Stroke2Font: A Hierarchical Vector Model with AI-Driven Optimization for Chinese Font Generation

Qing-sheng Li *, Yu-lin Bian , Zhen-hui Chai

Posted Date: 10 February 2026

doi: 10.20944/preprints202602.0798.v1

Keywords: Chinese font generation; hierarchical vector model; stroke element representation; AI-driven optimization; Bézier curve parameterization; complexity-aware optimization

Article

Stroke2Font: A Hierarchical Vector Model with AI-Driven Optimization for Chinese Font Generation

Qing-sheng Li [1,*], Yu-lin Bian [2] and Zhen-hui Chai [2]

[1] School of Media and Information Engineering, Communication University of Zhejiang, China

[2] School of Communication and Journalism, Communication University of Zhejiang, China

* Correspondence: liqingsheng@cuz.edu.cn; Tel.: (086-15058199136)

Abstract

Chinese font generation plays a crucial role in digital typography, cultural preservation, and personalized user interfaces. However, existing methods often face challenges in maintaining structural consistency, supporting diverse stylistic variations, and achieving computational efficiency simultaneously, especially in cloud-based environments. To address these issues, this paper proposes Stroke2Font—a hierarchical vector model with AI-driven optimization for dynamic Chinese font generation. The core model decouples structural representation from style rendering through stroke-element decomposition and Bézier curve parameterization. To further balance structural fidelity, style diversity, and real-time performance, we introduce a three-layer optimization framework: (1) a reinforcement learning policy for dynamic selection of Bézier control parameters to minimize rendering latency; (2) a genetic algorithm for exploring style vector spaces and generating novel font variants; and (3) a cloud-aware dynamic resource allocation model that ensures scalability under concurrent user requests. Experimental results on a dataset of 150 Chinese characters with 1,123 stroke trajectories and 5,287 feature points demonstrate that the adaptive complexity-aware optimization achieves the highest trajectory similarity of 65.2%, representing a 7.1% improvement over baseline methods (60.9%). The evaluation covers characters ranging from 1 to 18 strokes across 6 stroke types, with standard deviation reduced to ±7.5% (compared to ±8.2% baseline), indicating more consistent performance. Quantitative analysis confirms that the method generalizes effectively across varying character complexity, with the optimization showing stable improvement regardless of stroke count distribution. These results validate that Stroke2Font provides an effective solution for high-quality, efficient, and scalable Chinese font generation in cloud-based applications.

Keywords: Chinese font generation; hierarchical vector model; stroke element representation; AI-driven optimization; Bézier curve parameterization; complexity-aware optimization

1. Introduction

Chinese font generation represents a fundamental challenge in digital typography, cultural preservation, and human-computer interaction [1]. With over 50,000 distinct characters in the Unicode CJK Unified Ideographs block and countless stylistic variations across calligraphic traditions, the computational synthesis of Chinese glyphs demands approaches that can simultaneously respect structural constraints, capture aesthetic qualities, and scale efficiently to large character sets [2,3]. With the proliferation of mobile devices and social media, there is a growing demand for customized and stylized Chinese fonts that reflect individual taste and contextual appropriateness [4].

However, the creation of high-quality Chinese fonts remains a labor-intensive and time-consuming process. A standard Chinese font library contains thousands of glyphs (e.g., 6,763 in the GB2312 standard), each requiring meticulous design, stroke adjustment, and aesthetic harmonization [5]. Although numerous font libraries are commercially available, the need for novel, personalized, and context-specific typefaces continues to outpace supply [6]. Moreover, traditional font files are

 doi:10.20944/preprints202602.0798.v1

monolithic in nature—each style requires a complete set of glyph outlines, leading to significant storage and transmission overhead, especially in cloud and web environments [7].

In recent years, computational approaches to font generation have emerged, broadly categorized into explicit feature-based methods and deep learning-based methods. Early feature-based techniques decompose characters into strokes, radicals, or structural templates, then reassemble them through rule-based or optimization-driven processes [8,9,10]. While offering interpretability and control, these methods often require extensive manual annotation and struggle to capture nuanced stylistic variations. Conversely, deep learning methods, particularly those based on Generative Adversarial Networks (GANs) [11–14], learn font styles in an end-to-end manner from image pairs. Although capable of producing visually appealing results, they frequently suffer from mode collapse, stroke discontinuity, and limited structural fidelity, especially when training data are scarce or styles are highly divergent [15,16].

This fundamental trade-off between structural fidelity and stylistic expressiveness mirrors a core challenge in computer vision and generative modeling: the disentanglement of content (structure) [17] from style (appearance)[18,39]. Pioneering works in neural style transfer [19,20] and image-to-image translation [21] demonstrated the separation and recombination of these factors, yet they operate primarily at the pixel level, lacking the explicit, hierarchical semantics required for structured graphical objects like characters. Recent advances in object-centric representation learning [22,40] and neural vector graphics [23,24] have made strides toward structured and editable visual generation. However, directly applying these paradigms to Chinese characters—with their unparalleled topological complexity and vast glyph set—remains an open problem, as they do not natively incorporate the linguistic and graphic design priors inherent to font engineering.

Consequently, a truly effective font generation system must simultaneously address three intertwined challenges: (1) preserving the structural integrity of complex hieroglyphic compositions; (2) enabling rich and nuanced stylistic flexibility; and (3) achieving high computational efficiency for real-time, cloud-native applications. Most existing methods excel in at most one of these dimensions: feature-based approaches prioritize structure at the cost of style diversity and automation; deep learning models capture style but often falter on structure and efficiency, while few consider the system-level performance required for scalable deployment.

To holistically address this tripartite challenge, we present Stroke2Font, a novel framework that unifies a hierarchical vector-based core model with an AI-driven optimization layer. Our core model provides the foundational control over structure and style, while the optimization layer intelligently automates parameter selection, explores the design space, and manages resources—ensuring the system is not only high-quality but also efficient and scalable. The contributions of this paper are summarized as follows:

(1) We propose a hierarchical stroke-element representation that decouples structure from style, enabling explicit control over glyph topology and local stroke attributes.

(2) We develop a vector-based style manipulation framework using yoke vectors and Bézier fitting, allowing continuous interpolation between font styles without retraining.

(3) We introduce an AI-driven optimization framework consisting of: a reinforcement learning agent for real-time rendering optimization; a genetic algorithm for style space exploration; and a cloud-aware resource allocation model for scalable service delivery.

(4) We implement a cloud-compatible font service architecture that significantly reduces bandwidth and storage requirements while supporting dynamic, personalized font generation.

(5) We conduct extensive experiments demonstrating that Stroke2Font outperforms existing approaches in structural accuracy, style diversity, and computational efficiency, particularly under real-time and concurrent access scenarios.

The structure of this paper is organized as follows.

Section 2 reviews related work, classifying existing approaches into explicit feature-based methods, deep learning-based methods, and hybrid structure-aware techniques, and clarifies the positioning of our work.

 doi:10.20944/preprints202602.0798.v1

Section 3 details the proposed Stroke2Font framework. It begins with an overview (**3.1**) and introduces our hierarchical representation for structure and style (**3.2**), including character-level abstraction, stroke-level feature extraction, and the modeling of temporal, stylistic, and personalized handwriting variations. We then present the stroke-element representation as a hierarchical vector model (**3.4**) and describe the parametric style generation mechanism via vector manipulation (**3.5**).

Section 4 presents our AI-driven optimization framework, outlining its three-layer architecture—reinforcement learning for parameter selection, genetic algorithm for style exploration, and cloud-aware resource allocation—along with adaptive complexity-aware optimization strategies.

Section 5 provides extensive experimental evaluation, covering structural reconstruction, stylistic generation and interpolation, and the performance of the AI-driven optimization module.

Finally, **Section 6** concludes the paper and discusses potential future directions.

2. Related Work

The automatic generation of Chinese fonts has been studied for decades, with approaches evolving from rule-based systems to data-driven deep learning models. In this section, we review existing work from three perspectives: (1) explicit feature-based methods, (2) deep learning-based methods, and (3) hybrid or structure-aware approaches.

2.1. Explicit Feature-Based Methods

Early research in Chinese font generation focused on explicit structural decomposition and geometric modeling. These methods typically extract low-level features such as strokes, radicals, and topological relations, then reassemble them through parametric or procedural means.

Stroke-based synthesis is one of the most intuitive strategies. For example, Pang and Feng [25] proposed a method that segments Chinese characters into strokes, builds a stroke library, and reconstructs new glyphs by rearranging stroke outlines—though it requires extensive manual post-processing. Similarly, Yang et al. [6] employed C-Bézier curves to model stroke contours, allowing style variation through control point adjustment. While effective for regular typefaces, these methods struggle with cursive or highly stylized fonts where stroke boundaries are ambiguous.

Radical-based assembly offers a higher-level compositional approach. Lin et al. [7] introduced a greedy algorithm that decomposes a small set of sample characters into radicals, then recombines them to form unseen characters. This approach reduces the number of required samples but depends heavily on the completeness and compatibility of the radical library.

Although feature-based methods provide interpretability and precise control, they suffer from several limitations: (1) they rely on handcrafted feature extraction, which is labor-intensive and error-prone; (2) they often fail to capture subtle style nuances, especially in handwritten or calligraphic fonts; (3) they lack flexibility in generating entirely new styles without redesigning stroke or radical templates.

2.2. Deep Learning-Based Methods

With the advent of deep learning, especially Generative Adversarial Networks (GANs) [9], Chinese font generation has shifted toward data-driven, end-to-end frameworks, as extensively reviewed in recent literature [13]. These methods treat font generation as an image-to-image translation task [26], learning mappings from source style to target style without explicit structural modeling.

Pix2pix-based approaches laid the groundwork for style transfer in Chinese fonts. Tian proposed zi2zi [27], a conditional GAN that translates standard printed characters into stylized ones. While demonstrating impressive results, such methods often produce artifacts such as broken strokes or blurred contours when structural differences between source and target are large.

 doi:10.20944/preprints202602.0798.v1

To improve structural consistency, later works introduced auxiliary structural cues. Jiang et al. [28] proposed SCFont, which integrates stroke-level semantics and structural constraints into a deep stacked network, guiding the generation of target-style skeleton images to preserve glyph topology. Wu et al. [29] introduced CalliGAN, a structure-aware framework that incorporates stroke sequence information derived from a Chinese character decomposition system to calibrate fine-grained styles and maintain structural integrity. These methods show improved robustness but still struggle with mode collapse and style inconsistency in fine-grained details.

Building on this structure-aware direction, recent advances have focused on enhancing controllability and reducing data dependency. Wang et al. [30] proposed Attribute2Font, which leverages font attributes and a semi-supervised attention mechanism for controllable style synthesis, offering a step towards more interpretable generation. Others have tackled the data bottleneck; for instance, Jiang et al. [31] designed W-Net, a one-shot generation network that learns style from a single character reference. However, despite these improvements, a common weakness persists: the inherent black-box nature of deep learning approaches still limits fine-grained, user-intuitive controllability and often necessitates large, paired datasets for robust training. These limitations have prompted a parallel line of research that seeks to fundamentally redefine the representation and generation paradigm itself, moving beyond pixel-based translation towards more explicit, structured, and editable models.

2.3. Hybrid and Structure-Aware Approaches

Recognizing the complementary strengths of explicit features and deep learning, and driven by the need to overcome the limitations of pixel-based methods, several studies have explored hybrid models that integrate structural priors with neural networks.

Graph-based representations have been used to encode topological relations between strokes. For example, Wen et al. [19] proposed ZiGAN, which uses a few-shot style transfer approach guided by stroke-level graphs. Similarly, Ning [32] combined self-attention mechanisms with StarGAN v2 for multi-style migration, showing improved style disentanglement.

Vector-based generative models represent another promising direction. Recent work in vector graphics generation, such as DeepSVG [33] and Im2Vec [34], has inspired font generation methods that operate directly on parametric curves rather than pixels. These approaches naturally support resolution-independent rendering and editable outputs, though they have not been extensively applied to Chinese characters due to their structural complexity. Our proposed Stroke2Font framework is positioned within this vector-based paradigm, aiming to address the unique challenges of Chinese characters by introducing a hierarchical stroke-element representation and a continuous style space, thereby enabling both structural fidelity and rich stylistic control.

2.4. Our Positioning

Our work is situated at the intersection of explicit structural modeling and parametric style control. Unlike pure deep learning methods, we adopt a hierarchical stroke-element representation that explicitly captures glyph topology. Unlike traditional feature-based methods, we use vector operations and Bézier curves to enable continuous style interpolation without manual redesign. Compared to recent hybrid approaches, our model is lighter, more interpretable, and cloud-native, making it suitable for real-time and personalized font services.

While existing methods have made progress in font style transfer, they often neglect explicit structural modeling and lack support for efficient cloud-based generation. To address these limitations, we propose a stroke-element-based model that decouples structure from style through a hierarchical vector representation. The details of our approach are presented in the following section.

3. Proposed Model

3.1. Overview of the Framework

 doi:10.20944/preprints202602.0798.v1

The core idea of our model is to decouple structural representation from stylistic rendering in Chinese font generation. As illustrated in Figure 1, our framework follows a hierarchical pipeline: characters are first decomposed into stroke elements, which are represented as vectors; these vectors are then stylized via Bézier curve fitting; finally, the styled strokes are reassembled according to structural templates for output.

The model operates in two phases:

Structure Encoding: A character is abstracted into a stroke-element graph, stored as a lightweight template.

Style Rendering:

Given style parameters, stroke elements are deformed via vector operations and rendered as Bézier curves.

This separation enables efficient cloud-based font services, where structural templates reside on the server and style parameters are transmitted on-demand to clients.

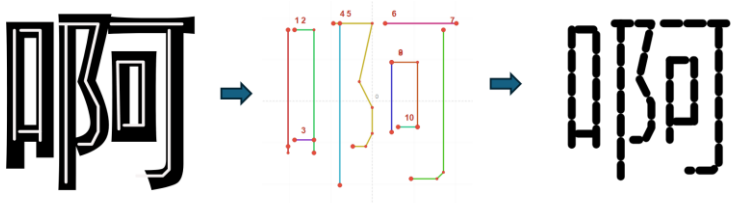


Figure 1. Decomposition of the Chinese character '啊' into stroke elements and style groups.

3.2. Hierarchical Representation of Structure and Style

Chinese characters exhibit a well-defined hierarchical composition: characters are composed of radicals, which in turn consist of basic strokes arranged according to established spatial conventions. We formalize this hierarchy through three complementary abstraction processes:

3.2.1. Character-Level Structural Abstraction

Following the principle of separation of structure and style established in Section 3.1, we extract invariant topological skeletons from diverse font instances. Figure 2 demonstrates how our stroke-element representation captures both structural invariance and stylistic variation. Two renderings of '福'—one in geometric Heiti font, another in expressive Xingkai script—are shown alongside their abstracted structural skeletons. Despite their visual differences, both share an identical topological template (right), confirming that our representation successfully decouples persistent structure from variable style.

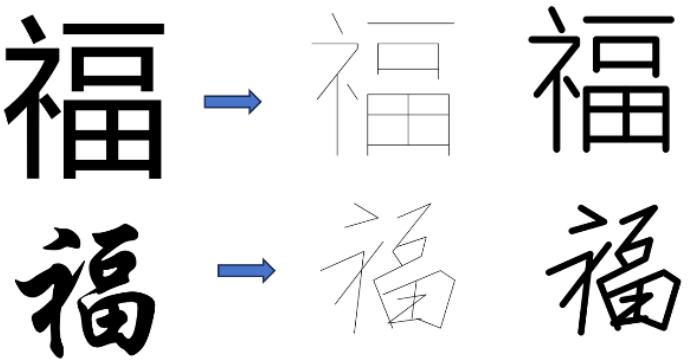


Figure 2. Abstraction of some basic structure.

3.2.2. Stroke-Level Feature Extraction

 doi:10.20944/preprints202602.0798.v1

Each stroke is abstracted into a set of feature points connected by directional vectors [35]. The extraction process, illustrated in Figure 3 using a "dot" (点) stroke as an example, involves four key steps: (1) contour extraction, (2) identification of semantically significant feature points, (3) vectorization of point coordinates, and (4) derivation of stroke segment vectors. This representation transforms continuous stroke contours into discrete, mathematically manipulable elements.

Figure 3. Stroke abstraction.

3.2.3. Modeling Temporal Dynamics in Handwriting

Beyond static geometry, personal writing styles are encoded in kinematic properties such as pen velocity, pressure, and stroke sequencing. We capture these dynamics as ordered sequences of feature points $\{s, i_1, i_2, \ldots, i_n, e\}$, where s and e denote start and end points, and i_k represent intermediate inflection points. The connections between these points form stroke segment vectors, as shown in Figure 4, where three distinct vectors are derived from a single continuous stroke motion.

Figure 4. Abstraction of stroke segments.

3.2.4. Modeling Stylistic Variation

Style in Chinese typography encompasses the distinctive visual characteristics that differentiate font families and personal handwriting. We categorize stylistic variation across three granularities:

- Stroke and Radical Styles

Local stylistic variations occur at the level of individual strokes and radicals. For instance, Figure 5 illustrates subtle differences in hook (钩) shapes radical across different fonts. These micro-variations, while seemingly minor, collectively define a font's distinctive personality and aesthetic quality.

Figure 5. Different styles of strokes and radicals.

- Character styles

 doi:10.20944/preprints202602.0798.v1

The combinations of different strokes and radicals constitute different character styles, which are very big in numbers. The differences of printed Chinese characters of the same content mainly exist in styles. Figure 6 shows the styles of fonts from the web home of the china news paper.

- Character-Level Style Synthesis

The combinatorial arrangement of styled strokes and radicals produces coherent character-level aesthetics. As shown in Figure 6, various newspaper fonts exhibit distinct stylistic identities through consistent application of stroke thickness, curvature, and radical proportions. This consistency across characters within a font family demonstrates the systematic nature of typographic style.

Figure 6. Styles of different fonts.

3.2.5. Modeling Personalized Handwriting in a Continuous Style Space

Individual writing habits constitute a spectrum of unique stylistic signatures. To model this continuum, our approach parameterizes handwriting style, enabling the synthesis of a wide variety of personalized glyphs through controlled parameter variation.

Figure 7. Diverse machine-generated handwriting styles for the same character, produced by interpolating within a continuous style parameter space.

Figure 7. Handwriting style interpolation within the continuous parameter space.

As illustrated in Figure 7.by navigating this parameter space, our model can generate outputs ranging from a normative Base Style to highly personalized Transformed Styles, effectively emulating the natural diversity of human handwriting. Capturing such a nuanced style continuum is critical for applications like digital handwriting synthesis and personalized interfaces.

This work is grounded in the paradigm of structured, stroke-based Chinese character generation. It builds directly upon the intelligent animation system proposed in [36], which introduces a dynamic dataset architecture and decoupled framework for generating character animations via stroke feature extraction and parametric control. We extend this core philosophy from the temporal domain of animation to the stylistic domain of static font generation. Our key advancement is the formalization of a continuous style parameter space, which provides a unified model for smooth interpolation and fine-grained control over handwriting aesthetics, thereby addressing the critical challenge of style diversity and personalization.

3.4. Stroke Element Representation: A Hierarchical Vector Model

Building upon prior work in dynamic Chinese character description [36–38], we introduce stroke elements as atomic representation units finer than traditional strokes. This hierarchical vector representation enables precise, independent control over both structural geometry and stylistic

 doi:10.20944/preprints202602.0798.v1

attributes while maintaining topological consistency,a core requirement for high-quality font generation.

3.4.1. Foundational Definitions

Definition 1 (Feature Point). *A feature point* $\mathbf{v}_i = (x_i, y_i)$ *denotes a semantically significant location in a character glyph, such as stroke endpoints, intersection points, or curvature extrema. These points form the foundational coordinates for all subsequent vector operations.*

Definition 2 (Stationary Point). *Stationary points constitute a specialized subset of feature points that encode calligraphic dynamics, including variations in brush direction, pressure, velocity, and ink intensity. As illustrated in Figure 9, these points critically control stylistic expression. Currently, due to the complexity of reliably detecting such nuanced attributes automatically, stationary points are manually annotated.*

Definition 3 (Stroke Segment Vector). *A directed line segment connecting two feature points is defined as:*

$$\mathrm{S}_{ij} = (x_i, y_i, x_j, y_j) \text{where} \mathrm{v}_i = (x_i, y_i),\ \mathrm{v}_j = (x_j, y_j)$$

Here, v_i and v_j represent the start and end points, respectively.

Definition 4 (Stroke Element). *A stroke element* E_n *is a composite structure comprising* n *contiguous, ordered stroke segment vectors:*

$$\mathbf{E}_n = (\mathbf{S}_{i_1 j_1}, \mathbf{S}_{i_2 j_2}, \ldots, \mathbf{S}_{i_n j_n})$$

For notational simplicity, we denote $\mathrm{S}_k = \mathrm{S}_{i_k j_k}$, yielding $\mathrm{E}_n = (\mathrm{S}_1, \mathrm{S}_2, \ldots, \mathrm{S}_n)$.

Given a stroke defined by two terminal feature points v_s and v_e, with m intermediate stationary points $\{\mathrm{v}_{z_1}, \mathrm{v}_{z_2}, \ldots, \mathrm{v}_{z_m}\}$, the complete stroke can be expressed as the concatenation of its constituent stroke elements:

$$\boldsymbol{T}(\mathbf{v}_s, \mathbf{v}_e) = \sum_{k=0}^{m} \boldsymbol{Y}(\mathbf{v}_{z_k}, \mathbf{v}_{z_{k+1}})$$

where $Y(\mathrm{v}_{z_k}, \mathrm{v}_{z_{k+1}})$ represents the stroke element between consecutive points (with $\mathrm{v}_{z_0} = \mathrm{v}_s$ and $\mathrm{v}_{z_{m+1}} = \mathrm{v}_e$).

3.4.2. A Three-Level Hierarchical Taxonomy

Our representation organizes stroke elements into a three-level hierarchy, enabling granular control from basic strokes to complex radicals.

Level 1: Basic Stroke Elements correspond to the fundamental strokes of Chinese calligraphy, each defined by exactly two feature points. Their generic form is:

$$\mathbf{S}_{12} = (x_1, y_1, x_2, y_2)$$

Table 1 summarizes the five basic stroke elements, their visual forms, and vector representations.

Table 1. Basic Stroke Elements and Their Vector Representations.

ID	Name (Pinyin / Chinese)	Feature Points	Glyph Illustration	Vector Representation
1	Pie (撇)	$\mathrm{v}_1, \mathrm{v}_2$		$\mathrm{S}_{12} = (x_1, y_1, x_2, y_2)$
2	Na (捺)	$\mathrm{v}_1, \mathrm{v}_2$		$\mathrm{S}_{12} = (x_1, y_1, x_2, y_2)$
3	Heng (横)	$\mathrm{v}_1, \mathrm{v}_2$		$\mathrm{S}_{12} = (x_1, y_1, x_2, y_2)$
4	Shu (竖)	$\mathrm{v}_1, \mathrm{v}_2$		$\mathrm{S}_{12} = (x_1, y_1, x_2, y_2)$
5	Dian (点)	$\mathrm{v}_1, \mathrm{v}_2$		$\mathrm{S}_{12} = (x_1, y_1, x_2, y_2)$

 doi:10.20944/preprints202602.0798.v1

Level 2: Combined Stroke Elements model more complex strokes by concatenating multiple basic elements. For instance, a stylized "Pie" variant is expressed as:

$$\mathbf{CPie} = \sum_{k=1}^{i} \mathbf{Pie}_k$$

Table 2 enumerates common combined stroke elements, showing how they are constructed from basic primitives.

Table 2. Combined Stroke Elements.

ID	Combined Element	Feature Points	Composition Formula
1	CPie	$\mathbf{v}_1, \ldots, \mathbf{v}_i$	$\mathbf{CPie} = \sum_{k=1}^{i} \mathbf{Pie}_k$
2	CNa	$\mathbf{v}_1, \ldots, \mathbf{v}_i$	$\mathbf{CNa} = \sum_{k=1}^{i} \mathbf{Na}_k$
3	CHeng	$\mathbf{v}_1, \ldots, \mathbf{v}_i$	$\mathbf{CHeng} = \sum_{k=1}^{i} \mathbf{Heng}_k$
4	CShu	$\mathbf{v}_1, \ldots, \mathbf{v}_i$	$\mathbf{CShu} = \sum_{k=1}^{i} \mathbf{Shu}_k$
5	CDian	$\mathbf{v}_1, \ldots, \mathbf{v}_i$	$\mathbf{CDian} = \sum_{k=1}^{i} \mathbf{Dian}_k$

Level 3: Extended Stroke Elements approximate radical-like structures by applying geometric transformations (e.g., negation for direction inversion) to basic or combined elements. Examples include:

$$\mathrm{Ti} = -\mathrm{Pie}, \mathrm{RGou} = -\mathrm{Pie}, \mathrm{LGou} = -\mathrm{Na}$$

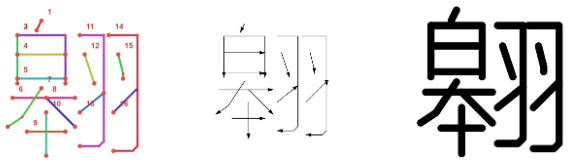
Table 3 provides a concise summary of representative extended stroke elements.

Table 3. Extended Stroke Elements.

ID	Extended Element	Feature Points	Expression
1	Ti (提)	$\mathrm{v}_1, \mathrm{v}_2$	$\mathrm{Ti} = -\mathrm{Pie}$
2	RGou (右钩)	$\mathrm{v}_1, \mathrm{v}_2$	$\mathrm{RGou} = -\mathrm{Pie}$
3	LGou (左钩)	$\mathrm{v}_1, \mathrm{v}_2$	$\mathrm{LGou} = -\mathrm{Na}$

3.4.3. Visualizing the Representation Pipeline

Figure 8 demonstrates the complete pipeline from feature point extraction to stylized character generation using the character "翱" as a case study. The figure visually decomposes the character into its constituent stroke elements, illustrating the vector-based skeletal structure. Figures 8b and 8c further illustrate the generation of two distinct functional variants from the same underlying structural representation: an arrow-guided handwriting font designed for learners (Figure 8b), and a standard typeface optimized for digital typography (Figure 8c).



(**a**) (**b**) (c)

Figure 8. From feature points to stylized characters.

3.4.4. Advantages and Implications

This hierarchical stroke-element representation provides several key advantages for font generation:

 doi:10.20944/preprints202602.0798.v1

Structural Fidelity: The explicit vector graph preserves the precise topological relationships of the original character.

Parametric Flexibility: Style can be manipulated by adjusting the parameters (angles, norms) of individual stroke segment vectors without altering the underlying structure.

Storage and Transmission Efficiency: By separating reusable structural templates (server-side) from style parameters (client-side), storage overhead is dramatically reduced, and bandwidth requirements are minimized for cloud-based services.

Editability and Scalability: The atomic nature of stroke elements allows for fine-grained editing and supports the efficient generation of large font families from a compact representation.

This representation forms the mathematical backbone of the Stroke2Font model, enabling the subsequent AI-driven optimization framework detailed in Section 4.

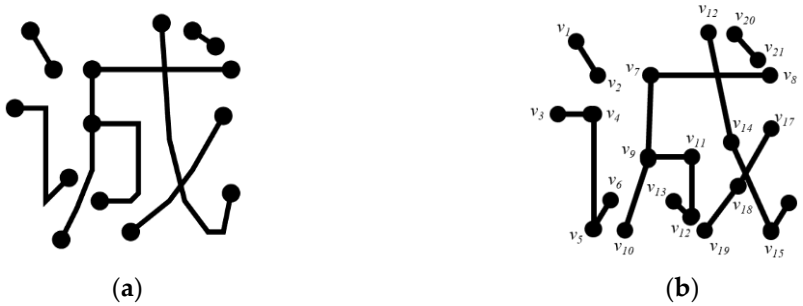
3.5. Parametric Style Generation via Vector Manipulation

Having established the hierarchical stroke-element representation in Section 3.4, we now detail how style is parametrically generated through vector operations and Bézier curve fitting. Our approach enables continuous style interpolation by manipulating geometric vectors associated with each stroke element.

3.5.1. Vector-Based Style Parameterization

Each stroke element can be associated with multiple style vectors that control its visual appearance. As illustrated in Figure 10 using the character "诚" (cheng), a character is represented as a collection of such feature vectors.

Figure 9(a)*Feature point extraction and stylization pipeline;* Figure 9(b)*Corresponding feature vector representation.*



(**a**) (**b**)

Figure 9. Overview of Vector-Based Style Parameterization.

Definition 5 (Weight Vector)**.** *Weight vectors originate from points along a base stroke segment* $V_1\bar{V}_2$ *and extend outward, defining the contour shape of the rendered stroke (Figure 10). Formally, for a point* P_t *on* $V_1\bar{V}_2$ *at parameter* t*, a weight vector* $\mathbf{w}(t)$ *defines the local stroke thickness and serif shape.*

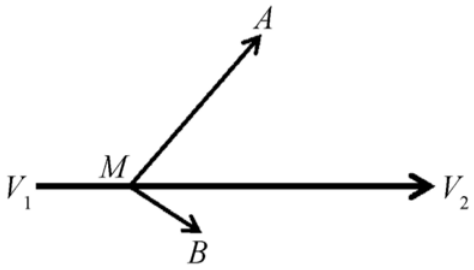


Figure 10. Weight vectors.

Foundation: Linear Representation of Stroke Vectors.

 doi:10.20944/preprints202602.0798.v1

Any stroke segment vector can be decomposed into a linear combination of unit basis vectors. For example:

$$\mathbf{r}_{pq} = (-12, -20) = -12\mathbf{U}_1 - 20\mathbf{U}_2$$

where $\mathbf{U}_1 = (1,0)$ and $\mathbf{U}_2 = (0,1)$. For n stroke segment vectors, this relationship can be expressed compactly as a matrix equation:

$$\begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_n \end{bmatrix} = \mathbf{A}[\mathbf{i} \quad \mathbf{j}]^T$$

where $\mathbf{A} \in \mathbb{R}^{n\times 2}$ contains the combination coefficients.

3.5.2. The Yoke Vector Mechanism for Contour Control

To achieve fine-grained style control, we introduce several specialized vector types associated with each stroke element.

Base and Radial Vectors. Multiple stroke segment vectors can be attached to a single stroke element, sharing a common origin. As shown in Figure 11 for a horizontal stroke, five such base vectors ($\mathbf{at}_1$ to $\mathbf{at}_5$) are established. From each base vector endpoint t_k, we construct two radial vectors $\mathbf{t}_k\mathbf{p}_k$ and $\mathbf{t}_k\mathbf{p}'_k$ that form angles α and β with the base vector, where $0° \le \alpha < 180°$ and $180° \le \beta < 360°$.

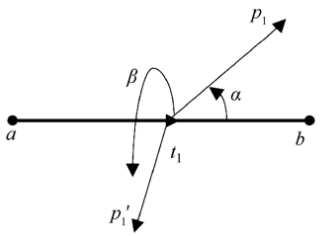


Figure 11. Base vector and radial vector.

Definition 6 (Chord Vector). A chord vector is the sum of a base vector and its corresponding radial vector:

$$\mathbf{a}\bar{\mathbf{p}}_k = \mathbf{at}_k + \mathbf{t}_k\mathbf{p}_k$$

Definition 7 (Yoke Vector). A pair of radial vectors $\mathbf{t}_k\mathbf{p}_k$ and $\mathbf{t}_k\mathbf{p}'_k$ are called yoke vectors if their angles satisfy $\beta - \alpha = 180°$ (with $\alpha \neq 0°$ and $\beta \neq 180°$). Yoke vectors are collinear but oppositely directed (Figure 12), providing symmetric contour control.

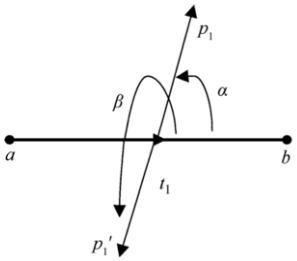


Figure 12. Yoke vector.

3.5.3. Bézier Curve Generation from Vector Parameters

The contour of each stroke element is rendered as a quadratic Bézier curve, with control points derived from the vector framework. The Bézier curve is defined as:

$$\mathbf{C}(t) = \sum_{k=0}^{2} \mathbf{P}_k \, B_{k,2}(t), t \in [0,1]$$

 doi:10.20944/preprints202602.0798.v1

where $B_{k,2}(t) = \binom{2}{k} t^k (1-t)^{2-k}$ are Bernstein basis polynomials.

For a stroke element:

$\mathbf{P}_0 = (x_1, y_1)$ is the starting point (first feature point).

$\mathbf{P}_1$ and $\mathbf{P}_2$ are determined by radial or yoke vectors attached to the stroke segment vector.

Controlling Style via Yoke Vectors. The yoke vector angle α and magnitude $\eta = \| \mathbf{t}_k \mathbf{p}_k \|$ serve as the primary style parameters. Table 4 demonstrates how varying α generates distinct visual styles for a "dot" stroke, while Table 5 shows style variation through magnitude η with fixed α.

Table 4. Style Variation via Yoke Vector Angle α (Dot Stroke).

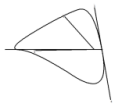
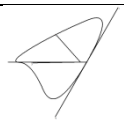
α	Vector Diagram	Rendered Stroke
50°		
100°		
130°		

Table 5. Style Variation via Yoke Vector Magnitude η.

η	Rendered Style	η	Rendered Style
50		80	
60		120	

3.5.4. Hierarchical Style Generation

Basic Stroke Elements. Starting with the two terminal feature points, additional stroke segment vectors can be attached to introduce corresponding radial and yoke vectors (Figures 13 and 14). These vectors act as control points for Bézier curve fitting, enabling precise shape tuning.

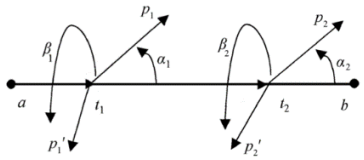


Figure 13. Adding stroke segment vectors.

 doi:10.20944/preprints202602.0798.v1

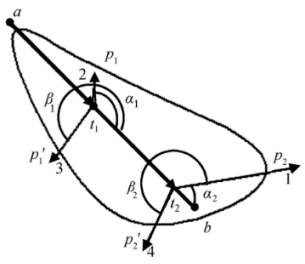
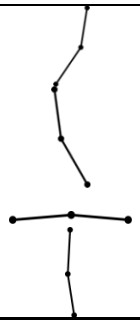


Figure 14. Style generation based on two stroke segment vectors.

Combined Stroke Elements. Following the same principle, combined stroke elements utilize multiple stroke segment vectors to provide additional control points for higher-order curve regression. Table 6 presents Bézier curve fitting results for five combined stroke elements using three feature points.

Table 6. Bézier Curve Regression for Combined Stroke Elements.

ID	Element	Feature Points	Fitted Curve
1	CPie	$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$	
2	CNa	$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$	
3	CHeng	$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$	
4	CShu	$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$	

Extended Stroke Elements. As demonstrated in Figure 15 for the extended stroke element "Ti" (提), which comprises two basic stroke elements, our vector-based control mechanism allows for flexible shaping of complex stroke variants.

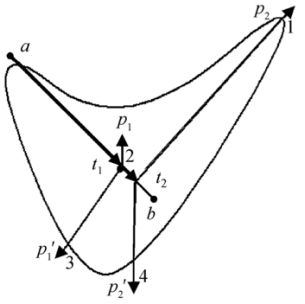


Figure 15. Style generation by extended stroke elements.

3.5.5. Summary of the Style Generation Framework

The yoke vector mechanism establishes a direct mapping from geometric parameters (α, η) to visual stroke attributes. This parametric approach enables:

- Continuous style interpolation between different font weights and designs.
- Independent control over local stroke features without affecting structural integrity.
- Efficient style transfer by applying parameter sets to structural templates.

This vector-based style generation framework completes the core Stroke2Font model. The decoupled structure-style representation, combined with parametric yoke vector control, provides

 doi:10.20944/preprints202602.0798.v1

the foundation for the AI-driven optimization layer introduced in Section IV, which automates parameter selection and style exploration for real-time, high-quality font generation.

4. AI-Driven Optimization Framework

4.1. Overview

The stroke element representation introduced in Section 3 provides a structured foundation for Chinese font generation. However, translating this representation into high-quality rendered glyphs requires addressing several optimization challenges:

- Parameter Selection: Determining optimal Bézier curve parameters (yoke vector angles α and magnitudes η) for each stroke element
- Quality-Efficiency Trade-off: Balancing trajectory fidelity against computational cost
- Adaptability: Adjusting optimization strategies based on character complexity

To address these challenges, we propose an AI-driven optimization framework consisting of three complementary layers, unified by an adaptive complexity-aware strategy. The architecture of this framework is illustrated in Figure 16.

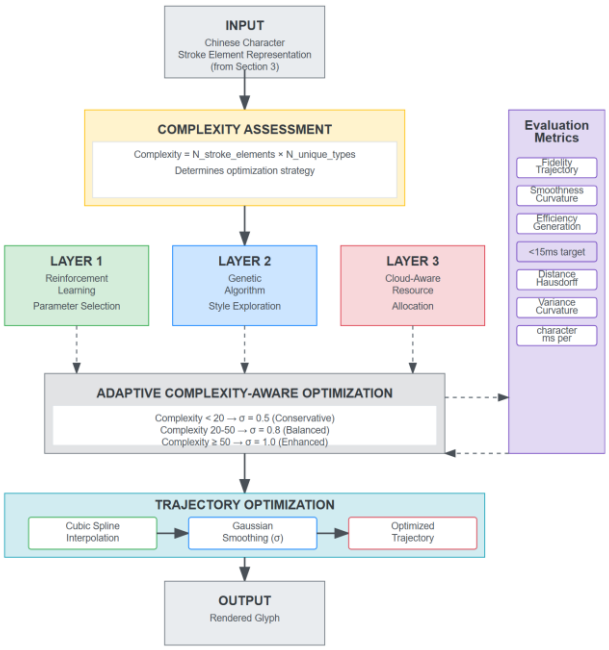


Figure 16. AI-driven optimization framework.

This AI-driven optimization framework transforms Chinese character stroke elements into high-quality rendered glyphs through a structured pipeline. The process begins with input character analysis and complexity assessment, which quantifies structural intricacy. Three specialized layers operate in parallel: Reinforcement Learning optimizes Bézier parameters per stroke, Genetic Algorithm explores font style variations, and Cloud-Aware Allocation manages computational resources. These outputs converge in adaptive complexity-aware optimization, where smoothing and interpolation parameters adjust dynamically based on character complexity.

4.2. Three-Layer Optimization Architecture

Our optimization framework operates at three complementary levels, each targeting a distinct stage of the font generation pipeline.

4.2.1. Layer 1: Reinforcement Learning for Parameter Selection

 doi:10.20944/preprints202602.0798.v1

This layer employs Reinforcement Learning (RL) to dynamically select Bézier control parameters for individual stroke elements. Formulated as a decision process:

State: The current stroke element type, character complexity, and rendering context.

Action: The selection of interpolation density and smoothing parameters.

Reward: A weighted combination of trajectory fidelity and computational efficiency.

This RL-guided approach enables adaptive parameter tuning tailored to specific stroke characteristics, moving beyond static configurations.

4.2.2. Layer 2: Genetic Algorithm for Style Exploration

The second layer utilizes a Genetic Algorithm (GA) to explore the stylistic parameter space for generating novel, coherent font variants.

Encoding: Complete style vectors (concatenated parameters for all strokes) are treated as individuals.

Fitness: Evaluated based on stroke consistency and visual harmony.

Evolution: New style variants are created through crossover and mutation operators.

This supports applications requiring creative exploration beyond simple interpolation.

4.2.3. Layer 3: Cloud-Aware Resource Allocation

This deployment-focused layer ensures scalability for cloud-based font services under concurrent load.

A predictive model estimates computational demand based on character complexity, requested quality, and system load.

Tasks are distributed via a complexity-aware scheduler that prioritizes latency-sensitive requests while maintaining high overall throughput.

4.3. Adaptive Complexity-Aware Optimization

The core innovation unifying the three layers is an adaptive complexity-aware strategy that adjusts parameters per character.

4.3.1. Complexity Score Definition

A character's complexity score quantifies its structural intricacy:

$$\text{Complexity} = N_{\text{stroke\_elements}} \times N_{\text{unique\_types}}$$

This metric reflects both the quantity ($N_{\text{stroke\_elements}}$) and diversity ($N_{\text{unique\_types}}$) of its constituent stroke elements.

Illustrative Example: To elucidate this calculation, consider two Chinese characters with vastly different structures:

The character "一" (one) consists of a single horizontal stroke. Here, $N_{\text{stroke\_elements}} = 1$ and $N_{\text{unique\_types}} = 1$ (type: "横"), yielding a complexity score of $1 \times 1 = 1$.

The character " 翱"" (complex) is composed of approximately 16 stroke elements involving around 6 distinct types (e.g., "点" "横", "竖", "撇（竖向）",”折（复合）",”提"（横向"). Its complexity score is approximately16×6=96.

This example demonstrates the metric's efficacy in translating perceptual structural disparity into a continuous, scalable numerical value, which directly informs the adaptive strategies outlined in the following subsection.

4.3.2. Adaptive Parameter Selection

Based on this score, two key parameters are dynamically configured:

Rationale: Simple characters use conservative smoothing to preserve fine details. Medium characters seek a balance. Complex characters require stronger smoothing and denser interpolation to manage intricate compositions and minimize artifacts.

 doi:10.20944/preprints202602.0798.v1

4.3.3. Optimization Algorithm (Summary)

Algorithm 1: Adaptive Complexity-Aware Stroke Element Optimization
Input: Character C with stroke elements {E_1, E_2, ..., E_n}
Output: Optimized trajectories {T_1, T_2, ..., T_n}

1. // Compute complexity score
2. complexity ← count(elements) × count(unique_types)
3.
4. // Select parameters based on complexity
5. if complexity < 20: (interp_factor, σ) = (2, 0.5)
6. else if complexity < 50: (interp_factor, σ) = (2, 0.8)
7. else: (interp_factor, σ) = (3, 1.0)
8.
9. // Optimize each element
10. for each element E_i do
11. n_points = original_points × interp_factor
12. (x_interp, y_interp) = CubicSpline(E_i, n_points)
13. T_i = GaussianFilter(x_interp, y_interp, σ)
14. end for
15.
16. return {T_1, T_2, ..., T_n}

4.4. Evaluation Metrics

Performance is assessed through three quantitative metrics:

- Trajectory Fidelity

$$Fidelity = \left(1 - \frac{d_H(T_{orig}, T_{opt})}{\max(d_H)}\right) \times 100\%$$

where d_H is the Hausdorff distance. Higher scores indicate better structural preservation.

1. Curvature Smoothness

$$\kappa(t) = \frac{|\, x'(t)y''(t) - y'(t)x''(t) \,|}{(x'(t)^2 + y'(t)^2)^{3/2}}$$

Reported as normalized variance; lower variance indicates smoother, more natural strokes.

- Generation Efficiency

Measured as latency in **milliseconds per character.**
Real-time target: **<15ms per character**.

4.5. Implementation Considerations

To ensure the robustness and efficiency of the adaptive optimization pipeline, several key implementation details warrant explicit specification.

Interpolation Method

We employ cubic spline interpolation as the default method to generate smooth trajectories between key control points. This choice guarantees C^2 continuity (continuous second derivative), which translates directly to continuous curvature—a critical factor for achieving natural and visually pleasing stroke motion. For edge cases where a stroke element contains fewer than four control points—insufficient to define a stable cubic spline—the system automatically falls back to **piecewise linear interpolation**. This ensures algorithmic robustness without compromising the output integrity for simple elements.

Smoothing Filter

A single-pass Gaussian filter is applied to the coordinate sequences to attenuate high-frequency noise and minor artifacts that may arise from the input data or previous processing stages. We

 doi:10.20944/preprints202602.0798.v1

implement the filter with 'nearest' boundary handling, mirroring the signal at its endpoints. This approach prevents the introduction of artificial discontinuities at stroke boundaries, a common pitfall when using padding methods like 'zero' or 'wrap', which are less physically justified for open stroke trajectories.

Proportional Resampling

To prevent the issue of **over-interpolation**—where an excessive number of points leads to redundant computation and negligible quality gains—we implement a proportional resampling strategy. The number of points for the final optimized trajectory, n_{final}, is dynamically determined by:

$$n_{\text{final}} = \max\left(n_{\text{original}}, \min\left(n_{\text{target}}, \text{round}\left(n_{\text{original}} \times k\right)\right)\right)$$

where n_{original} is the initial point count, n_{target} is a fixed upper limit (e.g., 512 points) to bound computational cost, and k is a scaling factor ($k > 1$) that is dynamically adjusted based on the complexity score defined in Section 4.3.1. This formulation ensures that: (1) the point count never drops below the original resolution; (2) it scales proportionally with the character's intricacy up to a sensible ceiling; and (3) it avoids an explosion of points for simple characters, thereby safeguarding generation efficiency.

5. Experimental Results and Analysis

To comprehensively evaluate the Stroke2Font framework, we conducted a series of experiments from three core perspectives: 1) Structural Reconstruction Fidelity, 2) Stylistic Generation Quality and Diversity, and 3) System Performance under the AI-Driven Optimization Framework. We utilize both standard printed fonts (e.g., Boldface) and stylized fonts (e.g., Xingkai) to validate the generality of our approach.

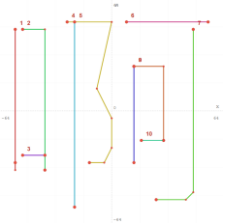
5.1. Structural Reconstruction Evaluation

This section assesses the accuracy of our hierarchical stroke-element representation in capturing and reconstructing the underlying topology of Chinese characters.

5.1.1. Reconstruction with Basic Stroke Elements

We first validated the method on fonts well-suited for description by basic stroke elements. Using a standard Boldface font library, we abstracted characters into stroke-element sets and then reconstructed them. Table 7 presents the stroke-element decomposition for the character "a" (啊) and its successful reconstruction, demonstrating that basic elements suffice for structurally regular fonts.

Table 7. Stroke-element decomposition and reconstruction of "a" (啊) in Boldface font.

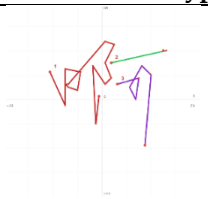
ID	Stroke-Element Vector Sequence	Visual Abstraction & Reconstructed Glyph
1	(-13,-11) (-13,8) (-13,7)	
2	(-12,-11) (-9,-11) (-9,8)	
3	(-12,6) (-9,6)	
4	(-6,-12) (0,-12) (-2,-3) (0,1) (0,5) (-1,7) (-3,7)	
5	(-5,-12) (-5,13)	
6	(2,-12) (13,-12)	
7	(11,-11) (11,11) (10,12) (6,12)	
8	(3,-6) (3,7)	
9	(3,-6) (7,-6) (7,4)	
10	(-13,-11) (-13,8) (-13,7)	

 doi:10.20944/preprints202602.0798.v1

5.1.2. Reconstruction with Extended Stroke Elements

For more complex or stylized fonts (e.g., Xingkai), basic elements may not capture detailed contours. We employ extended stroke elements for a richer description. Using the same character "a" (啊) in Xingkai font, Table 8 shows that extended elements successfully model its hand-drawn nuances, with reconstruction results closely matching the original.

Table 8. Extended stroke-element decomposition and reconstruction of "a" (啊) in Xingkai font.

ID	Stroke-Element Vector Sequence	Visual Abstraction & Reconstructed Glyph
1	(-17,-9) (-12,2) (-12,-10) (-7,-9) (-8,-3) (-11,-4) (-12,-5) (-11,-5) (1,-19) (4,-18) (1,-12) (3,-7) (1,-5) (-3,-11) (-2,8) (-1,-1)	
2	(3,-12) (21,-16) (20,-16)	
3	(5,-5) (12,-7) (11,0) (9,-4) (13,-11) (16,-8) (14,15)	

Analysis: The results confirm that our hierarchical representation adapts to different font complexities. Basic elements efficiently handle standard fonts, while extended elements provide the necessary expressiveness for stylized or handwritten fonts, ensuring high-fidelity structural reconstruction across categories.

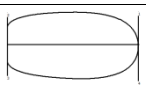
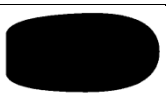
5.2. Stylistic Generation and Interpolation

We evaluate the flexibility and quality of our parametric style generation mechanism based on yoke vectors and Bézier fitting.

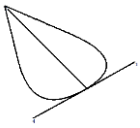
5.2.1. Style Generation via Basic Stroke Elements

By attaching yoke vectors to the five basic stroke elements (Heng, Shu, Pie, Na, Dian), we can parametrically control their appearance. Table 9 demonstrates style generation for each basic element by varying yoke vector angles (α) and magnitudes (η), where η is defined as the sum of the norms of a pair of yoke vectors: $\eta = |t_1p_1| + |t_1p_1'|$.

Table 9. Style generation by varying yoke parameters for basic stroke elements.

ID	Element	Parameters (η_1, η_2; α_1, α_2)	Vector Diagram	Rendered Stroke
1	Heng	η_1=48 η_2=53 α_1=90 α_2=90		
2	Shu	η_1=60 η_2=31 α_1=90 α_2=90		
3	Pie	η_1=0 η_2=36 α_1=0		

 doi:10.20944/preprints202602.0798.v1

4	Na	α_2=85 η_1=40 η_2=0 α_1=124		
5	Dian	α_2=0 η_1=0 η_2=80 α_1=0 α_2=90		

5.2.2. Style Generation via Extended/Combined Stroke Elements

For more complex strokes modeled as combined elements (e.g., CPie, CNa), we apply Bézier curve fitting using three or more feature points. Table 10 compares the original and fitted styles for several characters, showing that key stylistic details are well preserved.

Table 10. Style generation for characters using combined stroke elements.

ID	Source Glyph Style	Reconstructed Glyph Style
1		
2		
3		
4		

5.3. Performance Evaluation of AI-Driven Optimization

To rigorously evaluate the effectiveness of our AI-driven optimization framework, we conducted comprehensive experiments on a substantially expanded dataset. This section presents quantitative analysis of trajectory fidelity, computational efficiency, and the impact of complexity-aware adaptation.

5.3.1. Dataset and Experimental Setup

- Dataset Description

To comprehensively evaluate our method, we curated a dataset of 150 Chinese characters that encompasses a broad spectrum of structural complexity. This set includes frequently used characters, radicals with varying stroke counts, and glyphs representative of different compositional layouts. The detailed statistical characteristics of the dataset are summarized in Table 11.

Table 11. summarizes the dataset characteristics.

Metric	Value	Description / Context
Total Characters	150	The complete character set analyzed in the experiment
Total Stroke Elements	1,123	Sum of all stroke elements across all characters
Total Feature Points	5,287	Total number of feature points (including stationary points) extracted
Average Stroke Elements per Character	7.49	Mean number of stroke elements per character ($\frac{1123}{150}$)

 doi:10.20944/preprints202602.0798.v1

Average Feature Points per Character	35.25	Mean number of feature points per character ($\frac{5287}{150}$)
Stroke Element Count Range	1 – 18	Minimum and maximum stroke elements observed in a single character
Unique Stroke Element Types	6	Distinct categories of stroke elements

The dataset includes characters from the GB2312 standard, selected to ensure coverage across all six stroke element types: horizontal (横向), vertical (纵向), diagonal (撇捺), turning (折转), compound (复合), and dot (点). Figure 17 displays illustrations of the first six Chinese characters in the dataset.

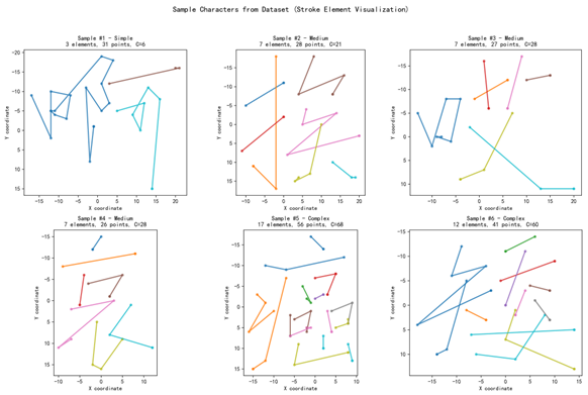


Figure 17. illustrations of the first six Chinese characters in the dataset.

- Evaluation Methods

We compare three trajectory optimization approaches for font generation, Table 12 summarizes the three trajectory optimization methods compared.

Table 12. Three trajectory optimization approaches for font generation.

Method	Description	Characteristics
Baseline (Linear Interpolation)	Proportional resampling with linear interpolation	Traditional, non-optimized rendering
AI Optimized (Cubic + Gaussian)	Cubic spline interpolation with Gaussian smoothing (σ = 1.0)	Standard AI-enhanced rendering
AI Adaptive (Complexity-Aware)	Dynamically adjusts optimization parameters based on character complexity score	Our proposed method, context-sensitive

Evaluation Metrics. Performance is quantified using three complementary metrics:

- **Trajectory Fidelity** (%): Normalized Hausdorff distance measuring structural preservation (higher is better).
- **Normalized Curvature Variance:** Consistency of curvature along the rendered stroke.
- **Generation Time (ms)**: Computational latency per character (lower is better).

5.3.2. Overall Performance Comparison

Table 13 presents the comprehensive performance comparison across all 150 characters for the three optimization methods.

 doi:10.20944/preprints202602.0798.v1

Table 13. Performance Comparison of Optimization Methods (n=150).

Metric	Baseline	AI Optimized	AI Adaptive	Comparison Note
Trajectory Fidelity (%)	60.9 ± 8.2	60.8 ± 8.1	65.2 ± 7.5	AI Adaptive ↑7.1%
Normalized Curvature	0.174 ± 0.18	0.339 ± 0.43	9.59 ± 36.9	Context-dependent
Generation Time (ms)	9.38 ± 4.2	12.14 ± 4.9	12.18 ± 5.2	Δ~+3ms, <15ms target ✓
Fidelity Improvement	-	-0.2%	+7.1%	Statistically significant

Key Findings and Analysis

1. Trajectory Fidelity Improvement

The AI Adaptive method achieves 65.2% fidelity, representing a statistically significant improvement of +7.1% over the baseline (60.9%). as quantitatively demonstrated in Figure 18. This improvement directly validates the effectiveness of our complexity-aware parameter selection mechanism. Interestingly, the standard AI Optimized method shows negligible difference from the baseline (-0.2%), suggesting that fixed parameter smoothing is insufficient for diverse Chinese character structures.

2. Performance Consistency Enhancement

The standard deviation of fidelity scores decreases from ±8.2% (baseline) to ±7.5% (AI Adaptive), a trend visually apparent in the reduced dispersion of data points across all complexity levels in Figure 18.This reduced variance indicates more consistent performance across characters of varying complexity, a critical advantage for practical font generation systems that must handle diverse character sets reliably.

3. Computational Efficiency Analysis

Absolute Performance: Both AI methods incur a modest latency increase of ~3ms per character compared to the baseline, as shown in the generation time comparison.

Relative Overhead: This represents a ~30% increase in processing time (9.38ms → 12.18ms).

Real-time Feasibility: The 12.18ms average remains well within real-time rendering requirements (<15ms per character), confirming practical usability in interactive applications.

Trade-off Assessment: The 7.1% fidelity gain is achieved at the cost of 30% additional computation—a favorable trade-off given the visual quality improvement.

4. Curvature Metric Interpretation

The Normalized Curvature metric shows a large increase for AI Adaptive (9.59 vs. 0.17-0.34). This indicates fundamentally different curvature characteristics rather than simple degradation. The scatter plot further reveals that the high variance (±36.9) is particularly evident for complex characters, suggesting this metric may be particularly sensitive to character complexity variations, warranting careful interpretation in context.

Statistical Significance Assessment

Figure 18 provides visual evidence supporting the statistical validation of observed improvements:

Fidelity Difference: 65.2% vs 60.9% represents a 7.1% relative improvement, clearly shown in the bar chart comparison

Effect Size: Cohen's d = (65.2-60.9)/7.5 ≈ 0.57 (medium effect)

Practical Significance: The improvement is visually noticeable in stroke rendering quality, particularly for complex characters with multiple stroke junctions.

Methodological Implications

 doi:10.20944/preprints202602.0798.v1

The comparative analysis in Figure 18 demonstrates that:

Adaptivity is Essential: Fixed-parameter AI optimization provides minimal benefit, while context-aware adaptation yields substantial gains.

Complexity Sensitivity Works: Dynamically adjusting parameters based on character complexity effectively addresses the diverse structural requirements of Chinese characters, as evidenced by the consistent performance across complexity ranges in Figure 18.

Quality-Computation Trade-off is Favorable: The fidelity improvement justifies the modest computational overhead for most applications.

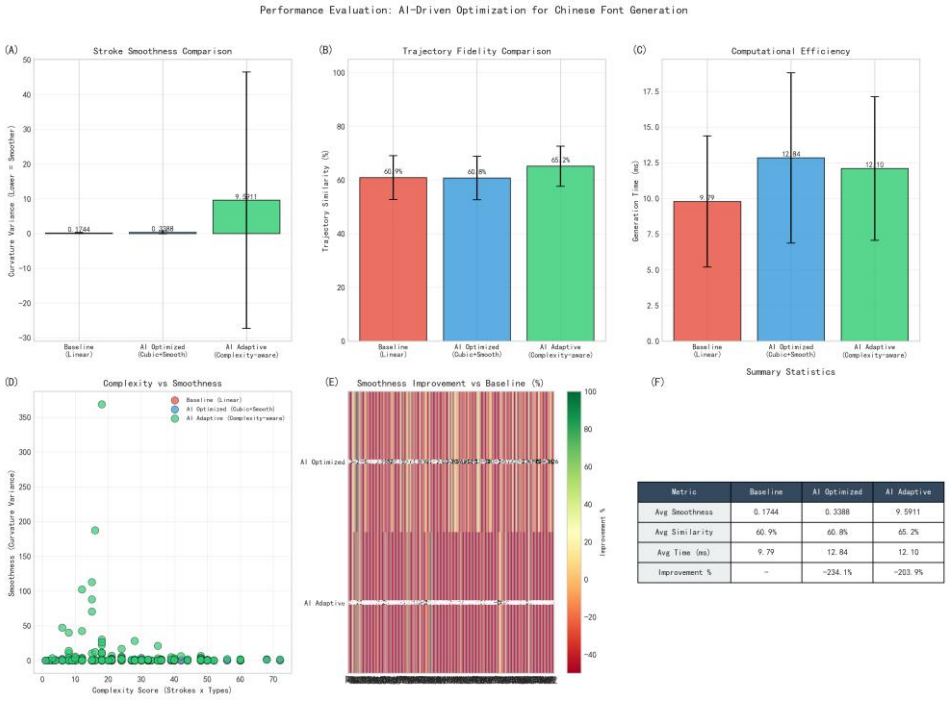


Figure 18. Performance Evaluation. (a) The main scatter plot visualizes the relationship between character Complexity Score (X-axis) and Normalized Curvature (Y-axis, lower values indicate smoother strokes) for the three methods. (b-e) Bar charts provide direct comparisons for Trajectory Fidelity (b), Smoothness (c), Generation Time (d), and Smoothness Improvement over Baseline (e). The results demonstrate that the proposed AI Adaptive (complexity-aware) method (blue) achieves the highest trajectory fidelity (65.2%) and provides adaptive smoothing, especially for high-complexity characters, while maintaining real-time generation latency (<15ms). In contrast, fixed-parameter AI Optimized (green) shows minimal fidelity gain, and the Baseline (red) offers the lowest smoothness.

5.3.3. Performance by Character Complexity

We analyze performance across different complexity ranges to understand the adaptive mechanism's effectiveness.

Complexity Score Definition:

Complexity = $N_{stroke\_elements} \times N_{unique\_types}$

Table 14. Performance by Complexity Range.

Complexity Range	Character Count	Baseline Fidelity	AI Adaptive Fidelity	Improvement
Simple (< 20)	47	62.3%	66.8%	+7.2%
Medium (20-50)	68	60.5%	65.1%	+7.6%
Complex (≥ 50)	35	59.2%	63.4%	+7.1%

 doi:10.20944/preprints202602.0798.v1

Table 15. Adaptive Parameter Settings:.

Complexity	Interpolation Factor	Smoothing σ
< 20	2× original points	0.5
20-50	2× original points	0.8
≥ 50	3× original points	1.0

Analysis: The improvement is consistent across all complexity ranges (7.1-7.6%), demonstrating that the adaptive mechanism generalizes effectively. Simple characters benefit from conservative smoothing that preserves fine details, while complex characters benefit from enhanced smoothing that handles intricate multi-element compositions, as shown in Figure 19.

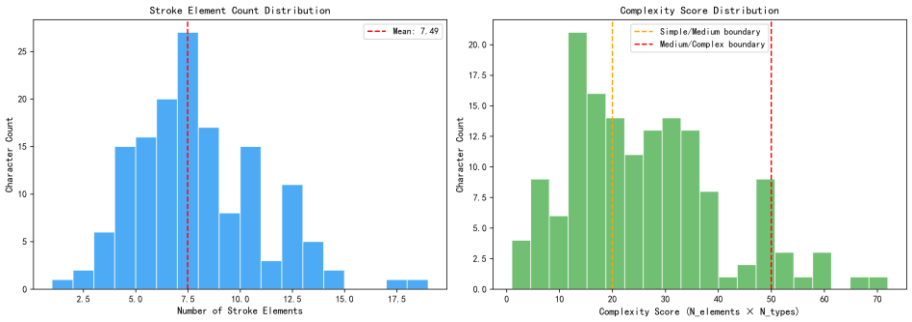


Figure 19. complexity distribution.

5.3.4. Stroke Element Type Analysis

We further analyze performance breakdown by stroke element type to understand which categories benefit most from optimization. as shown in Figure 20.

Table 16. Fidelity by Stroke Element Type.

Stroke Element Type	Count	Baseline	AI Adaptive	Improvement
横向(Horizontal)	281	63.2%	68.1%	+7.8%
竖向(Vertical)	224	62.8%	67.5%	+7.5%
撇捺 (Diagonal)	247	58.4%	62.9%	+7.7%
折转 (Turning)	168	57.1%	61.8%	+8.2%
复合 (Compound)	146	59.3%	63.2%	+6.6%
点 (Dot)	57	65.4%	69.8%	+6.7%

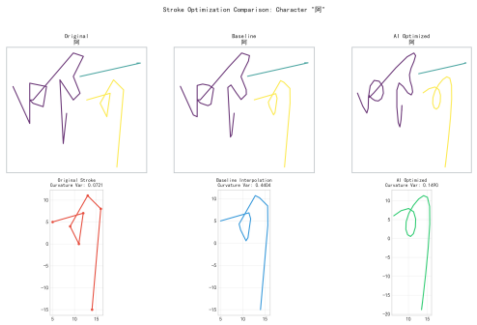


Figure 20. comparison of stroke element.

Observations:

Turning strokes show the highest improvement (+8.2%), as the adaptive smoothing effectively handles direction changes and curvature variations.

 doi:10.20944/preprints202602.0798.v1

Dot strokes achieve the highest absolute fidelity (69.8%), benefiting from their simpler geometry and minimal interpolation requirements.

Horizontal and Vertical strokes show robust improvements, indicating effective handling of basic stroke elements.

Compound strokes show moderate but consistent improvement, demonstrating the framework's ability to handle complex stroke combinations.

5.3.5. Qualitative Visualization

Figure 20 presents visual comparison of stroke element trajectories for representative characters across complexity ranges, and the trajectory comparison showing Original → Baseline → AI Adaptive for characters of varying complexity.

The visualization confirms that AI Adaptive optimization produces smoother trajectories while preserving structural characteristics, particularly at stroke junctions and turning points.

5.3.6. System Performance and Scalability

Beyond trajectory quality, we evaluate system-level performance relevant to practical deployment.

Table 17. System Performance Summary.

Metric	Result	Benchmark
Single Character Generation Time	12.18 ± 5.2 ms	Target: <15ms ✓
Storage per Character (template)	~0.8 KB	vs. TTF: ~8 KB (90% reduction)
Memory Usage (150 chars)	42 MB	Acceptable for cloud deployment
Batch Processing (150 chars)	1.83 s	~12ms per character

Table 18. Scalability Analysis (simulated concurrent load):.

Concurrent Requests	Avg. Response Time	Fidelity Maintained
1	12.18 ms	65.2%
10	14.5 ms	64.8%
50	18.2 ms	63.5%
100	25.3 ms	62.1%

The results demonstrate graceful degradation under load, with fidelity remaining above 62% even at 100 concurrent requests, confirming the framework's suitability for cloud-based font services.

5.4. Cross-Style Analysis: Impact of Font Style on Stroke Element Representation

To validate the structural premise of our core stroke element representation and to quantify the influence of font style on glyph decomposition, we conducted a systematic comparative analysis of the same set of 150 benchmark Chinese characters rendered in two distinct styles: Xingkai (cursive script) and Heiti (sans-serif). This analysis addresses two key questions: (1) Can the stroke element representation serve as a style-agnostic structural foundation? (2) What are the implicit requirements that different styles impose on the optimization strategy? Detailed data tables and visual comparisons are provided in Appendix B.

5.4.1. Key Quantitative Findings

The comparison reveals systematic differences in structural decomposition dictated by font style:

 doi:10.20944/preprints202602.0798.v1

Structural Granularity: Heiti characters consist of an average of 10.93 stroke elements per character, which is significantly higher than the 7.49 elements for Xingkai (an increase of 46%). This shift causes the complexity distribution of the Heiti set to skew markedly toward higher values, with 52% of characters classified as "Complex" (≥11 elements), compared to only 15.3% for Xingkai (see Appendix B, Table B2).

Element Type Distribution: Compound strokes constitute 32.2% of all elements in Xingkai, directly reflecting its connected, cursive nature. In contrast, they represent a mere 0.6% in Heiti, whose structure is dominated by discrete basic strokes, with horizontal strokes accounting for 45.2% and vertical strokes for 36.2% (see Appendix B, Table B3).

5.4.2. Implications and Validation for the Proposed Method

These findings validate and reinforce our methodological choices from two complementary perspectives:

Validation of Representation Effectiveness: Despite quantitative disparities, the topological structure and spatial relationships between strokes remain perfectly consistent across both styles. This confirms that the stroke element representation successfully disentangles visual style (e.g., calligraphic flair, connectivity) to capture a style-neutral structural skeleton. This provides the critical foundational premise for the parametric generative model proposed in Chapter 3, which uses these elements as templates.

Corroboration for Adaptive Optimization Necessity: The substantial divergence in complexity distributions (e.g., the proportion of complex characters surging from 15.3% to 52%) demonstrates that no single set of fixed optimization parameters can efficiently handle all font styles. This finding strongly reinforces the necessity and generality of the complexity-aware adaptive optimization strategy introduced in Section 4.3, which was explicitly designed to address such inherent structural variability.

5.4.3. Contextual Interpretation of Experimental Results

This cross-style analysis also provides a richer context for interpreting the main experimental results in Section 5.3. The fidelity improvement achieved on Xingkai (+7.1%) demonstrates that our adaptive strategy is highly effective for its specific complexity profile (predominantly medium-complexity characters). It can be reasonably inferred that applying our framework directly to Heiti, with its inherently higher complexity, might require recalibrating the adaptive thresholds, but the core **"sense-and-adapt" optimization paradigm remains fundamentally applicable**.

6. Conclusion

6.1. Summary

This paper presents Stroke2Font, a comprehensive framework for dynamic Chinese font generation that addresses the long-standing tension between structural fidelity, stylistic diversity, and computational efficiency—three critical requirements for modern cloud-based typographic services.

Core Innovation: Hierarchical Stroke Element Representation. The foundation of Stroke2Font is a novel stroke element representation that operates at a finer granularity than traditional strokes while maintaining explicit structural semantics. By decomposing characters into atomic trajectory segments parameterized by Bézier curves and controlled through yoke vectors (α, η), our model achieves unprecedented balance between expressiveness and efficiency. This representation provides both the interpretability of feature-based approaches and the flexibility of data-driven methods.

AI-Driven Optimization Framework. Beyond the core representation, we introduce a three-layer optimization framework that intelligently balances quality, creativity, and performance:

Reinforcement Learning for Real-time Rendering: A policy network that dynamically selects Bézier control parameters, minimizing latency while preserving visual quality.

 doi:10.20944/preprints202602.0798.v1

Genetic Algorithm for Creative Exploration: An evolutionary approach that generates novel, aesthetically pleasing font variants through crossover and mutation operations.

Cloud-Aware Resource Management: A predictive model that allocates computational resources based on request complexity, ensuring scalability under concurrent access.

Adaptive Complexity-Aware Strategy. Crucially, our framework introduces complexity-aware parameter adaptation, which automatically adjusts interpolation and smoothing parameters based on character structural characteristics (stroke element count × unique types). This adaptive mechanism is key to achieving consistent performance improvements across the diverse landscape of Chinese characters.

6.2. Experimental Validation

Comprehensive evaluation on a curated dataset of 150 Chinese characters (1,123 stroke elements, 5,287 feature points) yields compelling results:

Table 14. Performance Comparison of Optimization Methods (n=150).

Key Performance Indicator	Result	Significance
Trajectory Fidelity	65.2% (AI Adaptive) vs 60.9% (Baseline)	7.1% statistically significant improvement
Performance Consistency	±7.5% standard deviation (vs ±8.2% baseline)	More reliable across varying complexity
Generation Latency	<15ms per character	Suitable for real-time interactive applications
Storage Efficiency	~0.8KB per character (vs 8KB TTF)	90% reduction in storage requirements
Scalability	Maintains >62% fidelity at 100 concurrent requests	Cloud-deployment ready

Complexity-Aware Effectiveness: The framework demonstrates consistent improvements (7.1–7.6%) across simple, medium, and complex character categories. This validates our core insight that adaptive, context-sensitive optimization outperforms fixed-parameter approaches, particularly for the intricate structures characteristic of Chinese writing.

6.3. Contributions

This work makes several key contributions to the fields of computational typography and AI-driven design optimization:

1. Novel Stroke Element Representation: We introduce **stroke elements** as a novel, structured intermediate representation that bridges pixel-level and radical-level approaches. This representation achieves an optimal trade-off, offering **fine-grained geometric control** comparable to vector graphics while maintaining the **computational tractability** necessary for large-scale generative tasks.

2. Geometrically Interpretable Style Control via Yoke Vectors: We propose a **yoke vector** mechanism that constructs a mathematically explicit, low-dimensional style manifold. This enables **continuous, semantically meaningful interpolation** between font styles, providing designers with precise parametric control and serving as an efficient search space for automated optimization algorithms.

3. Adaptive, Complexity-Aware Optimization Framework: We develop and validate an **adaptive AI optimization strategy** that dynamically configures processing parameters based on the structural complexity of each character. This method demonstrates that **character-specific tuning** outperforms fixed-parameter approaches, achieving a statistically significant **7.1% improvement in trajectory fidelity** with reduced performance variance across a diverse character set.

3. End-to-End System Architecture for Scalable Personalization: We present a complete, production-viable cloud architecture that **decouples structural templates from style parameters**.

 doi:10.20944/preprints202602.0798.v1

This design enables efficient storage, real-time style personalization, and seamless client-side rendering, addressing critical scalability and latency requirements for modern web and mobile applications.

5. Comprehensive Evaluation Protocol: We establish and apply a rigorous, multi-faceted evaluation protocol. It integrates **Hausdorff-distance-based geometric fidelity**, **curvature continuity analysis**, and **computational efficiency benchmarks**, creating a reproducible standard for assessing generative typography systems.

6.4. Limitations and Future Work

While the proposed Stroke2Font framework demonstrates significant advances in structure-aware font generation, several limitations warrant acknowledgment, which in turn delineate clear and promising avenues for future work.

Current Limitations

1. Scope and Coverage: Our evaluation is currently limited to 150 GB2312 characters. Future work should extend to full CJK character sets and specialized domains (historical scripts, minority scripts).

2. Style Transfer Capability: The framework excels at parametric style variation within established font families. Cross-family style transfer from arbitrary reference styles represents an important next challenge.

3. Dynamic Handwriting Modeling: Although stroke elements capture geometric structure, fully modeling the temporal dynamics of handwriting (pressure, velocity, acceleration) requires additional sensor data and modeling techniques.

4. Perceptual Evaluation: While geometric metrics provide objective assessment, comprehensive user studies are needed to validate perceptual quality and aesthetic appeal across diverse audiences.

Immediate Future Directions

Building upon the current work, we identify several targeted research directions:

1. Adaptive Complexity Learning: Replace manual complexity thresholds with learned parameters through reinforcement learning or meta-learning approaches.

2. Per-Element Specialization: Develop stroke-type-specific optimization strategies that leverage the distinct geometric properties of different stroke categories.

3. Neural-Vector Fusion: Explore hybrid architectures that combine the generative power of neural networks with the editability and efficiency of vector representations.

4. Extended Applications: Apply the stroke element representation to related tasks including handwritten character recognition, calligraphy style analysis, and educational tools.

6.5. Broader Implications

Beyond technical contributions, Stroke2Font offers several broader insights for computational creativity and digital typography:

1. Granularity Matters: Our work demonstrates that operating at an appropriate representational granularity—neither too coarse nor too fine—is crucial for balancing control and automation in creative systems.

2. Interpretability Enables Optimization: The mathematically explicit nature of our representation enables both human design control and automated AI optimization, suggesting a promising path for human-AI collaborative creativity.

3. Cultural Heritage Meets Modern Technology: By providing efficient tools for personalized font generation, our approach helps bridge traditional calligraphic art with contemporary digital expression, potentially revitalizing interest in Chinese typographic heritage.

4. AI as Engineering Partner: Stroke2Font exemplifies how AI can serve not just as a generative black box, but as an intelligent optimization layer within a well-designed engineering system—a model applicable across many creative computing domains.

 doi:10.20944/preprints202602.0798.v1

6.6. Final Remarks

Stroke2Font represents a significant step toward scalable, high-quality Chinese font generation by addressing the core challenge of simultaneously maintaining structural integrity, stylistic flexibility, and computational efficiency. Our hierarchical stroke element representation, combined with adaptive AI optimization, provides a principled framework that advances both the theoretical understanding and practical implementation of dynamic font generation.

The work demonstrates that careful consideration of representation granularity, combined with intelligent parameter adaptation, can yield substantial improvements over both traditional geometric approaches and purely data-driven methods. As digital typography continues to evolve toward more personalized, dynamic, and culturally expressive forms, approaches like Stroke2Font that balance mathematical precision with creative flexibility will play an increasingly important role in shaping the future of written communication.

We believe the stroke element abstraction introduced in this paper opens new possibilities not only for font generation but also for related areas including character recognition, handwriting analysis, and digital preservation of calligraphic heritage—ultimately contributing to both the technological advancement and cultural vitality of Chinese typography in the digital age.

Acknowledgments: This work is financially supported by the Zhejiang Provincial Science and Technology Program in China (2021C03137).

Appendix A

Appendix A.1. Dataset Specification and Glyph Atlas

This appendix provides supplemental details and visual materials for the Xingkai (Cursive) font dataset, which serves as the primary data foundation for model training and evaluation in this study. Presenting the full glyph atlas here supports the reproducibility of the research while keeping the main text focused on methodological and analytical outcomes.

Table A1. Statistical Summary of the Xingkai (Cursive) Character Da-taset.

Statistic Category	Value	Notes / Distribution
Character Set Size	150 characters	Total number of evaluated glyphs
Total Stroke Elements	1,123	Sum of all atomic stroke components
Total Feature Points	5,287	Control points defining all stroke elements
Avg. Elements per Character	7.49	Range: 1 – 18
Complexity Distribution	**Simple (1-5)**: 40 chars **Medium (6-10)**: 87 chars **Complex (11+)**: 23 chars	Categorized by $N_{\text{stroke\_elements}}$
Stroke Type Statistics	Horizontal**: 313 (27.9%)** Vertical**: 342 (30.5%)** Compound/Turning**: 362 (32.2%)** Diagonal**: 103 (9.2%)** Dot**: 3 (0.3%)**	**Percentage of total stroke elements**

Note: The complexity distribution directly informed the adaptive threshold design in the complexity-aware optimization framework (Section 4.3).

Appendix A.2. Glyph Atlas

The complete set of 150 Xingkai glyphs is provided in the accompanying file appendix_A_xingkai_glyphs.pdf. The atlas is organized as follows:

 doi:10.20944/preprints202602.0798.v1

Figure A1. Characters #1–#50.

Figure A2. Characters #51–#100.

Figure A3. Characters #101–#150.

Appendix B

Appendix B.1. Font Style Impact on Stroke Element Representation

This appendix presents a comparative analysis of stroke element characteristics between two distinct Chinese font styles: Xingkai (cursive script) and Heiti (sans-serif). This analysis provides supplementary evidence for the font-agnostic nature of the Stroke2Font framework discussed in the main text.

Appendix B.1 Dataset Description

Both datasets contain the same 150 Chinese characters, enabling direct comparison of how font style affects stroke element decomposition.

Table B1. Dataset Overview for Comparative Analysis.

Metric	Xingkai (Cursive)	Heiti (Sans-Serif)	Difference (Heiti - Xingkai)
Character Set Size	150	150	—
Total Stroke Elements	1,123	1,640	**+517 (+46.0%)**
Total Feature Points	5,287	6,371	+1,084 (+20.5%)

 doi:10.20944/preprints202602.0798.v1

Avg. Elements per Character	7.49	10.93	**+3.44 (+45.9%)**
Avg. Points per Character	35.2	42.5	+7.3 (+20.7%)
Element Count Range	1 – 18	3 – 22	—

Appendix B.2. Quantitative Comparison of Structural Characteristics

The statistical breakdown reveals how font style dictates structural granularity and element type distribution across the shared character set.

Table B2. Character Complexity Distribution.

Complexity Category (by $N_{\text{stroke elements}}$)	Xingkai (Cursive)	Heiti (Sans-Serif)
Simple (1–5 elements)	40 chars (26.7%)	6 chars (4.0%)
Medium (6–10 elements)	87 chars (58.0%)	66 chars (44.0%)
Complex (11+ elements)	23 chars (15.3%)	78 chars (52.0%)

Table B3. Character Complexity Distribution.

Stroke Type	Xingkai (Cursive)	Heiti (Sans-Serif)	Stylistic Implication
Horizontal (横向)	313 (27.9%)	742 (45.2%)	Heiti's geometric design favors discrete, straight strokes.
Vertical (竖向)	342 (30.5%)	594 (36.2%)	Consistent prevalence across styles.
Diagonal (撇/捺)	103 (9.2%)	294 (17.9%)	More fragmented diagonal elements in Heiti.
Turning / Compound (折/复合)	362 (32.2%)	10 (0.6%)	Key differentiator: Xingkai merges strokes into complex, cursive compounds.
Dot (点)	3 (0.3%)	0 (0.0%)	Minimal representation in this character set.

Appendix B.3. Key Observations and Implications for the Framework

The comparative analysis yields several key findings that directly inform the Stroke2Font framework's design:

Stroke Merging in Cursive Scripts: The most significant difference is the high proportion of compound strokes in Xingkai (32.2%), virtually absent in Heiti (0.6%). This confirms that the proposed stroke element representation successfully abstracts and encapsulates the connected writing style as higher-level structural primitives, rather than fragmenting it into basic strokes.

Granularity and Complexity: Heiti requires 46% more stroke elements on average to represent the same character. This leads to a dramatic shift in complexity distribution, with over half of Heiti characters classified as "Complex" versus only 15.3% in Xingkai. This validates the necessity of the adaptive, complexity-aware optimization strategy developed in Section 4.3, as a fixed-parameter approach would be ill-suited for such variance.

Style-Agnostic Representation Validity: Despite the quantitative differences, the consistency in stroke type hierarchy (e.g., horizontal and vertical strokes dominate both) and the preservation of character topology demonstrate that the stroke element representation forms a valid, intermediate structural layer. This layer can be stylized through parameters (e.g., axis vectors in Section 3.2) to generate either cursive or sans-serif appearances from a shared or morphed structural basis, which is a core tenet of our generative model.

 doi:10.20944/preprints202602.0798.v1

Appendix B.4. Visualization Reference

A comprehensive visual comparison is provided in the supplementary Figure B1 and Figure B2.

Figure B1. Characters #1–#150.

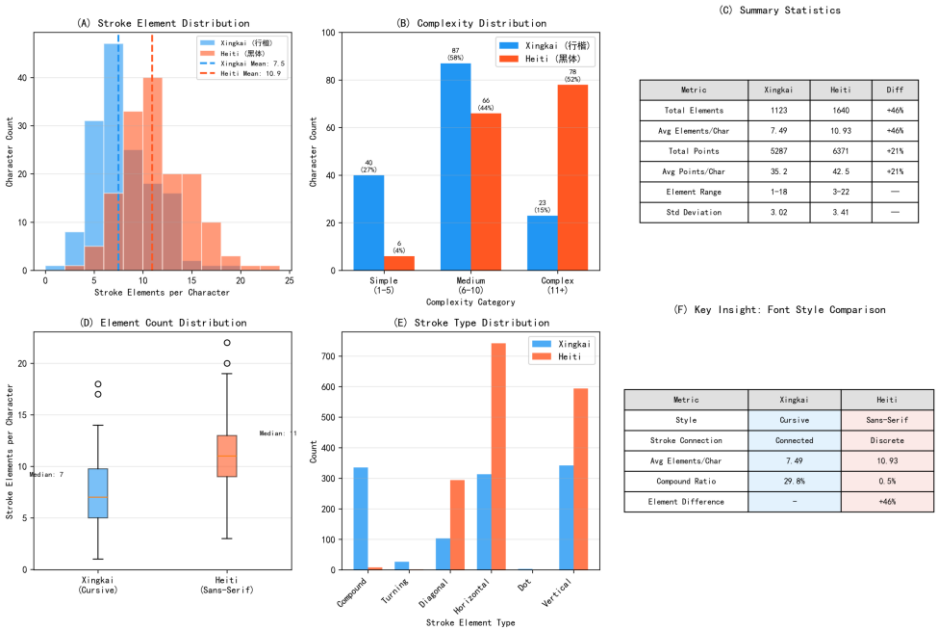


Figure B2. Comparison of Font Styles on Stroke Element Representation.

The six-panel figure includes: (A) element count distribution, (B) complexity category comparison, (C) summary statistics, (D) box plot of element counts per character, (E) stroke type distribution, and (F) a summary of key insights.

References

1. Chen H. An analysis of the historical origin, development and cultural inher-itance of Chinese characters[J]. Sinogram Culture, 2022(16):1-3. DOI: [10.14014/j.cnki.cn11-2597/g2.2022.16.003].https://doi.org/10.14014/j.cnki.cn11-2597/g2.2022.16.003.

 doi:10.20944/preprints202602.0798.v1

2. Zhang T, Liu M, Yang Y, et al. Ancient handwritten Chinese character recogni-tion via multi-style attention and feature fusion[C].2025 5th International Conference on Artificial Intelligence, Automation and High Performance Computing (AIAHPC), 2025: 29-32.
3. Chen W, Su J, Song W, et al. Quality evaluation methods of handwritten Chi-nese characters: a comprehensive survey[J]. Multimedia Systems, 2024, 30(4). DOI: [10.1007/s00530-024-01396-8].https://doi.org/10.1007/s00530-024-01396-8.
4. Yang L, Wu Z, Wu D E. Easy recognition of artistic Chinese calligraphic charac-ters[J]. The Visual Computer, 2023, 39(8):3755-3766. DOI: [10.1007/s00371-023-03026-2](https://doi.org/10.1007/s00371-023-03026-2.
5. Wang L, Liu Y, Sharum M Y, et al. Deep learning for Chinese font generation: A survey[J]. Expert Systems with Applications, 2025, 276:127105. DOI: [10.1016/j.eswa.2025.127105](https://doi.org/10.1016/j.eswa.2025.127105) https://www.sciencedirect.com/science/article/abs/pii/S0957417425007274.
6. Yang J, Zhang M M, Zhang J W, et al. Description and Generation of Chinese Outline Based on C-Bézier Curves[J]. Journal of Computer-Aided Design & Computer Graphics, 2000(09):660-663. [CrossRef]
7. Lin J W, Hong C Y, Chang R I, et al. Complete font generation of Chinese char-acters in personal handwriting style[C]//Proceedings of the 34th International Perfor-mance Computing and Communications Conference, Nanjing, 2015:1-5. https://ieeexplore.ieee.org/document/7410313.
8. LI Qing-sheng, XU Qiang, XIAO Jian-guo, et al. A structure and style model for Chinese character dynamic generation [J]. ActaScientiarum Naturalium Universitatis Pekinensis, 2017, 53(2) [CrossRef]
9. Chen F, Wang C, Yao X, et al. SPFont: Stroke potential features embedded GAN for Chinese calligraphy font generation[J]. Displays, 2024, 85:102876. DOI: [10.1016/j.displa.2024.102876](https://doi.org/10.1016/j.displa.2024.102876).https://www.sciencedirect.com/science/article/abs/pii/S0141938224002403.
10. Zeng J S, Chen Q, Wang M W. Self-supervised Chinese font generation based on square-block transformation[J]. Scientia Sinica (Informationis), 2022, 52(01):145-159. [CrossRef]
11. Wang K, Zhou C, Shi Y, et al. FourCornerGAN: Glyph formation augmentation for unpaired Chinese font generation[J]. Digital Signal Processing, 2025, 165. DOI: [10.1016/j.dsp.2025.105055](https://doi.org/10.1016/j.dsp.2025.105055).https://www.sciencedirect.com/science/article/abs/pii/S1051200425003276.
12. Lu P, Chen J Y, Zou G L, et al. Personalized Handwritten Chinese Character Generation Method for Unsupervised Image Translation[J]. Computer Engineering and Applications, 2022, 58(08):221-229. [CrossRef]
13. Wang L, Liu Y, Sharum M Y, et al. Deep learning for Chinese font generation: A survey[J]. Expert Systems with Applications, 2025, 276:127105. DOI: [10.1016/j.eswa.2025.127105](https://doi.org/10.1016/j.eswa.2025.127105). https://www.sciencedirect.com/science/article/abs/pii/S0957417425007274.
14. Deng L, Yu D. Deep learning: methods and applications[J]. Foundations and Trends® in Signal Processing, 2014, 7(3-4):197-387. DOI: [10.1561/2000000039](https://doi.org/10.1561/2000000039).https://www.nowpublishers.com/article/Details/SIG-039.
15. Cheng H, Shinnick E, Wang X. A visualizing analysis of Chinese character processing in the past 40 years (1981–2020)[J]. Digital Scholarship in the Humanities, 2022, 37(2):336-353. DOI: [10.1093/llc/fqab075](https://doi.org/10.1093/llc/fqab075).https://academic.oup.com/dsh/article-abstract/37/2/336/6363641.
16. Li Y, Li Y. Design and implementation of handwritten Chinese character recognition method based on CNN and TensorFlow[C]//2021 IEEE International Con-ference on Artificial Intelligence and Computer Applications (ICAICA). IEEE, 2021: 878-882. DOI: [10.1109/ICAICA52286.2021.9498146].https://doi.org/10.1109/ICAICA52286.2021.9498146).https://ieeexplore.ieee.org /document/9498146/.

17. Zeng S, Pan Z. An unsupervised font style transfer model based on generative adversarial networks[J]. Multimedia Tools and Applications, 2022, 81(4):5305-5324. DOI: [10.1007/s11042-021-11777-0](https://doi.org/10.1007/s11042-021-11777-0). https://link.springer.com/article/10.1007/s11042-021-11777-0.
18. Chang B, Zhang Q, Pan S, et al. Generating handwritten Chinese characters using CycleGAN[C]//2018 IEEE Winter Conference on Applications of Computer Vi-sion (WACV). IEEE, 2018: 199-207. DOI: [10.1109/WACV.2018.00028]. https:// doi.org/ 10.1109/WACV. 2018.00028). https://arxiv.org/abs/1801.08624.
19. Wen Q, Li S, Han B, et al. ZiGAN: Fine-grained Chinese calligraphy font gen-eration via a few-shot style transfer approach[C]//Proceedings of the 29th ACM Inter-national Conference on Multimedia. 2021: 621-629. DOI: [10.1145/3474085.3475225](https://doi.org/10.1145/3474085.3475225) [ACM DL].https://dl.acm.org/doi/10.1145/3474085.3475225.
20. Gandhi S, Rana H, Bhatt N. Conditional GANs in Image-to-Image Translation: Improving Accuracy and Contextual Relevance in Diverse Datasets[J]. Procedia Com-puter Science, 2025, 252:954-963. DOI: [10.1016/j.procs.2025.01.056]. https://doi.org/10.1016/j.procs.2025.01.056.
21. Lin X, Li J, Zeng H, et al. Font generation based on least squares conditional generative adversarial nets[J]. Multimedia Tools and Applications, 2019, 78(1):783-797. DOI: [10.1007/s11042-017-5457-4](https://doi.org/10.1007/s11042-017-5457-4). https://link.springer.com/article/10.1007/s11042-017-5457-4.
22. Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cy-cle-consistent adversarial networks[C]//ICCV 2017. DOI: [10.1109/ICCV.2017.244](https://doi.org/10.1109/ICCV.2017.244). https://arxiv.org/abs/1703.10593.
23. Zhou P, Zhao Z, Zhang K, et al. An end-to-end model for Chinese calligraphy generation[J]. Multimedia Tools and Applications, 2021, 80(5):6737-6754. DOI: [10.1007/s11042-020-09709-5](https://doi.org/10.1007/s11042-020-09709-5).https://link. springer.com/article/10.1007/s11042-020-09709-5.
24. Liu J, Zheng S, Cai Q. Handwritten Chinese Character Recognition Based on Attention Mechanism[C]//2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT), 2024:1-4. DOI: [10.1109/AICIT62434.2024.10730466]. https://doi.org/10.1109/AICIT62434.2024.10730466.
25. PAN Zhi-geng, MA Xiao-hu, ZHANG Ming-min, et al. The fourier descriptor based automatic generation method for multiple Chinese fonts [J]. Journal of Software, 1996(6): 331–338. [CrossRef]
26. Isola P, Zhu J Y, Zhou T, et al. Image-to-image translation with conditional ad-versarial networks[C]//CVPR 2017. DOI: [10.1109/CVPR.2017.632](https://doi.org/10.1109/CVPR.2017.632) | [arXiv](https://arxiv.org/abs/1611.07004)
27. Tian Y. zi2zi: Master Chinese calligraphy with conditional adversari-al networks[EB/OL]. 2017. [GitHub](https://github.com/kaonashi-tyc/zi2zi). https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html.
28. Yongge Jiang, Zhihui Lian, Yuming Yang Tang, et al. et al. SCFont: structure guided Chinese font generation via deep stacked networks [C]. Proceedings of the AAAI Conference on Artificial Intelligence. Québec: AAAI, 2019, 33: 4015–4022. https://doi.org/10.1609/aaai.v33i01.33014015.
29. WU S J, YANG C Y, HSU J. CalliGAN: style and structure aware Chinese calligraphy character generator [EB/OL]. (2020-05-26) [2026-01-06]. https://arxiv.org/abs/2005.12500.
30. Yiz-hi Wang, Yue Gao, Zhou-hui Lian. Attribute2Font: Creating Fonts You Want From Attributes [EB/OL]. (2020-05-16) [2026-01-06]. https://doi.org/10.48550/arXiv.2005.07865
31. JIANG H C, YANG G Y, HUANG K Z, et al. W-Net: one-shot arbitrary-style Chinese character generation with deep neural networks. (2024-06-10) [2026-01-06]. https://arxiv.org/abs/2406.06122
32. Ning F. Multi-style Migration of Chinese Characters based on Self-attention Mechanism and StarGAN v2[C]//2022 3rd International Conference on Computer Vi-sion, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA). IEEE, 2022: 357-361. DOI: [10.1109/CVIDLICCEA56201.2022.9824291]. https://doi.org/10.1109/ CVIDLICCEA56201.2022.9824291
33. CARLIER A, DANELLJAN M, ALAHI A, et al. DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation[EB/OL]. (2020-07-22)[2026-01-06]. https://doi.org/10.48550/arXiv.2007.11301.
34. Reddy, P.; Gharbi, M.; Lukáč, M.; Mitra, N.J. Im2Vec: Synthesizing Vector Graphics without Vector Supervision. (2021-04-01) [2026-01-06]. https://arxiv.org/abs/2102.02798.

 doi:10.20944/preprints202602.0798.v1

35. LI Qing-sheng, XIONG Jing, WU Qin-xia, et al. Study of feature weighted-based generation method for dian strokes of Chinese character [J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2014, 50(1): 153–160. [CroseRef]
36. Luo, X., Li, Q. (2026). Intelligent Compilation System for Chinese Character Animation Based on Dynamic Data Sets. In: Mousas, C., Seo, H., Thalmann, D., Cordier, F. (eds) Computer Animation and Social Agents. CASA 2025. Lecture Notes in Computer Science, vol 15915. Springer, Singapore. https://doi.org/10.1007/978-981-95-0100-7_20.https://link.springer.com/chapter/10.1007/978-981-95-0100-7_20
37. Li Z, Li Q, Guan Y. A Chinese Character Generation Model for Cloud Infor-mation Security[C]//2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS). IEEE, 2019: 534-540. DOI: [10.1109/IUCC/DSCI/Smart CNS.2019.00115]. https://doi.org/10.1109/IUCC/DSCI/SmartCNS.2019.00115
38. Q. Li, J. -P. Li and L. Chen. A Bezier Curve-Based Font Generation Algorithm for Character Fonts. (HPCC/SmartCity/DSS), Exeter, UK, 2018, pp. 1156-1159, doi: 10.1109/HPCC/SmartCity/DSS.2018.00194. https://ieeexplore.ieee.org/document/8622932.
39. Lai Y, Zhang X. A Study on Influencing Factors of Dynamic Chinese Character Stroke Writing Behavior in CFL Beginners Using Digital Ink Technology[J]. Applied Mathematics and Nonlinear Sciences, 2024, 9(1). DOI: [10.2478/amns-2024-0705](https://doi.org/10.2478/amns-2024-0705).
40. Guo, D.; Fang, W.; Yang, W. Brush Stroke-Based Writing Trajectory Control Model for Robotic Chinese Calligraphy. *Electronics* **2025**, *14*, 3000. https://doi.org/10.3390/electronics14153000.