Research Article

Image-Guided Rendering with an Evolutionary Algorithm Based on Cloud Model

Tao Wu

1School of Information Engineering, Lingnan Normal University, Zhanjiang 524048, China
2Guangdong Engineering and Technological Development Center for E-Learning, Zhanjiang 524048, China

Correspondence should be addressed to Tao Wu; wutao@whu.edu.cn

Received 24 July 2017; Revised 27 November 2017; Accepted 15 January 2018; Published 19 February 2018

Academic Editor: Muhammad Sajjad

Copyright © 2018 Tao Wu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The process of creating nonphotorealistic rendering images and animations can be enjoyable if a useful method is involved. We use an evolutionary algorithm to generate painterly styles of images. Given an input image as the reference target, a cloud model-based evolutionary algorithm that will rerender the target image with nonphotorealistic effects is evolved. The resulting animations have an interesting characteristic in which the target slowly emerges from a set of strokes. A number of experiments are performed, as well as visual comparisons, quantitative comparisons, and user studies. The average scores in normalized feature similarity of standard pixel-wise peak signal-to-noise ratio, mean structural similarity, feature similarity, and gradient similarity based metric are 0.486, 0.628, 0.579, and 0.640, respectively. The average scores in normalized aesthetic measures of Benford’s law, fractal dimension, global contrast factor, and Shannon’s entropy are 0.630, 0.397, 0.418, and 0.708, respectively. Compared with those of similar method, the average score of the proposed method, except peak signal-to-noise ratio, is higher by approximately 10%. The results suggest that the proposed method can generate appealing images and animations with different styles by choosing different strokes, and it would inspire graphic designers who may be interested in computer-based evolutionary art.

1. Introduction

Evolutionary design is an area in which artificial evolution is applied to problems in engineering, design, and the fine arts. One active research topic is the generation of images using evolutionary algorithms [1–7]. A specialized application of evolutionary design is the evolution of nonphotorealistic rendering (NPR) effects [8]. Also, it would be beneficial to researchers to develop and test the evolutionary algorithms for creating interesting and aesthetical images and videos. Generally, there have been a relatively large number of methods on evolutionary algorithms for NPR, and a lot of approaches have been proposed with relative success, but most of these methods fall into user-guided and interactive generation. Recently, some genetic programming based methods have been surfaced and received some attentions [9–12]. Various strokes are introduced into the evolutionary system, such as the line, the starburst, and the triangular stroke, and several aesthetic measures are used to calculate fitness values, including luminance direct match [9–11], deviation from normality, color palette matching, and multi-objective scoring [12, 13]. More recently, a multiagent based art production framework is proposed, which generates images using multiagents with chaotic dynamics features [14]. A methodology is applied for evolving image variants to maximize diversity in image feature metrics [15], and a cooperative coevolution strategy based on the Parisian evolution approach is introduced to produce artistic visual effects from an input image [16]. Still, evolutionary design or evolutionary art presents the opportunity to discover interesting variations of well-known visual styles, as well as completely new ones. The problem of evolving images without human-in-the-loop is still an open issue in evolutionary art; sustained effort is required for autonomous evolutionary art.

Furthermore, uncertainty is an inherent part of evolutionary art. It is easy to describe this process in natural language but difficult to simulate it using computer, and the automatic implementation with uncertainty is still a challenge. We believe that, cloud model, proposed by Li et al. [17, 18], can handle such uncertainty in a better way, since it
provides us with more design degrees of freedom and realizes the transformation between a qualitative concept described in words and its numerical representation [19, 20]. Cloud model is a cognitive model between a qualitative concept and its quantitative instantiations and successfully used in various applications. In recent years, cloud model has also been applied in the field of metaheuristics, such as cloud model-based genetic algorithm, cloud model-based evolutionary algorithm, and cloud artificial immune network algorithm [21–24]. By using cloud model, the cloud evolutionary algorithms will have stronger adaptability for the optimization problem and show the perspective of widespread application.

In this context, we propose an image-guided rendering with an evolutionary algorithm based on cloud model (iREC for short), and our intentions are two-fold: (1) How can the image-guided rendering be produced with a cloud model-based evolutionary algorithm? (2) How different types of strokes can affect the aesthetic qualities of the rendered images?

The cloud model-based NPR rendering consists of some steps, including preprocessing an input image by feature extraction, representing the evolution knowledge using cloud model, evolving the individual iteration by iteration and evaluating the fitness function pixel-by-pixel, rendering the canvas on the 2D space provided by the target image, and then generating the result image and animation.

Our method satisfies the CAD property, but what it meant is not the computer aided design, and it stands for cloud model-based evolutionary, automatic evaluation, and dynamic generation: (1) The proposed method involves a novel algorithm using cloud model-based evolutionary, although there have been some efforts in using evolutionary algorithm and genetic programming. (2) The proposed fitness-based reproduction for the evolution is with an uncertain measure by the forward and the backward cloud model generators, and the overall procedure is an automatic evaluation, but not directly by user interaction. (3) From the artist perspective, the focus is on creativity and iterative process, rather than optimization and the best image, and then the proposed method is a dynamic generation by iterations.

In this paper, we make the following contributions. First, we proposed a stroke-based NPR production framework that employs cloud model to achieve the uncertain evolutionary. The algorithm provides the stroke variation of type, color, and orientation. Second, we introduced six types of strokes (both empty and filled) and used the proposed framework to produce artworks. The proposed framework can generate interesting images and animations with the NPR style based on a given input image. Finally, we evaluated the proposed framework with visual comparisons, quantitative comparisons, and user studies to gain insight into the importance and unique affordances of the proposed method. It generally provides the users with the process of generating images, and the immersion effect is also enjoyable.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction of cloud model. In Section 3, we describe our iREC algorithm in detail: first the preprocessing of the image feature extraction, next the rendering algorithm, then cloud model-based representation, and evolutionary engine; the parameter configuration is also provided in this section. In Section 4, we investigate and suggest eight strokes for rendering, as well as two methods (filled and empty strokes), and then show several examples of the resulting images and animations. Finally, we discuss the results and give some ideas for future improvements in Section 5.

2. Cloud Model

Cloud model is the innovation and development of membership function in fuzzy theory and uses probability and mathematical statistics to analyze the uncertainty [18]. In theory, there are several forms of cloud model, successfully used in various applications [17, 18, 25–27]. However, the normal cloud model is commonly used in practice, and its theoretical foundation is the universality of normal distribution and bell membership function [17].

Let $U$ be a universe set described by precise numbers and $C$ be a qualitative concept related to $U$. Given a number $x \in U$, which randomly realizes the concept $C$, $x$ satisfies $x \sim N(Ex, En^2)$, where $En \sim N(En, He^2)$, and the certainty degree of $x$ on $C$ is as below:

$$
\mu(x) = \exp \left( - \frac{(x - Ex)^2}{2En^2} \right).
$$

In (1), the expression $z \sim N(a, b^2)$ denotes that $z$ satisfies the normal distribution with the mean $a$ and the variance $b^2$. Then the distribution of $x$ on $U$ is defined as a normal cloud and $x$ as a cloud drop.

The overall property of a concept $C(Ex, En, He)$ can be represented by three numerical characters of normal cloud model, expected value $Ex$, entropy $En$, and hyperentropy $He$. $Ex$ is the mathematical expectation of the cloud drop distributed in universal set $U$. $En$ is the uncertainty measurement of the qualitative concept, and it is determined by both randomness and fuzziness of the concept. $He$ is the uncertain measurement of entropy, which is determined by randomness and fuzziness of entropy $En$ [17]. It is worth noting that hyperentropy $He$ of a cloud model is a deviation measure from a normal distribution; hence, the distribution of cloud drops can be regarded as a generalized normal distribution.

The kernel of normal cloud model is the transform between qualitative concept and quantitative data, and it is realized by normal cloud generators [17, 26]. On one hand, forward normal cloud generator is the mapping from qualitative concept to quantitative values, and it produces the cloud drops to describe a concept when three numerical characters and the number of cloud drops are input. On the other hand, backward normal cloud generator provides the transformation from quantitative numerical values to quality concept, and a normal cloud model with three numerical characters is defined by computing mean, absolute central moment with the first order, and variance of sample data.

In the normal cloud model, each cloud drop contributes to the concept differently. The Gaussian function satisfies three sigma rule, and as a consequence, normal cloud model has 3En rule. The majority of cloud drops lie within the
Input: The original image and the parameters.

Output: The rendering image and animation.

1. Set initialization parameters, the current image $I_{res}$, the flag buffer $I_{flg}$, and $t = 0$.
2. Pre-process the original image, and obtain $I_{feature}$ (i phase, see Section 3.2).
3. Generate an initial population $(t)$ according to the forward normal cloud generator.
4. repeat
   5. for $j = 1 \rightarrow N_c$ do
     6. Calculate the fitness value of each individual in the $j$th population.
     7. Determine the top $N_e$ elite individuals according to the fitness values of $N_p$ individuals.
      8. for $k = 1 \rightarrow N_e$ do
        9. Draw with the $k$th individual of the $j$th population according to Algorithm 2 (R phase, see Section 3.3).
      10. end for
     11. Fitness-based reproduction (E phase, see Section 3.5), including selection, crossover, mutation models using backward normal cloud generator.
   12. end for
   13. Produce the current image $I_{res}$ as a frame of rendering animation and then $t = t + 1$.
   14. Generate the new population $(t)$ according to the forward normal cloud generator (C phase, see Section 3.4).
15. until Best individual is good enough or stopping criterion is satisfied.
16. Produce the current image $I_{res}$ as the final rendering image.

Algorithm 1: The iREC algorithm.

Input: The stroke $[X_c, Y_c, R, \theta]$, the current image $I_{res}$, the flag buffer $I_{flg}$.
Output: The new image $I_{res}$ and the new flag $I_{flg}$.
(1) Set centre of the brush to $(X_c, Y_c)$.
(2) Find the pixel set PS using the stroke $[X_c, Y_c, R, \theta]$ (see Section 3.3.1).
(3) for Each pixel in PS do
   4. if (the value of this pixel in $I_{flg}$ is not TRUE) then
      5. Update the pixel in the current image $I_{res}$ as Eq. (15).
      6. if (this pixel is acceptable, that is, $\text{diff}(x, y) < \sigma_{diff}$) then
         7. Set the pixel to TRUE in the flag buffer $I_{flg}$.
       8. end if
   9. end if
(10) end for
(11) return the current image $I_{res}$, the flag buffer $I_{flg}$.

Algorithm 2: The rendering algorithm.

interval $[Ex - 3En, Ex + 3En]$, and specifically, cloud drops within the interval $[Ex - 0.67En, Ex + 0.67En]$, called the backbone elements, only account for 22.33% of the universe set but constitute 50% of the cloud model. Then, cloud drops within the interval $[Ex - En, Ex + En]$ make up 33.33% of the universe and constitute 68.26% of the cloud. Similarly, other cases can be also calculated. More information about normal cloud model can be obtained from [17].

3. The Proposed Algorithm: iREC

3.1. An Overview of iREC. Algorithm 1 outlines the evolutionary process used. Given an input image, the proposed method automatically converts it into NPR style by the following steps:

System initialization (steps (1) and (2)) involves setting parameters, and reading in the source and color target images. Filters such as Sobel, moment, and luminosity are applied to the target image, and the filtered results are saved for future use. Some preprocessing is done, in preparation for fitness evaluation later. This is followed in step (3) by the generation of population based on forward cloud generator, to be used as the initial population for the evolutionary process. Steps (4) to (15) are the main evolutionary loop of the system. Each generation has a different population of cloud model-based representation. First, the fitness values of individuals are calculated in step (6). The processing of an individual begins by initializing the canvas. Using a given pixel selection procedure determined by the parameters $X_c$, $Y_c$, $R$, $\theta$, the individual is applied repeatedly to the canvas, resulting in a rendering of the target image. When complete, the canvas is analyzed and evaluated, and the resulting score is assigned as the individual’s fitness value. This is repeated for all population members. For each group of individuals, the top $N_e$ elite individuals are determined according to the fitness values of $N_p$ individuals. For each elite individual, we draw a stroke with the $k$th elite individual of the $j$th population according to Algorithm 2. Once all the population
has been evaluated, the fitness-based reproduction operators are applied to it, including selection, crossover, and mutation. Then, a cloud model-based reproduction is also conducted, and the gene cloud models are updated using backward normal cloud generator. With the incremental generation \( t \), the new individuals are produced according to the forward normal cloud generator, resulting in a new population of individuals. This repeats until best individual is good enough or another stopping criterion is satisfied (see Table 1).

### 3.2. Image Feature Extraction (i Phase)

Image feature extraction is the preprocessing of the proposed framework, and there are many methods, depending on users’ requirement, to solve this problem. In this paper, we use three representative types of feature extraction methods, such as converting color image to grayscale image, edge detection from image, and filtering an image. The result of \( i \) Phase is denoted as \( I_{\text{feature}} \). The example image and the processed images are shown in Figure 1(a). The grayscale image is obtained by the MATLAB function rgb2gray() applied to the color image. The edge image is obtained by the MATLAB function edge() with a Sobel operator applied to the original image. The filtered image is obtained by extracting red channel of object from the original image. The result images are also listed in Figures 1(b)–1(d), and we use these images as the target in the evolution. Of course, the image feature extraction is still an open problem.

### 3.3. The Rendering Algorithm (R Phase)

#### 3.3.1. Strokes for Painting

To render a stroke on the canvas, we introduce an extensible algorithm, whose input parameters include the center of the circumscribed circle of the stroke \((X_c, Y_c)\) and the corresponding circumradius \( R \), related to the location of strokes. A line can be determined by the center \( X_c, Y_c \), the length \( R \), and the orientation \( \theta \) of the stroke. The discrete point set of line stroke is generated by

\[
\begin{align*}
X &= X_c + R \cos(\theta), \\
Y &= Y_c + R \sin(\theta).
\end{align*}
\]

For the reference, Figure 2 gives the diagram of strokes.

#### 2. Crosshair Stroke

It is composed of two perpendicular lines, whose point set is as follows:

\[
\begin{align*}
X_1 &= X_c + R \cos(\theta), \\
X_2 &= X_c + R \cos(\theta + \pi), \\
Y_1 &= Y_c + R \sin(\theta), \\
Y_2 &= Y_c + R \sin(\theta + \pi),
\end{align*}
\]

where \( \alpha = 0 : 2\pi \) denotes the angle of inclination of a line.
Figure 1: The example image and its feature images: (a) the original image, (b) the grayscale image, (c) the edge image, and (d) the image only with red channel.

Figure 2: The diagram of strokes, line stroke, crosshair stroke, petal-type stroke, ellipse stroke, triangle stroke, and square stroke, from top to bottom and left to right.
(3) Petal-Type Stroke. It is like a flower with several petals, and the number of petals is controlled by a random integer \( \delta \in [3, 9] \). The discrete point set is

\[
X = X_c + R \sin^2 (\delta \theta) \cos (\theta),
\]

\[
Y = Y_c + R \omega_R \sin^2 (\delta \theta) \sin (\theta).
\]

(4) Circle and Ellipse Stroke. An ellipse stroke is a generalization of a circle stroke with \( \omega_R = 1 \). The discrete point set of ellipse stroke is generated by

\[
X = X_c + R \cos (\alpha) \cos (\theta) - R \omega_R \sin (\alpha) \sin (\theta),
\]

\[
Y = Y_c + R \cos (\alpha) \sin (\theta) + R \omega_R \cos (\alpha) \cos (\theta),
\]

where \( \alpha = 0 : 2\pi \) denotes the discretely sampled points from the given ellipse.

(5) Triangle Stroke. It is determined by three points \( A, B, C \), whose coordinates are

\[
X_C = X_c + R \cos (\theta),
\]

\[
X_B = X_c + R \cos (\theta + \beta),
\]

\[
X_A = X_c + R \cos (\beta + \alpha),
\]

\[
Y_C = Y_c + R \sin (\theta),
\]

\[
Y_B = Y_c + R \sin (\theta + \beta),
\]

\[
Y_A = Y_c + R \sin (\beta + \alpha),
\]

where \( \beta \in (0, \pi/2) \) and \( \alpha \in (0, \pi) \) denote the random increment related to the vertices of the triangle.

(6) Square and Rectangle Stroke. A rectangle stroke is a generalization of a square stroke with \( \omega_R = 1 \). The discrete point set of rectangle stroke is generated by

\[
X = X_c + d_X \cos (\theta) - d_Y \sin (\theta),
\]

\[
Y = Y_c + d_X \sin (\theta) + d_Y \cos (\theta),
\]

where \( d_X = [0, R, R, 0, 0] \), \( d_Y = [0, 0, R \omega_R, R \omega_R, 0] \) denote coordinate values of four vertexes.

For improving the painting, a guided search of color is used with the reference of the target image, and \( C_r, C_g, C_b, C_a \) are removed from the draw function of the strokes. Each pixel is painted using the corresponding pixel on the original image. This guarantees that pixels are painted only if they promote convergence towards the target. Conversely, a blind search of color is used without reference to the target image. Thus, \( C_r, C_g, C_b, C_a \) should be added into the strokes. In other words, \( C_r, C_g, C_b, C_a \) are involved into the evolution process. Obviously, the latter is a modification or variation of the former, and there is almost nothing changed in the algorithm, only additional four parameters are introduced into the evolution. To simplify the problem and facilitate convergence speed, we use the guided search in the paper, and \( C_r, C_g, C_b, C_a \) are purposely ignored in the evolution process. We represent each individual using a certain number of strokes. Generally, we mathematically define various strokes using a universal expression, and each stroke includes four components as \([X_c, Y_c, R, \theta]\), denoted by \( s \).

3.3.2. Genetic Representation. The canvas is initialized before each evolution process is executed. There are two options for canvas initialization: (1) copy the target image onto the canvas; (2) initialize the canvas to a given RGB color. With the latter option, the target image's pixel data is still accessible to the evolution process, but not on the canvas itself. In our experiments, we start the evolution process from a white canvas, that is, \( I_{res} \).

Each stroke can be considered as one chromosome. For each population, the number of individuals is denoted as \( N_p \), then we plot each stroke and construct an image with various strokes on the canvas. To confine the strokes in a certain range, we define the bounds of coordinate axis with the height \( h \) and the width \( w \), which is determined by the target image. For the purpose of visual aesthetics, the size of strokes is limited by two constant values \( \eta_{min}, \eta_{max} \). That is to say, \( X_c \in (0, w), Y_c \in (0, h) \) and \( R \in (\eta_{min}w, \eta_{max}w), R \omega_R \in (\eta_{min}h, \eta_{max}h) \). From the point of view of mathematical formalization, each individual is formed as \( I = [s_1; s_2; \ldots; s_{N_p}] \), where \( s_i = [X_{ci}, Y_{ci}, R_i, \theta_i] \) is defined on the above.

Since the position coordinates of pixels are integers, we rasterize the strokes using the round() function, as mentioned above. The used rasterization method is mainly due to two points: First, the too larger stroke will result in an inartistic rendering, the stroke is usually fine, the canvas is relatively larger, and the rounding approximation is rough but acceptable. Second, the accurate calculation is more time-consuming. Of course, the other accurate algorithms are also suitable, even more specially, including Bresenham’s line algorithm, the midpoint circle algorithm, and the drawing algorithm of ellipses and Bézier curves, as well as the scanline rendering algorithms for closed regions.

Every selected pixel \( (X_c, Y_c) \) represents the center of a stroke. Therefore, a set of pixels are rendered according to the type of stroke. Each central pixel can result in multiple paint pixels. In addition, it is common for pixels to be rerendered multiple times with the generation, which results in paint mixing effects. The rendering algorithm is described as follows: the current image and the flag buffer are updated from generation to generation. The current image of each generation is as a frame of the final animation. Once the evolution is complete, the current image is as the final result image.

For a selected central pixel \( (x, y) \), the difference between the target image and the temporary image is defined as follows:

\[
diff (x, y) = |I_{feature} (x, y) - I_{res} (x, y)|,
\]

where \( \text{diff} \) is the difference of intensity for gray image, and then the average difference of three channels if a color image is involved.
To encourage the production of images with clear paint strokes and effectively accelerate convergence, some simple pixel selection strategies were implemented. If the difference \( \text{diff}(x, y) \) is lower than a predefined threshold \( \sigma_{\text{diff}} \) (we fix in the experiment), the painted pixel is acceptable. In addition, a Boolean flag buffer is maintained alongside the canvas buffer. It is first initialized to false. When an individual stroke is applied to a canvas, every pixel colored by it is updated within the pixel set of this stroke according to (2) to (13), respectively. The flag therefore keeps a record of unpainted or unacceptable pixels, which can be used for finding locations of subsequent stroke application. The stroke specifies a position on the canvas, and the updated pixels can be drawn using at least three methods; that is, replace the existing pixels using the stroke pixels, mix pixels by a blending formula, and hold to preserve the existing pixels.

In the following experiments, we use the blending method, and the updated pixels are determined by the canvas values and the corresponding existing pixels, defined as follows:

\[
I_{\text{res}} = \omega I_{\text{res}} + (1 - \omega) I_{\text{stroke}},
\]

where \( I_{\text{stroke}} \) denotes a set of pixels related to the current stroke, and we employ \( \omega = 0.3 \).

The example strokes with 100 iterations are shown in Figure 3, which in order are line, crosshair, petal-type, ellipse, triangle, and square strokes, from left to right, from top to bottom. In addition, empty ellipse and empty square as other aesthetic styles, are also listed in Figure 3.

3.4. Cloud Model-Based Evolutionary Knowledge (C Phase).

The overview of evolutionary knowledge based on cloud model is shown in Figure 4. The key idea is gene cloud, which represents each grouping of genes in the community. Each gene cloud corresponds to a component of a set of individuals.

**Gene.** A gene in genetics is a locus or region of DNA that encodes a functional RNA or protein product and is the molecular unit of heredity. In this paper, a gene is as one of the attributions of the stroke, and we use four attributions, \( x_c, y_c, R, \theta \), as shown in Figure 2. Therefore, the genes are arranged in four groups (see Figure 4), with each group of four genes defining a stroke to be drawn on the image. Furthermore, some additional attributions can be also introduced to improve the aesthetics of result images and animations, for example, the gray level for the grayscale images and the number of the petals for the petal-shaped stroke.

**Individual.** In iREC, each grouping of genes is treated as an individual, in which the integers \( X_c \) and \( Y_c \) denote the center of the circumcircle of the stroke, while the floating numbers \( R \) and \( \theta \) define the radius and the skewness of the circumcircle of the stroke.

**Population.** A population is a summation of all the individuals of the same group, which are a part of particular cloud drops, defined by the population gene cloud, as described below. The number of the population individuals in a community is called the population size, denoted by \( N_p \).

**Elite Individual.** Population elite denotes the position of the optimal individual, namely, \( P_{\text{best}} \), and the population elite will
be saved and in the corresponding position a stroke will be drawn in the following rendering phase. The number of elite individuals in a population is called the elite size, denoted by $N_e$.

**Community.** A community is an iterative unit of a fixed size, which consists of several groups of populations connected by interbreeding relations. The number of the population groups in a community is called the community size, denoted by $N_c$.

**Gene Cloud.** Each grouping of genes in the community is represented by a cloud model, namely, gene cloud $g_i (E_x, E_n, H_e)$, where $i$ represents the $i$th gene and $E_x, E_n, H_e$ are three digital characteristics of cloud model. Each gene is produced by a gene cloud, using a forward normal cloud generator [17], during the evolution.

**Population Gene Cloud.** For a community, a set of cloud models is generated to represent the distribution characteristics of the same dimension of all individuals, called population gene cloud $[g_j^1, g_j^2, g_j^3, g_j^4]$, where $j$ represents the $j$th population. That is to say, a community is determined by the population gene clouds, which is a cloud matrix as below:

$$
\begin{bmatrix}
g_1^1 (E_x^1, E_n^1, H_e^1) & \cdots & g_1^i (E_x^i, E_n^i, H_e^i) & \cdots & g_1^4 (E_x^4, E_n^4, H_e^4) \\
\vdots & \cdots & \vdots & \cdots & \vdots \\
g_i^1 (E_x^1, E_n^1, H_e^1) & \cdots & g_i^i (E_x^i, E_n^i, H_e^i) & \cdots & g_i^4 (E_x^4, E_n^4, H_e^4) \\
\vdots & \cdots & \vdots & \cdots & \vdots \\
g_{N_c}^1 (E_x^{N_c}, E_n^{N_c}, H_e^{N_c}) & \cdots & g_{N_c}^i (E_x^{N_c}, E_n^{N_c}, H_e^{N_c}) & \cdots & g_{N_c}^4 (E_x^{4}, E_n^{4}, H_e^{4}) 
\end{bmatrix}
$$

(16)

Individual is the basic unit of evolutionary in the classical methods; in contrast, there is more than one population in our iREC, and the evolutionary unit is the population. An individual gene $v_{ijk}$ denotes the $i$th gene component from the $k$th individual and the $j$th population.

Therefore, considering each individual as a stroke, each component of the cloud model determined by the individuals can be considered as the population gene cloud. That is, the three numerical characters on $X_c$ are calculated by the backward cloud generator from the sequence...
\{X_{ci} \mid i = 1, 2, \ldots, N_p\}. The other components \(Y_c, R, \theta\) are processed in the same way.

3.5. Cloud Model-Based Evolutionary Engine (E Phase)

**Fitness.** Fitness function denotes adaptation degree of each individual aimed at their located environment in the community. It is used to evaluate the individual and decide which individual to retain or eliminate. In this paper, fitness function is the expression of the difference between the target image and the temporary image and is defined as follows:

\[
diff_i = \sum_{x \in [1, w]} \sum_{y \in [1, h]} \text{diff}(x, y),
\]

\[
f = \frac{1}{1 + \text{diff}_i},
\]

where \(\text{diff}_i\) is the sum of the difference pixel-by-pixel according to (14).

**Initialization.** The population size depends on the nature of the problem but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). In our method, the initial individuals are generated by the forward normal cloud generator, whose numerical characters are determined randomly.

**Selection.** In this stage, potentially useful solutions are selected for recombination. In iREC, the selection includes population selection and individual selection. Firstly, the fitness function is evaluated for each individual in each population, providing fitness values, which are then normalized. Normalization means dividing the fitness value of each individual by the sum of all fitness values, so that the sum of all resulting fitness values equals 1:

\[
\text{fitness} = \frac{f - f_{\text{worst}}}{f_{\text{best}} - f_{\text{worst}}},
\]

where \(f\) is the fitness value of the processed individual and \(f_{\text{worst}}\) and \(f_{\text{best}}\) denote the worst and the best fitness values in the population.

For individual selection, we use roulette wheel selection. The population is sorted by descending fitness values. Accumulated normalized fitness values are computed; the accumulated fitness value of an individual is the sum of its own fitness value plus the fitness values of all the previous individuals. A random number \(N_r = \text{rand}(1)\) between 0 and 1 is chosen. The selected individual is the first one whose accumulated normalized value is greater than \(N_r\).

There are two strategies of population selection. Population unified selection strategy is to select the elite individuals from all the populations and then assign them to each population as the seeds. Meanwhile, population independent selection strategy is the selection of elite individuals from each population, which are as the seeds in the next generation. The convergence rate of the former strategy is faster, while the latter one can well ensure the diversity of individuals. According to the needs of the aesthetic requirements, we used the population independent selection strategy.

**Crossover.** It is a process of taking more than one parent solution and producing a child solution from them. In iREC, crossover operation can only occur between two individuals from different populations. That is, the operator essentially acts on the population, but not individual. Crossover only occurs at a particular mating according to a predefined crossover probability \(P_c\). There are many crossover techniques, and we use the arithmetic mean value of the parent gene:

\[
v_{jk}' = \alpha_v v_{jk} + (1 - \alpha_v) v_{jk}^{r'},
\]

\[
v_{jk}^{r'} = \alpha_v v_{jk}^{l'} + (1 - \alpha_v) v_{jk}^{l'},
\]

where \(\alpha_v \in (0, 1)\) is a random number, namely, the crossover coefficient.

**Mutation.** This alters one or more gene values in a chromosome from its initial state. The selection rule remains the same for both crossover and mutation. Mutation only occurs according to a predefined mutation probability \(P_m\), and this probability should be set low:

\[
v_{jk}' = \left\{ \begin{array}{ll}
v_{jk} + (u_{jk} - v_{jk}) \beta_m \mod 2 = 1, \\
v_{jk} - (v_{jk} - i_{jk}) \beta_m \mod 2 = 1,
\end{array} \right.
\]

where \(G_m\) is the maximum generation and \(\beta_m\) is a random integer, namely, the mutation coefficient.

**Update.** In this operator, all the individuals are removed, the numerical characteristics of each gene cloud model are calculated by the backward cloud generator and then the individuals of the new generation are reproduced by population gene cloud. Compared with the traditional evolution operators, our method involves a cloud generator between qualitative concept and quantitative data.

**Pixel Ratio.** We refer to the proportion of the matched image pixels, whose value is true in the flag buffer \(f_{\text{flg}}\). The mathematical expression of the matched pixels can be

\[
\frac{\sum_{x \in [1, w]} \sum_{y \in [1, h]} f_{\text{flg}}(x, y)}{hw} \times 100\%.
\]

Figures 5(a)–5(f) show a sequence from an animated movie showing the target with a series of frames of the subject gradually emerging, as well as the enlarged portion marked with blue rectangle. The system starts with a white canvas, and more and more strokes are placed onto the canvas; then an aesthetic image is generated iteration by iteration. During the evolution, the flower is more and more clear, and the number of white pixels becomes lower and lower. For the reference, the video result (MP4 format) is attached in the Supplementary Materials available online (available here).
The intensity difference between the target image and the result image is listed in Figure 5(g). There are a lot of black pixels in the difference image. In other words, the final result is rather closer to the target image. Only pixels around the edge appear white and need to be improved. In fact, this is just the charm and interesting part of the proposed method, since the purpose of our method is to provide a NPR result, rather than the original image. Figure 5(h) shows the progression of fitness values, the proportion of matched pixels, and the intensity difference during the course of the evaluations for one run. Both of the curves of fitness values and matched pixels produce similar tendency with the various generations. The corresponding value is lower for an earlier generation and achieves a smooth increase with the growth of generation, but the growth slope of the matched pixels’ curve is more shapely. Contrastively, the value of intensity difference declines gradually if the number of iterations becomes more and more.

3.6. Parameter Configuration. Table 1 specifies the parameters used to conduct the experiments for evolving art images. Certainly, it is still an open question how to
choose the evolutionary parameters. In the evolutionary art, no human evaluation or interactive evolution is involved.

We list 12 parameters in Table 1, which is divided into four groups, with each group consisting of three parameters. For individual strategy, population size \( N_p \), community size \( N_c \), and elite ratio \( N_e \) are introduced in Section 3.4. For evolution strategy, selection strategy, crossover probability \( P_c \), and mutation probability \( P_m \) are mentioned in Section 3.5. For update strategy, pixel update method, minimum and maximum circumradius \( \eta_{\text{min}}, \eta_{\text{max}} \) are explained in Section 3.3.2. With a large circumradius, the texture is more rough; otherwise, it is fine and smooth. However, too large circumradius should lead to boring feeling, while too small circumradius indicates more white pixels, as shown in Figure 7, and we will further discuss this point. Our evolution process is repeated until a termination condition has been reached. For termination strategy, the terminating conditions include three components. \( G_m \) is the maximum generation of the evolution; that is, the evolution is stopped when \( t > G_m \). \( T_P \) is the threshold of pixel ratio, and the system is also terminated when a fixed number of pixels are acceptable. In addition, fitness difference threshold \( T_f \) denotes the difference between the highest ranking individual’s fitness values of two successive generations, which indicates that the result is reaching or has reached a plateau. In other words, if any one of the above three conditions is satisfied, the proposed algorithm is terminated.

4. Experimental Results

4.1. Experimental Setup. In this section, several experiments are presented to examine the performance of the proposed technique. We have conducted our experiments on a PC platform with an Intel 2.5 GHz Core i7-6500U CPU and 8 GB RAM, running on a Windows 10 operating system. We describe the experiments in evolving art using various measures and will answer our research questions of this paper by reporting various experimental results with a brief discussion.

4.2. Results

4.2.1. Various Strokes. The different shapes of strokes can enrich the visual outcome of the proposed system. We investigate six types of strokes according to Algorithm 2 and generate six sample images, including the line, crosshair, petal-type, triangle, square, and ellipse strokes. To make a deep observation, the enlarged details are also attached in Figure 6. Clearly, various strokes would show various styles of result images, and different types of strokes affect the aesthetic qualities of the outputs, which can be potentially applied to various fields. From the perspective of the visual results, the image by the filled square stroke is the most pleasing, since there is no break and no space in the corresponding output. For the reference, six video results (MP4 format) are attached in the Supplementary Materials available online (available here).
4.2.2. Filled Stroke and Empty Stroke. In this subsection, we compare both filled and empty strokes with the shape of square and ellipse to find which one generates more aesthetic pleasing images. Figure 7 shows a comparison of unfilled square and ellipse strokes for the same target image, compared with those by the filled ones in Figure 6. As shown in Figure 7(a), there exist some white pixels or pieces on the canvas, which have not been hit by the empty strokes during the whole evolution, while all of the result pixels have been covered by the filled strokes. At least, one cannot see white pixels in Figure 6 with the naked eye. From a visual perspective, we clearly find the filled strokes achieve better performance and generate more aesthetic images than empty ones. For the reference, two video results (MP4 format) are attached in the Supplementary Materials available online (available here). With the same number of strokes, the filled one obviously covers more pixels; thus white pixels rarely happen. However, this does not mean that the empty stroke will generate bad results. We believe that more individuals can solve this problem by increasing the population size or the community size. However, this should indicate more time costs and more memory spaces, so filled stroke is still recommended considering the performance.

Furthermore, we record the fitness values and the matched image pixels by various strokes within the evolution. For the randomness of the proposed method, we run 50 times for each image and average the results. As shown in Figure 8, we show the progression of fitness values and the matched image pixels during the course of the evaluations. All of the strokes show similar results. For each stroke, both of the curves of fitness values and matched pixels also produce similar tendency with the various generations. With an earlier generation, the fitness values and the matched pixels are very smaller and then increase with the growth of generation. More specifically, the change of the fitness values is more obvious. Intuitively, high fitness values indicate closer resemblance to the target and more matched pixels. In other words, with the growth of generation, the result images gradually reveal the target. Correspondingly, the number of the matched pixels becomes greater and greater. From another perspective, we compare the results by various strokes. Unsurprisingly, although it is subjective to say it, the
Figure 9: The visual results by various methods: (a) the original image and a subregion marked with blue, the result images by (b) the GIMP method, and (d) our method. (c) An enlarged portion of Figure 9(b) corresponding to the subregion marked with blue in Figure 9(a), (e) an enlarged portion of Figure 9(d) corresponding to the subregion marked with blue in Figure 9(a).

result images by filled square strokes, at least in our eyes, have better fitness and converge towards a solution faster than the others in the course of iterations. This is because that drawing more pixels in a filled stroke gives more opportunity to be closer to the target. In fact, the covered area by the filled square stroke is the biggest with the same length $R$ in Section 3.3.1, followed by that of the petal-type stroke.

4.2.3. Visual Comparison. We first examine how the proposed processing technique can benefit the NPR system. Considering that the filled square stroke seems beneficial to fitness scores and the convergence of the evolution, we have experimented with different images using filled square stroke and conducted several runs for each image. As shown in Figure 9, another eight input images of dramatically
Different scenes (portrait, landscape, seascape, and river land) are chosen for the demonstration out of several runs with different images. Note that to make the paper more compact, we have chosen to shrink all resulting images. However, this should not damage the image quality as each image is associated with a high resolution. Therefore any reader who is interested in seeing more details from an image should always enlarge the corresponding portion.

To show that our rendering technique has indeed good performance, we compare our results with those generated by GNU Image Manipulation Program (GIMP for short) (https://www.gimp.org/), which is a popular, free, and open-source raster graphics editor used for image retouching and editing and more specialized tasks.

The visual comparison results are shown in Figure 9. To make a deep observation, the enlarged details are also listed in Figure 9. For the reference, eight video results (MP4 format) are attached in the Supplementary Materials available online (available here). For these eight images, the rendering results by GIMP method and ours are both visually acceptable. Comparing each result image with the corresponding target, there is the very close visual effect between them, especially when one looks at them from a distance. However, in detail, the result images are covered by one rectangle stroke after another, and these rectangles are located on the canvas with a certain angle, which seem desultory but actually methodical. In other words, a cubism-like style is achieved by the proposed method. From the perspective of visual details, the results by GIMP method show some holes, while our results do not show this flaw, even though we observe them with an enlarge view. The results indicate that our method can be considered as an alternative to the GIMP method.

**4.2.4. Quantitative Comparison.** Furthermore, for these eight images, we record the fitness values and the matched image pixels within the evolution. For the randomness of the proposed method, we also run 50 times for each image and average the results. As shown in Figure 10, we show the progression of fitness values and the matched image pixels during the course of the evaluations. All of these images show similar tendency with the growth of generation. For each image, both of the curves of fitness values and matched pixels grow incrementally as the generation increased one by one, and the increasing degree of the earlier generations is higher than that of the latter. In other words, the cloud model-based evolutionary algorithm is able to find, in all runs and for all images, individuals with high fitness in relatively few generations. In the fitness values, image 4 comes first, and image 3 runs with the slowest convergence. That is to say, the result of Figure 10(a) indicates that the result image 4 is the closest resemblance to the original target and that 3 is the farthest. For the matched pixels, image 5 predominantly achieves the best result, while image 1 always ranks the last. Surely, the image size and the predefined parameters also have an effect on the scores of the matched pixels.

To measure the robustness of the considered methods, we study quantitative scores of the rendering outputs with various images. Table 2 shows the average response with respect to eight images by using four measures on feature similarity, including the standard pixel-wise peak signal-to-noise ratio (PSNR), mean structural similarity measure (MSSIM) [28], feature similarity (FSIM) [29], and gradient similarity based metric (GSM) [30]. These metrics aim to use computational models to measure the image quality consistently with subjective evaluations, with a full reference.

**Figure 10:** The quantitative results by various images in the evolution: (a) the fitness values, and (b) the matched image pixels.
Table 2: Quantitative evaluation results.

<table>
<thead>
<tr>
<th>Images</th>
<th>Methods</th>
<th>PSNR</th>
<th>MSSIM</th>
<th>FSIM</th>
<th>GSM</th>
<th>BL</th>
<th>FD</th>
<th>GCF</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number 1</td>
<td>GIMP</td>
<td>37.515</td>
<td>0.815</td>
<td>0.876</td>
<td>0.988</td>
<td>0.958</td>
<td>0.767</td>
<td>3.155</td>
<td>7.689</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>35.847</td>
<td>0.878</td>
<td>0.909</td>
<td>0.992</td>
<td>0.965</td>
<td>0.789</td>
<td>3.049</td>
<td>7.518</td>
</tr>
<tr>
<td>Number 2</td>
<td>GIMP</td>
<td>36.723</td>
<td>0.820</td>
<td>0.887</td>
<td>0.987</td>
<td>0.988</td>
<td>0.613</td>
<td>4.036</td>
<td>7.693</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>35.504</td>
<td>0.876</td>
<td>0.913</td>
<td>0.990</td>
<td>0.984</td>
<td>0.638</td>
<td>4.032</td>
<td>7.599</td>
</tr>
<tr>
<td>Number 3</td>
<td>GIMP</td>
<td>37.027</td>
<td>0.773</td>
<td>0.870</td>
<td>0.986</td>
<td>0.985</td>
<td>0.531</td>
<td>5.209</td>
<td>7.893</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>33.924</td>
<td>0.772</td>
<td>0.863</td>
<td>0.986</td>
<td>0.984</td>
<td>0.503</td>
<td>5.444</td>
<td>7.859</td>
</tr>
<tr>
<td>Number 4</td>
<td>GIMP</td>
<td>36.723</td>
<td>0.825</td>
<td>0.878</td>
<td>0.986</td>
<td>0.954</td>
<td>0.613</td>
<td>4.654</td>
<td>7.777</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>33.040</td>
<td>0.871</td>
<td>0.913</td>
<td>0.988</td>
<td>0.976</td>
<td>0.627</td>
<td>4.613</td>
<td>7.681</td>
</tr>
<tr>
<td>Number 5</td>
<td>GIMP</td>
<td>35.654</td>
<td>0.810</td>
<td>0.863</td>
<td>0.988</td>
<td>0.997</td>
<td>0.546</td>
<td>4.817</td>
<td>7.206</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>34.337</td>
<td>0.886</td>
<td>0.923</td>
<td>0.992</td>
<td>0.982</td>
<td>0.488</td>
<td>4.722</td>
<td>7.066</td>
</tr>
<tr>
<td>Number 6</td>
<td>GIMP</td>
<td>35.471</td>
<td>0.831</td>
<td>0.892</td>
<td>0.987</td>
<td>0.980</td>
<td>0.419</td>
<td>5.734</td>
<td>7.817</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>33.540</td>
<td>0.847</td>
<td>0.893</td>
<td>0.987</td>
<td>0.984</td>
<td>0.449</td>
<td>5.738</td>
<td>7.837</td>
</tr>
<tr>
<td>Number 7</td>
<td>GIMP</td>
<td>31.788</td>
<td>0.738</td>
<td>0.826</td>
<td>0.976</td>
<td>0.977</td>
<td>0.443</td>
<td>8.119</td>
<td>7.849</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>30.223</td>
<td>0.751</td>
<td>0.818</td>
<td>0.975</td>
<td>0.977</td>
<td>0.444</td>
<td>8.407</td>
<td>7.848</td>
</tr>
<tr>
<td>Number 8</td>
<td>GIMP</td>
<td>33.396</td>
<td>0.742</td>
<td>0.856</td>
<td>0.982</td>
<td>0.946</td>
<td>0.534</td>
<td>6.247</td>
<td>7.905</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>29.099</td>
<td>0.724</td>
<td>0.824</td>
<td>0.979</td>
<td>0.947</td>
<td>0.588</td>
<td>6.314</td>
<td>7.875</td>
</tr>
<tr>
<td>Average</td>
<td>GIMP</td>
<td>0.731</td>
<td>0.434</td>
<td>0.483</td>
<td>0.569</td>
<td>0.585</td>
<td>0.348</td>
<td>0.410</td>
<td>0.790</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>0.486</td>
<td>0.628</td>
<td>0.579</td>
<td>0.640</td>
<td>0.630</td>
<td>0.397</td>
<td>0.418</td>
<td>0.708</td>
</tr>
</tbody>
</table>

image, which is the original input image in this paper, and the distorted image refers to the stylized result.

PSNR index is a traditional image quality assessment metric, and we use it to calculate the peak signal-to-noise ratio for the stylized image with the original image. For PSNR measure, both the GIMP method and ours obtain the highest scores for image 1 as listed in Table 2, but it is noted that image 1 gains the least matched pixels in Figure 10(b). MSSIM index is a dominant method of measuring image quality from perceived change in structural information. For MSSIM measure, the GIMP method receives the highest scores for image 6, but ours for 5, which also ranks the first from the perspective of matched pixels, as shown in Figure 10(b). FSIM index is based on the fact that human visual system understands an image mainly according to its low-level features. For FSIM measure, the best results are still generated by images 6 and 5 using the GIMP method and our method, respectively. GSM index calculates luminance and contrast-structural changes with emphasis on gradient similarity. For GSM measure, most of images produce the higher scores. It means that the two methods well consider the gradients of original images, which convey very important visual information.

On the other hand, the aesthetic performance is a key problem to practical engineering applications of evolutionary art, and it is desired that the evolved images can be generated with a good visual effect. Although the aesthetics problem has a strong subjectivity and the computational aesthetics is one of the most challenging in image processing and computer graphics, we still introduce four common measures as a quantitative evaluation. In this subsection, the aesthetic measures used in our experiments include Benford’s law (BL) [31], fractal dimension (FD) [32], global contrast factor (GCF) [33], and Shannon’s Entropy (SE) [34]. For full details on these measures, readers are referred to the original papers [31–34]. The results using these four aesthetic measures are also listed in Table 2.

We use Benford’s law to measure the distribution of intensity of pixels in result images. BL index calculates the difference between the actual histogram and the Benford histogram. For BL measure, the GIMP method and our method produce similar scores, and both receive a higher value for image 2. FD index reflects the aesthetic preference of people for natural, artificial, and man-made fractals. Images with a higher fractal dimension are considered complex, and those with a lower dimension should be uninteresting. For FD measure, image 1 achieves the highest score using both the GIMP method and ours. GCF index calculates the contrast on various resolutions of an image, and a low value reflects a lower aesthetic effect. For GCF measure, the two methods gain a higher score for image 7. SE index is also a measure on the intensity of the pixels, and an image will score high on SE if its intensity values are distributed in a uniform way. For SE measure, image 8 always keeps the first rank using the related two methods.

For comparison purposes, each column of the score of each measure is normalized, and then we average the results to provide an overall evaluation. On the whole, our method generates the lower score of the average PSNR. However, it is well known that PSNR does not always agree with the subjective viewing results, particularly for the evaluation of NPR [35]. In addition, our method obtains the smaller average SE value. In other words, our method generates the result images with the intensity of a non-uniform distribution. For the other six measures, our method is always superior to the GIMP method. To some extent, it is indicated that our
method generates better results than the GIMP method, in a couple of respects, that is, image feature similarity and aesthetic performance.

4.3. User Evaluation. Unlike photorealistic rendering that can be easy measured by the close degree between the obtained image and the real photo, nonphotorealistic rendering technique is inspired by artistic styles and pays more attention to subjective feelings. For result artworks, different people may have different opinions depending on their artistic quality. So far, each objective index in Table 2 is only a possible reference for measuring the rendering effect of the NPR technique. Therefore, subjective evaluation is a more important complement to assess the nonphotorealistic rendering effect, as well as the assessment of the proposed method.

The objective of the user study is to see if there is a good performance of the proposed method. In the first study, nine testing images are mentioned above and their corresponding rendered results by GIMP and the proposed method are used to perform subjective evaluation. We asked 53 subjects to score each image on the basis of three criteria (image naturalness, image colorfulness, and visual comfort [36, 37]), and these subjects include 48 voluntary undergraduate students with some art knowledge and 5 professional artists. Each score is used a Likert scale in the range [1 10], and 10 means a perfect result.

The result animations by the proposed method are involved into the second study, whose objective is to investigate the performance of the generation process. The second study asked the subjects to score each animation, using a formal protocol designed to answer the following questions: (1) Fidelity. How do you evaluate the fidelity and intuitiveness of these animations? (2) Novelty. Do you find the novel features of them? (3) Interest. Can you gain something interesting from them? (4) Usability. Will you apply them effectively to your future needs? (5) Stability. Is there evidence that they show a uniform style? Each score is also in the range [1 10], and 10 means a perfect result.

Figure 11 shows a summary of the survey results. It can be seen from Figure 11(a) that users generally agree that the proposed method does improve considerably the performance compared with the GIMP method and produces better visual effects. Visual comfort refers to the subjective sensation of comfort [37]. Users rated the visual comfort of the proposed method as 7.6 on a scale of 1 (not comfort) to 10 (very comfort). Image colorfulness presents the degree of color vividness [36]. A rendered image of good quality should have vivid image color. Most of users felt that the proposed method closely mimics the feeling of real artists, while the GIMP method is characterized by artificial footprints. Image
naturalness discovers the correspondence degree between human perception and real world [36]. A rendered image of good quality should have natural image color. Our results also obtained better scores. This was due to the fact that the proposed method is based on the reference of the target image.

It can be seen from Figure II(b) that most of users confirm the performance of the proposed animations. Users commented that they liked the novel look and feel of result animations and would introduce the style into their needs, because of average usability rating 7.2 and 6.9, respectively. In particular, one user commented that the ability to slowly emerge with strokes is the most interesting part. However, as shown in Figure II(b), the professional artists are generally more critical for the rendering results, giving somewhat lower scores, especially for the fidelity and the stability. In addition, as one user reported, "it has yet to achieve the effect of real painting," and we believe that this is because the current implementation of the proposed method does not model the type of strokes in evolution population. In fact, an artist never uses only one stroke to paint. Thus, we modify step (9) of Algorithm 1 and randomly select one type of strokes defined in Section 3.3.1, totally 8 strokes, including line stroke, crosshair stroke, petal-type stroke, triangle stroke, and ellipse and rectangle stroke, both empty and filled. The implementation is easy, and only a switch parameter is used to control this function. Then we generate two sample result images as shown in Figures II(c) and II(d). One can observe that some new visual effects can be achieved. Compared with the pictures in Figure 9, the texture is more abundant and more complex since we used the mixed strokes to model the style of a real artist. Anyway, another possible fun can be provided by the proposed method. According to the user study, it is also noted that the proposed method is more suitable as a technique to serve the general public, rather than a professional art tool. In general, we believe that the approach can be a good alternative to GIMP for novice users, to convert their photos into NPR style, though it is not very powerful in its current state.

4.4. Discussions. In this subsection, we provide a brief discussion on the proposal. Relating to research question (1) in Section 1, we propose a framework for the image-guided rendering using a cloud model-based evolutionary algorithm, which can be divided into four parts, including image feature extraction (iPhase), strokes-based genetic representation and rendering (RPhase), cloud model-based evolutionary knowledge (CPhase), and cloud model-based evolutionary engine (EPhase). For the reference of the proposed framework, a recommended parameter configuration is also proposed. Our method involves a novel algorithm using cloud model-based evolutionary. In addition, the proposed fitness reproduction is an uncertain measure using the forward and backward normal cloud generator, and the overall procedure is an automatic evaluation, but not directly by user interaction. Relating to research question (2), we provide six different types of strokes in Section 3.3.1 and investigate the aesthetic qualities of the rendered result images. The experimental results show that the different shapes of strokes can enrich the visual outcome of the proposed system, and the filled square strokes achieve better performance and generate more aesthetic images from the perspective of the visual results. Furthermore, the temporary image for each generation produces a dynamic result by iterations, which is a much better fit to the user's needs. In addition, the proposed method has an immersion effect because it can provide the users with the process of generating images, that is the animations.

5. Summary and Conclusion

In the paper, we propose an algorithm of evolving art using cloud model. Furthermore, we introduce six types of strokes into the proposed framework of evolutionary art, both filled and empty. The proposed framework can generate interesting images and animations with the NPR style based on a given input image. The resulting animations have an interesting characteristic in which the target slowly emerges from a set of strokes. The experiment results, including visual comparisons, quantitative comparisons, and user studies, verify the efficiency and feasibility of the proposal, which can be considered as an alternative for novice users to the existing methods, to convert their photos into NPR style, for example, the GIMP method.

There are a couple of issues that will be considered in the future research: (1) As mentioned by [4], introducing other fitness functions to explore the new visual effect of evolved images, especially for more pleasuring texture, is currently under investigation and will be reported later; (2) More alternative evolutionary algorithms based on cloud model may be provided just as or more interesting than result images; thus the extension of the proposed method to improve the performance is well worth further studying on, for example, introducing computational ecosystems [38].

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by the Natural Science Foundation of Guangdong, China (no. 2017A030307030), the National Natural Science Foundation of China (under Grant no. 61402399), Foundation for Distinguished Young Teachers in Higher Education of Guangdong, China (no. YQ2014117), and Science and Technology Planning Project of Guangdong, China (no. 2014A020220012).

Supplementary Materials

Supplementary Data on Video Results: supplementary data associated with this article can be found online at the following site: https://ldr.vr.ms/fs/AjY7nx61oejrqb5Mo-gLaTVUUA. In the Supplementary Materials, we provide four “7-zip” archiver files, which include various video results with MP4 format, mentioned in Figures 5, 6, 7, and 9, respectively. (Supplementary Materials)
References


[33] K. Matkovic, L. Neumann, A. Neumann et al., “Global contrast factor—a new approach to image contrast in,” in Proceedings of


