Chat2SVG: Vector Graphics Generation with Large Language Models and Image Diffusion Models

Ronghuan Wu
City University of Hong Kong
rh.wu@my.cityu.edu.hk

Wanchao Su
Monash University
wanchao.su@monash.edu

Jing Liao*
City University of Hong Kong
jingliao@cityu.edu.hk

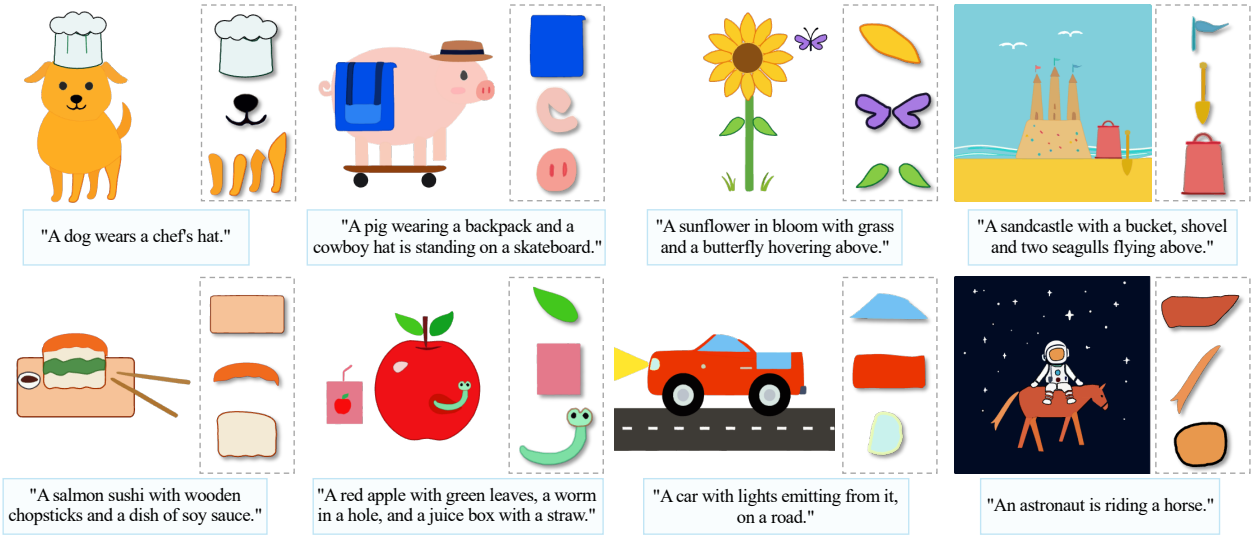

Figure 1. SVG examples generated by our Chat2SVG. We highlight some shapes to demonstrate semantic clarity and path quality.

Abstract

Scalable Vector Graphics (SVG) has become the de facto standard for vector graphics in digital design, offering resolution independence and precise control over individual elements. Despite their advantages, creating high-quality SVG content remains challenging, as it demands technical expertise with professional editing software and a considerable time investment to craft complex shapes. Recent text-to-SVG generation methods aim to make vector graphics creation more accessible, but they still encounter limitations in shape regularity, generalization ability, and expressiveness. To address these challenges, we introduce Chat2SVG, a hybrid framework that combines the strengths of Large Language Models (LLMs) and image diffusion models for text-to-SVG generation. Our approach first uses an LLM to generate semantically meaningful SVG templates from basic geometric primitives. Guided by image diffusion models, a dual-stage optimization pipeline refines paths in latent space and adjusts point coordinates to enhance geometric complexity. Extensive experiments show that Chat2SVG outperforms existing methods in visual fidelity, path regularity, and semantic alignment. Additionally, our system enables intuitive editing through natural language instructions, making professional vector graphics creation accessible to all users. Our code is available at https://chat2svg.github.io/.

1. Introduction

Scalable Vector Graphics (SVG), a vector image format based on geometric shapes, has become the standard for modern digital design, offering resolution independence and precise control over individual elements. However, creating high-quality SVG content remains challenging for non-expert users, as it requires both expertise with professional design software and considerable time investment to create complex shapes. To make vector graphics creation more accessible, recent research has focused on developing text-to-SVG systems that enable users to express their creative ideas through simple text prompts rather than complex manual editing.

Existing approaches have explored *image-based* meth-

*Corresponding Author.

ods for text-to-SVG generation, iteratively optimizing a large collection of shape elements (*e.g.*, cubic Bézier curves, typically 100 to 1000) by rendering them into images with differentiable rasterizers [20] and evaluating them using text-image similarity metrics like CLIP loss [28] and Score Distillation Sampling (SDS) loss [27]. While these image-based methods [9, 16, 43, 44, 50] can generate visually impressive SVG through the combination of numerous paths and powerful image models, they face a critical limitation in maintaining the regularity and semantics of shapes. Specifically, semantic components that should be represented as single elements often end up fragmented across multiple overlapping paths. Although these fragmented paths, when viewed collectively, can yield visually appealing outputs, they fundamentally conflict with professional design principles, where each semantic component is intentionally crafted as a single, regularized path.

Given SVG is defined using Extensible Markup Language (XML), another solution to text-to-SVG synthesis involves *language-based* methods, which treat SVG scripts as text input. Recent works [35, 41] have proposed specialized tokenization strategies to embed SVG scripts and trained Sequence-To-Sequence (seq2seq) models on domain-specific datasets (*e.g.*, icons and fonts). While these approaches achieve good generation quality within their training domains, they suffer from limited generalization due to the absence of large-scale, general-purpose text-SVG training data. Meanwhile, some works [26, 31] show that while state-of-the-art Large Language Models (LLMs) can generate basic geometric shapes (*e.g.*, circles and rectangles) with layouts matching text prompts, they struggle to produce the complex geometric details required for professional SVG applications. Consequently, the limited generalization and poor expressiveness of language-based methods hinder their adoption for text-to-SVG generation.

To address the aforementioned problems, we propose a novel hybrid framework that leverages the complementary strengths of *image-based* and *language-based* methods. Our approach first utilizes an LLM to synthesize SVG templates composed of basic primitives and then optimizes their geometric details guided by image diffusion models. This hybrid framework addresses the key limitations of both paradigms: The LLM-based template generation overcomes the domain-specific generalization constraints of traditional language models while ensuring shape regularity, as each element naturally corresponds to a single semantic component. The subsequent image-based optimization then enhances the expressiveness of these well-structured templates by capturing complex geometric details that LLMs alone struggle to produce. Specifically, to fully exploit LLMs' ability to create reasonable and complex layouts that match text prompts, we design an SVG-oriented prompt with multiple stages, including prompt expansion, SVG script generation, and visual refinement. To refine the generated SVG templates, we implement a dual-stage optimization strategy: (1) We first use an image diffusion model to synthesize detailed image-level targets, then follow [50] to optimize path-level latent vectors with a pre-trained SVG VAE, eliminating common issues such as self-intersections and jagged curves; (2) We further refine geometric details by directly optimizing the point coordinates to capture fine-grained visual elements. Comprehensive experiments show that Chat2SVG consistently outperforms existing methods regarding overall visual quality, individual path regularity, and semantic coherence. Furthermore, our system enables intuitive editing through iterative natural language instructions, making vector graphics creation more accessible to non-expert users. Our contributions are summarized as follows:

- **Hybrid SVG Generation Framework**. We introduce a novel text-to-SVG generation paradigm that combines Large Language Models with image diffusion models to produce high-quality SVG outputs.
- **SVG-Oriented Prompt Design**. We develop a specialized prompt system that directs LLMs to generate SVG templates using basic geometric primitives.
- **Dual-Stage Optimization**. We implement a two-phase optimization process that preserves the semantic meaning of each shape while eliminating artifacts such as self-intersecting and jagged paths.
- **Iterative Editing**. We enable iterative refinement of SVG through natural language instructions, making SVG creation more accessible.

2. Related Work

2.1. Large Language Models for Design

LLMs have emerged as powerful tools for graphics design tasks, demonstrating remarkable capabilities in understanding and reasoning about complex design specifications across diverse domains. Recent works have explored their applications in (1) visual layout and composition, including poster designs [5, 21, 34, 47], 3D scene arrangements [2, 10, 14, 32, 33, 38, 51], and placement of basic shapes for image generation [8, 46]; (2) content creation and manipulation for shapes [11], materials [15], and animation [22, 36]; and (3) design understanding [18, 42]. While these demonstrate LLMs' versatility, their potential for vector graphics remains unexplored. Our work addresses this gap by showing that LLMs can effectively generate structured SVG templates, extending their capabilities to the domain of vector graphics creation.

2.2. Text-Guided Vector Graphics Generation

Text-to-SVG generation approaches can be roughly categorized into *image-based* and *language-based* methods.

Image-based methods [9, 16, 43, 44, 50] start with a large collection of randomly initialized shapes, render vector graphics with a differentiable renderer [20], and iteratively optimize path parameters by minimizing text-image similarity losses (*e.g.*, CLIP [28] loss and Score Distillation Sampling [27] loss). Despite their visually pleasant appearance, these methods often generate SVG results containing multiple fragmented paths that lack individual semantic correspondence, which is undesirable for real design scenarios. Language-based methods [3, 35, 41] design specialized tokenization strategies to encode vector graphics into discrete tokens, which are then concatenated with text tokens, and train seq2seq models like autoregressive transformers [37] on domain-specific datasets (*e.g.*, icons and fonts). Although these approaches are conceptually elegant, their generalizability is limited by the scarcity of large-scale, general-purpose vector graphics datasets. Some works [26, 31] have explored LLMs' vector graphics generation capabilities, showing that while LLMs can create reasonable layouts matching text prompts, they struggle to produce complex geometric shapes. Our Chat2SVG combines image-based and language-based methods to address the limitations of each paradigm, producing semantically meaningful SVG where each path corresponds to a distinct visual element and contains refined geometric details.

2.3. Vector Graphics Representation Learning

Vector graphics representation learning is essential for sketch-based retrieval, reconstruction, and generation tasks. A seminal study, SketchRNN [12], combines a Recurrent Neural Network (RNN) and a VAE to learn vector representations. Lopes et al. [24] trained a VAE to represent image-level font styles, then passed the latent embedding into a decoder for vector font generation. To increase reconstruction quality, later approaches like Sketchformer [30] and DeepSVG [4] adopted Transformer [37] architectures, followed by dual-modality methods [23, 39, 40] that leveraged both vector and image features. However, encoding the entire complex SVG into a single latent embedding often causes detail loss. To address this limitation, Zhang et al. [50] proposed path-level representational learning, creating a smooth latent space for individual paths. We leverage these path-level latent embeddings in our optimization to eliminate self-intersecting and jagged paths.

3. Method

Our Chat2SVG, as shown in Figure 2, begins with an SVG-oriented prompting approach (Section 3.1) that guides an LLM to create reasonable SVG templates. We then perform a dual-stage optimization (Section 3.2) to enhance the geometric details of SVG templates by optimizing SVG paths in both latent and control point space, guided by image diffusion models.

3.1. SVG Template Generation with LLMs

Prompt Expansion. Users often provide vague and brief prompts to the LLM when describing their desired graphics. Such unstructured input may lead to oversimplified SVG (regarding both the number of elements and structures) that fails to reflect the intended design. Consequently, we propose a three-level prompt expansion strategy that systematically delineates the initial prompts: (1) Scene-level: Starting with an initial prompt, we instruct the LLM to analyze it holistically and identify essential objects that should appear. The LLM then expands the prompt by suggesting complementary objects to enhance scene completeness. (2) Object-level: For each object in the expanded prompt, we guide the LLM to systematically break it down into its components. For instance, when describing a lion, the LLM deconstructs it into distinct parts such as the body, head, mane, legs, eyes, ears, and tail. This decomposition ensures that no critical components are missing during SVG script generation. (3) Layout-level: After object decomposition, we direct the LLM to develop a comprehensive layout plan. This includes determining the position and size of each element on the canvas, selecting appropriate colors for visual harmony, and establishing clear spatial relationships to ensure a cohesive overall composition.

SVG Script Generation. After prompt expansion, we obtain a detailed scene description. We then convert this natural language specification into SVG scripts in XML format. Since the LLM has limitations in synthesizing geometrically complex paths, we constrain the shapes to a carefully selected set of basic primitives: rectangles, ellipses, lines, polylines, polygons, and short paths. We set the canvas size to 512×512. Each geometric element is assigned a unique ID and includes semantic annotations in its comments.

Visual Rectification. Since prompt expansion is purely text-based, even when the generated SVG script accurately follows the detailed prompt, visual inconsistencies (*e.g.*, misaligned components, disproportionate scaling, and incorrect path ordering) can emerge during rendering. For example, in Figure 2, the initial SVG template shows a lion with a misshapen mane and missing facial features. We thus adopt a visual rectification strategy in which we render the SVG and provide the rendered image back to the LLM (with vision capability) for inspection. The LLM analyzes the visual output, identifies any inconsistencies or oddities, and generates corrected SVG code. This visual refinement loop can be performed iteratively. In our experiments, we find that two iterations of refinement are typically sufficient to generate well-structured SVG templates with appropriate spatial layouts.

Furthermore, to enhance the quality of prompt expansion and SVG script generation, we provide curated in-context examples in the prompts. The complete set of prompts and examples is available in the supplementary material.

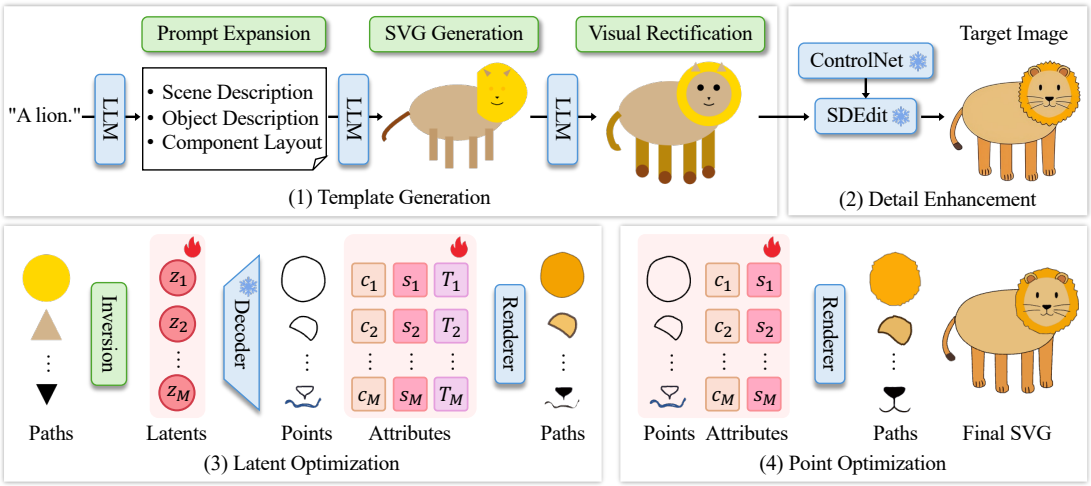

Figure 2. The system pipeline of Chat2SVG. Given a text prompt, our system first leverages an LLM to generate an SVG template composed of basic geometric primitives. The rendered template is enhanced through SDEdit [25] with ControlNet [49] to add visual details while preserving the overall composition, yielding a target image. The SVG then undergoes a dual-stage optimization process to match the target image. (1) Primitives are converted to latent embeddings through latent inversion and optimized along with their visual attributes (*i.e.*, filling colors c_i, stroke properties s_i, and transformation matrices T_i). (2) Point-level optimization is performed to refine the geometric details of SVG paths.

3.2. SVG Optimization Guided by Image Diffusion

Detail Enhancement. While the primitive shapes in the SVG template have accurate semantic meaning, they lack the geometric details necessary for a professional design. To improve visual expressiveness, we first render the SVG template into a raster image, and then use an image editing method to generate a more detailed version that serves as our optimization target. Specifically, we employ SDEdit [25], which enhances the input image by first adding noise and then progressively denoising it with an image diffusion model to produce outputs with richer details, such as the contour of the lion's mane illustrated in Figure 2. To maintain structural similarity between the enhanced output and the original image, we incorporate a ControlNet (tile version) [49] into our pipeline. By using Gaussian-blurred versions of the initial images as control signals, the ControlNet effectively maintains the overall composition throughout the enhancement process.

Apart from geometric details, this image-to-image translation process also introduces new semantic parts such as the lion's beard. To faithfully reproduce the target image, we propose using the Segment Anything Model (SAM) [17] to identify these new decorative details. Specifically, we denote the rendered images of the initial SVG template and its diffusion-enhanced counterpart as I_{template} and I_{target}, respectively. We apply SAM to extract mask sets from both images: $\{m_i\}_{i=1}^{Q}$ for the template image and $\{m_j\}_{j=1}^{K}$ for the target image, where Q and K represent the number of masks in each collection. Our approach is based on the observation that the diffusion process simultaneously generates fine decorative details that are absent in the original template (*e.g.*, the beard of the lion in Figure 2) while preserving the structural integrity of larger components (such as the body). Consequently, we implement an area-based mask-filtering strategy to identify these newly generated elements. For each mask m_j detected in I_{target}, we evaluate it against all masks in $\{m_i\}_{i=1}^{Q}$ using Intersection over Union (IoU) metrics [29]. When m_j overlaps with a template mask m_i but their IoU falls below a threshold, we classify m_j as a new decorative component. Alternatively, if their IoU exceeds the threshold, we interpret m_j as a variation of an existing component. For all filtered masks classified as new decorative shapes, we approximate their boundaries with polygons and integrate them into the SVG script.

We define an SVG script G as a collection of M paths (*i.e.*, shapes), $G = \{P_i\}_{i=1}^{M}$. Each path P_i consists of a sequence of N_i commands, $P_i = \{C_i^j\}_{j=1}^{N_i}$. A command $C_i^j = (U_i^j, V_i^j)$ is defined by its type $U_i^j \in \{\texttt{M}, \texttt{C}\}$ and its associated control points V_i^j, where `M` represents `Move` and `C` represents cubic Bézier curves. For consistency, we convert all other primitive shapes (*e.g.*, rectangles and ellipses) in the SVG template into cubic Bézier curves.

Previous image-based methods [9, 16, 43, 44] optimize SVG shapes by directly manipulating the control points V_i^j. Despite convenience, this approach often leads to artifacts such as self-intersecting curves and unnatural deformations. To overcome these limitations, Zhang et al. [50] introduced a path-level SVG VAE. This model's latent space effec-

tively captures common shape patterns and geometric constraints, allowing for the optimization of path latent vectors at a higher level to produce smooth outputs.

Latent Optimization. We leverage this pretrained SVG VAE to conduct the latent optimization. However, their SVG encoder expects a fixed number of commands (10 cubic Bézier curves), while our SVG primitives contain varying (typically fewer) commands. To resolve this mismatch, we develop a latent inversion process that converts our primitives into latent embeddings. We start with a randomly sampled latent vector z and decode it into a command sequence. Then we evenly sample points $X = \{x_i\}_{i=1}^{N}$ along its contour. Similarly, we sample an equal number of points $Y = \{y_j\}_{j=1}^{N}$ along the contour of our target primitive shape. Using these point sets, we compute the Earth Mover's Distance (EMD) [1, 7, 48]:

$$\ell_{\text{EMD}}(X, Y) = \min_{\phi: X \to Y} \sum_{x \in X} \|x - \phi(x)\|_2 , \tag{1}$$

where ϕ represents a bijective mapping between the point sets. By minimizing this EMD loss, we gradually optimize the latent embedding z to match the target primitive.

After obtaining latent vectors $\{z_i\}_{i=1}^{M}$ for all paths, we jointly optimize these vectors along with visual attribute parameters: filling colors $\{c_i\}_{i=1}^{M}$, stroke properties (color and width) $\{s_i\}_{i=1}^{M}$, and transformation matrices $\{T_i\}_{i=1}^{M}$. At each optimization iteration t, we decode the latent vectors into command sequences and apply transformation matrices to the control points. We then combine color and stroke attributes together, and use a differentiable rasterizer [20] to render the complete SVG into an image I_t. Our optimization objective consists of three loss terms: (1) An MSE loss ℓ_{MSE} between the rendered image I_t and target image I_{target} to ensure visual similarity; (2) A curvature loss that reduces sharp bends and fluctuations along the contour by computing the discrete second derivative:

$$\ell_{\text{curvature}} = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N_i - 2} \left\| V_i^j - 2V_i^{j+1} + V_i^{j+2} \right\|_2^2}{\sum_{i=1}^{M} (N_i - 2)}; \tag{2}$$

(3) A path-level IoU loss that preserves the overall composition and prevents paths from drifting too far from their initial positions:

$$\ell_{\text{IoU}} = \frac{1}{M} \sum_{i=1}^{M} \left(1 - \frac{|m_i^t \cap m_i^0|}{|m_i^t \cup m_i^0|} \right), \tag{3}$$

where m_i^t and m_i^0 denote the binary masks of the i-th path at the current iteration t and its initial state, respectively.

The final loss function combines these three terms with empirically determined weights:

$$\ell_{\text{latent}} = \ell_{\text{MSE}} + \lambda_1 \ell_{\text{curvature}} + \lambda_2 \ell_{\text{IoU}}, \tag{4}$$

where $\lambda_1 = 5e-4$ and $\lambda_2 = 5e-6$.

Point Optimization. While latent optimization effectively aligns SVG shapes with their target positions and contours, it has a notable limitation: the resulting shapes tend to be overly smooth, lacking the intricate details that are often essential in professional vector graphics. For example, in Figure 2, the lion's mane after latent optimization does not capture the fine contour present in the target image. Therefore, we add a second optimization stage that directly refines the control points V_i^j of the paths, similar to previous image-based works [9, 16, 43, 44]. To achieve finer granularity in shape control, we split each cubic Bézier curve at its midpoint, effectively doubling the number of control points in the SVG. This increased control point density allows for more precise shape adjustments. The point optimization stage employs a loss function that combines the MSE loss and the curvature loss:

$$\ell_{\text{point}} = \ell_{\text{MSE}} + \lambda_3 \ell_{\text{curvature}}. \tag{5}$$

The parameter λ_3 decreases linearly from $1e-3$ to $5e-5$ throughout the optimization process.

3.3. Iterative Editing

During SVG script generation (Section 3.1), we instruct the LLM to annotate the semantic label of each path. This design enables users to refine the initial SVG template using natural language, as the LLM can precisely locate and modify specific elements based on semantic understanding.

Our system supports iterative refinement through multiple rounds of natural language instructions. However, when passing the edited SVG template image into SDEdit, image diffusion models may introduce different shape variations and decorative details, which compromise the user's intention to maintain the already optimized shapes. To preserve consistency across editing iterations, we instruct the LLM to output both the modified SVG script and a list of specifically changed paths. This precise tracking enables selective optimization of modified shapes while preserving unaltered elements from previous iterations, maintaining visual coherence throughout the entire refinement process.

4. Experiments

Implementation Details. In our experiments, we use `claude-3-5-sonnet` as our backbone LLM model due to its leading generation capabilities. To evaluate our approach, we create a prompt set by having the LLM generate 125 text prompts across five categories: animals, food, objects, scenes, and novel concept combinations. For each prompt, we generate one template and perform 2 rounds of visual rectification. This process is repeated 5 times, yielding $(1 + 2) \times 5 = 15$ candidate SVG templates per prompt. These templates are rendered as images, and we follow a

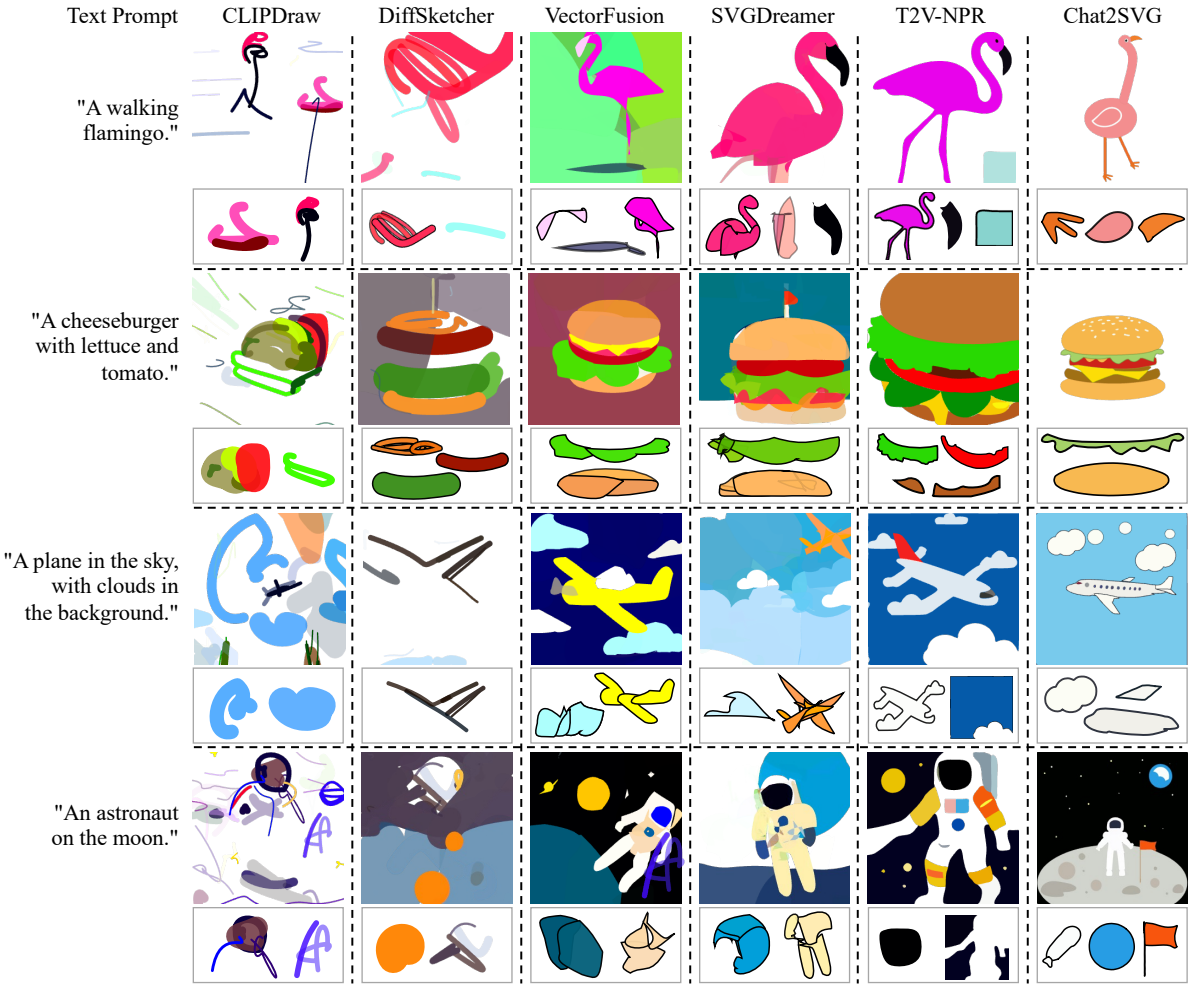

Figure 3. **Qualitative Comparison**. (1) Methods refining open-ended strokes, *i.e.*, CLIPDraw [9] and DiffSketcher [43], often produce distorted and disorganized strokes to approximate objects, presenting messy appearance and poor text alignment. (2) VectorFusion [16] and SVGDreamer [44] produce elements that consist of multiple jagged, irregular, and fragmented shapes, such as the body of the flamingo (first row) and the plane (third row). (3) T2V-NPR [50] attempts to resolve these issues by learning a latent representation of paths and merging fragmented shapes. However, it still cannot guarantee the semantic meanings of the paths, leading to less-semantic paths such as a plane body with surrounding clouds in the third row. In contrast, our method produces SVG with superior text alignment, higher visual quality, and well-structured paths exhibiting geometric regularity and clear semantic definition.

standard practice to select the highest quality SVG using the ImageReward [45] metric. Detailed optimization configurations are provided in the supplementary materials.

Baselines. We compare our approach against five state-of-the-art image-based text-to-SVG generation methods: CLIPDraw [9], DiffSketcher [43], VectorFusion [16], SVGDreamer [44], and T2V-NPR [50]. Current language-based methods [35, 41] are confined to specific categories, with no support for general text-to-SVG generation, so we do not include them in our comparison. For a fair comparison, we set these baseline methods to use a similar number of shapes as our optimized SVG.

Evaluation Metrics. We evaluate the quality of our generated SVG across three dimensions: visual fidelity, path regularity, and semantic alignment.

- *Image-level Fidelity*. We evaluate visual fidelity using a ground-truth dataset of well-designed vector graphics, specifically 52,805 colored SVG files downloaded from SVGRepo[1]. We compute the Fréchet Inception Distance (FID) [13] between the rendered images of our generated SVG and this professional design collection, using features extracted by the CLIP image encoder [28].
- *Path-level Regularity*. We assess path quality using a transformer-based Auto-Encoder trained on the *FIGR-8-SVG* [6] dataset, following the DeepSVG [4] architec-

[1] https://www.svgrepo.com

Method	Image FID ↓	Path FID ↓	Text Alignment ↑
CLIPDraw [9]	46.77	70.13	0.3048
DiffSketcher [43]	44.89	66.48	0.2623
VectorFusion [16]	39.52	56.79	0.2982
SVGDreamer [44]	35.48	47.95	0.2919
T2V-NPR [50]	39.86	42.03	0.3078
Chat2SVG (Ours)	**33.31**	**39.07**	**0.3090**

Table 1. Quantitative comparison of text-to-SVG generation methods across image fidelity, vector regularity, and text alignment.

ture. Our model, trained with reconstruction tasks, encodes each drawing command C_i^j into a latent vector. We represent each path as the mean of its commands' latent embeddings, and calculate the FID between these path representations and the ground truth paths from *FIGR-8-SVG*. This metric quantifies how closely our generated paths align with professional SVG design patterns.

- *Text-level Alignment*. We assess semantic alignment by computing the CLIP score [28] between the text prompt and the rendered SVG image.

4.1. Comparison with Existing Methods

Quantitative Comparison. Table 1 shows the evaluation metrics for all baseline methods. Our method achieves the best image FID score, indicating that our generated SVG closely aligns with professional design patterns. Regarding path regularity, our method yields paths that are closest to the professionally designed SVG, as evidenced by the lowest path FID. Furthermore, our Chat2SVG achieves the highest score in text-SVG alignment, validating the significance of LLM-generated SVG templates.

Qualitative Comparison. In Figure 3, we present a side-by-side comparison between our Chat2SVG and the baselines. Methods that optimize open-ended strokes based on CLIP or Diffusion models (*i.e.*, CLIPDraw [9] and DiffSketcher [43]) produce results with messy visual appearances and poor text alignment. These methods sometimes fail to synthesize complete objects, such as the flamingo (first row) and the plane (third row) in Figure 3. In contrast, our method generates SVG results with clear layouts and greater fidelity to the prompts.

Methods that optimize closed shapes via score distillation of diffusion models (*i.e.*, VectorFusion [16], SVGDreamer [44], and T2V-NPR [50]) can better align with the input text, but still produce fragmented paths with limited semantic meaning. For instance, in the plane example (third row of Figure 3), VectorFusion and SVGDreamer create planes and clouds composed of multiple jagged, irregular, and fragmented shapes, which only appear meaningful when viewed as a whole. This hinders convenient editing by graphic designers. T2V-NPR [50] addresses the issue of jagged paths using a path VAE (although it can be overly smooth at times) and reduces fragmented shapes by merging shapes with similar colors. However, its merging operation ignores the semantic meaning of shapes, resulting in the body of the plane being merged with the surrounding clouds. Additionally, the semantic meaning of the path is sometimes unclear, as exemplified by the cloud in the bottom-right corner, whose contour is actually approximated by the blue background instead of a true cloud shape. In contrast, our Chat2SVG adopts a unique approach by using LLMs to generate paths representing semantic components and performing dual-stage optimization based on image diffusion to enhance path expressiveness, thus ensuring both path regularity and visual fidelity of generated SVG.

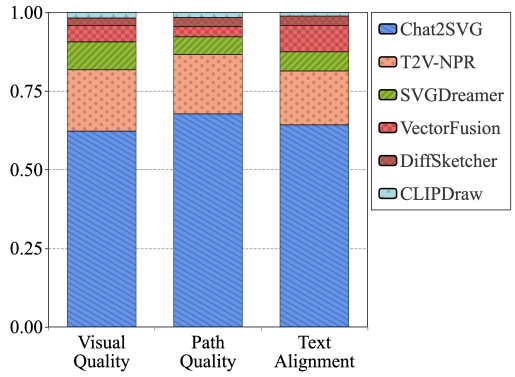

Figure 4. **User Study**. Our Chat2SVG achieves the highest user selection ratio across all three evaluation criteria.

User Study. To further compare Chat2SVG with baseline methods, we conducted a user study examining the same key aspects: (1) overall visual aesthetics, (2) path regularity, and (3) text-SVG alignment. We randomly sampled 20 prompts and generated SVG outputs using our approach and five baseline methods. For each prompt, participants were shown all generated SVG results along with the corresponding text prompt and highlighted paths. They were then asked to select the highest quality result regarding the three evaluation criteria. We recruited 31 participants (18 male, 13 female) through university mailing lists, with a mean age of 26 years. Among them, 17 participants reported prior experience in graphic design, providing a balanced mix of expert and novice perspectives. Analyzing the average selection ratio across all three metrics (Figure 4), we found that Chat2SVG-generated results were consistently preferred by participants over baseline methods.

4.2. Ablation Study

We conduct ablation experiments to evaluate the effectiveness of key components in our pipeline (Section 3.1). First, we remove the SVG template generation and use randomly initialized shapes to approximate target images. As shown in the second row of Table 2, both image FID and path FID

Chat2SVG Variant	Image FID ↓	Path FID ↓	Text Alignment ↑
Default	**33.31**	39.07	**0.3090**
No SVG Template	47.21	70.13	0.2968
No SVG Optimization	33.45	**36.12**	0.3044

Table 2. Qualitative results of the ablation study.

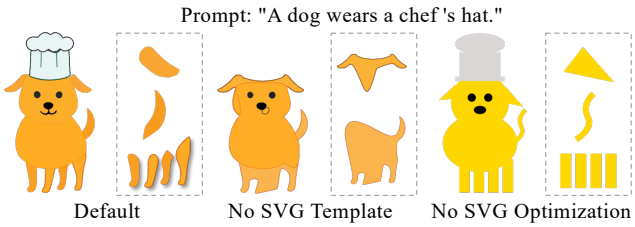

Figure 5. Qualitative results of ablation study.

values increase substantially. This performance degradation can be attributed to the substantial gap between randomly initialized shapes and target shapes, making optimization more challenging. Meanwhile, in the second column of Figure 5, the dog's legs and ears lack semantic-clear components, highlighting the importance of SVG template generation. Second, we eliminate the dual-stage optimization. As shown in the third row of Table 2, this results in the lowest path FID, as primitives generated by LLMs naturally align with the path regularity presented in the ground-truth dataset. However, the visual outcome in Figure 5 lacks the necessary details, indicating that the overall visual quality suffers without dual-stage optimization. A more detailed ablation study is provided in the supplementary material.

4.3. Iterative Editing Results

In this section, we present the iterative editing results. As demonstrated in Figure 6, users can progressively refine the SVG through multiple rounds of natural language instructions. During editing, we preserve the optimized shapes from previous iterations while only modifying the specified shapes. The results showcase how Chat2SVG can accurately interpret and execute user editing requests, maintaining both semantic coherence and visual quality throughout the modifications. Additional editing examples are provided in the supplementary material.

5. Conclusion

In this paper, we present Chat2SVG, a novel paradigm for text-to-SVG generation that combines LLMs and image diffusion models. Through our carefully designed SVG template generation and dual-stage SVG optimization pipeline, our method generates high-quality SVG outputs that exhibit strong visual fidelity, path regularity, and text alignment.

Despite our method's superior results, there are several

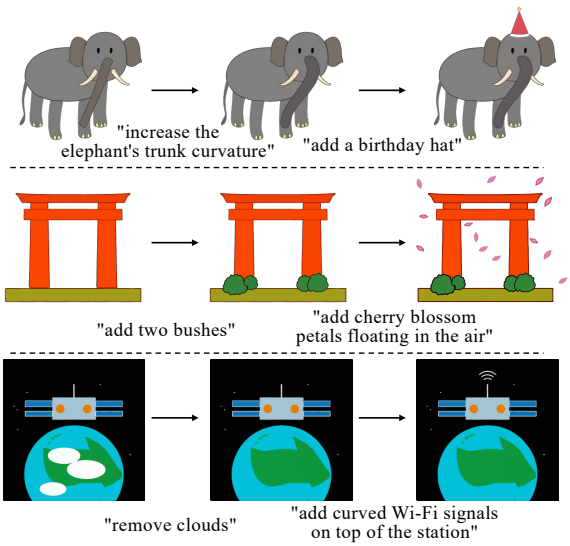

Figure 6. **Iterative Editing**. We perform two rounds of refinement on each SVG template and show the optimized output. This figure shows editing types including deletion, modification, and addition.

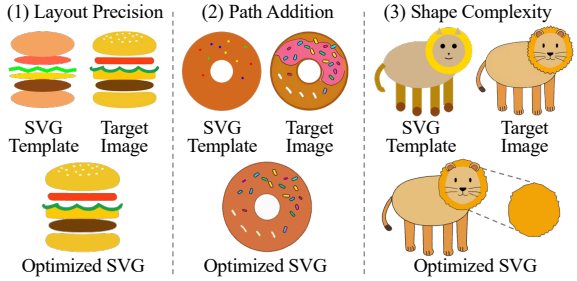

Figure 7. **Limitations** of our method include imprecise layout generation, missing important visual elements, and insufficient shape complexity.

aspects that could be further improved: (1) LLMs may produce imprecise visual layouts, such as the burger in Figure 7. This can be alleviated by collecting an SVG dataset and extracting the corresponding layouts to fine-tune LLMs. (2) Our approach uses SAM to identify new parts. However, SAM may overlook important regions, like the pink frosting on the donut in Figure 7. Incorporating semantic-aware SAM [19] could potentially produce more accurate results. (3) The pretrained SVG VAE from [50] has a fixed number of commands. This makes it hard to represent intricate shapes. For example, the lion's mane in Figure 7 lacks sufficient detail. Training a new SVG VAE with varying-length commands would help solve this problem. Our work can be extended to generate other types of vectorized data, such as icons and fonts. Additionally, through consistent editing of SVG across multiple frames, coupled with shape interpolation, our Chat2SVG has the potential to facilitate keyframe animation generation.

Acknowledgement

The work described in this paper was fully supported by a GRF grant from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region, China [Project No. CityU 11216122].

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *International Conference on Machine Learning*, pages 40–49, 2018. 5

[2] Rio Aguina-Kang, Maxim Gumin, Do Heon Han, Stewart Morris, Seung-Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Qiuhong Anna Wei, Kailiang Fu, and Daniel Ritchie. Open-universe indoor scene generation using LLM program synthesis and uncurated object databases. In *arXiv preprint arXiv:2403.09675*, 2024. 2

[3] Jonas Belouadi, Anne Lauscher, and Steffen Eger. AutomaTikZ: Text-guided synthesis of scientific vector graphics with TikZ. In *arXiv preprint arXiv:2310.00367*, 2023. 3

[4] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. DeepSVG: A hierarchical generative network for vector graphics animation. In *Advances in Neural Information Processing Systems*, pages 16351–16361, 2020. 3, 6

[5] Yutao Cheng, Zhao Zhang, Maoke Yang, Hui Nie, Chunyuan Li, Xinglong Wu, and Jie Shao. Graphic design with large multimodal model. In *arXiv preprint arXiv:2404.14368*, 2024. 2

[6] Louis Clouâtre and Marc Demers. FIGR: Few-shot image generation with reptile. In *arXiv preprint arXiv:1901.02199*, 2019. 6

[7] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 605–613, 2017. 5

[8] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Wang, and William Wang. LayoutGPT: Compositional visual planning and generation with large language models. In *Advances in Neural Information Processing Systems*, pages 18225–18250, 2024. 2

[9] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. CLIPDraw: Exploring text-to-drawing synthesis through language-image encoders. In *Advances in Neural Information Processing Systems*, pages 5207–5218, 2021. 2, 3, 4, 5, 6, 7

[10] Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. AnyHome: Open-vocabulary generation of structured and textured 3D homes. In *European Conference on Computer Vision*, pages 52–70, 2025. 2

[11] Aditya Ganeshan, Ryan Y. Huang, Xianghao Xu, R. Kenny Jones, and Daniel Ritchie. ParSEL: Parameterized shape editing with language. In *arXiv preprint arXiv:2405.20319*, 2024. 2

[12] David Ha and Douglas Eck. A neural representation of sketch drawings. In *arXiv preprint arXiv:1704.03477*, 2017. 3

[13] 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. In *Advances in Neural Information Processing Systems*, pages 6629–6640, 2017. 6

[14] Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. SceneCraft: An LLM agent for synthesizing 3D scenes as Blender code. In *International Conference on Machine Learning*, pages 19252–19282, 2024. 2

[15] Ian Huang, Guandao Yang, and Leonidas Guibas. BlenderAlchemy: Editing 3D graphics with vision-language models. In *arXiv preprint arXiv:2404.17672*, 2024. 2

[16] Ajay Jain, Amber Xie, and Pieter Abbeel. VectorFusion: Text-to-SVG by abstracting pixel-based diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1911–1920, 2022. 2, 3, 4, 5, 6, 7

[17] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *IEEE International Conference on Computer Vision*, pages 4015–4026, 2023. 4

[18] Peter Kulits, Haiwen Feng, Weiyang Liu, Victoria Abrevaya, and Michael J. Black. Re-thinking inverse graphics with large language models. In *arXiv preprint arXiv:2404.15228*, 2024. 2

[19] Feng Li, Hao Zhang, Peize Sun, Xueyan Zou, Shilong Liu, Jianwei Yang, Chunyuan Li, Lei Zhang, and Jianfeng Gao. Semantic-SAM: Segment and recognize anything at any granularity. In *arXiv preprint arXiv:2307.04767*, 2023. 8

[20] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Ragan-Kelley Jonathan. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics*, 39 (6):1–15, 2020. 2, 3, 5

[21] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang Yang, JianGuang Lou, and Dongmei Zhang. LayoutPrompter: Awaken the design ability of large language models. In *Advances in Neural Information Processing Systems*, pages 43852–43879, 2024. 2

[22] Vivian Liu, Rubaiat H. Kazi, Li-Yi Wei, Matthew Fisher, Timothy Langlois, Seth Walker, and Lydia Chilton. LogoMotion: Visually grounded code generation for content-aware animation. In *arXiv preprint arXiv:2405.07065*, 2024. 2

[23] YingTian Liu, Zhifei Zhang, YuanChen Guo, Matthew Fisher, Zhaowen Wang, and SongHai Zhang. DualVector: Unsupervised vector font synthesis with dual-part representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 14193–14202, 2023. 3

[24] Raphael G. Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *IEEE International Conference on Computer Vision*, pages 7930–7939, 2019. 3

[25] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, JunYan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *arXiv preprint arXiv:2108.01073*, 2021. 4

[26] OpenAI. GPT-4 technical report. In *arXiv preprint arXiv:2303.08774*, 2023. 2, 3

[27] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *arXiv preprint arXiv:2209.14988*, 2022. 2, 3

[28] Alec Radford, Jong-Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021. 2, 3, 6, 7

[29] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019. 4

[30] Leo S. F. Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 14153–14162, 2020. 3

[31] Pratyusha Sharma, Tamar Rott Shaham, Manel Baradad, Stephanie Fu, Adrian Rodriguez-Munoz, Shivam Duggal, Phillip Isola, and Antonio Torralba. A vision check-up for language models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 14410–14419, 2024. 2, 3

[32] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3D-GPT: Procedural 3D modeling with large language models. In *arXiv preprint arXiv:2310.12945*, 2023. 2

[33] Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. SceneMotifCoder: Example-driven visual program learning for generating 3D object arrangements. In *arXiv preprint arXiv:2408.02211*, 2024. 2

[34] Zecheng Tang, Chenfei Wu, Juntao Li, and Nan Duan. LayoutNUWA: Revealing the hidden layout expertise of large language models. In *arXiv preprint arXiv:2309.09506*, 2023. 2

[35] Zecheng Tang, Chenfei Wu, Zekai Zhang, Mingheng Ni, Shengming Yin, Yu Liu, Zhengyuan Yang, Lijuan Wang, Zicheng Liu, Juntao Li, and Nan Duan. StrokeNUWA: Tokenizing strokes for vector graphic synthesis. In *arXiv preprint arXiv:2401.17093*, 2024. 2, 3, 6

[36] Tiffany Tseng, Ruijia Cheng, and Jeffrey Nichols. Keyframer: Empowering animation design using large language models. In *arXiv preprint arXiv:2402.06071*, 2024. 2

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017. 3

[38] Can Wang, Hongliang Zhong, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Chat2Layout: Interactive 3D furniture layout with a multimodal LLM. In *arXiv preprint arXiv:2407.21333*, 2024. 2

[39] Yizhi Wang and Zhouhui Lian. DeepVecFont: Synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics*, 40(6):1–15, 2021. 3

[40] Yuqing Wang, Yizhi Wang, Longhui Yu, Yuesheng Zhu, and Zhouhui Lian. DeepVecFont-v2: Exploiting Transformers to synthesize vector fonts with higher quality. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 18320–18328, 2023. 3

[41] Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. IconShop: Text-guided vector icon synthesis with autoregressive Transformers. *ACM Transactions on Graphics*, 42(6):1–14, 2023. 2, 3, 6

[42] Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. GPT-4V (ision) is a human-aligned evaluator for text-to-3D generation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 22227–22238, 2024. 2

[43] Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. DiffSketcher: Text guided vector sketch synthesis through latent diffusion models. In *Advances in Neural Information Processing Systems*, pages 15869–15889, 2023. 2, 3, 4, 5, 6, 7

[44] Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. SVGDreamer: Text guided SVG generation with diffusion model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4546–4555, 2024. 2, 3, 4, 5, 6, 7

[45] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems*, pages 15903–15935, 2024. 6

[46] Yutaro Yamada, Khyathi Chandu, Yuchen Lin, Jack Hessel, Ilker Yildirim, and Yejin Choi. L3GO: Language agents with chain-of-3D-thoughts for generating unconventional objects. In *arXiv preprint arXiv:2402.09052*, 2024. 2

[47] Tao Yang, Yingmin Luo, Zhongang Qi, Yang Wu, Ying Shan, and Chang Wen Chen. PosterLLaVa: Constructing a unified multi-modal layout generator with LLM. In *arXiv preprint arXiv:2406.02884*, 2024. 2

[48] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2018. 5

[49] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *IEEE International Conference on Computer Vision*, pages 3836–3847, 2023. 4

[50] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Text-to-vector generation with neural path representation. *ACM Transactions on Graphics*, 43(4):1–13, 2024. 2, 3, 4, 6, 7, 8

[51] Qihang Zhang, Chaoyang Wang, Aliaksandr Siarohin, Peiye Zhuang, Yinghao Xu, Ceyuan Yang, Dahua Lin, Bolei Zhou,

Sergey Tulyakov, and Hsin-Ying Lee. SceneWiz3D: Towards text-guided 3D scene composition. In *arXiv preprint arXiv:2312.08885*, 2023. 2