IJDAR (2004)
Digital Object Identifier (DOI) 10.1007/s10032-004-0131-6

International Journal on IJDAR Document Analysis and Recognition

Combining shape and physical models for online cursive handwriting synthesis*

Jue Wang[2], **Chenyu Wu**[3], **Ying-Qing Xu**[1], **Heung-Yeung Shum**[1]

[1] Microsoft Research Asia, Sigma Center, Zhichun RD, Beijing 100080, China
[2] Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA
[3] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Received: 26 April 2003 / Accepted: 27 September 2004
Published online: ♣ 2004 – © Springer-Verlag 2004

Abstract. This paper proposes a novel learning-based approach to synthesizing cursive handwriting of a user's personal handwriting style by combining shape and physical models. In the training process, some sample paragraphs written by a user are collected and these cursive handwriting samples are segmented into individual characters by using a two-level writer-independent segmentation algorithm. Samples for each letter are then aligned and trained using shape models. In the synthesis process, a delta log-normal model based conditional sampling algorithm is proposed to produce smooth and natural cursive handwriting of the user's style from models.

Keywords: Handwriting synthesis – Cursive script – Handwriting segmentation – Delta log-normal model – Conditional sampling

1 Introduction

Pen-based interfaces are now a hotspot in human-computer interface (HCI) research because in many situations, a pen together with a notepad is more convenient than a keyboard or a mouse. The flourish of pen-based devices brings a great demand for various handwriting computing techniques. Hierarchically speaking, these demands can be categorized into two levels. The first level is handwriting recognition; techniques in this level make it possible for computers to understand the information involved in handwriting. The second level is handwriting modulation, for instance, handwriting editing, error correction, script searching, etc. These applications have gained much attention recently due to their increasing importance on newly developed pen-based devices such as PDAs and Tablet PCs. Suppose that when writing a note on a Tablet PC the computer could automatically correct some written errors and generate some predefined handwriting sentences of the user's writing style; this would make the writing process more effective. Furthermore, handwriting is preferable to typed text in some cases because it adds a personal touch to the communication. Most of these applications bring an urgent requirement for handwriting synthesis techniques.

The problem of handwriting modeling and synthesis has been addressed for a long time, and there are many related studies in the literature. Generally speaking, these approaches can be divided into two categories according to their principles: movement-simulation techniques and shape-simulation methods. Movement-simulation approaches are based on motor models [1] and try to model the process of handwriting production. In [2] and [3], modulation models were used to model and represent the velocity of handwriting trajectory. A delta log-normal model [4] was exploited in [5] for handwritten script compression. However, these studies were mainly focused on the representation and analysis of real handwriting signals rather than handwriting synthesis. Extensive studies on motor models can give some insight but cannot directly provide a solution to the synthesis of a novel handwriting style, especially with respect to cursive writing.

Shape-simulation techniques only consider the static shape of handwriting trajectory. They are more practical than movement-simulation techniques when the dynamic information of handwriting samples is not available and the trajectory has been resampled by other processors, such as recognizers, as addressed in [6]. A straightforward approach is proposed in [7], where handwriting is synthesized from collected handwritten glyphs. Each glyph is a handwriting sample of two or three letters. When synthesizing a long word, this approach simply juxtaposes several glyphs in sequence and does not connect them to generate fluent handwriting. In [8], the problem of learning personal handwriting style is addressed and a learning-based cursive handwriting synthesis approach is proposed. Handwriting trajectories are modeled by landmark-based splines, and the problem

* Jue Wang and Chenyu Wu completed this work while interns at Microsoft Research Asia.
Correspondence to: Jue Wang
(e-mail: juew@u.washington.edu)

of learning personal handwriting style is transformed to statistically analyze the distribution of landmark points. However, this approach cannot scale up to larger numbers of allographs since splines are not optimal for modeling the abundant straight lines and points of high curvature in mixed-style handwriting. Another issue of this approach is that it tries to learn a separate model for the connection of each pair of letters, and this is impractical since the size of the training set is always limited.

To obtain more satisfying handwriting synthesis results, we first propose two constraints that a successful handwriting synthesis algorithm should meet:

1. The shapes of letters in the synthesized cursive handwriting should be close to the shapes of those in the training samples. In this way the personal handwriting style can be preserved.
2. The connection between synthesized letters should be smooth and natural to produce fluent cursive script.

To achieve these objectives, we combine the advantages of the shape-simulation and the movement-simulation methods to propose a novel cursive handwriting synthesis technique. Handwriting samples are collected and segmented into individual characters using a two-level writer-independent segmentation algorithm. Like widely used statistical shape models [9,10], handwriting strokes of the same character are automatically matched and aligned first, and the shape variation of the training set is then learned by PCA models. To produce fluent handwriting, we exploit the delta log-normal model and propose a conditional sampling method, which has the ability to selectively generate novel letters from models and connect them smoothly.

The remainder of this paper is organized as follows. Section 2 presents our methods for training data preparation and segmentation. Section 3 describes the handwriting stroke alignment and training algorithms. Section 4 presents the conditional sampling technique in the synthesis strategies. Some experimental results are shown in Sect. 5, and discussions and conclusions are presented in Sect. 6.

2 Sample collection and segmentation

2.1 Sample collection

For an initial user, he/she is required to write some sample paragraphs for the training purpose, on a newly developed Tablet PC which incorporates an integral display. There are about 200 words in these paragraphs and each letter has appeared more than five times. In the preprocessing step, these handwriting samples firstly pass through a low pass filter and then be re-sampled to produce equi-distant points. Since we will train a separate model for each character, handwritten letters will be extracted from these cursive scripts, using the proposed cursive handwriting segmentation technique.

2.2 Sample segmentation

2.2.1 Overview. In general, there are three categories of methods for handwriting segmentation [11]. One is the *segmentation-based recognition* method, which uses local hints, such as the points of maximum curvature, to over-segment the handwriting trajectories. Then a recognition engine analyzes these segments and estimates which segments should be grouped to output a character. Another approach, called *recognition-based segmentation*, is devised in an inverse manner in which segmentation points are given by the recognizer after recognition. These two methods rely heavily on the performance of the recognition engine. The third method is *level-building* [12], which simultaneously outputs the recognition and segmentation results.

In our case, although the paragraphs written by the user are known, the recognition problem still exists, since the user may write several strokes for a word and the content of each stroke is unknown. To avoid propagating recognition errors to segmentation, we adopt the framework of level-building for handwriting segmentation, in which segmentation and recognition are merged to give an optimal result.

As agreed by most researchers, it is impossible to achieve a correct ratio of 100% for handwriting recognition and segmentation. Given the fact that there is always a severe problem of erroneous labels being attached to handwriting trajectories, some interactive scenarios have also been employed in the system. When the user is writing samples, the segmentation points will be automatically computed and displayed on the handwriting trajectories of finished words and the user enabled for manual adjustment. Given the promising segmentation result of the proposed algorithm, the chances that manual adjustment will be required are dramatically diminished.

2.2.2 A two-level framework. First we give a formal description of the framework of traditional handwriting segmentation approaches. Let $S = \{z_1, \ldots, z_T\}$ be a temporal handwriting sequence, where z_t is a low-level feature that denotes the coordinate and velocity of the sequence at time t. The segmentation problem is to find an identity string $\{I_1, \ldots, I_n\}$, with the corresponding segments of the sequence $\{S_1, \ldots, S_n\}, S_1 = \{z_1, \ldots, z_{t_1}\}, \ldots, S_n = \{z_{t_{n-1}+1}, \ldots, z_T\}$, that best explain the sequence

$$\begin{aligned}&\{I_1^*, \ldots, I_n^*\}\\&= arg\max p(S_1, \ldots, S_n|I_1, \ldots, I_n)p(I_1, \ldots, I_n)\\&= arg\max p(I_1)p(S_1|I_1)\prod_{i=2}^{n} p(S_i|I_i)p(I_i|I_{i-1}),\end{aligned} \quad (1)$$

where it is assumed that each segment S_i is conditionally independent of other variables given the corresponding identity I_i. Usually HMMs or neural networks are trained to obtain the likelihood $p(S_i|I_i)$. We also implemented HMM-based segmentation algorithm under this framework for testing and comparing.

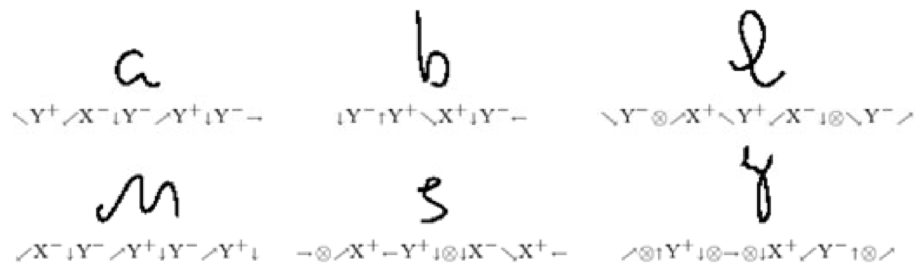


Fig. 1. Script codes of some handwriting characters

For the training of the writer-independent segmentation system, a large handwriting database has been built, which includes about 100,000 words written by more than 100 people. In the experiments, the low-level feature-based segmentation algorithm works well for a small number of writers. However, as more and more writing styles are added, it degrades rapidly in capturing the variance of distinctly different writing styles. For effective handling of the dramatic variance between different writing styles, a script code is calculated from handwriting data as the middle-level feature for subclass clustering. Five kinds of key points on the handwriting sequence are extracted: points of maximum/minimum x-coordinate, points of maximum/minimum y-coordinate and crossing points, denoted as X^+, X^-, Y^+, Y^- and $\bigotimes$ respectively. The average direction of the interval sequence between two adjacent key points is calculated and quantized to eight directions, denoted as $\rightarrow\nearrow\uparrow\leftarrow\swarrow\downarrow\searrow$. Any character can be uniquely coded by means of this notation, as shown in Fig. 1. Based on the distance between script codes, samples of each character are divided into several clusters, and those in the same cluster have a similar structural topology.

Since the length of script code might not be the same in all cases, it is implausible to directly compute the distance. In this study the script code is modeled as a homogeneous Markov chain. Given two script codes T_1 and T_2, we may compute the stationary distributions π_1, π_2 and transition matrix A_1, A_2. The similarity between two script codes is measured as

$$\begin{aligned} d(T_1,T_2) = \exp\Big\{ &-\frac{\lambda_1}{2}[KL(\pi_1,\pi_2)+KL(\pi_2,\pi_1)] - \\ &-\frac{\lambda_2}{2}[KL(A_1,A_2)+KL(A_2,A_1)] \\ &-\lambda_3(n_1-n_2)^2\Big\} \qquad (2) \end{aligned}$$

where $KL(\pi_1,\pi_2)=\sum_{l=1}^{n}\pi_1(l)\log\frac{\pi_1(l)}{\pi_2(l)}$ is the Kullback–Leibler (KL) divergence between two distributions. The positions of π_1, π_2, A_1, and A_2 are enforced symmetrically in Eq. 2. λ_1, λ_2, and λ_3 balance the variance of the KL divergence and the difference in code length. If both the stationary distribution and the transition matrix of two script codes are matched well, and their code lengths are almost the same, the similarity measure $d(T_1,T_2)$ is close to 1.

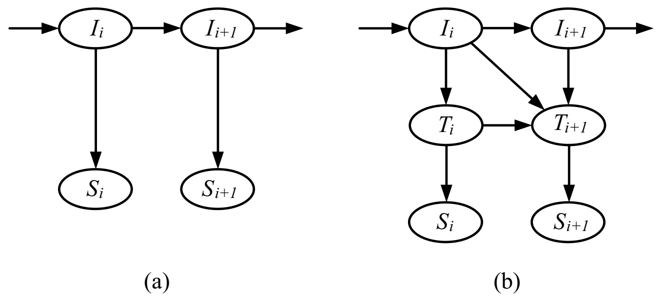


Fig. 2. a Graph model of one-level segmentation framework. **b** Graph model of proposed two-level Bayesian framework

After introducing the script code as middle-level features, the optimization problem becomes

$$\begin{aligned} &\{I_1^*,\ldots,I_n^*\} \\ &= arg\max p(S_1,\ldots,S_n|I_1,\ldots,I_n)p(I_1,\ldots,I_n) \\ &= arg\max p(S,T|I_1,\ldots,I_n)p(I_1,\ldots,I_n) \\ &= arg\max p(I_1)p(T_1|I_1)p(S_1|T_1,I_1) \\ &\prod_{i=2}^{n} p(I_i|I_{i-1})p(T_i|I_i,T_{i-1},I_{i-1})p(S_i|T_i,I_i)\,. \qquad (3) \end{aligned}$$

The graph model of the two-level framework and the traditional one-level framework are compared in Fig. 2. In a pruning level-building architecture, introducing the script code T not only improves the accuracy of segmentation but also dramatically reduces the computational complexity of level-building.

2.2.3 Results. Both the one-level and the two-level algorithms are tested on separate test sets. Segmentation points given by these algorithms are checked by a human being to determine whether they are true or false. The results are listed in Table 1. For comparison, the result of a standard recognition engine is also listed. It is obvious that the two-level algorithm greatly improves the recognition and segmentation ratio compared with traditional low-level algorithm. Furthermore, when doing online segmentation in our system, the word written by the user is known and we can use this information as an additional constraint. By doing this, a correct segmentation ratio of about 86% can be achieved in practical use.

3 Learning strategies

Character samples are segmented from cursive handwriting samples using the methods described above and col-

Table 1. Comparison of the recognition and segmentation results of different methods

Method	Recognition rate		Segmentation rate
	Individual character	Cursive script	
Recognition engine	56.1%	70.5%	—
Low-level	93.7%	46.5%	69.1%
Two-level	95.2%	71.9%	81.3%

lected alphabetically. Samples for each character are first aligned, and then the variation of their trajectories is analyzed by applying PCA.

3.1 Data alignment

3.1.1 Trajectory matching. For training purposes, handwriting trajectory must be represented by a vector of points with a fixed dimension, and these points can be consistently located from one sample to another. In general, extracting these points is a curve-matching problem, and many approaches have been proposed for solving it [13]. For handwriting data, the matching problem can be simplified by exploiting the extensive studies on handwriting trajectory models.

Experimental results in [5] show that a handwriting trajectory can be decomposed into a sequence of static pieces, and each static piece has a time-invariant curvature. Motivated by this study, a hierarchial handwriting alignment algorithm is proposed in which handwriting trajectory is segmented into static pieces by landmark points, and these pieces are matched with each other first. Since each piece is a simple arc, points can be equidistantly sampled from it to represent the stroke.

The landmark-point-extracting method is the same as the one proposed in [5]. Three kinds of points are extracted: pen-down and pen-up points, local extrema of curvature, and inflection points of curvature. The curvature of each point is approximated by the change in the angle of the tangent. In our experiments, the trajectory of a handwriting sample can be divided into as many as six pieces including late strokes such as t crossings and i dots, depending on its topological structure. For a writer with a consistent writing style, samples of the same character are mostly composed of the same number of pieces and they match each other naturally. In this case, a small number of abnormal samples will be discarded. Otherwise, handwriting samples will be divided into subclasses according to their decomposition modes, and for each subclass a separate model will be trained.

3.1.2 Training set alignment. Using the methods described above, a handwriting sample can be represented by a point vector

$$X = \{(x_1^1, x_2^1, \ldots, x_{n_1}^1), \ldots, (x_1^s, \ldots, x_{n_s}^s), \\ (y_1^1, y_2^1, \ldots, y_{n_1}^1), \ldots, (y_1^s, \ldots, y_{n_s}^s)\} , \quad (4)$$

where s is the number of static pieces segmented from the sample and n_i is the number of points extracted from the ith piece. The following problem is to align different vectors into a common coordinate frame. An iterative algorithm is proposed for this purpose that is similar to the one proposed in [14]. For alignment, we estimate an affine transform for each sample that transforms the sample into the coordinate frame. A deformable energy-based criterion is defined as:

$$E = -\log\left[\frac{1}{N_s}\sum_{i=1}^{N_s}\exp\left(-\frac{||X_i - \overline{X}||^2}{2\cdot V_x}\right)\right] , \quad (5)$$

where N_s is the point number in the vectors, $\overline{X}$ is the mean vector calculated as

$$\overline{X} = \frac{1}{N_s}\sum_{i=1}^{N_s} X_i , \quad (6)$$

and V_x is the variance of the Gaussian calculated as

$$V_x = \frac{1}{N_s}\sum_{i=1}^{N_s} ||X_i - \overline{X}||^2 . \quad (7)$$

The algorithm is formally described as follows:

```
1. Maintain an affine transform matrix $U_i$ for each sample, which is set to identity initially.
2. Compute the deformable energy-based criterion $E$.
3. Repeat until convergence:
    (a) For each sample $X_i$,
        i. For each one of the six unit affine matrixes [14] $A_j, j = 1, \ldots, 6$,
            A. Let $U_i^{\text{new}} = A_j \cdot U_i$.
            B. Apply $U_i^{\text{new}}$ to the sample and recalculate the criterion $E$.
            C. If $E$ has been reduced, accept $U_i^{\text{new}}$, otherwise:
            D. Let $U_i^{\text{new}} = A_j^{-1} \cdot U_i$ and apply again. If $E$ has been
               reduced, accept $U_i^{\text{new}}$, otherwise revert to $U_i$.
4. End.
```

The algorithm is essentially a hill-climbing method, and we use simple unit step in optimization, which is not optimal but gives satisfying results in our system. Refer to [15] for more efficient methods.

3.2 Shape models

Aligned vectors form a distribution in the N_s-dimensional space. By modeling this distribution, new examples can be generated that are similar to those in the training set. Like the Active Shape Model [10], principal component analysis is applied to the data. Formally, the covariance of the data is calculated as

$$S = \frac{1}{s-1}\sum_{i=1}^{s}(X_i - \overline{X})(X_i - \overline{X})^T . \quad (8)$$

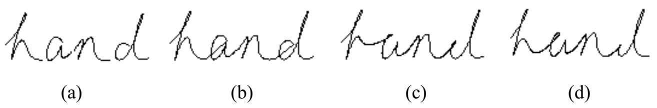


Fig. 3. **a** Synthesized letters from models. **b** Connecting letters directly. There are some discontinuities that cannot appear in a natural handwriting trajectory. **c** Connecting adjacent letters smoothly by deforming their head and tail parts. Although the connection is smooth, shapes of some letters are no longer consistent with models. **d** Connecting letters using the proposed conditional sampling method. The connection is smooth, and the shapes of letters are consistent with models

Then the eigenvectors ϕ_i and corresponding eigenvalues λ_i of S are computed and sorted so that $\lambda_i \geq \lambda_{i+1}$. The training set is approximated by

$$X \approx \overline{X} + \Phi b \,, \tag{9}$$

where $\Phi = (\phi_1|\phi_2|\ldots|\phi_t)$ represents the t eigenvectors corresponding to the largest eigenvalues and b is a v_t-dimensional vector given by

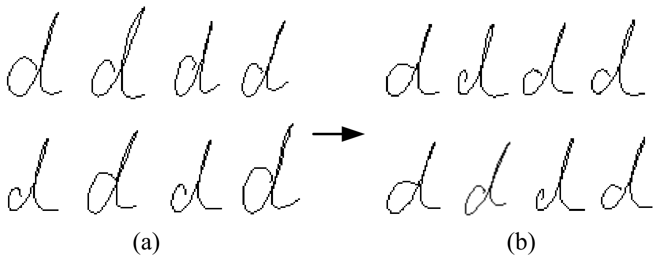
$$b = \Phi^T (X - \overline{X}) \,. \tag{10}$$

By varying the elements in b, new handwriting trajectory can be generated from this model. Furthermore, by applying limits of $\pm 3\sqrt{\lambda_i}$ to the elements b_i, we can ensure that the generated trajectory will be similar to those in the training set.

For late strokes such as t crossings and i dots, instead of training their models separately, we assume that they are virtually connected with the previous strokes of the same letters and train the models for the pairs of strokes together by combining their aligned vectors in sequence.

4 Synthesis strategies

To synthesize a word with learned models, we first generate each individual letter in the word. Then the baselines of these letters are aligned and juxtaposed in a sequence. The following step is to concatenate letters with their neighbors to form a cursive handwriting. However, this objective cannot be easily achieved. As shown in Fig. 3a, directly connecting adjacent letters will produce some discontinuities in the handwriting trajectory. By deforming the head part and tail part of the trajectory of each character, we can obtain a smoother connection, as shown in Fig. 3b. However, we cannot ensure that the deformed letters will still be similar to those in the training set. To solve this problem, a delta log-normal model based conditional sampling algorithm is proposed that can not only produce smooth connections between characters but also ensure that the deformed characters will be consistent with the models, which will be shown in Fig. 5 and Fig. 7. The term *conditional sampling* comes from the fact that in the algorithm, each letter is generated from models under the effects and constraints of its neighbors.



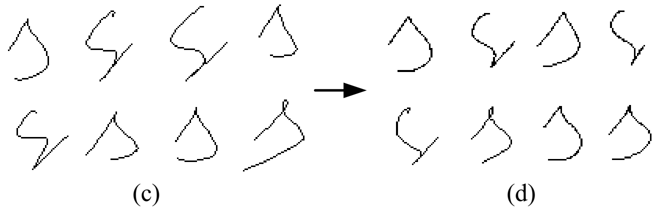


Fig. 4. **a** Samples of letter "d" segmented from training data of a user. **b** Synthesized "d" from trained models. **c** Samples of "s" segmented from training data. **d** Synthesized "s" from trained models

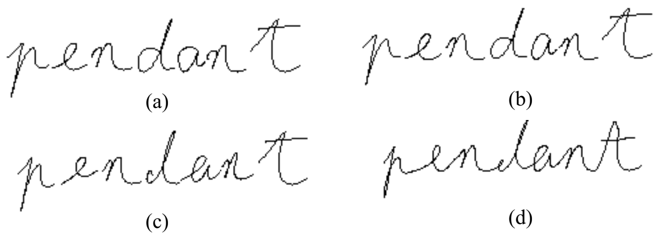


Fig. 5. **a** Initially sampled letters of the word "pendant." **b** Intermediate result after the first iteration of the conditional sampling algorithm. **c** Intermediate result after the second iteration of the conditional sampling algorithm. **d** Final result of the conditional sampling algorithm

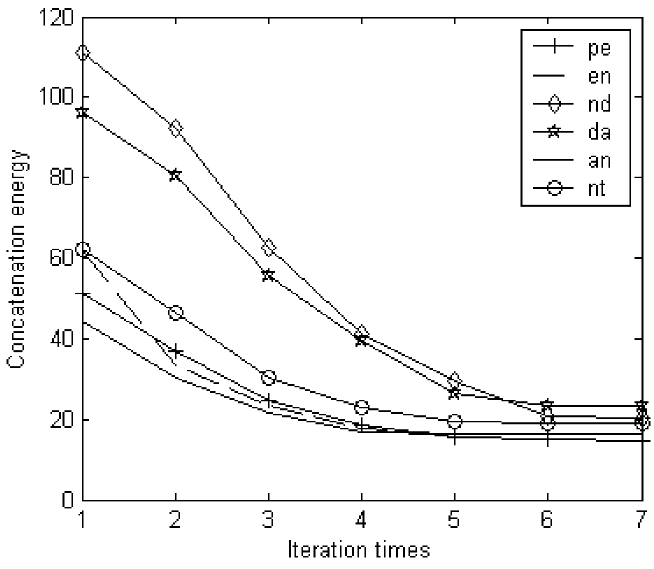


Fig. 6. Concatenation energy between pairs of letters in Fig. 5 in the iterative process of the conditional sampling algorithm

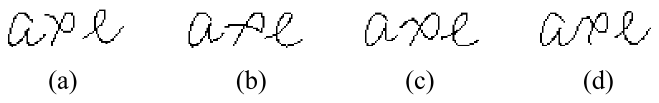


Fig. 7. **a** Initially sampled letters of the word "axe." **b** Intermediate result after the first iteration of the conditional sampling algorithm. **c** Intermediate result after the second iteration of the conditional sampling algorithm. **d** Final result of the conditional sampling algorithm

4.1 Delta log-normal model

The delta log-normal model is a powerful tool in analyzing rapid human movements. It describes a neuromuscular synergy in terms of the agonist and antagonist systems involved in the production of these movements [5]. (See [4,16] for details of this model.) With respect to handwriting generation, the movement of a simple stroke is controlled by velocity [17]. The magnitude of the velocity is described as

$$v_\xi(t) = D_1 \Lambda(t; t_0, \mu_1, \sigma_1^2) - D_2 \Lambda(t; t_0, \mu_2, \sigma_2^2)\,, \tag{11}$$

where

$$\Lambda(t; t_0, \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}(t-t_0)} \times \exp\left\{-\frac{[\ln(t-t_0)-\mu]^2}{2\sigma^2}\right\} t_0 \le t \tag{12}$$

is a log-normal function, where t_0 represents the activation time, D_i the amplitude of impulse commands, μ_i the mean time delay, and σ_i the response time of the agonist and antagonist systems, respectively, on a logarithmic scale axis.

The angular velocity can be expressed as

$$\theta(t) = \theta_0 + \int_{t_0}^{t} c_0 v_\xi(u)du\,, \tag{13}$$

where θ_0 is the initial direction and c_0 is a constant. The angular velocity is calculated as the derivative of $\theta(t)$:

$$v_\theta(t) = c_0 v_\xi(t)\,. \tag{14}$$

Given $v_\xi(t)$, the curvature along a stroke piece is calculated as

$$c(t) = \lim_{\triangle s \to 0} \frac{\triangle\,\alpha}{\triangle\,s} = \lim_{\triangle t \to 0} \frac{v_\theta(t)\cdot \triangle\,t}{v_\xi(t)\cdot \triangle\,t} = c_0\,. \tag{15}$$

This means that the curvature along a stroke piece is time invariant. Thus, the static shape of the piece is an arc, characterized by:

$$S =<\theta, c, D>\,, \tag{16}$$

where $\theta = \theta_0$ is the initial angular direction, $c = c_0$ is the constant curvature, and $D = D_1 - D_2$ is the arc length.

4.2 Conditional sampling

First, the trajectories of synthesized handwriting letters are decomposed into static pieces, using the method described in previous sections. The first piece of a trajectory is called the *head piece*, and the last piece is called the *tail piece*, denoted as $S_h =<\theta_h, c_h, D_h>$ and $S_t =<\theta_t, c_t, D_t>$, respectively. In the concatenation process, the trajectories of letters will be deformed to produce a natural cursive handwriting, by changing the parameters of the head and the tail pieces from S_h and S_t to S_h^* and S_t^*, respectively. Some energy is defined to guide the deformation process. A deformation energy of a stroke is defined as

$$E_d^{h/t} = (\theta^*_{h/t} - \theta_{h/t})^2 + (c^*_{h/t} - c_{h/t})^2 + (D^*_{h/t} - D_{h/t})^2\,. \tag{17}$$

A concatenation energy between the *ith* letter and the $(i+1)th$ letter is defined as

$$E_c(i, i+1) = \lambda_1\left[E_d^t(i) + E_d^h(i+1)\right] + \lambda_2[(c_t^*(i) + \theta_t^*(i)\cdot D_t^*(i) - c_h^*(i+1))^2] + \lambda_3[||p_t(i) - p_h(i+1)||^2]\,, \tag{18}$$

where $p_t(i)$ is the end point of the tail piece of the ith letter and $p_h(i+1)$ is the start point of the head piece of the next letter. There are three items in the defined concatenation energy. The first item constrains the deformation to be smooth and gradual. The second item encourages the angular direction at the end point of the previous letter to be the same as that of the start point of the next letter. The third item diminishes the interval between the two letters. By minimizing the second and the third items, the two letters are forced to connect with each other smoothly and naturally.

The concatenation energy of a whole word is calculated as

$$E_c = \sum_{i=2}^{N_l - 1} E_c(i, i+1)\,, \tag{19}$$

where N_l is the number of letters in this word.

Minimizing the concatenation energy can produce a smooth and natural cursive trajectory. However, as addressed above, we must ensure that the deformed letters are consistent with models. A sampling energy is defined to achieve this objective. For a deformed trajectory, a corresponding v_t-dimensional vector b^* can be calculated from the model using Eq. 10. The sampling energy is calculated as

$$E_s = \sum_{i=1}^{v_t} f\left(b_i^*/\left(3\sqrt{\lambda_i}\right)\right)\,, \tag{20}$$

where the function f is defined as

$$f(x) = \left\{\begin{array}{ccc} 0 & : & |x| <= 1 \\ x^2 & : & |x| > 1 \end{array}\right.\,. \tag{21}$$

The whole energy formulation is finally given as

$$E = \lambda_c \cdot E_c + \lambda_s \cdot \sum_{i=1}^{N_l} E_s(i)\,. \tag{22}$$

To minimize the energy, an iterative approach is proposed and is formally described as follows:

1. Randomly generate a vector $b(i)$ for each letter initially, where $|b_j(i)| < 3\sqrt{\lambda_j}$.
2. Generate trajectories S_i of letters according to their corresponding vectors $b(i)$ and calculate an affine transform T_i for each letter, which transforms it to its desired position.
3. For each pair of adjacent letters $\{S_i, S_{i+1}\}$, deform the pieces in these letters to minimize the concatenation energy $E_c(i, i+1)$.
4. Project the deformed shape into the model coordinate frame: $X_i = T^{-1}S_i$.
5. Update the model parameters to match X_i: $b(i) = \Phi(i)^T(X(i) - \overline{X(i)})$.
6. Apply constraints on $b(i)$ by making $|b_j(i)| <= 3\sqrt{\lambda_j}$.
7. If not converged, return to step 2.

In step 3, the concatenation energy is to be minimized. Formally, a vector of parameter $P(i, i+1)$ is to be calculated as

$$\begin{aligned} P^*(i,i+1) &= <\theta_t^*(i), c_t^*(i), D_t^*(i), \\ &\qquad \theta_h^*(i+1), c_h^*(i+1), D_h^*(i+1)> \\ &= arg \min_{P(i,i+1)} E_c(i,i+1)\,. \end{aligned} \tag{23}$$

This is a multidimensional nonlinear optimization problem, and there exist many methods to solve it. In this study, we use the Direction Set (Powell's) method [18] to minimize the energy. After the iterative process is finished, we make a check on the concatenation energy between each pair of adjacent letters. A predefined threshold E_c^T is used to determine whether to connect these letters. If $E_c(i, i+1) < E_c^T$, the two letters are connected and a local smooth operation is applied at the connecting point. Otherwise, the two letters are left unconnected.

5 Experimental results

The whole system is built on a PC, and handwriting trajectory is collected from a graphical tablet (Wacom Sapphire) connected to it. The system is evaluated on samples written by 15 persons, and models are trained to capture the characteristics of each user's personal writing style.

Figure 4 shows some letters segmented from the training data and some randomly generated letters from trained models. It shows that the synthesized letters have shapes similar to those in the training data.

Figure 5 shows some intermediate results of the conditional sampling algorithm in synthesizing the word "pendant." In the experiments, we set $\lambda_1 = 25$, $\lambda_2 = 50$, and $\lambda_3 = 1$ in Eq. 18. The concatenation energy of each pair of adjacent letters is plotted in Fig. 6. It shows that the concatenation energy drops rapidly in the iterative process and will typically be stable after five or six iterations. Figure 7 shows intermediate results of the conditional sampling in synthesizing the word "axe," and the concatenation energy of pairs of letters are plotted in Fig. 8. Since the concatenation energy between the letters x and e cannot be minimized, these two letters are not able to be connected. This result is consistent with the writing samples of the user, in which the letter x is never connected with the following letters. These results clearly suggest that the proposed conditional sampling algorithm can give satisfying results on concatenating individual letters to cursive handwriting under the constraint of models.

Figure 9a shows some samples written by a user, and Fig. 9b shows the same words synthesized from this user's models. Visual examination suggests that, although trajectories in the two notes are different, their writing styles are consistent. Figure 10 shows the results of another writing style.

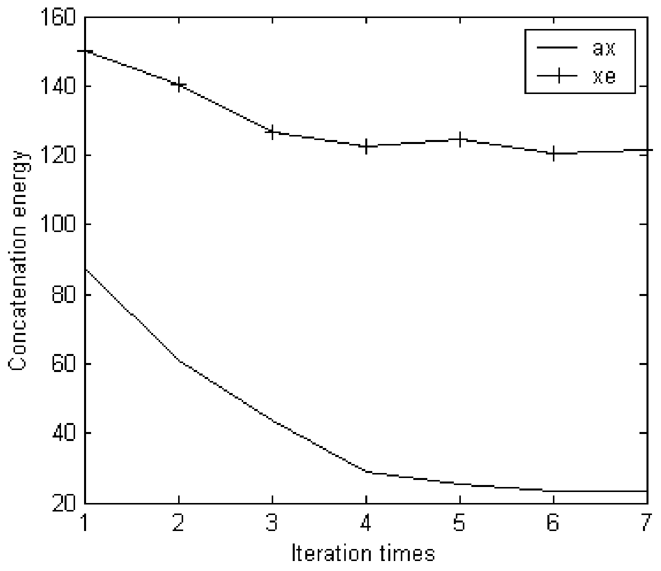


Fig. 8. Concatenation energy between pairs of letters in Fig. 7 in the iterative process of the conditional sampling algorithm

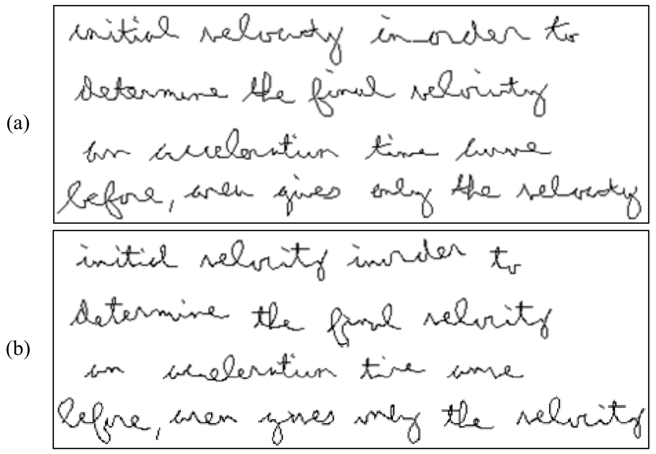

(a)
initial velocity in order to
determine the final velocity
for acceleration time curve
before, even gives only the velocity
(b)
initial velocity inorder to
determine the final velocity
in acceleration time curve
before, even gives only the velocity

Fig. 9. **a** Writing samples of a user. **b** The same words synthesized from models of this user

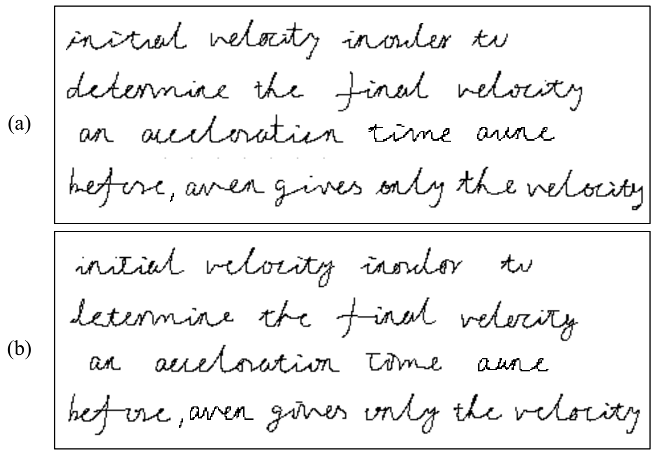

(a)
initial velocity inorder to
determine the final velocity
an acceleration time curve
before, given gives only the velocity
(b)
initial velocity inorder to
determine the final velocity
an acceleration time curve
before, given gives only the velocity

Fig. 10. **a** Writing samples of another user. **b** The same words synthesized from models of this user

6 Discussion and conclusion

Handwriting synthesis has many important applications to facilitate users work and personalize communication on pen-based devices. This study addresses the problem of learning personal writing styles from limited samples and producing novel scripts of the same styles. Cursive handwriting samples are first segmented into individual characters by a two-level writer-independent segmentation algorithm. Samples of the same character are aligned into a common coordinate frame and learned by shape models that have the ability to generate shapes similar to those in the training set. To synthesize cursive script, individual letters are generated from models and concatenated by a conditional sampling algorithm. Some experimental results are shown to demonstrate the effectiveness of the proposed algorithms.

Although the proposed method is superior to the direct sampling method [7], its performance is still limited by samples used for training since the shape models can only generate novel shapes within the variation of training samples. To produce more variant and natural handwriting, users are required to give more handwriting samples, and sometimes this is not practical. A better way to solve this problem might be to build models of different writing styles on a large data set and match each user's writing style to an existing model and make some modification on the model, according to the small number of input samples.

Another problem that needs to be addressed is that, although some experimental results are shown, it is still not known how to make an objective evaluation on the synthesized scripts and compare different synthesis approaches. Extensive studies exist on the individuality of handwriting [19], and some rules proposed in those studies may be helpful in achieving such a goal.

References

1. Plamondon R, Maarse F (1989) An evaluation of motor models of handwriting. IEEE Trans PAMI 19(5):1060–1072
2. Singer Y, Tishby N (1993) Dynamical encoding of cursive handwriting. In: Proc. IEEE CVPR
3. Chen H, Agazzi O, Suen C (1997) Piecewise linear modulation model of handwriting. In: Proc. 4th international conference on document analysis and recognition, Ulm, Germany, pp 363–367
4. Plamondon R, Alimi A, Yergeau P, Leclerc F (1993) Modeling velocity profiles of rapid movements: a comparative study. Biol Cybern 69:119–128
5. Li X, Parizeau M, Plamondon R (1998) Segmentation and reconstruction of on-line handwritten scripts. Pattern Recog 31(6):675–684
6. Beigi H (1996) Pre-processing the dynamics of on-line handwriting data, feature extraction and recognition. In: Proc. 5th international workshop on frontiers of handwriting recognition, Colchester, UK, pp 255–258
7. Guyon I (1996) Handwriting synthesis from handwritten glyphs. In: Proc. 5th international workshop on frontiers of handwriting recognition, Colchester, UK
8. Wang J, Wu C, Xu YQ, Shum HY, Ji L (2002) Learning-based cursive handwriting synthesis. In: Proc. 8th international workshop on frontiers of handwriting recognition, Canada
9. Duta N, Jain AK, Dubuisson-Jolly M-P (2001) Automatic construction of 2D shape models. IEEE Trans PAMI 23(5):433–446
10. Cootes T, Taylor C (2000) Statistical models of appearance for computer vision. Draft report, Wolfson Image Analysis Unit, University of Manchester, UK
11. Casey RG, Lecolinet E (1996) A survey of methods and strategies in character segmentation. IEEE Trans PAMI 18(7):690–706
12. Proctor S, Illingworth J, Mokhtarian F (2000) Cursive handwriting recognition using hidden Markov models and a lexicon-driven level building algorithm. IEE Proc Vis Image Signal Process 147(4):332–339
13. Munich ME, Perona P (1999) Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In: Proc. IEEE international conference on computer vision, Greece
14. Miller E, Matsakis N, Viola P (2000) Learning from one example through shared densities on transforms. In: Proc. IEEE CVPR, Hilton Head, SC
15. Rowley H (1999) Neural network-based face detection. Ph.D Thesis, Sect. 2.3, Carnegie Mellon University, Pittsburgh
16. Guerfali W, Plamondon P (1995) The delta lognormal theory for the generation and modeling of handwriting recognition. In: Proc. ICDAR, pp 495–498
17. Plamondon R, Guerfali W (1996) Why handwriting segmentation can be misleading? In: Proc. 13th international conference on pattern recognition, Vienna, Austria, pp 396–400
18. Press W, Teukolsky S, Vetterling W, Flannery B (1992) Numerical recipes in C, 2nd edn. Cambridge University Press, Cambridge, UK
19. Srihari S, Cha S, Arora H, Lee S (2002) Individuality of handwriting. J Forensic Sci 47(4):1–6

Jue Wang received his BE and MS from the Department of Automation, Tsinghua University, Beijing, China in 2000 and 2002. He is currently a Ph.D. student in the Department of Electrical Engineering, University of Washington in Seattle. His research interests include image processing, computer vision, pattern recognition, and multimedia networks.

Chenyu Wu received her BE and MS from the Department of Automation, Tsinghua University, Beijing, China in 2000 and 2003. She is currently a Ph.D. student in the Robotics Institute, Carnegie Mellon University. Her research interests include image processing, computer vision, and pattern recognition.

Ying-Qing Xu received his BS in mathematics from Jilin University (China) in 1982 and his Ph.D. in computer graphics from the Chinese Academy of Sciences in 1997. He is a researcher at Microsoft Research Asia. His current research interests include photorealistic rendering, physical-based animation, cartoon generation, and image-based rendering.

Heung-Yeung Shum joined Microsoft Research after receiving his Ph.D. in robotics from the School of Computer Science, Carnegie Mellon University, in 1996. He is assistant managing director of Microsoft Research Asia and also leads the research group on visual computing. His research interests include computer vision, computer graphics, video representation, learning, and visual recognition. Before joining Microsoft Research, he worked at Digital Equipment Corporation's Cambridge Research Lab, Apple Computer's Interactive Media Lab, and RealSpace, Inc. He holds more than 20 US patents.