ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Handwriting Robot With User-Customized Font Style

Team #35

ZHIXIANG LIANG
(zliang18@illinois.edu)
ZIFAN YING
(zifany4@illinois.edu)
XUANCHENG LIU
(zl124@illinois.edu)
MINGCHEN SUN
(msun52@illinois.edu)

TA: Tielong Cai, Tianci Tang
Supervisor: Prof. Gaoang Wang

May 20, 2025

Abstract

This project presents an automated system capable of replicating a person's unique handwriting style with high fidelity. By integrating machine learning-based handwriting analysis with a robotic writing mechanism, the system enables personalized document creation for applications such as secure document signing, artistic reproduction, and individualized correspondence. The system architecture consists of three main subsystems: a handwriting sample analysis module, a few-shot font generation model, and a robotic handwriting actuator. The process begins by analyzing and segmenting handwritten samples into standardized grayscale images, which are then used to train a machine learning model that captures the individual writing style. The resulting stroke data is transmitted to a robotic subsystem equipped with high-resolution stepper motors and an ESP32-based controller, which executes precise pen movements according to standard G-code instructions. This approach enables efficient and accurate replication of personal handwriting using a compact and programmable hardware setup.

Keywords: Handwriting Replication, Few-shot Learning, Robotic Writing System.

Contents

1 Introduction **1**
1.1 Overivew . . . 1
1.2 Visual-Aid . . . 2
1.3 High-Level Requirements . . . 2

2 Design **4**
2.1 Block Diagram . . . 4
2.2 Physical Design . . . 5
2.3 Subsystem Overview . . . 7
2.3.1 Few-shot Font Generation System . . . 8
2.3.2 Pen Trajectory Extraction System . . . 10
2.3.3 Robotic Handwriting Subsystem . . . 12
2.3.4 Communication and Control Subsystem . . . 14
2.3.5 User Interface Subsystem . . . 17

3 Cost and Schedule **18**
3.1 Cost Analysis . . . 18

4 Discussion of Ethics and Safety **20**
4.1 Ethical Considerations . . . 20
4.2 Safety Considerations . . . 20

References **22**

Appendix A Supplementary Requirements and Verification Table **23**

1 Introduction

1.1 Overivew

Handwriting remains a personal and unique form of expression, yet current digital and automated writing solutions lack the ability to accurately replicate individual handwriting styles. Existing methods either rely on digital fonts that imitate handwriting or require complex, manual customizations [1]. This project addresses the need for an automated system that can learn and reproduce a person's unique handwriting style with high fidelity [2]. By integrating machine learning-based handwriting analysis with robotic writing mechanisms, this system enhances document personalization, enabling applications in personalized correspondence, secure document signing, and artistic reproduction.

In this project, our solution consists of a complete pipeline integrating three core subsystems: a handwriting sample analysis subsystem, a few-shot font generation subsystem, and a robotic handwriting mechanism. Initially, the user's handwritten samples are processed and segmented into standardized grayscale images, serving as input to a few-shot font generation system. Using advanced machine learning techniques, this subsystem extracts and learns the user's handwriting style. Finally, generated stroke data is sent to a robotic handwriting subsystem, which physically replicates the handwriting using precise motor control.

The robotic handwriting subsystem comprises a two-axis rail system driven by high-resolution stepper motors and an additional axis for pen actuation. Motor control signals are generated by an ESP32 module, which also manages data communication with the host computer via UART, Bluetooth, or WLAN. Stroke data transmitted to the ESP32 follow the standard G-code path format, specifying precise movement commands (coordinates and pen lift/drop states).

1.2 Visual-Aid

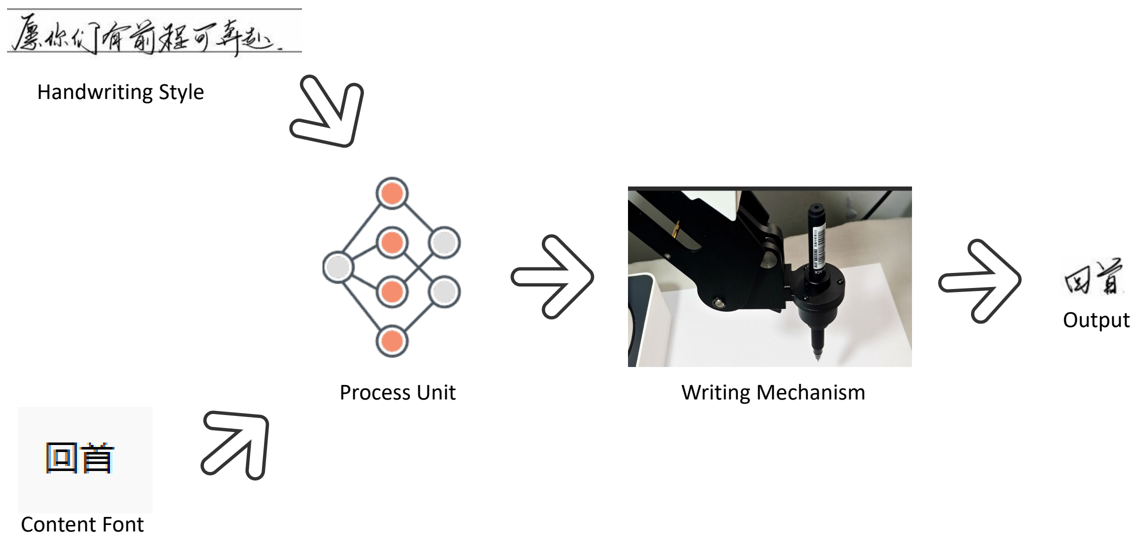


Figure 1: The visual illustration of our project: a user's handwriting sample and content text are processed through a neural network, enabling a robotic arm to reproduce the text in the user's unique handwriting style.

1.3 High-Level Requirements

Few-shot Font Generation System

- The font generation inference should process each character in under 500 milliseconds, ensuring efficiency for practical usage scenarios.
- The system should reliably handle variations in handwriting input quality, maintaining style fidelity with less than 5% performance degradation in the presence of moderate image noise or distortion.

Robotic Handwriting Subsystem

- The subsystem must reproduce handwriting with positional accuracy within ±0.1 mm to achieve clear and authentic handwriting replication.
- Control signal synchronization must maintain a maximum timing deviation of less than 1 ms between ESP32 signals and motor response to ensure

smooth and continuous pen movements.

- Communication between the ESP32 module and the host computer must achieve a data transmission reliability greater than 99.9% with latency below 100 ms, ensuring real-time handwriting reproduction.

2 Design

2.1 Block Diagram

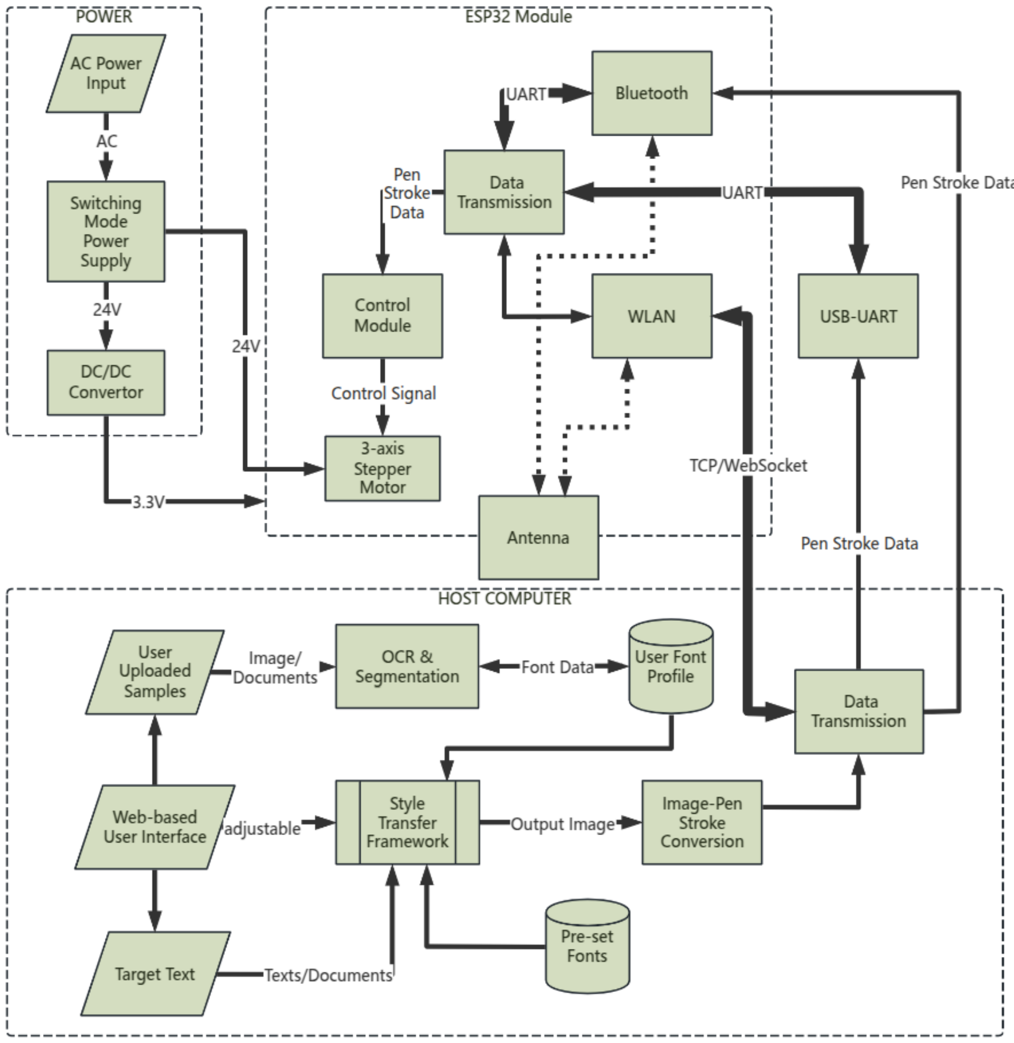
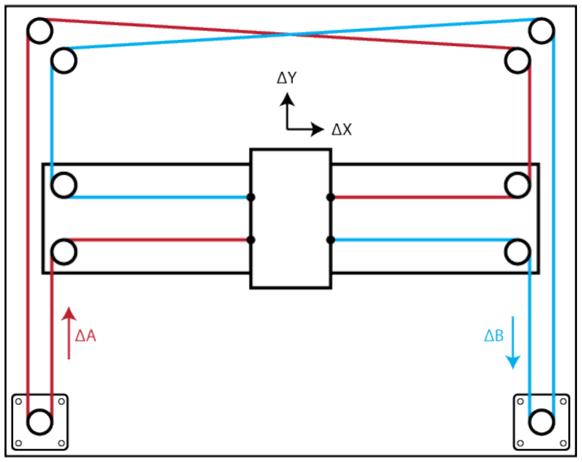


Figure 2: Block Diagram of Our System

Design Overview: The complete system is composed of four modular subsystems: (1) Few-shot Font Generation, (2) Image-to-Stroke Converter, (3) Robotic Handwriting Mechanism, and (4) Communication and Control. Each subsystem is independently testable and interfaces with well-defined protocols and formats (e.g., SVG/G-code, UART). These blocks collectively ensure the system meets all high-level requirements for fidelity, precision, and responsiveness.

2.2 Physical Design



Equations of Motion:

$\Delta X = \frac{1}{2}(\Delta A + \Delta B)$, $\Delta Y = \frac{1}{2}(\Delta A - \Delta B)$

$\Delta A = \Delta X + \Delta Y$, $\Delta B = \Delta X - \Delta Y$

Figure 3: Schematic of CoreXY Mechanism

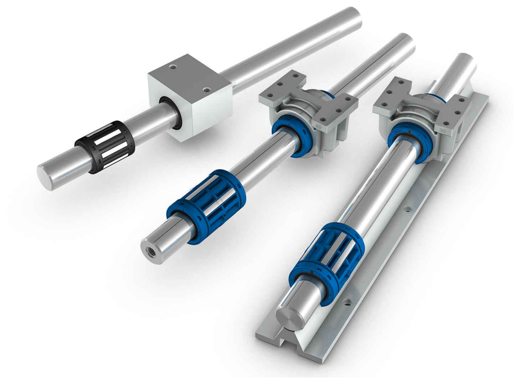


Figure 4: Linear axis and bearing

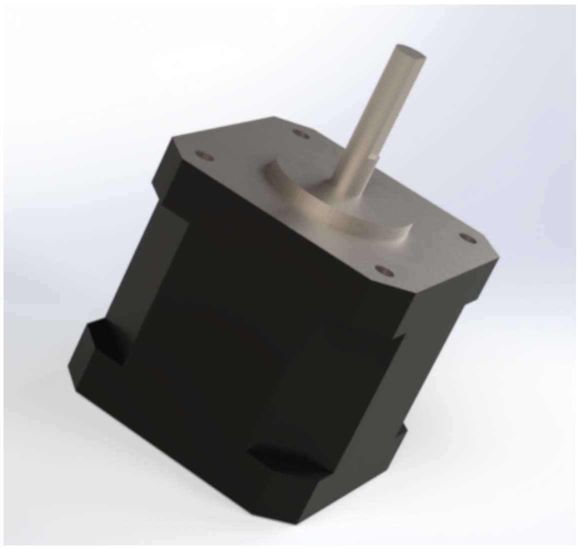


Figure 5: 42mm stepper motor

Mechanical Structure: The handwriting robot frame employs a CoreXY mechanism (shown in Fig. 3)for the X-Y planar motion, ensuring high speed and reduced inertia for better precision. A Z-axis stepper motor handles pen actuation. Stepper motors, belts, and linear rails are mounted on a rigid 3D-printed or aluminum frame. PCB mounts for the ESP32 and drivers are placed on the side, isolated from mechanical vibration. The linear rail consists of linear axis and linear bearings, as shown in Fig. 4.

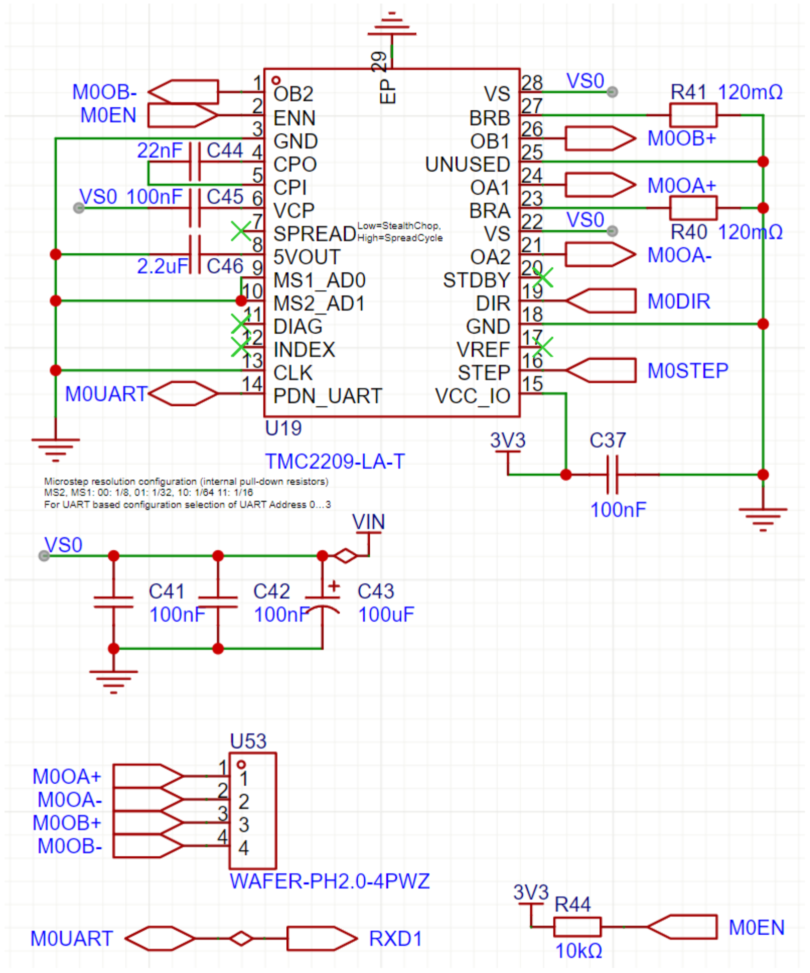


Figure 6: TMC2209 Circuit Schematic

PCB Design: It is a 2-layer PCB with ESP32 module and TMC2209 stepper driver. The TMC2209 circuit schematic is shown in Figure 6.

2.3 Subsystem Overview

The Handwriting Robot System is composed of the following main subsystems:

2.3.1 Few-shot Font Generation System

Our system utilizes a few-shot font generation approach to produce fonts that match the style specified by the user through a small number of sample fonts. Specifically, the font samples provided by the user are segmented into 128 × 128 grayscale images, which serve as references for style learning. Then, based on the content specified by the user, the system generates fonts that imitate the user's writing style as shown in Figure 7.

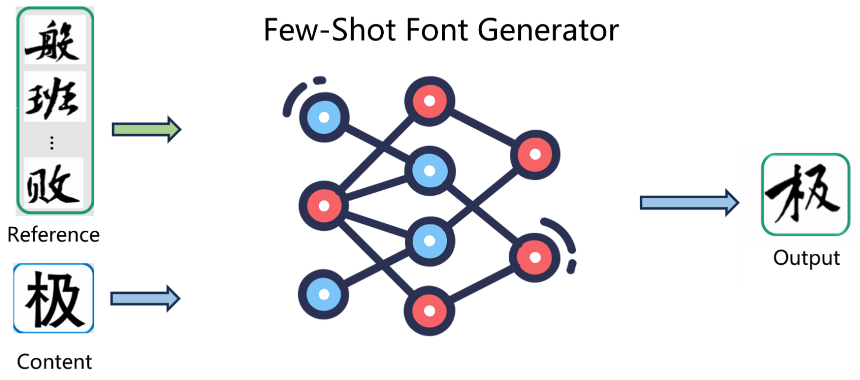


Figure 7: Overview of the Few-shot Font Generator

Building upon the established workflow of GAN-based [3] architectures and self-reconstruction frameworks as described in previous studies [4], our generator is capable of adapting to various scripts with minimal reliance on labeled data. The training process is divided into two main parts:

1. **Pre-train the content encoder and obtain component representations**:
   - The content encoder E_c takes the content image as input to extract its structural representation f_c. It is pre-trained via VQ-VAE [5] to obtain the component-wise latent codes e_c. The style encoder is designed to learn the style representations from reference images y.
   - The loss function used here consists of an L_1-based reconstruction loss and latent loss.
     $$L_{pre} = ||x_c - \hat{x}_c||_1 + \alpha||\text{sg}[z_e] - e_c||_2^2 + \beta||z_e - \text{sg}[e_c]||_2^2$$
   - After this pre-training task, we fix the content encoder E_c as well as the component codebook e_c to build the font generation model. The

style encoder E_s has the same architecture as E_c, but it is trained from scratch.

2. **Train the complete font generation model**:

   - The content encoder E_c and the component codebook e_c are kept fixed during this period. Generator G and Discriminator D are trained via adversarial learning.

     For Discriminator D:

     $$\mathcal{L}_{adv}^{D} = -\mathbb{E}_{\hat{y}_c \sim p_{data}} \min(0, -1 + D_{s,c}(\hat{y}_c)) - \mathbb{E}_{y_c \sim p_G} \min(0, -1 - D_{s,c}(y_c))$$

     For Generator G:

     $$\mathcal{L}_{adv}^{G} = -\mathbb{E}_{y_c \sim p_G} D_{s,c}(y_c)$$

     where p_G denotes the set of generated images, and p_{data} denotes the set of real glyph images.

   - To make the generated character y_c learn pixel-level and feature-level consistency with ground truth $\hat{y}_c$, we employ an L_1 loss on both image and image features as follows:

     $$\mathcal{L}_{img} = \mathbb{E}[\|y_c - \hat{y}_c\|_1]$$
     $$\mathcal{L}_{feat} = \mathbb{E}\left[\sum_{l=1}^{L} \left\|f^{(l)}(y_c) - f^{(l)}(\hat{y}_c)\right\|_1\right],$$

     where $f_l(y_c)$ and $f_l(\hat{y}_c)$ represent the intermediate features in the lth layer of D.

   - Given two different reference sets but sharing the same font style, the associated stylized components for these sets should be the same. If the reference character sets have different font styles, the component styles should be distinguished. The style contrastive loss is as follows:

     $$\mathcal{L}_{cst} = -\log\left(\frac{\exp\left(\sum_{n=1}^{N} e_s^{nT} e_{s+}^{n}\right)}{\exp\left(\sum_{n=1}^{N} e_s^{nT} e_{s+}^{n}\right) + \sum_{N_s} \exp\left(\sum_{n=1}^{N} e_s^{nT} e_{s-}^{n}\right)}\right),$$

     where $e_s^{nT} e_{s+}^{n}$ represents positive samples and $e_s^{nT} e_{s-}^{n}$ represents negative samples.

- Finally, we optimize the whole model by the following full objective function:

$$\min_{E_s,G}\max_{D}\mathcal{L}^{D}_{adv}+\mathcal{L}^{G}_{adv}+\lambda_1\mathcal{L}_{img}+\lambda_2\mathcal{L}_{feat}+\lambda_3\mathcal{L}_{cst}.$$

  The λ_1, λ_2 and λ_3 is weighting hyperparameter. We set them to 1, 1, and 0.1, respectively.

In the process of model training, we collected 230 Chinese fonts and generated 3,500 Chinese characters for each font according to the first-level Chinese Character Table. All character pictures are normalized to 128×128. A standard font template (FZGuoMJDTJW) is selected for extracting the glyph content features and it is also used for pre-training the VQ-VAE to get a set of common parts codebooks.

For training of the encoder, we set the embedding dimension d to 256, the batch size to 256, and the iteration steps to 50,000. Then, for training the whole font generation model, we set the batch size to 8, the number of attention heads to 8 in three stacked transformer layers, and iteration steps to 300,000.

2.3.2 Pen Trajectory Extraction System

Motivated by prior work on stroke-based handwriting synthesis [6], once the 128×128 grayscale character images are generated, we use a trained model to predict and extract the pen trajectory information necessary for downstream robotic writing execution. Specifically, we employ a CNN-LSTM backbone and incorporate dynamic window generation together with a differentiable cropping-pasting component, as show in Figure 8. At each time step t, the current canvas containing glyph I is convolved to infer a quadratic Bézier stroke [7],

$$B_t=(p,x_0,y_0,x_1,y_1,x_2,y_2,w_0,w_1)_t,$$

where p indicates the pen state, with $p=1$ for drawing and $p=0$ for lifting the utensil. The set $\mathcal{O}_i,\ i\in\{1,2,3\}$ denotes the control points, and w_0,w_1 represent the initial and final widths, respectively. We design several unsupervised loss functions to guide the stroke decomposition and generation process:

- **Perceptual Loss:** Measures the similarity between the rendered glyph and the target glyph using feature representations extracted from a pre-trained network.

$$\mathcal{L}_{\text{perc}}=\sum_{j\in J}\frac{1}{D_jH_jW_j}\left\|\phi_j(\hat{y})-\phi_j(y)\right\|_1$$

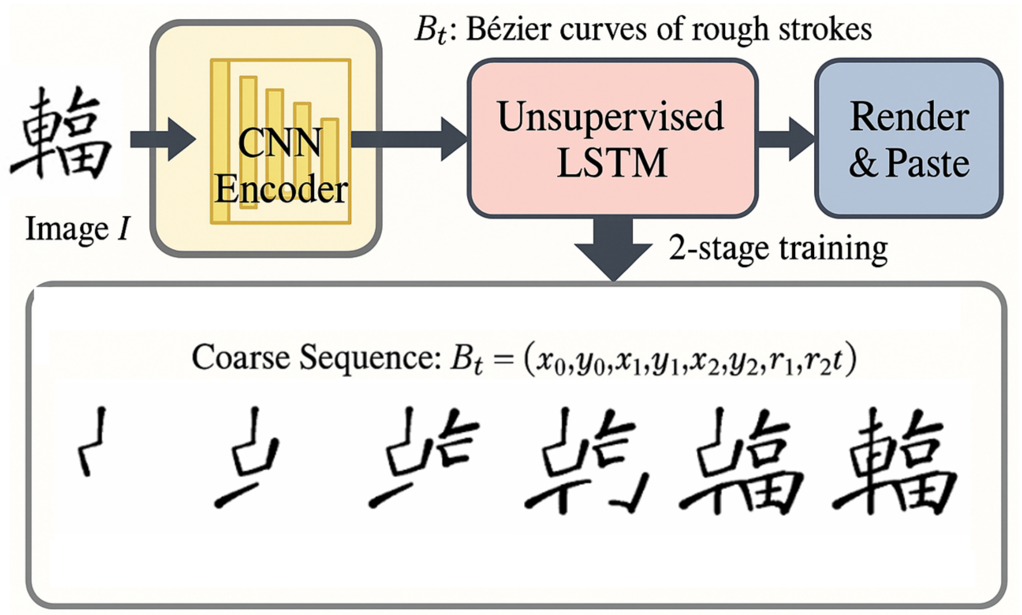

B_t: Bézier curves of rough strokes
Image I
CNN Encoder
Unsupervised LSTM
Render & Paste
2-stage training
Coarse Sequence: $B_t = (x_0, y_0, x_1, y_1, x_2, y_2, r_1, r_2 t)$

Figure 8: The framework for extracting the strokes of font images

- **Regularization Loss:** Penalizes redundant strokes by encouraging the model to minimize unnecessary lifting actions during stroke generation.

$$\mathcal{L}_{\text{reg}} = \frac{1}{T}\sum_{t=1}^{T}(1 - p_t)$$

- **Smoothness Loss:** Encourages continuity and smoothness between adjacent strokes by maximizing the cosine similarity of their directions.

$$\mathcal{L}_{\text{smo}} = \sum_{t=1}^{T} p_t \cdot \frac{1 + \mathcal{C}(\mathcal{O}_1\mathcal{O}_0,\, \mathcal{O}_1\mathcal{O}_2)}{2}$$

- **Angle Loss:** Constrains the starting direction of strokes to follow natural writing angles, ensuring realistic and physically plausible trajectories.

$$\mathcal{L}_{\text{ang}} = \frac{1}{T}\sum_{t=1}^{T}\begin{cases}\mathcal{C}(\vec{S}_t,\, \vec{a}), & \mathcal{C}(\vec{S}_t,\, \vec{a}) \geq 0.5\\ 0, & \mathcal{C}(\vec{S}_t,\, \vec{a}) < 0.5\end{cases}$$

The model is trained in two stages. In the first stage, model is first pre-trained on the QuickDraw dataset, which contains large-scale hand-drawn sketches with diverse styles. It learns basic decomposition abilities using perceptual and regularization losses:

$$\mathcal{L}_{\text{phase1}} = \mathcal{L}_{\text{perc}} + \lambda_1 \mathcal{L}_{\text{reg}}$$

In the second stage, the model is trained on glyph samples extracted from the KaiTi-GB2312 and KanjiVG datasets, both of which provide high-quality handwritten or typographic character decompositions. Smoothness and angle losses are incorporated to further refine the results:

$$\mathcal{L}_{\text{phase2}} = \mathcal{L}_{\text{perc}} + \mathcal{L}_{\text{reg}} + \lambda_2 \left(0.5\, \mathcal{L}_{\text{smo}} + \mathcal{L}_{\text{ang}}\right)$$

Here, λ_1 and λ_2 are coefficients that gradually increase during training to balance different loss terms. This two-stage dataset strategy helps the model learn both general stroke decomposition and detailed, style-specific features.

2.3.3 Robotic Handwriting Subsystem

The Robotic Handwriting Subsystem is designed to execute precise handwriting motions by controlling two-dimensional planar movements and the vertical positioning of the pen relative to the writing surface. It employs a CoreXY mechanical arrangement combined with a servo-driven Z-axis, which efficiently balances precision, speed, and mechanical simplicity.

- **CoreXY Motion Platform** The subsystem utilizes a CoreXY mechanism, which is characterized by its use of two stepper motors working in tandem via an arrangement of belts and pulleys. This approach significantly reduces the moving mass, as only the pen carriage moves, minimizing inertia and vibrations, and improving positional accuracy. Two NEMA 17 stepper motors (42mm size, 40mm height) rated at 24V, 1.5A, and providing a maximum torque of 0.4 Nm, with a standard step angle of 1.8 degrees (200 steps per revolution), drive the planar motion through 3M timing belts and 20-tooth pulleys. This configuration results in a theoretical spatial resolution of 0.1 mm per full step, enhanced further through microstepping capabilities.

- **Pen Actuation Mechanism (Z-axis)** Due to the minimal vertical travel required (approximately 5-10 mm) and the lightweight nature of the pen load, the Z-axis employs an RC servo motor (MG90S) combined with a slider-crank mechanism for compactness and simplicity. The MG90S servo operates based on PWM signals with a scale of 90° per millisecond pulse width variation and features a 5μs deadband. Thus, the highest theoretical angular resolution is approximately 0.45°, corresponding to a linear resolution of approximately 0.01875 mm. However, practical resolution is somewhat lower due to mechanical tolerances and servo precision.

- **Mechanical Frame and Bearings** All structural components are fabricated using 3D printing technology to facilitate rapid prototyping, customization, and lightweight construction. The axes are supported by linear ball bearings coupled with precision-ground shafts, specifically 8mm shafts for X-axis, and a lighter 6mm shaft for the Y and Z axes. This ensures smooth linear motion, minimal friction, and stable positioning of the writing tool. The CoreXY setup employs carefully aligned idler pulleys and belt tensioning systems to maintain geometric accuracy and consistent belt tension, thus minimizing backlash and ensuring repeatability.
- **Stepper Motor Drivers and Microstepping** To achieve precise and smooth motion control, the system utilizes Trinamic TMC2209 stepper drivers, renowned for their quiet operation and high microstepping accuracy. These drivers support configurable microstepping modes (8, 16, 32, or 64 steps) and feature advanced MicroPlyer™ interpolation technology, enabling up to 256 interpolated microsteps per full step. This microstepping significantly enhances smoothness, reduces audible motor noise, and provides a theoretical minimum positional resolution down to approximately 0.003125 mm per microstep when operating at 1/32 microstepping.
- **Trajectory Execution and Control** Movement trajectories are encoded as standardized G-code instructions generated by the handwriting analysis subsystem. The robotic subsystem's ESP32-based microcontroller, running FluidNC firmware, interprets these instructions and generates corresponding real-time step and direction signals. This setup allows efficient translation from digital handwriting style data into precise physical pen movements.

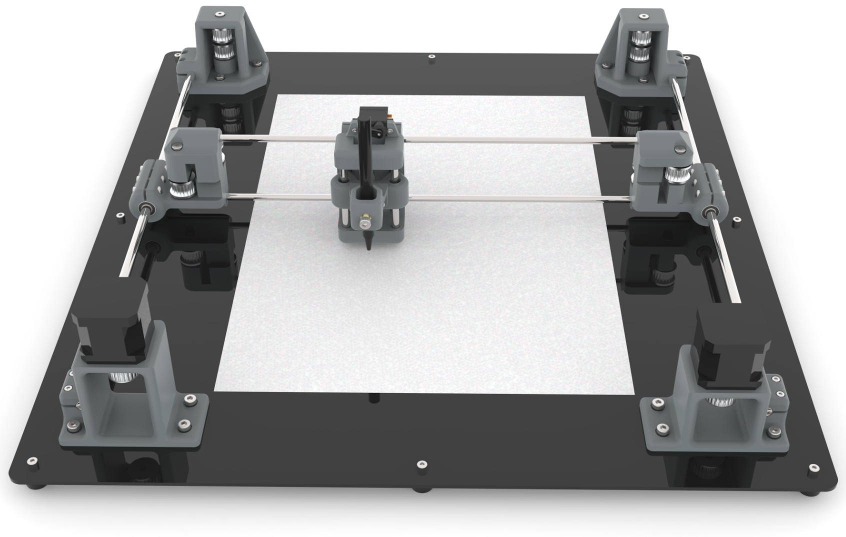


Figure 9: Robot rendering

Overall, this design balances simplicity, accuracy, and reliability, enabling high-fidelity robotic handwriting replication suitable for diverse applications ranging from personalized documents to secure signature reproduction.

2.3.4 Communication and Control Subsystem

The Communication and Control Subsystem is designed to manage real-time control, user interactions, and data handling necessary for the robotic handwriting replication process. At the heart of this subsystem is an ESP32-WROOM-32UE microcontroller, chosen for its robust processing capabilities, built-in wireless connectivity (WLAN), and flexibility.

Microcontroller and Firmware: The subsystem employs an ESP32-WROOM-32UE module with an external antenna, providing stable and extended wireless connectivity. It operates FluidNC firmware, a powerful CNC-oriented open-source firmware, optimized for real-time stepper motor control. FluidNC offers a comprehensive web-based user interface accessible via WLAN, functioning both as a Wi-Fi station and as an access point, enabling seamless remote interaction and control.

Power Management: The subsystem is powered through an external 24V input supply. Voltage regulation is accomplished using an AP63205 switching regulator, converting 24V to a stable 5V supply. The 5V output is protected and combined with USB VBUS through Schottky diode OR-ing circuitry, ensuring reliability and safe operation. Further regulation from 5V to 3.3V, essential for the ESP32 and logic-level components, is handled by an AMS1117 linear regulator.

USB Communication and Firmware Updates: USB communication and firmware updates are facilitated via a CH343P USB-to-UART bridge, using USB Type-C connectors. This setup supports reliable firmware flashing, configuration file updates, and real-time debugging.

Peripherals and Storage: The subsystem integrates a 128x64 OLED display module (SSD1106 driver) communicating via the I^{2}C protocol, providing immediate feedback and essential status information. Data and G-code files are stored on a MicroSD card formatted with a FAT32 file system, ensuring ample storage space and ease of file management.

Motor Driver Configuration: Three Trinamic TMC2209 stepper motor drivers manage the precise movement of the robotic subsystem. These drivers support motors up to 2.8A (peak) with multiple microstepping modes (8, 16, 32, or 64 steps). Key features include MicroPlyer™ interpolation for up to 256 microsteps, StealthChop2™ for silent operation, SpreadCycle™ for dynamic performance, and StallGuard4™ combined with CoolStep™ technology for load detection and energy-efficient motor control. Voltage input range is flexible from 4.75V to 29V DC, and the drivers include integrated diagnostics and protection circuits.

Servo Control: The ESP32 directly controls the pulse-width modulation (PWM) signals required for precise servo positioning of the Z-axis (pen lifting mechanism), ensuring smooth and accurate vertical motion.

Heat Management and Protection: Thermal management is achieved using dedicated heat dissipation fins for the TMC2209 drivers and the ESP32 module, supplemented by DC 5V cooling fans. Robust electrical safety and reliability are ensured by employing resettable fuses and transient voltage suppression (TVS) diodes for the USB, 24V supply, and 3.3V lines, providing comprehensive ESD protection.

Communication and User Interaction: User interaction and control are enhanced through FluidNC's web-based interface, allowing seamless G-code execution, real-time monitoring of axes positions, and direct filesystem access. Configuration files and firmware updates can be efficiently managed through UART connections, ensuring ease of setup, maintenance, and continuous system optimization.

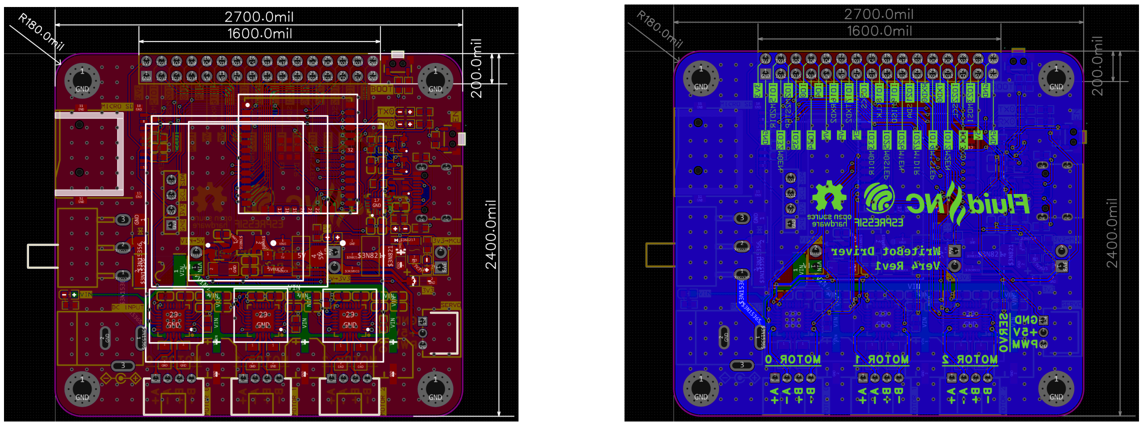


Figure 10: PCB Design: Front and Back Sides

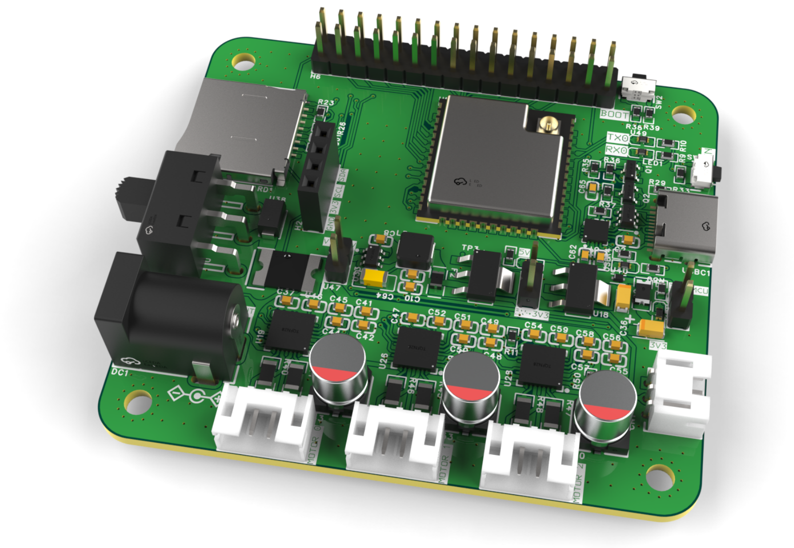


Figure 11: PCB rendering

2.3.5 User Interface Subsystem

The web-based user interface, developed using ReactJS and NodeJS, provides an intuitive and efficient platform for users to interact with the robotic handwriting subsystem. The interface comprises three main pages, each supporting both light and dark modes:

Multi-Character G-Code Editor: This page enables users to manage and refine pen stroke data effectively. Users can import pen stroke data from JSON files or real-time socket.io connections. Fine-tuning capabilities include applying low-pass filters and various interpolation methods such as CatMull-Rom and linear interpolation. Users can also adjust the X/Y coordinate ranges, scales, and apply random offsets. Customizable settings for pen lift/lower Z-values, G-code prefixes/suffixes, and feed rates further enhance usability. A live preview canvas provides immediate visual feedback, displaying strokes and G-code simulations dynamically.

Writing Pad: The Writing Pad allows users to input handwritten characters directly via a digital canvas. After each character, users identify the corresponding symbol. Upon collecting four characters, users enter text to generate the final output. Images are cropped into standardized 128-pixel squares and sent to the backend Few-shot Font Generation and Pen Trajectory Extraction Systems. Processed responses are then relayed to the Multi-Character G-Code Editor through the socket.io server.

Image Upload: This feature is similar to the Writing Pad, with users uploading images containing handwritten characters. Users draw bounding boxes around each character, specify their identities, and follow the same backend processing workflow. This page simplifies converting handwritten content from external sources into usable digital formats for robotic replication.

3 Cost and Schedule

3.1 Cost Analysis

Labor Cost: We assume a reasonable undergraduate engineering rate of \$40/hour. Each team member contributes approximately 100 hours over the course of the semester.

- Labor cost per member = \$40/hour $\times$ 100 hours = \$4,000
- Total labor cost (4 members): \$16,000

Part	**Part Number**	**Qty**	**Cost (CNY)**
Stepper Motors	42BYGH47 (42mm)	2	40.00
Belts	3M (3mm pitch)	4m	184.40
Pulleys	3M (3mm pitch)	1 set	0.00
Linear Ball Bearings	LM8UU (8mm& 6mm)	13	50.00
Linear Shafts		2m	18.50
RC Servo	MG90S	1	8.83
3D Printed Parts			0.00
DC Power Supply	24V5A	1	40.00
Screws			118.05
Soldering supplies		1 set	36.22
Lubricants	WD-40	1	44.41
Misc Hardware (Wires, Connectors, etc.)			247.15
Tools (Screwdriver, Pliers, etc.)		5	189.73
Laser stencil		3	62.00
Custom PCB		4	350.42
electronic components		4 sets	680.45
Total			**2070.16**

Table 1: Parts Cost Breakdown

Parts and Components:

Grand Total:

- Labor: $16,000
- Parts: $288
- **Total Estimated Cost: $16,288**

4 Discussion of Ethics and Safety

4.1 Ethical Considerations

The handwriting robot system presents several ethical considerations due to its ability to replicate personalized handwriting styles. One primary concern is the potential misuse for document forgery or impersonation. To mitigate this risk, we propose the following:

- Implement user authorization before accepting handwriting input or generating outputs.
- Embed subtle digital watermarks in stroke data to allow traceability of generated content.
- Restrict the use of this tool to offline, user-verified environments to avoid mass impersonation scenarios.

Another critical ethical aspect involves user data privacy. Handwriting is considered a form of biometric information. Therefore, we will:

- Obtain explicit consent for collecting and using user handwriting samples.
- Store all data locally, encrypted on the host system.
- Comply with data protection standards such as GDPR in handling personal biometric data.

4.2 Safety Considerations

The robot includes moving mechanical parts and electronic components that could pose risks during operation. Our safety strategy includes the following elements:

- **Mechanical Safety:** The CoreXY frame and motion subsystem will include limit switches at all axes and physical bumpers to prevent over-travel.
- **Emergency Stop:** A physical emergency stop button will be placed within easy reach to immediately disable all motor drivers.
- **Firmware Limits:** FluidNC firmware will be configured with software-defined maximum velocities, accelerations, and travel limits.
- **Electrical Safety:** The ESP32 control board and stepper motor drivers will be enclosed in a protective case. Wiring will be routed through secure channels

to prevent short circuits.

- **Component Durability:** Belts and rails will be regularly inspected for wear, and a maintenance log will be maintained throughout the testing phase.

These measures ensure the system adheres to IEEE ethical guidelines and prioritizes both user safety and responsible technology deployment.

References

[1] K. Deekshitha, C. Patange, M. Harshitha, and P. Y.J, "Automated handwriting machine," in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, 2023, pp. 1–4. DOI: 10.1109/I2CT57861.2023.10126330.

[2] R. A. Xiong, "Cost-effective design of scribe ai – robotic handwriting systems using raspberry pi pico and 3d printing," in *2024 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, IEEE, Nov. 2024, pp. 1–6. DOI: 10.1109/lisat63094.2024.10807994. [Online]. Available: http://dx.doi.org/10.1109/LISAT63094.2024.10807994.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative adversarial networks*, 2014. arXiv: 1406.2661 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1406.2661.

[4] W. Pan, A. Zhu, X. Zhou, B. K. Iwana, and S. Li, *Few shot font generation via transferring similarity guided global style and quantization local style*, 2023. arXiv: 2309.00827 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2309.00827.

[5] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, *Neural discrete representation learning*, 2018. arXiv: 1711.00937 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1711.00937.

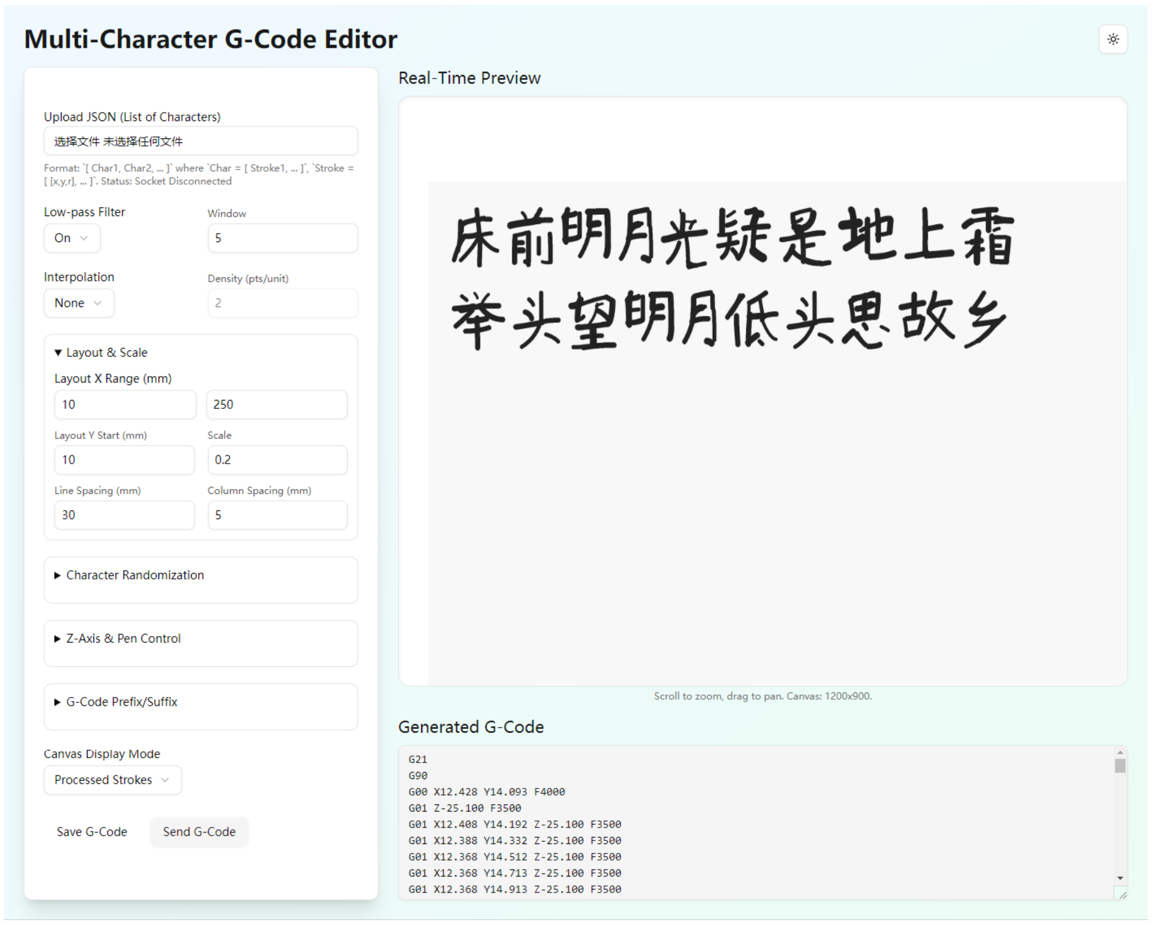
[6] Y. Luo, Z. Wu, and Z. Lian, *Callirewrite: Recovering handwriting behaviors from calligraphy images without supervision*, 2024. arXiv: 2405.15776 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2405.15776.

[7] Wikipedia contributors. "Bézier curve — wikipedia, the free encyclopedia." Accessed: 2025-05-23. (2024), [Online]. Available: https://en.wikipedia.org/wiki/B%C3%A9zier_curve.

Appendix A Supplementary Requirements and Verification Table

Type	Requirement	Verification	Pass
Few-shot Font Generation	The system must accurately generate a font glyph with visual similarity to 3–5 user-provided examples, achieving a cosine similarity score ≥ 0.85 in feature space.	Use a pretrained font encoder (e.g., VGG-style) to extract features of the generated and reference glyphs. Compute cosine similarity over 100 test characters and verify the average similarity is ≥ 0.85.	Yes
Few-shot Font Generation	Generated glyphs must be output in 128×128 grayscale format with pixel intensity range [0, 255] and structural integrity (no more than 2% missing regions).	Perform pixel value range check and apply structural similarity index (SSIM) against original glyphs. Verify that SSIM ≥ 0.95 and missing stroke pixels do not exceed 2%.	Yes
Few-shot Font Generation	The training process of the VQ-VAE module must converge within 50,000 iterations with reconstruction loss below 0.02.	Log loss values every 100 iterations during VQ-VAE training. Plot final reconstruction loss and verify it is below threshold.	Yes
Few-shot Font Generation	The GAN-based generator must complete training within 500,000 steps and produce results within 1 second per glyph at inference time on a GPU.	Time the inference speed across 100 characters. Verify average latency ≤ 1.0 s using a standard RTX 4060 or equivalent GPU.	Yes
Pen Trajectory Extraction	The system must convert a 128×128 grayscale image into a binary skeleton with stroke width of 1 pixel and connectivity preserved.	Visualize skeleton overlay on original image and use a flood-fill-based test to confirm that all stroke regions remain connected post-skeletonization.	Yes

Type	**Requirement**	**Verification**	**Pass**
Pen Trajectory Extraction	The recursive stroke tracing must produce polylines covering $\geq$ 95% of the skeleton pixels.	Count total skeleton pixels and compare to the union of all pixels along output polylines. Confirm coverage is at least 95%.	Yes
Pen Trajectory Extraction	The system must identify endpoints and junctions with $\pm$2 pixel tolerance from ground truth labels.	Compare system output with manually labeled ground-truth junction/endpoints across 100 characters. Confirm spatial offset does not exceed 2 pixels for over 98% of points.	Yes
Pen Trajectory Extraction	Vectorization output must be completed within 300 ms per character on average for real-time robotic control.	Measure processing time across 100 test images. Report mean and standard deviation. Verify average time is $\leq$ 300 ms.	Yes



Multi-Character G-Code Editor

Upload JSON (List of Characters): 选择文件 未选择任何文件

Format: `[ Char1, Char2, ... ]` where `Char = [ Stroke1, ... ]`, `Stroke = [ [x,y,r], ... ]`. Status: Socket Disconnected

Low-pass Filter: On — Window: 5

Interpolation: None — Density (pts/unit): 2

▼ Layout & Scale

Layout X Range (mm): 10 — 250

Layout Y Start (mm): 10 — Scale: 0.2

Line Spacing (mm): 30 — Column Spacing (mm): 5

▶ Character Randomization

▶ Z-Axis & Pen Control

▶ G-Code Prefix/Suffix

Canvas Display Mode: Processed Strokes

Real-Time Preview

床前明月光疑是地上霜
举头望明月低头思故乡

Scroll to zoom, drag to pan. Canvas: 1200x900.

Generated G-Code

```
G21
G90
G00 X12.428 Y14.093 F4000
G01 Z-25.100 F3500
G01 X12.408 Y14.192 Z-25.100 F3500
G01 X12.388 Y14.332 Z-25.100 F3500
G01 X12.368 Y14.512 Z-25.100 F3500
G01 X12.368 Y14.713 Z-25.100 F3500
G01 X12.368 Y14.913 Z-25.100 F3500
```

Figure 12: Multi-Character G-Code Editor

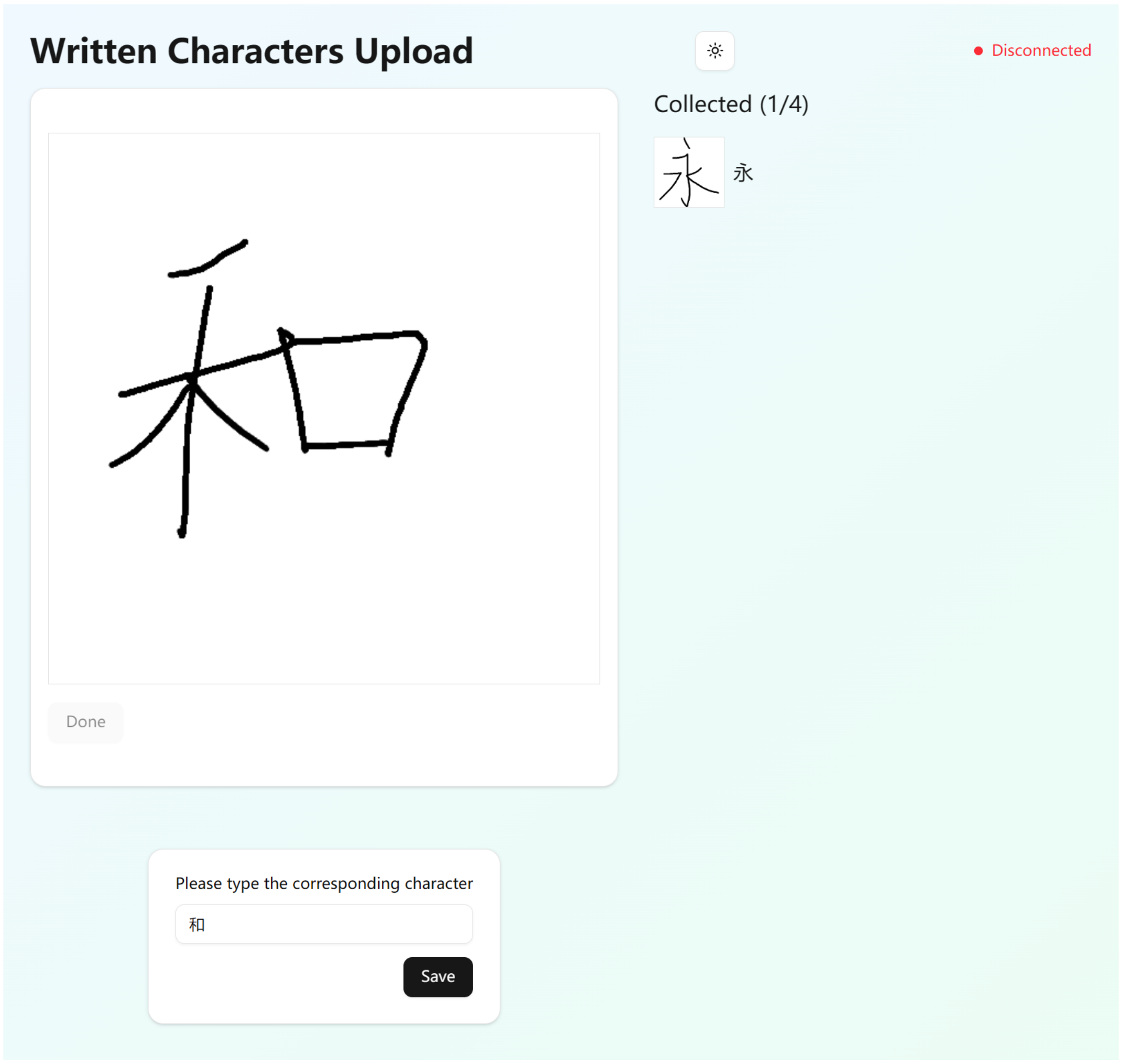

Written Characters Upload
● Disconnected
Collected (1/4)
永
Done
Please type the corresponding character
和
Save

Figure 13: Writing Pad

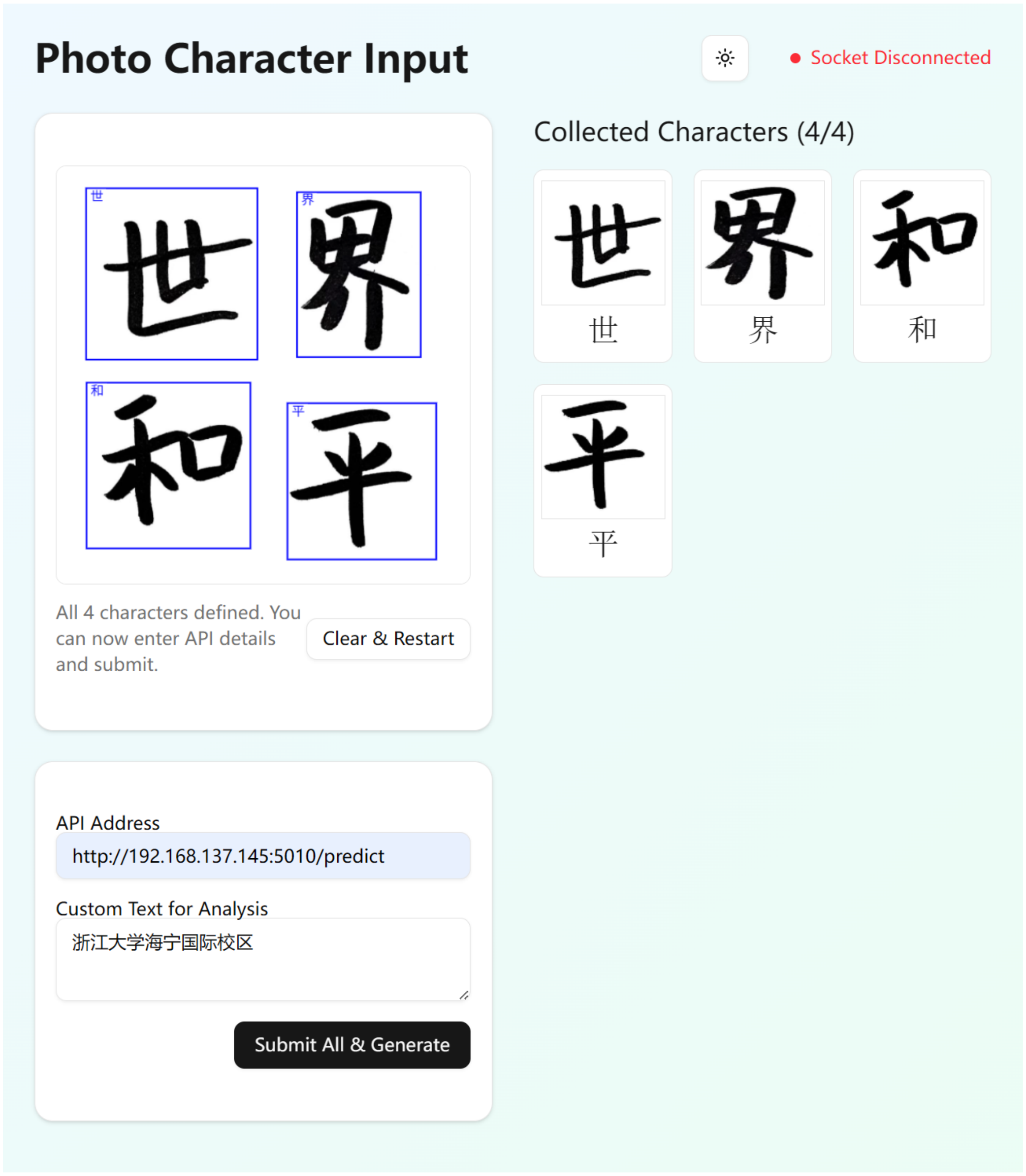


Photo Character Input

Socket Disconnected

Collected Characters (4/4)

世 界 和 平

All 4 characters defined. You can now enter API details and submit.

Clear & Restart

API Address

http://192.168.137.145:5010/predict

Custom Text for Analysis

浙江大学海宁国际校区

Submit All & Generate

Figure 14: Image Upload