FontCopilot: Towards Generalist Multimodal Large Language Models for Holistic Chinese Font Engineering

Yu Liu[1,2], Yang Liu[1], Ying Gao[1], Yang Ding[3], and Cunrui Wang[1,2] *

[1] Dalian Chinese Font Design Technology Innovation Center, Dalian Minzu University, 116600 Dalian, China
ethanliuyu@foxmail.com,{202412052003,202412052007}@stu.dlnu.edu.cn
[2] Key Laboratory of Education Informatization for Nationalities, (Yunnan Normal University), Ministry of Education, Kunming 650500, China
wcr@dlnu.edu.cn
[3] School of Computer Science and Technology, Changchun University of Science and Technology, 130022 Changchun, China
2023200170@mails.cust.edu.cn

Abstract. Chinese font production is a complex task that balances visual consistency, vector editability, and topological integrity at a massive character scale. Existing fragmented toolchains, relying on manual rules, often lead to geometric redundancy and defects. This paper introduces FontCopilot, a multimodal large language model that performs raster-to-vector conversion, Bézier curve sparsification, defect marking, and geometric correction within a single model. We elevate font engineering from explicit geometric rule constraints to structured glyph code understanding and rewriting, serializing Bézier curves into SVG XML. This enables the model to perform topological reorganization in the code space and internalize implicit designer priors as structured reasoning constraints. FontCopilot integrates few-shot reference guidance and zero-shot generalization mechanisms, balancing specific style transfer and universal standard reconstruction. To advance the development of a unified font engineering model, we constructed the CFCopilot-8M dataset, covering 300 font types and 8 million glyph samples. Under progressive multi-task hybrid training, the glyphs processed by FontCopilot show consistent improvements in editability, contour compactness, and defect correction, validating its industrial-grade font production value.

Keywords: Chinese Glyph Optimization · Glyph Image Vectorization · Glyph Defect Detection · Scalable Vector Graphics· Multimodal Large Language Models

1 Introduction

Designing Chinese character libraries covering tens of thousands of glyphs is a complex system engineering task that combines artistic creativity and engineer-

* Corresponding author.

ing constraints. In the process of converting design drafts into final products, steps such as vectorization, defect detection, defect repair, and contour normalization must maintain visual consistency while ensuring the precision and simplicity of geometric topology. However, due to the massive character data and the discrepancies arising from multi-person collaboration, font engineering has encountered bottlenecks in both efficiency and quality. To address these challenges, integrating intelligent automation systems into the design loop has become a necessary condition for achieving efficient font engineering.

Faced with the continuously growing demand for character sets, Chinese font generation has become an important direction for supporting large-scale character set automation. Existing research often models this as image translation, leading to two main approaches: one uses paired data for end-to-end supervised learning to synthesize font images [22, 37, 47]; the other adopts a few-shot paradigm by decoupling style and content, introducing self/unsupervised mechanisms, and generalizing to unseen styles with minimal reference glyphs [20,28,36]. However, the outputs of these methods often remain in the pixel-domain raster glyphs, which differ from the Bézier curve vector fonts required for font production. To meet the requirements for lossless scaling and editability, bitmap images must be further converted into Bézier parameterized vector outlines [40]. Existing methods for this conversion are mainly divided into traditional edge-tracing and curve-fitting approaches [5, 9, 12, 32], and sequence generation methods that directly predict coordinate sequences [21, 23, 24, 41]. However, in real font library production, both manual and automated vectorization often fail to meet delivery standards. Vector outlines frequently contain redundant subpaths and overly dense control points, raising editing and rendering costs; at the same time, they may exhibit local geometric and visual defects, making it difficult to meet the stringent requirements of industrial-grade font production.

Eliminating redundant paths and topological defects in the engineering practice faces a dual dilemma: on one hand, the vast character sets make manual verification extremely inefficient [38]; on the other hand, defect determination is highly dependent on font style, and traditional mathematical models based on explicit rules struggle to learn clear decision boundaries [22]. Existing methods often reduce the problem to low-dimensional geometric fitting and reparameterization based on error metrics [9], which makes it difficult to explicitly incorporate expert-level design priors. When dealing with complex strokes and stylized details, these methods tend to fall into semantic blind spots, resulting in new redundancies and defects [7, 30]. Therefore, the ideal solution requires a paradigm shift from "passive geometric fitting" to "active semantic inference" of design intentions. Multimodal large language models (MLLMs), with their ability to learn implicit structural priors from massive data and their powerful sequence inference capabilities at the code level [4, 26, 31, 39, 44], provide a new technical route for jointly handling redundancy compression and defect determination within a unified framework.

To address the issue of vector redundancy and quality defects in Chinese font engineering, as well as the industry's demand for high-standard editable vec-

tor glyphs, this paper proposes FontCopilot, a unified Chinese font engineering framework based on MLLM. This framework discards the pipeline-style post-processing relying on explicit rules and local error metrics, instead integrating vectorization, contour compression, defect detection, and geometric correction into sequence modeling, achieving a paradigm shift from "passive geometric constraints" to "active intent inference." To fit complex font engineering tasks into a unified sequence modeling paradigm, we reconstruct glyph contours defined by Bézier curves into declarative SVG XML code. This representation serves as an intermediary bridge between visual perception and geometric logic, enabling deep semantic alignment between the pixel space and the parameter space, while also granting the model the ability to perform topological reorganization directly at the code level. The proposed method offers flexibility: on one hand, by borrowing from the concept of few-shot learning, the model can adaptively transfer and normalize specific styles by perceiving topological hints from a small number of standard samples; on the other hand, in a zero-shot setting, the model leverages internalized implicit design priors to perform end-to-end glyph refinement and industrial-grade normalization. To support these capabilities, we constructed the CFCopilot-8M dataset, which covers 300 fonts and contains over 8 million samples, enabling the model to absorb industrial-grade design priors fully and ultimately achieve high-precision automatic conversion from raster images to production-grade vector glyphs. This demonstrates the immense potential of integrating with professional Chinese font design workflows. The contributions of this paper are summarized as follows:

- Proposed FontCopilot, a multitask font engineering framework based on MLLM, which unifies Chinese glyph vectorization, vector optimization, defect detection, and geometric correction into code-space structured understanding and rewriting tasks, enabling multitask collaboration in a single model for font production. The code is public at: `https://github.com/UnqiueGG/FontCopilot`
- By normalizing Bézier curve contours into declarative SVG XML code representations, the processing of Chinese glyphs is unified as a code understanding and generation task executable by MLLM, allowing the model to perform topological-level editing and inject design prior constraints in the code space.
- The proposed framework can extract specific style topological features with a small number of reference samples and, in a zero-shot setting, rely on internalized expert-level design priors for autonomous inference, enhancing adaptability in complex industrial environments.
- Constructed the CFCopilot-8M dataset, which contains 300 fonts and 8 million professional-grade glyph samples, providing scalable data support for the model to learn professional design priors and convert them into implicit constraints.

2 Related Work

2.1 Chinese Font Generation

Creating a Chinese font often requires designers to manually draw glyphs for tens of thousands of characters, which is both costly and time-consuming [38]. Font generation methods, by learning the mapping from source fonts to target fonts, can synthesize large-scale character set font images with only a small number of reference glyphs, significantly reducing manual effort. Existing research employs various neural network architectures to learn the content and style representations of glyphs and their mappings, including CNN [33], RNN [34], Transformer [19, 27], GAN [11, 29], and Diffusion Model [20, 25, 45], enabling high-quality glyph synthesis and automated production on a large character set scale. However, the outputs of existing Chinese font generation methods are often limited to pixel-based raster images, while standard digital fonts are typically represented by Bézier curve parameterized vector outlines to meet the requirements of lossless scaling and editability, thus restricting the applicability of these methods.

2.2 Vector Font Reconstruction

Vector fonts encode glyph contours as editable Bézier path topologies, enabling resolution-independent rendering and production-ready standardization. Currently, the field of vector font reconstruction has primarily developed two main paradigms: optimization-based methods and learning-based methods. Optimization-based methods, relying on differentiable rasterization techniques [15], iteratively adjust control points to fit the glyph contours [18, 30, 40, 43]. Although these methods do not require labeled data, they often produce excessive redundant nodes, leading to overly complex paths. In contrast, learning-based methods predict the parameters of Bézier curves through end-to-end learning [23, 35, 41], which offers significant advantages in modeling complex structures. However, existing reconstruction methods often produce overly dense control points and redundant segments, increasing storage, rendering, and editing costs, and still require substantial manual cleanup to fix geometric defects.

2.3 Multimodal Large Language Models

MLLMs provide general representations and inference capabilities for cross-modal understanding and generation tasks by integrating visual perception and language reasoning within a unified sequence modeling framework. Closed-source models such as GPT-4V [1] and Gemini [8] demonstrate exceptional reasoning abilities, showcasing the superiority of MLLMs in reasoning tasks. Meanwhile, the release of open-source models like BLIP-2 [14], LLaVA [16,17], and Qwen [3] has advanced MLLM from general visual question answering to domain-specific structured generation, enabling complex visual tasks to be uniformly expressed

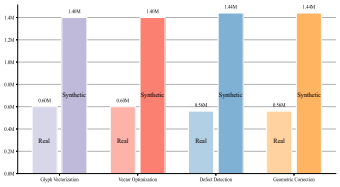

Fig. 1: Overall real and synthetic data composition across the four tasks in CFCopilot-8M.

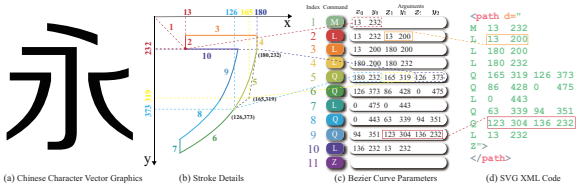

Fig. 2: (a) Chinese vector glyphs. (b) Stroke-level topological structure. (c) Bézier curve control points and parameter representation. (d) Serialized SVG XML code.

as controllable code generation and rewriting processes. We serialize Bézier contours into SVG XML code and unify glyph vectorization, sparsification, defect marking, and geometric correction into understanding and rewriting tasks within the code space, thus laying the foundation for a single model framework that covers multiple tasks in font production.

3 CFCopilot-8M Datasets

3.1 Datasets

We build CFCopilot-8M, a large-scale vector glyph dataset with 300 fonts and 8M samples for training and evaluation. As shown in Fig. 1, each task contains 2M samples. We adopt a two-stream data construction: 1) Design-stream data collected from designers' manual drawing processes; 2) Algorithm-stream data synthesized by generative models. The construction pipeline of CFCopilot-8M combines manually drafted patterns with algorithmically generated samples, enabling the resulting data distribution to effectively cover the diverse vectorization challenges encountered in real-world font engineering.

3.2 Structured Glyph Representation

As shown in Fig. 2, to bridge the modal gap between 2D geometric topology and 1D language sequences, we represent glyphs drawn with Bézier curves as SVG XML sequences that can be modeled by MLLMs. SVG expresses both local geometry and global topology simultaneously through "drawing instructions + geometric parameters," allowing vectorization, optimization, and correction in font engineering to be unified as the understanding and rewriting of structured code. Specifically, we linearize the glyph contour into an ordered drawing sequence and encode movement, line segments, curves, and closure operations with the instruction set $\mathcal{V} = \{M, L, H, V, C, Q, Z\}$, transforming implicit topological constraints into stable "instruction–parameter" dependencies. This facilitates MLLM's inference of control point layouts and subpath organization in long-range contexts.

4 Font Copilot

4.1 Method Overview

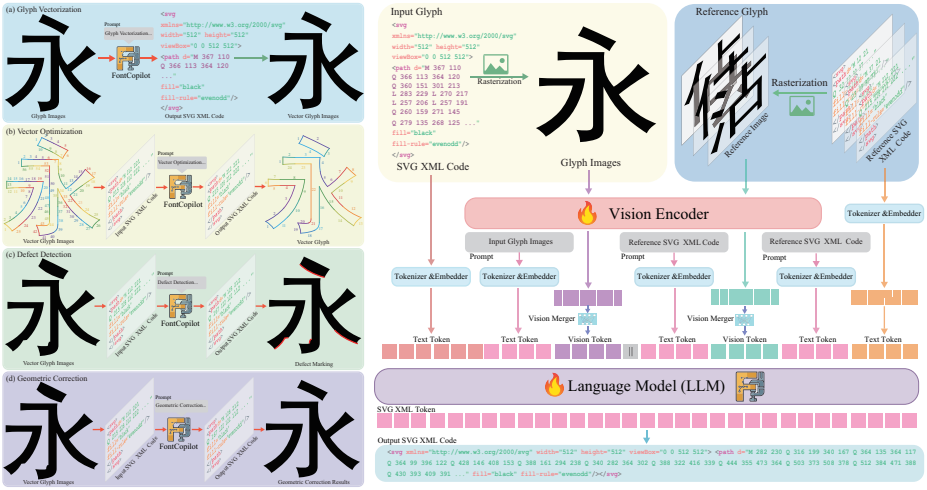

Fig. 3: FontCopilot is a multitask font engineering framework that supports glyph vectorization, vector optimization, defect detection, and geometric correction. The symbol || denotes sequence concatenation, which serves to support switching between these two modes.

As shown in Fig. 2, FontCopilot proposes a unified MLLM framework for Chinese font production, with SVG XML code as the core intermediate representation. This framework transforms font engineering from a "rule-driven post-processing pipeline" into "structured understanding and rewriting in code space" (Sec. 3). Given a glyph image I and an input SVG XML code S, the model autoregressively outputs the target sequence $S^{\star}$ under the task instruction T, thereby covering vectorization, vector optimization, defect marking, and geometric correction within the same paradigm (Sec. 4.2). To meet both internal font consistency and cross-font generalization, we design two inference paradigms: Few-shot Reference-Guided and Zero-shot Universal (Sec. 4.3). The model structure consists of a native resolution visual encoder and a language model with strong code capabilities, and stable long-range SVG generation is achieved through visual token injection and multi-layer feature infusion (Sec.4.4). The training adopts a progressive multi-stage strategy to unify alignment, rewriting, and multi-task instruction learning (Sec.4.5).

4.2 Task Definitions

Glyph Vectorization. End-to-end and few-shot Chinese font generation methods have already synthesized high-quality raster glyphs, but the deliverable

glyphs are editable vector outlines defined by Bézier curves. To address this, we train FontCopilot to learn the end-to-end mapping from glyph images to SVG XML, treating vectorization as a structured code generation task (Fig. 3 (a)). Given a raster glyph image $I \in \mathbb{R}^{H\times W\times 3}$, the model generates an SVG sequence $S = t_1, \ldots, t_L$, such that the rendered result $R(S)$ approximates I and satisfies the Bézier topology constraints. We extract visual features Z_v as prefix tokens injected into the MLLM, and autoregressively minimize the negative log-likelihood:

$$\mathcal{L}_{vec} = -\sum_{i=1}^{L} \log P_\theta(t_i \mid I, t_{<i}; \theta). \tag{1}$$

Unlike pixel-wise fitting, FontCopilot directly infers stroke trajectories and node distributions from visual features.

Vector Optimization. Manually or automatically vectorized glyphs often contain redundant paths and overly dense control points, reducing editability and rendering efficiency. In the vector font optimization task, directly labeling redundant control points is costly, and the optimal topology relies on expert intuition, making it difficult to quantify using rules. Therefore, we model vector optimization as a structured rewriting task in the SVG code space, allowing the model to learn the semantic compression process from non-standard to standard forms (Fig. 3 (b)). Specifically, given a non-standard sequence $S^{nc} = \{t_1, \ldots, t_L\}$, the model generates a more compact and standardized sequence $S^c = \{t'_1, \ldots, t'_{L'}\}$. During training, we use autoregressive supervision to minimize the negative log-likelihood:

$$\mathcal{L}_{opt} = -\sum_{i=1}^{L'} \log P_\theta(t'_i \mid S^{nc}, I, \mathcal{R}, t'_{<i}; \theta), \tag{2}$$

where I is the rendered image that constrains appearance consistency, and $\mathcal{R} = \{(I_r, S_r)\}_{k=1}^{K}$ consists of a small number of reference characters from the same font to provide stroke organization and topological preferences.

Defect Detection. Chinese font defects exhibit high variability and style dependence, making them difficult to define exhaustively using discrete categories. At the same time, industrial scenarios are more concerned with "where the defect occurs in the contour" for quick subsequent corrections. Therefore, we frame defect detection as a structured marking task in the SVG code space (Fig. 3 (c)). Given a glyph image I and its corresponding SVG sequence S, the model outputs an SVG sequence with defect markings $\hat{S}^d$, which locates and annotates defects while keeping the original geometry unchanged. During training, autoregressive conditional likelihood supervision is used, minimizing:

$$\mathcal{L}_{det} = -\sum_{i=1}^{L_d} \log P_\theta(\hat{t}_i \mid I, S, \mathcal{R}, \hat{t}_{<i}; \theta), \tag{3}$$

where $\hat{S}^d = \{\hat{t}_1, \ldots, \hat{t}_{L_d}\}$ is the output sequence with defect markings. Since defect markings are encoded at the SVG contour level, this representation establishes a direct correspondence between "defect semantics" and "contour po-

sitions" while preserving geometric accuracy, allowing the model to achieve fine-grained localization through structured reasoning at the code level.

Geometric Correction. In addition to geometric redundancy, vector contours often contain issues such as sudden stroke width changes, local breaks, or unsmooth bends, which require correction of control points. We model geometric correction as an SVG code rewriting task. Given a sequence S to be corrected and its rendered image I, the model generates a corrected sequence S^r that eliminates local geometric anomalies while maintaining overall appearance and style consistency (Fig. 3 (d)). Specifically, we use the marked sequence $\hat{S}^d$ obtained from defect detection as explicit correction cues and introduce a small number of reference characters from the same font, $\mathcal{R}$, to constrain the strokes. During training, the ground truth sequence S^{gt} serves as supervision, and we minimize the autoregressive conditional likelihood:

$$\mathcal{L}_{rect} = -\sum_{i=1}^{L_r} \log P_\theta(t_i^r \mid I, S, \hat{S}^d, \mathcal{R}, t_{<i}^r; \theta), \tag{4}$$

where $S^r = \{t_1^r, \ldots, t_{L_r}^r\}$. Since correction directly occurs at the SVG code level, the model can adjust control points, smooth curvature, and perform local reparameterization on relevant segments under the constraint of localization cues.

4.3 Refinement Paradigms

To adapt to both specific font local normalization preferences and cross-font universal engineering standards, we propose two complementary inference paradigms:

Few-shot Reference-Guided. In font engineering, contour topology and parameterization are largely driven by designer priors. Although these priors are hard to formalize, they exhibit stable within-font regularities. To inject such within-font consistency into inference, we sample a small set of high-quality, normalized glyphs from the target font S as a reference set $\mathcal{R}_s = \{(I_r^{(k)}, S_r^{(k)})\}_{k=1}^K$. For any input (I, S^{nc}), the model generates a canonical code under the reference context, i.e., $S^* \sim P_\theta(S^* \mid I,\, S^{nc},\, \mathcal{R}_s)$. Guided by these few-shot exemplars, the model aligns to font-specific engineering preferences, performs topology reorganization and control-point sparsification in SVG code space, and preserves both visual fidelity and within-font consistency.

Zero-shot Universal. For large-scale unseen fonts, we use Zero-shot Universal inference. Trained on the multi-font, multi-task CFCopilot-8M supervision, the model learns generic glyph structure and engineering norms as an inductive bias over SVG code. At inference, no reference set is needed; given the glyph image I and input SVG S, it outputs a canonical sequence $S^* \sim P_\theta(S^* \mid I,\, S)$. The model rewrites the SVG to remove redundant paths, sparsify dense control points, and stabilize topology, producing industrial-grade editable geometry.

4.4 Architecture

Dynamic Resolution. In Chinese font engineering, character structural details are highly dependent on pixel-level information. When the input is scaled or

forced downsampled, it can easily distort the original aspect ratio and spatial structure of the glyph, introducing geometric deformation. Therefore, we adopt a visual encoder design that supports native resolution. Given a Chinese glyph image $I \in \mathbb{R}^{H\times W\times 3}$, where $m, n \in \mathbb{N}$ and satisfy $H = mP$, $W = nP$, we divide I into $m \times n$ non-overlapping patches of fixed size $P \times P$. Let the (i, j)-th patch be:

$$I_{i,j} = I\big[(i-1)P + 1 : iP,\ (j-1)P + 1 : jP,\ :\big] \in \mathbb{R}^{P\times P\times 3}, \tag{5}$$

where $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$. We flatten each patch and perform a linear transformation to obtain the patch embedding:

$$\mathbf{e}_{i,j} = W_e \,\mathrm{vec}(I_{i,j}) + b_e \in \mathbb{R}^{D_v}. \tag{6}$$

We flatten all patches into a sequence of length $L_v = mn$, denoted as $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_{L_v}\}$, and inject 2D positional information using 2D RoPE to maintain the spatial topology of the strokes. Since H, W are variable and no forced scaling is applied, this design maximizes the preservation of the original aspect ratio and spatial structure.

Vision Encoder. The visual encoder consists of 16 Transformer blocks. In the first 6 layers, we use progressive multi-scale window attention with $(w_1, w_2, w_3) = (2, 4, 8)$, progressively expanding the local receptive field. A shifted window strategy between adjacent blocks is used for cross-window information interaction, enhancing local geometric detail perception and capturing stroke boundaries and local deformations. In the deeper layers, we switch to window attention with $w_4 = 16$, and global attention is inserted in layers {7, 10, 13, 16} to model long-range dependencies across strokes. The sequence E is input into the vision encoder, producing the visual token representation: $Z_v \in \mathbb{R}^{L_v\times D_v}$.

Visual Token Aggregation. To accommodate the input length limitations of the LLM and reduce sequence redundancy, we rearrange Z_v into a 2D grid $\mathcal{F} \in \mathbb{R}^{m\times n\times D_v}$, and perform spatial aggregation on adjacent 2×2 neighborhoods. The aggregated features are defined as:

$$v_{i,j} = \big[\mathcal{F}_{2i-1,\,2j-1} \ \oplus\ \mathcal{F}_{2i,\,2j-1} \ \oplus\ \mathcal{F}_{2i-1,\,2j} \ \oplus\ \mathcal{F}_{2i,\,2j}\big], \tag{7}$$

where $\oplus$ denotes vector concatenation, $i \in \left\{1, \ldots, \frac{m}{2}\right\}$,$j \in \left\{1, \ldots, \frac{n}{2}\right\}$. We then use a two-layer MLP to map the aggregated features into the embedding space of the LLM:

$$v'_{i,j} = W_2\big(\sigma(W_1\mathbf{v}_{i,j} + b_1)\big) + b_2 \in \mathbb{R}^D. \tag{8}$$

This process reduces the visual sequence length to $L'_v = \frac{mn}{4} = \frac{HW}{4P^2}$, significantly lowering the number of visual tokens while preserving semantic and geometric structural information, making it more efficient for injection as prefix tokens into the MLLM.

Language Modeling. To accomplish multiple tasks such as glyph vectorization, contour sparsification, defect marking, and geometric correction within a unified framework, we choose Qwen-2.5-Coder-7B [13] as the backbone language

model and represent the font engineering process as a conditional generation and structured rewriting of SVG XML code sequences.

For any sample to be processed, we organize the input into a unified multimodal conditional context X, which contains three complementary pieces of information: 1) the visual token representation Z_v obtained from the glyph image I through the visual encoder; 2) the SVG sequence S corresponding to the current vector description of the glyph; 3) the few-shot reference set $\mathcal{R} = \{(I_r^{(k)}, S_r^{(k)})\}_{k=1}^{K}$, which explicitly injects topological organization preferences and parameterization styles of the same font. We concatenate the above information into a context sequence according to a fixed template, with visual tokens injected as prefixes into the input embedding space of the language model. The language model autoregressively models its generation distribution under the conditional context X:

$$P_\theta(S^\star \mid X) = \prod_{i=1}^{L^\star} P_\theta(t_i \mid t_{<i},\, X)\,, \quad t_{<i} = \{t_1, \ldots, t_{i-1}\}. \tag{9}$$

Multi-Scale Feature Projection. During long-range autoregressive generation of SVG code, the model may encounter issues such as the weakening of geometric information and the decay of reference constraints across layers. To mitigate these degradations, we introduce a multi-scale feature projection and cross-layer injection mechanism. Visual geometric features are injected into the language model's hidden states at multiple decoding layers, enhancing fine-grained structural fidelity and improving topological consistency under reference guidance. Specifically, we extract multi-scale features $F^{(k)}$ from the visual encoder and project them into the language model's hidden space using a two-layer MLP:

$$V^{(k)} = g^{(k)}(F^{(k)}) \in \mathbb{R}^{L_v' \times D}, \quad k \in \{1, 2, 3\}, \tag{10}$$

where D is the embedding dimension of the language model. Let the decoder's l-th layer hidden state be $H^{(l)}$, and the visual injection position index set be $\mathcal{I}_{\text{vis}}$. We perform residual injection at predefined layers $\mathcal{L}_{\text{inj}} = \{l_1, l_2, l_3\}$ on the visual positions:

$$\tilde{H}^{(l_k)}_{\mathcal{I}_{\text{vis}}} = H^{(l_k)}_{\mathcal{I}_{\text{vis}}} + V^{(k)}, \quad H^{(l_k+1)} = \mathcal{B}^{(l_k)}(\tilde{H}^{(l_k)}). \tag{11}$$

This cross-modal residual connection allows the model to repeatedly refer back to the geometric details of the glyph image and the structure of reference characters during decoding, without increasing the sequence length.

4.5 Training Strategy

We use a three-stage training scheme to unify multi-task font engineering and support both zero-shot and few-shot inference. **Stage 1** freezes the language model and trains the vision stack to align glyph appearance with SVG topology; **Stage 2** unfreezes the language model to rewrite non-canonical SVG into canonical SVG. **Stage 3** applies multi-task instruction tuning with reference dropout and a complexity curriculum to improve cross-font generalization.

4.6 Data Augmentation

To improve generalization and mitigate overfitting, we apply a shape-invariant SVG code augmentation strategy: 1) Curve Splitting: Randomly dividing Bézier curves to introduce parameter redundancy; 2) Start-point Randomization: Cyclically shifting closed-path commands to eliminate start-point bias; 3) Subpath Reordering: Shuffling independent contours to decouple writing order and focus on essential topology.

5 Experiments

5.1 Baselines and Evaluation Metrics

Baselines. As baselines from the graphics community, we adopt Potrace [32] and Adobe Image Trace (AIT) [2], which represent the practical upper bound of current non-learning-based vectorization tools. In addition, to evaluate the adaptability of general-purpose vision–language models to specialized vector glyph tasks, we instruction-tune a series of SOTA MLLMs. The baseline models include LLaVA-V1.6-7B [17], Yi-VL-6B [46], DeepSeek-VL-7B [26], and the Qwen-VL family (Qwen2.5-VL-3B [4], Qwen3-VL-8B [3]).
Evaluation Metrics. We evaluate FontCopilot using a multi-perspective framework that integrates raster-based visual metrics (IoU, MSE [42]), geometric structural metrics (CD [10], DTW [48]), and the efficiency metric Reduction Ratio (RR).

5.2 Comparison with Baselines

Table 1: Quantitative comparison across four tasks: (a) glyph vectorization, (b) vector optimization, (c) defect detection, and (d) geometric correction.

Method	(a) Glyph Vectorization				(b) Vector Optimization					(c) Defect Detection				(d) Geometric Correction			
	IoU ↑	MSE ↓	CD ↓	DTW ↓	IoU ↑	MSE ↓	CD ↓	DTW ↓	RR (%) ↑	IoU ↑	MSE ↓	CD ↓	DTW ↓	IoU ↑	MSE ↓	CD ↓	DTW ↓
Potrace	0.978	0.009	0.0075	4.821	–	–	–	–	–	–	–	–	–	–	–	–	–
AIT	**0.982**	**0.004**	**0.0041**	3.945	–	–	–	–	–	–	–	–	–	–	–	–	–
StarVector-8B	0.752	0.095	0.1582	4.125	0.812	0.072	0.1458	3.522	64.21	0.725	0.068	0.8149	3.1152	0.655	0.115	0.2150	5.125
DeepSeek-VL-7B	0.768	0.088	0.1425	3.854	0.835	0.065	0.1245	3.104	65.80	0.742	0.062	0.7316	2.8883	0.682	0.108	0.1945	4.854
Qwen2.5-VL-3B	0.795	0.072	0.1154	3.450	0.864	0.048	0.0892	2.551	68.45	0.785	0.053	0.5977	2.6256	0.715	0.092	0.1654	4.250
Yi-VL-6B	0.812	0.065	0.0985	3.105	0.881	0.042	0.0754	2.218	69.12	0.815	0.045	0.4612	2.3586	0.742	0.085	0.1425	3.815
LLaVA-V1.6-7B	0.835	0.054	0.0812	2.742	0.895	0.036	0.0621	1.954	70.05	0.842	0.035	0.3393	1.9833	0.768	0.072	0.1152	3.454
Qwen3-VL-8B	0.865	0.042	0.0645	2.154	0.923	0.021	0.0385	1.525	71.88	0.875	0.020	0.1312	1.3981	0.825	0.055	0.0845	2.652
FontCopilot	**0.925**	**0.028**	**0.0355**	**1.352**	**0.965**	**0.018**	**0.0158**	**1.142**	**76.52**	**0.938**	**0.011**	**0.0519**	**1.009**	**0.908**	**0.032**	**0.0455**	**1.452**

Quantitative Evaluation. For glyph vectorization, Tab. 1 (a) reveals that graphics-based methods fail to effectively simplify glyph structures, instead introducing significant geometric redundancy. For vector optimization, Tab. 1 (b) reflects that while general MLLM methods can achieve Bézier curve sparsification, they often result in significant coordinate drift and topological distortion. In

contrast, FontCopilot achieves extremely high node compression rates while generating vector glyphs that precisely fit the target ground truth. Furthermore, for defect detection, Tab. 1 (c) validates FontCopilot's advantage in capturing subtle topological anomalies, significantly outperforming general multimodal models that struggle to precisely locate fine-grained features. In the geometric correction task, Tab. 1 (d) shows that, compared to the structural distortions often generated by general models during reconstruction, our method effectively fixes local geometric defects while maintaining high vector quality and more accurately restoring complex glyph structures.

Fig. 4: Qualitative comparison of FontCopilot against state-of-the-art MLLMs across four key typography tasks.

Qualitative Evaluation. Fig. 4 presents a qualitative comparison between FontCopilot and current mainstream MLLMs across four core font engineering tasks. In vectorization and vector optimization, baseline models often produce topological distortions, while our method generates smooth contours with both high visual fidelity and high node compression rates, thanks to structured code inference. In defect detection and geometric correction, general multimodal models struggle to accurately locate fine-grained anomalies and often disrupt the original stroke structure during repairs. In contrast, FontCopilot leverages internalized design priors to achieve precise localization and local reparameterization, effectively eliminating geometric anomalies while fully preserving the structural features of the original glyphs, meeting stringent industrial delivery standards.

Human Evaluation. We conducted a blind study with 20 professional type designers on 100 randomly sampled, topologically diverse Chinese characters. Participants rated randomized outputs on a 5-point Likert scale under four task-specific criteria: Raster-to-Vector Accuracy (vectorization), Editability & Topology Compactness (optimization), Defect Localization Quality (detection), and Style-Consistent Geometry Rectification (correction). To quantify practical efficiency, we further recorded the Fix Time (seconds) required to manually refine each output to production standards.

Table 2: Comprehensive human evaluation across all tasks.

Method	Task-specific Quality				Time (s) ↓
	V↑	O↑	D↑	C↑	
Manual	–	–	–	–	184.1
Potrace	3.92	1.35	–	–	295.2
AIT	4.15	2.05	–	–	182.4
StarVector-8B	2.88	2.52	1.55	1.88	112.4
DeepSeek-VL-7B	3.05	2.65	1.92	2.05	108.2
Qwen2.5-VL-3B	3.15	2.85	2.45	2.45	88.8
Yi-VL-6B	3.28	2.92	2.88	2.75	72.5
LLaVA-V1.6-7B	3.32	3.12	2.28	2.55	55.5
Qwen3-VL-8B	3.55	3.68	3.10	2.85	45.4
FontCopilot	**4.12**	**4.18**	**4.22**	**4.15**	**24.5**
Ground Truth	*4.65*	*4.52*	*4.61*	*4.65*	–

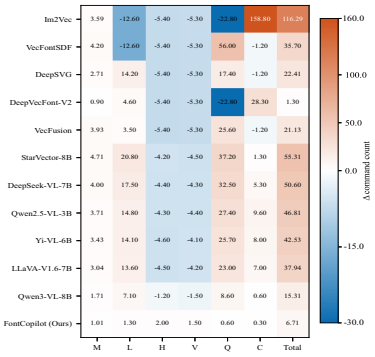

Fig. 5: Difference in SVG command counts (Δ) compared to human designs.

As shown in Tab. 2, classical graphics-based methods achieve higher raster-to-vector accuracy (V) but score poorly on editability and topology compactness (O) due to the lack of structural semantics and production constraints. General-purpose MLLMs exhibit some reasoning capability, yet their limited fine-grained geometric control hampers defect localization (D) and style-consistent geometry rectification (C), leading to longer fix time. In contrast, FontCopilot attains the best scores across all four criteria and reduces the per-glyph fix time to 24.5s, demonstrating strong suitability for production-grade deployment.

5.3 Analysis

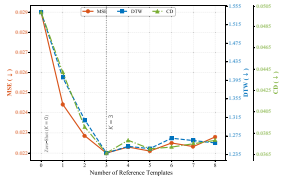

Fig. 6: Impact of the reference template count (K) on geometric error metrics.

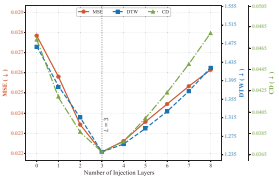

Fig. 7: Impact of visual-injection layer count (L) on geometric error metrics.

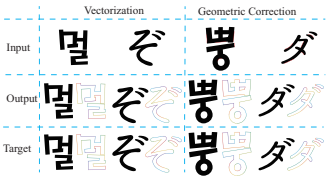

Fig. 8: Cross language generalization of vectorization and geometric correction.

Topological Compactness. Fig. 5 shows the difference (Δ) between the number of SVG commands generated by different glyph image vectorization methods and human-designed ground truth. Traditional graphics methods, including Im2Vec [30], VecFontSDF [43], DeepSVG [6], DeepVecFont [40], DeepVecFont-v2 [41], VecFusion [35], rely on stacking numerous curves to passively fit edges,

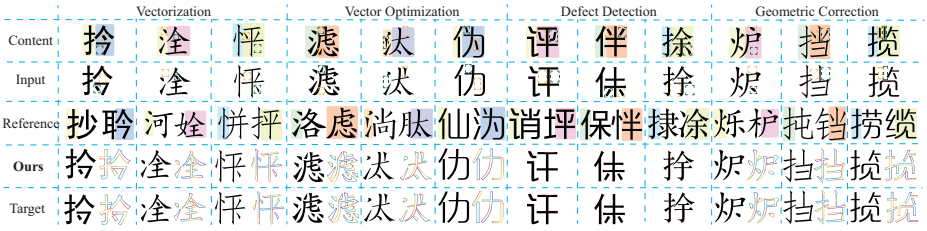

Fig. 9: Qualitative generalization study on manually constructed pseudo-characters.

Fig. 10: Component consistency across characters. Identical components preserve the same vector structure.

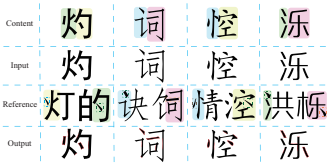

Fig. 11: Stroke-style conflict defect detection results.Erroneous strokes are flagged as defects.

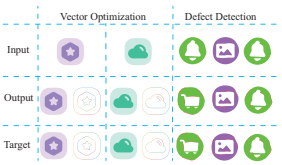

Fig. 12: Generalization results on vector optimization and defect detection for icons.

leading to significant geometric redundancy. While general multimodal large language models have basic coding capabilities, they lack specific topological constraints, which often results in unnecessary path segmentation. Thanks to the internalization of expert-level design priors, FontCopilot minimizes redundancy in all types of commands.

Reference Count Sensitivity. We evaluated the model's performance for $K \in \{0, 1, \ldots, 8\}$. Here, $K = 0$ corresponds to the Zero-Shot Universal baseline model with no reference input. As shown in Fig. 6, the model reaches a global minimum at $K = 3$, exhibiting the lowest topological distortion and reconstruction error. However, as the number of reference samples increases further, the model's performance does not yield additional benefits, and instead, metrics begin to plateau and show slight fluctuations.

Effectiveness of Multi-Layer Visual Injection. To evaluate the effectiveness of the multi-scale visual feature injection mechanism, we conducted experiments with varying numbers of injection layers $L \in \{0, \ldots, 8\}$. As shown in Fig. 7, the performance exhibits a quadratic trend with increasing L, reaching a global optimum at $L = 3$. The experiments demonstrate that moderate feature injection enables the most robust alignment between visual structural priors and SVG parameterized output, without compromising the native code reasoning capabilities of the large language model.

Generalization on Pseudo-Characters. We create pseudo-characters by adding or removing strokes and evaluate generalization. As shown in Fig. 9, FontCopilot generalizes to unseen topologies across all four tasks.

Component Consistency. As shown in Fig. 10, the generated results are geometrically consistent across characters, with control-point layouts aligned to the reference glyph.
Style Sensitivity. Fig. 11 shows that, in defect detection, FontCopilot is strongly guided by style cues and flags strokes that conflict with the reference glyph as defects.
Generalization. We evaluate glyph vectorization and geometric correction on Korean and Japanese scripts; as shown in Fig. 8, FontCopilot generalizes to unseen writing systems. We further test vector optimization and defect detection on vector icons; Fig. 12 shows that its learned geometric priors transfer effectively to general planar vector graphics.

6 Conclusion

This paper presents FontCopilot, the first framework to unify glyph vectorization, Bézier contour sparsification, defect marking, and geometric correction in Chinese font production into a single multimodal large language model. We consolidate pipeline processes relying on explicit geometric rules into the understanding and rewriting of structured glyph code, while combining few-shot reference guidance and zero-shot general optimization to achieve conditional adaptation and robust generalization for target fonts. Based on the constructed CFCopilot-8M, extensive experiments demonstrate its practical value for serving industrial-grade font production. **Limitation**: Since the model is partially trained on algorithmically generated data, its learned priors may still be affected by synthetic data distributions and may require further validation on more diverse real-world font engineering scenarios. **Negative Impact:** Misuse of the proposed framework may lead to unauthorized imitation or reconstruction of commercial fonts, potentially infringing upon the rights and commercial interests of font designers and rights holders.

Disclaimer

The CFCopilot-8M dataset collected and used in this paper is intended solely for academic research and non-commercial purposes. All copyrights, trademarks, and other intellectual property rights associated with the original content remain the property of their respective rights holders.

Acknowledgements

This study was supported in part by the Xingliao Talent Program under Grant XLYCE2504027, the Liaoning Provincial Science and Technology Plan Joint Program (Technology R&D Program Project) under Grant 2024JH2/102600108, and the Foundation of the Key Laboratory of Education Informatization for Nationalities (Yunnan Normal University), Ministry of Education, under Grant EIN2024B002.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023). https://doi.org/10.48550/arXiv.2303.08774
2. Adobe Inc.: Image Trace Panel Options. Adobe (2025), https://helpx.adobe.com/illustrator/using/image-trace.html, accessed: 2026-01-19
3. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025). https://doi.org/10.48550/arXiv.2511.21631
4. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J.: Qwen2.5-vl technical report (2025). https://doi.org/10.48550/arXiv.2502.13923
5. Bessmeltsev, M., Solomon, J.: Vectorization of line drawings via polyvector fields. ACM Trans. Graph. **38**(1), 1–12 (2019). https://doi.org/10.1145/3202661
6. Carlier, A., Danelljan, M., Alahi, A., Timofte, R.: Deepsvg: A hierarchical generative network for vector graphics animation. In: Adv. Neural Inform. Process. Syst. vol. 33, pp. 16351–16361 (2020)
7. Chin, H.Y., Shen, I.C., Chiu, Y.T., Shamir, A., Chen, B.Y.: Autosketch: Vlm-assisted style-aware vector sketch completion. In: Proceedings of the SIGGRAPH Asia 2025 Conference Papers. pp. 1–11 (2025). https://doi.org/10.1145/3757377.3763828
8. Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al.: Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261 (2025). https://doi.org/10.48550/arXiv.2507.06261
9. Dominici, E.A., Schertler, N., Griffin, J., Hoshyari, S., Sigal, L., Sheffer, A.: Polyfit: Perception-aligned vectorization of raster clip-art via intermediate polygonal fitting. ACM Trans. Graph. **39**(4), 77–1 (2020). https://doi.org/10.1145/3386569.3392401
10. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 2463–2471 (2017). https://doi.org/10.1109/cvpr.2017.264
11. Guo, X., Zhao, L., Wu, X., Yang, H., Man, Y.: A chinese font generation method based on dynamic convolution improved generative adversarial network. In: 2024 International Conference on Culture-Oriented Science & Technology (CoST). pp. 12–17. IEEE (2024). https://doi.org/10.1109/CoST64302.2024.00012
12. Hoshyari, S., Dominici, E.A., Sheffer, A., Carr, N., Wang, Z., Ceylan, D., Shen, I.C.: Perception-driven semi-structured boundary vectorization. ACM Trans. Graph. **37**(4), 1–14 (2018). https://doi.org/10.1145/3197517.3201312
13. Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Lu, K., et al.: Qwen2. 5-coder technical report. arXiv preprint arXiv:2409.12186 (2024). https://doi.org/10.48550/arXiv.2409.12186
14. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: Int. Conf. Mach. Learn. pp. 19730–19742. PMLR (2023)

15. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Trans. Graph. **39**(6), 1–15 (2020). https://doi.org/10.1145/3414685.3417871
16. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 26286–26296 (2024). https://doi.org/10.1109/cvpr52733.2024.02484
17. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: Adv. Neural Inform. Process. Syst. vol. 36, pp. 34892–34916 (2023)
18. Liu, Y.T., Zhang, Z., Guo, Y.C., Fisher, M., Wang, Z., Zhang, S.H.: Dualvector: Unsupervised vector font synthesis with dual-part representation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 14193–14202 (2023). https://doi.org/10.1109/CVPR52729.2023.01364
19. Liu, Y., Lian, Z.: Fonttransformer: Few-shot high-resolution chinese glyph image synthesis via stacked transformers. Pattern Recognition **141**, 109593 (2023). https://doi.org/10.1016/j.patcog.2023.109593
20. Liu, Y., Ding, Y., Khalid, F.B., Wang, C., Wang, L.: Few-shot font generation via denoising diffusion and component-level fine-grained style. Expert Systems with Applications **296**, 128987 (2026). https://doi.org/10.1016/j.eswa.2025.128987
21. Liu, Y., binti Khalid, F., binti Mustaffa, M.R., bin Azman, A.: Dual-modality learning and transformer-based approach for high-quality vector font generation. Expert Systems with Applications **240**, 122405 (2024). https://doi.org/10.1016/j.eswa.2023.122405
22. Liu, Y., binti Khalid, F., Wang, C., binti Mustaffa, M.R., bin Azman, A.: An end-to-end chinese font generation network with stroke semantics and deformable attention skip-connection. Expert Systems with Applications **237**, 121407 (2024). https://doi.org/10.1016/j.eswa.2023.121407
23. Liu, Y., Khalid, F.B., Wang, C., Mustaffa, M.R.B., Azman, A.B.: Diffvecfont: fusing dual-mode reconstruction vector fonts via masked diffusion transformers. In: International Conference on Computational Visual Media. pp. 339–363. Springer (2025). https://doi.org/10.1007/978-981-96-5812-1_18
24. Liu, Y., Khalid, F.B., Wang, L., Zhang, Y., Wang, C.: Elegantly written: Disentangling writer and character styles for enhancing online chinese handwriting. In: Eur. Conf. Comput. Vis. pp. 409–425. Springer (2024). https://doi.org/10.1007/978-3-031-73242-3_23
25. Liu, Y., Wang, C., Feng, L., Zhang, J., Lu, B.: Gracefully air-written: Enhancing the legibility and style consistency of in-air handwriting. In: AAAI. vol. 40, pp. 17598–17607 (2026). https://doi.org/10.1609/aaai.v40i21.38815
26. Lu, H., Liu, W., Zhang, B., Wang, B., Dong, K., Liu, B., Sun, J., Ren, T., Li, Z., Yang, H., et al.: Deepseek-vl: towards real-world vision-language understanding. arXiv preprint arXiv:2403.05525 (2024). https://doi.org/10.48550/arXiv.2403.05525
27. Ma, J., Xiang, X., He, Y.: Masking cascaded self-attentions for few-shot font-generation transformer. In: Asian Conf. Comput. Vis. pp. 256–272 (2024). https://doi.org/10.1007/978-981-96-0917-8_15
28. Pan, W., Zhu, A., Zhou, X., Iwana, B.K., Li, S.: Few shot font generation via transferring similarity guided global style and quantization local style. In: Int. Conf. Comput. Vis. pp. 19449–19459 (2023). https://doi.org/10.1109/iccv51070.2023.01787

29. Park, J., Hassan, A.U., Choi, J.: Ccfont: Component-based chinese font generation model using generative adversarial networks (gans). Applied Sciences **12**(16), 8005 (2022). https://doi.org/10.3390/app12168005
30. Reddy, P., Gharbi, M., Lukac, M., Mitra, N.J.: Im2vec: Synthesizing vector graphics without vector supervision. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 7342–7351 (2021). https://doi.org/10.1109/cvpr46437.2021.00726
31. Rodriguez, J.A., Puri, A., Agarwal, S., Laradji, I.H., Rodriguez, P., Rajeswar, S., Vazquez, D., Pal, C., Pedersoli, M.: Starvector: Generating scalable vector graphics code from images and text. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 16175–16186 (2025). https://doi.org/10.1109/cvpr52734.2025.01508
32. Selinger, P.: Potrace: Transform bitmaps into vector graphics. http://potrace.sourceforge.net (2003), accessed: 2025-01-01
33. Sun, D., Ren, T., Li, C., Su, H., Zhu, J.: Learning to write stylized chinese characters by reading a handful of examples. In: IJCAI. pp. 920–927. ijcai.org (2018). https://doi.org/10.24963/IJCAI.2018/128
34. Tang, S., Xia, Z., Lian, Z., Tang, Y., Xiao, J.: Fontrnn: Generating large-scale chinese fonts via recurrent neural network. Comput. Graph. Forum **38**(7), 567–577 (2019). https://doi.org/10.1111/cgf.13861
35. Thamizharasan, V., Liu, D., Agarwal, S., Fisher, M., Gharbi, M., Wang, O., Jacobson, A., Kalogerakis, E.: Vecfusion: Vector font generation with diffusion. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 7943–7952 (2024). https://doi.org/10.1109/cvpr52733.2024.00759
36. Wang, C., Zhou, M., Ge, T., Jiang, Y., Bao, H., Xu, W.: Cf-font: Content fusion for few-shot font generation. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1858–1867 (2023). https://doi.org/10.1109/cvpr52729.2023.00185
37. Wang, C., Ding, Y., Liu, Y., Zhan, G., Li, Z.: Chinese font generation from stroke semantic and attention mechanism. Journal of Computer-Aided Design & Computer Graphics **34**(8), 1229–1237 (2022). https://doi.org/10.3724/SP.J.1089.2022.19125
38. Wang, L., Liu, Y., Sharum, M.Y., Yaakob, R., Kasmiran, K.A., Wang, C.: Deep learning for chinese font generation: A survey. Expert Systems with Applications **276**, 127105 (2025). https://doi.org/10.1016/j.eswa.2025.127105
39. Wang, Q., He, J., Pan, Y., Yeo, S.Y., Yang, X., Li, S.: Monosr: Open-vocabulary spatial reasoning from monocular images. arXiv preprint arXiv:2511.19119 (2025). https://doi.org/10.48550/arXiv.2511.19119
40. Wang, Y., Lian, Z.: Deepvecfont: synthesizing high-quality vector fonts via dual-modality learning. ACM Trans. Graph. **40**(6), 1–15 (2021). https://doi.org/10.1145/3478513.3480488
41. Wang, Y., Wang, Y., Yu, L., Zhu, Y., Lian, Z.: Deepvecfont-v2: Exploiting transformers to synthesize vector fonts with higher quality. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 18320–18328 (2023). https://doi.org/10.1109/cvpr52729.2023.01757
42. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004). https://doi.org/10.1109/tip.2003.819861
43. Xia, Z., Xiong, B., Lian, Z.: Vecfontsdf: Learning to reconstruct and synthesize high-quality vector fonts via signed distance functions. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1848–1857 (2023). https://doi.org/10.1109/cvpr52729.2023.00184

44. Xing, X., Hu, J., Liang, G., Zhang, J., Xu, D., Yu, Q.: Empowering llms to understand and generate complex vector graphics. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 19487–19497 (2025). https://doi.org/10.1109/cvpr52734.2025.01815
45. Yang, Z., Peng, D., Kong, Y., Zhang, Y., Yao, C., Jin, L.: Fontdiffuser: One-shot font generation via denoising diffusion with multi-scale content aggregation and style contrastive learning. In: AAAI. vol. 38, pp. 6603–6611 (2024). https://doi.org/10.1609/aaai.v38i7.28482
46. Young, A., Chen, B., Li, C., Huang, C., Zhang, G., Zhang, G., Wang, G., Li, H., Zhu, J., Chen, J., et al.: Yi: Open foundation models by 01. ai. arXiv preprint arXiv:2403.04652 (2024). https://doi.org/10.48550/arXiv.2403.04652
47. Zeng, J., Chen, Q., Liu, Y., Wang, M., Yao, Y.: Strokegan: Reducing mode collapse in chinese font generation via stroke encoding. In: AAAI. vol. 35, pp. 3270–3277 (2021). https://doi.org/10.1609/aaai.v35i4.16438
48. Zhu, Y., Shasha, D.: Warping indexes with envelope transforms for query by humming. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. pp. 181–192 (2003). https://doi.org/10.1145/872757.872780