE stands out as the most critical component; it introduces spatial awareness at the token level by modeling the continuous metric nature of the geometric space.

Analysis of Spatially Aware Learning Objective. Beyond coordinates, we further explore the impact of spatially-aware supervision on color tokens. Although color values are represented as discrete tokens, they naturally correspond to RGB triplets, where nearby colors should be treated as related. In our implementation, two color tokens are considered neighbors only when the absolute difference of each RGB channel is no larger than 32, e.g. #66ddff and #77eeff. In our ablation, removing spatially-aware color supervision (*w/o Color Neighbor Loss*) causes "color collapse," where the model produces mismatched hues or reverts to extreme colors such as pure black or white. Similarly, removing the geometric prior for coordinate tokens (*w/o Coord Neighbor Loss*) often results in meaningless fragments and disconnected paths, since minor spatial deviations are treated as completely incorrect predictions. By applying NCCE to both coordinates and colors, ControlSVG mitigates fragmentation and color mismatch issues, producing SVGs that better reflect conditions.

4.4 Other Conditional Guidance and Robustness

Component			Image Quality			Condition Following		
DVF	GST	NCCE	FID↓	Aesth.↑	CLIP↑	SSIM↑	Dino Sim.↑	Dist. Trans.↓
			109.7	4.673	0.264	0.755	0.764	16.470
✓			103.5	4.688	0.274	0.774	0.804	15.779
	✓		99.8	4.721	0.286	0.783	0.824	15.612
✓	✓		96.3	4.697	0.281	0.781	0.816	15.672
✓		✓	**93.7**	**4.744**	**0.292**	**0.792**	**0.838**	**15.458**

Table 2: Ablation study on core components. The top row is OmniSVG (FT).

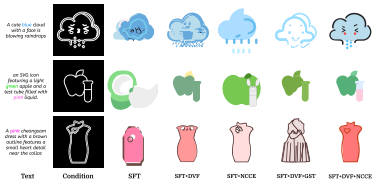

Figure 5: Qualitative results of ablation for core components.

Spatial Target		Image Quality			Condition Following		
Coord.	Color	FID↓	Aesth.↑	CLIP↑	SSIM↑	Dino Sim.↑	Dist. Trans.↓
✓		96.2	4.736	0.284	0.787	0.829	15.576
	✓	98.3	4.701	0.278	0.781	0.811	15.760
✓	✓	**93.7**	**4.744**	**0.292**	**0.792**	**0.838**	**15.458**

Table 3: Impact of applying Spatial Loss to different token sequences.

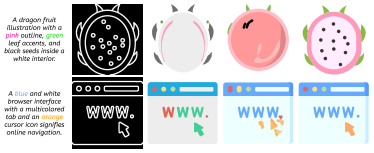

Figure 6: Qualitative results of ablation for NCCE Loss.

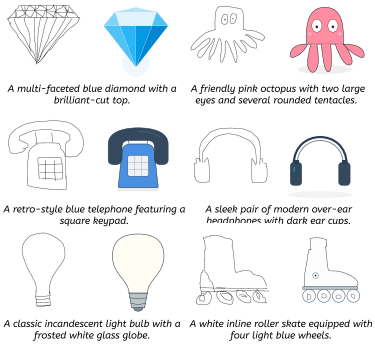

Figure 7: Qualitative results of TU-Berlin dataset.

To evaluate the robustness of ControlSVG, we validate the generalizability against out-of-domain human inputs by testing our model on real human-drawn sketches from the TU-Berlin dataset [29], as illustrated in Fig. 7.

Moreover, we extend our method to other conditioning signals, focusing on scribble and image-to-SVG tasks. These experiments serve as supplementary validations of our framework's versatility. As detailed in the supplementary material, these modalities provide different amounts of visual information: scribbles are the sparsest and mainly specify coarse structure, Canny edges provide denser local boundaries, and images further contain appearance, layout, and color cues. Under sparse scribble conditions, ControlSVG can still recover missing details by leveraging the text prompt and learned visual priors, yielding strong structural alignment. Under image-to-SVG conditions, the richer condition naturally provides more complete guidance, leading to improved condition following and aesthetic quality while preserving native autoregressive SVG generation.

5 Conclusion

In this paper, we study fine-grained spatial control for autoregressive SVG generation and propose ControlSVG, a controllable framework that primarily targets structural-condition-to-SVG generation. ControlSVG introduces Dynamic Visual Feedback (DVF), which constructs a "See-and-Draw" loop by rasterizing intermediate SVG sequences and reinjecting the rendered canvas as visual feedback. This mechanism helps the model perceive the evolving canvas state and improves long-range structural consistency during sequential generation. We further introduce Neighborhood Cluster Cross-Entropy (NCCE), a spatially aware objective that accounts for the geometric neighborhood of discretized coordinate and color tokens, improving robustness against small quantization deviations and reducing fragmented or misaligned paths. Extensive experiments on Canny-Edge-to-SVG generation demonstrate that ControlSVG achieves strong spatial alignment and competitive visual quality compared with existing raster-to-vector, text-to-SVG, and multimodal baselines. Additional exploratory results on human-drawn sketches, scribble-to-SVG, and image-to-SVG settings suggest that the proposed framework has potential to generalize beyond Canny-based control, although these broader settings remain preliminary and warrant further investigation.

6 Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Nos. 62302297, 62272447, and 62472285), the Fundamental Research Funds for the Central Universities (YG2024QNA44), and the Engineering and Physical Sciences Research Council [No. EP/Y028805/1].

References

[1] Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Fukun Yin, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. OmniSVG: A unified scalable vector graphics generation model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[2] Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering LLMs to understand and generate complex vector graphics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19487–19497, 2025.

[3] Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. StarVector: Generating scalable vector graphics code from images. *arXiv preprint arXiv:2312.11556*, 2023.

[4] Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. IconShop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023.

[5] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Zhe Lin, Rita Singh, and Bhiksha Raj. ControlVAR: Exploring controllable visual autoregressive modeling. *arXiv preprint arXiv:2406.09750*, 2024.

[6] Jiteng Mu, Nuno Vasconcelos, and Xiaolong Wang. EditAR: Unified conditional generation with autoregressive models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7899–7909, 2025.

[7] Zongming Li, Tianheng Cheng, Shoufa Chen, Peize Sun, Haocheng Shen, Longjin Ran, Xiaoxin Chen, Wenyu Liu, and Xinggang Wang. ControlAR: Controllable image generation with autoregressive models. *arXiv preprint arXiv:2410.02705*, 2024.

[8] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. DeepSVG: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361, 2020.

[9] Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2SVG: Vector graphics generation with large language models and image diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23690–23700, 2025.

[10] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[11] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-Adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4296–4304, 2024.

[12] Can Qin, Shu Zhang, Ning Yu, Yihao Feng, Xinyi Yang, Yingbo Zhou, Huan Wang, Juan Carlos Niebles, Caiming Xiong, Silvio Savarese, et al. UniControl: A unified diffusion model for controllable visual generation in the wild. *arXiv preprint arXiv:2305.11147*, 2023.

[13] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K Wong. Uni-ControlNet: All-in-one control to text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:11127–11150, 2023.

[14] Ziyu Yao, Jialin Li, Yifeng Zhou, Yong Liu, Xi Jiang, Chengjie Wang, Feng Zheng, Yuexian Zou, and Lei Li. CAR: Controllable autoregressive modeling for visual generation. *arXiv preprint arXiv:2410.04671*, 2024.

[15] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[16] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, pages 19730–19742. PMLR, 2023.

[17] Black Forest Labs. FLUX.1: text-to-image generation of the next generation. `https://blackforestlabs.ai/`, 2024. Accessed: 2026-01-18.

[18] Zhenyu Wang, Jianxi Huang, Zhida Sun, Yuanhao Gong, Daniel Cohen-Or, and Min Lu. Layered image vectorization via semantic simplification. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7728–7738, 2025.

[19] Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. SVGDreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4546–4555, 2024.

[20] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[21] Google DeepMind. Gemini 3 pro, 2025. URL `https://deepmind.google/models/gemini/pro`. Accessed: 2026-01-18.

[22] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017.

[24] Christoph Schuhmann. Improved aesthetic predictor. `https://github.com/christophschuhmann/improved-aesthetic-predictor`, 2022. Accessed: 2026-01-18.

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[26] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

[27] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[28] Ricardo Fabbri, Luciano Da F Costa, Julio C Torelli, and Odemir M Bruno. 2D Euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys (CSUR)*, 40(1): 1–44, 2008.

[29] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4):44:1–44:10, 2012.

[30] Qwen Team. Qwen2.5-vl, January 2025. URL `https://qwenlm.github.io/blog/qwen2.5-vl/`.

[31] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[33] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL `http://arxiv.org/abs/2403.13372`.

[34] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 2009.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the main contribution, namely controllable SVG generation with visual feedback conditioning and a spatially aware objective. The claims are supported by the experimental comparisons and ablation studies in the main paper, while broader generalization beyond Canny conditions is described as preliminary.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The supplementary material includes a dedicated Limitations section. It discusses the path-only SVG representation, the current single-condition setting, and the dependence on the Qwen2-VL-3B backbone.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper is empirical and does not present formal theoretical results, theorems, or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper describes the model design, training objective, datasets, baselines, metrics, and implementation details needed to reproduce the main experimental setup. The supplementary material further reports the training framework, hyperparameters, visual-feedback strategy, and compute resources.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The code is avaliable at `https://github.com/IApple233/Controlsvg`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper specifies the datasets, evaluation split size, conditioning construction, baselines, metrics, backbone, optimizer, learning rate schedule, LoRA rank, batch size, sequence length, precision, training framework, and compute resources in the main paper and supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

   Justification: The paper reports aggregate quantitative results and a user preference study, but it does not report error bars, confidence intervals, or statistical significance tests. We leave multi-seed or repeated-run uncertainty analysis to future work.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The supplementary material reports that training uses 8 NVIDIA H20 GPUs and that 3 epochs take approximately 4 days. It also reports the model scale, batch setting, precision, sequence length, and training framework.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work uses open-source datasets and frameworks, preserves submission anonymity, and reports a low-risk user preference study with informed participants. We are not aware of any aspect of the research that violates the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The supplementary material includes a broader impact discussion. The main positive impact is assisting design and vector-content creation from lightweight spatial guidance such as hand-drawn sketches; potential concerns are limited but include respecting the licenses and terms of source data, models, and downstream design assets.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The initial submission does not release a new model or dataset, and the work is focused on controllable SVG generation for design assistance rather than high-risk data or model release. Future code release will include appropriate documentation and usage terms.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes].

Justification: Existing assets, including MMSVG, TU-Berlin sketches, Qwen2-VL, LLaMA-Factory, BLIP-2, OmniSVG, and other baseline methods, are credited through citations. These assets are open-source or publicly available, and we follow their intended research use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A].

Justification: The initial submission does not release new datasets, model checkpoints, or code packages. We plan to release the code after it is organized, at which time documentation and license information will be provided.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes].

Justification: The paper reports a user study with 13 CV/CG professionals and describes the pairwise comparison task. The supplementary material provides additional details about the instructions, randomized side-by-side interface, voluntary unpaid participation, and evaluation criteria; screenshots of the interface will be included in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The user study involved informed participants performing low-risk visual preference comparisons. No sensitive personal data is involved. No IRB approval or equivalent institutional review was obtained.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: The method is built on vision-language/autoregressive model components, including Qwen2-VL and LLaMA-Factory, and the paper describes their role in the proposed controllable SVG generation framework. Use of MLLMs as baselines is also described in the experiments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

A Technical appendices and supplementary material

B Overview

In this supplementary material, we mainly present the following contents:

- More implementation details of our model structure and training strategy in Sec. B.1.
- More ablation studies in Sec. B.2.
- More details about the baselines in Sec. B.3.
- More details about the user study in Sec. B.4.
- Extension results on scribble-to-SVG, image-to-SVG, and human-drawn sketches in Sec. C.
- Limitations and future research directions in Sec. C.3.
- More qualitative comparisons in Sec. C.2.

B.1 Implementation Details

Model Configuration. We fine-tune the Qwen2-VL-3B-Instruct [30] model using Low-Rank Adaptation (LoRA) [31] with a rank of $r = 128$ applied to all linear layers. The training is conducted for 3 epochs using the AdamW [32] optimizer with a cosine learning rate scheduler (initial learning rate 5×10^{-4}, warmup ratio 0.05). We employ an effective batch size of 8 (per-device batch size of 1 with 8 gradient accumulation steps) and utilize `bf16` mixed precision. The maximum sequence length is set to $8,192$ tokens. All experiments are implemented within the LLaMA-Factory [33] framework.

Compute Resources. The model is trained on 8 NVIDIA H20 GPUs. Training for 3 epochs takes approximately 4 days under our implementation.

Datasets. We utilize the MMSVG-Icon and MMSVG-Illustration datasets proposed by OmniSVG [1]. To provide spatial guidance, we extract structural conditional guidance using Canny [34] and Scribble [10] operators, with all images resized to a resolution of 448×448 pixels.

Training and Inference Protocol of Dynamic Visual Feedback. As discussed in Sec. 3.2, our architecture mandates that only the most proximate image token serves as an auxiliary condition for subsequent predictions. Due to the continuity constraints of Rotary Positional Embeddings (RoPE), inserting multiple visual feedback tokens into a single long training sequence would introduce additional positional gaps between consecutive SVG paths. We therefore decompose each original data entry into multiple sub-sequences. Specifically, each sub-sequence shares the same initial visual condition and text prompt, while a visual feedback token rendered from the preceding SVG paths is inserted after a specific path-level prefix. This construction allows the model to learn the feedback-conditioned prediction behavior without repeatedly interrupting the original SVG token order. To manage the resulting data expansion while preserving the model's native non-feedback generation capability, we apply this sequence-splitting strategy exclusively to the Illustration and Icon datasets.

The feedback update is synchronized with the path-level structure of the SVG token sequence. Following the OmniSVG tokenizer, each SVG path is terminated by a special end-of-path token. We use the emission of this end-of-path token as the trigger for one rasterization step. That is, once a complete path has been produced, the current SVG prefix is rendered into a raster image and then encoded by the frozen vision encoder to obtain the visual feedback token for subsequent prediction. This path-level trigger avoids rendering incomplete path fragments and ensures that feedback is constructed from semantically meaningful SVG units.

In practice, an intermediate SVG prefix may occasionally contain invalid or unclosed path segments, especially during inference. In such cases, the renderer discards the invalid segments and renders only the valid paths in the current prefix. The original token sequence is not modified, and the generation process continues normally. The rendered image is used solely as the auxiliary visual condition for the following decoding steps.

During training, we adopt teacher forcing and render the ground-truth SVG prefix to obtain the feedback image. This is necessary because the model is supervised by cross-entropy against the ground-truth token sequence; therefore, the feedback image and the target next tokens should

correspond to the same underlying SVG condition. During inference, the ground-truth prefix is unavailable, so we instead render the currently generated SVG prefix. The resulting feedback image reflects the model's current canvas state and is used as the dynamic visual condition for predicting subsequent paths, which is consistent with the intended "see-and-draw" mechanism.

Instructions

In Fig. 8, we show the specific system prompts and instruction templates employed in our framework.

B.2 More Ablation Studies

Window Size of NCCE Loss. We investigate the impact of the neighbor window size in Neighborhood Cluster Cross-Entropy (NCCE) loss, which defines the spatial radius for tokens considered as "neighbors." Specifically, given a target point at (x, y), any token whose coordinates (x', y') satisfy $\max(|x - x'|, |y - y'|) \leq w$ is treated as a neighbor. As shown in Table 4, a window size of $w = 5$ achieves the best performance across. We observe that a window size that is too small ($w = 1, 3$) fails to provide sufficient local geometric context, leading to suboptimal smoothness and structural consistency. Conversely, an excessively large window ($w = 10$) tends to introduce noise from distant, unrelated path segments, slightly degrading the precision of condition following.

Instructions For Different Tasks

- **Employed BLIP2 for SVG Captioning**: You are a helpful assistant. Your task is to describe this image in a single sentence, including the object, its color, and its overall arrangement. For example: "Yellow cheers with glasses of alcohol drinks." / "Heart emojis represent love on Valentine's Day."

- **Canny/Scribble-to-SVG**: You are a helpful SVG Generation assistant, designed to generate SVG. We provide the condition image and it's text description as input, generate SVG based on the input. I will randomly provide some render image of current paths during generation to help you better understand the current result."

- **Image-to-SVG**: You are a helpful SVG Generation assistant, designed to generate SVG. We provide an image and it's text description as input, generate SVG based on the input. I will randomly provide some render image of current paths during inference.

Figure 8: Instructions for Different Tasks. Instructions including captions, condition-to-SVG based on different signals like canny, scribble and image.

Window Size	Image Quality			Condition Following		
	FID↓	Aesth.↑	CLIP↑	SSIM↑	Dino Sim.↑	Dist. Trans..↓
1	101.2	4.693	0.279	0.769	0.814	15.653
3	98.6	4.701	0.283	0.783	0.821	15.601
5 (Ours)	**93.7**	**4.744**	**0.292**	**0.792**	**0.838**	**15.458**
10	95.4	4.702	0.288	0.784	0.829	16.121

Table 4: Ablation study of different neighbor window sizes in NCCE loss.

Analysis of Dynamic Visual Feedback (DVF) Refresh Rate. The Dynamic Visual Feedback (DVF) mechanism enables the model to perceive the rendered state of previously generated paths by injecting image tokens into the sequence context. We define the "Refresh Rate" as the probability of this injection occurring whenever a path-level end-of-sequence (<EOS>) token is reached. While image

tokens do not require additional generation time, they extend the total sequence length, increasing inference latency due to the larger context in the attention mechanism. As shown in Table 5, even the 0% baseline (without visual feedback) outperforms the fine-tuned OmniSVG (FT). Introducing a moderate refresh rate significantly improves structural precision and image quality; however, a 100% refresh rate leads to performance degradation, likely due to the accumulation of intermediate rendering noise and over-guidance. Empirically, a 30% refresh rate achieves the optimal balance, yielding peak performance with a manageable 12% increase in computational overhead compared to the baseline.

Refresh Rate	**Image Quality**			**Condition Following**			**Lat.(s)**
	FID↓	Aesth.↑	CLIP↑	SSIM↑	Dino Sim.↑	Dist.↓	
0%	97.2	4.665	0.285	0.760	0.795	15.700	82.7
10%	95.8	4.692	0.287	0.772	0.814	15.820	85.7
30% (Ours)	**93.7**	**4.744**	**0.292**	**0.792**	**0.838**	**15.458**	92.4
50%	94.4	4.718	0.290	0.788	0.832	15.612	98.9
100%	105.3	4.512	0.276	0.714	0.748	17.240	108.4

Table 5: Performance and latency analysis of different DVF refresh rates. The 30% rate provides the best quality-speed trade-off, while all rates except 100% outperform OmniSVG (FT).

B.2.1 Transferability to LLM4SVG

We evaluate DVF and NCCE on MMSVG-Illustration using LLM4SVG [2], whose coordinates are serialized as text with the native Qwen tokenizer. DVF can be applied directly. To adapt NCCE, we identify coordinate spans, parse and normalize their numerical values, and construct neighborhood supervision over the corresponding token spans. All variants share the same backbone, training data, optimization configuration, and decoding strategy.

Method	FID↓	Aesth.↑	CLIP↑	SSIM↑	DINO↑	DT↓
LLM4SVG	122.5	4.511	0.267	0.698	0.752	25.360
LLM4SVG + NCCE	116.7	4.548	0.272	0.721	0.771	22.684
LLM4SVG + DVF	<u>105.8</u>	<u>4.632</u>	<u>0.284</u>	<u>0.756</u>	<u>0.801</u>	<u>18.740</u>
LLM4SVG + DVF + NCCE	**102.9**	**4.647**	**0.287**	**0.764**	**0.809**	**18.126**

Table 6: Transferability of DVF and NCCE to LLM4SVG on MMSVG-Illustration. Bold and underlined values indicate the best and second-best results, respectively.

Table 6 shows that both components improve the baseline, and their combination achieves the best results across all six metrics. These results support transfer beyond OmniSVG, although NCCE requires adaptation to the coordinate tokenization.

B.3 Details about Baselines

B.3.1 Controllable Image Generation

Since standard controllable image generation baselines do not inherently prioritize the synthesis of vector-like content, we augment the conditioning text prompts to better align their outputs with the SVG domain. Specifically, in addition to the original task-specific captions, we append a fixed set of stylistic suffixes: *"flat vector art, SVG style, simple cartoon style."* For instance, a typical prompt used for evaluation would be: *"A yellow smiley face is shown on a white background, flat vector art, SVG style, simple cartoon style."*

B.3.2 Text-to-SVG

For the OmniSVG baseline, we employ the default configuration of the OmniSVG-1.1 (3B) [1] model for text-to-SVG synthesis. Regarding SVGDreamer [19], we utilize the SIVE framework, where the number of vector paths for the foreground and background is strictly constrained to 32 each. Furthermore, the number of optimization iterations is set to 2,000 to ensure high-quality vectorization and convergence.

B.3.3 VLLMs

In this section, we present the specific prompts and instructions used for our framework.

```
# SVG Generation Instruction

# Task
Generate a high-quality SVG reconstruction based on the provided condition
image.  Your goal is to transform the raster input into precise vector
paths while maintaining structural and semantic integrity.

# Strict Constraints
   - Canvas Size:  The viewBox and dimensions must be exactly 200x200.
   - Output Format:  Output only the raw SVG code.
   - No Explanation:  Do not provide any commentary, introductory text, or
   concluding remarks.
   - Code Purity:  Provide raw XML/SVG data without conversational
   wrappers or Markdown code blocks.

# Technical Requirements
   - Structural Fidelity:  Vector paths must strictly align with the
   contours and boundaries defined in the condition image.
   - Color Mapping:  Sample hex color codes directly from the condition
   image to maintain visual consistency.
   - Layering Logic:  Order SVG elements from background to foreground to
   prevent incorrect overlapping of shapes.
   - Coordinate Clipping:  Ensure all path coordinates are normalized
   within the 0 to 200 range.
```

B.4 Details about User Study

We conducted a pairwise preference study to evaluate whether the generated SVGs better follow the input condition while maintaining acceptable visual quality. The study involved 13 participants with computer vision or computer graphics backgrounds. Each participant evaluated 30 comparison cases, including 15 icon cases and 15 illustration cases.

For each case, participants were shown the input condition and two anonymized SVG results in a randomized left-right order. One result was generated by ControlSVG and the other was generated by one of the compared baselines. Participants were asked to choose A, B, or Tie. The primary criterion was condition following, i.e., whether the generated SVG matched the structure and layout of the input condition; overall visual aesthetics was used as a secondary criterion when both results followed the condition similarly.

Participants were informed about the purpose and procedure of the study before taking part. Participation was voluntary and unpaid. We did not collect sensitive personal information, and the study only involved low-risk visual preference judgments over generated SVG results. The screenshot of the user-study interface is shown in Fig. 9.

C Extension Results on Scribble- and Image-to-SVG

Although our main benchmark focuses on Canny-to-SVG generation, ControlSVG is not restricted to a single type of visual condition. We further evaluate the model on scribble-to-SVG and image-to-SVG settings. These experiments are designed to test whether the proposed visual-feedback conditioning mechanism can generalize from edge-like structural conditions to noisier human-drawn scribbles and full raster images. As shown in Fig. 10 and Tab. 7, they show that ControlSVG remains competitive under both scribble-to-SVG and image-to-SVG settings. For scribble-to-SVG, ControlSVG achieves better condition-following metrics than GPT-5-Thinking and Gemini3-Pro on both Icon and Illustration subsets, indicating that the proposed visual-feedback mechanism is effective for sparse and noisy structural inputs. However, MLLMs obtain stronger image-quality and

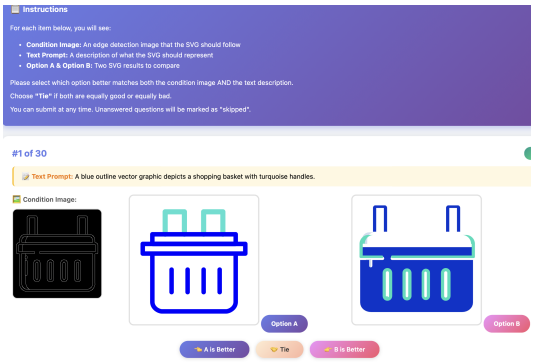

Figure 9: Screenshot of the user-study interface.

Task	Dataset	Method	Image Quality			Condition Following		
			FID↓	Aesth.↑	CLIP↑	SSIM↑	Dino Sim.↑	Dist. Trans.↓
Scribble-to-SVG	MMSVG-Icon	GPT-5-Thinking	133.8	4.724	0.299	0.452	0.598	44.630
		Gemini3-Pro	**124.6**	**4.841**	**0.311**	0.498	0.632	39.280
		ControlSVG (Ours)	151.9	4.586	0.276	**0.624**	**0.701**	**30.760**
	MMSVG-Illustration	GPT-5-Thinking	104.9	4.703	0.306	0.431	0.566	48.520
		Gemini3-Pro	**96.3**	**4.826**	**0.325**	0.477	0.614	42.670
		ControlSVG (Ours)	119.4	4.552	0.286	**0.587**	**0.673**	**35.740**
Image-to-SVG	MMSVG-Icon	GPT-5-Thinking	151.7	4.692	0.287	0.812	0.858	13.720
		Gemini3-Pro	**112.5**	**4.913**	**0.315**	**0.925**	**0.932**	**5.740**
		ControlSVG (Ours)	118.9	4.864	0.302	0.897	0.910	6.920
	MMSVG-Illustration	GPT-5-Thinking	101.6	4.641	0.307	0.781	0.829	20.330
		Gemini3-Pro	**85.7**	**4.861**	**0.329**	**0.872**	**0.895**	**9.420**
		ControlSVG (Ours)	89.4	4.792	0.317	0.846	0.881	10.800

Table 7: Quantitative comparison of GPT-5-Thinking, Gemini3-Pro, and ControlSVG (Ours) under Scribble-to-SVG and Image-to-SVG settings on MMSVGBench. **Bolding** indicates the best performance in each group, and underlining indicates the second-best performance.

semantic-alignment scores, likely due to their broader visual priors. For image-to-SVG, ControlSVG achieves performance close to Gemini3-Pro and generally outperforms GPT-5-Thinking, suggesting that our framework can extend beyond Canny-edge conditions, although full-image conditioning remains a challenging setting where large proprietary MLLMs retain advantages in some metrics. ControlSVG produces clean and structurally aligned SVGs under both types of conditions.

C.1 Evaluation on Human-Drawn Sketches

We further evaluate on real human-drawn sketches from TU-Berlin [29]. Since no paired target SVG is available, the input sketch serves as the reference for DINO similarity, SSIM, and Distance Transform (DT). DINO measures coarse structural agreement, while SSIM and DT measure finer spatial alignment. For human evaluation, eight computer-vision researchers each rank four anonymized outputs on all 30 examples, independently assessing visual quality and condition adherence from 1 (best) to 4 (worst). Each method receives $8 \times 30 = 240$ rankings per criterion.

Method	FID↓	Aesth.↑	CLIP↑	Visual rank↓	SSIM↑	DT↓	DINO↑	Adherence rank↓
GPT-5-Thinking	128.9	4.582	0.279	2.68	**0.446**	**61.76**	0.641	2.74
Gemini 3 Pro	**111.6**	4.724	**0.294**	**1.73**	0.388	70.12	0.682	1.76
OmniSVG (FT)	147.3	4.405	0.258	3.74	0.317	84.83	0.559	3.82
ControlSVG (Ours)	115.4	**4.738**	0.289	1.85	0.401	67.95	**0.689**	**1.68**

Table 8: Automatic metrics and mean human rankings on TU-Berlin. SSIM, DT, and DINO use the input sketch as the structural reference. Bold and underlined values indicate the best and second-best results, respectively.

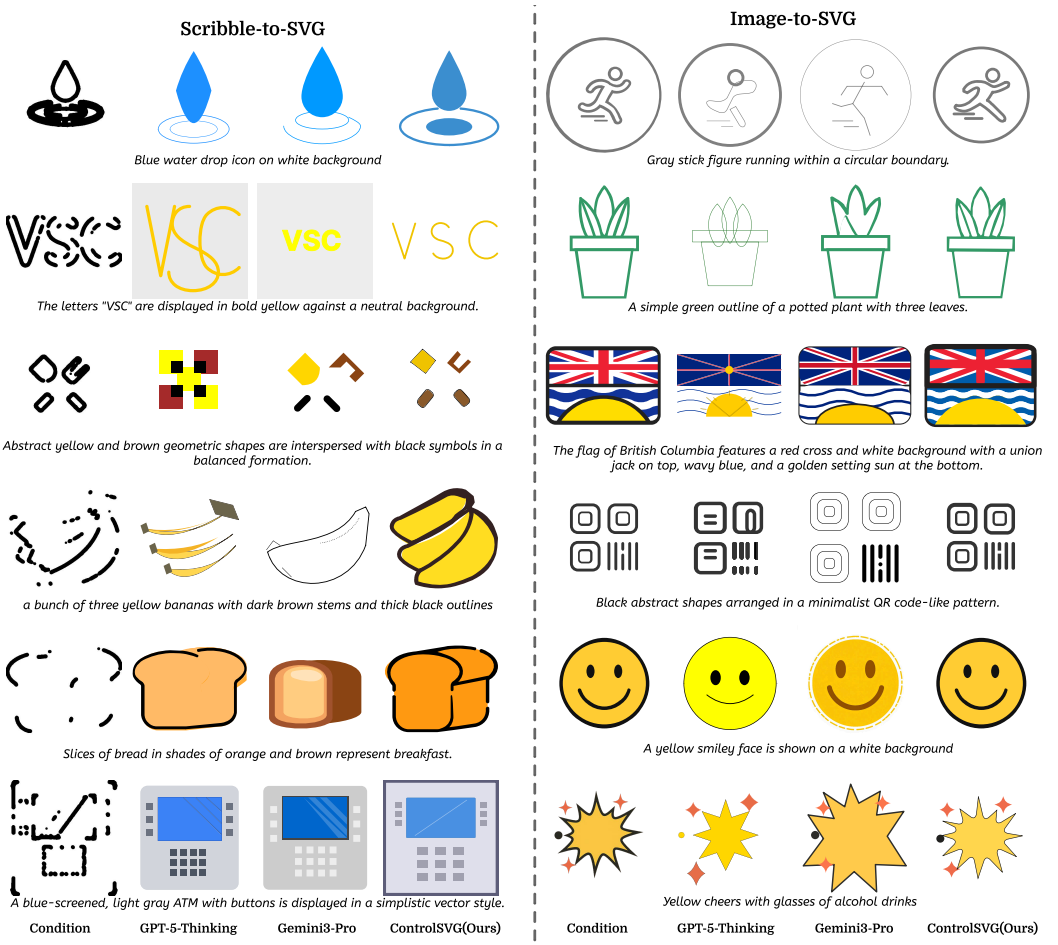

Figure 10: Qualitative comparison with GPT-5-Thinking [20] and Gemini3-Pro[21], on Scribble-to-SVG and Image-to-SVG tasks.

As shown in Table 8, ControlSVG achieves the best mean adherence rank and DINO similarity, while Gemini 3 Pro receives the best visual-quality rank. GPT-5-Thinking leads in SSIM and DT, indicating that fine spatial alignment and human-perceived adherence can favor different outputs when sketches omit details.

C.2 More Qualitative Comparisons

We provide more qualitative comparisons between our ControlSVG with the state-of-the-art methods in Figs. 11, 12. The compared baselines include:

1. **Condition-to-Image Generation + Vectorization**: SD3-ControlNet [10], ControlAR [7], and Flux-Kontext [17]. Since these models output raster images, we vectorize their outputs using LIVSS [18] with a fixed budget of 32 paths (average path number of native SVG generation methods) to ensure a fair comparison;

2. **Text-to-SVG Generation**: SVGDreamer [19] and OmniSVG [1].

3. **Multi-modal Large Language Models (MLLMs)**: GPT-5-thinking [20], Gemini-3-Pro [21], and Qwen3-VL-Max [22]. These models are prompted to directly synthesize SVG code based on the provided visual conditions and textual instructions.

C.3 Limitations

Despite the promising results, our current work has some limitations that provide directions for future research. First, our framework relies on a homogenized representation where all SVG elements are approximated using the `path` attribute. While versatile, encoding diverse primitives (e.g., text, circles, or rectangles) exclusively through path-based special tokens inevitably introduces redundancy. For instance, rendering typography via complex vector paths is significantly less efficient than utilizing native SVG `text` elements. Expanding the vocabulary to include a broader set of specialized SVG tokens could enhance both representation efficiency and semantic clarity. Second, our current model is constrained to single-condition control, requiring independent training for each specific spatial guidance signal. This limits the framework's flexibility in multi-task scenarios. Future work will focus on developing a unified model that leverages a multi-modal vision encoder to integrate precise spatial controls into the SVG generation process, primarily focusing on Canny-based control and further providing preliminary extensions to sketches, scribbles, and images. Third, the current implementation is built upon Qwen2-VL-3B-Instruct as the vision-language backbone. Although this backbone already provides effective multi-modal perception and SVG-code modeling ability, it may still limit the upper bound of visual reasoning, long-context modeling, and fine-grained coordinate prediction. Replacing the current backbone with stronger VLMs, such as Qwen3 or other models, may further improve both visual fidelity and condition following, especially for complex image-to-SVG and scribble-to-SVG cases. We therefore regard backbone scaling and stronger multi-modal SVG pretraining as important directions for future research.

C.4 Broader Impact

ControlSVG can support design and vector-content creation by allowing users to produce SVG files from lightweight spatial guidance such as hand-drawn sketches. This may reduce the manual effort required to create editable vector graphics, while users should still respect the licenses and terms of any source data, models, or downstream design assets used with the system.

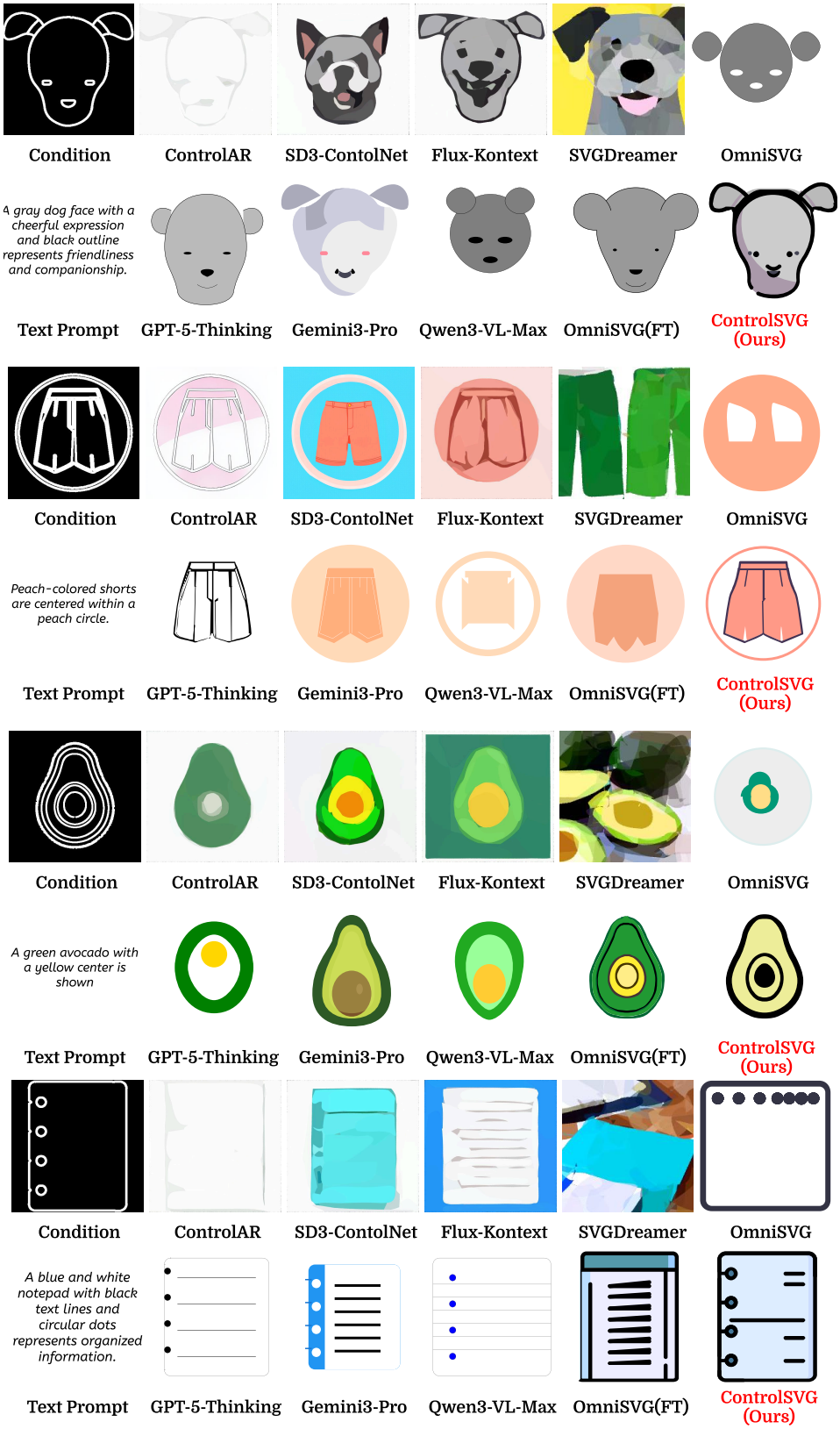

Figure 11: More qualitative comparisons between our ControlSVG and the state-of-the-art methods.

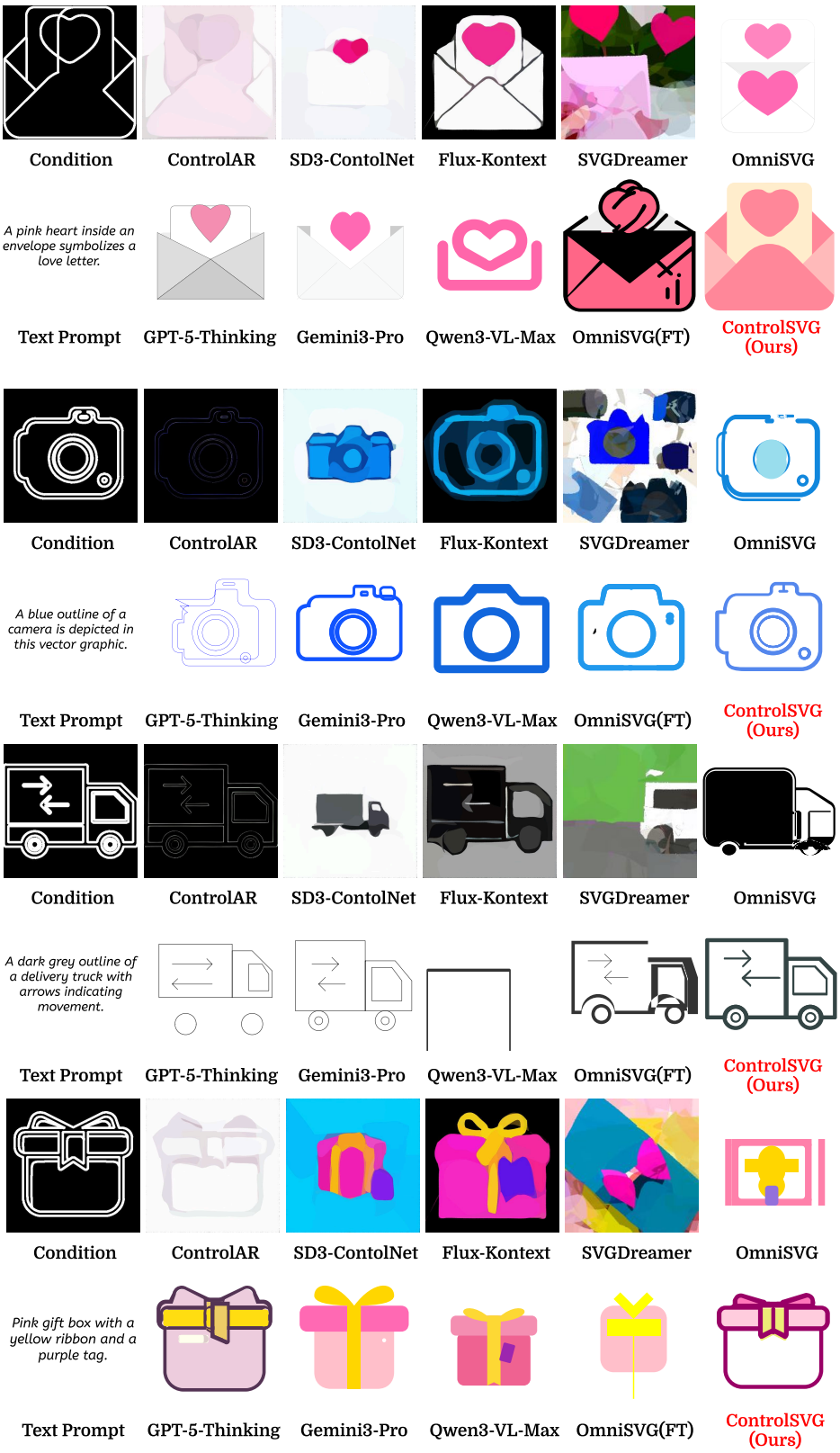

Figure 12: More qualitative comparisons between our ControlSVG and the state-of-the-art methods.