Integration of an Actor-Critic Model and Generative Adversarial Networks for a Chinese Calligraphy Robot

Ruiqi Wu[a], Changle Zhou[a], Fei Chao[a,b,*], Longzhi Yang[c], Chih-Min Lin[d], Changjing Shang[b]

[a]*Department of Artificial Intelligence, School of Informatics, Xiamen University, 361005, China*
[b]*Department of Computer Science, Aberystwyth University, UK*
[c]*Computer Science and Digital Technologies Department, Northumbria University, UK*
[d]*Department of Electrical Engineering, Yuan Ze University, Taiwan*

Abstract

As a combination of robotic motion planning and Chinese calligraphy culture, robotic calligraphy plays a significant role in the inheritance and education of Chinese calligraphy culture. Most existing calligraphy robots focus on enabling the robots to learn writing through human participation, such as human-robot interactions and manually designed evaluation functions. However, because of the subjectivity of art aesthetics, these existing methods require a large amount of implementation work from human engineers. In addition, the written results cannot be accurately evaluated. To overcome these limitations, in this paper, we propose a robotic calligraphy model that combines a generative adversarial network (GAN) and deep reinforcement learning to enable a calligraphy robot to learn to write Chinese character strokes directly from images captured from Chinese calligraphic textbooks. In our proposed model, to automatically establish an aesthetic evaluation system for Chinese calligraphy, a GAN is first trained to understand and reconstruct stroke images. Then, the discriminator network is independently extracted from the trained GAN and embedded into a variant of the reinforcement learning method, the "actor-critic

*Corresponding author
Email addresses: wuruiqi@stu.xmu.edu.cn (Ruiqi Wu), 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), cns@aber.ac.uk (Changjing Shang)

model", as a reward function. Thus, a calligraphy robot adopts the improved actor-critic model to learn to write multiple character strokes. The experimental results demonstrate that the proposed model successfully allows a calligraphy robot to write Chinese character strokes based on input stroke images. The performance of our model, compared with the state-of-the-art deep reinforcement learning method, shows the efficacy of the combination approach. In addition, the key technology in this work shows promise as a solution for robotic autonomous assembly.

Keywords: Robotic calligraphy, motion planning, deep reinforcement learning, generative adversarial nets, robot control

2010 MSC: 00-01, 99-00

1. Introduction

Calligraphic robots, as an integration of intelligent robots and human culture, have broad applications in cultural inheritance and education [1]. In the task of robotic calligraphy, a robot must plan and execute writing actions according to signals given by humans or the environment. To achieve this task, the model must have many abilities, such as perceiving environmental information [2], planning and executing complex actions [3], and evaluating writing results [4]. To implement these capabilities, many technologies, such as robot motion planning [5], human-computer interaction [6, 7], and evaluation method construction [8], are required. In particular, these abilities are fundamental technologies for autonomous robots in industrial and daily-life applications. Therefore, if a method can successfully enable robots to learn human calligraphy automatically, such a method can also be applied to other complex tasks for robots [9], such as robot-based product assembly systems [10, 11].

Researchers have studied the field of robots learning calligraphy from various points of view. The primary issue in this field is determining how to evaluate the writing quality of robotic calligraphy. In response to this problem, several researchers use human aesthetic feedback as the evaluation criterion [12, 13]. For example, Chao et al. developed a human-robot interaction model to evaluate writing results [14, 15]. Other researchers manually designed evaluation methods to evaluate the quality of calligraphy [16, 17, 18]. For example, Zhou et al. developed a probability-based evaluation method to evaluate the quality of Chinese Calligraphy [19]. Another important issue in the field of robotic calligraphy is how robots learn to write calligraphy. Within the methods proposed in recent years, researchers mainly used the following two ways to solve this problem [20] [21] [22]: (1) Humans teach robots calligraphy through human demonstrations and human-computer interactions [23] [24]; and (2) Robots learn calligraphy by using calligraphy data to train themselves [25] [26] [27].

The human-robot interaction method leads robots to write and avoids the problem of evaluating the results of the writing; however, humans must participate in the entire training phase, and such participation greatly increases the human workload [28] [29]. In addition, human-robot interaction is mostly based on visual information, which requires robots to accurately recognize human actions through visual information [30, 31]. However, this type of method makes the learning effect dependent on the success rate of action recognition [32, 33]. The data used by the robot to learn to write are a set of pictures of characters or a set of writing motion trajectories [34]. In addition, to ensure the quality and diversity of the writing results of a robot, human engineers must add as many types of fonts as possible to the training data and even directly use computer fonts. However, the size of the font data set limits the quality and diversity of the writing results [35]. Another drawback of this type of method is that a human must also design evaluation rules to assess the learning effects of the robots.

In response to the problems in the above methods, and inspired by the process of humans learning to write calligraphy, we propose a new learning model to enable a calligraphy robot to learn to write Chinese character strokes by using images of Chinese character strokes. The process used by humans to learn writing is divided into two steps: First, humans must understand a character image from a textbook and mentally reproduce the character. Second, humans learn to write a target character based on the mentally reproduced image and improve the quality of the writing by comparing the writing results with reference images. The advantage of this learning procedure is as follows: without any guidance or evaluations from other persons, a human can still learn writing by himself or herself. Inspired by this human learning process, we propose an approach that, based on input stroke images, enables a robot to simulate how to learn to write Chinese strokes. Thus, the first step of the learning process can be regarded as an image reconstruction task. In particular, generative adversarial networks (GANs) are very suitable for handling this type of task. The second step can be seen as an action-evaluation-improving action process. This process is also the working mechanism of reinforcement learning. Therefore, this study combines GAN and deep reinforcement learning methods to simulate the calligraphy learning process of humans. The GAN model is used to simulate the process of human understanding and reproduce input stroke images. The deep reinforcement learning method is used to simulate the second process in which humans learn to write based on mentally reproduced images. In addition, we apply the GAN model to the reinforcement learning algorithm as a reward function to assist in the training of the robot. To reduce the complexity of the method, the writing behaviour of the robot is modelled as a continuous control problem. Therefore, the deep reinforcement learning model is optimized by the deep deterministic policy gradient (DDPG) algorithm.

Our method differs from the traditional methods in that no human guidance is required and that the evaluation function is learned from the training data. Comparative experiments reveal that the proposed approach requires neither human guidance nor human-designed evaluation rules. Additionally, the size of the data set does not limit the quality and diversity of the strokes generated by the method. Even if the data set is small, the method still enables the robot to achieve high-quality writing results. The main contributions of this work are as follows: (1) The GAN method is used as a reward function to train the deep reinforcement learning methods. (2) The writing behaviour of the robot is modelled as a continuous control problem; thus, the DDPG algorithm is applied to optimize the integration models of the GAN and deep reinforcement learning.

The remainder of this paper is organized as follows: Section 2 introduces the related background knowledge of the proposed model. Section 3 describes the proposed model, which writes strokes according to specified stroke labels and styles. Section 4 specifies the experimental procedures and discusses the experimental results. Section 5 concludes the work and points out directions for future work.

2. Background

2.1. Generative Adversarial Network

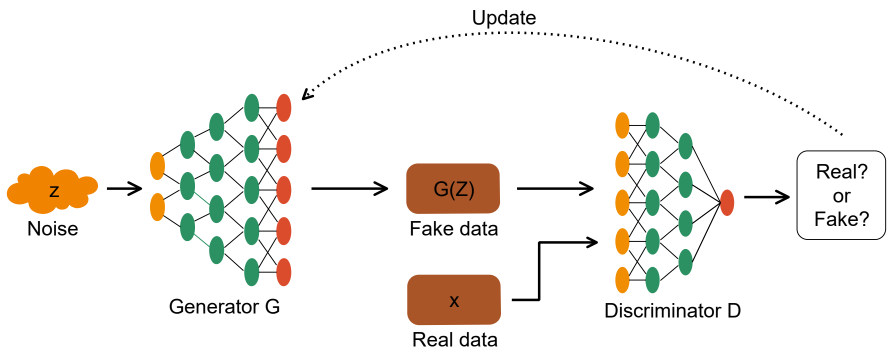


Figure 1: The flowchart of the GAN model.

The GAN model is a combination of a generative model and a discriminative model. The GAN consists of two networks: a generator network and a discriminator network [36]. The function of the discriminator is to determine whether the input sample lies within the distribution of the real data set. The goal of the generator is to generate samples within the distribution of the real data set. When the distance between the two distributions of the real data set and the generated samples is minimal, the generator is optimal. These two sub-models play a continuous game where the generator learns to produce increasingly realistic samples, and the discriminator learns to become increasingly powerful in distinguishing the generated data from the real data. The original GAN model is shown in Fig. 1. The generator takes noise as input and generates samples; the discriminator receives samples from both the generator and the training data and then distinguishes between the two sources. These two networks are trained simultaneously, with the expectation that samples indistinguishable from real data will be generated.

The properties of the GAN model allow the generator network to generate data that does not exist in the real data set but is in the same data distribution as the real data. As an unsupervised learning method, GAN has been widely used in many tasks, including image generation, semantic segmentation, video prediction, etc. Therefore, scholars have proposed many variations of GAN, including DCGAN [37], Wasserstein GAN [38], SeqGAN [39], etc.

2.2. Actor-Critic Model

Deep reinforcement learning, which combines the perception ability of deep learning with the decision-making ability of reinforcement learning [40], directly outputs corresponding control actions based on the input of high-dimensional state information [41]. Deep learning has achieved unprecedented performance in a variety of contexts due to its ability to automatically learn salient feature representations without the use of manual feature engineering. However, a shortcoming of deep learning is that these extracted features are ad hoc, labour-intensive, and not necessarily generalizable to multiple contexts; thus, this shortcoming limits the applicability of deep learning. Reinforcement learning plays an important role in generating a large quantity of training data with a low cost. Therefore, deep reinforcement learning has excellent prospects for application [42, 43]. As an artificial intelligence approach that is closer to the human developmental model, deep reinforcement learning has surpassed human players in many tasks [44].

As a kind of deep reinforcement learning structure, the actor-critic model (AC model) is applied to many continuous control tasks and, in several tasks, performs superior to humans [45]. The AC model consists of two modules: the actor network and the critic network. The function of the critic network is to learn the evaluation mechanism of the environment according to the interaction results of the agent and the environment. The goal of the actor network is to find the optimal action strategy for an agent based on the feedback of the critic network. Unlike the GAN model, to find the optimal strategy, these two sub-modules cooperate with each other. On the one hand, the actor explores a large number of experiences to help the critic fit the evaluation mechanism of the environment. On the other hand, the critic guides the actor to find the optimal strategy through a policy update algorithm. Thus, these two sub-modules are trained simultaneously. The actor model, which does not require an optimal action sample, finds the optimal action based on the exploration mechanism and feedback of the critic. Therefore, the AC model can be applied, as an efficient algorithm, to the field of robotic automatic control.

3. Proposed approach

3.1. Overview

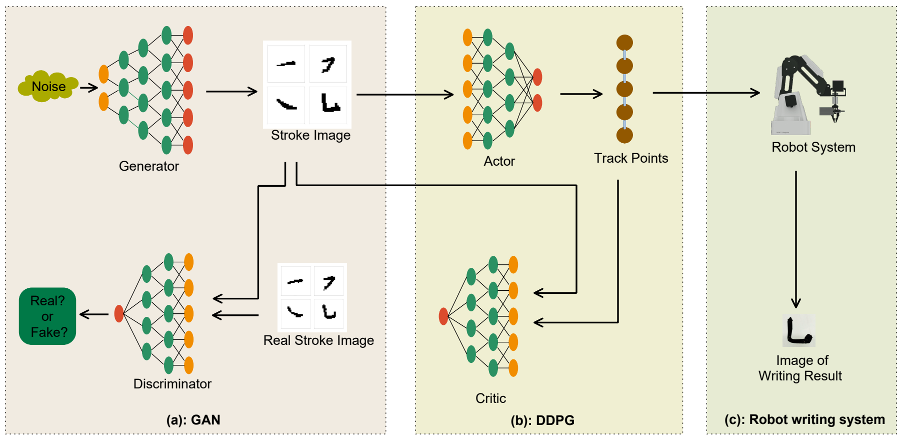


Figure 2: The flowchart of the proposed approach to robotic handwriting. The approach contains three modules: (a) GAN model, (b) actor-critic model, and (c) robot writing module.

To solve the problems that exist in a robot learning to write Chinese character strokes from images, an approach combining the GAN model and the actor-critic model is proposed. Shown in Fig. 2 is the flowchart of the proposed approach, which consists of three modules: (a) GAN module, (b) actor-critic module, and (c) robot writing module. After pre-training, the GAN module supports two functions for the actor-critic model: (1) the discriminator performs as an evaluation function for the actor-critic model, and (2) the stroke images generated by the generator network are used as the environmental states. With the assistance of the GAN module, the actor-critic module learns to write strokes from stroke images. The robotic system performs the writing motions of these strokes. The robotic system also captures images of the writing results and passes them to the GAN module to train the actor-critic module.

The main function of the robot writing module is to write strokes based on the generated results of the actor. During the training phase of the proposed approach, the function of the robot writing module is to provide an image of the writing result to the discriminator for training the critic network. To accomplish these functions, the robot writing module must complete two tasks: (1) Write corresponding strokes on the writing board based on the stroke action generated by the actor network. (2) Capture the stroke image after the robot arm writes the stroke.

The GAN model consists of two components: the generator network and the discriminator network. The goal of the generator is to generate the stroke image that is as close as possible to the stroke images in the training data set. Conversely, the goal of the discriminator is to determine whether an input is real or produced by the generator network. The process of a robot learning to write according to an image can be modelled as a reinforcement learning process. The image generated by the generator network is denoted as the environmental state. The writing action of the robot is seen as a reaction to the state. According to the evaluation of the environment, the robot learns the best action strategy in this state. Therefore, the actor-critic model is applied in the proposed approach, which is used for a robot to learn control of continuous motions. Similar to the GAN model, the actor-critic model consists of two networks: the critic and the actor. The critic network learns how to evaluate the agent actions for interacting with the environment and provides feedback for the action generated by the actor module. Then, according to the feedback of the critic network, the actor learns the optimized action policy of the agent in the environment.

When the actor-critic model is used for the task of robot control, one challenge is that the evaluation function is indispensable. Therefore, in the proposed approach, the discriminator network of the GAN model replaces the evaluation function. When the GAN model is fully trained, the discriminator approximates the distribution boundary of the stroke image;thus, the discriminator is qualified to evaluate the writing result of the robot. Based on the evaluation results of the discriminator, the actor-critic model learns how to find the optimized writing actions, which correspond to the input stroke image.

3.2. GAN Model

In the proposed approach, the GAN model is used to generate and evaluate the quality of stroke images. The input of the generator network is a set of Gaussian noises. According to these noises, a stroke image is generated by the generator network. In addition, the input of the discriminator is a stroke image, which is from a real data set or generated by the generator. During GAN model training, the function of the discriminator is to identify the source of the stroke image.

In addition, when the GAN model is applied in the training of the actor-critic model, the structure of the discriminator and the loss function of the model must be modified. In the vanilla GAN model, the output of the discriminator is the label of the input data, whose range is from 0 to 1. Moreover, the sigmoid function is used as the activation function of the output layer in the discriminator. Such a sigmoid function brings a nonlinear processing feature into the network. However, the discriminator is unsuitable as an evaluation function to properly assess the actions of the agent. To solve this problem, the sigmoid function is removed from the output layer of the discriminator in the proposed approach. With this modification, the output range $[0, 1]$ of the discriminator is also removed. Thus, the discriminator is better used as an evaluation function to evaluate the actions of the agent.

Due to the modification of the discriminator output layer, the loss functions of the discriminator and the generator also must be adjusted. In the vanilla GAN model, the loss functions of the discriminator and generator are calculated by the cross entropy between the predicted and real labels. In the GAN model, the discriminator and the generator are usually denoted as D and G, respectively. The loss function of the generator is given by Eq. 1. According to the input noises, z, the generator, G, generates a fake image. Note that z is usually sampled from a Gaussian noise distribution. Then, the cross entropy is calculated based on the output of the discriminator, D. Similarly, the loss function of the discriminator is defined by Eq. 2.

$$L_G = \mathbb{E}\left[log(D(G(z)))\right]. \tag{1}$$

$$L_D = \mathbb{E}\left[log(D(x))\right] + \mathbb{E}\left[log(1 - D(G(z)))\right]. \tag{2}$$

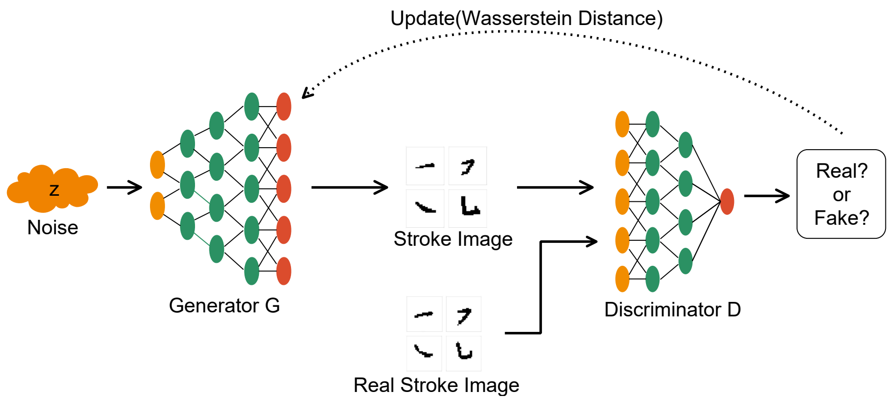


Figure 3: The flowchart for generating stroke images using the GAN model.

Note that, since the architecture of the GAN model is changed, the loss functions from the original GAN model are no longer suitable for the proposed approach. Thus, the loss function from the Wasserstein GAN is introduced into the model [38]. Wasserstein GAN is a variant of the GAN model that uses the Wasserstein distance to replace the K-L divergence to measure the distance between the two data distributions. Therefore, the loss functions of the GAN model are presented in Eq. 3 and Eq. 4. The process of generating stroke images using the GAN model is shown in Fig. 3.

$$L_G = \mathbb{E}\left[-D(G(z))\right]. \tag{3}$$

$$L_D = \mathbb{E}\left[D(G(z))\right] - \mathbb{E}\left[D(x)\right]. \tag{4}$$

In the proposed approach, the discriminator and generator are represented by two multi-layer neural networks. The discriminator network consists of two convolutional layers and two fully connected layers. The input and output layers contain 1,024 and 1 neurons, respectively. In contrast, the generator network consists of a fully connected layer and two transposed conventional layers. The input and output layers of the generator contain 128 and 1,024 neurons, respectively. The activation function of the hidden layers of both networks is leaky ReLU, except that of the output layer in the discriminator network.

3.3. Actor-Critic Network

As shown in part (b) of Fig. 2, the actor-critic model consists of two networks: critic and actor. The actor network, A, generates the corresponding action based on the input environmental states. The critic network, C, predicts the reward value of the action based on the environmental states and the output action from the actor network. As shown in Fig. 4, in the robot learning writing task, the input of the actor network is a Chinese character stroke image generated by the generator. With this image, the actor network generates a set of robot actions used to write strokes. The critic network predicts the reward value of the action according to the action and stroke image generated by the generator. Then, the critic guides the actor to update its action strategy. Note

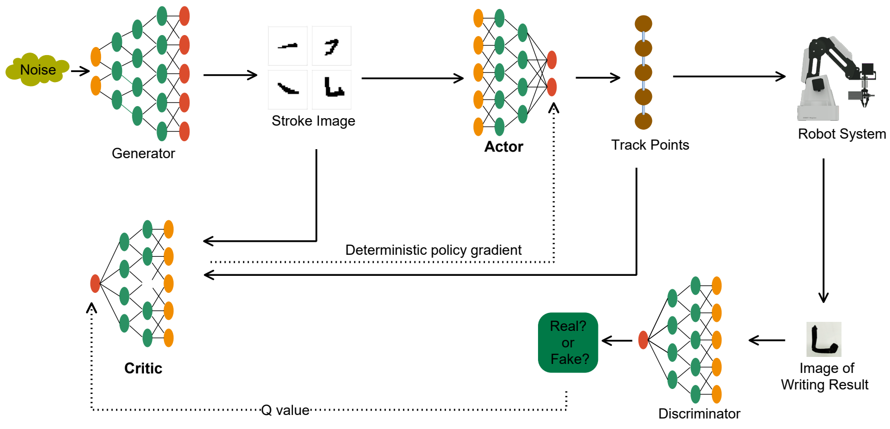


Figure 4: The flowchart for generating stroke action using the actor-critic model.

that the training of the critic network is based on the discriminator's evaluation of the stroke action. Before the discriminator network evaluates the action, the action must be written by the robot writing module and converted into an image. Therefore, the discriminator calculates a reward for the stroke action based on this image.

The evaluation function provided by the discriminator network evaluates only the quality of the entire stroke; the evaluation function cannot evaluate the effect of the stroke writing process. Therefore, the minimum action unit of the writing is to write a complete stroke. In other words, the actor network must generate complete writing actions for writing a stroke within one generation. In this case, the robot writing task is converted to a special type of deep reinforcement learning task where the robot requires a one-time interaction with its environment. Similar to the multi-armed bandit problem [46, 47], the robot writing task becomes a single-step reinforcement learning task, which does not consider the change process of the environment state task as a Markov decision process (MDP). In addition, the actor-critic model is a combination algorithm of policy iteration and value iteration; i.e., it updates the model parameters at each step. Thus, the actor-critic model can also be used for the single-step reinforcement learning task. Since MDP is not a necessary assumption for the environment in the single-step reinforcement learning task, its optimal policy is to maximize single-step rewards. When the actor-critic model is applied to the single-step reinforcement learning task, the Bellman equation for dealing with the MDP character of the environment is redundant. Therefore, the Bellman equation for calculating the Q value is omitted in our method. Additionally, the critic network directly predicts the reward value of the action instead of the Q value.

In terms of the above considerations, the stroke action generated by the actor network is composed of a series of stroke trajectory points, which indicate the end position information of a brush pen. Thus, stroke action, A, is denoted as $A = (a_1, a_2, a_3, \cdots, a_i)$, where i denotes the length of the stroke action, and a_i is denoted as $a_i = (x_i, y_i, z_i)$, which indicates that the position of the end of the brush pen in a Cartesian coordinate system at time i. Therefore, the output size of the actor network is determined by the length of the stroke action, which is $3i$ in this work.

Since the reward value of a stroke action depends on the evaluation function, the loss function, L_C, of the critic network is defined as follows:

$$L_C = \frac{1}{N}\sum_{i=1}^{N}(y_i - D(\hat{x}_i))^2, \tag{5}$$

where y_i denotes the real reward of the stroke action, $\hat{x}_i$ denotes the stroke image generated by the robot, and N denotes the size of each training batch.

In addition, the deep deterministic policy gradient (DDPG) algorithm [45] is used to update the parameters of the actor network. The DDPG algorithm updates the parameters of the actor network by applying the chain rule to the expected return from the start distribution, J, with respect to the actor parameters [45]. The deterministic policy gradient is calculated as follows:

$$\nabla_{\theta^\mu} J = \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q(s, a|\theta^Q)|s = s_t, a = \mu(s_t)\nabla_{\theta^\mu}\mu(s|\theta^\mu)|s = s_t\right], \tag{6}$$

where $\mu(s|\theta^\mu)$ denotes the parameterized actor network, $Q(s, a|\theta^Q)$ denotes the critic network, and θ^Q denotes the parameterized critic network.

Similar to the structure of the discriminator network, the actor network consists of two convolutional layers and two fully connected layers. Since the input to both discriminator and actor is stroke images, the discriminator and the actor share the first two convolutional layers of the discriminator. In the proposed model, the length of stroke action, i, is set to 6; therefore, the output layer of the actor network consists of 18 neurons. The structure of the critic network is simpler than that of the actor and consists of five fully connected layers. Because the inputs of the critic are stroke image and stroke action, the input and output layers contain 1,048 and 1 neurons, respectively. To better represent stroke action, the output layer of the action network uses a sigmoid function as the activation function. Except for that of the output layer in the actor network, the network layers of the actor-critic model use a leaky ReLU as the activation function.

3.4. Robot Writing Module

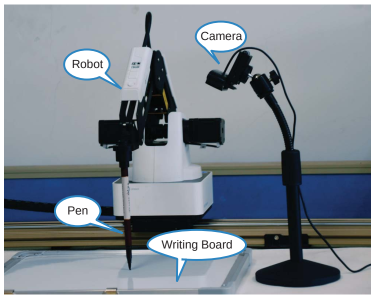


Figure 5: The hardware system of the calligraphy robot.

Fig. 5 shows the hardware system for the experimental robot system. The robot system consists of a 4-DOF robotic arm, a calligraphy brush pen, a writing board, and a camera. The camera is used to capture the written content on the writing board.

Since the stroke action generated by the actor network is a set of Cartesian coordinates representing the end position of the brush pen, the robotic arm must translate the Cartesian coordinates into the angular values of the robot's

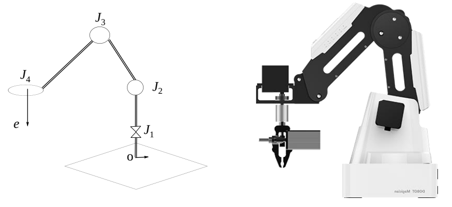


Figure 6: The structure of the robotic arm.

four joints. The structure of the robotic arm is shown in Fig. 6. The Cartesian coordinates of the brush pen, a_i, are seen as the Cartesian distance from the end of calligraphy brush, e, to the origin position, o. When a_i is given, the four joint values of the robotic arm are calculated by inverse kinematics. To reduce the complexity of the inverse kinematics calculation, the j_4 joint is fixed at the time of writing, and only the first three joints are used to control the brush writing.

To reduce noise interference, the image captured by the camera must be processed before being passed to the discriminator network. The camera in the robot writing system is a high-resolution camera. The size of the captured image is 1920×1080 pixels. Because the original image captured by the camera is large, the image must be preprocessed by the camera module before being sent to other models. The image processing module preprocesses the image as follows: (1) An image of the fixed area of the writing board after the robot has finished writing the stroke is captured. (2) The captured image is converted to a binary image. (3) The binary image is scaled to fit the size of the discriminator input.

3.5. Model Training

In the proposed model, the two networks of the GAN model and the two networks of the actor-critic model must be trained. The GAN model provides the environmental states and the evaluation function for the actor-critic model;

therefore, the GAN model must be trained prior to the training of the actor-critic model. In this case, the training of the entire model is divided into two steps: (1) Training the GAN model and (2) Training the actor-critic model.

3.5.1. Training of GAN model

The two networks of the GAN model are trained based on the parameter updating relations defined in Eq. 3 and Eq. 4. As shown by the dotted line in Fig. 3, the discriminator is trained according to the input stroke images and corresponding labels. The generator is trained based on the output of the discriminator. To balance the learning speed of these two networks, the training policy of the GAN model is set where the discriminator is trained once and the generator is trained twice within a training epoch. The size of the stroke image is set to 32×32 pixels.

3.5.2. Training of the actor-critic model

After the GAN model completes training, the discriminator and the generator are used for the training of the actor-critic model. As shown by the dotted line in Fig. 4, the training sample of the critic consists of two parts: the stroke image generated by the generator and the stroke action generated by the actor. The corresponding training labels are represented by the evaluation of the stroke action by the discriminator. Before the discriminator evaluates the stroke action, the robot writing system must convert the action into an image. The actor is trained according to the evaluation of the critic to the actions and state images. Based on the input action and images, the critic provides a deterministic policy gradient to the actor, which is used to update the parameters of the actor. Similar to the training policy of the GAN model, to ensure that the learning speeds of the two networks are balanced, the training policy of the actor-critic model is set such that the critic network is trained once and the actor network is trained twice. In addition, to prevent the critic network from over-fitting when it is learning from the discriminator network, the discriminator network still trains itself while the actor-critic model is training. Unlike the training in the GAN model, for every 1,000 training epochs in the training of the actor-critic model, the discriminator network trains five epochs.

Note that the loss function of the discriminator is also modified. Thus, the writing images captured by the robot writing module are embedded in the new loss function, defined as follows:

$$L_D = \mathbb{E}\left[\lambda D(G(z)) + (1-\lambda)D(\hat{x})\right] - \mathbb{E}\left[D(x)\right]. \tag{7}$$

where λ is a hyper-parameter introduced into the loss function, λ is used to control the weights of the generator and actor networks during the training of the discriminator network, and $\hat{x}$ represents the writing result generated by the robot writing module based on the output of the actor network.

In the training of the actor-critic model, a terminal threshold, τ, is set to terminate the training process. The setting of τ is done in consideration of two factors: (1) whether the critic network converges, and (2) whether, based on the current critic network, the actor network converges. To determine whether the actor network converges, we calculated the ratio of the loss value of the critic to the loss value before 100 training epochs. The loss value ratio is expressed as follows:

$$R_d = \left| \frac{L_{t-100} - L_t}{L_{t-100}} \right|, \tag{8}$$

where t denotes the training epochs of the actor-critic model, and L_t denotes the loss value of the critic network in the i-th training epochs. The discriminator is used as an evaluation function to assign an evaluation score for each input image. Thus, after t epochs of training, the image score is represented as $D(\hat{x}_t)$. Consequently, a score ratio, R_s, of $D(\hat{x}_t)$ and $D(\hat{x}_{t-100})$ measures whether the actor network converged. The calculation of the score ratio, R_s, is as follows:

$$R_s = \left| \frac{D(\hat{x}_{t-100}) - D(\hat{x}_t)}{D(\hat{x}_{t-100})} \right|, \tag{9}$$

According to Eq. 8 and Eq. 9, the terminal threshold τ is calculated as follows:

$$\tau = \begin{cases} 1 & \text{if } R_d \leq \tau_d, \text{ and } R_s \leq \tau_s \\ 0 & \text{else} \end{cases}, \tag{10}$$

where τ_d and τ_s are two hyper-parameters. When the value of τ is 1, the training of the actor-critic model is terminated.

To explore continuous control tasks for the calligraphy robot, noises are added to the generated actions during the training process. The Ornstein-Uhlenbeck process [45] is also used to generate explored actions. The Ornstein-Uhlenbeck process produces temporally correlated explorations in physical control problems. Since the robot model generates the entire action one time, when adding noise, multiple actions generated by the actor network share an Ornstein-Uhlenbeck process. Thus, the Ornstein-Uhlenbeck process is reset every 1,000 training epochs. Finally, the summarized training processes of the proposed model are shown in Algorithm 1.

4. Experiments and analysis

4.1. Experimental details

To verify the validity of the proposed model, the model was applied to an experiment in writing Chinese character strokes. Then, a set of ablation experiments was designed to verify the performance of each sub-module of the proposed model. Finally, to compare the performance of the proposed method with other methods, a set of comparative experiments was designed.

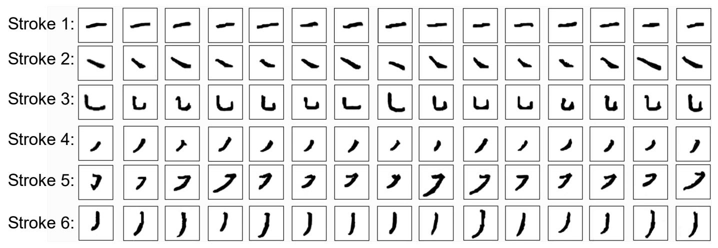


Figure 7: Training samples used in the experiment; six strokes are labelled from stroke 1 to 6.

Algorithm 1 Training phase of method

```
Require: Real stroke image data set, $X$, and random number, $Z$;
 1: Initialize $G$, $D$, $A$, $t$, and $C$ with random weights;
 2: for $t$ in $p$ do
 3:     Produce a set of random number $z$;
 4:     Generate stroke image, $x_g$, through $G$;
 5:     Output $D(x)$ and $D(x_g)$ according to $x$ and $x_g$;
 6:     Calculate $G_{loss}$ and $D_{loss}$ by Eq. 3 and Eq. 4;
 7:     Use $G_{loss}$ to update $G$ parameters;
 8:     Use $D_{loss}$ to update $D$ parameters;
 9:     $t++$
10: end for
11: while $\tau \neq 1$ do
12:     Generate stroke image, $x_g$, through $G$;
13:     Generate stroke action, $a$, according to $x_g$;
14:     Robot writes the stroke accumulate to $a$; get an image $\hat{x}$;
15:     Input $a$ and $\hat{x}$ into $C$, and get $\nabla_{\theta^\mu}$ by Eq. 6;
16:     Use $\nabla_{\theta^\mu}$ to update $A$ parameters;
17:     if $t$ mod $2 == 0$ then
18:         Input $\hat{x}$ into $D$, and get $D(\hat{x})$;
19:         Update $C$ parameters by Eq. 5;
20:     end if
21:     if $t$ mod $1000 == 0$ then
22:         Generate stroke image, $x_g$, through $G$;
23:         Output $D(x)$, $D(x_g)$ and $D(\hat{x})$ according to $x$, $x_g$ and $\hat{x}$;
24:         Calculate $D_{loss}$ by Eq. 7;
25:         Use $D_{loss}$ to update $D$ parameters;
26:     end if
27:     $t++$
28: end while
```

In these experiments, the training of all the methods is based on a stroke image data set extracted from a set of simple characters from Chinese Calligraphic textbooks. In addition, this stroke data set contains six different types of Chinese character strokes. The data set contains more than 3,500 stroke samples, with each stroke having more than 500 samples. Fig. 7 illustrates some of the stroke samples from the training data set.

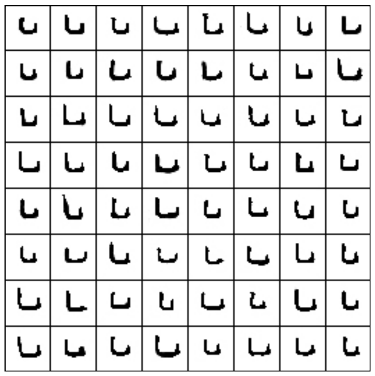


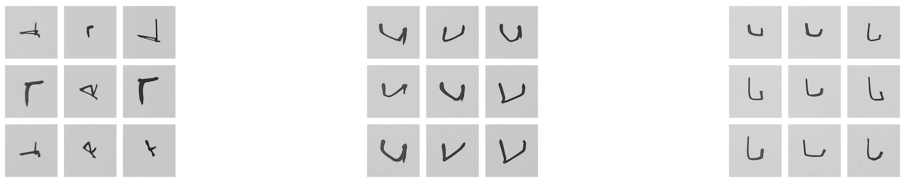
Figure 8: The stroke image is generated by the generator network after 2000 training epochs.

All the networks in the proposed approach are optimized by the Adam optimization algorithm [48], and the learning rate is set to 0.001. In the training of the actor-critic model, the hyper-parameter, λ, is set to 0.05. The two parameters, τ_d and τ_s, of the terminal threshold are set to 0.03 and 0.06, respectively. The hyper-parameters θ and σ of the Ornstein-Uhlenbeck process are set to 0.15 and 0.2, respectively. In addition, the size of the mini-batches is set to 64 for all networks. Empirically, the GAN model generates high-quality images after 2,000 training epochs. For example, after 2,000 training epochs, the effect of the generated stroke image is shown in Fig. 8. Therefore, in this experiment, the pre-training times of the GAN model, p, are set to 2,000. The values of all parameters are shown in Table. 1. All neural networks in the proposed approach are constructed using TensorFlow 1.5 [49], which is an open-source library for deep learning. Moreover, the proposed approach is trained on a GPU-based algorithm computer.

Hyper-parameters	**Values**	**Hyper-parameters**	**Values**
learning rate	0.001	τ_d	0.03
batch size	64	τ_s	0.06
θ	0.15	σ	0.2
λ	0.05	p	2000

Table 1: The values of hyper-parameters.

4.2. Experimental results



(a) Early Phase (b) Medium Phase (c) Final Phase

Figure 9: Robotic writing results in different training stages.

Fig. 9 shows the writing results of one stroke in three training phases: early, medium, and final. In the early phase of the GAN model, the actor-critic model did not gain any knowledge on stroke writing; therefore, the writing trajectories shown in Fig. 9a are more like randomly generated lines. As shown in Fig. 9b, after a number of training epochs, the model writes the approximate contour of the stroke in the medium stages of training. Then, more writing details are obtained by the model; thus, the model successfully learns how to write the stroke in the final phase. As shown in Fig. 9c, the writing results demonstrate very high quality.

Fig. 10 shows the process steps of the robotic arm writing a stroke. In Step 1, the robotic arm controls the brush pen attached to the arm to reach a predefined position. Then, from Step 2 to Step 5, the robotic arm moves the brush pen according to the action trajectory points generated by the actor

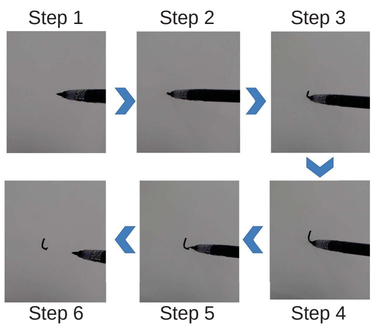


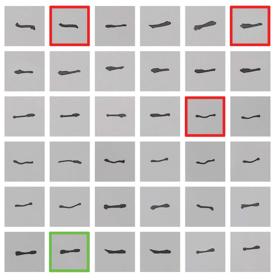
Figure 10: Robotic arm in action writing a stroke.

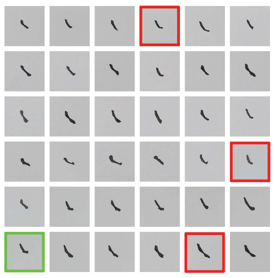
network. The arrows in Fig. 10 indicate the sequence of motion of the brush. Finally, when all the trajectory points of the stroke have been executed, the brush returns to a predefined position (See Step 6).

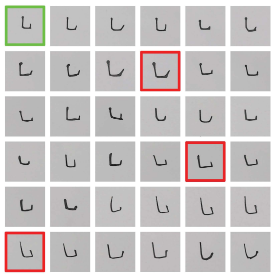
The writing performance of the proposed approach on the stroke data set is shown in Fig. 11. After the training of the model is complete, the model can successfully write all six strokes of the stroke data set. Although there are still a few incorrect strokes, most of the results show the main features of strokes. In addition, among the results, the quality of several writings has reached the level of human writing (e.g., see the strokes marked with a green box in Fig. 11). In addition, as shown in the strokes marked with a red box in Fig. 11, for each type of stroke, one batch of stroke action generated by the actor network is different from the others. This shows that the noise mechanism of the actor-critic model is effective in exploring stroke action diversity.

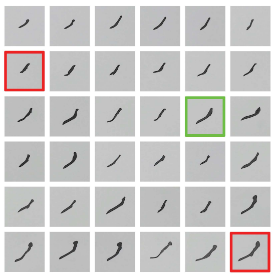
4.3. Ablation experiments

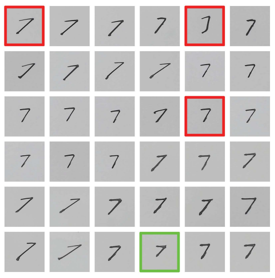
To determine the advantages of the GAN model of our method and to analyse the writing results using numerical values, in this section, two ablation experiments are introduced: the DDPG method and no-noise method. The DDPG method is from our previous work [50], in which a typical actor-critic model was used. Compared with our current work, the DDPG method merely applied a manually designed reward function rather than a GAN model. Without the GAN model, the DDPG method became a purely deep reinforcement learning


(a) Stroke 1


(b) Stroke 2


(c) Stroke 3


(d) Stroke 4


(e) Stroke 5

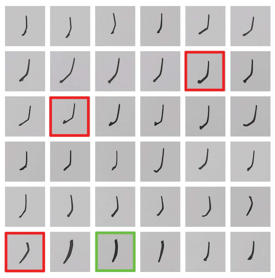

(f) Stroke 6

Figure 11: Writing results of all the six strokes.

approach. In the no-noise experiment, the noise generated by the Ornstein-Uhlenbeck process was removed during the training of the actor-critic module. This ablation experiment was used to evaluate the performance impact of the exploration mechanism of the proposed model. Note that the DDPG experiment and the no-noise experiment were trained on the same stroke data set as that of the proposed method.

In the DDPG experiment, the cosine similarity was used as the reward function. Cosine similarity is usually used to measure the similarity between two vectors; if the two vectors are similar, their cosine value is larger and close to 1; otherwise, their cosine distance is smaller. Since the GAN model is replaced by the cosine similarity, the state image is also replaced by the stroke image from the data set. In this experiment, the state image from the data set is represented by s_{img}, and the image of the writing result captured by the robot writing module is represented by a_{img}. The cosine distance, $R(a, s)$, between s_{img} and a_{img} is used as the reward value of the generated action. The reward value combines with the action of the agent and state image, s_{img}, to train the critic network. Thus, $R(a, s)$ is defined as follows:

$$R(a, s) = \frac{s_{img} \cdot a_{img}}{\|s_{img}\| \|a_{img}\|}, \tag{11}$$

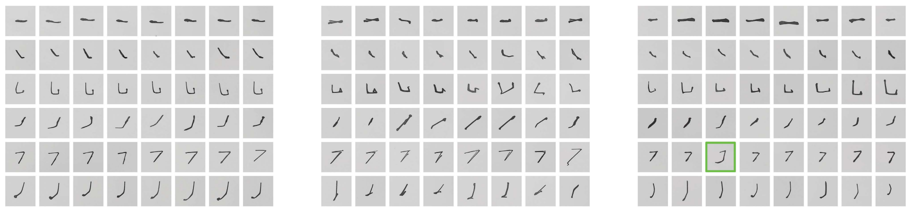

(a) No-noise method (b) DDPG method (c) Our method

Figure 12: Comparison of the writing results. (a), (b) and (c) represent the writing results of the no-noise, DDPG and proposed methods, respectively.

Fig. 12 shows the experimental results. Although the no-noise method also learns to write these six strokes, it can be seen from Fig. 12a that the generated results, compared with other methods, lose many local features of the strokes. This phenomenon is more significant for complex strokes. For example, the no-noise method cannot generate smooth stroke 5 (shown in the highlighted picture in Fig. 12c), which is generated by our proposed method. In addition, all the writing results are very consistent. In other words, the writing style of the method lacks diversity. Therefore, through this experiment, the exploration mechanism is proven to have an impact on the local features and diversity of the generated results. Moreover, the results generated by the DDPG model (Fig. 12b) are worse than the results generated in the other two sets of experiments. Fig. 12b shows that there is a major difference between the experimental results and the real strokes that still cannot be eliminated. The reason for this difference may be due to the limitations of the reward function.

To further objectively evaluate the performance of the model and collectively measure the similarity between training samples and generated samples, the Frechet inception distance (FID) [51] was introduced into the experiment; the more similar two image data sets are, the smaller their FID values are. In this experiment, the FID values between the training data set and the stroke images written by the three models to measure their performance were calculated and are shown in Table 2.

Writing results	**DDPG method**	**No-noise method**	**Our method**
Stroke 1	85.89	64.80	**62.14**
Stroke 2	91.69	68.04	**61.11**
Stroke 3	93.70	61.15	**55.24**
Stroke 4	88.02	72.12	**62.45**
Stroke 5	92.67	54.58	**53.51**
Stroke 6	94.39	59.73	**56.04**

Table 2: FID values of the writing results and baselines.

Each row of Table 2 represents the FID values for a class of stroke over the three methods. The scores for the no-noise method are between 54 and 73; the scores for the DDPG method are between 85 and 95; and the score range of the proposed method, 53 to 63, is the lowest. From Table 2, the scores between the three methods are consistent with the difference in human observation shown in Fig. 12. For example, the FID values of the DDPG model for each type of stroke are larger than the FID values of the proposed model, especially for strokes 3 and 5.

Through this ablation experiment, we found that the GAN model plays a key role in the proposed method. Compared with the traditional reinforcement learning method, the method using the discriminator network as the reward function leads the model to perform better. Furthermore, compared with the manually designed reward function, the GAN model has two advantages as a reward function: (1) The discriminator network evaluates the sample more accurately; thus, the GAN model is more suitable for guiding the learning of the agent. (2) The design of the GAN model is convenient. Compared with the artificially designed reward functions, the design of the GAN model does not need to overly consider the details of the sample.

4.4. Comparative analysis

The proposed method was also compared with another existing GAN-based calligraphy robot study. This comparison study, proposed by Chao et al. [35], applies the stochastic policy gradient method to the GAN model. The essence of the task of robots learning to write through images is the mapping of data from image space to action space. Therefore, the traditional GAN models cannot be used directly in these tasks. To address this problem, Chao et al. use the policy gradient method in the training of the generator network to implement the backpropagation of the gradient information.

Similar to the ablation experiment in the previous section, the proposed method and the GAN method are also compared on the same stroke data set, and the FID evaluation method is used to measure the performance of both. The experimental results, summarized in Table 3, show that our method performs

Writing results	**GAN method**	**Our method**
Stroke 1	59.62	62.14
Stroke 2	60.82	61.11
Stroke 3	58.49	**55.24**
Stroke 4	56.19	62.45
Stroke 5	64.89	**53.51**
Stroke 6	57.77	**56.04**

Table 3: FID value comparison of our model and the comparison models.

similarly to the method of Chao et al. However, the proposed method generates better results for strokes 3, 5 and 6.

In the work of Chao et al., the action trajectory points are represented in the form of discrete data; therefore, the size of their GAN model is very large. In contrast, our method represents the action trajectory points in the form of continuous data; therefore, our GAN model size is much smaller. The reason is because in the GAN based method, the trajectory point is represented as discrete data; $804 \cdot i$ neurons are used to represent a trajectory sequence, where i denotes the length of the trajectory sequence. In contrast, our model uses continuous data to represent the trajectory points, and $3 \cdot i$ neurons are used to express a trajectory sequence. Therefore, the network's size has been significantly reduced.

In summary, our method successfully solves the learning task of writing Chinese strokes through stroke images. Compared with the existing methods, our method has two characteristics: (1) Compared with a method based on reinforcement learning, our method has a simple and accurate reward function, which ensures that our method generates high-quality writing actions and optimizes the learning efficiency of the agent. (2) Compared with a method based on the GAN model, in our method, the robot writing action is presented as a set of continuous values, which simplifies the complexity of the calligraphic learning.

5. Conclusion

In this work, a robotic calligraphy learning approach that mimicks a human's learning process was developed. A GAN model simulated the behaviour of humans recognizing images and re-imaged in their minds. A deep reinforcement learning method simulated the mechanism of a human learning to write. Finally, we used the trained GAN model to assist in the training of the deep reinforcement learning method. Without a large number of training samples and a manually designed evaluation function, the reinforcement learning automatically learns to write with the assistance of the GAN model. The training results show that the robot successfully learns how to write strokes.

We conducted a series of ablation experiments to test the importance and efficiency of the various components of the model. The experimental results demonstrate that the addition of the GAN module as a reward function for the deep reinforcement learning method is a key factor in the successful training of the model. Moreover, exploration noise played an important role in the learning speed and learning efficiency of the model. In addition, because our model is a continuous control structure, the entire GAN model has a simpler network architecture.

Although the proposed approach is promising for a robot learning to write strokes from images, it still has room for improvement. For example, a traditional GAN model, which is used mainly to deal with image generation problems, is used in our method. Thus, our model generates and evaluates the entire action directly and does not consider the temporal nature of the writing action. If we divide the action into a series of sub-actions and train the GAN model to appropriately evaluate these sub-actions, then the GAN model will provide a more accurate reward function for the deep reinforcement learning model.

Acknowledgement

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 (Nos. 61673322, 61673326, and 91746103), the Fundamental Research Funds for the Central Universities (No. 20720190142), Natural Science Foundation of Fujian Province of China (Nos. 2017J01128 and 2017J01129), and the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 663830.

References

[1] H. Zeng, Y. Huang, F. Chao, C. Zhou, Survey of robotic calligraphy research, CAAI Transactions on Intelligent Systems 11 (1) (2016) 15–26. `doi:10.11992/tis.201507067`.

[2] F. Chao, F. Chen, Y. Shen, W. He, Y. Sun, Z. Wang, C. Zhou, M. Jiang, Robotic free writing of chinese characters via human–robot interactions, International journal of humanoid robotics 11 (01) (2014) 1450007.

[3] V. Potkonjak, Robot handwriting: Why and how?, in: A. Kecskeméthy, V. Potkonjak, A. Müller (Eds.), Interdisciplinary Applications of Kinematics, Springer Netherlands, Dordrecht, 2011, pp. 19–35.

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

[5] K. Sasaki, K. Noda, T. Ogata, Visual motor integration of robot's drawing behavior using recurrent neural network, Robotics and Autonomous Systems 86 (2016) 184 – 195. `doi:https://doi.org/10.1016/j.robot.2016.08.022`.
URL `http://www.sciencedirect.com/science/article/pii/S0921889016305383`

[6] X.-h. Ma, Q.-z. Kong, W.-m. Ma, X.-w. Zhang, 4-dof lettering robot's trajectory planning, Mech. Eng. Autom 165 (5) (2010) 161–163.

[7] 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`.

[8] J. Garrido, W. Yu, A. Soria, Human behavior learning for robot in joint space, Neurocomputing 155 (2015) 22 – 31. `doi:https://doi.org/10.1016/j.neucom.2014.12.068`.
URL `http://www.sciencedirect.com/science/article/pii/S0925231214017329`

[9] D. Berio, S. Calinon, F. F. Leymarie, Generating calligraphic trajectories with model predictive control, in: Proceedings of the 43rd Graphics Interface Conference, GI '17, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2017, pp. 132–139. `doi:10.20380/GI2017.17`.
URL `https://doi.org/10.20380/GI2017.17`

[10] T. R. Savarimuthu, A. G. Buch, C. Schlette, N. Wantia, J. Romann, D. Martinez, G. Aleny, C. Torras, A. Ude, B. Nemec, A. Kramberger, F. Wrgtter, E. E. Aksoy, J. Papon, S. Haller, J. Piater, N. Krger, Teaching a robot the semantics of assembly tasks, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (5) (2018) 670–692. `doi:10.1109/TSMC.2016.2635479`.

[11] K. Lee, S. Joo, H. I. Christensen, An assembly sequence generation of a product family for robot programming, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 1268–1274. `doi:10.1109/IROS.2016.7759210`.

[12] P. Liang, C. Yang, Z. Li, R. Li, Writing skills transfer from human to robot using stiffness extracted from semg, in: 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015, pp. 19–24. `doi:10.1109/CYBER.2015.7287903`.

[13] C. Yang, S. Chang, P. Liang, Z. Li, C. Su, Teleoperated robot writing using emg signals, in: 2015 IEEE International Conference on Information and Automation, 2015, pp. 2264–2269. `doi:10.1109/ICInfA.2015.7279663`.

[14] F. Chao, Y. Huang, X. Zhang, C. Shang, L. Yang, C. Zhou, H. Hu, C.-M. Lin, A robot calligraphy system: From simple to complex writing by human gestures, Engineering Applications of Artificial Intelligence 59 (2017) 1 – 14. `doi:https://doi.org/10.1016/j.engappai.2016.12.006`.
URL `http://www.sciencedirect.com/science/article/pii/S0952197616302329`

[15] F. Chao, Y. Sun, Z. Wang, G. Yao, Z. Zhu, C. Zhou, Q. Meng, M. Jiang, A reduced classifier ensemble approach to human gesture classification for robotic chinese handwriting, in: 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2014, pp. 1720–1727. `doi:10.1109/FUZZ-IEEE.2014.6891656`.

[16] 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.

[17] 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`.

[18] Z. Ma, J. Su, Stroke reasoning for robotic chinese calligraphy based on complete feature sets, International Journal of Social Robotics 9 (4) (2017) 525–535. `doi:10.1007/s12369-017-0410-2`.
URL `https://doi.org/10.1007/s12369-017-0410-2`

[19] D. Zhou, J. Ge, R. Wu, F. Chao, L. Yang, C. Zhou, A computational evaluation system of chinese calligraphy via extended possibility-probability distribution method, in: 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2017, pp. 884–889. `doi:10.1109/FSKD.2017.8393393`.

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

[21] R. B. Warrier, S. Devasia, Iterative learning from novice human demonstrations for output tracking, IEEE Transactions on Human-Machine Systems 46 (4) (2016) 510–521. `doi:10.1109/THMS.2016.2545243`.

[22] H. Yin, P. Alves-Oliveira, F. S. Melo, A. Billard, A. Paiva, Synthesizing robotic handwriting motion by learning from human demonstrations, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, AAAI Press, 2016, pp. 3530–3537.
URL `http://dl.acm.org/citation.cfm?id=3061053.3061114`

[23] J. Li, W. Sun, M. Zhou, X. Dai, Teaching a calligraphy robot via a touch screen, in: 2014 IEEE International Conference on Automation Science and Engineering (CASE), 2014, pp. 221–226. `doi:10.1109/CoASE.2014.6899330`.

[24] V. Mohan, P. Morasso, J. Zenzeri, G. Metta, V. S. Chakravarthy, G. Sandini, Teaching a humanoid robot to draw 'shapes', Autonomous Robots 31 (1) (2011) 21–53. `doi:10.1007/s10514-011-9229-0`.
URL `https://doi.org/10.1007/s10514-011-9229-0`

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

[26] H. Lifei, Z. Jing, H. Lingling, An improved algorithm for extracting the skeletons of the Chinese calligraphy, Microcomputer & Its Applications 17 (2011) 025.

[27] F. Yao, G. Shao, Modeling of ancient-style Chinese character and its application to CCC robot, in: Networking, Sensing and Control, 2006. IC-

NSC'06. Proceedings of the 2006 IEEE International Conference on, IEEE, 2006, pp. 72–77.

[28] F. Chao, Z. Wang, C. Shang, Q. Meng, M. Jiang, C. Zhou, Q. Shen, A developmental approach to robotic pointing via human-robot interaction, Information Sciences 283 (2014) 288 – 303, new Trend of Computational Intelligence in Human-Robot Interaction. `doi:https://doi.org/10.1016/j.ins.2014.03.104`.
URL `http://www.sciencedirect.com/science/article/pii/S0020025514004010`

[29] P. M. Yanik, J. Manganelli, J. Merino, A. L. Threatt, J. O. Brooks, K. E. Green, I. D. Walker, A gesture learning interface for simulated robot path shaping with a human teacher, IEEE Transactions on Human-Machine Systems 44 (1) (2014) 41–54. `doi:10.1109/TSMC.2013.2291714`.

[30] J. Bandera, J. Rodriguez, L. Molina-Tanco, A. Bandera, A survey of vision-based architectures for robot learning by imitation, International Journal of Humanoid Robotics 9 (01) (2012) 1250006.

[31] H. Lin, Y. Huang, Visual matching of stroke order in robotic calligraphy, in: 2015 International Conference on Advanced Robotics (ICAR), 2015, pp. 459–464. `doi:10.1109/ICAR.2015.7251496`.

[32] 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) (2017) 253.

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

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

[35] F. Chao, J. Lv, D. Zhou, L. Yang, C. Lin, C. Shang, C. Zhou, Generative adversarial nets in robotic chinese calligraphy, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 1104–1110. `doi:10.1109/ICRA.2018.8460787`.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[37] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434.

[38] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein gan, arXiv preprint arXiv:1701.07875.

[39] L. Yu, W. Zhang, J. Wang, Y. Yu, Seqgan: Sequence generative adversarial nets with policy gradient., in: AAAI, 2017, pp. 2852–2858.

[40] E. Walraven, M. T. Spaan, B. Bakker, Traffic flow optimization: A reinforcement learning approach, Engineering Applications of Artificial Intelligence 52 (2016) 203 – 212. `doi:https://doi.org/10.1016/j.engappai.2016.01.001`.
URL `http://www.sciencedirect.com/science/article/pii/S0952197616000038`

[41] Y. Cheng, Y. Huang, B. Pang, W. Zhang, Thermalnet: A deep reinforcement learning-based combustion optimization system for coal-fired boiler, Engineering Applications of Artificial Intelligence 74 (2018) 303 – 311. `doi:https://doi.org/10.1016/j.engappai.2018.07.003`.
URL `http://www.sciencedirect.com/science/article/pii/S0952197618301477`

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347.

[43] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, J. Ba, Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 5279–5288.
URL `http://papers.nips.cc/paper/7112-scalable-trust-region-method-for-deep-reinforcemen` `pdf`

[44] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[45] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971.

[46] M. Martín, A. Jiménez-Martín, A. Mateos, A numerical analysis of allocation strategies for the multi-armed bandit problem under delayed rewards conditions in digital campaign management, Neurocomputing 363 (2019) 99 – 113. `doi:https://doi.org/10.1016/j.neucom.2019.06.052`.
URL `http://www.sciencedirect.com/science/article/pii/S0925231219309373`

[47] M. Martín, A. Jiménez-Martín, A. Mateos, Possibilistic reward methods for the multi-armed bandit problem, Neurocomputing 310 (2018) 201 – 212. `doi:https://doi.org/10.1016/j.neucom.2018.04.078`.
URL `http://www.sciencedirect.com/science/article/pii/S0925231218305630`

[48] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `http://tensorflow.org/`

[50] R. Wu, W. Fang, F. Chao, X. Gao, C. Zhou, L. Yang, C. Lin, C. Shang, Towards deep reinforcement learning based Chinese calligraphy robot, in: 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2018, pp. 507–512. `doi:10.1109/ROBIO.2018.8664813`.

[51] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 6626–6637.