CODINGIMAGE: PROGRAMMABLE IMAGE EDITING VIA VECTOR GRAPHICS CODE

Anonymous authors
Paper under double-blind review

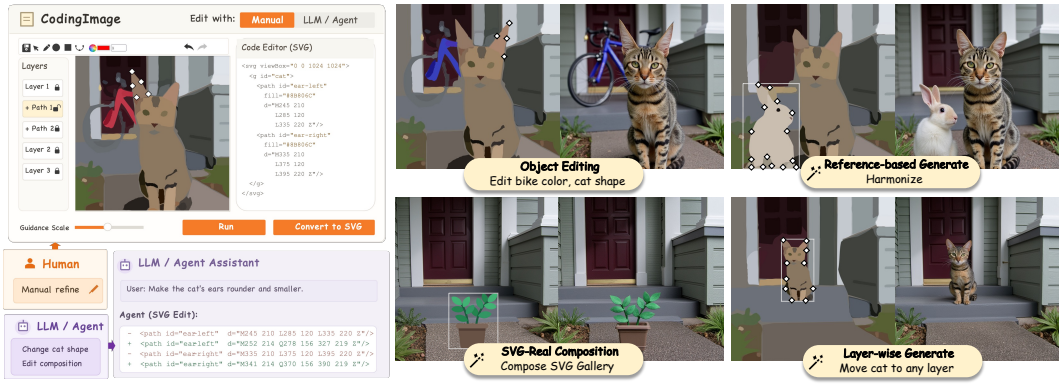


Figure 1: Overview of the proposed **CodingImage** framework. *(a)* An image is parsed into semantically named SVG paths, which an LLM/agent can rewrite from a natural-language instruction (*"rounder and smaller ears"*) and a human can refine by dragging control points. *(b)* Representative applications of the resulting SVGs. CodingImage offers (1) *Easy-to-control*: layer-wise object insertion, removal, modification, color adjustment, shape editing, and flexible composition; (2) *High fidelity*: caching semantic and color information through hierarchical SVG representations; and (3) *Strong input-generation alignment*: precise semantic and structural alignment between SVG inputs and generated images.

ABSTRACT

Fine-grained image editing calls for intuitive, element-level control. Scalable Vector Graphics (SVG) code offers a shared editing interface: large language models (LLMs) can rewrite it from natural-language instructions, while humans can directly refine its paths. However, conventional image-to-SVG methods face two challenges: their outputs may lack the semantic alignment and hierarchy needed for intuitive editing and manipulation, and pursuing high reconstruction fidelity can make vectorization prohibitively time-consuming. We introduce **CodingImage** to address both challenges. It efficiently parses images into hierarchical vector graphics (VGs) that are semantically aligned and structurally coherent. For photorealistic synthesis, we further propose Vector-Guided Noise Prediction, which conditions the diffusion model on these VGs and guides generated images to follow their structure and semantics. This enables precise control over geometry, color, and object semantics. Experiments demonstrate CodingImage's effectiveness in image editing, object-level manipulation, and fine-grained content creation.

1 INTRODUCTION

Recent progress in image generation (Rombach et al., 2022; Ho et al., 2020; Peebles & Xie, 2023; Labs, 2024) has led to remarkable advances in visual quality. However, generated images can still be difficult to refine at the element level: local details may not match user intent, and changing an object's shape or color, or adding and removing objects, often requires complex prompt engineering or specialized pipelines. These operations can also introduce artifacts or disrupt background consistency.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Prior work addresses controllability through sketches (Wang et al., 2021), layouts (Zhao et al., 2020; Sun & Wu, 2021), and drag-based interactions (Pan et al., 2023). While useful, these controls can be coarse or semantically unaware, making it difficult to edit an element and its attributes precisely. Recent work explores agentic image editing, with multimodal agents decomposing instructions, invoking editing tools, and iteratively refining results (Zeng et al., 2026; Zhu et al., 2026). GenClaw (Ye et al., 2026) uses code-generated sketches to guide agentic image generation. Our focus is an image-to-SVG-to-image editing loop: semantically structured SVG paths provide a shared interface for agent edits and human refinement, coupled with a trained SVG-to-image model.

However, to extract the SVG code from images, existing image-to-SVG (i.e., image vectorization) methods (Ma et al., 2022; Hirschorn et al., 2024; Wang et al., 2025) face two challenges: their outputs may lack the semantic alignment and hierarchy needed for intuitive editing and manipulation, and pursuing high reconstruction fidelity can make vectorization prohibitively time-consuming. This motivates a coding-centered question: can SVG code support fine-grained image generation and editing while remaining semantically structured and practical to produce?

To address this question, this work introduces **CodingImage**, a framework that makes image generation and editing programmable through SVG code. It efficiently parses images into hierarchical vector graphics (VGs) that are semantically aligned and structurally coherent, and our pipeline achieves a 7× speedup over LIVSS under the matched optimization settings in Table 2, while producing VGs that are easy to modify. Because these representations are SVG code, an LLM/agent can rewrite them from natural-language instructions (e.g., *"make the ears rounder and smaller"*), while a human can refine their paths directly. For photorealistic synthesis, we further propose *Vector-Guided Noise Prediction*, in which a tunable encoder predicts initial noise from VGs, aligning structural and semantic features to guide image generation. Together, these components translate edits to SVG code into photorealistic outputs and support applications including image editing, object-level manipulation, realistic and vector graphics composition, and localized de-artifacting. Our contributions are:

- We propose a controllable image generation framework, CodingImage, which is built upon a newly proposed *Simplified Vector Graphics* code. It offers interpretable, programmable representations that can be manipulated by both LLM/agent and human conveniently.
- We present *Vector-Guided Noise Prediction*, where a tunable encoder derives the spatially variant initial noise of diffusion models from vector graphics, thereby aligning structural and semantic features to translate editable SVG code into controllable photorealistic images.
- Experimental results show that our framework supports element-wise re-generation and modification, and significantly improves controllable image generation, enabling layout-wise generation, object-level editing, and composition.

2 PRELIMINARY

Flow models (Liu et al., 2023; Lipman et al., 2023; De Bortoli et al., 2022) learn a velocity field that transports noise to data. In latent text-to-image models such as FLUX (Labs, 2024), an image is encoded as $\boldsymbol{z}_0 = \mathrm{Enc}(I)$, and a DiT-style transformer (Peebles & Xie, 2023) predicts the latent velocity. Given Gaussian noise $\boldsymbol{z}_1 \sim \mathcal{N}(0, \mathbf{I})$, the interpolation path is

$$\boldsymbol{z}_t = (1-t)\boldsymbol{z}_1 + t\boldsymbol{z}_0, \quad t \in [0,1]. \tag{1}$$

Conditioned on c, the velocity network v_θ learns the target $\boldsymbol{z}_0 - \boldsymbol{z}_1$ through

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{z}_0, \boldsymbol{z}_1, t}\left[|v_\theta(\boldsymbol{z}_t, t, c) - (\boldsymbol{z}_0 - \boldsymbol{z}_1)|_2^2\right]. \tag{2}$$

At inference, integrating $d\boldsymbol{z}_t/dt = v_\theta(\boldsymbol{z}_t, t, c)$ from $t = 0$ to $t = 1$ transports noise to the data latent, which is decoded into an image.

Vector graphics (VG) represent images not as dense pixels but as geometric primitives such as points, lines, and curves. Among different vector primitives, Bézier curves are widely used for representing smooth boundaries. Most image vectorization methods model an image as a collection of closed regions, each defined by an ordered sequence of Bézier segments $\Omega = \{B_{i_1}, \ldots, B_{i_L}\}$ forming a continuous closed curve. Each Bézier segment $B_i(t)$ of degree M is parameterized as

$$B_i(t) = \sum_{j=0}^{M} \binom{M}{j} (1-t)^{M-j} t^j P_j^{(i)}, \quad t \in [0,1], \tag{3}$$

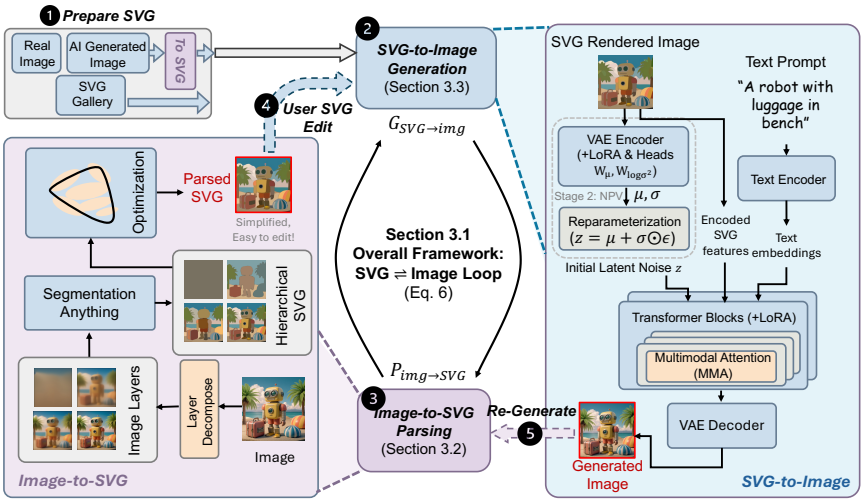


Figure 2: Overall framework of our CodingImage and its workflow. ① **Prepare SVG:** the input is obtained by converting a real or AI-generated image into SVG, or by selecting an existing SVG from a gallery. ② **SVG-to-Image:** the SVG information will be conditioned using token concatenation and noise prediction from vectors (NPV) module. The NPV module incorporates the SVG condition and integrates trainable LoRA adapters and prediction heads to estimate the mean and variance of the initial noise, rather than directly sampling from Gaussian noise. *If the user wishes to re-generate or modify specific parts, we proceed with steps* ③–⑤. ③ **Image-to-SVG:** the generated image is converted back into SVG using a diffusion model to produce multiple layers, followed by SAM to generate semantic masks for each layer, and further refined via 2D Gaussian optimization. ④ **SVG Editing:** the SVG is edited as code by an LLM/agent from an instruction, or by hand for exact geometry. ⑤ **Re-generation:** the modified SVG is used as guidance to synthesize the result.

where $P_j^{(i)} \in \mathbb{R}^2$ and we use cubic Bézier curves ($M = 3$) in all experiments.

The emergence of *differentiable vector graphics rendering* (DiffVG (Li et al., 2020) and Bézier Splatting (Liu et al., 2025)) makes it possible to propagate gradients through rasterization. This allows the control points to be treated as *optimizable parameters*, refined by minimizing the discrepancy between the rasterized result and the target image. A common objective is the pixel-wise reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \left\| I_{\text{real}} - R(\{\Omega_k\}) \right\|_1, \tag{4}$$

where R denotes the differentiable rasterizer applied to the set of regions $\{\Omega_k\}$. This formulation establishes a direct link between continuous vector representations and gradient-based optimization, providing the foundation for trainable and editable vector graphics.

3 Methodology

3.1 Overall Framework

Unlike prior controllable image generation models based on sketches, layouts, or depth maps (Wang et al., 2021; Zhao et al., 2020; Tan et al., 2025), this work adopts **vector graphics**, like SVG, as the conditioning representation, offering a more flexible, structured, and easy-to-edit description of visual content. Vector graphics inherently capture the hierarchical relationships between foreground objects and background elements within a scene, enabling richer control and composition. This structure makes them more intuitive for user manipulation, enabling direct editing and control over individual components. In addition, vector graphics seamlessly integrate with existing SVG libraries, allowing easy reuse and composition of existing rich design assets.

To this end, we design a loop of **SVG⇌image** consisting of two key components: (1) an *SVG-guided image generation module* that ensures the synthesized image faithfully follows both user-manipulated vector graphics and the accompanying text prompt (see Section 3.3), and (2) an *image-to-SVG parsing module* that converts images back into vector graphics with hierarchical semantic representations (see

Section 3.2). Together, these components form a closed loop that enables element-wise controllable generation and iterative refinement through user or agent edits. The SVG code serves as a shared editing interface: an LLM/agent can modify semantically meaningful paths from natural-language instructions, while a human can inspect and refine those same paths directly. Formally, the loop can be expressed as

$$\mathcal{S} \xrightarrow{G_{\text{SVG}\to\text{Img}}} \hat{\mathcal{I}} \xrightarrow{P_{\text{Img}\to\text{SVG}}} \hat{\mathcal{S}} \xrightarrow{\text{edit}} \hat{\mathcal{S}}' \xrightarrow{G_{\text{SVG}\to\text{Img}}} \hat{\mathcal{I}}', \tag{5}$$

where $\mathcal{S}$ denotes the input SVG representation, $\hat{\mathcal{I}}$ is the generated image, $\hat{\mathcal{S}}$ is the reconstructed SVG, $\hat{\mathcal{S}}'$ is the user-modified SVG, and $\hat{\mathcal{I}}'$ is the re-generated image.

3.2 Efficient and Simplified Vector Graphics

Unlike existing image vectorization approaches (Adobe Inc., 2025; Li et al., 2020; Ma et al., 2022), which primarily focus on reconstruction quality and often produce overly complex and fragmented SVGs without semantic alignment, existing designs lead to limited editability and low efficiency, making such representations unsuitable as conditioning inputs for image generation. To address this issue, we propose an Image-to-SVG parsing module that efficiently delivers an *structurally simple, semantically aligned, and hierarchical* vectorized representation, which allows users to easily identify and manipulate complete objects that match their intent, and select an object together with all of its sub-parts in a unified and intuitive manner. Figure 3 illustrates an example of hierarchical and simplified vector graphics generated by our method.

Hierarchical and semantic-aligned initialization. We leverage visual foundation models, including Segment Anything (Kirillov et al., 2023) and diffusion model (Rombach et al., 2022), to initialize hierarchical and semantically aligned vector graphics, inspired by prior layer decomposition methods such as LIVSS (Wang et al., 2025). Specifically, diffusion priors are used to generate a sequence of progressively simplified images $\{I_S^t\}_{t=1}^n$, where smaller t captures global structure and larger t preserves finer details. Each I_S^t is segmented by Segment Anything to obtain semantic masks, which are then assigned to layers based on simplification level, mask size, and overlap with previously assigned masks. Redundant masks with significant occlusion are discarded, yielding hierarchical masks

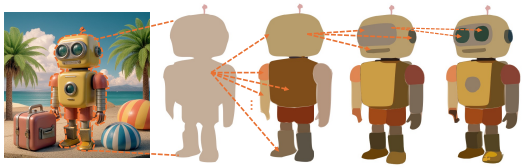


Figure 3: Our hierarchical and simplified vector graphics. As illustrated, the SVG is decomposed into the robot "body", which further contains semantic parts such as the "head", "upper body", "legs", and "shoes". The "head" is hierarchically subdivided into the "eye region", "ears", and "mouth", and the "eye region" is further decomposed into the "left eye" and "right eye".

$$\mathcal{M} = \{\mathcal{M}_k\}_{k=1}^{K}, \qquad \mathcal{M}_k = \left\{I_{\text{mask}}^{k,j}\right\}_{j\in\mathcal{J}_k}, \tag{6}$$

where k indexes the hierarchy level and $|\mathcal{J}_k|$ may vary across layers. Each mask is associated with a parent mask in the preceding layer. Building upon the hierarchical masks, we initialize each semantic region by polygonizing its boundary and converting it into an editable SVG path: we simplify the polygon (Douglas–Peucker), and if it remains overly complex, we split it at the longest diagonal and fit each sub-chain with a few cubic Bézier segments, limited at 8 segments per side to ensure editability. This compact, hierarchical SVG code makes edits local: an LLM/agent can target a semantic path, while a human can inspect the result and refine its geometry by editing the same path or its Bézier control points.

Efficient image vectorization via differentiable rasterization. To enable fast image vectorization within image-to-SVG loop, we adopt Bézier Splatting (Liu et al., 2025) as an efficient differentiable renderer. However, directly adopting Bézier Splatting introduces a challenge in preserving the initial hierarchical semantic alignment. the standard alpha blending formulation permits opacity manipulation, which may compromise semantic-region exclusivity and hierarchy during optimization.

To address this issue, we enforce a semantic-aligned rendering formulation in which each vector graphic exclusively belongs to a single semantic region within a layer. Since semantic regions are not additive under alpha blending, we fix the opacity of all regions to 1.0, preserving hierarchical semantic alignment by optimizing boundaries rather than opacity.

Under this formulation, we optimize the hierarchically initialized vector graphics $\{\Omega_k\}$ with explicit structural supervision. Specifically, we introduce a structure loss that matches each rendered SVG region to its corresponding semantic mask. The overall optimization objective is

$$\mathcal{L}(\{\Omega_k\}) = \sum_{k=1}^{K} \left\| I^k_{\text{mask}} - I^k_{\text{vec}} \right\| + \gamma \, \mathcal{L}_{\text{recon}}, \tag{7}$$

where I^k_{vec} denotes the SVG rendering at layer k, $\mathcal{L}_{\text{recon}}$ is the image-level reconstruction loss defined in Eq. 4, and γ balances structural alignment and photometric fidelity. This objective enforces semantic boundary consistency while maintaining accurate appearance reconstruction.

3.3 Vector-Guided Controllable Image Generation

Since our SVG representation preserves richer and more accurate structural information than prior conditions such as sketches or depth maps, we introduce a dedicated Noise Prediction from Vector (NPV) module along with a two-stage training strategy within the SVG-to-Image module. This design better aligns SVG conditions with the generated images, particularly with respect to color fidelity and boundary geometry. Prior training-free editing methods manipulate initial noise to preserve input fidelity (Rout et al., 2024; Meng et al., 2022), while FlashIn (Wang, 2026) learns an image-to-noise mapping for fast inversion. Our NPV module instead predicts the mean and variance of an initial noise distribution conditioned on a rendered SVG, guiding photorealistic synthesis from edited vector geometry. This allows the model to more effectively inject conditional information while preserving high fidelity in the generated results.

In detail, we introduce a **two-stage training strategy** for the SVG-guided image generation process $G_{\text{SVG}\to\text{Img}}$. In the **first stage**, we integrate LoRA modules into transformer blocks to adapt the original text-to-image model, *i.e.*, Flux.1-dev (Labs, 2024), to text and SVG conditioned generation model. The image branch encodes the rendered SVG into condition tokens, which are concatenated with noisy image tokens along the sequence dimension for joint processing. In parallel, the text branch encodes the accompanying description to provide semantic guidance. These two branches are fused through multimodal attention, allowing the model to jointly attend to visual and textual cues (Tan et al., 2025). At each attention layer, the text (t), latent (z), and SVG-condition (c) tokens are concatenated along the token dimension to form the joint queries, keys, and values: $Q = [Q_t; Q_z; Q_c]$, $K = [K_t; K_z; K_c]$, and $V = [V_t; V_z; V_c]$. The model then applies standard scaled dot-product attention, $\text{MMA}(Q, K, V) = \text{softmax}\Big(QK^T/\sqrt{d}\Big)\, V$, where d is the key dimension.

We then propose **Noise Prediction from Vectors (NPV)** as the **second stage** to further strengthen the structural alignment between SVGs and generated images. Specifically, we add a noise prediction module that estimates the mean and variance of the initial noise of flow matching model, such that the guidance from SVGs is directly injected at the beginning phase of the generation process. We extend the Flux model's VAE encoder with low-rank adapters (LoRA) and lightweight head tuning to obtain the latent distribution parameters. Given an input image I, the encoder produces the mean μ and the log-variance $\log \sigma^2$ as

$$\mu, \log \sigma^2 = \big(W_\mu, W_{\log \sigma^2}\big) * \phi\big(\text{Enc}_{\widetilde{\theta}}(I)\big)\,, \tag{8}$$

where $\text{Enc}_{\widetilde{\theta}}(\cdot)$ denotes the frozen Flux VAE encoder plugged with learnable LoRA modules, where the parameters are modified as $\widetilde{\theta} = \theta^{(0)} \cup \{\Delta W_l\}_l$ with LoRA updates injected into selected layers. Here, ϕ represents a GroupNorm layer followed by a SiLU activation, and $W_\mu, W_{\log \sigma^2}$ are parallel 1×1 convolutional heads predicting μ and $\log \sigma^2$, respectively.

With these parameters, we sample the latent code z via the reparameterization trick:

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{9}$$

The training objective consists of three components: the flow matching loss $\mathcal{L}_{\text{FM}}$ from Eq. 2, the KL loss $\mathcal{L}_{\text{KL}}$ to minimize the divergence from the standard normal distribution, and a covariance loss $\mathcal{L}_{\text{cov}}$ to encourage spatial independence across latent channels:

$$\mathcal{L}_{\text{KL}} = \frac{1}{2}\, \mathbb{E}\Big[\mu^2 + \sigma^2 - \log \sigma^2 - 1\Big]. \tag{10}$$



Figure 4: Our CodingImage supports various controllable image generation and editing tasks.

$$\mathcal{L}_{\text{cov}} = \frac{1}{C(C-1)} \sum_{i \neq j} R_{ij}^2, \qquad R = \frac{\tilde{\mu}\,\tilde{\mu}^\top}{\|\tilde{\mu}\|^2}, \tag{11}$$

where $\tilde{\mu}$ is the channel-normalized mean vector, and C is the number of elements in the window. To avoid the large memory overhead and artifacts caused by computing covariance with a global fixed window, we randomly select N patches of size $p \times p$ and compute the mean covariance loss over the patches. The final objective is a weighted combination, as:

$$\mathcal{L} = \mathcal{L}_{\text{FM}} + \beta\,\mathcal{L}_{\text{KL}} + \lambda\,\mathcal{L}_{\text{cov}}. \tag{12}$$

3.4 LLM/Agent-Assisted SVG Editing with Human Refinement

Recent multimodal SVG models such as DuetSVG (Zhang et al., 2026) combine visual and code representations for SVG generation. Our workflow connects code-edit suggestions and human refinement to photorealistic image synthesis. This subsection details the SVG-editing stage (Step 4 in Figure 2). For example, a user asks, "Make the kite teal and lengthen its tail, while keeping the beach unchanged." The system can send this prompt together with the current SVG code to a configured model or agent API, such as ChatGPT (GPT-5.6 Luna) (OpenAI, 2026) or Qwen Code (Qwen Code, 2026). The model is asked to identify the kite's semantic path, return a minimal SVG patch that changes its fill and geometry, and leave unrelated paths untouched. The user then previews the candidate, drags Bézier control points or adjusts SVG attributes in an editor, and approves the refined code. This SVG is passed to the SVG-to-Image module for photorealistic synthesis; if another edit is needed, the generated image can be parsed back into SVG and the loop repeated. These APIs are optional backends for code-edit suggestions, with human review and refinement retained. See Figure 8 in Appendix A for the detailed process.

4 Experiments

4.1 Experimental Setups

Implementation details. Our CodingImage builds on FLUX.1-dev (Labs, 2024), a latent rectified flow transformer for text-to-image generation. We apply LoRA (Hu et al., 2022) to the transformer blocks of the base model, using a default rank of 4, with the LoRA scale set to 0 by default for non-condition tokens, introducing approximately 14.5M trainable parameters for stage one. In addition, we apply LoRA with rank of 8 to the Flux VAE encoder at both the down and middle stages, introducing approximately 135K trainable parameters to predict the initial noise. Further training details are provided in Appendix B.

Dataset construction. We filter around 5M images and their corresponding text prompts from the LAION-400M dataset, retaining only those with resolutions larger than 1024×1024. All training data are resized to 512×512 for training. We then employ our proposed efficient Image-to-SVG module to convert these images into SVGs, thereby constructing {image, SVG, text} triplets for the training set. We adopt this dataset consistently for both stage one and stage two training processes.

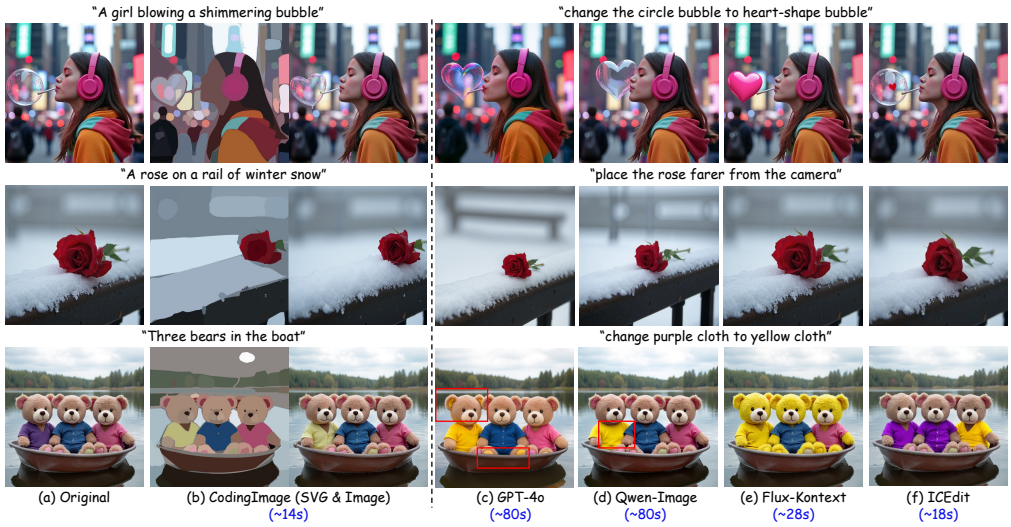


Figure 5: Visual comparisons with text-prompt-guided editing methods, including state-of-the-art open-source and commercial solutions such as GPT-4o (Hurst et al., 2024), Qwen-Image (Wu et al., 2025), Flux-Kontext (Labs et al., 2025), and ICEdit (Zhang et al., 2025). Editing cases such as shape modification, object repositioning, and color adjustment are readily supported by our VG representation, whereas text-guided editing often fails.



Figure 6: (a) Ablation study comparing results with and without the proposed Noise Prediction from Vectors (NPV) module, as well as training NPV module with different iterations. (b) PSNR and FID performance variations under different vector scales used to adjust the conditioning strength. By adjusting the vector scale, our method can flexibly adapt to complex appearance effects such as water reflections, smoke, and lighting variations, without being overly constrained by the SVG geometry.

4.2 Applications

Figure 4 showcases the applications supported by our CodingImage method, while Figure 5 compares its editing performance with recent state-of-the-art approaches. CodingImage enables a wide range of re-generation and editing tasks, including:

Human–LLM collaborative editing. CodingImage supports object replacement and changes to shape, color, and position, optionally guided by reference images (Figure 4(c)). In the experiments, we use GPT-5.6 Luna (OpenAI, 2026) in ChatGPT for initial SVG edits followed by human refinement. In the first example of Figure 5, ChatGPT reshapes the bubble into a heart in the converted SVG. The user refines its angle and size before SVG-guided image synthesis. Appendix A presents the complete human–LLM collaborative editing process, including image-to-SVG conversion, LLM editing, human refinement, and final image generation.

Layer-wise generation. CodingImage enables controllable layer-wise generation, where different semantic layers (*e.g.*, background, mid-level structures, and foreground details) can be synthesized independently. For example, the object can be positioned at any layer, such as placing the building behind existing trees, inserting furniture within the mid-level interior space. This design enables flexible scene composition while keeping local edits semantically consistent and visually faithful, through SVG that preserves semantic features via hierarchical layering.

Real & SVG composition. By leveraging vector-based SVG representations, our approach enables flexible composition of SVGs and real images within a single complex scene. Vector primitives can

Table 1: Quantitative comparison with controllable generation baselines across different representation types including sketch, depth, stroke, segmentation mask, and our proposed SVG, along with our variants trained with and without noise prediction from vectors (NPV). We report both reconstruction and editing quality, with the best results highlighted in **bold**.

Representation	Base Model	Reconstruction				Editing			
		FID↓	LPIPS↓	SSIM↑	PSNR↑	FID↓	LPIPS↓	SSIM↑	PSNR↑
Canny	FLUX.1	17.38	0.4582	0.42	12.28	22.48	0.5639	0.37	11.54
Depth	FLUX.1	19.65	0.5572	0.36	11.60	22.94	0.5932	0.36	11.37
Stroke	FLUX.1	22.42	0.4721	0.45	17.13	25.35	0.5249	0.44	14.87
Segmentation Mask	FLUX.1	17.75	0.4513	0.43	15.78	21.97	0.5382	0.43	14.34
Ours (SVG) w/o NPV	FLUX.1	16.03	0.3687	0.47	17.42	18.87	0.3901	0.47	17.25
Ours (SVG) w NPV	FLUX.1	**15.52**	**0.3515**	**0.49**	**17.77**	**17.84**	**0.3836**	**0.50**	**17.72**

be rearranged, combined, or modified to form realistic scenes such as the cat in a bookstore shown in Figure 4(a), and also allow the insertion of realistic objects (e.g., the man) into SVG-based conditions.

Localized de-artifacts. CodingImage can also address visual artifacts. When fine-grained details in the initial generation, such as fingers, are rendered incorrectly (*e.g.*, wrong number of fingers), manual adjustments or SVG-based guidance enable the re-generated image to be refined and corrected as shown in Figure 7.

4.3 Method Comparison

To evaluate the representation ability and controllability of CodingImage, we construct an evaluation set by collecting 5K paired source–target images from existing datasets (Yu et al., 2025). Each source image is converted into five structural conditions, including Canny edges, depth maps, strokes (Meng et al., 2022), Segmentation Mask (Kirillov et al., 2023), and our proposed SVG representation, using publicly available methods. We then perform conditional generation to reconstruct the source image from each condition. Specifically, we adopt the well-trained Canny- and depth-conditioned models released in OminiControl (Tan et al., 2025), while the stroke and segmentation mask-conditioned model are trained with the same data as our SVG-conditioned model. All methods, including ours, are built upon the FLUX.1-dev base model. For evaluation, we measure reconstruction quality with FID (Heusel et al., 2017), PSNR (Hore & Ziou, 2010), SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). Beyond reconstruction quality, we assess controllability by examining the editability of different representations. Given source–target pairs, we compute modification masks and automatically apply them to the representations (*e.g.*, mask $\times$ $\text{target}_{\text{SVG}} + (1 - \text{mask}) \times \text{source}_{\text{SVG}}$). The edited representations are then used for conditional generation, and the resulting images are compared against the target images to quantify editing performance. Table 1 summarizes the reconstruction and editing results, demonstrating SVG's effectiveness for information caching and editing.

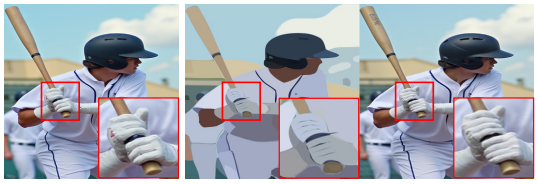


Figure 7: Our CodingImage can be applied to artifact removal. Fine-grained elements in the original generation, such as fingers, are sometimes rendered incorrectly (*e.g.*, **three fingers** instead of four in a frontal view). Manual corrections or guidance in SVG allow the re-generated image to be refined to eliminate such artifacts.

4.4 Ablation Study

The effect of noise prediction. In stage one, training treats SVGs as conditional inputs via token concatenation. Stage two adds noise prediction through the NPV module, further aligning SVG structure with generated outputs. Table 1 compares reconstruction quality with and without NPV, showing that incorporating NPV leads to better alignment in the generated results. Besides, as shown in Figure 6(a), incorporating NPV makes the structure of the generated results align more closely with the input SVG, for example, the mountain silhouette. Importantly, the mountain's reflection in the generated image still obeys symmetry about the waterline, as encoded by the base model's

physics-consistent prior, despite a non-physical input SVG (*e.g.*, asymmetric reflection edges). With more stage-two training iterations, adherence to the input structure strengthens, sometimes forcing physically implausible reflections.

Effect of vector scale adjustment. We adjust the strength of the SVG condition by varying a scaling factor, where we plot the PSNR and FID metrics under different scaling values, demonstrating how the factor influences both reconstruction quality and perceptual fidelity as shown in Figure 6(b).

Table 2: Comparison of reconstruction quality and efficiency between the existing semantic-layered image vectorization method LIVSS (Wang et al., 2025) and our proposed efficient, semantically simplified image vectorization module.

Method	PSNR ↑	Time (s) ↓	Speedup
LIVSS (Wang et al., 2025)	16.960	18.564	1.0×
Our image vectorization	**17.353**	**2.656**	**7.0**×

Effect of efficient and semantic-aligned vector graphics. To assess the effectiveness of our efficient and semantic-aligned image-to-SVG module, we compare it with the previous semantic-layered vectorization method LIVSS (Wang et al., 2025) on the DIV2K dataset (Agustsson & Timofte, 2017), which contains 1K high-quality images. All methods are optimized for 30 steps and resized to 512×512 resolution for a fair comparison. Table 2 reports results on both efficiency and reconstruction quality. The results demonstrate the advantages of our compact initialization and efficient Bézier Splatting rasterizer: under the same 30 optimization steps, our method achieves higher PSNR while being $7\times$ faster.

5 Related Work

Image vectorization. Traditional tools such as Illustrator's Image Trace are non-differentiable and often produce complex shapes that are difficult to edit. DiffVG (Li et al., 2020) introduces differentiable rasterization, enabling gradient-based optimization of Bézier curves. Building on this, LIVE (Ma et al., 2022) and O&R (Hirschorn et al., 2024) develop initialization and optimization strategies for compact, topology-preserving representations. LIVSS (Wang et al., 2025) further integrates Segment Anything (SAM) (Kirillov et al., 2023) and diffusion priors to align vector layers with semantic content. More recently, AmodalSVG (Hu et al., 2026) recovers complete objects as editable vector layers, and Vector Scaffolding (Lee et al., 2026) coordinates optimization across scales to improve vectorization. We combine Bézier Splatting (Liu et al., 2025) with compact semantic paths and structural supervision to produce editable conditions for photorealistic generation.

Controllable image generation. Controllable generation in diffusion models has progressed from early text-to-image systems (Rombach et al., 2022; Saharia et al., 2022) to spatially guided methods such as ControlNet (Zhang et al., 2023); UniControl (Qin et al., 2023) further unifies diverse spatial conditions under a Mixture-of-Experts (MoE) paradigm. However, because these approaches inject spatial condition features into the denoising hidden states, they are best suited to aligned inputs and struggle with misaligned or subject-driven generation. Extensions like IP-Adapter (Ye et al., 2023) (cross-attention with an auxiliary encoder) strengthen identity preservation, yet most methods still operate in pixel space (Qi et al., 2023; Ruiz et al., 2023; Tumanyan et al., 2023; Brooks et al., 2023; Kim et al., 2022; Kawar et al., 2023). Recent layered approaches include Qwen-Image-Layered (Yin et al., 2026), which decomposes RGB images into editable RGBA layers, and MagicQuill V2 (Liu et al., 2026), which separates content, spatial, structural, and color cues. Our interface represents object geometry and appearance as SVG code that LLMs can directly read and modify from natural-language instructions. Vector-conditioned noise prediction translates these code-level edits into photorealistic images, enabling programmable image editing.

6 Conclusion

In this work, we have presented a new paradigm for controllable image generation based on semantic-aligned vector graphics. By decomposing images into hierarchical, semantically aligned vector representations and integrating them into a noise-guided generation framework, our method enables precise element-level control over geometry, color, and object semantics. The proposed approach not only delivers photorealistic outputs consistent with user edits but also supports a wide range of applications, from intuitive image editing to fine-grained object manipulation.

AI USE STATEMENT

We used generative AI tools to polish the manuscript's language, check its formatting, and provide guidance on the preparation and layout of selected figures. The authors are responsible for reviewing AI-assisted edits and ensuring the accuracy of the final manuscript.

REFERENCES

Adobe Inc. Adobe illustrator. https://www.adobe.com/products/illustrator.html, 2025.

Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 126–135, 2017.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18392–18402, 2023.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. *Advances in neural information processing systems*, 35:2406–2422, 2022.

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.

Or Hirschorn, Amir Jevnisek, and Shai Avidan. Optimize & reduce: a top-down approach for image vectorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 2148–2156, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pp. 2366–2369. IEEE, 2010.

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

Juncheng Hu, Ziteng Xue, Guotao Liang, Anran Qi, Buyu Li, Sheng Wang, Dong Xu, and Qian Yu. Amodalsvg: Amodal image vectorization via semantic layer peeling, 2026. URL https://arxiv.org/abs/2604.10940.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6007–6017, 2023.

Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2426–2435, 2022.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Jaerin Lee, Kanggeon Lee, and Kyoung Mu Lee. Vector scaffolding: Inter-scale orchestration for differentiable image vectorization, 2026. URL https://arxiv.org/abs/2605.11913.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Max Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xi Liu, Chaoyi Zhou, Nanxuan Zhao, and Siyu Huang. B\'ezier splatting for fast and differentiable vector graphics rendering. *arXiv preprint arXiv:2503.16424*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Zichen Liu, Yue Yu, Hao Ouyang, Qiuyu Wang, Shuailei Ma, Ka Leong Cheng, Wen Wang, Qingyan Bai, Yuxuan Zhang, Yanhong Zeng, Yixuan Li, Xing Zhu, Yujun Shen, and Qifeng Chen. Magicquill v2: Precise and interactive image editing with layered visual cues. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22467–22477, June 2026.

Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16314–16323, 2022.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

OpenAI. GPT-5.6 System Card. https://deploymentsafety.openai.com/gpt-5-6, July 2026. Published July 9, 2026.

Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 conference proceedings*, pp. 1–11, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15932–15942, 2023.

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.

Qwen Code. Rest api integration guide. https://qwenlm.github.io/qwen-code-docs/en/developers/rest-api-integration/, 2026. Accessed September 26, 2026.

Joan Rodriguez, Haotian Zhang, Abhay Puri, Haoran Dai, Tianyang Zhang, Meng Lin, Rishav Pramanik, Xiaoqing Xie, Marco Terral Rodriguez, Darsh Kaushik, Aly Shariff, Perouz Taslakian, Spandana Gella, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Vectorgym: A multi-task benchmark for svg code generation, sketching and editing, 2026. URL https://arxiv.org/abs/2603.29852.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Semantic image inversion and editing using rectified stochastic differential equations. *arXiv preprint arXiv:2410.10792*, 2024.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22500–22510, 2023.

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.

Wei Sun and Tianfu Wu. Learning layout and style reconfigurable gans for controllable image synthesis. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5070–5087, 2021.

Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. Ominicontrol: Minimal and universal control for diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14940–14950, 2025.

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1921–1930, 2023.

Guangzhi Wang. Flashin: Fast and accurate image inversion for real-time image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30425–30434, June 2026.

Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Sketch your own gan. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 14050–14060, 2021.

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*, pp. 7728–7738, 2025.

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.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Junyan Ye, Jun He, Zilong Huang, Dongzhi Jiang, Xuan Yang, Rui Chen, and Weijia Li. Genclaw: Code-driven agentic image generation, 2026. URL https://arxiv.org/abs/2605.30248.

Shengming Yin, Zekai Zhang, Zecheng Tang, Kaiyuan Gao, Xiao Xu, Kun Yan, Jiahao Li, Yilei Chen, Yuxiang Chen, Heung-Yeung Shum, Lionel M. Ni, Junyang Lin, and Chenfei Wu. Qwen-image-layered: Towards inherent editability via layer decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16196–16205, June 2026.

Qifan Yu, Wei Chow, Zhongqi Yue, Kaihang Pan, Yang Wu, Xiaoyang Wan, Juncheng Li, Siliang Tang, Hanwang Zhang, and Yueting Zhuang. Anyedit: Mastering unified high-quality image editing for any idea. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26125–26135, 2025.

Ziyun Zeng, Hang Hua, and Jiebo Luo. Mira: Multimodal iterative reasoning agent for image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9563–9573, June 2026.

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*, pp. 3836–3847, 2023.

Peiying Zhang, Nanxuan Zhao, Matthew Fisher, Yiran Xu, Jing Liao, and Difan Liu. Duetsvg: Unified multimodal svg generation with internal visual guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10219–10229, June 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. In-context edit: Enabling instructional image editing with in-context generation in large scale diffusion transformer. *arXiv preprint arXiv:2504.20690*, 2025.

Bo Zhao, Weidong Yin, Lili Meng, and Leonid Sigal. Layout2image: Image generation from layout. *International journal of computer vision*, 128(10):2418–2435, 2020.

Zichen Zhu, Yuheng Sun, Mingxuan Zhu, Wenjie Ma, Situo Zhang, Zhexiang Wang, Ziyue Yang, Danyang Zhang, Kunyao Lan, Zihan Zhao, Dingye Liu, Siqi Xiang, Lu Chen, and Kai Yu. Iea: Amateur-friendly conversational image editing agent via three stages of multitask alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 8672–8683, June 2026.

A HUMAN-IN-THE-LOOP SVG EDITING

Figure 8 illustrates the collaborative editing process for the first-row example in Figure 5. We use GPT-5.6 Luna (OpenAI, 2026) in ChatGPT to produce the initial SVG edit from the user's instruction; the user then refines the heart's orientation and size before image synthesis.

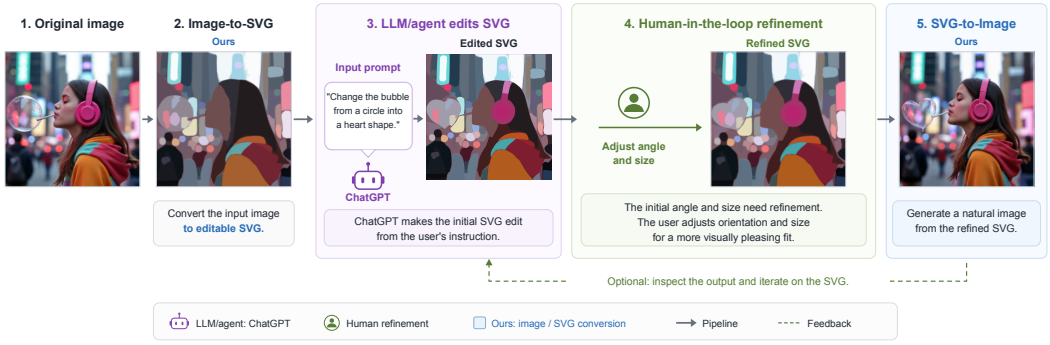


Figure 8: Human-in-the-loop SVG editing. (1–2) Our Image-to-SVG module converts the original image into editable SVG. (3) ChatGPT initially reshapes the round bubble into a heart according to the user's prompt. (4) The user inspects this edit and refines the heart's orientation and size, yielding the refined SVG shown here. (5) Our SVG-to-Image module generates the final natural image. The dashed arrow indicates optional further SVG editing after inspecting the output. The displayed SVGs are rasterized previews of the editable vector representations.

B TRAINING DETAILS

Our model is trained with a batch size of 1 and gradient accumulation over 4 steps. We employ the Prodigy optimizer with safeguard warmup and bias correction enabled, setting the weight decay to 0.01. We set hyperparameter $\gamma = \beta = \lambda = 1$ and the number of sampled patches as $N = 8192$ and patch size $p = 4$. All experiments are conducted on 4 NVIDIA A100 GPUs (80GB each). For stage one, our model is trained for 50K iterations, while stage two is trained for 10K iterations.

C ADDITIONAL RESULTS

C.1 VISUAL COMPARISONS ON VARIOUS GUIDANCE SCALE

We employ the guidance scale factor to control the alignment between the SVG conditions and the generated results. This factor also governs how the edited object interacts with its surrounding environment. Specifically, it determines whether the generation should adhere more strictly to the input SVG—enabling object-level edits without modifying shadows or reflections—or whether it should instead prioritize the physical realism enforced by the baseline model as shown in Figure 9. Users can flexibly adjust the guidance scale for different scenarios.

C.2 MORE RESULTS OF NOISE PREDICTION FROM VECTOR

To evaluate the effectiveness of NPV, we also integrate it into other conditional generation settings (i.e., Canny and Stroke). The results in the table below (with and without NPV) show that NPV generally improves reconstruction quality across different conditioning types as shown in Table 3. We further apply NPV to a general editing method (i.e., Flux-Kontext), where fidelity and generative diversity naturally form a trade-off in editing tasks, the resulting performance varies depending on the specific case as shown in Figure 10.

Table 3: Ablation studies on adding our proposed Noise Prediction from Vector (NPV) on various conditional-generation tasks, including Canny, Stroke, and our proposed SVG conditions.

Representation	Base Model	Reconstruction				Editing			
		FID↓	LPIPS↓	SSIM↑	PSNR↑	FID↓	LPIPS↓	SSIM↑	PSNR↑
Canny	FLUX.1	17.38	0.4582	0.42	12.28	22.48	0.5639	0.37	11.54
Canny w NPV	FLUX.1	16.89	0.4501	0.44	12.65	21.76	0.5573	0.41	12.32
Stroke	FLUX.1	22.42	0.4721	0.45	17.13	25.35	0.5249	0.44	14.87
Stroke w NPV	FLUX.1	21.88	0.4682	0.47	17.33	25.01	0.5204	0.46	15.13
Ours (SVG) w/o NPV	FLUX.1	**16.03**	**0.3687**	**0.47**	**17.42**	**18.87**	**0.3901**	**0.47**	**17.25**
Ours (SVG) w NPV	FLUX.1	**15.52**	**0.3515**	**0.49**	**17.77**	**17.84**	**0.3836**	**0.50**	**17.72**



Figure 9: Our CodingImage framework allows flexible adjustment of the guidance scale to control how strongly generation aligns with the input conditions. This scale determines whether generation should adhere closely to the SVG and allow precise object-level edits without altering shadows or reflections, or prioritize the physical realism captured by the base model. For example, users can control whether a mountain's reflectance follows its true geometry or the SVG guidance.

C.3 Compare Our Image-to-SVG with Image Vectorization Methods

We provide several visual comparisons to highlight the differences between our SVG representation and existing image vectorization methods. Methods that optimize primarily for pixel fidelity can indeed preserve fine details accurately, but they often produce highly complex SVG curves, making user editing difficult. In contrast, our Image-to-SVG module is designed to prioritize semantic object structures, enabling much more intuitive and convenient modifications. For instance, as shown in Figure 11, if a user wishes to adjust the shape or position of an object, such as a girl's arm, our SVG representation makes these edits significantly easier.

C.4 More Qualitative Results Across Various Applications

We provide more visual results to evaluate the performance of our CodingImage framework on both image generation and vector-based image editing tasks. The results demonstrate that our method can faithfully generate high-quality images from vector representations while preserving fine-grained details during editing operations. Figure 12 presents several representative examples, including original SVG inputs, generated images, edited SVG and edited results. These examples illustrate the robustness of CodingImage in maintaining structural and detail consistency, and producing visually coherent results even under complex editing scenarios.

C.5 Qualitative Results of Our Efficient and Simplified Vector Graphics

We provide visual examples to demonstrate the results of our proposed image-to-SVG module. Figure 13 shows several examples, including input images, vectorized outputs. The results demonstrate that our approach achieves a favorable trade-off between representation compactness and visual fidelity, yielding a user-friendly, semantically aligned SVG representation.

C.6 Limitations

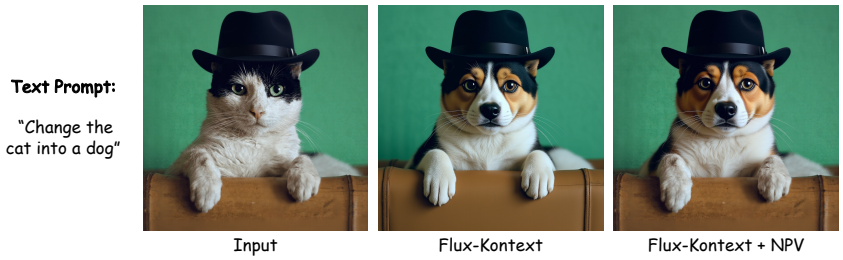


Figure 10: An example of integrating our Noise Prediction from Vector (NPV) module into a general editing method (i.e., Flux-Kontext). NPV typically **enhances fidelity** during the editing process. For example, the **texture of the table below and the shape and position of the animal's grasping hand are better preserved**.



Figure 11: Comparison between our Image-to-SVG (I2S) and Adobe's Image-to-SVG converter. Our I2S module is **explicitly optimized around semantic objects**, making it much **easier** for users to modify meaningful parts of the image. For example, if a user wants to adjust **the shape or position of an arm**, our SVG structure is significantly simpler than existing vectorization methods, which greatly facilitates intuitive editing.

Although multimodal models increasingly support SVG generation and editing, reliable execution of complex instructions remains challenging, as highlighted by VectorGym (Rodriguez et al., 2026). Our workflow therefore retains human inspection and refinement of suggested SVG edits to check geometry and preserve unrelated elements. This interaction limits full automation, and errors in image-to-SVG parsing can also propagate to subsequent edits. Improving code-edit reliability and semantic parsing is an important direction for future work.

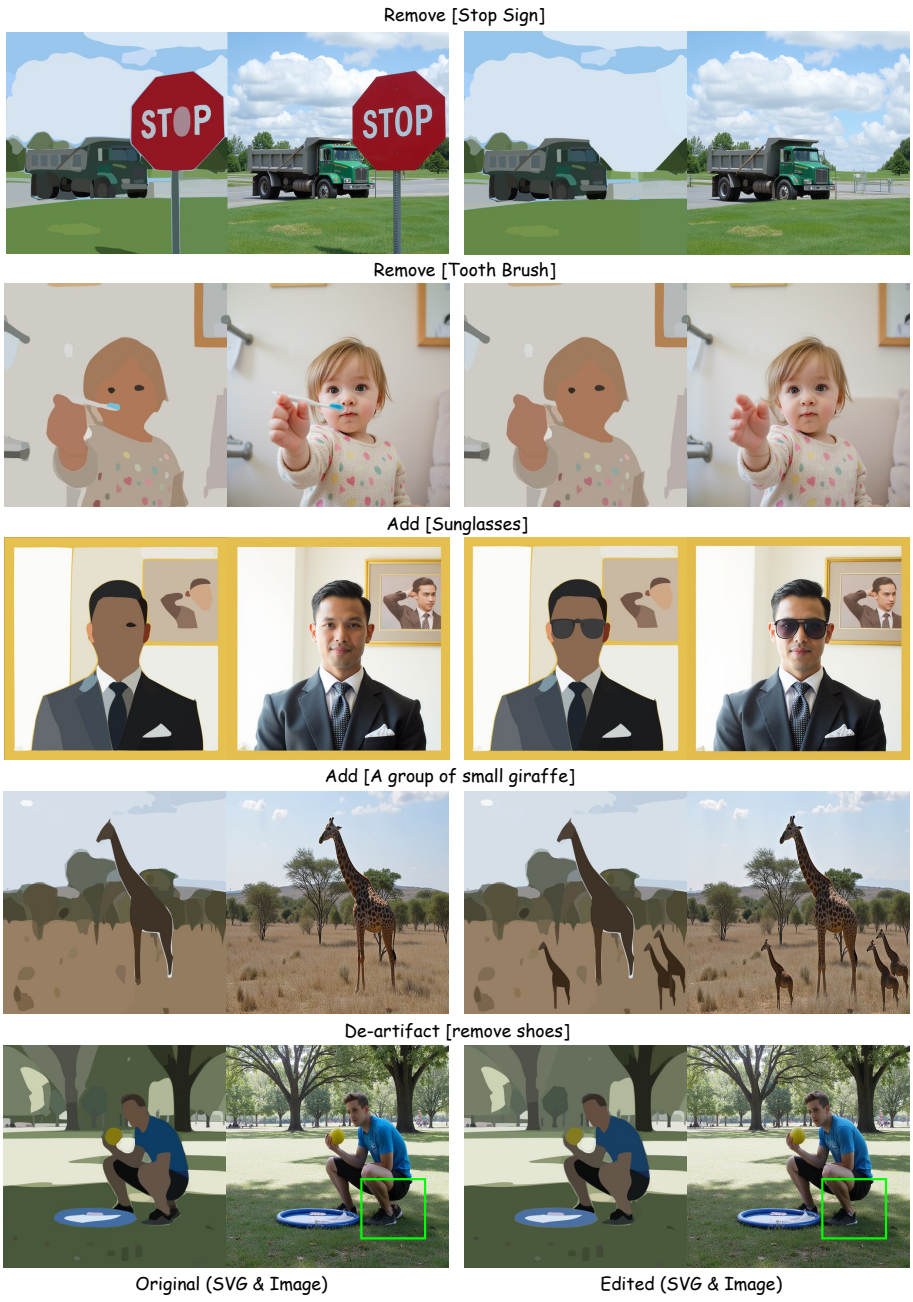


Figure 12: Images generated and re-generated using CodingImage. Each row shows SVG input, generated image, edited SVG and edited image, highlighting both generation fidelity and editing flexibility.



Figure 13: Real-world images in the first and third columns are sampled from the DIV2K (Agustsson & Timofte, 2017) dataset, while the second and fourth columns show their corresponding SVG conversions.