A Data-driven Robotic Chinese Calligraphy System using Convolutional Auto-Encoder and Differential Evolution

Xingen Gao[a], Changle Zhou[a], Fei Chao[a,b,*], Longzhi Yang[c], Chih-Min Lin[d], Tao Xu[b], Changjing Shang[b], Qiang Shen[b]

[a] *Cognitive Science Department, School of Information Science and Engineering, Xiamen University, China*
[b] *Institute of Mathematics, Physics and Computer Science, Aberystwyth University, UK*
[c] *Department of Computer and Information Sciences, Northumbria University, UK*
[d] *Department of Electrical Engineering, Yuan Ze University, Taiwan*

Abstract

The Chinese stroke evaluation and generation systems required in an autonomous calligraphy robot play a crucial role in producing high-quality writing results with good diversity. These systems often suffer from inefficiency and non-optima despite of intensive research effort investment by the robotic community. This paper proposes a new learning system to allow a robot to automatically learn to write Chinese calligraphy effectively. In the proposed system, the writing quality evaluation subsystem assesses written strokes using a convolutional auto-encoder network (CAE), which enables the generation of aesthetic strokes with various writing styles. The trained CAE network effectively excludes poorly written strokes through stroke reconstruction, but guarantees the inheritance of information from well-written ones. With the support of the evaluation subsystem, the writing trajectory model generation subsystem is realized by multivariate normal distributions optimized by differential evolution (DE), a type of heuristic optimization search algorithm. The proposed approach was validated and evaluated using a dataset of nine stroke categories; high-quality written

*Corresponding author
Email addresses: 31520160154529@stu.xmu.edu.cn (Xingen Gao), dozero@xmu.edu.cn (Changle Zhou), fchao@xmu.edu.cn (Fei Chao), longzhi.yang@northumbria.ac.uk (Longzhi Yang), cml@saturn.yzu.edu.tw (Chih-Min Lin), tax2@aber.ac.uk (Tao Xu), cns@aber.ac.uk (Changjing Shang), qqs@aber.ac.uk (Qiang Shen)

strokes have been resulted with good diversity which shows the robustness and efficacy of the proposed approach and its potential in autonomous action-state space exploration for other real-world applications.

Keywords: Robotic Chinese calligraphy, data-driven evaluation model, convolutional auto-encoder, differential evolution

2010 MSC: 00-01, 99-00

1. Introduction

The main focus of research for robotic writing is the trajectory generation methods for robots to write characters or letters using a pen or ink brush [1, 2, 3, 4, 5], but this is not necessarily the case for Chinese calligraphy robots. Chinese calligraphy, an aesthetic presentation of Chinese characters, implicitly expresses the emotion of the artist, in addition to convey the message that the characters imply. Consequently, the research on robotic Chinese calligraphy is interdisciplinary across robotics and arts [6, 7], which makes the designing of Chinese calligraphy robotic systems challenging. Nevertheless, this special challenge represents a new dimension of development to further advance the robotic field by better replicating human intelligence in machines. Accordingly, the techniques required in writing high-quality aesthetic strokes can be readily transferred to other industries [8, 9], such as industrial welding and medical rehabilitation, which witnesses the importance of the research in robotic Chinese calligraphy from a technical point of view. Culture-wise, the presentation of this oriental tradition through modern digital technologies inspires creativity and helps interest development in Chinese calligraphy. The challenge aforementioned can be expressed as two technical difficulties.

The development of an effective and efficient writing result evaluation mechanism forms the first technical difficulty. Almost all conventional robotic writing systems use pre-defined and simplistic evaluation criteria that often lead to a lack of diversity in the writing results. For instance, the curve fitting approach proposed by Yao et al. [10, 11] matches the writing trajectories to the

pre-defined image patterns of strokes; the stroke generation system reported in Kwok et al. [12] uses prepared and inflexible stroke boundaries; the visual feedback based robotic drawing system presented by Mueller et al. [13] only considers the overlay of a writing result and its reference stroke. All these methods require manual implementations to obtain fixed image patterns or stroke boundaries, which often lead to monotonous writing results. Several other projects [14, 15, 16] manually analyze the geometric features of calligraphy to design the evaluation methods for robotic Chinese calligraphy. One notable exception is the data-driven approach reported Chao et al. [17], which introduce the Generative Adversarial Nets (GAN) to a robotic manipulator to generate writing trajectories. This work takes advantage of a discriminative model in GAN, which serves as the feedback system for learning robotic writing. However, the GAN-based models only discriminate between the instances of true stroke probability distribution and that of the writing results, which limits the assignment of an evaluation score to each writing sample and thus the quality of writing outputs from expectations.

The learning-based robotic writing systems usually suffer from local minima when finding optimal trajectory models using gradient-based methods, which poses the second difficulty. For instance, a gradient descent algorithm to decrease writing errors was proposed by Mueller et al. [13], which itself can be time-wise inefficient when learning complex strokes and characters. Specifically, once a practical robot entity is involved in a gradient descent algorithm-based learning system, it is difficult to back-propagate errors. To address this, a policy gradient method is used in the work of Chao et al. [17] based on several specific measures, to train the trajectory model. However, the overall performance is still not as good as expected due to the high variance in estimating the policy gradients. These difficulties reveal the necessity to globally optimize the writing system during the training stage.

This paper proposes a data-driven robotic Chinese calligraphy system, which enables a robot to autonomously learn to generate writing trajectory models, to address these technical difficulties. In particular, a convolutional auto-encoder

(CAE) [18] is applied as the data-driven writing result evaluation subsystem. The CAE firstly extracts stroke features which encode stroke images as low dimensional codes. Then, each written image is compared with the whole dataset, and the well-written strokes are used as candidates for reconstruction with no or very marginal loss of information. The comparison between the input image and the reconstruction provides an effective means for writing result evaluation, which is used to support the automatic development of optimal writing trajectory models. Enlightened by the recent work reported in [19, 20, 21, 22, 23], which use evolutionary algorithms (EA) for agent development, differential evolution (DE) [24] is adapted in this work to obtain optimal writing trajectory models with the support of the CAE-based evaluation subsystem. In contrast to the traditionally used gradient-based algorithms, DE is highly parallelizable and gradient-free, and thus more likely to escape from local optimums.

The proposed learning framework allows the robot to autonomously develop high-performance writing skills and avoid monotonous writing results. The experimentation in this work was conducted based on a dataset with 9 stroke categories. The experimental results demonstrate the superiority of the proposed work in helping robots to generate high-quality Chinese calligraphy strokes with good diversity over others. The main contributions of this work are twofold: 1) a CAE is applied to evaluate the quality of strokes to support automatic writing trajectory model generation; 2) DE was adapted to the robotic manipulator for generating writing trajectory models with the support of the CAE-based evaluation subsystem.

The remainder of this paper is organized as follows: Section 2 serves as a brief introduction to the theoretical underpinning CAE and DE. Section 3 specifies the proposed system, which allows a calligraphy robot to automatically learn to write strokes with high quality and good diversity. Section 4 presents the experimental set up and discusses the experimental results. Section 5 concludes the paper and points out important future work.

2. Background

This work applies a CAE [18] for the evaluation subsystem, due to its strong feature extraction capability; whilst DE [25, 26, 27] is applied to find optimal writing trajectory models because of the readiness to use, reliable, and time efficiency. These fundamental techniques are reviewed in this section.

2.1. Convolutional Auto-Encoder

A conventional auto-encoder is a fully connected neural network that consists of two parts: an encoder and a decoder. The encoder maps high dimensional data into a latent code (lower dimensional), while the decoder reconstructs the initial data from the latent code. Auto-encoders are commonly used to reduce dimensionality and extract features. A CAE is a variant of an auto-encoder [18] whose encoder and decoder are convolutional and deconvolutional neural networks, respectively. The neural networks allow the CAEs to capture the spatial information of the data; therefore, CAEs are more suitable for applications related to image processing [28, 29, 30, 31]. The main components of a CAE, including an encoder and a decoder, are briefed as follows:

Encoder. The encoder consists of convolution and pooling layers. A convolution layer convolves the input image with parametric filters (convolution kernels) to produce feature maps of an image. The k-th feature map of a mono-channel input, $\boldsymbol{x}$, is given by:

$$\boldsymbol{h}^k = \sigma(\boldsymbol{x} \otimes \boldsymbol{W}^k + \boldsymbol{b}^k), \tag{1}$$

where $\sigma(\cdot)$ is an activation function (normally non-linear); $\otimes$ denotes the 2D convolution operation; $\boldsymbol{W}^k$ and $\boldsymbol{b}^k$ denote the weight and bias of the k-th filter. A pooling layer partitions the input feature map into a set of rectangular regions and outputs an aggregation of each rectangular region; for instance, maximum pooling aggregates the values in the region to its maximum value, while average pooling to the average of all values in the region. By representing

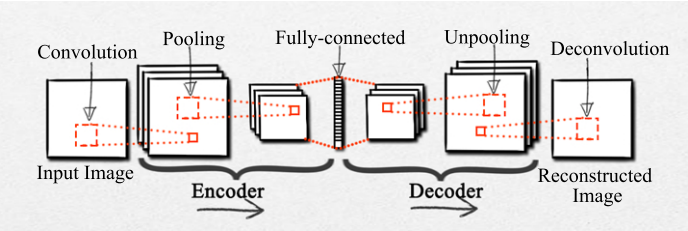

Figure 1: A CAE typically consists of convolution, pooling, fully-connected, unpooling, and deconvolution layers.

the whole region using only a representative value, the pooling layer reduces the computational requirement in the upper layers.

Decoder. A decoder consists of deconvolution and unpooling layers. A deconvolution layer performs an inverse operation of the convolution layer; in other words, a deconvolution layer reconstructs images from feature maps. Reconstruction is realized using:

$$\boldsymbol{y} = \sigma(\sum_{k=1}^{\mathrm{H}} \boldsymbol{h}^k \otimes \widetilde{\boldsymbol{W}}^k + \boldsymbol{c}), \tag{2}$$

where $\boldsymbol{h}^k$ denotes the k-th of all H feature maps; $\widetilde{\boldsymbol{W}}^k$ corresponds to the flip operation over both dimensions of the weights; $\boldsymbol{c}$ denotes the bias. Also, an unpooling layer, performing the reverse operation of pooling, reconstructs the original size of each rectangular region. Interesting, as shown in Fig. 1, an encoder can be connected with a decoder via fully-connected layers.

Objective function.. There are a good set of object functions that can be sued here. The most commonly used objective function is the mean squared error (MSE) between the input data, $\boldsymbol{x}$, and the reconstructed data, $\boldsymbol{y}$:

$$error = \frac{1}{2N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{y}_i)^2. \tag{3}$$

All the parameters of a CAE can be easily optimized through a back-propagation algorithm.

2.2. Differential Evolution

Differential evolution, a gradient-free method, has proven effective in global optimization [32, 33, 34, 35]. Like other EA, DE is a population-based algorithm that explores search space by repeating the cycle of selection and reproduction for a population of parameter vectors. Different from other EAs, a differential mutation operation is employed to generate scaled mutated individuals. Such an operation ensures that mutants do not duplicate existing individuals [25]. The population structure and the main procedures of the DE algorithm are summarized as follows:

Population Structure. The population consists of Np D-dimensional real-valued vectors, called target vectors. Denote the i-th target vector of the population in the g-th generation as follows:

$$\boldsymbol{\omega}_i(g); i = 1, 2, ..., Np; g = 1, 2, ..., g_{max}, \tag{4}$$

where g_{max} is a preset maximum number of generations. Every element of each D-dimensional vector is bounded. Both upper and lower boundaries must be specified:

$$\omega_j^{\mathrm{L}} < \omega_{j,i}(g) < \omega_j^{\mathrm{U}}; j = 1, 2, ..., D. \tag{5}$$

Initialization. Once the initialization bounds have been specified, a random number generator assigns a value within the prescribed range to every parameter of each vector. Such an operation is described as follows:

$$\omega_{j,i}(0) = \omega_j^{\mathrm{L}} + \mathrm{rand}(0, 1)(\omega_j^{\mathrm{U}} - \omega_j^{\mathrm{L}}), \tag{6}$$

where $\mathrm{rand}(0, 1)$ is the evaluation of a uniform random distribution on the interval [0, 1).

Mutation. After initialization, to produce an intermediary population of Np vectors, every three different, randomly chosen target vectors are combined

into one mutant vector by using the differential mutation operation:

$$\boldsymbol{v}_i(g) = \boldsymbol{\omega}_{r_1}(g) + F \cdot (\boldsymbol{\omega}_{r_2}(g) - \boldsymbol{\omega}_{r_3}(g)); i \neq r_1 \neq r_2 \neq r_3, \tag{7}$$

where $\boldsymbol{v}_i(g)$ denotes the i-th mutant vector of the intermediary population in the g-th generation. The scale factor, F, is a positive real-value number, typically less than 1, that controls the rate of the evolution.

Crossover. To complement reproduction, DE crosses each target vector with a mutant vector to form an offspring vector:

$$u_{j,i}(g) = \begin{cases} v_{j,i}(g), & \text{if } \mathrm{rand}_j(0,1) \leq Cr \text{ or } j = j_{\mathrm{rand}} \\ u_{j,i}(g), & \text{otherwise} \end{cases}; j = 1, 2, ..., D. \tag{8}$$

Here j denotes the index of the j-th element of a vector; $\mathrm{rand}_j(0,1)$ is the evaluation of a uniform random distribution for the j-th dimension; Cr is the crossover rate; $j_{\mathrm{rand}} \in \{1, 2, ..., Np\}$ indicates a randomly chosen index that ensures at least one element from the mutant vector is adopted.

Selection. In a comparison of each offspring vector with the target vector, the better vectors have better chances to survive to the next generation. The selection operation is described as:

$$\boldsymbol{\omega}_i(g+1) = \begin{cases} \boldsymbol{u}_i(g), & \text{if } \mathrm{f}(\boldsymbol{u}_i(g)) \geq \mathrm{f}(\boldsymbol{x}_i(g)) \\ \boldsymbol{\omega}_i(g), & \text{otherwise,} \end{cases} \tag{9}$$

where $\mathrm{f}(\cdot)$ is the objective function to be maximized, called fitness function. After updating the new population, the cycle of mutation, crossover, and selection is repeated until a termination condition is met, e.g., the optimum is located or the number of generations reaches a predefined maximum.

3. Proposed System

3.1. System Overview

The proposed system allows a writing robot to automatically learn to write diverse, high-quality Chinese character strokes. The proposed system is illus-

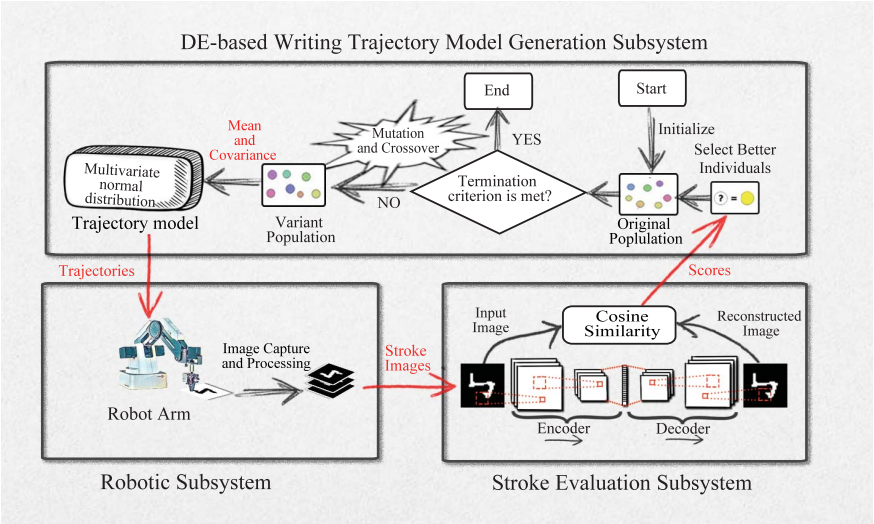

Figure 2: The overview of the proposed framework for robot handwriting. DE is employed to find optimal trajectory models. A CAE-based stroke evaluation subsystem is used to score the writing results and thus providing a fitness function to the DE-based Learning subsystem. The cycle of selection and reproduction is repeated until the optimum is located or the number of generations reaches a predefined maximum.

trated in Fig. 2, which consists of a CAE-based stroke evaluation subsystem and a DE-based writing trajectory model generation subsystem. The CAE-based stroke evaluation subsystem is used to score the writing results and thus providing a fitness function to the DE-based Learning subsystem. The writing trajectory model generation subsystem is responsible to develop trajectory writing models for strokes, each trajectory model is optimized by DE algorithm to writing high-quality results of a stroke.

The overall working procedure of the proposed system is: First of all, the DE randomly generates an original population of individuals, each of which is composed of the parameters (i.e., the means and the covariance matrices) of the multivariate normal distributions. After the mutation and crossover operations, the variant population is converted to the multivariate normal distributions. After a sampling of a distribution, a set of writing trajectory points of a stroke can be obtained from its corresponding multivariate normal distribution. Every

three variables of a trajectory point jointly define the position of the end-effector at a certain point in time. The robotic arm receives the trajectory points and produces the stroke writing movements. The writing results are captured by a vision camera and processed by the stroke evaluation system in providing evaluation scores representing the quality of the images. The stroke evaluation system applies a conventional auto-encoder network to reconstruct the input image; then, the cosine similarity method is used to measure the similarity between the input image and reconstructed ones. The cosine similarity scores are then used by the DE as the fitness function, which selects better individuals to produce a new generation of mean and covariance population. This working procedure will repeat this loop until the termination criterion is reached.

An unsupervised learning tool, CAE, is applied to build the stroke evaluation subsystem, which learns the stroke features extracted from a stroke image dataset. Note that, the presented evaluation subsystem does not simply compare the writing results with the only standard stroke image, but with the entire dataset. Stroke images written by the robot are sent to a trained CAE that produces reconstructed images. During this process, well-written stroke images are reconstructed with little information loss; that is, the CAE, in this case, serves as a "filter" of poorly produced stroke images. The similarity between an original stroke image and its reconstructed one is positively related to the quality of the stroke.

Compared with other EAs, DE has the features of fast convergence speed and processing stability. Although the particle swarm optimization (PSO) features the fast convergence speed, PSO is easily affected by the size of parameters and original population. In contrast to Covariance matrix adaptation evolutionary strategies (CMA-ES) [36], DE is easy to escape from local optimums. Moreover, DE is simpler to be implemented in the proposed approach. Hence, the writing trajectory model generation subsystem applies the DE algorithm to learn each stroke trajectory information.

3.2. Stroke Evaluation Subsystem

The stroke images are evaluated using a data-driven approach, and thus the CAE needs to be trained using a training dataset. The trained CAE firstly extracts stroke features and reconstruct the image using the feature values; then the cosine similarity between a stroke image and its reconstructed version is used to measure the quality of the stroke.

3.2.1. Stroke Reconstruction by CAE

The first task in building a stroke evaluation subsystem is the concise and accurate representation of stroke images, i.e., stroke features extraction, to eliminate noise, background and other irrelevant information. This is implemented in this work using the CAE with a deep architecture organized in nine hidden layers, as detailed in Table 1. The CAE consists of three convolution layers, three fully-connected layers, and three deconvolution layers. Since the training dataset is of relatively low dimension, there is not any pooling layer structured in the architecture. The first convolution layer has 32 kernels; each kernel produces a feature map with a resolution of 14×14 pixels. The second and third convolution layers contain 64 and 128 kernels, each leading to a feature map with a resolution of 7×7 and 4×4, respectively. The three convolution layers are followed by three fully-connected layers. In particular, the second fully-connected layer projects the input into a latent space, which is represented by a latent code system. The hidden layers are ended with three deconvolution layers that perform the inverse operations of the above three convolution layers.

The MSE, as shown in Eq. 3, is a widely used loss function for training the CAE, and it is also applied in this work. To optimize the objective function, the Adam algorithm as specified in [37] is adopted, which is an optimization algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. In this work, all the weights and bias listed in Table 1 in the CAE network are optimized by the Adam algorithm.

Table 1: The architecture and parameters of the CAE for stroke evaluation model

Layer	Parameters	Dimensions
Input	-	1×28×28
Conv. 1	32 4×4 kernels with $stride = 2$ and $pad = 1$	32×14×14
Conv. 2	64 4×4 kernels with $stride = 2$ and $pad = 1$	64×7×7
Conv. 3	128 4×4 kernels with $stride = 2$ and $pad = 1$	128×4×4
Fully-connected 1	1024 neurons	1024
Fully-connected 2	512 neurons	512
Fully-connected 3	1024 neurons	1024
Deconv. 1	128 4×4 kernels with $stride = 2$ and $pad = 1$	128×4×4
Deconv. 2	64 4×4 kernels with $stride = 2$ and $pad = 1$	64×7×7
Deconv. 3	32 4×4 kernels with $stride = 2$ and $pad = 1$	32×14×14
Output	-	1×28×28

3.2.2. Stroke Evaluation by Cosine Similarity

The trained CAE, regarded as a stroke "filter", compresses the written stroke images into latent codes and then reconstructs the images by decoding the latent codes. This can be used to effectively eliminate poorly generated strokes for further use; the quality of a produced stroke is measured by computing the similarity between the stroke image and its reconstruction. Because all images used in this work are gray scale, cosine similarity [38] can be applied to score the stroke:

$$\text{Score}([x_1, x_2, ..., x_N]) = \frac{[x_1, x_2, ..., x_N] \cdot [\hat{x}_1, \hat{x}_2, ..., \hat{x}_N]}{\sqrt{\sum_i x_i} \times \sqrt{\sum_i \hat{x}_i} + \alpha}, \tag{10}$$

where $[x_1, x_2, ..., x_N]$ denotes original image vector; $[\hat{x}_1, \hat{x}_2, ..., \hat{x}_N]$ denotes reconstructed image vector; and α is a positive, near zero constant for avoiding zero denominator.

3.3. Writing Trajectory Model Generation Subsystem

Multivariate normal distributions are used in this work to represent the writing trajectory models of strokes. DE is employed to find the optimal pa-

rameters of the models, which is supported by the CAE-based stroke evaluation subsystem as presented in the last section in providing the fitness function.

3.3.1. Writing Trajectory Model Specification

The writing movement of a stroke is a sequence of brush pen positions; therefore, this work mainly focuses on the sequence of the end-effector positions of a calligraphy robot. In the proposed system, it is considered as a $3 \times T$ dimensional random vector given by

$$\boldsymbol{m} = (p_{\mathrm{x},1}, p_{\mathrm{y},1}, p_{\mathrm{z},1}, p_{\mathrm{x},2}, p_{\mathrm{y},2}, p_{\mathrm{z},2}, , ..., p_{\mathrm{x},T}, p_{\mathrm{y},T}, p_{\mathrm{z},T}), \tag{11}$$

where T is the number of trajectory points; $p_{\mathrm{x},t}, p_{\mathrm{y},t}, p_{\mathrm{z},t}$ $(t = 1, 2, ..., T)$ jointly represents the position of the end-effector in the working space at the moment t; $p_{\mathrm{x},t}, p_{\mathrm{y},t}$ determine the 2D trajectory of a stroke; $p_{\mathrm{z},t} \in (0, 5); t = 1, 2, ..., T$ determines the pressure sequence of the writing brush.

In this paper, the writing trajectory random vector is considered to be normally distributed. Therefore, multivariate normal distributions are used to describe the rules of the writing movement of the strokes. Indeed, multivariate normal distributions are important in statistics and are often used to represent real-valued random vectors whose distributions are unknown. The multivariate probability density of the distribution is defined as:

$$\mathrm{p}(\boldsymbol{m}) = \frac{\exp[-\frac{1}{2}(\boldsymbol{m} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{m} - \boldsymbol{\mu})]}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}}, \tag{12}$$

where $\boldsymbol{m}$ denotes the $3 \times T$ dimensional random vector of writing movement; $\boldsymbol{\mu}$ is mean or expectation of the distribution; $\boldsymbol{\Sigma}$ is a covariance matrix; $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$.

Let $\boldsymbol{\omega}$ denotes the parameters of the distribution (i.e. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$), the writing movement distribution is written in the following notation:

$$\boldsymbol{m} \sim \mathrm{p}_{\boldsymbol{\omega}}(\boldsymbol{m}) = \mathrm{N}(\boldsymbol{m}|\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{13}$$

To facilitate the learning process, simple multivariate normal distributions are used to specify diagonal covariance matrices, rather than the full covariance ma-

trices. The writing sequence of a stroke's trajectory points is sorted in ascending order of $p_{\mathrm{x}}, p_{\mathrm{y}}$.

3.3.2. Writing Trajectory Model Optimization by DE

The parameters of the writing trajectory model need to be optimized to maximize the expected score given by the stroke evaluation subsystem as reported in Section 3.2. Because the writing trajectory model is a distribution, the expectation is naturally selected to measure its performance. The overall objective function (or fitness function) is designed as follows:

$$\mathrm{f}(\boldsymbol{\omega}) = \mathrm{E}_{\boldsymbol{m}\sim\boldsymbol{p}_{\boldsymbol{\omega}}(\boldsymbol{m})}[\mathrm{Score}(\mathrm{W}(\boldsymbol{m}))], \tag{14}$$

where $\mathrm{f}(\cdot)$ is the fitness function; $\boldsymbol{m}$ denotes the trajectory random vector; $\boldsymbol{\omega}$ denotes the parameters of the distribution (i.e. $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$); $\mathrm{W}(\cdot)$ presents the writing process of the robotic system; $\mathrm{Score}(\cdot)$ denotes a stroke scoring function. In this case, the expectation is the long-run average value of the trajectory samples that are drawn from the writing movement distribution.

The optimal means and covariance matrices of the trajectory distributions for every stroke must be obtained, which is implemented by the DE. In every generation, by using differential mutation and crossover operations, the population of parameter vectors (or individuals in EA terminology) varies; a writing robot writes strokes based on the trajectories drawn from the distributions that parametrized by the vectors in both the original and variant populations; then, the fitness scores of the vectors are computed for the better individual selection. The higher scoring parameter vectors survive to the next generation. The iterations continue until the objective is optimized or the maximum number of iterations reached. The training procedure is summarized in the pseudo-code as shown in Algorithm 1.

3.4. Robotic System

As depicted in Fig. 3, the robotic system used in this research consists of a 4-axis robotic arm and a camera mounted on a bracket. A brush pen is mounted

Algorithm 1 Strokes Trajectory Model Optimization by DE.

Require: Np: the size of population; G: the maximum number of generations; $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_{Np}$: the parameters of the multivariate normal distributions;

Ensure: optimal parameter $\boldsymbol{\omega}_\Delta$

initialize $\boldsymbol{\omega}_i, i = 1, 2, ..., Np$ by Eq. 6 and $g = 1$;
repeat
  MUTATION:
  for each $i \in [1, Np]$ **do**
    perform the mutation operation by Eq. 7 to obtain mutant vector $\boldsymbol{v}_i(g)$.
  end for
  CROSSOVER:
  for each $i \in [1, Np]$ **do**
    perform the crossover operation by Eq. 8 to obtain offspring vector $\boldsymbol{u}_i(g)$.
  end for
  SELECTION:
  Robot writes strokes by using trajectories drawn from $\mathrm{p}_{\boldsymbol{u}_i}(\boldsymbol{j})$, then the image samples $\{\boldsymbol{x}_{\boldsymbol{u}_i,1}, \boldsymbol{x}_{\boldsymbol{u}_i,2}, ..., \boldsymbol{x}_{\boldsymbol{u}_i,l}\}$ are obtained.
  Robot writes strokes by using trajectories drawn from $\mathrm{p}_{\boldsymbol{\omega}_i}(\boldsymbol{j})$, then the image samples $\{\boldsymbol{x}_{\boldsymbol{\omega}_i,1}, \boldsymbol{x}_{\boldsymbol{\omega}_i,2}, ..., \boldsymbol{x}_{\boldsymbol{\omega}_i,l}\}$ are obtained.
  if $\frac{1}{l}\sum_{k=1}^{l} \mathrm{Score}(\boldsymbol{x}_{\boldsymbol{\omega}_i,k}) < \frac{1}{l}\sum_{k=1}^{l} \mathrm{Score}(\boldsymbol{x}_{\boldsymbol{u}_i,k})$ **then**
    $\boldsymbol{\omega}_i \leftarrow \boldsymbol{u}_i$
    if $\frac{1}{l}\sum_{k=1}^{l} \mathrm{Score}(\boldsymbol{x}_{\boldsymbol{\omega}_i,k}) < \frac{1}{l}\sum_{k=1}^{l} \mathrm{Score}(\boldsymbol{x}_{\Delta,k})$ **then**
      $\boldsymbol{\omega}_\Delta \leftarrow \boldsymbol{\omega}_i$
      save $\{\boldsymbol{x}_{\boldsymbol{\omega}_i,1}, \boldsymbol{x}_{\boldsymbol{\omega}_i,2}, ..., \boldsymbol{x}_{\boldsymbol{\omega}_i,l}\}$ as $\{\boldsymbol{x}_{\Delta,1}, \boldsymbol{x}_{\Delta,2}, ..., \boldsymbol{x}_{\Delta,l}\}$;
    end if
  end if
  g=g+1
until model converges or $g > G$

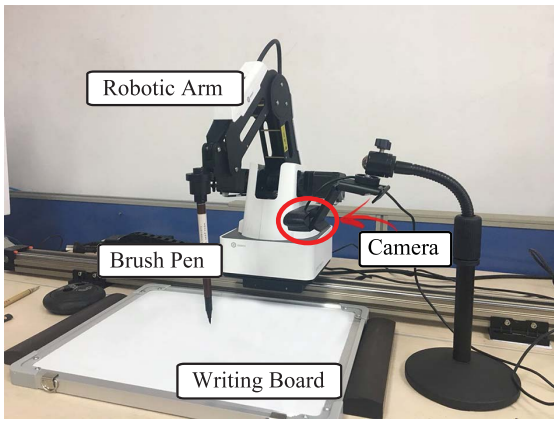

Figure 3: The robotic hardware for writing Chinese strokes.

at the end-effector of the robotic arm. The writing occurs within the working range of the arm. A whiteboard, placed flat in front of the robot, is the writing area for the robot.

A designed conversion function converts positions of the stroke trajectories into the calligraphic robot coordinates, $(p_{\mathrm{x}}, p_{\mathrm{y}}, p_{\mathrm{x}})$, which is given as follows:

$$\begin{cases} p_{\mathrm{x}}^{\gamma} = x_{\mathrm{S}} + \gamma \cdot p_{\mathrm{x}} \\ p_{\mathrm{y}}^{\gamma} = y_{\mathrm{S}} + \gamma \cdot p_{\mathrm{y}} \\ p_{\mathrm{z}}^{\gamma} = z_{\mathrm{S}} + \gamma \cdot p_{\mathrm{z}} \end{cases} \tag{15}$$

where γ denotes a scale parameter that controls the size of the strokes; x_{S}, y_{S}, and z_{S} jointly define the initial position for each stroke; p_{x}, p_{y}, and p_{z} jointly define the position of the end-effector at a certain point of time.

Since only the joint angle values control the electrical motor of the robot arm, the obtained writing trajectory is transformed from the sequence of the end-effector positions to the joint parameters. The transformation process is done by inverse kinematics calculation.

The configuration of the robotic arm is illustrated in Fig. 4, which includes the setup of joints, links and coordinate frames of joints. The robotic arm has four linked parts, (d, a_1, a_2, and a_3, with lengths of 103, 140, 160, and 70 mm, respectively) and four revolute joints (three active joints o_1, o_2, o_3, and a passive

joint o_4). The origin coordinate frame is defined by x_0, y_0, and z_0. In this setup, the z_0-axis is vertical with the writing board; the y_0-axis is parallel with the writing board; and the x_0-axis is vertical with the plane that is defined by the axes x_0 and z_0.

The Denavit and Hartenberg (D–H) convention is used to analyze the forward and inverse kinematics of the robot's manipulator. The D–H parameters are listed in Table 2. In this table, α_{i-1}, a_{i-1}, d_i, and θ_i are link twist, link length, link offset, and joint angle, respectively. The inverse kinematics analysis of the robot arm is obtained from the forward kinematics. Thus, if the positions of the robotic arm are obtained, the four joint angles are calculated using the following equations:

$$\theta_1 = arctan\frac{p_y}{p_x} \tag{16}$$

$$\theta_2 = arctan\frac{(p_x c_1 + p_y s_1)(a_1 + a_2 c_3) + (p_z - d_1)a_2 s_3}{(a_1 + a_2 c_3)^2 a_2^3 {s_3}^2 - (p_x c_1 + p_y s_1)(a_1 + a_2 c_3) - (p_z - d)a_2 s_3} \tag{17}$$

$$\theta_3 = arctan\frac{(p_x c_1 + p_y s_1)^2 + (p_z - d)^2 - a_1^2 - a_2^2}{\sqrt{2a_1 a_2 - (p_x c_1 + p_y s_1)^2 - (p_z - d)^2 + a_1^2 + a_2^2}} \tag{18}$$

$$\theta_4 = -(\theta_2 + \theta_3) \tag{19}$$

where p_x, p_y, and p_z denote the elements of the position vector, respectively; s_i and c_i represent $sin\theta_i$ and $cos\theta_i$, respectively.

After writing a stroke, the robotic arm returns to a pre-defined initial position; then, the camera captures the written stroke as an image, which is in turn delivered to the stroke evaluation subsystem. To be consistent with the representation of strokes in the training dataset, the black strokes with a white background are then inverted to white lines with a black background.

4. Experiments

The implementation of the experiment mainly involves two training processes to train the stroke evaluation model using CAE and the writing trajectory

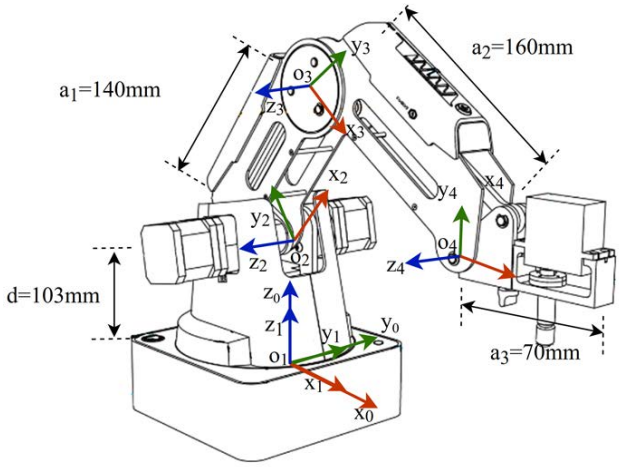

Figure 4: The configuration of the robot arm.

Table 2: D–H parameter table

Link	Link Twist	Link Length	Link Offset	Joint Angle
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d	θ_1
2	$\frac{\pi}{2}$	0	0	θ_2
3	0	a_1	0	θ_3
4	0	a_2	0	θ_4

models using DE. A training dataset and a test dataset of nine types of Chinese character strokes were used in this experiment, to verify the function of the stroke evaluation models.

4.1. Training and Test Dataset

The training dataset used in this work consists of nine types of Chinese character stroke images. The stroke extraction method proposed by Lian et al. [39] was applied in this work to obtain a stroke image dataset from a number of Chinese calligraphic textbooks. The dataset contains 4,500 of greyscale images,

each with a resolution of 28×28; each stroke has 500 samples. A part of the stroke images in the training dataset are illustrated in Fig. 5. From top to bottom, the strokes are as follows: (a) horizontal, (b) short left-falling, (c) long left-falling, (d) right-falling, (e) horizontal and left-falling, (f) vertical and turn-right hook, (g) horizontal, fold and hook, (h) vertical, fold and curved-hook, and (i) horizontal and curved-hook.

The test dataset of stroke images used for functional verification was obtained based on the following:

Step 1. For each of the nine stroke types, 100 stroke test images with different quality were collected through two channels: 33 of the test images were collected from the training dataset, and the remaining 67 were collected from the intermediate writing results produced by using the approach proposed in Chao et al. [17].

Step 2. Ten undergraduate students were individually invited and appropriately consented to classify all 900 stroke images into three grades of quality, including poor, acceptable, and good, to support this research.

Step 3. All the 900 stroke images were labeled by the levels that most assessors chose. The test images, which obtained scores in a 5-5 tie, are excluded in the test dataset and replaced by other stroke images.

4.2. Functional Verification of the Stroke Evaluation Model

The stroke evaluation model, which is presented in Section 3, was trained by using the training dataset of the strokes; then, the test stroke images, which are labeled in three grades, were used to verify that the stroke evaluation model can measure the quality of the strokes written by the calligraphy robot.

All 900 test images were scored by the evaluation model. The evaluation results are summarized in Fig. 6. For every type of the strokes, the box plot is used as a visual statistical tool to display the distribution of the evaluation results. Each plot presents the relationships between the scores from the evaluation model and grades from human regarding the same stroke (i.e. quality of the

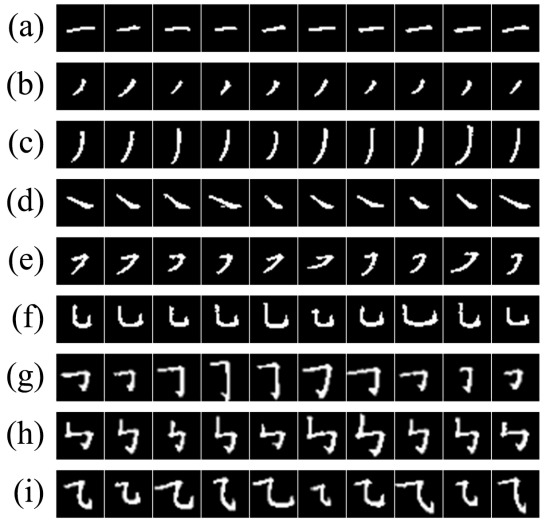

Figure 5: Exampler training samples used in the experiment, each row shows one type of a stroke with various variants.

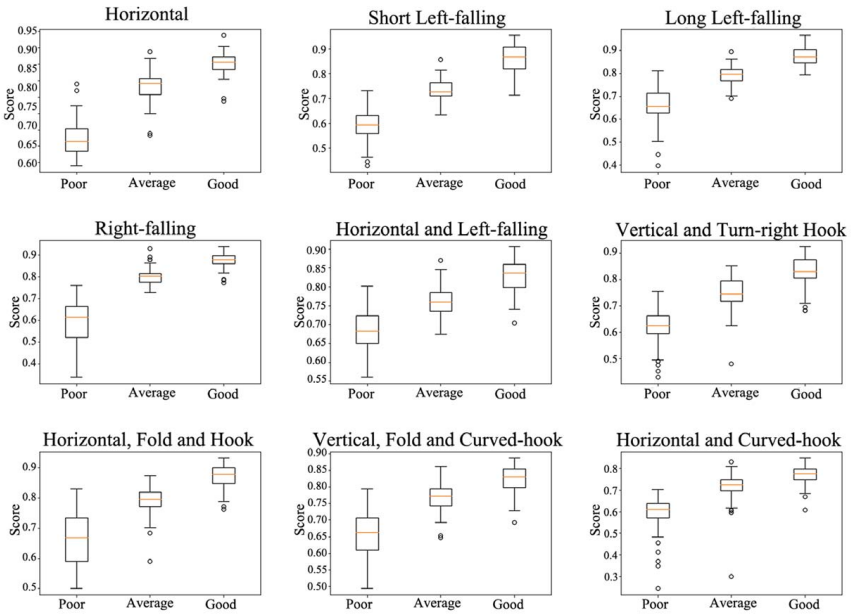

Figure 6: Box plots of scores by the stroke evaluation model. Each box plot presents a positive correlation between the scores by evaluation model and the quality levels of the strokes.

stroke). For example, the "horizontal stroke" plot has three groups including "Poor", "Average" and "Good". For poor strokes, the bulk of scores (between first quartile and third quartile) lies below 0.7; for average strokes, the bulk of scores (between first quartile and third quartile) lies between 0.8 and 0.85; for good strokes, the bulk of scores (between first quartile and third quartile) lies above 0.85. This figure clearly demonstrates that there is a positive correlation between the scores led by evaluation model and the quality levels graded by the assessors.

4.3. Writing Trajectory Model Generation

The generated writing trajectory models for strokes were also validated and evaluated. Each model was generated and optimized for one stroke, and thus nine models were generated and tested in this experiment. For each model, the number of trajectory points (T), the population size (Np), the maximum number of generations (G), the scale factor (F), and the crossover rate (Cr) were set as 6, 60, 2000, 0.9, and 0.6 in this experiment, respectively.

4.3.1. Learning Process

Each stroke costed a total of 2,000 generations for the stroke trajectory model to learn. The learning progress was divided in 15 stages for each stroke trajectory model, each stage represents 133 generations. Each stage was illustrated as a representative writing results as shown in Fig. 7; therefore, the training history of each model is represented as 15 points in time as shown in each row of the figure. Each row in Fig. 7 illustrates continuing improvement for a stroke trajectory model. The progress for the nine strokes was almost identical: during the early generations, the written strokes were shapeless and very difficult to recognize; in the medium generations, the written strokes showed rough target shapes; in the final generations, the writing results demonstrated strokes of high quality.

In addition, as shown in Fig. 8, the performance (i.e., fitness) of the trajectory model steadily increased over the training process of 2,000 generations. The

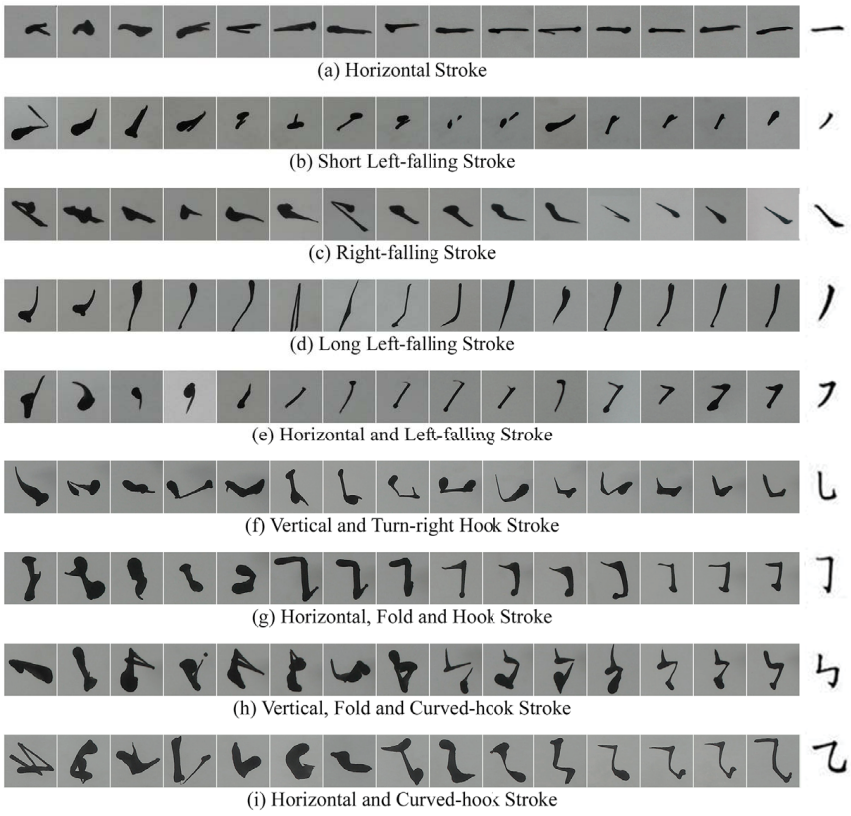

Figure 7: Learning progress of the nine strokes. The images in a row are representatives of trajectory written results along the timeline of generations.

solid points indicate the average fitness of all the individuals in each generation. The range of the fitness of each 100 generations is indicated by a vertical line. In the beginning phase of the training, the fitness range was large; however, the ranges were significantly reduced during the optimization process. Noticeably, each learning curve is smooth, and there is not any sharp change in these curves. The smooth curves imply that the proposed system has good learning stability.

4.3.2. Writing Results

The robotic writing process is illustrated in Fig. 9 where the robotic arm is writing a horizontal, fold and hook stroke. The action sequence in this figure

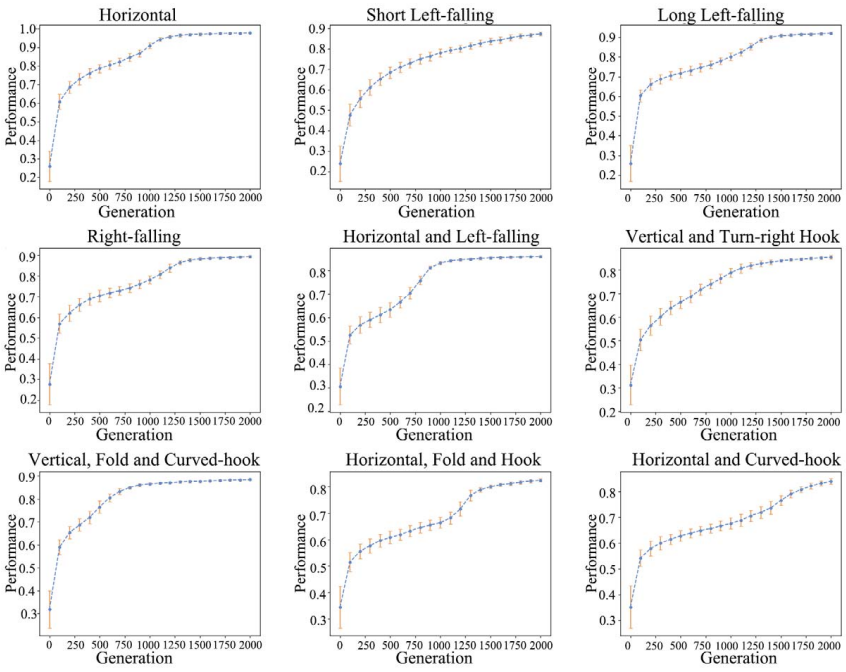

Figure 8: Convergence curves for the learning process of the nine strokes. The solid points indicate the average fitness of all the individuals in each generation. The range of the fitness of all the individuals in each generation is indicated by a solid line between the minimum and maximum values.

is indicated by the arrows. The final writing results for all nine types of stroke are shown in Fig. 10; 25 stroke images were generated for each stroke. All these strokes were written by the calligraphy robot using the trajectories drawn from the optimal trajectory distributions. These writing results are of very high quality, which exhibited the effectiveness of the proposed system. These results also demonstrated that the writing results regarding the same stroke are slightly different from each other. Therefore, the proposed system can learn the strokes in a flexible way, instead of simply learning the trajectory positions as used by most of the existing approaches in the literature.

Furthermore, in order to clearly demonstrate the diversity generation ability of the proposed work, Fig. 11 demonstrates two types of writing styles, which are

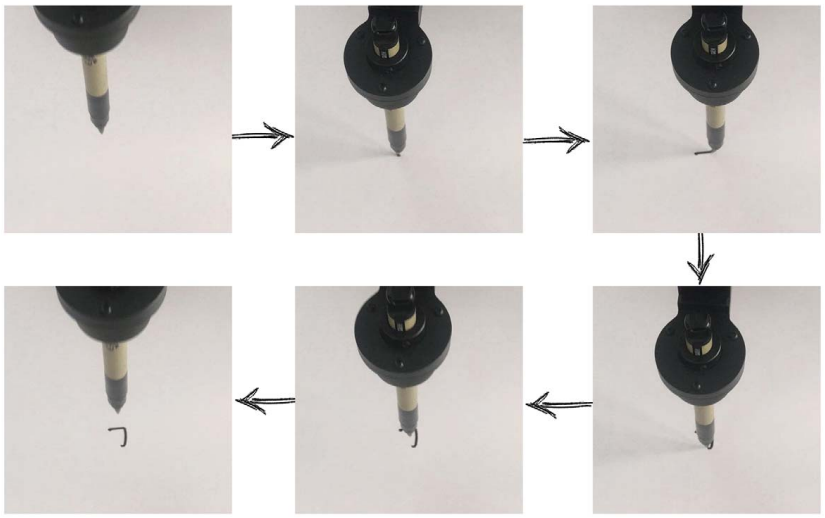

Figure 9: Robotic arm is writing a stroke.

labeled as "dull" and "sharp", of the "Right-falling Stroke" and "Vertical and Turn-right Hook Stroke". Through sampling corresponding multivariate normal distribution, various writing trajectories with good writing quality were generated by the calligraphy robot. Each trajectory in the same category contains variations in trajectory length and inclination of the strokes amongst others, which collectively demonstrate the capability of the proposed system in producing strokes with different styles.

4.4. Discussion and Comparison

The experimental results demonstrate that the proposed system successfully develops the capacity for a robot to write Chinese character strokes. In reference to the existing approaches reported in the literature, the proposed approach has two distinctive advantages:

(1) Data-driven mechanism to evaluate the quality of strokes based on the features of a collection of variations. Many existing evaluation for robotic calligraphy [10, 11, 12, 13, 14, 15, 16] was performed based on a

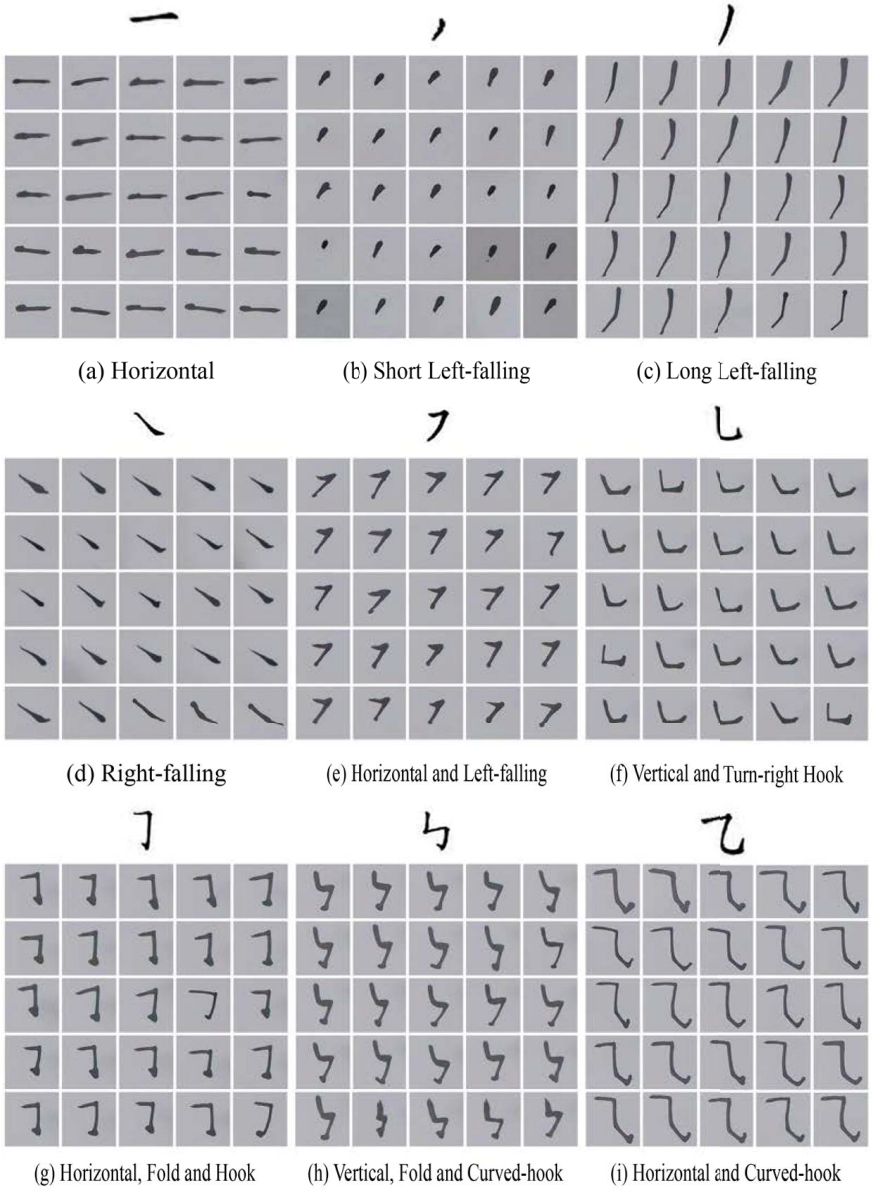

Figure 10: Final writing results of all nine strokes. All these strokes were written by the calligraphy robot using trajectories drawn from the optimal trajectory distributions.

predefined, simplistic evaluation criterion. Their evaluation methods required a large number of human engineer's efforts; these methods usually limit the

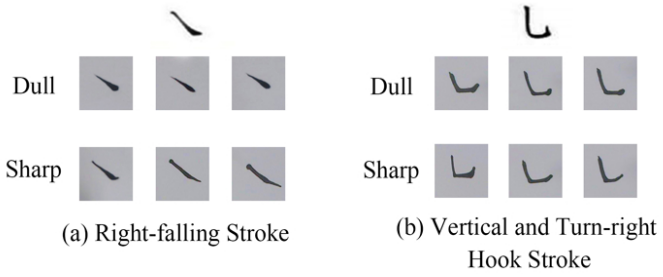

Figure 11: Selected strokes of "Right-falling" and "Vertical and Turn-right Hook" demonstrate different styles.

diversity of the writing results due to its hard-wired mechanism. However, the proposed evaluation method applied a CAE to extract the features of stroke images from a prepared stroke image dataset. The data-driven mechanism ensured the diversity of the writing results. The writing results in the proposed system were compared with the entire stroke dataset, rather than with a single fixed reference point. The proposed evaluation method is also readily applicable to evaluate the quality of Chinese characters.

Note that the GAN-based approach is also data-driven, but its evaluation model (i.e. the discriminative model) discriminates the sources of the input instances, either provided by a human or generated by a calligraphy robot [17]. The discrimination in this work cannot exhibit the writing quality of the calligraphy robot. However, the proposed evaluation model here can determine and make use of the quality of strokes, so as to improve the quality of the writing results. In addition, in contrast to the GAN-based method, a well-trained stroke evaluation subsystem in the proposed system leads to a steadily improving training process, but the GAN-based approach involves some oscillations.

(2) DE-based framework effectively applied to obtain the trajectory writing models of strokes. In contrast to the work reported in [10, 11, 12, 13], the work reported herein is able to discover the optimal generative models of stroke trajectory generation, which is more challenging compared to

the optimization of stroke trajectories generation itself. Due to the non-linear feature of such an optimization problem, it is difficult, if not impossible, for the gradient-based approaches to obtain a stable solution efficiently. For example, the GAN-based robotic calligraphy system used 5,000 epochs for training, but the performance was hard to converge. In contrast, DE was applied in this work to find the optimal trajectory models for strokes, and the experimental results demonstrated the efficiency of the proposed DE-based learning system, which discovered the optima in just 2,000 generations. In addition, the writing performance was stable without any incorrect trajectories generated. Last but not least, the learning curve of the proposed approach is very deep, witnessed by some reasonable results generated in the first 200 generations. Furthermore, another potential reason that the proposed framework showed high efficiency is the lower dimension of the trajectory parameters. However, if we use non-linear representations with high dimensional data to replace our current trajectory model, our DE-base optimization process is still available and might still perform the high-efficiency characteristic; in addition, several other heuristic search algorithms that are good at large scale data, such as Competitive Swarm Optimizer (CSO) [40], can be also introduced into our framework.

5. Conclusion

This paper presented a new learning framework for calligraphy robots to autonomously learn to write Chinese strokes. The proposed framework used CAE to build the mechanism to evaluate the quality of strokes and applied DE to automatically develop writing trajectory models for strokes. The proposed system is driven by data, and thus it is robust and of wider applicability. The proposed system was evaluated using a dataset with nine strokes; the writing results show that the robot powered by the proposed system successfully learned the ability to write high quality and diverse strokes.

There is still room to improve this work. We believe that multivariate normal distributions can be applied as a simpler representation mechanism for the

trajectory generative model used in this approach; therefore, due to the artificial neural network's better representational ability of multivariate normal distributions, we will focus on incorporate neural networks into the trajectory generative model in future. In addition, the current stroke writing system learns Chinese strokes only; however, we will further develop the system to write complete Chinese characters. Third, the proposed algorithm ignored the trajectory writing order of the sampling points, further efforts will focus on this.

Acknowledgment

The authors are very grateful to the anonymous reviewers for their constructive comments which have helped significantly in revising this work. This work was supported by the National Natural Science Foundation of China (No.61673322, 61673326, and 91746103), the Fundamental Research Funds for the Central Universities (No. 20720160126), Natural Science Foundation of Fujian Province of China (No. 2017J01128 and 2017J01129), and the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement (No. 663830).

References

References

[1] J. H. M. Lam, Y. Yam, Stroke trajectory generation experiment for a robotic Chinese calligrapher using a geometric brush footprint model, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 2315–2320. doi:10.1109/IROS.2009.5354709.

[2] Y. Sun, H. Qian, Y. Xu, A geometric approach to stroke extraction for the Chinese calligraphy robot, in: IEEE International Conference on Robotics and Automation, 2014, pp. 3207–3212. doi:10.1109/ICRA.2014.6907320.

[3] Y. Sun, H. Qian, Y. Xu, Robot learns Chinese calligraphy from demonstrations, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 4408–4413. doi:10.1109/IROS.2014.6943186.

[4] B. Zhao, M. Yang, H. Pan, Q. Zhu, J. Tao, Nonrigid point matching of Chinese characters for robot writing, in: IEEE International Conference on Robotics and Biomimetics, 2017, pp. 762–767. doi:10.1109/robio.2017.8324509.

[5] F. Chao, Y. Huang, C. Lin, L. Yang, H. Hu, C. Zhou, Use of automatic Chinese character decomposition and human gestures for chinese calligraphy robots, IEEE Transactions on Human-Machine Systems 49 (1) (2019) 47–58. doi:10.1109/THMS.2018.2882485.

[6] N. Huebel, E. Mueggler, M. Waibel, R. Dandrea, Towards robotic calligraphy, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5165–5166. doi:10.1109/IROS.2012.6386275.

[7] F. Chao, Y. Huang, X. Zhang, C. Shang, L. Yang, C. Zhou, H. Hu, C. Lin, A robot calligraphy system: From simple to complex writing by human gestures, Engineering Applications of Artificial Intelligence 59 (2017) 1–14. doi:10.1016/j.engappai.2016.12.006.

[8] Z. Ju, X. Ji, J. Li, H. Liu, An integrative framework of human hand gesture segmentation for human–robot interaction, IEEE Systems Journal 11 (3) (2017) 1326–1336. doi:10.1109/JSYST.2015.2468231.

[9] D. Chen, G. Li, Y. Sun, J. Kong, G. Jiang, H. Tang, Z. Ju, H. Yu, H. Liu, An interactive image segmentation method in hand gesture recognition, Sensors 17 (2).

[10] F. Yao, G. Shao, J. Yi, Extracting the trajectory of writing brush in Chinese character calligraphy, Engineering Applications of Artificial Intelligence 17 (6) (2004) 631–644. doi:10.1016/j.engappai.2004.08.008.

[11] F. Yao, G. Shao, J. Yi, Trajectory generation of the writing brush for a robot arm to inherit block style Chinese character calligraphy techniques, Advanced Robotics 18 (3) (2004) 331–356. doi:10.1163/156855304322972477.

[12] K. W. Kwok, S. M. Wong, K. W. Lo, Y. Yam, Genetic algorithm-based brush stroke generation for replication of Chinese calligraphic character, in: IEEE International Conference on Evolutionary Computation, 2006, pp. 1057–1064. doi:10.1109/CEC.2006.1688426.

[13] S. Mueller, N. Huebel, M. Waibel, R. Dandrea, Robotic calligraphy - learning how to write single strokes of Chinese and Japanese characters, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, pp. 1734–1739. doi:10.1109/IROS.2013.6696583.

[14] W. Li, Y. Song, C. Zhou, Computationally evaluating and synthesizing Chinese calligraphy, Neurocomputing 135 (2014) 299–305. doi:10.1016/j.neucom.2013.12.013.

[15] M. Wang, Q. Fu, X. Wang, Z. Wu, M. Zhou, Evaluation of Chinese calligraphy by using dbsc vectorization and icp algorithm, Mathematical Problems in Engineering 2016 (2016) 1–11. doi:10.1155/2016/4845092.

[16] Z. Ma, J. Su, Aesthetics evaluation for robotic Chinese calligraphy, IEEE Transactions on Cognitive and Developmental Systems 9 (1) (2017) 80–90. doi:10.1109/TCDS.2016.2645598.

[17] F. Chao, J. Lv, D. Zhou, L. Yang, C.-M. Lin, C. Shang, C. Zhou, Generative adversarial nets in robotic Chinese calligraphy, in: 2018 IEEE International Conference on Robotics and Automation, 2018, pp. 1104–1110.

[18] J. Masci, U. Meier, D. C. Ciresan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: International Conference on Artificial Neural Networks, 2011, pp. 52–59. doi:10.1007/978-3-642-21735-7_7.

[19] T. Salimans, J. Ho, X. Chen, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning, arXiv: Machine Learning (2017) 1–13.
URL https://arxiv.org/pdf/1703.03864

[20] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, J. Clune, Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning., arXiv: Neural and Evolutionary Computing (2017) 1–16.
URL https://arxiv.org/abs/1712.06567

[21] H. Li, Q. Zhang, J. Deng, Biased multiobjective optimization and decomposition algorithm, IEEE Transactions on Cybernetics 47 (1) (2017) 52–66. doi:10.1109/TCYB.2015.2507366.

[22] X. Ma, Q. Zhang, G. Tian, J. Yang, Z. Zhu, On tchebycheff decomposition approaches for multiobjective evolutionary optimization, IEEE Transactions on Evolutionary Computation 22 (2) (2018) 226–244. doi:10.1109/TEVC.2017.2704118.

[23] Y. Zhou, Y. Xiang, Z. Chen, J. He, J. Wang, A scalar projection and angle-based evolutionary algorithm for many-objective optimization problems, IEEE Transactions on Cybernetics (2018) 1–12doi:10.1109/TCYB.2018.2819360.

[24] R. Storn, K. V. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.

[25] K. Price, R. M. Storn, J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series), Springer-Verlag New York, Inc., 2005.

[26] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-

the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31. doi:10.1109/TEVC.2010.2059031.

[27] S. Das, S. S. Mullick, P. N. Suganthan, Recent advances in differential evolution - an updated survey, Swarm and evolutionary computation 27 (2016) 1–30. doi:10.1016/j.swevo.2016.01.004.

[28] V. A. Knyaz, O. V. Vygolov, V. V. Kniaz, Y. Vizilter, V. Gorbatsevich, T. Luhmann, N. Conen, Deep learning of convolutional auto-encoder for image matching and 3d object reconstruction in the infrared range, in: IEEE International Conference on Computer Vision Workshop, 2017, pp. 2155–2164. doi:10.1109/iccvw.2017.252.

[29] W. Luo, J. Li, J. Yang, W. Xu, J. Zhang, Convolutional sparse autoencoders for image classification, IEEE Transactions on Neural Networks and Learning Systems (2017) 1–6.doi:10.1109/TNNLS.2017.2712793.

[30] M. Ribeiro, A. E. Lazzaretti, H. S. Lopes, A study of deep convolutional auto-encoders for anomaly detection in videos, Pattern Recognition Letters 105 (2017) 13–22. doi:10.1016/j.patrec.2017.07.016.

[31] H. Huang, X. Hu, Y. Zhao, M. Makkie, Q. Dong, S. Zhao, L. Guo, T. Liu, Modeling task fmri data via deep convolutional autoencoder, IEEE Transactions on Medical Imaging (2017) 411–424.doi:10.1109/TMI.2017.2715285.

[32] H. Li, Q. Zhang, Q. Chen, L. Zhang, Y. Jiao, Multiobjective differential evolution algorithm based on decomposition for a type of multiobjective bilevel programming problems, Knowledge Based Systems 107 (2016) 271–288. doi:10.1016/j.knosys.2016.06.018.

[33] M. Z. Ali, N. H. Awad, P. N. Suganthan, R. G. Reynolds, A modified cultural algorithm with a balanced performance for the differential evolution frameworks, Knowledge Based Systems 111 (2016) 73–86. doi:10.1016/j.knosys.2016.08.005.

[34] H. Shao, H. Jiang, F. Wang, H. Zhao, An enhancement deep feature fusion method for rotating machinery fault diagnosis, Knowledge Based Systems 119 (2017) 200–220. doi:10.1016/j.knosys.2016.12.012.

[35] E. Hancer, B. Xue, M. Zhang, Differential evolution for filter feature selection based on information theory and feature ranking, Knowledge Based Systems 140 (2018) 103–119. doi:10.1016/j.knosys.2017.10.028.

[36] Z. Li, Q. Zhang, A simple yet efficient evolution strategy for large-scale black-box optimization, IEEE Transactions on Evolutionary Computation 22 (5) (2018) 637–646. doi:10.1109/TEVC.2017.2765682.

[37] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations (2015) 1–15.
URL https://arxiv.org/abs/1412.6980

[38] J. Ye, Cosine similarity measures for intuitionistic fuzzy sets and their applications, Mathematical and Computer Modelling 53 (1) (2011) 91 – 97. doi:https://doi.org/10.1016/j.mcm.2010.07.022.
URL http://www.sciencedirect.com/science/article/pii/S0895717710003651

[39] Z. Lian, B. Zhao, J. Xiao, Automatic generation of large-scale handwriting fonts via style learning, in: SIGGRAPH ASIA 2016 Technical Briefs, 2016, p. 12. doi:10.1145/3005358.3005371.

[40] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Transactions on Cybernetics 45 (2) (2015) 191–204. doi:10.1109/TCYB.2014.2322602.