ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

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

April 14, 2025

Contents

1 Introduction

1.1 Problem and Solution 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

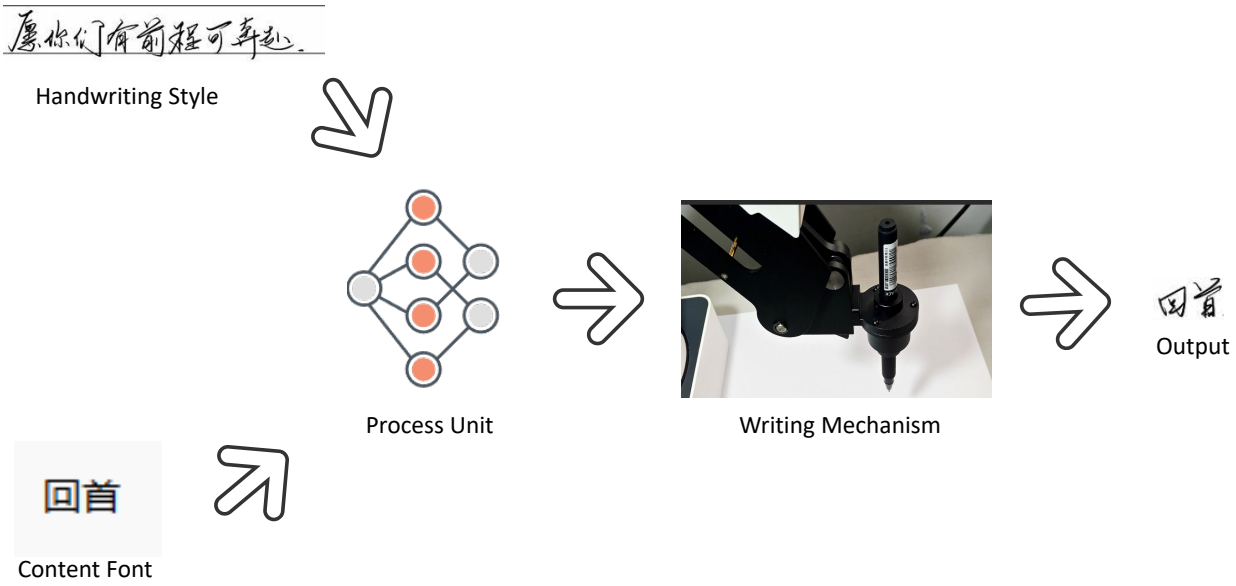


Figure 1: The visual illustration of our project: a user's handwriting sample and content text are processed through a neural network, enabling a robotic arm to reproduce the text in the user's unique handwriting style.

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

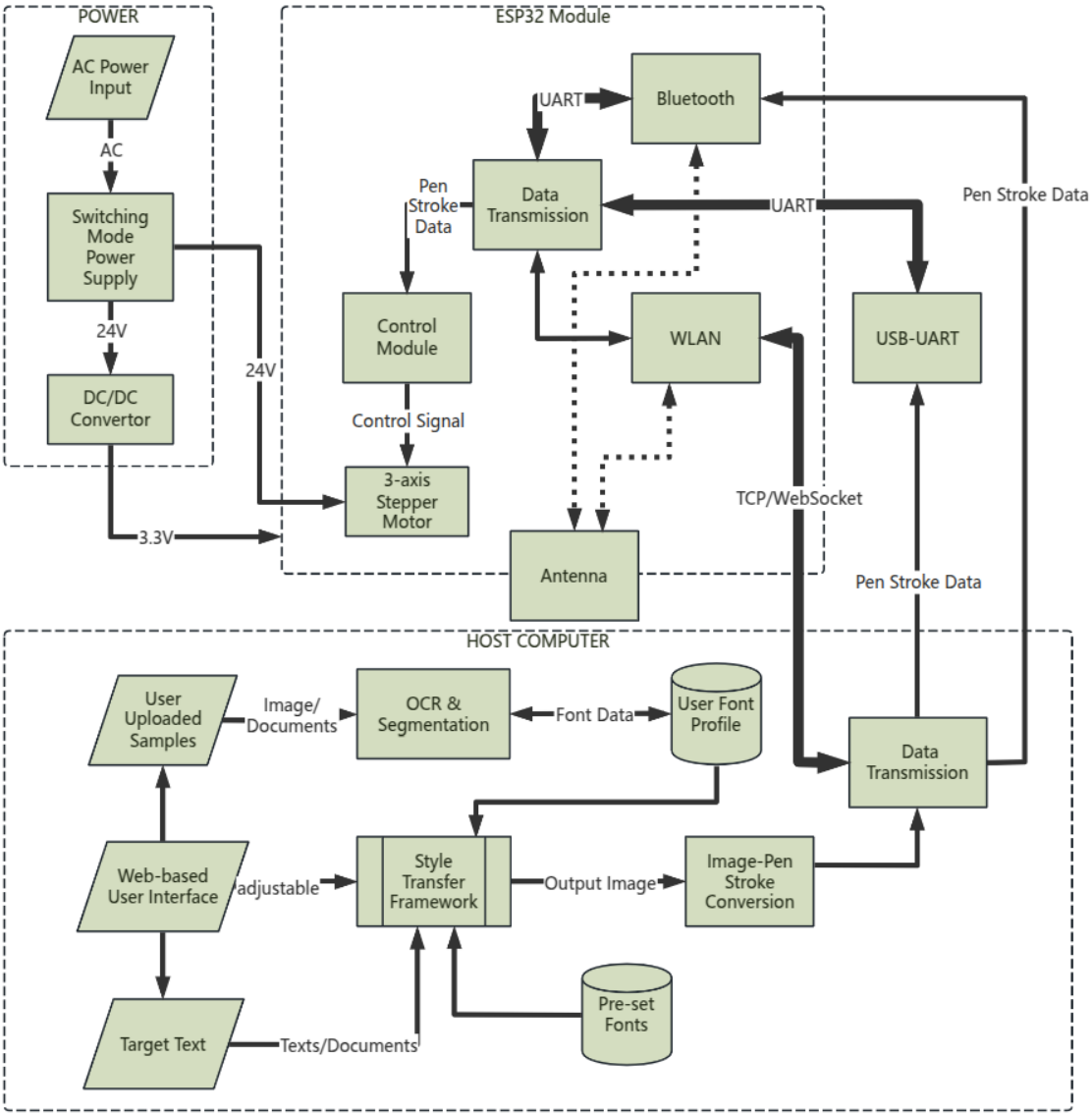
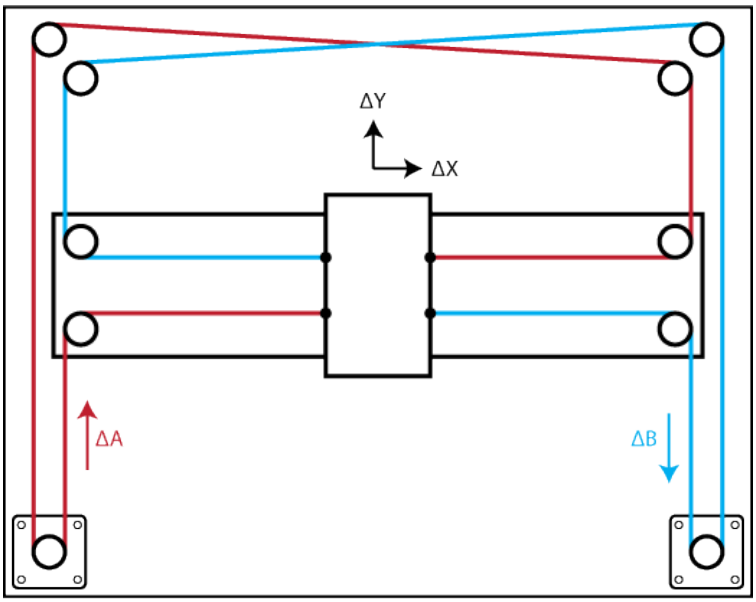


Figure 2: Block Diagram of Our System

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:

$$\Delta X = {}^{1}/_{2}(\Delta A + \Delta B), \quad \Delta Y = {}^{1}/_{2}(\Delta A - \Delta B)$$

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

Figure 3: Schematic of CoreXY Mechanism

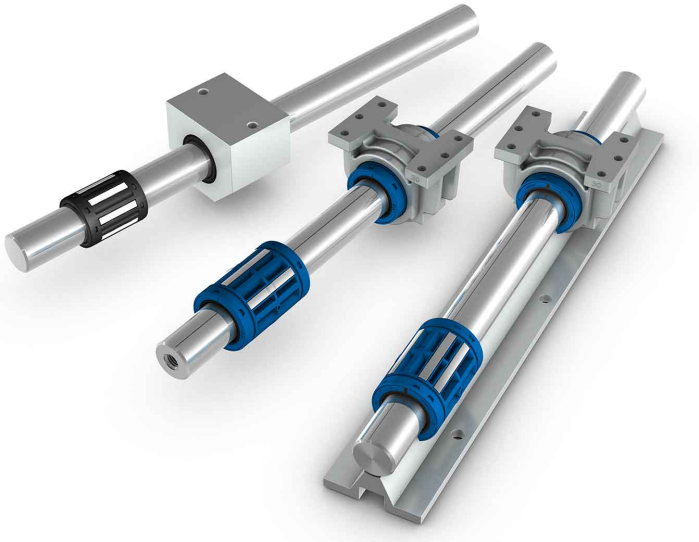


Figure 4: Linear axis and bearing

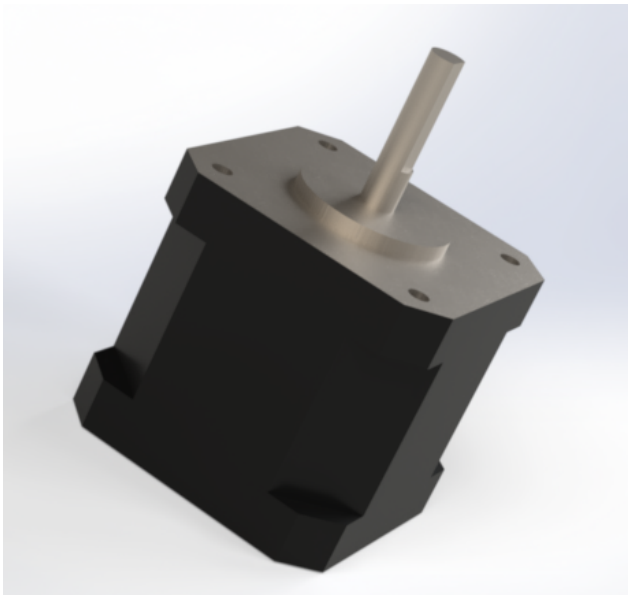


Figure 5: 42mm stepper motor

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.

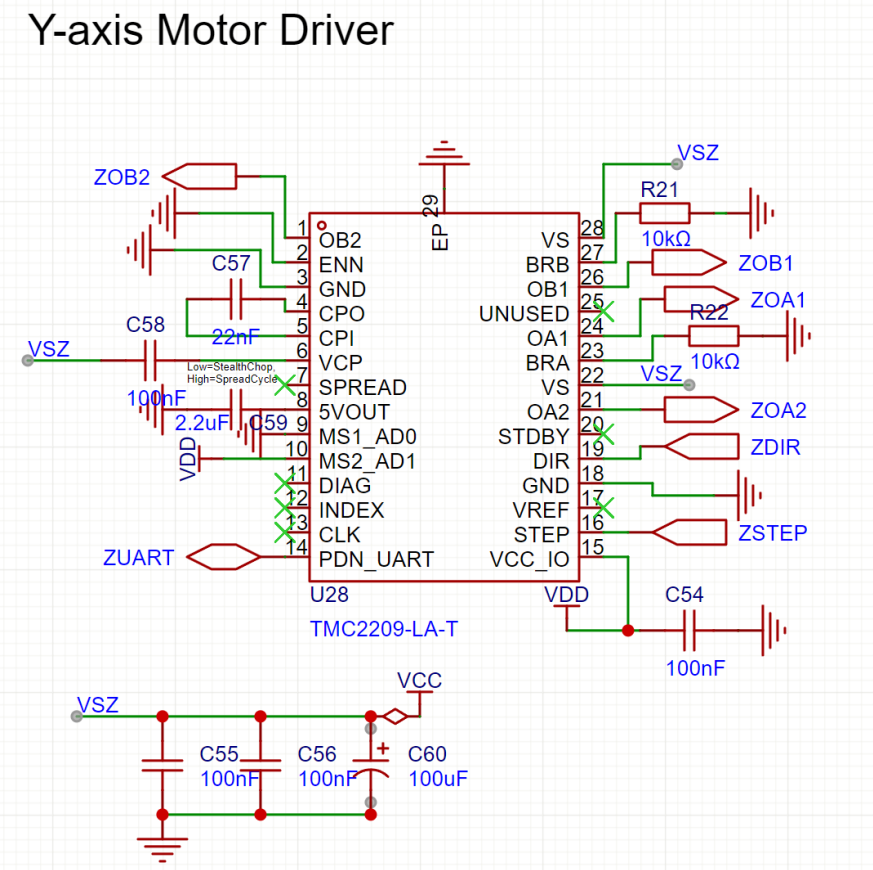


Figure 6: TMC2209 Circuit Schematic

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 3 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. We extracts vectorized stroke paths from binary skeleton images by recursively splitting the image into smaller regions using optimal horizontal or vertical cuts based on pixel sparsity and shape heuristics. Since Chinese characters are primarily composed of straight-line strokes, the endpoints and junctions extracted during vectorization effectively capture

the overall contour of each character, making them suitable for downstream robotic arm execution.

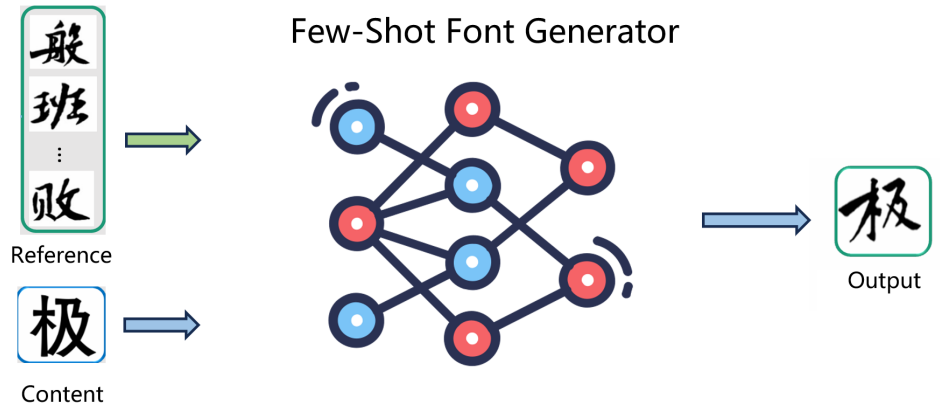


Figure 7: Few-shot Font Generator

For few-shot font generator, we employ a few-shot font generation system inspired by VQ-Font [3], which integrates two essential components:

- **Global Style Aggregator (GSA):** This component uses character similarity to guide and ensure consistency in the overall style across similar characters.
- **Local Style Aggregator (LSA):** Utilizing vector quantization and cross-attention, LSA captures intricate details, allowing for the learning of elements without the need for manual definition.

The content decoder (generator) is trained using Generative Adversarial Networks (GANs) combined with a self-reconstruction framework, enabling its application across various scripts without requiring extensive labeled data. The implementation of the few-shot font generation system is expected to align with the following requirements:

- **Training Few-shot Font Generator:** The training happens in two stages. The first stage is the pre-training of VQ-VAE [4]. This step uses 3,000 Chinese characters from a few fonts, each resized to 128x128 pixels, to learn local style components. It employs a Vector Quantized Variational Autoencoder (VQ-VAE) with a reconstruction loss and a latent loss, using an embedding dimension of 256, a batch size of 256, and 50,000 iteration steps.The second stage is the training of GAN [5]. After pre-training, the full font generation model is trained using a Generative Adversarial Network (GAN). It includes a fixed pre-trained content encoder, a style encoder trained from scratch, and other components like style aggregators and a decoder. The training uses a batch size of 48, 8 attention heads in three stacked transformer layers, and runs for 500,000 iteration steps.
- **Extracting Handwritten Characters:** Begin by having the user write Chinese characters on grid paper and then capturing an image of this paper, which is subsequently converted to grayscale. Following this, identify the grid edges and segment each

character area accordingly, removing any unnecessary blank spaces to isolate the character itself. Finally, resize the extracted character images to 128×128 pixels, ensuring their original aspect ratio is preserved, to create uniform grayscale images.

- **Inference with Desired Content:** Given a target character and 3-5 reference glyphs, the font generator first encodes the content separately, extracts style features from references, and combines them for generation. The Global Style Aggregator (GSA) uses similarity to weight overall style while the Local Style Aggregator (LSA) applies vector quantization and cross-attention to transfer details. This inference method is efficient, processing components in one pass, ideal for font library creation. The similarity-guided GSA enhances flexibility, adapting style to content structure. For optimal inference, we will use 3-5 clear references and avoid rare or intricate inputs.

Table 1: R/V Table: Few-shot Font Generation System

Requirement	**Verification Procedure**
The system must accurately generate a font glyph with visual similarity to 3–5 user-provided examples, achieving a cosine similarity score $\geq$ 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.
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%.
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.
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.

2.3.2 Pen Trajectory Extraction System

Once the 128×128 grayscale character images are generated, we apply a stroke vectorization algorithm [1] to extract the pen trajectory information necessary for downstream robotic writing execution. The algorithm operates on binary skeleton images obtained from the grayscale input via thresholding and skeletonization (e.g., Zhang-Suen [6]). It is designed to recursively decompose the character image into meaningful stroke segments by analyzing image sparsity and structural heuristics.

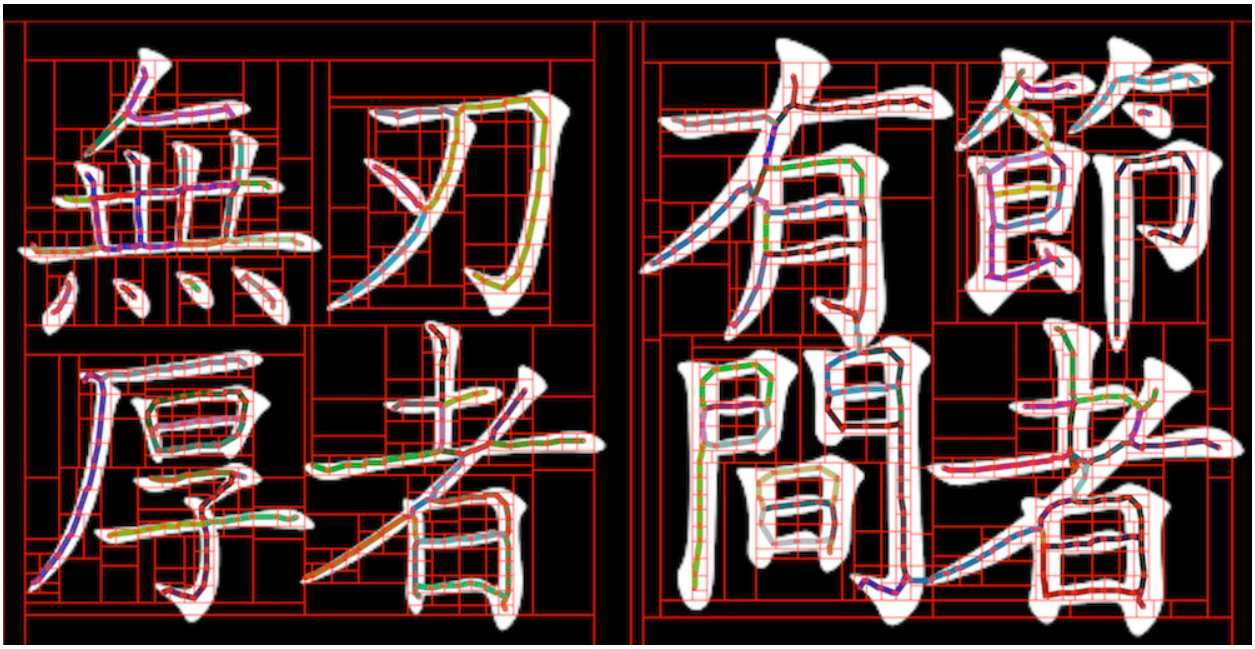


Figure 8: Visualization of vectorized stroke extraction results applied to various shapes and characters.

The algorithm follows a divide-and-conquer strategy. If the image is sufficiently small, it enters a base-case stroke analysis routine. Otherwise, it scans the image to find an optimal vertical or horizontal split line that minimizes foreground pixel density while preserving stroke continuity—i.e., ensuring no stroke crosses through the corners of the split regions. The image is divided along this line into two subimages, which are then recursively processed. After processing, the resulting polyline segments from each subimage are merged. If an endpoint from one side lies near the split line, it is matched with a neighboring endpoint on the other side to form a continuous stroke. Matching is based on proximity constraints that account for small differences in position due to skeleton curvature.

At the smallest scale, the algorithm performs a border-walk around the subimage to detect transition points between background and foreground pixels. These points are treated as entry or exit locations for stroke paths. Each is connected to the center of the

[1] https://github.com/LingDong-/skeleton-tracing

subimage, forming an initial set of line segments. Several heuristics refine this representation: if there are exactly two endpoints, the segment is likely a straight stroke and is directly connected; if there are three or more, the region likely contains a junction, and the stroke center is adjusted to the densest 3×3 subregion; if only one endpoint is detected, it is connected to the center directly.

This recursive procedure produces a compact, vectorized representation of each character, capturing key structural features such as stroke direction, endpoints, and junctions. The output is a set of polylines approximating the pen movement, making it highly suitable for robotic arm execution. Because Chinese characters predominantly consist of linear strokes and junctions, this method effectively transforms static image data into actionable motion trajectories, thereby bridging the gap between font generation and physical handwriting.

Table 2: R/V Table: Pen Trajectory Extraction System

Requirement	**Verification Procedure**
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.
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%.
The system must identify endpoints and junctions with ± 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.
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 ≤ 300 ms.

2.3.3 Robotic Handwriting Subsystem

The Robotic Handwriting Subsystem is responsible for executing precise two-dimensional handwriting motions and controlling the pen's vertical contact with the writing surface. To achieve this, we implement a CoreXY mechanism driven by three stepper motors: two for planar movement (X and Y) and one for the Z-axis (pen actuation).

- **CoreXY Motion Platform:** CoreXY is a type of Cartesian motion system optimized for speed, precision, and mechanical efficiency. Instead of assigning one stepper motor per axis, CoreXY uses two motors operating in tandem over a system of belts and idlers to control motion in both X and Y directions. Motor A and Motor B operate together or in opposition to create diagonal, vertical, or horizontal movement. This results in lower moving mass (only the toolhead moves), faster motion, and reduced vibration. We use two NEMA 17 stepper motors with 1.8° resolution (200 steps/rev), driving GT2 belts through 20T pulleys, providing a theoretical spatial resolution of 0.1 mm or better.
- **Pen Actuation Mechanism (Z-axis):** A third stepper motor with a lead screw actuator or cam linkage is used to raise and lower the pen. This allows the pen to lift when transitioning between strokes or repositioning. The vertical axis has a limited travel range (e.g., 5–10 mm), and the motor will be configured with software endstops to prevent over-travel. A microswitch or optical endstop will be mounted for homing the Z-position.
- **Mechanical Frame and Bearings:** The structural frame is assembled from aluminum extrusions (e.g., 2020 profile) and supported by V-slot wheels or linear rails to reduce backlash and improve repeatability. The CoreXY idler pulleys are mounted on machined plates to maintain belt tension and geometric symmetry.
- **Trajectory Execution:** The stroke trajectories are provided in G-code format generated by the Image-to-Stroke Converter subsystem. This G-code is parsed and executed by the FluidNC firmware running on the ESP32 microcontroller, which generates the step and direction signals in real time.
- **Precision Targeting:** Based on 1.8°/step motors and 20-tooth GT2 pulleys (2 mm pitch), each step moves the belt by 0.1 mm. Microstepping via TMC2209 drivers (configured for 1/16 or 1/32 microstepping) allows effective resolutions of 0.00625–0.003125 mm per microstep, greatly enhancing smoothness and detail fidelity.

This subsystem fulfills the high-level requirement of producing handwritten output with ±0.1 mm accuracy, while maintaining smooth, lifelike motion and sharp stroke transitions.

2.3.4 Communication and Control Subsystem

The Communication and Control Subsystem bridges the data pipeline between high-level stroke generation and the low-level motor signals required for mechanical execution. It is built around the ESP32 microcontroller and the open-source FluidNC firmware, which enables parsing and real-time execution of G-code commands.

- **ESP32 Microcontroller:** The ESP32-WROOM module serves as the central processing unit for the robot. It features dual-core Xtensa processors and hardware peripherals (UART, SPI, PWM, Wi-Fi, Bluetooth), making it ideal for embedded motion control. The ESP32 handles:

  - Receiving stroke data via UART, Wi-Fi, or Bluetooth
  - Interpreting G-code commands (positioning, pen up/down)
  - Generating step/direction signals for all three motors
  - Monitoring limit switches and emergency stop signals
- **FluidNC Firmware:** FluidNC is a lightweight G-code interpreter and motion control firmware designed for CNC and 3D printing applications on ESP32. It offers:
  - G-code streaming from SD card or over serial/network
  - Configuration of CoreXY kinematics with high-speed planning
  - Support for spindle/laser/pen tool types with Z-axis lift
  - Runtime adjustable feedrates, acceleration, and homing

  Configuration is handled through a YAML file uploaded to the ESP32 via the FluidNC web interface. Axes are mapped to specific GPIO pins, with real-time step rates reaching over 50 kHz per axis.
- **Motor Driver Interface:** Step/direction signals are sent from ESP32 GPIOs to TMC2209 stepper drivers via direct wiring or UART-controlled interface. The TMC2209 drivers provide microstepping, current control, and stall detection, ensuring smooth motion and reduced noise.
- **Stroke Command Protocol:** The stroke data is converted into G-code format by the Image-to-Stroke Converter. These G-code commands follow standard syntax (e.g., 'G1 X10 Y10 Z0 F1200') and specify:
  - XY coordinates in mm
  - Pen actuation states via Z-axis ('Z0' for down, 'Z5' for up)
  - Feedrate in mm/min

  Commands are streamed to the ESP32 in real-time or preloaded onto an SD card for standalone operation.

This subsystem ensures high synchronization fidelity between command reception and motor actuation, fulfilling latency and accuracy requirements. It also abstracts away low-level control logic, allowing modular upgrades or replacements of stroke-generation methods.

2.4 Tolerance Analysis

2.4.1 Few-shot Font Generation System

- **Reference Sample Variations:**

The performance metrics of the model improve with an increase in reference samples from 1 to 8, plateauing between 5 to 8 samples. The model exhibits high tolerance with 3 to 8 references but performs unsatisfactorily with fewer than 3 due to insufficient style cues. Hence, at least 3 references are needed for reliable style transfer, which limits the technique's application in scenarios with very limited data.

- **Quality and Noise in Input Data:**

  The model is designed to work with high-quality reference images and has not been tested on noisy inputs. The style encoder's ability to handle noise may be limited due to its reliance on precise feature extraction from clean images. It is crucial for us to accurately extract and segment user-provided inputs to yield outputs that contain the user's stylistic font for the model's use.

- **Training Parameters and Resource Intensity:**

  The model's training process involves substantial resources, including large batch sizes and many iterations. The model likely has low tolerance for reduced training settings, as these could disrupt convergence and affect the stability of the GAN training. This indicates a high demand for computational resources during training.

2.4.2 Robotic Handwriting & Communication Subsystem

- **Mechanical Tolerance:** Precision in handwriting replication depends critically on mechanical tolerances. Rails and belts/screws must ensure minimal backlash and slippage. Any mechanical tolerance exceeding ±0.1 mm may visibly distort handwriting. Careful design and routine calibration are required.
- **Timing and Signal Accuracy:** Stepper motors rely on precise timing of control signals. Variations greater than 1 ms in signal generation or processing can degrade handwriting quality. This subsystem requires stringent synchronization between the ESP32 signals and the motor response, necessitating real-time firmware optimization.
- **Communication Stability:** The subsystem's performance is sensitive to communication reliability. If communication latency or packet loss exceeds thresholds (0.1%), pen movements may become irregular, resulting in discontinuities or unintended strokes. Robust error-checking protocols and efficient data encoding should mitigate these risks.

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

Table 3: Parts Cost Breakdown

Part	**Manufacturer**	**Part Number**	**Qty**	**Cost (CNY)**
42mm Stepper Motors	Generic	42BYGH47	3	72.00
GT2 Belts + Pulleys	Generic	GT2-6mm	1 set	20.00
Linear Axis	Generic	8mm	2m	10
Linear Bearings	Generic	LM8UU	4	5
3D Printed Parts	In-house	—	—	40
Power Supply	MeanWell	LRS-100-24V4.16A	1	30
Custom PCB & electronic components	JLCPCB	—	1	250
Misc Hardware (screws, wires)	—	—	—	10.00
Total				**437$\approx$ $60.0**

Parts and Components:

Grand Total:

- Labor: $16,000
- Parts: $60
- **Total Estimated Cost: $16,060**

3.2 Schedule

Table 4: Project Schedule by Week

Week	Milestone
14 Apr.	3D modeling of the robot
14 Apr.	Train few-shot font generation model, evaluate output quality
14 Apr.	Implement character segmentation, font preprocessing pipeline
18 Apr.	Implement stroke extraction and G-code generator
18 Apr.	CoreXY mechanical assembly and ESP32 driver board fabrication
18 Apr.	PCB soldering assembly
21 Apr.	Integrate hardware with FluidNC firmware and establish communication
21 Apr.	Testing, debugging, and validation

Task Sharing: Team members will collaborate on all subsystems, with focus areas divided as follows:

- Zhixiang: Font generation, preprocessing pipeline, software integration
- Zifan: Mechanical assembly, ESP32 firmware, control hardware, PCB design & soldering
- Xuancheng: PCB validation, testing, User Interface Design
- Mingchen: Data point generation, User Interface Design

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] 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.

[4] 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.

[5] 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.

[6] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984, ISSN: 0001-0782. DOI: 10.1145/357994.358023. [Online]. Available: https://doi.org/10.1145/357994.358023.