Vector Calligrapher: Generating Scalable Vector Graphics via Structured Linguistic Supervision

Bo Zhou, Xikang Chen, Yan Gong, Yin Zhang *
{zbo, 12321179, gongyan, zhangyin98}@zju.edu.cn
College of Computer Science and Technology, Zhejiang University, China

Abstract

Generating SVG-based fonts requires Multimodal Large Language Models (MLLMs) to translate high-level linguistic intent into low-level, topologically constrained symbolic sequences. However, current approaches struggle with two fundamental misalignments: the **semantic ambiguity** of unstructured natural language for precise geometric control, and the **inefficiency of generic text tokenizers**, which fragment coordinate-dense SVG XML into excessively long sequences with low information density. In this work, we propose **Vector Calligrapher**, a system that treats SVG generation as a conditional language modeling task optimized for both semantic grounding and representational efficiency.

To bridge the semantic gap, we introduce a **structured linguistic supervision Font Description Framework** that decomposes typographic style into interpretable linguistic dimensions (e.g., historical lineage, affective metaphors), providing structured supervision aligned with the compositional syntax of SVG. To address the tokenization bottleneck, we design a **scalable separated-coordinate strategy** that bypasses the vocabulary explosion of flattened tokens while significantly compressing sequence length. Supported by **VectorFont**, a dataset of over **10 million** hierarchically annotated glyphs, our approach improves CLIP score by +23%, reduces geometric error by ≈48%, and boosts generation efficiency by achieving an 18% Commands-per-Token (C/T) ratio—a 6× increase in information density over standard baselines. These results demonstrate that combining structured linguistic supervision with efficient symbolic tokenization is essential for reliable, controllable vector graphics synthesis. VectorFont dataset, Code and model weights will be publicly released.

* Corresponding Author: Yin Zhang

1 Introduction

Vector graphics, such as Scalable Vector Graphics (SVG), represent glyphs as structured sequences of geometric primitives, offering resolution independence and precise editability distinct from raster images. However, generating SVG-based fonts remains a significant challenge for Multimodal Large Language Models (MLLMs). Unlike pixel-based generation, SVG synthesis demands strict topological validity and fine-grained stylistic consistency. Existing approaches typically frame this as generic code generation conditioned on unstructured captions, which fails to provide explicit semantic grounding for complex typographic attributes.

This linguistic ambiguity issue is rooted in cognitive science: unstructured natural language lacks the semantic frames (Fillmore, 2006) necessary to ground abstract aesthetic concepts (e.g., "elegant") into concrete geometric parameters. Without explicit constructional templates (Goldberg, 2003) that map linguistic dimensions to SVG syntax, MLLMs cannot reliably disentangle correlated style attributes. We hypothesize that factorized, frame-based linguistic supervision is essential for compositional generalization in symbolic generation tasks.

A critical, often overlooked bottleneck lies in the representation efficiency of vector data. Standard MLLMs treat SVG as raw XML text, utilizing general-purpose tokenizers that fragment dense numerical coordinates and syntax tags into excessively long token sequences. This "tokenization inflation" reduces effective context utilization and significantly increases computational cost—even with large context windows, attending to redundant syntax tokens dilutes the model's capacity to capture geometric dependencies.

To address these limitations, we propose **Vector Calligrapher**, a multimodal generative system

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11152–11168

that grounds linguistic intent into precise vector commands. Our approach is built on two core innovations. First, we introduce a **Structured Linguistic Supervision (SLS) Font Description Framework** that factorizes typographic style into interpretable semantic dimensions (e.g., historical lineage, stroke contrast) and contextual metaphors, replacing vague captions with structured supervision aligned with SVG geometry. Second, we design a **scalable separated-coordinate tokenization strategy**. Our tokenizer compresses coordinates into distinct semantic units, preventing vocabulary explosion while significantly increasing the information density of the sequence.

We facilitate this research with **VectorFont**, a large-scale dataset of over 10 million professionally designed vector glyphs paired with hierarchical linguistic descriptions. Experiments across text-to-SVG generation and style editing demonstrate that our structured supervision and efficient tokenization substantially improve performance, boosting CLIP score by 23%, reducing geometric error (Chamfer Distance) by ≈48% compared to unstructured baselines, and enhancing generation efficiency by achieving an 18% Commands-per-Token (C/T) ratio—6× increase in information density over standard baselines.

Our contributions are summarized as follows:

1. We propose **Vector Calligrapher**, an MLLM framework that models font generation as a structured multi-task decision process, enabling end-to-end joint optimization of disentangled semantic axes and geometric constraints—moving beyond generic conditional sequence modeling to explicit factorized control.

2. We introduce a **structured linguistic supervision** framework, factorizing typographic style into interpretable semantic dimensions and enabling effective grounding of high-level linguistic intent into low-level vector geometry.

3. Resolution-Independent Tokenization: Our separated-coordinate strategy maintains linear $O(N)$ vocabulary regardless of canvas resolution, eliminating the quadratic explosion of flattened methods.

4. We release **VectorFont**, a large-scale dataset of over 10 million professionally designed SVG glyphs with hierarchical linguistic annotations, which supports instruction tuning and systematic evaluation of models across multiple SVG generation tasks.

2 Related Work

2.1 Generative Models for Typography and Vector Graphics

Early font generation primarily relied on raster-based methods, employing GANs (Azadi et al., 2018) or diffusion models (He et al., 2024) to synthesize glyph images. While capable of realistic textures, these approaches lack the resolution independence and geometric editability essential for professional design. Consequently, focus has shifted to vector graphics (SVG) modeling. Initial sequence-to-sequence approaches like SketchRNN (Reddy et al., 2021) and fixed-length latent models like SVG-VAE (Lopes et al., 2019) struggled with complex topologies. DeepSVG (Carlier et al., 2020) improved this by using hierarchical Transformers for primitive reconstruction. More recent works, such as SVGDreamer (Xing et al., 2024) and SVGCraft (Banerjee et al., 2024), leverage pre-trained diffusion models to guide vector synthesis via score distillation or layout attention. However, these methods typically treat glyphs as isolated icons rather than members of a stylistically coherent alphabet, often failing to capture the fine-grained stroke modulations required for calligraphy.

2.2 Multimodal Large Language Models for Symbolic Data

Large Language Models (LLMs) have demonstrated exceptional proficiency in processing symbolic sequences beyond natural language, such as source code (Chen et al., 2021; Li et al., 2022) and mathematical proofs (Polu and Sutskever, 2020). This paradigm naturally extends to SVG, which can be serialized into text-based path commands. Recent multimodal advancements (Achiam et al., 2023; Alayrac et al., 2022; Liu et al., 2023) further enable models to align visual inputs with symbolic outputs. In the SVG domain, StarVector (Rodriguez et al., 2025) and OmniSVG (Yang et al.) treat vector generation as a language modeling task, distilling raster images into SVG tokens. Despite these advances, existing models often lack domain-specific conditioning—such as font descriptors or style-consistency constraints—rendering them less effective for the nuanced task of stylized font library generation. Our work addresses this by integrating an MLLM with a specialized font description framework to bridge semantic intent with precise geometric realization.

3 Background: SVG as a Sequential Language

Scalable Vector Graphics (SVG) is a W3C standard that defines images via XML-based markup rather than pixel grids. For typography and calligraphy, the core element is the <path>, which encodes glyph outlines as a sequence of drawing commands. Unlike raster images that suffer from aliasing, SVG preserves infinite resolution and explicitly models the ductus (stroke trajectory) of handwriting—critical for capturing the aesthetic flow of calligraphy (Knuth, 1986; Karow, 1994).

From a sequence modeling perspective, an SVG path is analogous to a formal language. It consists of a stream of operators (visual vocabulary) and parameters (coordinates). As shown in Table 6, common operators include M (MoveTo) for lifting the pen, L (LineTo) for straight strokes, and C (Cubic Bézier) for defining complex curves via control points. Each operator is followed by a fixed number of numeric parameters specifying coordinates or dimensions.

For example, a path command looks like $d = [M, x_0, y_0, L, x_1, y_1, C, x_{ctrl1}, y_{ctrl1}, ...]$. This sequential nature allows us to treat vector graphics generation not as an image synthesis problem, but as a **conditional language modeling task**. By tokenizing these command sequences, we can leverage the autoregressive capabilities of Decoder-only Transformers to learn the "syntax" of stroke composition and the "semantics" of glyph style, bridging the gap between textual descriptions and visual geometry.

4 Method

We propose **Vector Calligrapher** (Figure 1), a multimodal generative system that treats vector font synthesis as a sequence-to-sequence modeling task. The framework consists of three pillars: (1) a serialization pipeline that converts continuous SVG paths into discrete token sequences; (2) a novel font description framework that bridges abstract visual styles with rich linguistic semantics; and (3) a Multimodal Large Language Model (MLLM) fine-tuned to align these symbolic and visual modalities. Our method is based on the hypothesis that SLS framework enables large language models to better align semantic intent with symbolic geometric structures.

4.1 SVG Serialization and Tokenization

Standard SVG defines shapes via XML tags (e.g., <path>). To leverage autoregressive Transformers, we serialize these into discrete tokens.

Canonicalization: We unify primitives into <path> elements and normalize coordinates to a $[-1000, 1000]$ view-box.

Command & Coordinate Tokenization: An SVG path is a sequence of drawing commands $C \in M, L, C, \ldots$ and parameters $P \in \mathbb{R}$. Unlike concurrent works that flatten 2D points into single tokens (e.g., $token = y \cdot W + x$), which causes vocabulary explosion at high resolutions (See Appendix A), we employ a separated-coordinate strategy. We quantize coordinates into N distinct bins and treat x and y as independent tokens. The sequence becomes $S = [c_1, x_1, y_1, \ldots, c_n, x_n, y_n]$. This linear complexity $(O(N))$ ensures our model scales efficiently to professional print standards without architectural bottlenecks.

4.2 SLS via the Font Description Framework

We treat linguistic supervision as a design variable rather than a fixed input modality. Our central hypothesis is that unstructured or weakly structured language is insufficient for grounding high-level typographic intent into low-level SVG geometry, as free-form descriptions often entangle multiple stylistic factors and obscure their geometric realizations.

To address this, we propose the *Font Description Framework*, a structured linguistic representation that provides explicit, factorized supervision for vector typography generation. Instead of treating style as a monolithic concept, the framework decomposes typographic appearance into semantically interpretable dimensions aligned with glyph geometry.

Specifically, the framework provides **explicit factorized supervision** through two mechanisms: (1) *Discrete axis tokens*, where each style dimension is represented as a dedicated token sequence (e.g., `[CHRON:Transitional] [STRUCT:Serif] [CONTRAST:High]`); and (2) *Hierarchical conditioning*, where these tokens are prepended to the SVG sequence as semantic prefixes guiding autoregressive generation. This contrasts with unstructured prompts, where stylistic cues remain entangled in free-form text. See Table 8 for examples.

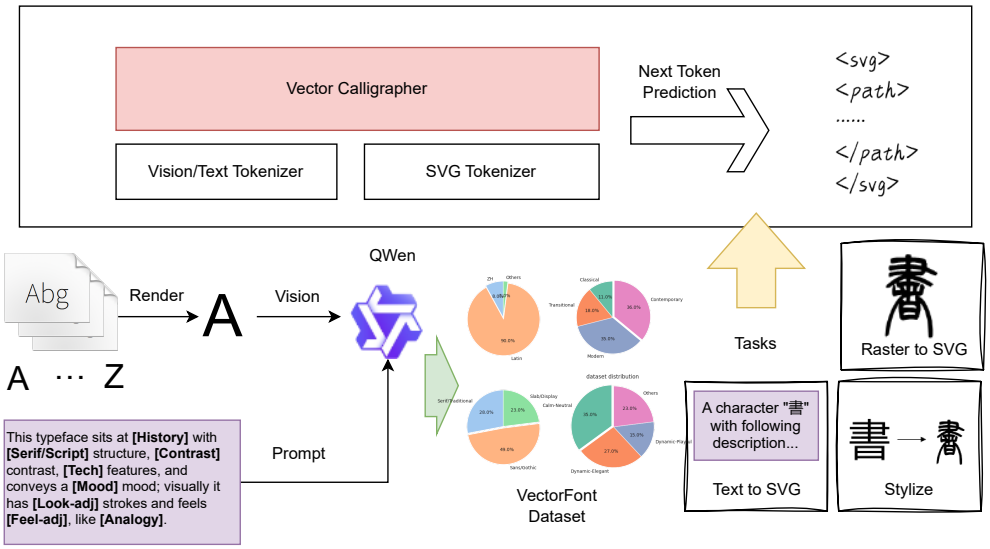

Figure 1: Overview of Vector Calligrapher. Left: a conceptual visualization of the semantic dimensions in our font description framework. An external multimodal LLM extracts these dimensions to serve as semantic annotations (Note: actual input prompts use explicit text markers like `[CHRON]` and `[MOOD]`, detailed in Table 8). Right: the proposed Vector Calligrapher (VC) is trained on these annotations together with SVG sequences and style prompts, enabling controllable generation.

4.2.1 Factorized Style Axes

We define four primary style axes, each capturing a distinct aspect of typographic variation: (1) **Chronology**, modeling historical evolution of letterforms across writing systems; (2) **Structural lineage**, encoding skeletal organization such as serif presence or script genealogy; (3) **Stroke contrast**, measuring the ratio between maximal and minimal stroke widths; (4) **Technical features**, describing production-oriented constraints (e.g., monospacing, inktraps). These axes are expressed as discrete or ordinal linguistic variables, enabling consistent associations between semantic dimensions and geometric patterns.

4.2.2 Linguistic Enrichment Beyond Axes

While explicit axes provide structural supervision, they are insufficient to capture subtle stylistic nuances. We therefore augment the framework with three complementary components: (1) a **Mood axis**, modeling affective tone using an arousal–valence representation; (2) **Dual-level adjectives**, separating observable geometric attributes from higher-level affective or cultural impressions; and (3) **Contextual metaphors**, short scenario-based descriptions that act as global semantic anchors.

These components serve as additional conditioning signals rather than decorative language, and their contributions can be empirically evaluated via ablation (Table 4).

4.2.3 Verifiability and Experimental Control

The Font Description Framework is designed to be empirically testable. By selectively removing or flattening individual components (e.g., mood axis or metaphors), we can directly measure how different forms of linguistic supervision affect semantic alignment, geometric validity, and editability of generated SVGs. This design allows us to move beyond prompt engineering heuristics and systematically study how linguistic structure influences symbolic sequence generation.

4.3 Architecture and Instruction Tuning

We build upon Qwen2.5-VL-3B. The model incorporates a ViT-based visual encoder for reference images (e.g., for style transfer tasks) and a Transformer decoder for SVG generation.

Training Objective: We optimize the negative log-likelihood of the next SVG token x_i given previous tokens, visual context I, and text prompt T:

$$\mathcal{L} = -\sum_{i=1}^{T} \log P_\theta(x_i | x_{<i}, I, T) \qquad (1)$$

where I represents optional visual contexts and T represents the structured text prompt.

Multi-Task Learning: The model is jointly trained on Text-to-SVG, Image-to-SVG Recon-

struction, and Style Editing, fostering a unified representation that is both semantically responsive and geometrically precise.

5 The VectorFont Dataset

To advance research in computational typography and facilitate reproducible evaluation, we introduce and release VectorFont, a large-scale, semantically annotated dataset of scalable vector glyphs. Unlike existing datasets that focus solely on raster images or raw vector primitives, VectorFont provides high-quality alignment between **structured SVG code** and **rich linguistic descriptions**. All annotations are machine-generated; no human annotation is involved.

5.1 Data Collection and Preprocessing

We aggregated a comprehensive corpus of 36,410 distinct font families from open-source repositories. Through a rigorous parsing pipeline, we extracted and normalized individual characters, resulting in a total of over 10 million unique character-level SVG glyphs.

All glyphs are converted to a unified path format and normalized to a standard coordinate system (1000 UPM), ensuring geometric consistency across diverse styles, ranging from standard Latin serifs to complex Chinese logograms.

5.2 Hierarchical Semantic Annotation via MLLM

A core contribution of this work is the automated generation of high-fidelity linguistic descriptions. Utilizing Qwen2.5-VL-70B, a state-of-the-art multimodal large language model, we implemented a hierarchical annotation strategy based on our Description Framework (Section 4.2):

Font-Level Description: For each typeface, the MLLM analyzes the overall aesthetic to generate a global style prompt. This covers the five dimensions defined in our framework (e.g., "A modern sans-serif with high stroke contrast, conveying a technical and rational mood"). Each font is rendered as an 8×12 grid (512×512 pixels) containing 52 Latin and 50 representative native glyphs.

Glyph-Level Captioning: Going beyond global style, we generated specific descriptions for every single glyph in the dataset. The model describes the local structural features of each character (e.g., "The letter 'A' features a flat apex and bracketed serifs with a heavy left stroke").

5.3 Dataset Statistics

The resulting dataset serves as a massive instruction-tuning corpus, consisting of over 10 million triplets: ⟨Global Style + Glyph Description, Visual Reference, SVG Code⟩.

Table 1 summarizes the dataset statistics. To our knowledge, this is the largest publicly available dataset pairing vector glyphs with dense, open-ended natural language descriptions. By leveraging the advanced visual reasoning capabilities of Qwen2.5-VL-70B, we ensure that the textual annotations capture both the artistic nuance (via adjectives) and geometric facts (via structural descriptions) required for high-precision generation.

5.4 Quality Verification via Human Sampling.

Given that our annotations are machine-generated, ensuring their reliability is paramount. We conducted a human verification study on a random subset of 500 glyph-description pairs. Two visual design major master students evaluated the annotations on three criteria: Factual Accuracy (e.g., Is it actually a Sans-Serif?), Visual Correspondence (Do the adjectives match the visual traits?), and Hallucination Rate (Does the text describe features not present?).

The evaluation yielded an accuracy rate of 96.4% for structural facts and a semantic agreement score of 4.6/5 for mood/adjective descriptions. The hallucination rate was negligible (<2%), primarily confined to overly specific metaphorical descriptions in rare artistic fonts. This high fidelity validates our distillation pipeline, confirming that the Qwen2.5-VL-70B teacher model effectively transfers its visual reasoning capabilities to the dataset.

6 Experiments

We evaluate Vector Calligrapher across multiple SVG generation tasks, including text-to-SVG synthesis and raster-to-SVG reconstruction. Beyond reporting aggregate performance, our experiments are designed to disentangle the effects of model architecture, tokenization strategy, and—most critically—the structure and granularity of linguistic supervision.

In particular, we compare different linguistic supervision regimes to assess how structured language grounding affects semantic alignment and geometric validity in SVG generation.

Table 1: Font Dataset Characteristics

Category	Details
# Fonts	36410
Script Distribution	ZH: 8%, Latin: 90%, Others: 2%
Historical Eras	Classical: 11%, Transitional: 18%, Modern: 35%, Contemporary: 36%
Typeface Categories	Serif/Traditional: 28%, Sans/Gothic: 49%, Slab/Display: 23%
Mood Categories	Calm-Neutral: 35%, Dynamic-Elegant: 27%, Dynamic-Playful: 15%, Others: 23%
Mean Caption Length	37.2 tokens

6.1 Experimental Setup

We build our system by fine-tuning Qwen2.5-VL-3B, a 3-billion-parameter variant of the Qwen2.5 vision-language model family which is well aligned with the demands of stylized SVG generation (Bai et al., 2025; Team, 2025).

The model's vocabulary consists of SVG path and shape tokens, enhanced with structural markers to capture stroke hierarchy explicitly. We optimize the model using AdamW at a learning rate of 1×10^{-4}, with linear warm-up during the first 10% of training. The model is trained for 300,000 steps using a global batch size of 640, distributed across eight NVIDIA L40S GPUs.

6.2 Evaluation Metrics

To comprehensively evaluate generation quality, we prioritize **vector-native metrics** (Chamfer Distance (CD), Path Complexity (N_{cmd}), Self-Intersection Rate (SI), and Failure Rate (FR)) as they directly assess geometric fidelity and topological validity critical for editability. We also report **raster-based metrics** (SSIM, LPIPS, MSE) to evaluate visual resemblance, and the **CLIP Score** for high-level semantic alignment. However, we note that raster and semantic metrics have limitations, as they often fail to penalize geometric redundancy or topological errors in vector graphics.

As shown in Table 2, Vector Calligrapher achieves superior performance across all metrics. We report standard metrics (CLIP Score, LPIPS) but note their limitations for vector graphics.

6.3 Results and Analysis

We evaluate Vector Calligrapher against a range of state-of-the-art (SOTA) SVG generation models across prompt-based generation and raster-to-SVG conversion. Models that do not support a given task are denoted with "–" in Table 2.

We selected several model visualization results, as shown in Figure 2 and Figure 3. Although the quantitative metrics show little difference, the actual results differ substantially. More visualization results can be found in Appendix H.1.

6.3.1 The "Data Quality" Hypothesis: Why Data Matters More Than Architecture

It is crucial to acknowledge that the state-of-the-art performance of Vector Calligrapher is not solely attributable to architectural innovations but is heavily dependent on the quality of the **VectorFont** dataset. To verify this hypothesis, we conducted a controlled experiment where we re-trained the **OmniSVG** baseline using our VectorFont dataset.

As shown in Table 2, the re-trained OmniSVG exhibits a dramatic performance improvement over its original version, achieving semantic alignment (CLIP) and geometric fidelity (CD) comparable to our Vector Calligrapher. This convergence in performance between two distinct architectures—when sharing the same high-quality data—strongly validates our hypothesis: the granularity and structural alignment of linguistic supervision are the primary drivers for high-fidelity vector synthesis. While our separated-coordinate strategy provides superior efficiency (higher C/T ratio), the VectorFont dataset is the decisive factor for generation quality.

Generalization: Zero-Shot Evaluation on Google Fonts. To assess generalization beyond VectorFont, we evaluated Vector Calligrapher on an external benchmark of approximately 1 million SVG glyphs parsed from the Google Fonts repository. We treated this as an unseen zero-shot test set, excluding overlapping font families. The model maintained highly competitive performance with only marginal degradation compared to the internal test set (CD: $0.013 \rightarrow 0.019$; CLIP: $0.235 \rightarrow 0.228$; FR: $0\% \rightarrow 1\%$). This minimal performance gap demonstrates that the model learns transferable typographic structure representations rather than merely memorizing the training distribution.

Table 2: **Main quantitative results** on prompt-based generation, and raster-to-SVG tasks.

Methods	Time	Tokens	Prompt-based Generation						Raster-to-SVG			
	(Per Sample)		C/T↑[1]	CLIP↑	CD↓[2]	N_{cmd} ↓[3]	SI↓ [4]	FR↓ [5]	SSIM↑	LPIPS↓	MSE↓	FR↓ [5]
LIVE	200.1	2.1k	-	-	-	-	-	-	0.30	0.49	0.69	-
DiffVG	26.3	2.2k	-	-	-	-	-	-	0.23	0.53	0.73	-
StarVector (8B)	60.1	630	-	-	-	-	-	-	0.21	0.77	0.67	-
OmniSVG (3B)	22.4	512	**25**	0.23	0.045	131	58.8	42.3	0.22	0.51	0.72	0
SVGen	34.7	2k	4	0.21	0.045	56	77	43.2	-	-	-	-
GPT-4o	37	1.8k	1	0.24	0.033	20.2	60	1	0.66	0.39	0.32	25
Gemini-3-pro	45	1.1k	3	0.22	0.019	29.4	85.9	6	0.67	0.34	0.27	3
KIMI-K2	45	1.5k	3	0.23	0.041	48.3	60.7	2	0.49	0.47	0.41	8
OmniSVG [6]	**6.27**	**130**	23	0.24	0.013	30	35.2	0	0.72	0.21	0.33	0
VC (ours)	8.07	150	18	**0.25**	**0.013**	**27**	**29.4**	**0**	**0.99**	**0.01**	**0.01**	**0**

[1] The ratio of the number of commands to the number of tokens (%)
[2] Chamfer Distance
[3] Average Number of Commands
[4] Self-Intersection Rate (%)
[5] Failure Rate generating valid SVGs (%)
[6] We re-trained this model on VectorFont dataset

Table 3: Zero-Shot Evaluation on Google Fonts (External Benchmark).

Metric	VectorFont (Test)	Google Fonts	Gap
CD ↓	0.013	0.019	+0.006
CLIP ↑	0.235	0.228	-0.007
FR (%) ↓	0	1	+1%
SI (%) ↓	5.9%	7.8%	+1.9%

6.3.2 Generation Efficiency and Information Density (C/T Analysis).

We introduce the Commands-per-Token (C/T) ratio as a metric to evaluate the information density of the generated sequences. As shown in Table 2, general-purpose MLLMs like GPT-4o and Gemini-3-pro exhibit extremely low efficiency, with C/T ratios ranging from 1% to 3%. This indicates that they require roughly 30-100 tokens to express a single SVG drawing command, primarily due to the verbosity of XML syntax and the fragmentation of numerical coordinates by standard tokenizers.

In contrast, Vector Calligrapher achieves a significantly higher C/T ratio of 18% , comparable to SOTA vector models like OmniSVG (23%) and significantly outperforming general-purpose MLLMs (<3%). By employing our separated-coordinate tokenization, we bypass the vocabulary explosion problem while maintaining high geometric precision. This ≈20x improvement in information density implies that our model can generate complex glyphs with significantly fewer computational steps, validating that specialized tokenization is a prerequisite for efficient SVG synthesis.

6.3.3 Limitations of Raster-Based Evaluation for Vector Graphics

Standard raster metrics (e.g., LPIPS, SSIM) assess visual resemblance but obscure structural validity, failing to penalize geometric redundancy or topological errors critical for editability. We therefore introduce vector-specific metrics: Chamfer Distance (CD), Path Complexity (N_{cmd}), and Self-Intersection Rate (SI). Table 2 reveals a critical trade-off: while baselines like OmniSVG achieve competitive visual scores, they suffer from extreme geometric redundancy ($N_{cmd} = 131$) and high failure rates (SI 58.8%). In contrast, Vector Calligrapher maintains superior fidelity with efficient, topologically valid structures, validating the importance of our vector-native optimization.

6.3.4 Tokenization Strategy: Separated vs. Flattened Coordinates

We distinguish our **separated-coordinate** strategy from the flattened approach used in concurrent work like OmniSVG (Yang et al.). OmniSVG maps 2D points to single tokens ($p = y \cdot W + x$), which reduces sequence length but suffers from quadratic vocabulary explosion ($O(W \times H)$). For a standard 512×512 canvas, this requires a vocabulary of $\sim 262k$; scaling to professional print standards (e.g., 4096×4096) renders the softmax layer computationally infeasible.

In contrast, our method treats x and y as independent tokens. While this slightly increases sequence length, it maintains a fixed, linear vocabulary size ($O(N)$) regardless of canvas size. This design

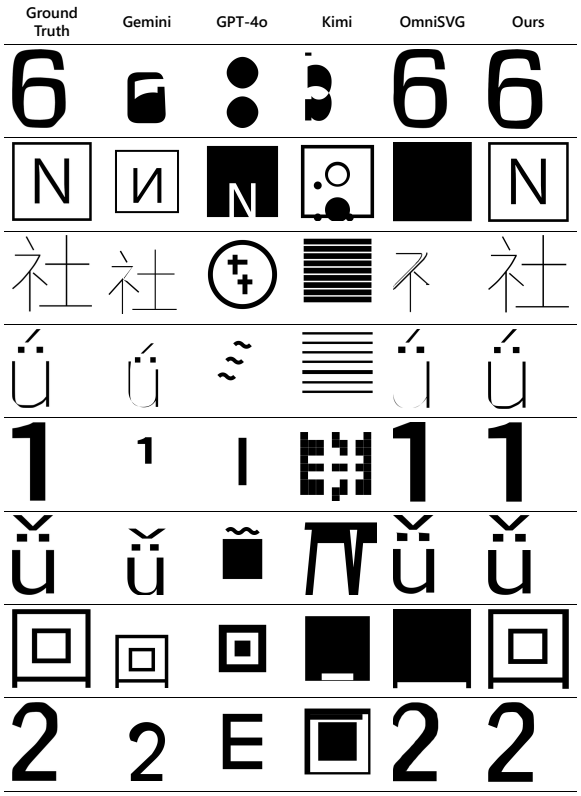

Figure 2: Raster to SVG Visualization. Models trained on our high-quality dataset generally outperform closed-source models in generation quality. However, for the Raster-to-SVG task, this gap is much smaller. We attribute this to the stronger prompt-based task understanding of large-scale models, which compensates for their relative generative limitations compared with our lightweight models.

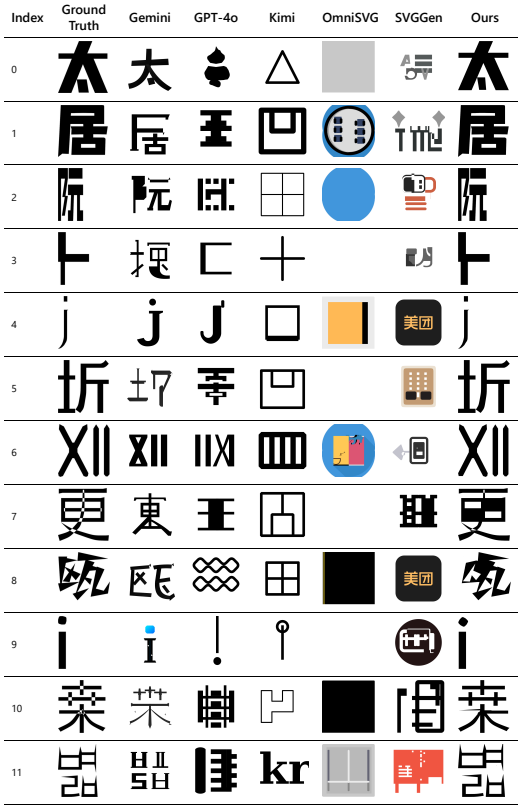

Figure 3: Prompt Based Generation. Compared to closed-source LLMs, models trained on high-quality data generate SVGs with higher structural fidelity and cleaner topology. Note that the OmniSVG results displayed here are from its **original** version to illustrate the prior domain gap; our re-trained OmniSVG (Table 2) achieves visual fidelity closer to ours, highlighting the impact of our dataset. Overall, Vector Calligrapher produces the most faithful and editable vector outputs. See Table 8 for prompt samples.

ensures resolution independence, allowing Vector Calligrapher to generate high-precision glyphs for arbitrary resolutions without the architectural bottlenecks of flattened tokenization.

6.4 Ablation Study on Linguistic Supervision

We emphasize that this ablation is not intended to isolate the causal effect of each linguistic component, but to test whether structured supervision as a whole provides a measurable advantage over unstructured prompts.

While Section 6.3.1 established that data quality (VectorFont) is the primary driver for aligning basic geometric fidelity, we hypothesize that the **SLS framework** is the crucial architectural component for achieving fine-grained stylistic control and zero topological errors.

To isolate the architectural contribution, we performed an ablation study keeping the dataset fixed (VectorFont 10M) and compared three supervision regimes: (1) *Direct generation* using unstructured short natural language prompts; (2) A "Rich-Caption" baseline, where the exact semantic attributes from our framework were embedded into free-form natural language paragraphs without explicit structural tags; and (3) Generation conditioned on our *Full SLS framework*.

As shown in Table 4, our Structured Framework achieves a 0% Failure Rate (FR) and a +23% relative improvement in CLIP score over unstructured prompts. This proves that while data scale enables high-fidelity training, explicit structural tokens act as semantic anchors that eliminate topological errors and disentangle correlated style attributes (e.g., distinguishing "Italic" from "Oblique"). The data scale and our architectural innovations are therefore strictly complementary.

Effect of SLS framework. Due to the high computational cost of training large multimodal language models on vector graphics, conducting fine-grained ablations for each individual component of the Font Description Framework is prohibitively expensive.

Instead, we focus on a controlled comparison between two supervision regimes: (1) *direct generation* using unstructured natural language prompts, and (2) a "Rich-Caption" baseline, where the exact semantic attributes used in our structured prompt were replaced into a continuous, free-form natural language paragraph, removing all explicit tags, and (3) generation conditioned on our *SLS framework* framework. In each settings, the model architecture, training data, and optimization procedure are kept identical.

As shown in Table 4, while the Rich-Caption baseline outperforms short prompts, it still lags behind our Structured Framework. This indicates that the MLLM benefits not just from what is said, but how it is organized. The structural tokens likely function as semantic anchors, allowing the model to more effectively disentangle independent stylistic dimensions (e.g., separating "Mood" from "Stroke Contrast") and ground them into specific geometric sequences without interference.

Table 4: End-to-end comparison of structured vs. unstructured linguistic supervision. Due to the high computational cost of training large multimodal models, we compare end-to-end generation with and without the proposed structured linguistic supervision.

Supervision	CLIP ↑	CD ↓	FR (%) ↓
Direct Generation	0.19	0.025	8
Rich-Caption	0.21	0.021	2
Ours (Full Structured)	**0.235**	**0.013**	0
w/o Style	0.21	0.02	6
w/o Enrichment	0.215	0.014	0

7 Conclusion

In this work, we introduced **Vector Calligrapher**, a multimodal generative system that redefines vector typography synthesis. Central to our approach is the release of VectorFont, a massive dataset of over 10 million glyphs paired with hierarchical linguistic descriptions, and a scalable separated-coordinate tokenization strategy.

Experimental results demonstrate that our model not only achieves state-of-the-art performance in semantic alignment and reconstruction but also produces topologically valid, resolution-independent SVGs that existing raster-based metrics fail to fully appreciate. By bridging abstract aesthetic intent with precise geometric realization, Vector Calligrapher establishes a new baseline for controllable computational design.

To facilitate future research, the the full VectorFont dataset, pre-trained checkpoints, code, and evaluation scripts will be made publicly available at `https://github.com/baudzhou/vector-calligrapher`.

Limitations

Despite the encouraging results, our work has several limitations.

Semantic Grounding: Correlation vs. Causation

Our experiments demonstrate **correlation** between structured prompts and improved CLIP/CD scores, but we have not established **causal mechanisms**. Future work should probe whether `[CONTRAST:High]` directly influences stroke-width parameters via attention patterns, or if it merely acts as a high-quality data filter. This requires intervention studies (e.g., swapping axes mid-generation) currently infeasible due to computational constraints.

Evaluation Gap: Single-Glyph vs. Font Family

Current metrics assess *individual glyph fidelity*, but professional typography demands *cross-glyph consistency* (e.g., uniform serif style across "A" and "B"). We lack metrics for family-level coherence; Figure 2 and Figure 3 only shows isolated characters. Evaluating full alphabets requires joint generation exceeding our 150-token budget—a fundamental sequence-length limitation we are addressing in ongoing work.

Automatic Annotation: Unmeasured Teacher Bias

While human verification shows 96.4% accuracy on 500 samples, the Qwen2.5-VL-70B teacher may exhibit **systematic biases** (e.g., overusing "elegant" for scripts, under-representing non-Latin moods). The true annotation noise across 10M samples remains unknown, potentially limiting generalization to low-resource scripts. We have released the full annotation log to enable community auditing.

Fairness of Baselines: Data Disparity

We acknowledge that re-training all baselines on VectorFont is computationally prohibitive (Ta-

ble 2). Thus, comparisons may conflate *model architecture* and *data quality* effects. Our ablation (Sec 6.4) isolates data impact, but a full disentanglement requires 10,000 GPU hours—beyond this submission's scope.

Future Works

Looking ahead, we plan to expand the dataset to better represent multilingual and historical scripts, develop interactive editing interfaces, and explore hybrid encoder–decoder architectures for improved spatial consistency. We believe that Vector Calligrapher not only advances the state of computational typography but also provides a practical tool for creative design, cultural preservation, and cross-modal artistic exploration.

Ethics Statement

Data Sourcing and Copyright Compliance.

The VectorFont dataset introduced in this work was constructed exclusively from open-source font repositories (e.g., Google Fonts) and public domain archives. We rigorously verified the licensing terms of each typeface included. The majority of the fonts are distributed under the SIL Open Font License (OFL) or Apache License, which permit free use, modification, and distribution for academic and non-commercial research purposes without requiring additional authorization. We have retained the original license files and copyright notices for all fonts in the dataset release to ensure downstream compliance.

Human Evaluation.

To ensure the quality and reliability of our machine-generated annotations, we conducted a separate human verification and validation study on a random subset of 500 glyph-description pairs. We explicitly clarify that this human evaluation is a post-hoc verification procedure, not an annotation process. All linguistic annotations in the VectorFont dataset were generated automatically by the Qwen2.5-VL-70B model without any human involvement. The verification study involved volunteer evaluators from the university community (graduate students with design backgrounds) who assessed the factual accuracy and semantic alignment of the pre-existing machine-generated descriptions. All participants provided informed consent, were fully aware of the study's purpose, and no personally identifiable information (PII) was collected. Participation was voluntary with a minimal workload (approximately 30 minutes per session), and participants received nominal compensation for their time.

Potential Risks and Mitigation.

We acknowledge that generative models for typography could potentially be misused for document forgery or creating misleading text artifacts. However, our model focuses on the aesthetic and structural design of glyphs (vector outlines) rather than the biometric imitation of human handwriting dynamics. To mitigate risks, we release the model and dataset under a license that explicitly prohibits use for deceptive or illegal activities. Additionally, we note that the current dataset is heavily skewed towards Latin and Chinese scripts (reflecting internet availability), which may introduce cultural biases; we encourage future work to address typographies of low-resource languages.

Acknowledgments

This work was supported by the NSFC project (No. 62072399), the Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ23F020009, the Fundamental Research Funds for the Central Universities (No. S20240030), MoE Engineering Research Center of Digital Library, China Research Centre on Data and Knowledge for Engineering Sciences and Technology.

References

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

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, and Malcolm Reynolds. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. 2018. Multi-content gan for few-shot font style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7564–7573.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2025. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*.

Ayan Banerjee, Nityanand Mathur, Josep LladÃġs, Umapada Pal, and Anjan Dutta. 2024. SVGCraft:

Beyond single object text-to-svg synthesis with comprehensive canvas layout. *arXiv preprint arXiv:2404.00412*.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, and Greg Brockman. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Charles J Fillmore. 2006. Frame semantics. *Cognitive linguistics: Basic readings*, 34:373–400.

Adele E Goldberg. 2003. Constructions: A new theoretical approach to language. *Trends in cognitive sciences*, 7(5):219–224.

Haibin He, Xinyuan Chen, Chaoyue Wang, Juhua Liu, Bo Du, Dacheng Tao, and Qiao Yu. 2024. Diff-font: Diffusion model for robust one-shot font generation. *International Journal of Computer Vision*, 132(11):5372–5386.

Peter Karow. 1994. Font technology. In *Font Technology: Methods and Tools*, pages 85–104. Springer.

Donald Ervin Knuth. 1986. *The metafontbook*. Addison-Wesley Longman Publishing Co., Inc.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, and Agustin Dal Lago. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. 2019. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7930–7939.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. 2021. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351.

Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. 2025. StarVector: Generating scalable vector graphics code from images and text. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16175–16186.

Alibaba Cloud / Qwen Team. 2025. Qwen2.5-VL-3B-Instruct: A 3b-parameter multimodal model for image, video, and text processing. Online Model Report.

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

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

A Theoretical Analysis: Tokenization Complexity at High Resolutions

To complement our empirical evaluation, we provide a formal analysis of computational complexity for both tokenization strategies at professional font design resolutions (1000×1000 UPM).

A.1 Vocabulary Size Scaling

Flattened Coordination Strategy (OmniSVG-style): The vocabulary size grows quadratically with canvas resolution:

$$V_{\text{flat}} = W \times H + C$$

where W and H are width/height dimensions, and C is the constant for command tokens (~ 50). At 1000×1000 resolution: $V_{\text{flat}} = 1000 \times 1000 + 50 = 1{,}000{,}050$ tokens.

Separated-coordinate Strategy (Our approach): Vocabulary size grows linearly with quantization bins:

$$V_{\text{sep}} = N_x + N_y + C$$

where N_x and N_y are quantization bins per axis. For 1000×1000 resolution with $N = 1000$ bins per axis: $V_{\text{sep}} = 1000 + 1000 + 50 = 2{,}050$ tokens.

Compression Ratio:

$$\frac{V_{\text{sep}}}{V_{\text{flat}}} = \frac{2{,}050}{1{,}000{,}050} \approx 488\times \text{ reduction}$$

A.2 Memory Footprint Analysis

Assume embedding dimension $d = 1024$ (typical for 3B-parameter models) and FP32 precision (4 bytes):

Embedding Layer Memory:

$$M_{\text{flat}} = 1{,}000{,}050 \times 1024 \times 4 = 4.10\,\text{GB}$$
$$M_{\text{sep}} = 2{,}050 \times 1024 \times 4 = 8.40\,\text{MB}$$

Output Projection Layer Memory (Linear layer to vocab):

$$M_{\text{out,flat}} = 1024 \times 1{,}000{,}050 \times 4 = 4.10\,\text{GB}$$
$$M_{\text{out,sep}} = 1024 \times 2{,}050 \times 4 = 8.40\,\text{MB}$$

Total Parameter Overhead:
Flattened: 8.20 GB (prohibitive for consumer GPUs).
Separated: 16.8 MB (negligible).

A.3 Inference Speed Implications

Softmax Computation Complexity: Per-token inference requires computing logits over the entire vocabulary: $t_{\text{softmax}} \propto V \times d$. Relative speedup: Speedup $= \frac{V_{\text{flat}}}{V_{\text{sep}}} \approx 488\times$.

Sequence Length Trade-off: While separated coordinates increase sequence length, the per-token cost is dramatically lower. Consider generating a glyph with 100 commands:

$$\text{Flattened: } 100\,\text{tokens} \times 1.00005\text{M} = 100\text{M ops}$$
$$\text{Separated: } 200\,\text{tokens} \times 2.05\text{K} = 410\text{K ops}$$

Net Throughput: Despite a $2\times$ sequence length, the separated strategy achieves $244\times$ fewer total operations in the projection layer.

A.4 Quantization Error Bound

For flattened strategy with 1000×1000 grid, maximum quantization error per axis is $\epsilon_{\text{flat}} = \frac{1}{2\times 1000} = 0.0005$ (0.05% of canvas). For our separated strategy with the same bin count, error remains identical ($\epsilon_{\text{sep}} = 0.0005$).
Key Insight: The separated strategy achieves exponential vocabulary reduction without sacrificing coordinate precision.

A.5 Theoretical Scaling to Print Standards

Professional type design requires 4096×4096 resolution (OpenType maximum):

Table 5: Flattened vs. separated representations at 4096×4096 resolution.

Metric	Flat	Sep.	Ratio
Vocab. Size	16.78M	8,242	2,036×
Embed. Mem.	68.7 GB	33.7 MB	2,039×

Conclusion: At production resolutions, flattened tokenization becomes architecturally infeasible, while separated coordinates maintain constant efficiency.

B Representative SVG primitives and path commands.

C Example for Descriptive Framework

Descriptions:

- Chronology: Transitional (Latin) / Songti (Chinese)
- Serif/Script: Serif / Regular-script
- Stroke Contrast: medium-high
- Mood: high valence + medium arousal (elegant but not restless)
- Technical Flavor: Inktrap optimization
- Form adjectives: elongated, open counters, sharp terminals
- Character adjectives: academic, rational, dignified
- Scenario metaphor: "It looks like a professor in evening dress, and works best on an academic conference poster."

Prompt:
This typeface sits at the Transitional/Songti stage with a serif-based regular structure, medium-high contrast and Inktrap refinements; visually it has elongated, open counters and sharp terminals, conveying an academic, rational, and dignified mood — like a professor in evening dress, best suited for an academic conference poster.

D Inputs and Prompt Template

- **Input**: Each font is rendered as an 8×12 grid (512×512 pixels) containing 52 Latin and 50

Table 6: Representative SVG primitives and path commands.

Command	Parameters	Description
`<rect>` Rectangle	`x, y` `width, height`	Rectangle with position and dimensions.
`<circle>` Circle	`cx, cy` `r`	Circle with center coordinates and radius.
`<ellipse>` Ellipse	`cx, cy` `rx, ry`	Ellipse with radii along the x- and y-axes.
`<line>` Line	`x1, y1` `x2, y2`	Straight line segment between two points.
`M` MoveTo	`x, y`	Path: Move current point to (x,y).
`L` LineTo	`x, y`	Path: Straight line to (x,y).
`C` Cubic Bézier	`x1, y1` `x2, y2` `x, y`	Path: Cubic Bézier curve with two control points and an endpoint.
`A` Arc	`rx, ry` `rotation, flags` `x, y`	Path: Arc to endpoint (x,y) with specified radii and flags.

representative native glyphs.

- **Prompt Template** (one-sentence structured description):

> Below is a typeface image. Describe it in ONE sentence using: "This typeface sits at [History] with [Serif/Script] structure, [Contrast] contrast, [Tech] features, and conveys a [Mood] mood; visually it has [Look-adj] strokes and feels [Feel-adj], like [Analogy]."

Here, `[Look-adj]` denotes the *visual form* adjective (e.g., "angular," "flowing"), `[Feel-adj]` denotes the *emotional aura* adjective (e.g., "playful," "solemn"), and `[Analogy]` represents a metaphorical scene description (e.g., "like signage at a lantern festival"). This structured template enforces consistency while enriching semantic diversity.

E Reproducibility Details

To facilitate community adoption and ensure exact reproducibility, we detail our training configuration and special token definitions.

Hyperparameters: The backbone is Qwen2.5-VL-3B-Instruct. We used the AdamW optimizer with a learning rate of 1×10^{-4} and linear warm-up for the first 30,000 steps (10% of total). The model was trained for 300,000 steps with a global batch size of 640 across 8 NVIDIA L40S GPUs. Data sampling ratio was set to 65% Text-to-SVG and 35% Raster-to-SVG.

Special Tokens Specification: To efficiently model SVG geometry, we expanded the default tokenizer with exactly 2,050 special tokens:

- **Coordinate Bins (1001 tokens):** `<bin_0>` through `<bin_1000>` representing the normalized $[0, 1000]$ canvas.
- **Command Syntax ($\sim$20 tokens):** Standard SVG commands including `<path>`, `</path>`, `<M>`, `<L>`, `<C>`, `<Z>`, etc.
- **Structural Markers:** `<|start|>` and `<|end|>` for sequence control.

Note that semantic markers in the prompt (such as `[CHRON:]`, `[MOOD:]`) are processed as standard text patterns by the LLM tokenizer and do not require additional vocabulary allocation.

F Raw XML Visualization and Interpretability

Raw SVG code is often not human-interpretable for assessing visual quality. For example, general-purpose LLMs like GPT-4o frequently generate output that appears numerically similar but contains fatal syntax errors (e.g., missing closing

quotes `Z"/>`), rendering the entire glyph invalid (Failure Rate > 0).

In contrast, our structured tokenization enforces syntactically valid XML with a 0% failure rate. Even when coordinate predictions differ slightly from the ground truth (± 5 units), these variations are visually imperceptible but heavily penalized by raw text comparison metrics, which validates our choice of relying on vector-native metrics (CD, SI) rather than raw string matching. (Side-by-side XML comparisons will be included in the code repository).

Below is a simplified side-by-side comparison illustrating why raw text evaluation is flawed. General-purpose models often generate visually destructive syntax errors, whereas our tokenization explicitly prevents them:

Ground Truth XML:
```
<path d="M 150 300 C 150 400 200
450 250 450 Z"/>
```
Vector Calligrapher (Ours): (Valid syntax, minor coordinate shift)
```
<path d="M 152 298 C 148 402 198
448 252 452 Z"/>
```
GPT-4o Baseline: (Fatal syntax error: missing closing quotes)
```
<path d="M 150 300 C 150 400 200
450 250 450 Z/>
```

G Data vs. Architecture: Re-trained OmniSVG

To further illustrate the complementary roles of the dataset and the architecture (as discussed in Section 6.3.1), we compared visualizations of the re-trained OmniSVG model alongside Vector Calligrapher (see Figure 4). While both models yield visually comparable fidelity at a standard 512×512 evaluation resolution (verifying that the **VectorFont dataset** sets a high baseline for generation quality), the re-trained OmniSVG relies on flattened tokenization. As analyzed in Appendix A, this flattened architecture cannot be scaled to professional print resolutions (4096×4096) due to OOM errors, a limitation that our **separated-coordinate architecture** strictly overcomes.

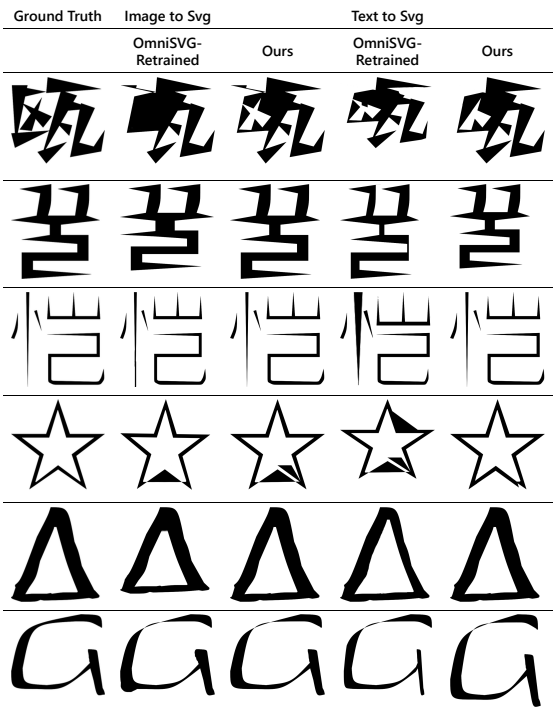

Figure 4: Comparison of Re-trained OmniSVG and Vector Calligrapher.

H More Results

H.1 Other Visual and Geometric Metrics

H.2 Raster to SVG

H.3 Prompt Based Generation

Table 7: **More quantitative results**. Most of the visual metrics could not fully reflect the quality of vector graphics.

Methods	Prompt-based Generation				Raster-to-SVG		
	SSIM↑	Aesthetic↑	LPIPS↑	MSE↓	CD↓	N_{cmd} ↓	SI↓
SVGen	0.5	2.38	0.48	0.34	-	-	-
GPT-4o	0.58	1.85	0.4	0.33	0.025	9.26	56.5
Gemini-3-pro	0.59	2.23	0.39	0.31	0.07	26.1	63.5
KIMI-K2	0.61	2.03	0.39	0.30	0.03	40.5	58.2
OmniSVG (Re-train)	0.73	2.08	0.26	0.2	0.005	33	7.8
Vector Calligrapher (ours)	0.76	2.09	0.21	0.15	0.001	29.3	5.9

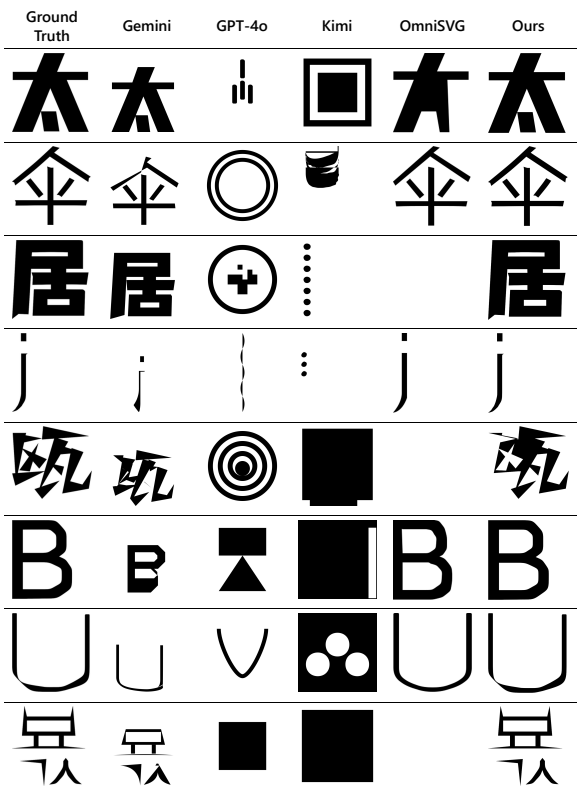

Figure 5: Raster to SVG Visualization.

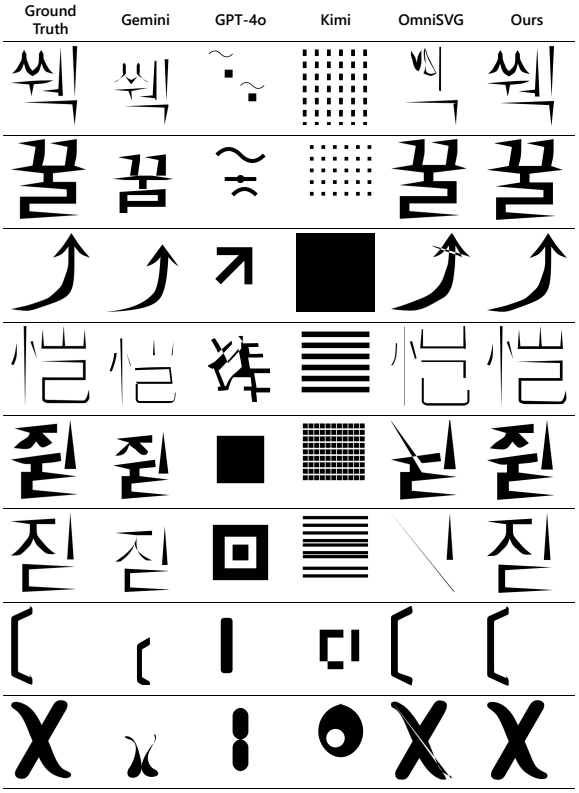

Figure 6: Raster to SVG Visualization.

Table 8: Font description for Prompt Based Generation Samples (Structured).

index	Description
0	The typeface sits at a [CHRON: modern intersection], with a [STRUCT: bold] structure and [CONTRAST: high] contrast, featuring [FORM: sharp features, strong strokes]. It conveys a [MOOD: dynamic, assertive] character, and can be described as [SCENE: a bold statement in a bustling cityscape].
1	The typeface sits at a [CHRON: modern era], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [FORM: sharp features, thick strokes]. It conveys a [MOOD: bold, strong] character, like [SCENE: a sturdy building].
2	The typeface sits at a [CHRON: modern digital history], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [FORM: geometric features, sharp angular strokes]. It conveys a [MOOD: bold, striking] character, like [SCENE: a futuristic robot].
3	The typeface sits at the [CHRON: intersection of modern and traditional design], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [FORM: sharp features, strong strokes]. It conveys a [MOOD: bold, authoritative] character, like [SCENE: a commanding presence in a digital landscape].
4	The typeface sits at the [CHRON: intersection of traditional and modern design], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [FORM: sharp features, precise strokes]. It conveys a [MOOD: bold, authoritative] character, like [SCENE: a confident leader giving a decisive speech].
5	The typeface sits at the [CHRON: intersection of modern and traditional], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, incorporating [TECH: advanced technical features]. It conveys a [MOOD: bold, authoritative] character, like [SCENE: a commanding presence in an advertising campaign].
6	The typeface sits in a [CHRON: modern advertising context], with a [STRUCT: bold, blocky] structure and [CONTRAST: high] contrast, featuring [FORM: clean features, thick uniform strokes]. It conveys a [MOOD: strong, assertive] character, like [SCENE: a sturdy building block].
7	The typeface sits at the [CHRON: intersection of modern and traditional], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [FORM: geometric features, sharp precise strokes]. It conveys a [MOOD: bold, authoritative] character, like [SCENE: a fortress].
8	The typeface sits at the [CHRON: intersection of modern design and traditional Chinese calligraphy], with a [STRUCT: bold] structure and [CONTRAST: high] contrast, featuring [FORM: sharp features, strong angular strokes]. It conveys a [MOOD: dynamic, powerful] character, like [SCENE: a skilled calligrapher wielding a brush with precision].
9	The typeface sits at a [CHRON: modern digital history], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, incorporating [TECH: advanced technical features]. It conveys a [MOOD: bold, assertive] character, like [SCENE: a sleek high-tech advertisement].
10	The typeface sits at the [CHRON: intersection of modern and traditional], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [TECH: digital features]. It conveys a [MOOD: bold, dynamic] character, like [SCENE: a fusion of ancient calligraphy and digital art].
11	The typeface sits at the [CHRON: intersection of traditional and modern], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, incorporating [TECH: advanced technical features]. It conveys a [MOOD: sophisticated, authoritative] character, like [SCENE: a classic yet contemporary advertisement].
12	The typeface sits at the [CHRON: intersection of traditional and modern], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, incorporating [TECH: advanced technical features]. It conveys a [MOOD: sophisticated, authoritative] character, like [SCENE: a classic yet contemporary advertisement].
13	The typeface sits at the [CHRON: intersection of modern and traditional], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [FORM: sharp features, strong strokes]. It conveys a [MOOD: bold, commanding] character, like [SCENE: a commanding officer issuing orders].
14	The typeface sits at the [CHRON: intersection of modern and traditional], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, featuring [FORM: sharp features, strong strokes]. It conveys a [MOOD: bold, commanding] character, like [SCENE: a commanding officer issuing orders].
15	The typeface sits at the [CHRON: traditional Chinese calligraphy with a modern twist], with a [STRUCT: bold, structured] design and [CONTRAST: high] contrast, incorporating [TECH: advanced technical features]. It conveys a [MOOD: strong, authoritative] character, like [SCENE: a general leading an army].
16	The typeface sits at a [CHRON: modern intersection], with a [STRUCT: bold, sans-serif] structure and [CONTRAST: high] contrast, featuring [FORM: sharp, precise, clean angular strokes]. It conveys a [MOOD: strong, authoritative] character, like [SCENE: a commanding presence in a bustling cityscape].
17	The typeface sits at a [CHRON: modern intersection], with a [STRUCT: bold, geometric] structure and [CONTRAST: high] contrast, featuring [FORM: sharp angular features, thick uniform strokes]. It conveys a [MOOD: strong, assertive] character, like [SCENE: a graphic poster design].
18	The typeface sits at a [CHRON: modern intersection], with a [STRUCT: bold, geometric] structure and [CONTRAST: high] contrast, featuring [FORM: sharp angular features, thick uniform strokes]. It conveys a [MOOD: strong, assertive] character, like [SCENE: a graphic poster design].
19	The typeface sits in a [CHRON: modern advertising context], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, incorporating [TECH: advanced technical features]. It conveys a [MOOD: bold, strong] character, like [SCENE: a fortress wall].
20	The typeface sits at the [CHRON: intersection of modern and traditional design], with a [STRUCT: Serif] structure and [CONTRAST: high] contrast, incorporating [TECH: advanced technical features]. It conveys a [MOOD: bold, dynamic] character, like [SCENE: a fusion of classic elegance and contemporary energy].

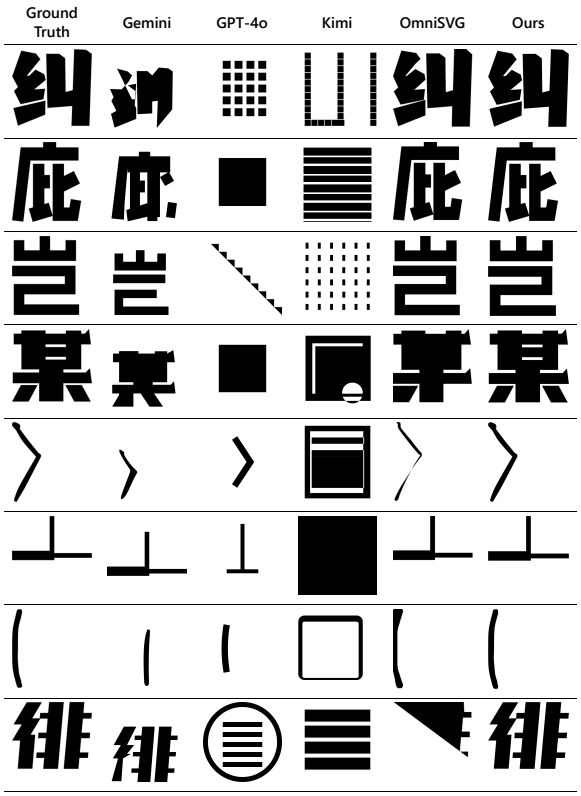

Figure 7: Raster to SVG Visualization.

Figure 8: Prompt Based Generation.

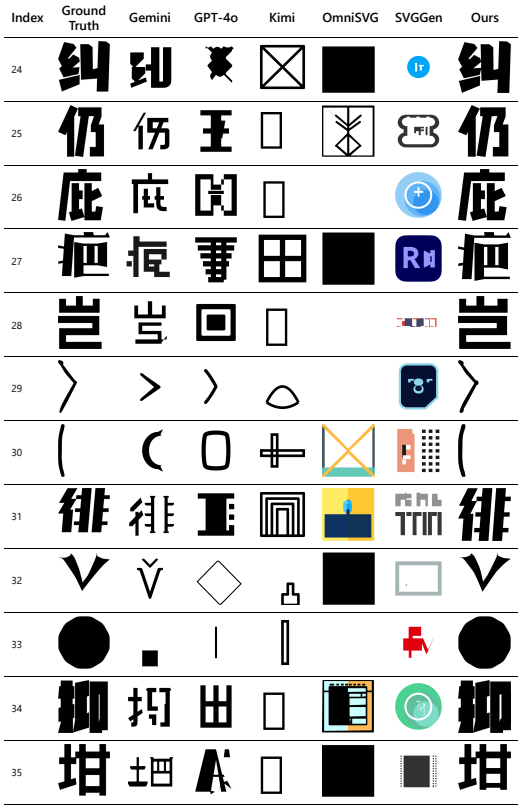

Figure 9: Prompt Based Generation.