Solving Robotic Trajectory Sequential Writing Problem via Learning Character's Structural and Sequential Information

Quan-Feng Li, Zhi-Hua Guo, Fei Chao *Member, IEEE,* Xiang Chang, Longzhi Yang *Senior Member, IEEE,*, Chih-Min Lin, *Fellow, IEEE,* Changjing Shang, and Qiang Shen

***Abstract*—The writing sequence of numerals or letters often affects aesthetic aspects of the writing outcomes. As such, it remains a challenge for robotic calligraphy systems to perform, mimicking human writers' implicit intention. This paper presents a new robot calligraphy system that is able to learn writing sequences with limited sequential information, producing writing results compatible to human writers with good diversity. In particular, the system innovatively applies a Gated Recurrent Unit (GRU) network to generate robotic writing actions with the support of a pre-labeled trajectory sequence vector. Also, a new evaluation method is proposed that considers the shape, trajectory sequence, and structural information of the writing outcome, thereby helping ensure the writing quality. A swarm optimization algorithm is exploited to create an optimal set of parameters of the proposed system.The proposed approach is evaluated using Arabic numerals, and the experimental results demonstrate the competitive writing performance of the system against state-of-the-art approaches regarding multiple criteria (including FID, MAE, PSNR, SSIM, and PerLoss), as well as diversity performance concerning variance and entropy. Importantly, the proposed GRU-based robotic motion planning system, supported with swarm optimization can learn from a small dataset, while producing calligraphy writing with diverse and aesthetically pleasing outcomes.**

***Index Terms*—Robotic calligraphy, robotic motion planning, Gated Recurrent Unit network.**

I. Introduction

Robotic advances are making an increasingly significant impact on human culture preservation and culture education promotion through robotic writing, dance and painting etc. [1]. Such advances continuously contribute to better addressing the challenges of robotic calligraphy writing, given the underpinning of complicated control algorithms to drive robotic end-effectors to write numerals, letters, or complex Chinese characters [2]–[6]. Robotic writing still faces several key challenges despite the promising results achieved so far. Many learning-based approaches to robotic calligraphy have attempted to build automatic learning calligraphy robots, but the writing sequences generated by such approaches are usually not in accordance with human writing habits. One idea is to manually pre-define a robot's joint angles for each writing action to write characters [7], but such methods usually require heavy manual inputs from human engineers. Another good attempt is to use the imitation learning method [8]–[11], which is independent to the control or programming model of the robot. However, this method is still labor-intensive albeit a noticeable improvement, in addition to a side effect of poor generalization ability.

Deep learning approaches, such as generative adversarial nets (GAN) and long and short term memory (LSTM), have also been innovatively applied to robotic calligraphy to address the key challenge of writing sequence generation with the support of large training data sets. For example, several studies [12]–[16] employed GAN-based methods to produce stroke trajectories, one of which was improved by using heuristic search algorithms leading to better performance [17]. Although these methods can realize the writing of Chinese strokes, the writing sequence of strokes is often not in accordance with the rules predefined by humans. This was further enhanced by the integration of LSTM networks [18]–[21]. In Rahmatizade's work, GAN was used to transform an input image into the low dimensional space, and LSTM was then applied to predict the joint parameter values of the robot [9]. This approach effectively addresses the writing sequence challenge, but its success is based on the availability of a large training dataset with a wealth of action sequence information.In addition, related studies such as trajectory planning for hypersonic reentry vehicles [22], [23], unmanned vehicles [24], and spacecrafts [25] share the similar research challenge with our robotic writing trajectory planning.

Another key challenge in robotic calligraphy is the aesthetic evaluation of robotic writing results with particular considerations of diversity. Most of the existing robotic writing systems do not have a realistic aesthetic evaluation mechanism to critically assess the writing performance [3]. Instead, several studies adopted the Fréchet inception distances (FID) or the restoring accuracy of an Autoencoder network to represent the aesthetic performance [4], [26], [27]. These methods well examined the writing results based on distribution, but often ignored the structural information of letters or numerals representing diversity. This leads to a research gap of effective aesthetic evaluation functions based on structural features for

Q.-F. Li, Z.-H. Guo, and F. Chao are with the Department of Artificial Intelligence, School of Informatics, Xiamen University, China. F. Chao, X. Chang, C. Shang, and Q. Shen are with the Institute of Mathematics, Physics, and Computer Science, Aberystwyth University, UK. L. Yang is with the Department of Computer and Information Sciences, Northumbria University, UK. C.-M. Lin is with the Department of Electrical Engineering, Yuan Ze University, Taiwan. Corresponding Author: Fei Chao e-mail: (fchao@xmu.edu.cn)

This work was supported in part by the Natural Science Foundation of Fujian Province of China (2021J01002) and Strategic Partner Acceleration Award (80761-AU201), funded under the Ser Cymru II programme, UK.

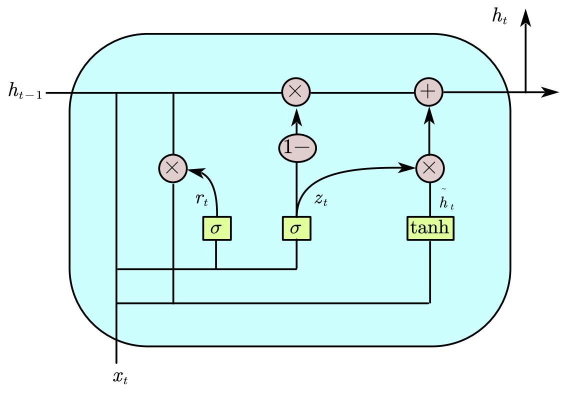


Fig. 1. Structure of GRU

better diversity.

This paper reports a new automatic writing action generator for robotic calligraphy using a Gated Recurrent Unit network (GRU) [28], [29], to address the aforementioned challenges of writing sequence and result diversity. The GRU network is comprised of a sequence of GRU units, each generating a trajectory point. The generated trajectory point is utilized by a calligraphy robot to write a part of a numeral. The image of the writing result is then fed into the next GRU unit to create the next trajectory point. When the numeral is completed, the shape similarity, writing sequence, and structural information are evaluated, which are in turn used in the objective function of a Competitive Swarm Optimization algorithm [30] to train the GRU network for better performance. The proposed system is able to perform competitively after a number of iterations.

The major contributions of this paper include 1) a GRU-based robotic motion planning system with the support of competitive swarm optimization, 2) accurate writing sequence generation based on a small dataset, and 3) the implementation of the proposed robotic calligraphy on a robot arm for calligraphy writing with diverse and aesthetically pleasing results.

II. Preliminary

A. Gated Recurrent Unit Network

The key fundamental algorithms adopted in this work is the Gated Recurrent Unit (GRU) networks, which is an improved variant of Recurrent Neural Network [31]. A typical GRU network as shown in Fig. 1 involves two "gates": an update gate z, and a reset gate r. The update gate defines the amount of previous memory saved and available to the current time step, and the reset gate specifies the way of combining the new input information with the previous memory. Given a time t, the update and reset gates can be expressed as:

$$z_t = \delta(W_z \cdot [h_{t-1}, x_t]), \tag{1}$$

$$r_t = \delta(W_r \cdot [h_{t-1}, x_t]), \tag{2}$$

where $\delta(\cdot)$ is a $\mathrm{sigmoid}(\cdot)$ activation function; W_z and W_r denote the weights of the update and reset gates, respectively; X_t represents the input at the t-th time; and h_{t-1} indicates the previous activation state. The current hidden state h_t is a linear interpolation between the previous activation state h_{t-1} and a candidate activation $\tilde{h}_t$:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{3}$$

where $\odot$ is an element-wise multiplication and $\tilde{h}_t$ is determined by:

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t]). \tag{4}$$

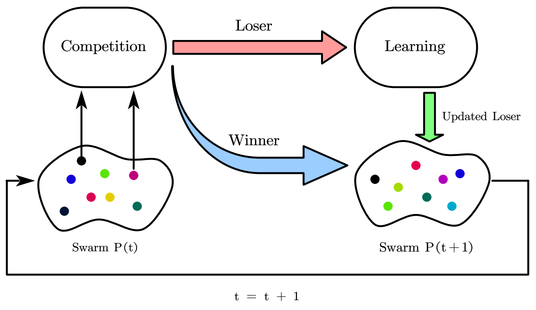


Fig. 2. Basic procedure of CSO

B. Competitive Swarm Optimizer

The Competitive Swarm Optimizer (CSO) algorithm is a heuristic search algorithm [30], inspired by the particle swarm optimization [32], [33]. In a CSO algorithm, the update of a particle involves neither the best position of every particle, nor the global best position. Instead, a pair-wise competition mechanism is used. During each optimization iteration, the particles that lose the competition with their peers will update their positions by learning from the winners.

The procedure is shown in Fig. 2. In the t-th generation, two particles are randomly selected from the population $SwarmP(t)$ to have a competition, then the winner and loser are determined by their fitness in the environment. The winner particle will retain the original position, and the loser particle will update its position by learning from the winner. After a number of these selection-competition-update cycles, the winners and losers will jointly form the population of the $(t+1)$-th generation, $(SwarmP(t+1))$, to facilitate the next iteration.

III. Proposed Method

A. Approach Overview

The proposed system allows calligraphy robots to learn to write high-quality numerals in a way close to human writing habits, especially the order of each writing part. The learning process is summarized in Fig. 3, through five subsystems:(1) VAE-based stroke feature extraction module, (2) GRU-based trajectory generator module, (3) robotic system module, (4) evaluation system module, and (5) optimization module.

First of all, a pre-processing is required to extract the mean and covariance information of sample images, so that the means and covariances are normalized to an interval to control output diversity. In order to reduce the difficulty of the pre-processing, a Variational Autoencoder (VAE) network

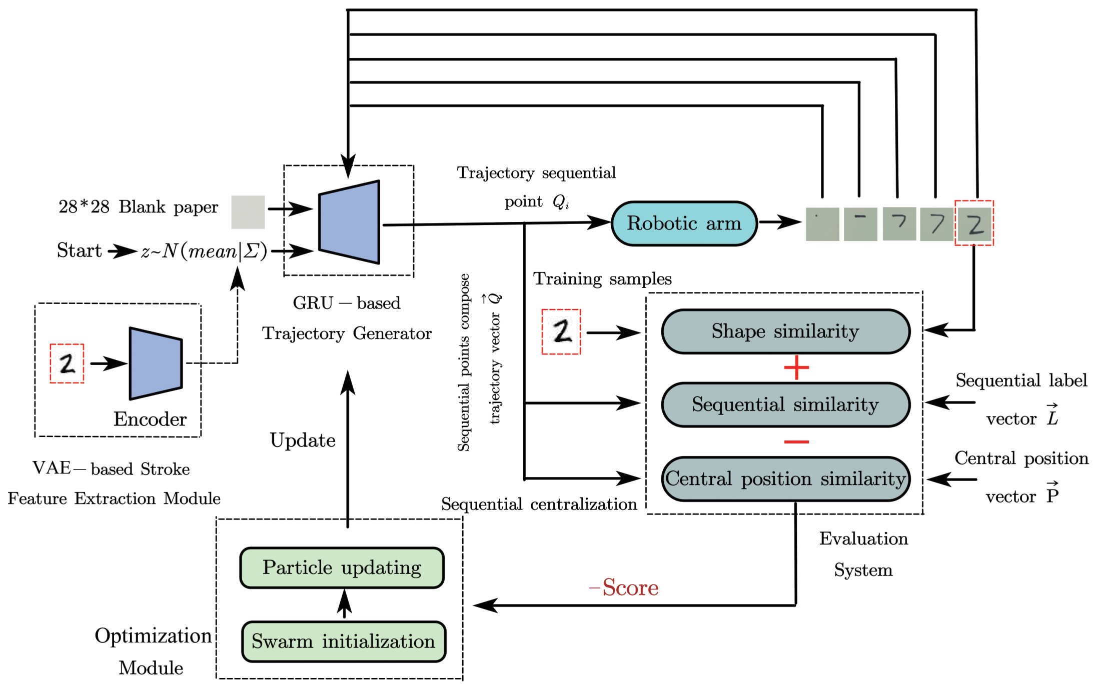


Fig. 3. Training procedure of the proposed robotic writing system.

[34] is adopted here to extract the mean and covariance information. In the pre-processing, training sample images are input to VAE, and the output is images generated by VAE. The parameters of VAE are optimized by minimizing the mean square error between the output distribution of the encoder and Kullback-Leibler (KL) divergence of the multivariate standard Gaussian distribution.

When the pre-processing has been completed, the procedure of Fig. 3 starts. The trajectory generator consists of a cyclic GRU network. The input of the GRU unit is a 28×28 blank image, and the input of the hidden layer is a Gaussian noise, $z \sim N(\mu|\Sigma)$, whose mean μ and covariance Σ are determined by the pre-processing. Then, the GRU-based generator receives the Gaussian sampling to output a new trajectory position, $\vec{Q}$. Then, the new trajectory position is converted into the robot's joint values by inverse kinematics calculation. The robot arm uses these new manipulator joint values to link the previous point output from the previous GRU cell to the new point.

A camera captures the image of the current stroke and sends the captured image to the next loop of the GRU network. This loop repeats until the numeral has been completed. Then, an evaluation system calculates (1) shape similarity between the generated result and the training sample, (2) trajectory sequence similarity between the result and its sequence label, $\vec{L}$, and (3) central point vector loss. These three similarities are regarded as the GRU's performance, which is sent to the optimization module to optimize the GRU-based generator. The writing of numerals is used as a running example for technical explanation and experimentation in this paper, but the proposed system can be readily applied to the automatic writing of other letters or characters. The implementation of each system in the training procedure is specified as follows:

B. VAE-based Stroke Feature Extraction

A traditional Autoencoder network consists of two parts: an encoder and a decoder. The encoder compresses the input, i.e. high-dimensional data, into a latent code, and the decoder attempts to reconstruct the input from the latent code. The Variational Autoencoder (VAE) network is an upgraded version of the auto-encoder network by [35], [36]. VAE learns the feature distribution of the input data set. In this work, the input of the VAE is a stroke training sample X, and the output of the VAE is a reconstruction of input data by the VAE network, denoted as Y. The main components of the VAE utilized in this work are introduced as follows:

Encoder: The encoder is mainly composed of a convolution layer and a pooling layer. In the convolution layer, the input images are convoluted to extract different features of the images. Suppose the single channel input of the encoder is image data x^{n-1}, the n-th feature map x^n can be calculated as:

$$x^n = \zeta(x^{n-1} \otimes W^n + b^n), \tag{5}$$

where $\zeta(\cdot)$ denotes an activation function $\mathrm{R}elu(\cdot)$; W^n and b^n express the weights and bias of the n-th convolution kernel, respectively. The generated feature map goes through a pooling layer which divides the input feature map into several rectangular regions and aggregates each rectangular region as its output. Particularly in this work, the outputs of the encoder are two feature vectors, including a mean value vector μ and a covariance value vector Σ.

Decoder: Assuming $\varepsilon \sim N(0, I)$, that is, ε obeys the multivariate standard Gaussian distribution $N(0, I)$, where 0 and I represent the 0 matrix mean vector and the identity matrix vector, respectively.The input of the decoder z is computed as

Eq. 6. Given the output of the $(n-1)$-th decoder layer y^{n-1}, the output of the n-th decoder layer y^n is computed as Eq. 7:

$$z = \mu + \varepsilon * \Sigma, \tag{6}$$

$$y^n = \delta(G^n y^{n-1} + c^n) \tag{7}$$

where Σ denotes a diagonal matrix with the value of vector σ as diagonal elements, $*$ denotes the vector ε is multiplied by the diagonal elements of the matrix Σ, and the result is in the form of a vector, $\delta(\cdot)$ denotes an activation function $\mathrm{sigmoid}(\cdot)$; G^n and c^n represent the weights and bias of the n-th full-connection layer. In this work, the final output of the encoder after training is a series of stroke trajectory sequences.

Loss Function: This work adopts the most commonly used loss function, that is the sum of the mean square error (MSE) between the input data X and the reconstructed data Y, and the Kullback-Leibler (KL) divergence between the encoder output and the standard Gaussian distribution. Note that the purpose of the KL divergence is to assist the encoder to obtain the posterior probability distribution $P(X|z_i)$, so as to approximate the real probability distribution $P(Z)$ of latent codes. Because $P(Z)$ is assumed to be a standard multivariate Gaussian distribution $N(0, I)$, the KL loss function is then used to calculate the distance between $P(X|z_i)$ and $N(0, I)$. The $Loss(X, Y)$ is defined as:

$$\begin{aligned} Loss(X,Y) = &\frac{1}{2m}\sum_{i=1}^{m}(x_i - y_i)^2 \\ &+ KL[N(\mu(x), \textstyle\sum(x))\|N(0,I)] \end{aligned} \tag{8}$$

where $\mu(x)$ denotes the average value of the output of the encoder; $\sum(x)$ denotes the covariance of the output of the encoder; x_i and y_i denotes a single sample in the input data X and reconstructed data Y, respectively. The parameters of the VAE are optimized by using the back-propagation algorithm due to its effectiveness and wide availability although multiple other approaches may also be applied.

The VAE network structure used in this method is introduced as follows: The encoder is a convolutional neural network, which consists of two convolution layers, two maximum pooling layers, and two fully-connected layers. The first convolution layer consists of 10 convolution kernels, each kernel's size is 5×5. Each convolution kernel generates a feature image of 12×12 pixels. The second convolution kernel contains 20 convolution kernels, each kernel's size is 5×5. Each convolution kernel generates a feature image with a resolution of 8×8 pixels. The second-largest pooling layer contains 20 convolution kernels, each of which is 2×2 in size. Each convolution kernel generates a feature image with a resolution of 4×4 pixels.

In the fourth fully-connected layer, the two outputs for the means and covariances respectively contain 28 neurons. The consideration of this setting is: Within the subsequent trajectory generator networks, i.e., GRU networks, the dimensions of the means and covariances of the pre-trained VAE encoder respectively equal to those of the means and covariances of the multivariate Gaussian distribution of the sampled trajectory points; in addition, the trajectory points sampled from the distribution are written in a blank image with the 28×28 dimensions; therefore, the dimension value, 28, is more in line with the dimensions of writing image. The decoder consists of only two fully-connected layers. The two fully connected layers have 100 and 784 neurons, respectively. After a pre-training, the encoder part is removed and the decoder is improved by using the GRU network.

Pre-training: In the VAE's pre-training phase, the input of VAE is the stroke training sample, and the output is the reconstructed image of VAE. The parameters of the network are optimized by minimizing the mean square errors of the input and output images, the KL divergences of the output distribution and multivariate Gaussian distribution of the VAE encoder, via an ordinary back-propagation method.

C. GRU-based Trajectory Generator

The trajectory generator uses a cycle-GRU network to generate the probability distribution of every trajectory point. In each cycle, the GRU network generates a mean value of a ternary Gaussian distribution. The covariance matrix of the ternary Gaussian distribution is determined by the hidden state h of the GRU network in the loop. Then, the calligraphy robot generates a digital trajectory point coordinate $Q_i = (x_i, y_i, z_i)$ by sampling the ternary Gaussian distribution. x_i, and y_i determine the horizontal and vertical coordinates of the robot in its writing range, and z_i represents the height of the brush of the robot system from the writing plane. This height can be used to control the thickness of the writing stroke (stroke width information). The robot system links from the position of the last track point to the current track point position, which is regarded as one complete writing. According to the complexity of writing each number, we can set different network loop times for different numbers. For example, the stroke of Numeral 1 is relatively simple, thus the number of network cycles is set to 3. Another example is that the stroke of Numeral 5 is more complicated, thus the number of network cycles is set to 8. The specific number of cycles for each numeral is detailed in the experimental section. If the preset number of network cycles is set to k, then all trajectory point value vectors $\vec{Q} = [Q_0, Q_1, \cdots, Q_{k-1}]$ can be finally obtained.

Fig. 4 shows the basic structure of the stroke trajectory generator model of Numeral 2. The cyclic network module circulates 5 times in total. Each cyclic network module consists of a GRU, a fully connected layer, an activation function layer, and a robot system. The size of the hidden state h of the gated recursive unit is set to 28 dimensions, the fully connected layer has 3 neurons, and the activation function uses the $sigmod$ activation function. The above parameters are set based on experimental experience. In the first recurrent network module, the input of the GRU is a 28 $\times$28 pixel blank image stretched into a one-dimensional vector p_0. The initial value h_0 of the hidden layer of the GRU is set to Gaussian noise. The mean value and covariance of Gaussian noises are determined by the encoder output of the pre-trained VAE; because it contains some characteristic information of the training sample. Subsequently, the hidden layer state h_1 of

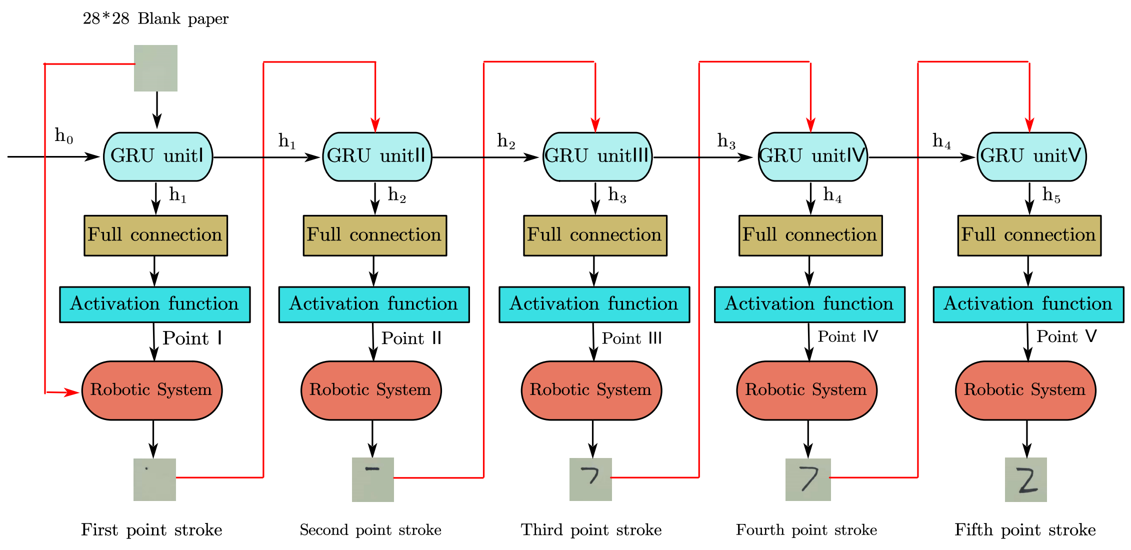


Fig. 4. A block diagram showing the structure of the generator module.

the GRU is used as the input of the fully connected layer, and a mean value of the first multivariate Gaussian distribution is obtained through the calculation of the fully connected layer and the activation function. The calculation for this step is as follows:

$$h_i = GRU\ unit_i(p_{i-1}, h_{i-1}), i \in (0, k] \tag{9}$$

where $GRU\ unit_i$ denotes the i-th GRU, h_i denotes the hidden layer state of the i-th gated recursive unit, and k represents that the network module has cycled k times. p_{i-1}, a 28×28-dimensional vector, denoting the output of the previous cycle of the GRU, which also serves as the input to the current cycle.

As mentioned above, h_i is used to predict the mean value of the multivariate Gaussian distribution through the calculation of the fully connected layer and the activation function. The covariances obtained from the pre-trained VAE are the covariances in the multivariate Gaussian distribution; then h_i are the means of the multivariate Gaussian distribution. The sampled trajectory points are obtained by sampling this multivariate Gaussian distribution. The i-th mean μ_i of the multivariate Gaussian distribution is calculated as follows:

$$\mu_i = \delta(f(h_i)), i \in (0, k] \tag{10}$$

where $f(\cdot)$ denotes the calculation of the fully connected layer, and the variable value calculated by the $\delta(\cdot)$ activation function through the fully connected layer is mapped to a value between 0 and 1.

The i-th covariance of the multivariate Gaussian distribution adopts the covariance output by the encoder of the pre-trained VAE, which is denoted as Σ_i. In this way, we can sample the multivariate Gaussian distribution and three-dimensional coordinates $Q_1(x_1, y_1, z_1)$. Then, the i-th probability density expression and sampling process of the ternary Gaussian distribution $N(\mu_i, \Sigma_i)$ can be expressed as follows:

$$N(x_i|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp[-\frac{1}{2}(x - \mu_i)^T \Sigma^{-1}(x - \mu_i))], \tag{11}$$

$$Q_i \sim T \times N(x_i|\mu_i, \Sigma_i) \tag{12}$$

where x_i is the sampled value from $N(\mu_i, \Sigma_i)$ and Q_i denotes the coordinates of the three-dimensional trajectory points obtained by sampling. Note that when we sample the ternary Gaussian distribution, the sampled value is multiplied by T. The mean value obtained by sampling the multivariate Gaussian distribution is between 0 and 1, and the range of the horizontal and vertical coordinates of our trajectory points is between 0 and 28. Therefore, the sampling of the horizontal and vertical coordinates must be multiplied by a value T, where T is equal to 28, to obtain a better coordinate value. The third dimension of Q_i,z_i represents the height of the brush of the robot system from the robot writing plane. Empirically, z_i is set to between 0 and 5. In summary, for the abscissa and ordinate, T is set to 28, and for the height coordinate, T is set to 5.

Then, the currently obtained coordinates Q_i are sent into the robot system. After the inverse kinematics calculation, the robot system converts the coordinates into its motion joint values, which are used by the robot to complete the writing of this step, and obtains the image of the i-th stroke. Then, input the i-th stroke image into the GRU's $(i + 1)$-th unit, and the hidden layer state h_i of the i-th GRU unit is used as the initial hidden state of the $(i+1)$-th GRU unit. Next, the state passes through the fully connected layer, activation function layer, and robot system, and then obtain the $(i+1)$-th stroke image. This cycle repeats until a complete Numeral 2 is written.

We use the GRU network for two reasons: 1) GRU networks are more robust than recurrent neural networks; since GRU

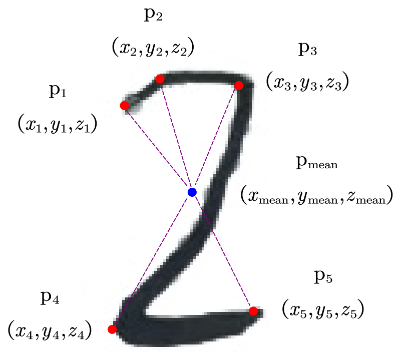


Fig. 5. Sequential points of Numeral 2

networks allow means of multivariate Gaussian distributions to obtain more generalized values to bring better diversity writing performance. 2) The total number of parameters of a GRU network is smaller than that of an LSTM model, resulting in fewer computational costs in training. We take use of the cyclicity and excellent network performance of the GRU network to imitate each stroke written by the robot as a GRU recurrent layer, which realizes the sequential writing ability of the robot. In addition, the Competitive Swarm Optimizer evolutionary computing method is used to solve the problem that the robot cannot optimize its trajectory generator network by using error back-propagation methods, so that the robot can learn well and autonomously.

D. Evaluation System

The writing quality of the numerals is evaluated by combing three similarities, including the shape similarity, trajectory sequence similarity, and central position similarity, which jointly represent the overall loss. The shape loss is obtained by calculating the similarity between the input numerals and the robot's writing results. The trajectory sequence loss is represented by the similarity between the trajectory sequence label vector and the generated trajectory sequence by the generator. The central position loss is defined as the gap between the central positions of the generated writing result and a training sample.

The calculation method of the central position loss is shown in Fig. 5. This figure takes Numeral 2 as an example; the trajectory sequence (five points in total) of Numeral 2 is represented by: $\vec{Q} = [P_1, P_2, P_3, P_4, P_5]$, where the i-th point coordinate is denoted as $P_i = (x_i, y_i, z_i)$. The coordinate of the central point of the trajectory sequence $P_m = (x_m, y_m, z_m)$ is defined by: $x_m = \frac{\sum_i^N x_i}{N}$, $y_m = \frac{\sum_i^N y_i}{N}$ and $z_m = \frac{\sum_i^N z_i}{N}$.

Denote the central position sequence vector of $\vec{Q}$'s as $\vec{M}$, which is defined as: $\vec{M} = [(x_1 - x_m, y_1 - y_m, z_1 - z_m),(x_2 - x_m, y_2 - y_m, z_2 - z_m),(x_3 - x_m, y_3 - y_m, z_3 - z_m),(x_4 - x_m, y_4 - y_m, z_4 - z_m),(x_5 - x_m, y_5 - y_m, z_5 - z_m)]$. In this work, the cosine similarity, $\psi(\cdot)$, is employed to represent the shape, trajectory sequence, and central position similarities. The cosine similarity is calculated as follows:

$$\psi(p, \hat{p}) = \frac{p \cdot \hat{p}}{\|p\| \cdot \|\hat{p}| + \eta}, \tag{13}$$

where p and $\hat{p}$ are two vectors that need to be calculated; and η is a positive number close to 0.

Based on the cosine similarity, the fitness function expressing the quality of the writing result is defined as:

$$f = -\alpha \cdot \psi(P, P') - \beta \cdot \psi(Q, Q') + \gamma \cdot \psi(M, M'), \tag{14}$$

where f denotes the fitness value; $\psi(\cdot)$ is the cosine similarity function; P and P' are vectors transformed from a numeral sample and the corresponding robotic writing result, respectively; Q and Q' are vectors representing the ground truth writing sequence and the trajectory sequence generated by the proposed GRU-based generator; M and M' the vectors obtained by centralizing Q and Q', respectively; and α, β, and γ are weights to represent the significance of the three types of losses. In the experiment, α, β and γ are set to 0.2, 0.5, and 0.3, respectively. Note that the number of each similarity's target samples is set to five in the valuation systems, i.e., a human engineer labels only five samples for each numeral's writing sequence.

E. Optimization module

In order to optimize the trajectory generator, the Competitive Swarm Optimizer (CSO) algorithm is employed [30]. The CSO algorithm optimizing the GRU network process is specified as follows: **Step 1**: Initialize the particle dimension in the particle swarm of the CSO algorithm, (i.e., total number of GRU networks parameters). Here, the number of particles in the initialized particle swarm is n, each of which represents one GRU network. **Step 2**: Determine whether the CSO algorithm has reached the termination condition, which is the iteration amount of the CSO algorithm. **Step 3**: If the termination condition has not been reached, according to the fitness value of each particle (i.e., the output of the loss function in this paper), the evolution and selection of particles are used to obtain the next generation of particle swarms (i.e., the new parameters of the GRU network). After the last iteration is terminated, the optimal particle value is obtained as the parameter value of the final GRU network. **Step 4**: Update the parameters of the n GRU network with the values of the new generation particle swarms and return to Step 3.

Writing trajectory points obtained from the GRU network are sent to the robot system and are converted into the robotic joint coordinates, thereby obtaining the final robot writing results. This process must involve the robot's manipulator hardware, which is unknown to the GRU network. Therefore, final written errors cannot be optimized by the back-propagation algorithm for the GRU network. Alternatively, we use the CSO evolutionary calculation method to update the parameters of the GRU network according to the CSO's fitness values defined in Eq. (14). The adaptation of CSO with key parameter setup is discussed in the rest of this section.

Swarm: The swarm consists of N_t particles, where t is the index of the current iteration. Each particle $P(t)$ contains the position and velocity information of the writing point, which are collectively represented as vectors $\vec{x(t)}$ and $\vec{v(t)}$, $(\vec{x(t)}, \vec{v(t)} \in R^D)$; the dimension D of the vector is equal to the number of parameters used in the GRU-based generator. The value of each dimension of the D dimension vector is bounded:

$$x_j^{min} < x_{i,j}(t) < x_j^{max}; \quad j = 1, 2, \cdots, N_t, \tag{15}$$

where x_j^{min} and x_j^{max} denote the minimum and maximum values of the j-th dimension of each position vector $\vec{x(t)}$ in the t-th generation population. The minimum and maximum values can be determined by the j-th parameter in the decoder of the GRU-based generator.

Initialization of Swarm: After the boundaries of particle position and velocity are established, the population is randomly initialized:

$$\begin{cases} x_{i,j}(0) & = x_j^{min} + rand\_uniform(0,1)(x_j^{max} - x_j^{min}) \\ v_{i,j}(0) & = 0 \end{cases} \tag{16}$$

where $rand\_uniform(0,1)$ is a randomly generated number between 0 and 1. $x_{i,j}(0)$ and $v_{i,j}(0)$ represent the spatial position and speed of initialization, respectively. In this work, the velocity values are initialized to 0.

Particle Updating: Within the population of N_t particles, every particle is paired with another particle, so as to obtain N_t/2 pairs. The winner and loser particles can be determined by comparing the fitness values of each pair. In this work, the particle with a lower objective value is the winner. Suppose P_1(t) and P_2(t) are a pair of competitors in the t-iteration. The competition process for $P_{winner}(t)$ and $P_{loser}(t)$ are defined as:

$$P_{winner}(t) = \begin{cases} P_1(t), & \text{if } g(P_1(t)) \leq g(P_2(t)) \\ P_2(t), & \text{otherwise} \end{cases} \tag{17}$$

where $g(\cdot)$ is the fitness function as discussed in Section III-D, and the optimization algorithm aims to minimize this function to generate an optimal set of parameters of the trajectory generator.

From this, the loser position vector $\vec{x_{loser}}(t)$ updates its position information by learning from the winner based on the following principle:

$$\begin{aligned} \vec{v_{loser}(t+1)} &= \vec{\beta_1}(t)\vec{v_{loser}}(t) + \vec{\beta_2(t)}(\vec{x_{winner}}(t) - \vec{x_{loser}}(t)), \\ &+ \varphi\beta_3\vec{(t)}(\chi\vec{(t)} - \vec{x_{loser}}(t)) \\ \vec{x_{loser}(t+1)} &= \vec{x_{loser}}(t) + \vec{v_{loser}(t+1)}, \end{aligned} \tag{18}$$

where $\vec{x_{loser}}(t)$ and $\vec{v_{loser}}(t)$ respectively denote the position and velocity vector of the loser in $P_1(t)$ and $P_2(t)$; $\vec{\beta_1(t)}$, $\vec{\beta_2(t)}$, and $\vec{\beta_3(t)}$ are three randomly generated vectors in the t-th competition; $\chi\vec{(t)}$ is the average position of all particles in the t-th iteration; and φ is the parameter that controls the effect of $\chi\vec{(t)}$.

After all the losers updated their positions with the support of the winners, the CSO then moves to the next generation of competitive evolution. This evolutionary process is repeated until the termination condition is met, that is either an optimal fitness value or a predetermined maximum algebra g is achieved. The training procedure is presented as a pseudo-code listed in Algorithm 1.

Algorithm 1 Training Procedure Pseudo-code

```
Require:Population size $N_t$, maximum algebra $g$, and CSO algorithm control average position parameters $\varphi$. Number $k$ of GRU units in GRU network. An blank vector $p_0$, Gaussian distribution noise $h_0$ and covariance $\Sigma$ determined by VAE encoder output.
1: Use Eq. (16) to initialize $N_t$ GRU network weights;
2: for j in 0:g do
3:     Input $p_0$, $h_0$ into $N_t$ GRU;
4:     for i in 0:k do
5:         Use Eqs. (9), (10), (11) and (12) to sample a trajectory point $Q_i$;
6:         Robot writes the trajectory, which is captured as a input image for the next cycle;
7:     end for
8:     $N_t/2$ pairs of GRU particles compete. Use Eq. (17) to decide which one is a loser. Use Eq. (18) to update the loser parameters.Then the loser and winner particles go to the next iteration.
9: end for
Output: At the end of the iteration, assign the value of the best particle to the parameters of the GRU network, which is GRU network optimal weights.
```

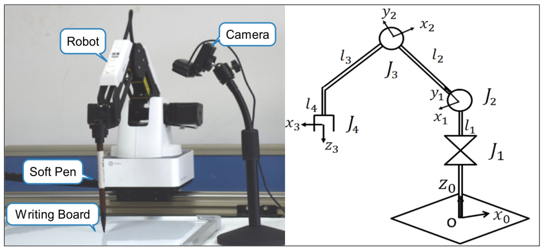


Fig. 6. The hardware of the proposed framework and the structure of the robot

F. Robotic system

The calligraphy robot system used in this work is shown in the left side of Fig. 6, which consists of a three degrees-of-freedom robot arm and a camera mounted on a bracket. A brush pen is mounted at the end actuator of the robot arm. The whiteboard placed under the pen is the writing area of the robot. When the robot completes the writing of one numeral, the pen returns to its initial position. Then, the camera captures the writing result as an image.

The mechanical setup of the robot is shown on the right side of Fig. 6. In the figure, l represents the mechanical link; x, y, and z are the coordinate axis of the robot; and j is the steering gear of the robotic arm. The robot converts the three-dimensional coordinate point Q_i into three joint values θ_i = ($\theta_i^{(1)}$, $\theta_i^{(2)}$, $\theta_i^{(3)}$) of the robot by inverse kinematics calculation, using the approach as specified in the work of [14].

The robot continues to write strokes by following the trajectory points generated in the series of GRU units; in other words, the coordinate point Q_{i-1} of the previous GRU unit is connected to the coordinate point Q_i led by the current GRU unit. For the very first GRU unit, there is no robotic action but only the coordinate is generated. Once the numeral is written, the camera system of the robot then captures the written image, binarizes the result, and cuts the result to 28×28 pixels. The process is expressed as $W(\cdot)$. This image is used as the input p_i for the next GRU unit. The process of robot writing is expressed as:

$$p_i = \begin{cases} W(IK(Q_i)), i = 0 \\ W(IK(Q_{i-1}), IK(Q_i)), i > 0 \end{cases} \tag{19}$$

where the transformation process of inverse kinematic is defined as $IK(\cdot)$. Finally, the output of the generated model is expressed as: $p_k = G(p_0, h_0)$, where $G(\cdot)$ represents the trajectory generator.

In particular, recall the writing training process: First, a trajectory sequence is obtained through the GRU trajectory network and the robot writes the sequence; then the CSO algorithm optimizes the GRU network using sampled and evaluated results. Thus, implementing this process leads to the time complexity of O(k*tg), passing k GRU units and a pair of particles' time in the CSO O(1). Let the number of iterations and particle size be n and m respectively, then the total time complexity of the proposed algorithm is: O(nm*k*tg).

IV. Experimentation

To validate and evaluate the proposed system, the proposed system was physically implemented and the writing system was optimized through two training processes: (1) using the traditional VAE training method (gradient back-propagation) to train the encoder part of VAE; and (2) using the CSO algorithm to optimize the trajectory generator. The experimentation was performed on ten Arabic numerals. A comparative study using the proposed system and the state-of-the-art GAN-based [13] and reinforcement learning-based (RL-based) robotic writing methods [4] is also reported in this section.

A. Datasets and Settings

The "MNIST" data set was utilized in the experiments, which includes 60,000 handwritten images of ten numerals from "0" to "9". Each image has been pre-processed with the same size of 28×28 grey-scale pixels. However, in the experiment, only two hundred samples for each numeral were randomly selected as our experimental training samples. Because our experiment only requires a small number of samples to complete the training, selecting 200 samples helps improve the training efficiency. Several training data utilized in this work are shown in Fig. 7.

The population size N_t, maximum algebra g, and CSO algorithm control average position parameters φ, were set to 300, 2,400, and 0.25, respectively. In addition, α, β and γ in Eq. 14 are set to 0.2, 0.5, and 0.3, respectively. The loop number k of the GRU-based trajectory generator is set according to

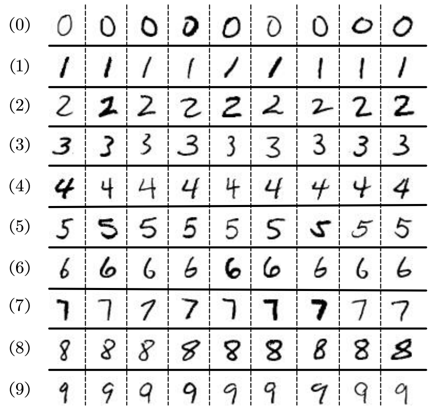


Fig. 7. Illustrative training samples of ten numerals from "0" to "9" used in the experiment, each row shows one type of a number with various variants.

the writing complexity of each numeral, and the range of k is set from 3 to 8. Among them, the k values of numerals 0 to 9 are respectively set to: $[7, 3, 5, 7, 4, 8, 7, 3, 7, 7]$. The parameters in the paper are the best results obtained by tuning. The corresponding tuning process is not shown due to the length of the article.

B. Writing Results

The final result of the ten Arabic numerals is shown in Fig. 8, with each numeral written 9 times. The written results of each numeral are not identical to each other. Most of the differences are subtle, but some are significant, such as the result of Numeral 2. This phenomenon demonstrates that the proposed system did not simply replicate a pre-defined template, but used a Gaussian distribution noise to generate diverse outputs to better simulate human writing results.

To reflect the efficiency of the proposed method, we have now carried out experimental investigations into the training and test writing time on the number '5' (implemented on a Linux workstation with Intel(R) Core(TM) i7-6850K, one 1080ti and PyTorch 1.10). The training time is 4.1 hours, but the test writing time is the sum of the time spent for acquiring the trajectory sequence (0.57 seconds) and the writing time of the robot (3.0 seconds), which merely costs 3.57 seconds in total.

The writing process of the ten numbers are demonstrated in Fig. 9. Numerals with complex shapes were set to have more GRU units, such as Numerals 5, 8, and 9. In this figure, the writing sequences of all numerals are in line with human writing habits. This presents the most significant contribution of this paper. This function was achieved by using a sequence label with sequential points. In addition, the diversity of the writing results shown in Fig. 8 proved that the sequential points did not limit the system to generate various outputs.

The dynamic properties of the algorithm are mainly reflected in the robot's writing processes within the GRU units. Take three writing processes of Numeral '2' as an example, as shown in Fig. 10. Each of these writing processes starts with a different position, leading to a different position of

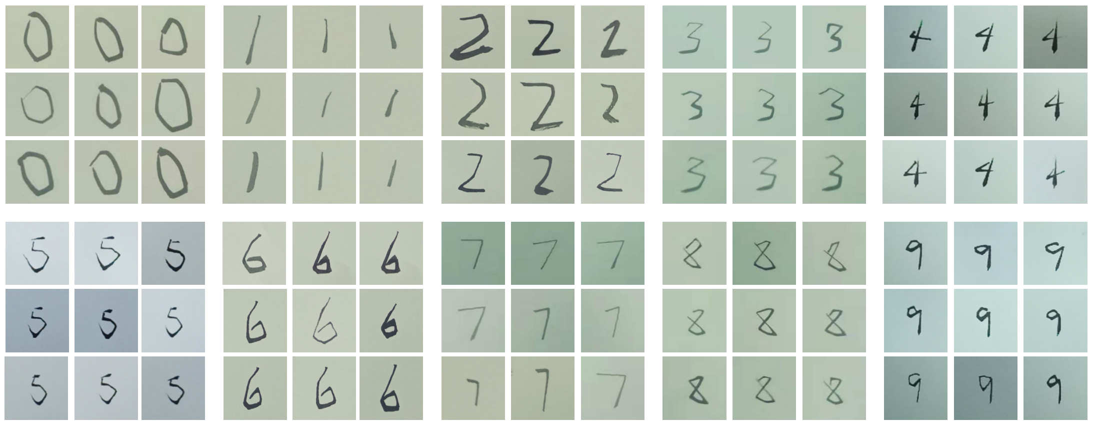


Fig. 8. The final writing results of the ten numerals.

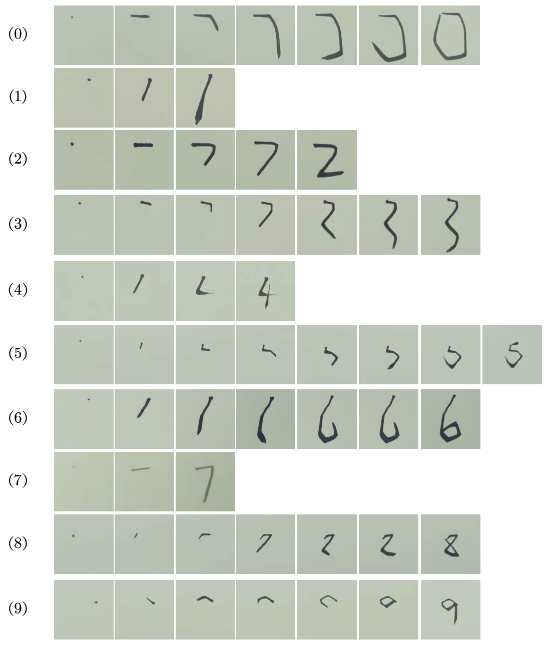


Fig. 9. The writing process of ten numerals.

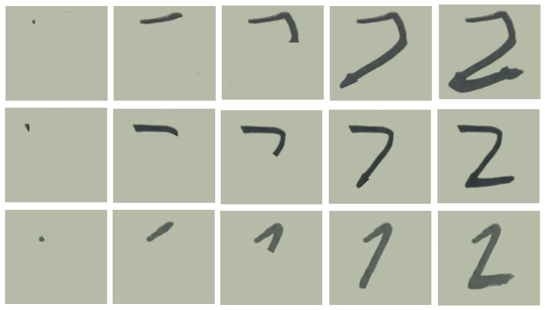


Fig. 10. Three step-by-step writing processes of Numeral ’2’.

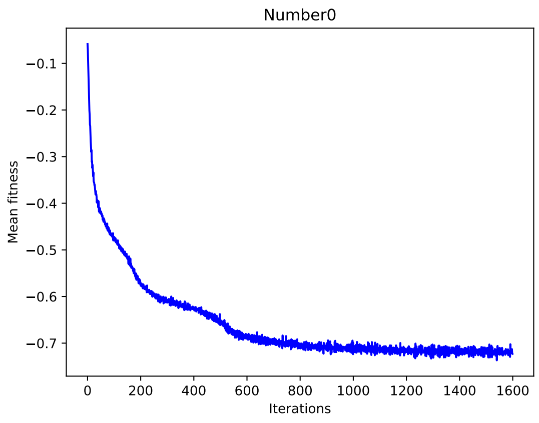


Fig. 11. Mean particle fitness value vs. number of iterations for numeral ’0’.

the subsequent strokes, so that the final writing outcomes are diverse. The dynamic properties inherent within the writing processes are enabled because of the fact that in each GRU unit, the input is a captured image including written trajectories so far, with the hidden layer state of the output taken as the mean of the ternary Gaussian distribution of the input, and that the trajectory points are sampled from such a distribution, calculated by the central loss function.

As with common practice in the literature, whether the optimal state has been reached is determined with regard to the curve of the mean fitness of the population particles changing against the number of iterations during the optimization process of the CSO. For example, from the change curve of Numeral ‘0’ as shown in Fig. 11, it can be observed that a stable state is reached after 1,200 iterations, indicating that the optimal GRU network parameters cannot be meaningfully optimized any further.

C. Ablation Experiment

In order to verify the effectiveness of the proposed central position similarity, a new fitness function is defined as:

$$F' = -\lambda \cdot \psi(P, P') - (1-\lambda) \cdot \psi(Q, Q'), \tag{20}$$

where λ is a parameter to reconcile the balance between image similarity and sequence similarity. λ is empirically set to 0.4 that is the best result obtained by tuning. In contrast to Eq. (14), Eq. (20) removes the central position similarity loss. Thus, we speculate that the diversity of writing results will be affected.

TABLE I
DIVERSITY ASSESSMENT. VARIANCE AND ENTROPY ARE USED AS TWO INDICATORS TO EVALUATE THE DIVERSITY OF WRITING RESULTS. LARGER VALUES INDICATE BETTER DIVERSITY. EQS. (14) AND (20) RESPECTIVELY INDICATE THE RESULTS WITH AND WITHOUT THE CENTER LOSS.

Numerals	Variance		Entropy	
	Eq.(20)	Eq.(14)	Eq.(20)	Eq.(14)
0	0.201	**0.294**	0.565	**0.615**
1	0.172	**0.221**	0.21	**0.309**
2	0.232	**0.305**	0.397	**0.493**
3	0.224	**0.282**	0.413	**0.543**
4	0.197	**0.233**	0.471	**0.551**
5	0.213	**0.246**	0.487	**0.645**
6	0.252	**0.283**	0.37	**0.502**
7	0.198	**0.242**	0.298	**0.408**
8	0.23	**0.277**	0.498	**0.631**
9	0.216	**0.243**	0.365	**0.527**

We measure the diversity of a data set by calculating the variance and entropy [39], [40] of the image data set. Table I shows the variances and entropies of the numeral training data set, the writing result set produced by the ablation experiment, and the writing result set of the methods in this chapter. The greater the variance and entropy of the data set, the greater the internal differences of the data set, and the higher the diversity of the data set. In Table I, the variance and entropy of each numeral set with the CenterLoss are higher than that of the corresponding numeral set without the CenterLoss in the ablation experiment. Note that the larger variance and entropy of our proposed method are shown in bold in Table I. Therefore, our method is superior to the method in the ablation experiment in both visual and quantitative analysis. These results prove the effectiveness of our method to improve the diversity of writing results.

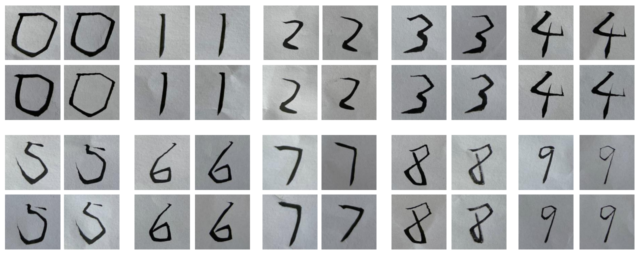
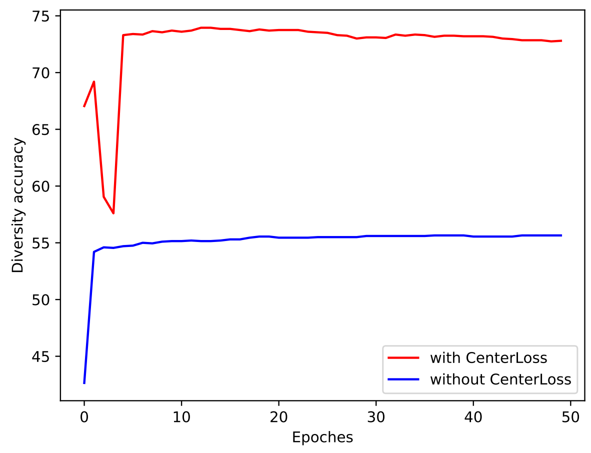


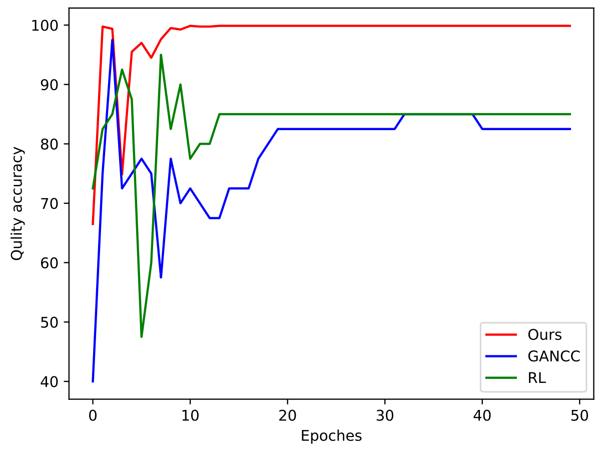
Fig. 12. The writing results of ablation experiment.

The final writing results of the ten Arabic numerals are shown in Fig. 12. Each numeral has four writing results. Compared with the experimental results in Fig. 8, we cannot clearly identify the differences amongst each numeral's results; the writing of each numeral tends there is no diversity shown to one fixed pattern.

Moreover, to exhibit the diverse advantages of our method, the GANtrain [41] method was added in a new ablation experiment. In the GANtrain method, a VGG16 [42] is adopted to predict each input image's class. Therefore, in the new ablation, the writing results were used as the train set of the VGG16 and training samples were used as the test set. Thus, higher predict accuracies of the VGG16 represented higher recall rates, i.e., better writing diversity. Therefore, Fig. 13-(a) verifies the predicted accuracies with the CenterLoss are much higher than those without the CenterLoss.



(a)



(b)

Fig. 13. The GANtrain and GANtest compare results. (a) represents diversity curves and (b) represents a quality comparison curve of multiple writing methods.

D. Comparison

To verify the advantages of the proposed method compared to other methods, a GAN-based robotic writing system [13] and a reinforcement learning-based robotic writing system [4] is used for comparison. These methods did not support the sequential writing feature. Therefore, we used the Fréchet Inception Distance (FID) [37], MAE, PSNR, SSIM [38], and PerLoss [39] to compare the writing qualities of these studies. Here, the PerLoss is the L_1 loss of the middle layer feature map by using a VGG16 classifier. Note that the VGG classifier was pretrained by a dataset with ten numerals. The FID, MAE, and PerLoss with smaller values indicate that the written result is closer to the training data set, while PSNR and SSIM are on the contrary. Table II shows the results between the results generated by the three methods.

The best values are highlighted in bold in this table. Our proposed method archived the best performance for all

TABLE II
QUALITY ASSESSMENT. FRÉCHET INCEPTION DISTANCE (FID) [37], MAE, PSNR, SSIM [38], AND PERLOSS [39] ARE USED TO COMPARE THE WRITING QUALITIES OF OURS, GAN-BASED [13] AND RL-BASED METHODS [4].

Numerals	FID			MAE			PSNR			SSIM			PerLoss		
	GANCC	RL	Ours	GANCC	RL	Ours	GANCC	RL	Ours	GANCC	RL	Ours	GANCC	RL	Ours
0	63.96	N/A	**52.98**	0.1669	N/A	**0.1314**	8.6415	N/A	**9.8041**	0.4751	N/A	**0.5562**	0.3426	N/A	**0.2874**
1	54.62	50.04	**22.94**	0.0817	0.1012	**0.0552**	11.9416	10.4787	**13.9254**	0.4346	0.2692	**0.5906**	0.3539	0.4652	**0.3029**
2	37.28	53.51	**28.33**	0.1813	0.1982	**0.1157**	8.0771	7.5871	**10.3467**	0.306	0.2314	**0.543**	0.4383	0.5212	**0.3109**
3	32	55.52	**27.01**	0.1835	0.2039	**0.11**	8.0293	7.4583	**10.6377**	0.2814	0.1388	**0.5514**	0.4685	0.5319	**0.304**
4	**22.85**	N/A	57.82	0.1544	N/A	**0.1264**	8.7693	N/A	**9.8044**	0.3372	N/A	**0.3902**	0.4311	N/A	**0.344**
5	**35.2**	N/A	73.07	0.1956	N/A	**0.15**	7.8008	N/A	**8.9918**	0.2807	N/A	**0.4389**	0.4483	N/A	**0.3554**
6	42.76	N/A	**37.98**	0.134	N/A	**0.0997**	9.5891	N/A	**11.1094**	0.4099	N/A	**0.54**	0.3719	N/A	**0.2803**
7	49.78	43.51	**34.67**	0.113	0.1499	**0.0871**	10.5046	8.9214	**11.7104**	0.4547	0.2348	**0.5499**	0.3285	0.4889	**0.285**
8	49.99	N/A	**46.95**	0.1956	N/A	**0.1301**	7.6917	N/A	**9.8308**	0.3145	N/A	**0.559**	0.4667	N/A	**0.2927**
9	45.54	N/A	**41.68**	0.2179	N/A	**0.0946**	7.0867	N/A	**11.3901**	0.2011	N/A	**0.5568**	0.4912	N/A	**0.2652**

numerals in MAE, PSNR, SSIM, and PerLoss. It is also clear that except for Numerals 4 and 5 in FID, the proposed system achieved better performance on the rest eight numerals. Since Numerals 4 and 5 are composed of two trajectories, more training iterations might be required for the proposed system. It is still difficult for the RL-based method to write complex numerals, such as 0, 4, 5, 6, 8, 9 in Table II. Therefore, the comparison proved that the proposed method not only can achieve the sequential writing of numerals, but also can have a better writing quality than the GAN-based and RL-based methods.

To qualitatively compare the proposed method with similar studies, the quality of writing is evaluated by the GANtest [41]. In contrast to the GANtrain, the writing results of the three methods were used as the test set of a VGG16, and the training samples were used as the training set. Thus, higher predict accuracies of the VGG16 represented higher accuracy rates. Therefore, Fig. 13-(b) verifies the prediction accuracies of our method were the best in the comparison.

V. Conclusion

This paper presented a new learning system for robotic writing. This method used GRU units to form a trajectory generator of numerals. With the assistance of a sequence label, the system can generate accurate trajectories, which were close to human's writing habits. In addition, a pre-training operation was employed to explore the input distributions to produce system inputs, and the CSO heuristic search algorithm was used to optimize the trajectory generator. The system was evaluated on Arabic numerals using the MNIST data set. Comprehensive experiments demonstrated the comparative performance of the proposed approach in reference to the state-of-the-art methods, especially with a writing sequence in alignment with human writing habits.

Whilst this work has addressed a number of important issues for robotic character writing, several difficulties have been encountered when we carried out the investigation. In particular, the following two have drawn our specific attentions to: (1) For writing different numbers, the number of cycles of the GRU unit cannot be automatically determined but has to be empirically or manually specified. A mechanism for setting this parameter through self-adaptation during the learning process remains active research. (2) The training time for numeric writing is relatively long (although this does not affect the very efficient run-time performance). Alternative meta-learning methods [43], [44] are worth of being considered as a piece of interesting future work.

Acknowledgments

The authors are very grateful to the anonymous reviewers for their constructive comments which have helped significantly in revising this work.

References

[1] F. Chao, Y. Huang, C.-M. Lin, L. Yang, H. Hu, and C. Zhou, "Use of automatic Chinese character decomposition and human gestures for Chinese calligraphy robots," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 1, pp. 47–58, 2018.

[2] D.-t. Liang, D. Liang, S.-m. Xing, P. Li, and X.-c. Wu, "A robot calligraphy writing method based on style transferring algorithm and similarity evaluation," *Intelligent Service Robotics*, pp. 1–10, 2019.

[3] X. Gao, C. Zhou, F. Chao, L. Yang, C.-M. Lin, and C. Shang, "A robotic writing framework–learning human aesthetic preferences via human-machine interactions," *IEEE Access*, vol. 7, pp. 144 043–144 053, 2019.

[4] R. Wu, C. Zhou, F. Chao, L. Yang, C.-M. Lin, and C. Shang, "Integration of an actor-critic model and generative adversarial networks for a Chinese calligraphy robot," *Neurocomputing*, vol. 388, pp. 12 – 23, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231220300886

[5] P. Schaldenbrand and J. Oh, "Content masked loss: Human-like brush stroke planning in a reinforcement learning painting agent," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* AAAI Press, 2021, pp. 505–512.

[6] H. Ma, Q. Zhou, H. Li, and R. Lu, "Adaptive prescribed performance control of a flexible-joint robotic manipulator with dynamic uncertainties," *IEEE Transactions on Cybernetics*, 2021.

[7] X. Zhang, Y. Li, Z. Zhang, K. Konno, and S. Hu, "Intelligent Chinese calligraphy beautification from handwritten characters for robotic writing," *The Visual Computer*, vol. 35, no. 6-8, pp. 1193–1205, 2019.

[8] F. Chao, Y. Huang, X. Zhang, C. Shang, L. Yang, C. Zhou, H. Hu, and C. M. Lin, "A robot calligraphy system: From simple to complex writing by human gestures," *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 1–14, 2017.

[9] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3758–3765.

[10] H. Chen, "Robotic manipulation with reinforcement learning, state representation learning, and imitation learning (student abstract)," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February*

2-9, 2021. AAAI Press, 2021, pp. 15 769–15 770. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17881

[11] M. Sharifi, A. Zakerimanesh, J. K. Mehr, A. Torabi, V. K. Mushahwar, and M. Tavakoli, "Impedance variation and learning strategies in human-robot interaction," *IEEE Transactions on Cybernetics*, 2021.

[12] J. Zeng, Q. Chen, Y. Liu, M. Wang, and Y. Yao, "Strokegan: Reducing mode collapse in chinese font generation via stroke encoding," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 3270–3277.

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

[14] R. Wu, C. Zhou, F. Chao, L. Yang, C.-M. Lin, and C. Shang, "Ganc-robot: Generative adversarial nets based Chinese calligraphy robot," *Information Sciences*, vol. 516, pp. 474 – 490, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025519312071

[15] T. Wang, W. Q. Toh, H. Zhang, X. Sui, S. Li, Y. Liu, and W. Jing, "Robocodraw: Robotic avatar drawing with gan-based style transfer and time-efficient path optimization," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 10 402–10 409. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/6609

[16] Y. Gao and J. Wu, "Gan-based unpaired chinese character image translation via skeleton transformation and stroke rendering," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 646–653. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5405

[17] Q. Li, F. Chao, X. Gao, L. Yang, C.-M. Lin, C. Shang, and C. Zhou, "A robotic chinese stroke generation model based on competitive swarm optimizer," in *Advances in Computational Intelligence Systems*, Z. Ju, L. Yang, C. Yang, A. Gegov, and D. Zhou, Eds. Cham: Springer International Publishing, 2020, pp. 92–103.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] D. Kong and F. Wu, "Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 2341–2347. [Online]. Available: https://doi.org/10.24963/ijcai.2018/324

[20] J. Wang, T. Sun, B. Liu, Y. Cao, and H. Zhu, "Clvsa: A convolutional lstm based variational sequence-to-sequence model with attention for predicting trends of financial markets," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 3705–3711. [Online]. Available: https://doi.org/10.24963/ijcai.2019/514

[21] D. Li, R. Rzepka, M. Ptaszynski, and K. Araki, "Emoji-aware attention-based bi-directional gru network model for chinese sentiment analysis." in *LaCATODA/BtG@ IJCAI*, 2019, pp. 11–18.

[22] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, and Y. Xia, "Trajectory planning for hypersonic reentry vehicle satisfying deterministic and probabilistic constraints," *Acta Astronautica*, vol. 177, pp. 30–38, 2020.

[23] C. Runqi, A. Tsourdos, A. Savvaris, C. Senchun, and X. Yuanqing, "High-fidelity trajectory optimization for aeroassisted vehicles using variable order pseudospectral method," *Chinese Journal of Aeronautics*, vol. 34, no. 1, pp. 237–251, 2021.

[24] R. Chai, A. Tsourdos, A. Savvaris, S. Wang, Y. Xia, and S. Chai, "Fast generation of chance-constrained flight trajectory for unmanned vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 2, pp. 1028–1045, 2020.

[25] R. Chai, A. Tsourdos, H. Gao, Y. Xia, and S. Chai, "Dual-loop tube-based robust model predictive attitude tracking control for spacecraft with system constraints and additive disturbances," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 4, pp. 4022–4033, 2021.

[26] X. Gao, C. Zhou, F. Chao, L. Yang, C.-M. Lin, T. Xu, C. Shang, and Q. Shen, "A data-driven robotic chinese calligraphy system using convolutional auto-encoder and differential evolution," *Knowledge-Based Systems*, vol. 182, p. 104802, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705119302771

[27] D. Xie, X. Zhang, Q. Gao, J. Han, S. Xiao, and X. Gao, "Multiview clustering by joint latent representation and similarity learning," *IEEE transactions on cybernetics*, vol. 50, no. 11, pp. 4848–4854, 2019.

[28] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[29] Z. Li, P. Wang, H. Lu, and J. Cheng, "Reading selectively via binary input gated recurrent unit," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 5074–5080. [Online]. Available: https://doi.org/10.24963/ijcai.2019/705

[30] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2014.

[31] W. Zheng and G. Chen, "An accurate gru-based power time-series prediction approach with selective state updating and stochastic optimization," *IEEE Transactions on Cybernetics*, 2021.

[32] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3. IEEE, 1999, pp. 1945–1950.

[33] Q. Yang, W.-N. Chen, T. Gu, H. Jin, W. Mao, and J. Zhang, "An adaptive stochastic dominant learning swarm optimizer for high-dimensional optimization," *IEEE Transactions on Cybernetics*, 2020.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[35] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-user linking via variational autoencoder," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3212–3218. [Online]. Available: https://doi.org/10.24963/ijcai.2018/446

[36] D. Jin, B. Li, P. Jiao, D. He, and W. Zhang, "Network-specific variational auto-encoder for embedding in attribute networks," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 2663–2669. [Online]. Available: https://doi.org/10.24963/ijcai.2019/370

[37] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[38] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 2366–2369.

[39] H. Yakura, Y. Koyama, and M. Goto, "Tool-and domain-agnostic parameterization of style transfer effects leveraging pretrained perceptual metrics," *arXiv preprint arXiv:2105.09207*, 2021.

[40] Q. Zhang, "Learning nash equilibria in zero-sum stochastic games via entropy-regularized policy approximation," Ph.D. dissertation, Georgia Institute of Technology, 2020.

[41] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my GAN?" in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 213–229.

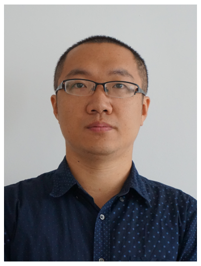
[42] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2016, pp. 1–6.

[43] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," 2020.

[44] J. Liu, F. Chao, L. Yang, C.-M. Lin, C. Shang, and Q. Shen, "Decoder choice network for metalearning," *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

Quanfeng Li received the B.Eng. degree from Guangdong University of Technology, Guangzhou, China, in 2016 and the M.Eng. degree from Xiamen University, Xiamen, China, in 2021. He works as research assistance with the School of Informatics, Xiamen University. His research interests include machine learning, robot calligraphy, and recommended system.

Zhihua Guo received the B.Sc. degree in Computer Science from Taiyuan University of Technology in 2019. He is working towards his M.Eng. Degree in Artificial Intelligence from the School of Informatics, Xiamen University, China. His research interests include robotic calligraphy and meta learning algorithms.



Fei Chao (M'11) received the B.Sc. degree in Mechanical Engineering from the Fuzhou University, China, and the M.Sc. Degree with distinction in Computer Science from the University of Wales, Aberystwyth, U.K., in 2004 and 2005, respectively, and the Ph.D. degree in robotics from the Aberystwyth University, Wales, U.K. in 2009. He is currently an Associate Professor with the School of Informatics, Xiamen University, China; andand he is an adjunct research fellow with the Department of Computer Science, Aberystwyth University. Dr Chao has published more than 100 peer-reviewed journal and conference papers. His research interests include machine learning algorithms and robotics.

Xiang Chang received his BEng. degree in Cognitive Science at the Department of Artificial Intelligence, Xiamen University in 2019. Currently, he is a senior PhD student at the Department of Computer Science, Aberystwyth University. His research interests include deep reinforcement learning based robotic motion planning.

Longzhi Yang (M'12-SM'17) is currently a Programme Leader and a Reader with Northumbria University, Newcastle upon Tyne, U.K. His research interests include computational intelligence, machine learning, big data, computer vision, intelligent control systems, and the application of such techniques in real-world uncertain environments. He is the Founding Chair of the IEEE Special Interest Group on Big Data for Cyber Security and Privacy. Dr. Yang received the Best Student Paper Award from the 2010 IEEE International Conference on Fuzzy Systems.

Chih-Min Lin (M'87-SM'99-F'10) was born in Taiwan, in 1959. He received the B.S. and M.S. degrees from Department of Control Engineering and the Ph.D. degree from Institute of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 1981, 1983 and 1986, respectively. He is currently a Chair Professor and the Vice President of Yuan Ze University, Chung-Li, Taiwan. His current research interests include fuzzy neural network, cerebellar model articulation controller, intelligent control systems and signal processing. He has published more than 170 journal papers. He also serves as an Associate Editor of IEEE Transactions on Cybernetics and IEEE Transactions on Fuzzy Systems. He is an IEEE Fellow.

Changjing Shang received a Ph.D. in computing and electrical engineering from Heriot-Watt University, UK. She is a University Research Fellow with the Department of Computer Science, Institute of Mathematics, Physics and Computer Science at Aberystwyth University, UK. Prior to joining Aberystwyth, she worked for Heriot-Watt, Loughborough and Glasgow Universities. Her research interests include pattern recognition, data mining and analysis, space robotics, and image modelling and classification.

Qiang Shen received the Ph.D. in Computing and Electrical Engineering (1990) from Heriot-Watt University, Edinburgh, U.K., and the D.Sc. in Computational Intelligence (2013) from Aberystwyth University, Aberystwyth, U.K. He holds the Established Chair in Computer Science and is the Pro Vice-Chancellor: Faculty of Business and Physical Sciences, Aberystwyth University. His research interests include computational intelligence and its application in robotics. He has authored two research monographs and over 400 peer-reviewed papers, including one receiving an Outstanding Transactions Paper Award from the IEEE.