

Beyond Pixels: Exploring Human-Readable SVG Generation for Simple Images with Vision Language Models

Tong Zhang¹ Haoyang Liu² Peiyan Zhang³ Yuxuan Cheng⁴ Haohan Wang²

¹Fudan University ²University of Illinois Urbana-Champaign

³The Hong Kong University of Science and Technology

⁴The Chinese University of Hong Kong, Shenzhen

tongzhang25@m.fudan.edu.cn hl57@illinois.edu zhangao@cse.ust.hk

hanxuanwxss@gmail.com haohanw@illinois.edu

Abstract

In the field of computer graphics, the use of vector graphics, particularly Scalable Vector Graphics (SVG), represents a notable development from traditional pixel-based imagery. SVGs, with their XML-based format, are distinct in their ability to directly and explicitly represent visual elements such as shape, color, and path. This direct representation facilitates a more accurate and logical depiction of graphical elements, enhancing reasoning and interpretability. Recognizing the potential of SVGs, the machine learning community has introduced multiple methods for image vectorization. However, transforming images into SVG format while retaining the relational properties and context of the original scene remains a key challenge. Most vectorization methods often yield SVGs that are overly complex and not easily interpretable. In response to this challenge, we introduce our method, Simple-SVG-Generation (S^2VG^2). Our method focuses on producing SVGs that are both accurate and simple, aligning with human readability and understanding. With simple images, we evaluate our method with reasoning tasks together with advanced language models, the results show a clear improvement over previous SVG generation methods. Furthermore, we construct the novel CLEVR-SVG benchmark and demonstrate that state-of-the-art LLMs can perform rigorous spatial reasoning zero-shot directly on our generated semantic SVG code with over 91.1% accuracy. We also conducted surveys for human evaluation on the readability of our generated SVGs, the results also favor our methods.

Project page: <https://beyondpixels.stonezhang.com/>. Code, generated datasets, and inference evaluation logs are available at <https://github.com/tongmd/S2VG2-CLEVR>.

1. INTRODUCTION

Rival image representation techniques coexist within computer graphics: bitmap images, which consist of pixel matrices, and vector images, depict sequences of artistically drawn shapes. Image Rasterization is particularly well understood and implemented, yet modern-day trends show a rising interest towards image vectorization for its inherent advantages, such as scale adaptability and resolution independence, which are vital for usage in contemporary interfaces [6]. However, vectorization poses numerous challenges, particularly in generating vector images or Scalable Vector Graphics (SVGs) [29] that are both human-readable and retain the semantic relevance of the original raster image.

Contrasting raster images, which are an assembly of ordered pixels, SVGs describe images through a set of parametric shape primitives which enables numerous benefits including smaller file sizes and resolution-independence [22]. Despite being inherently more compact and flexible, ensuring that the SVG representations retain the semantics of the original image and at the same time are efficiently generable remains a difficult task. Figure 1 provides a visual representation of the fundamental components of an SVG. It illustrates how simple geometric shapes are coded and rendered into a composite image.

Recent years have seen considerable progress in image vectorization, primarily through two technical advancements: implementing advanced generative models [1, 15] and deploying sophisticated differentiable rendering methods [12, 16]. However, these developments overlooked key characteristics of SVGs, such as their XML foundation and their complexity beyond simple paths. Specifically, these methods typically yield SVGs that are unreadable, as they overly rely on certain strategies like the DiffVG, which generates SVGs with a fixed number of paths where many are unnecessarily obscured by others [12]. Moreover,

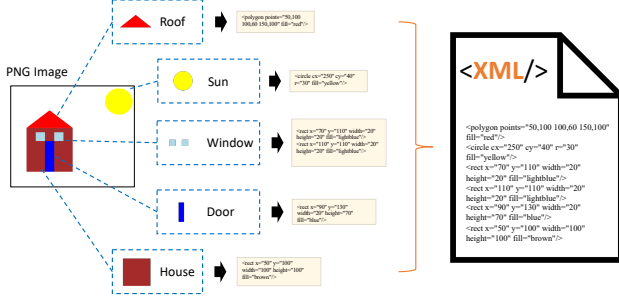


Figure 1. Demonstrates an SVG representation of a simplified house, translating basic geometrical shapes into their corresponding SVG code elements and then into a visual structure, highlighting the vectorization process.

these images frequently contain curves that overshoot the viewBox attribute’s boundaries, resulting in a vectorized image that fails to capture the intended semantic information. These methods also spend quite a long time to generate SVGs for image preprocessing and optimization process. Therefore, a simple but practical method for capturing the essence of image vectorization is sought after. More crucially, as Large Language Models (LLMs) increasingly serve as agents for graphic design and visual reasoning, there is an urgent need for SVGs to be *human-readable* and *semantically structured*. While traditional methods (e.g., VTracer, Im2Vec [23]) or deep-learning approaches (e.g., DeepSVG) excel at tracing complex curves for visual fidelity, they produce “path soups”—thousands of meaningless Bezier curves that are entirely opaque to LLMs. Our goal is to bridge this gap by generating SVGs composed of fundamental, semantic primitives that LLMs can naturally read, edit, and reason over.

In this paper, we propose a novel approach, S^2VG^2 , which leverages the capabilities of vision language models to overcome these challenges. S^2VG^2 shows excellent performance in generating clean, understandable SVG representations from complex raster images, while maintaining their meaning from a comprehensible, human perspective. Efficacy of S^2VG^2 is evaluated rigorously by benchmarking it against three key metrics: pixel and feature level similarity to the original bitmap; simplicity and readability of the resulting image in terms of the number of shapes and the parameters defining them; and the speed at which the vectorized image is generated.

To bridge the gap between pixel-level rendering and explicit symbolic representation, we formalize the problem of *semantic vectorization*. Unlike existing approaches that aggressively optimize bezier curves for pixel-level fidelity at the cost of structural bloat, S^2VG^2 employs a novel two-stage pipeline. In the first stage, a Vision-Language Model acts as a discrete topological planner, autoregressively de-

coding a sequence of interpretable shape primitives (e.g., `<rect>`, `<circle>`). In the second stage, an inverse rendering engine refines the continuous parameters (e.g., colors, coordinates) of these fixed semantic primitives via gradient descent. This explicitly decouples the generation of readable scene graphs from the continuous optimization of visual alignment.

Moreover, it poses an innovative perspective on vectorization process by incorporating vision language models. S^2VG^2 unravels a new realm of possibilities for pretrained vision large language models in visual representation and understanding.

Pioneering SVG-based Visual Reasoning. Looking beyond mere vectorization, our method acts as a foundational pioneer for the emerging paradigm of structured visual reasoning. Recently, state-of-the-art multimodal large language models (such as the Gemini 1.5/2.0 series [25] and GPT-5 class models) are increasingly leveraging explicit SVG representations to convert raw raster images into semantic scene graphs. By parsing an image into human-readable SVG codes, these frontier models can perform exact, symbolic-level spatial reasoning, editing, and grounding. Our work directly anticipates and demonstrates this pipeline—transforming images into a discrete semantic representation (SVG) specifically to enable and evaluate downstream logical reasoning.

Our main contributions can be summarized as follows:

- **Neuro-symbolic Vectorization Framework:** We propose S^2VG^2 , a two-stage approach coupling discrete Vision-Language Model autoregressive decoding with continuous differentiable rendering. This explicitly ensures that the generated SVGs are both highly faithful to original pixels and inherently interpretable as semantic scene graphs.
- **Zero-Shot Visual Reasoning Engine:** We demonstrate that generating explicit, parameter-driven SVGs natively empowers foundation LLMs to perform rigorous spatial mathematics and visual reasoning. Our method produces drastically smaller SVG sizes (< 400 bytes on average) and dominates established baselines across LPIPS, SSIM, and logical readability metrics.
- **New Evaluation Benchmark (CLEVR-SVG):** We introduce CLEVR-SVG, a procedural evaluation framework designed to test deep multi-hop logic on generated code. Using pure-text representations produced by S^2VG^2 , state-of-the-art LLMs (e.g., Gemini 3.1 Pro Preview) achieve 91.1% zero-shot accuracy, empirically validating the impact of explicit scene graphs in bridging vision and language.

2. RELATED WORK

2.1. Scalable Vector Graphics

Vector graphics provide an alternative representation framework for visual content, delineating images as collections of parameterized shape primitives, such as polygons, circles, and rectangles [21]. This contrasts with conventional raster-based images composed of pixel grids. Notably, vector graphics leverage these primitives’ geometric attributes, represented by coordinate sets defining contours and associated color values. This representation is widely supported by web browsers, requiring no specialized software or plugins for rendering. The succinctness and infinite scalability of vector graphics result in adaptability via easy adjustments in stroke or color parameters.

The Scalable Vector Graphics format (SVG) takes advantage of this intrinsic characteristic of vector graphics by encoding images as XML-based text files, specifying geometric entities and their pertinent attributes. This XML structure empowers facile manipulation and editing, rendering SVG particularly versatile for web applications and graphic design tasks. In the context of this paper, we harness the potential of vision language models (VLMs) [30] to bridge the gap between pixel-based images and the SVG format. This strategic integration of VLMs ensures the fidelity of generated SVGs in terms of readability, simplicity, and preservation of semantic coherence.

2.2. Image Vectorization

Rasterization and vectorization are dual problems in image processing. Various techniques, including algorithmic and machine learning-based methods [26], have been explored. Algorithmic approaches encompass mesh-based [8, 13, 33] and curve-based methods [3, 32], each with their own advantages and limitations. Machine learning-based methods, however, offer promising avenues for automated vector image generation [6].

Existing machine learning-compatible vectorization methods, such as Mang2Vec [24] and DVoTD [7], exhibit limitations. Mang2Vec struggles with color images and often generates SVGs with an excessive number of shapes, impacting both efficiency and semantic clarity. Other methods, like DiffVG [12] and LIVE [16], leverage iterative processes and optimization techniques. While pioneering, they often require manual tuning of path counts and produce visually complex but semantically meaningless paths. Similarly, auto-tracing tools like VTracer or generative models like DeepSVG and Im2Vec [23] focus heavily on topological interpolation or font/icon synthesis. However, these methods are fundamentally orthogonal to our objective: they optimize for pixel-perfect curves rather than extracting a semantic, primitive-based, and human-readable scene graph necessary for neuro-symbolic reasoning.

2.3. Image Captioning

Image captioning [27] is the task of generating a coherent natural language description for an image, representing a vibrant and actively researched intersection of computer vision and natural language processing in recent years. The central objective of image captioning lies in the development of models capable of learning the intricate mapping between visual and textual domains, with the aim of producing meaningful and precise image descriptions.

In contemporary approaches to image captioning, notable efforts have been channeled towards elevating the quality of generated captions. These endeavors encompass the integration of attention mechanisms, reinforcement learning techniques, and the utilization of pretrained vision language models. Attention mechanisms [18] empower models to selectively focus on specific regions within an image during the caption generation process. Reinforcement learning [17], on the other hand, equips models with the capacity to refine their captioning skills by learning from feedback received during training.

Recent advancements have introduced large language models (LLMs) like GPT-4 [19], which have showcased remarkable multi-modal capabilities. A prominent research focus has been the alignment of image encoders with these pretrained vision language models [9, 14, 34], representing a compelling category within the broader domain of pretrained vision language models (VLMs). These pretrained VLMs have the ability to encapsulate extensive vision-language correspondence knowledge, facilitating zero-shot predictions through the matching of embeddings derived from both images and texts.

3. Methodology

We introduce **S²VG²** (Simple Semantic Vector Graphics Generation), a novel framework that synergizes Vision-Language Models (VLMs) and differentiable rendering to generate human-readable, semantically aligned Scalable Vector Graphics (SVG). Section 3.1 outlines the overall architecture, while Section 3.2 details the fine-tuning strategy and inference-time continuous optimization.

3.1. Architecture

Our architecture bridges visual perception and language semantics to natively produce SVG tokens. While **S²VG²** is model-agnostic, our primary instantiation adopts BLIP [11] for its proven efficiency. The visual encoder, instantiated as a Vision Transformer (ViT) [5], extracts robust global and local context. The language decoder, initialized from a pretrained BERT [4] with causal attention masking, integrates these visual embeddings via cross-attention. This setup enables the model to auto-regressively predict standard SVG elements (e.g., `<circle>`, `<rect>`) rather than generat-

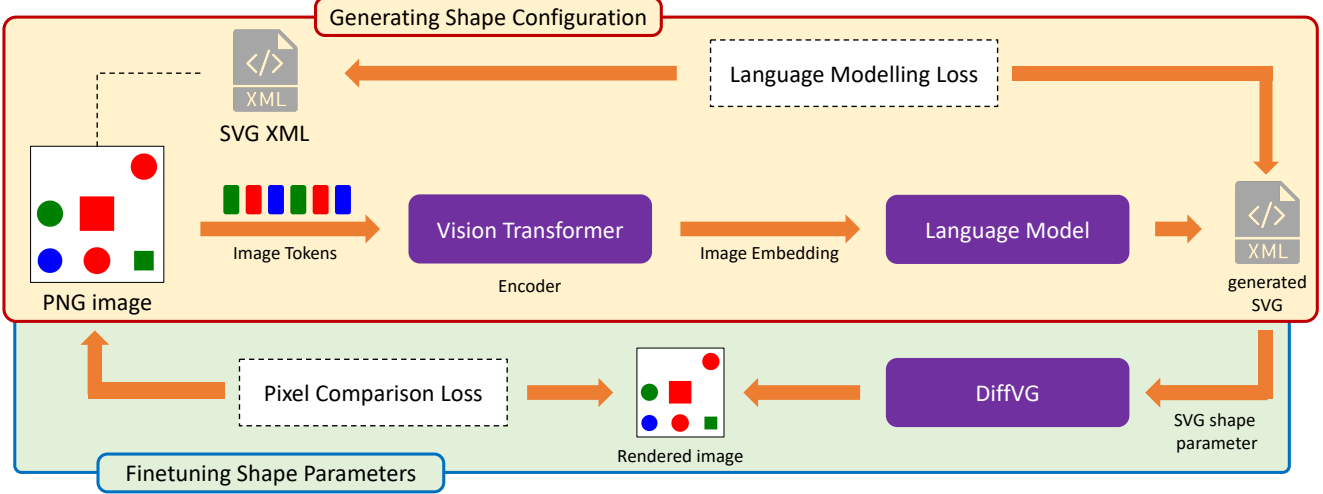


Figure 2. **Overview of S^2VG^2 .** Our framework extracts global and patch-level image features using a Vision Transformer (ViT), which are then cross-attended by a causal text decoder to auto-regressively generate readable SVG tokens. During inference, differentiable rendering (DiffVG) is applied to fine-tune shape parameters.

ing opaque bezier path soups.

3.2. Training and Inference Formulation

Training Stage: Visual-Semantic Alignment We formalize SVG generation as an image-conditioned sequence generation task. Given a dataset of raster images and their corresponding SVG codes $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $y_i = (y_i^{(1)}, y_i^{(2)}, \dots, y_i^{(l_i)})$, the objective is to maximize the conditional log-likelihood of the SVG tokens.

The ViT processes $\mathbf{x}_i$ into M patch embeddings $\mathbf{V}_i = \{\mathbf{v}_i^{(1)}, \dots, \mathbf{v}_i^{(M)}\}$. The text decoder then cross-attends to $\mathbf{V}_i$ to predict the next token:

$$\hat{P}(y_i^{(t)} | y_i^{(<t)}, \mathbf{x}_i) = f_\theta(y_i^{(t)} | \mathbf{V}_i, y_i^{(1)}, \dots, y_i^{(t-1)}) \quad (1)$$

where f_θ represents the conditional generative model parameterized by θ . We optimize the model using standard cross-entropy loss:

$$\mathcal{L}_{\text{NLL}}(\theta) = - \sum_{i=1}^N \sum_{t=1}^{l_i} \log \hat{P}(y_i^{(t)} | y_i^{(<t)}, \mathbf{x}_i) \quad (2)$$

Inference Stage: Differentiable Shape Refinement

During inference, the auto-regressive decoder outputs an initial SVG prediction $\hat{y}_i$. While VLMs excel at capturing discrete semantic structures (e.g., shape types, relative ordering), they often struggle with continuous quantitative reasoning (e.g., exact pixel coordinates or precise scaling).

To bridge this gap, we apply a continuous optimization step. We parse the predicted SVG $\hat{y}_i$ into a set of parameterized primitives $S_i = \{(s_i^{(j)}, \phi_i^{(j)})\}_{j=1}^{r_i}$, where $s_i^{(j)}$ is the

categorical shape type and $\phi_i^{(j)}$ represents its continuous attributes (position, scale, rotation, color).

Using DiffVG [12], a differentiable SVG renderer, we map S_i back to the pixel space:

$$\tilde{\mathbf{x}}_i = \text{DiffVG}(S_i) \quad (3)$$

We freeze the semantic shape types s_i and optimize the continuous parameters ϕ_i via gradient descent to minimize the photometric error between the rendered image $\tilde{\mathbf{x}}_i$ and the input $\mathbf{x}_i$:

$$\mathcal{L}_{\text{refine}}(\phi_i) = \alpha \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|_1 + \beta \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|_2^2 \quad (4)$$

where α and β are weighting hyperparameters. This hybrid approach guarantees that the final SVG is both semantically interpretable and visually accurate.

4. Experiments

4.1. Datasets

In this section, we provide a detailed description of the experimental datasets utilized to evaluate the performance of our S^2VG^2 method. We introduce two distinct datasets: SVG-SHAPE and SVG-Transform.

4.1.1 SVG-SHAPE Dataset

To thoroughly assess the capabilities of our S^2VG^2 method, we have constructed a novel dataset, named SVG-SHAPE. This dataset, depicted in Figure 3a, is designed to challenge and evaluate the performance of SVG generation algorithms in visual question answering contexts. Details of the dataset construction are provided in Algorithm 1.



Figure 3. Images samples from the SVG-SHAPE and SVG-Transform datasets.

Algorithm 1: Construction of the SVG-SHAPE Dataset

Input: Dataset Size N , Number of Grids G
Output: Dataset D of image-SVG pairs

- 1 Initialize an empty dataset $D \leftarrow \emptyset$;
- 2 **for** $i \leftarrow 1$ **to** N **do**
- 3 Initialize an empty SVG document svg ;
- 4 **for** $j \leftarrow 1$ **to** G **do**
- 5 $s \leftarrow \text{random_shape}()$; // circle, rectangle, or triangle
- 6 $c \leftarrow \text{random_color}()$; // red, green, or blue
- 7 $z \leftarrow \text{random_size}()$;
- 8 Create SVG element E using (s, c, z) ;
- 9 Append E to svg ;
- 10 Add svg to D ;
- 11 Rasterize all SVGs in D into RGB images;

The SVG-SHAPE dataset is specifically crafted for visual question answering tasks. It features images where diverse shapes, varying in color and size, are strategically placed on a 3x3 grid. Each image is associated with two types of tasks: one that requires inferring the presence of specific shapes, for example, "Is there a green circle?", and another that involves determining relative positions, such as "Is there a red triangle positioned above a blue shape?"

4.1.2 SVG-Transform Dataset

To enhance the capabilities in SVG generation and manipulation, we present the SVG-Transform dataset. This dataset, showcased in Figure 3b, is designed to challenge and assess SVG generation techniques in a more intricate scenario: performing geometric transformations on shapes. Details of the dataset construction are provided in Algorithm 2.

4.2. Training Settings

In our experimental setup, we conduct a comparative analysis between our innovative method and established baseline techniques in the realm of SVG generation. Training involves two distinct datasets: the SVG-SHAPE dataset, which includes 50,000 image-SVG pairs, and the more comprehensive SVG-Transform dataset, encompass-

Algorithm 2: Construction of the SVG-Transform Dataset

Input: Dataset Size N
Output: Dataset D of image-SVG pairs

- 1 Initialize an empty dataset $D \leftarrow \emptyset$;
- 2 **for** $i \leftarrow 1$ **to** N **do**
- 3 Initialize an empty SVG document svg ;
- 4 $s \leftarrow \text{random_shape}()$;
- 5 $z \leftarrow \text{random_size}()$;
- 6 $T \leftarrow \text{random_transform}()$; // translate, scale, skewX, skewY, rotate
- 7 Create SVG element E using (s, z) with transform T ;
- 8 Append E to svg ;
- 9 Add svg to D ;
- 10 Rasterize all SVGs in D into RGB images;

ing 500,000 pairs. These datasets provide a solid foundation for evaluating the efficacy and adaptability of S²VG².

The architectural backbone of our model integrates the Vision Transformer (ViT) with BERT parameters adapted from the BLIP model. This model is pre-trained on a diverse array of datasets, including the Visual Genome (VG) [10], the SBU Captions Dataset [20], and the extensive Conceptual Captions 12M dataset [2]. This pre-training equips our model with a nuanced understanding of complex visual-textual relationships.

For the optimization process, we utilize the Adam optimizer, setting the learning rate to 1e-5 and the weight decay coefficient at 0.05. Training is conducted on a powerful computational setup comprising four NVIDIA A40 GPUs, and we maintain a batch size of 16.

4.3. Evaluation

4.3.1 Image Quality

We evaluate the vectorization quality of S²VG² through both quantitative and qualitative analysis, focusing on the differences between the input targets and the SVG rendered images.

The baseline for comparison is established using the original input images (Figure 4a). SVGs generated by S²VG² (Figure 4b) exhibit high fidelity by directly predicting the semantic geometry. In comparison, images vectorized by the LIVE method (Figure 4c) display noticeable distortions. While increasing the path count or iteration budget for LIVE could improve its visual fidelity, doing so would further bloat the SVG file with thousands of redundant bezier curves, completely destroying its readability for LLMs. Thus, we configured LIVE (10 paths, 50 iterations) to match a similar token-complexity budget as our method to ensure a fair comparison on *semantic* vectorization. in

shape geometry and color hues. The vectorization results from GPT-4V (Figure 4d) show the most significant deviations in shape accuracy and color representation. It can only reliably identify some relative positional relationships between objects within the image. Additionally, we vectorize images using DiffVG, setting the path number to 50 and iterating 500 times, a configuration chosen to balance the complexity of SVG tokens against each method’s ability to render the images accurately. The readable SVG code produced by S^2VG^2 for both the SVG-Transform and SVG-SHAPE datasets can be seen in Figure 5, illustrating the method’s effectiveness in generating interpretable and semantically coherent SVGs.

This visual comparison highlights the superior performance of S^2VG^2 in preserving the integrity of the original designs during vectorization. It is important to note that the specific settings for LIVE and DiffVG, particularly in terms of path number and iteration count, play a crucial role in this evaluation.



Figure 4. Images from different image vectorization methods.

To more precisely quantify the quality of the vectorized images, we employ several image quality metrics for comparison: Learned Perceptual Image Patch Similarity (LPIPS)[31], Structural Similarity Index (SSIM)[28], as well as L1 and L2 norms. These metrics offer a comprehensive evaluation, shedding light on the perceptual and structural similarities, along with the error magnitudes, between the generated images and their original counterparts.

Method	LPIPS↓	SSIM↑	L1↓	L2↓
S^2VG^2	0.00395	0.99501	0.92041	0.63227
DiffVG	0.06075	0.97949	8.66592	3.39910
LIVE	0.01835	0.99418	15.30974	0.79269

Table 1. Comparison of Image Quality Metrics for Different Methods

The results in Table 1 reveal that S^2VG^2 surpasses other methods in all these metrics. Notably, it records the lowest score in LPIPS, suggesting a higher perceptual resemblance to the original images. In terms of structural integrity, it achieves the highest SSIM score. Furthermore, its lower values in both L1 and L2 norms indicate a minimal pixel-level error throughout the vectorization process.

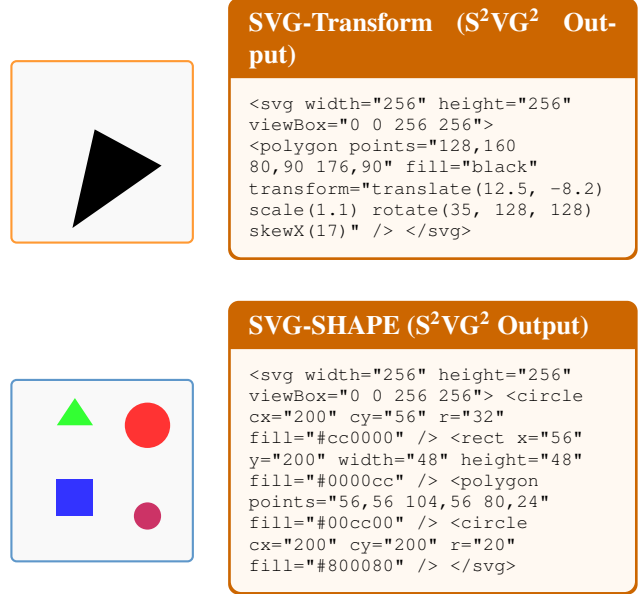


Figure 5. The top image showcases a shape from the SVG-Transform dataset, with its SVG code generated by S^2VG^2 , highlighting the human-readable format and precise transformation parameters. The bottom image, from the SVG-SHAPE dataset, presents an array of geometric shapes in vibrant colors. The accompanying SVG code, also produced by S^2VG^2 , clearly outlines the shapes and their attributes, emphasizing the method’s effectiveness in generating interpretable and semantically coherent SVGs.

4.4. Readability Evaluation

4.4.1 Visual Question Answering

Visual Question Answering (VQA) is a task that combines image understanding and language generation to answer questions about images. In the context of our work, VQA is employed to assess the readability of the generated SVG. As shown in Figure 4, the process starts with the conversion of raster images into SVG format using S^2VG^2 . These SVGs, coupled with relevant questions, are then fed into large language models (LLMs) for VQA. The LLM’s performance in answering these questions provides insight into the quality of the SVGs in terms of their readability and fidelity to the original images.

In our evaluation framework, we leverage GPT-4-32k, a variant of the GPT-4 model designed to handle inputs with extended token lengths up to 32,000 tokens. This adaptation is crucial as it accommodates the lengthy and complex SVGs typically generated by existing vectorization methods, which often exceed the standard token limits of conventional language models.

As shown in Table 2, our method demonstrates superior readability performance. Its accuracy (acc) closely matches

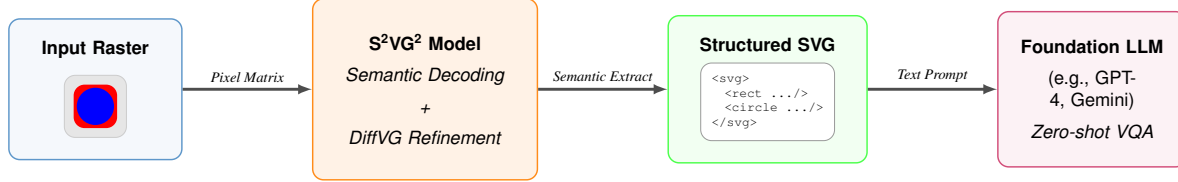


Figure 6. **Evaluation Pipeline for Semantic Vectorization.** To systematically assess the human and machine readability of the generated SVGs, we bypass the vision encoder of foundation LLMs. Instead, the raw SVG text generated by S²VG² is directly fed as a pure text prompt into the LLM to perform Visual Question Answering (VQA) tasks, measuring the fidelity and semantic logic of the vectorization.

Method	acc	acc1	acc2
GT	87.725	99.25	76.2
S ² VG ²	87.75	99.05	76.45
LIVE	50.075	50.1	50.05
DiffVG	50.0	50.0	50.0

Table 2. Accuracy metrics for two VQA tasks. Acc1 represents the accuracy for identifying the existence of specific shapes within the SVG, while Acc2 relates to the accuracy of determining the relative position of shapes.

the Ground Truth (GT), the optimal benchmark in this context. Particularly in identifying the existence of specific shapes (acc1), our method reaches near-perfect accuracy.

In tasks involving the assessment of relative shape positioning (acc2), our method significantly outperforms the LIVE and DiffVG methods. This highlights S²VG²’s effectiveness in preserving spatial relationships and contextual integrity of the original images during conversion.

In summary, the SVGs produced by S²VG² exhibit high readability in VQA tasks, attesting to the method’s ability to generate SVGs that are visually precise and semantically detailed.

4.4.2 Zero-Shot Visual Reasoning on CLEVR-SVG

To explicitly validate the efficacy of S²VG² as a structured semantic scene graph for foundation language models, we constructed **CLEVR-SVG**, a new visually dense reasoning benchmark. CLEVR-SVG features hundreds of procedurally generated canvases with varying geometries, scales, and colors. Crucially, each sample requires deep multi-hop logic, covering spatial relations (e.g., “furthest object to the left”), conditional counting, and attribute queries (e.g., color-binding to specific shapes).

We provide the raw SVG code generated by our method directly to **Gemini 3.1 Pro Preview** without any raster pixel inputs. Evaluated on 101 generated test scenes, the LLM achieved an astounding **91.1% overall zero-shot accuracy** entirely from reading the S²VG² semantic primitives. This experiment categorically confirms that our explicit representation natively bypasses standard visual hallucinations

and seamlessly empowers pure-text LLMs to execute rigorous spatial mathematics. The complete dataset and reproducible evaluation scripts are open-sourced at <https://github.com/tongmd/S2VG2-CLEVR>.

4.4.3 User Study

We conduct a user study to assess the effectiveness of our SVG generation method, S²VG², in comparison to other methods like LIVE and DiffVG. The study involves a total of 19 questions, divided into two parts.

The first part includes 9 questions focused on evaluating the participants’ ability to infer the existence and relative positions of shapes within the SVGs. The objective is to determine how accurately users interpret and understand the spatial arrangement and presence of shapes in the SVGs. The second part comprises 10 questions where participants are asked to select the SVG output from each method that they find easiest to understand.

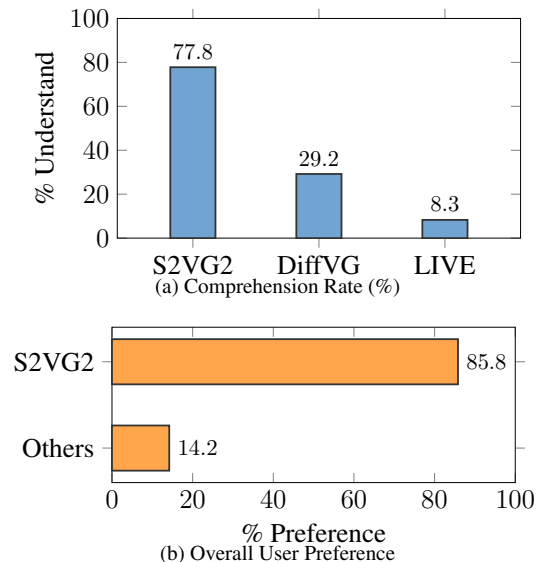


Figure 7. **User Study Results.** (a) S²VG² achieves a significantly higher comprehension rate compared to path-based methods. (b) 85.8% of participants strongly preferred the semantic SVG output of S²VG².

The user study (Figure 7) clearly demonstrates our method’s superiority in SVG readability. 77.8% of participants could easily interpret SVGs generated by S²VG² (Figure 7a). In contrast, the majority of users struggled with the LIVE method (61.1% could not understand, 30.6% answered incorrectly) and DiffVG (70.8% indicated a lack of understanding). Crucially, 85.8% of participants explicitly favored S²VG² as the most comprehensible SVG output (Figure 7b). These results confirm that our primitive-based vectorization is significantly more aligned with human cognitive processing than complex path-based alternatives.

The results from the user study conclusively show that S²VG² significantly enhances the readability of SVGs. They are more comprehensible compared to the alternatives, likely due to the reduced complexity and smaller file sizes of the SVGs generated by our method.

4.4.4 Complexities Reflected in SVG File Characteristics

To further our understanding of the practical implications of SVG generation methods, we delve into the complexities as reflected in the file characteristics. We consider the file size and tokens number as a proxy for the complexity of the SVGs, which also has a direct impact on the interpretability and computational efficiency of these vector images.

Method	Mean Size (bytes)	Std Dev (bytes)
S ² VG ²	390.507	112.957
LIVE	6776.223	148.000
DiffVG	28354.357	196.729

Table 3. Comparison of Mean File Sizes and Standard Deviations

Method	Mean Length (tokens)	Std Dev (tokens)
S ² VG ²	186.72	50.394
LIVE	3624.951	86.126
DiffVG	15736.102	131.635

Table 4. Comparison of Mean Token Lengths and Standard Deviations

As depicted in Tables 3 and 4, S²VG² demonstrates a clear advantage in generating more concise and less complex SVG files. This is evidenced by the significantly smaller mean file sizes and shorter token lengths. Such streamlined files suggest a higher degree of readability and simplified vector representations, which could lead to more efficient computation and easier user interpretation.

Aligned with the outcomes of our user study, the SVGs from S²VG² were preferred for their clarity and simplicity,

underscoring the critical nature of an optimized vectorization process. In essence, the ability to generate SVGs that are both computationally efficient and user-friendly is paramount, particularly when scaling up for broader application in the field of computer graphics and beyond.

5. Discussion

This section delves into the implications of our findings from the S²VG² approach, highlighting its strengths, acknowledging potential limitations, and identifying opportunities for future research.

5.1. Implications of Findings

S²VG² leverages a pre-trained Vision Transformer and a causal BERT decoder to generate simplified yet semantically rich SVGs. This approach yields SVGs that closely resemble original images while explicitly structuring semantic components. Quantitatively, it achieves superior LPIPS and SSIM scores compared to path-optimization methods, demonstrating its adeptness in maintaining both visual fidelity and structural integrity for graphic design automation.

5.2. Enhanced Image Understanding via SVG

Our framework showcases SVGs as a medium for advanced image comprehension. By transforming raster images into structured, parameterized primitives, we encapsulate explicit semantic information. This enables discrete manipulation of image components, facilitating tasks like layout editing and automated visual analysis. Unlike pixel grids or bezier path-soups, our method provides a natively editable, scalable, and LLM-friendly scene representation.

5.3. Limitations

While S²VG² advances semantic SVG generation, we acknowledge two key limitations:

- **Handling Complex Arbitrary Shapes:** Our current evaluation emphasizes foundational geometric and relational understanding (SVG-SHAPE, SVG-Transform, CLEVR-SVG) rather than highly complex real-world vector graphics like intricate icons, fonts, or natural scenes with complex Bezier topologies. Mastering explicit relational semantics is a necessary first step; scaling to arbitrary real-world elements is our immediate future work.
- **Token Constraints:** The VLM’s fixed context window naturally restricts the complexity of the generated SVG. Scenes densely populated with hundreds of elements may exceed token limits, leading to oversimplified outputs. Furthermore, indirect parameter optimization via DiffVG could be improved by integrating end-to-end differentiable backpropagation directly into the VLM.

6. Conclusion

In this work, we introduce S^2VG^2 , a novel approach for generating Scalable Vector Graphics (SVG) through the integration of vision-language models. S^2VG^2 skillfully navigates the complexities of image vectorization, creating SVGs that are not only accurate but also preserve the semantic essence of the original images. This method marks a notable advance in making SVGs more understandable and manageable, enhancing their interpretability for human users.

Through rigorous experiments and comparative analyses, S^2VG^2 is proven to significantly contribute to the field of simple SVG generation. It exemplifies the successful integration of vision and language models in tackling complex graphical tasks. Our findings open new pathways for exploration in SVG generation, hinting at exciting possibilities for automating graphic design and enhancing tools for visual reasoning.

References

- [1] Alexandre Carrier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation, 2020. **1**
- [2] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts, 2021. **5**
- [3] Wen Dai, Tao Luo, and Jianbing Shen. Automatic image vectorization using superpixels and random walkers. In *2013 6th International Congress on Image and Signal Processing (CISP)*, pages 922–926, 2013. **3**
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. **3**
- [5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. **3**
- [6] Maria Dziuba, Ivan Jarsky, Valeria Efimova, and Andrey Filchenkov. Image vectorization: a review, 2023. **1, 3**
- [7] Vage Egiazarian, Oleg Voynov, Alexey Artemov, Denis Volkhonskiy, Aleksandr Safin, Maria Taktasheva, Denis Zorin, and Evgeny Burnaev. *Deep Vectorization of Technical Drawings*, page 582–598. Springer International Publishing, 2020. **3**
- [8] Gerben Hettinga, Jose Echevarria, and Jiri Kosinka. *Efficient Image Vectorisation Using Mesh Colours*. Eurographics Association, 2021. Italian Chapter Conference 2021 : Smart Tools and Apps in Graphics ; Conference date: 28-10-2021 Through 29-10-2021. **3**
- [9] Wonjae Kim, Bokyoung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision, 2021. **3**
- [10] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations, 2016. **5**
- [11] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022. **3**
- [12] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020. **1, 3, 4**
- [13] Zicheng Liao, Hugues Hoppe, David Forsyth, and Yizhou Yu. A subdivision-based representation for vector image editing. *IEEE Transactions on Visualization and Computer Graphics*, 18:1858–1867, 2012. **3**
- [14] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. **3**
- [15] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. *CoRR*, abs/1904.02632, 2019. **1**
- [16] 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 conference on computer vision and pattern recognition*, 2022. **1, 3**
- [17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. **3**
- [18] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention, 2014. **3**
- [19] OpenAI. Gpt-4 technical report, 2023. **3**
- [20] Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images using 1 million captioned photographs. In *Neural Information Processing Systems (NIPS)*, 2011. **5**
- [21] Zhong-Ren Peng and Chuanrong Zhang. The roles of geography markup language (gml), scalable vector graphics (svg), and web feature service (wfs) specifications in the development of internet geographic information systems (gis). *Journal of Geographical Systems*, 6:95–116, 2004. **3**
- [22] Antoine Quint. Scalable vector graphics. *IEEE MultiMedia*, 10(3):99–102, 2003. **1**
- [23] Pradyumna Reddy, Michaël Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351, 2021. **2, 3**
- [24] Hao Su, Xuefeng Liu, Jianwei Niu, Jiahe Cui, Ji Wan, Xinghao Wu, and Nana Wang. Marvel: Raster gray-level manga vectorization via primitive-wise deep reinforcement learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023. **3**
- [25] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan

- Schalkwyk, Andrew M Dai, Anja Hiraoka, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 2
- [26] Xingze Tian and Tobias Gunther. A survey of smooth vector graphics: Recent advances in representation, creation, rasterization and image vectorization. *IEEE transactions on visualization and computer graphics*, PP, 2022. 3
- [27] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator, 2015. 3
- [28] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 2004. 6
- [29] World Wide Web Consortium. Scalable vector graphics (svg) 1.0 specification, 2001. 1
- [30] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey, 2023. 3
- [31] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6
- [32] Shuang Zhao, Frédo Durand, and Changxi Zheng. Inverse diffusion curves using shape optimization. *IEEE Transactions on Visualization and Computer Graphics*, 24(7):2153–2166, 2018. 3
- [33] Hailing Zhou, Jianmin Zheng, and Lei Wei. Representing images using curvilinear feature driven subdivision surfaces. *IEEE Transactions on Image Processing*, 23:3268–3280, 2014. 3
- [34] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 3