Style-Preserving English Handwriting Synthesis

Zhouchen Lin[a, *] Liang Wan[b]

[a]*Microsoft Research, Asia, Zhichun Road #49, Haidian District, Beijing 100080, P.R. China*

[b]*Chinese University of Hong Kong, Shatin, Hong Kong*

Abstract

This paper presents a novel and effective approach to synthesize English handwriting in the user's writing style. We select the most important features that depict the handwriting style, including character glyph, size, slant, and pressure, special connection style, letter spacing, and cursiveness. The features can be efficiently computed with the aid of our specially designed sample collecting user interface. Given ASCII text, the user handwriting is synthesized hierarchically. First, character glyphs are sampled and shape variation is added. Second, words are generated by aligning the character glyphs on the baseline with proper horizontal inter-character space and vertical offset from the baseline. The heads and tails of the letters may be trimmed to avoid severe overlap and facilitate possible connections between neighboring letters. Adjacent letters may be connected to each other by polynomial interpolation. Finally, after the pressure is assigned, the handwriting is rendered word by word and then line by line. The experimental results prove the capability of our system to adapt to the user writing style.

Key words: handwriting, handwriting synthesis, writing style, cursive, connection.

* Corresponding author. Tel: +86-10-62617711, ext. 3143; Fax: +86-10-88097306.

Email addresses: zhoulin@microsoft.com (Zhouchen Lin), lwan@cse.cuhk.edu.hk (Liang Wan)

1 Introduction

Pen-based computing has become an active research area in human-computer interaction with the flourish of many pen-based devices such as Tablet PCs, Personal Digital Assistants (PDAs), and Electronic White-boards. Besides handwritten document analysis (e.g. [17]), pen-based user interface (UI, e.g. [18]), and handwriting recognition (e.g. [19]), handwriting manipulation, such as handwriting editing, error correction, and script searching, is also a hot topic in pen-based computing. In contrast, handwriting synthesis, i.e., converting ASCII text into the user's personal handwriting, is an important yet much less explored problem. It adds a personal touch to communications, e.g., enabling the receiver of an email to read the handwriting of the sender [21]. Like wallpapers and favorite software settings, synthesized handwriting also contributes to the personalization of one's computing devices [21]. Moreover, it can free the user from lengthy and stressful writing, e.g., when preparing many handwritten documents such as greeting cards with different text. Handwriting synthesis may also be helpful to forensic examiners [22], the disabled [21], and the handwriting recognizer (by generating more training or testing samples for the recognizer [23]).

Existing methods for handwriting synthesis can be roughly divided into two categories. The first one is based on the handwriting reconstruction process [1,2], in which the handwriting trajectory is analyzed and modeled by velocity or force functions. Though physically plausible, these methods may not be convenient for synthesizing non-cursive handwriting. The second category involves glyph-based methods [8–10], which usually record the glyphs directly and reuse or sample the glyphs when synthesis. These methods require intensive user involvement in the sample collection process and cannot produce various handwriting styles, e.g., from handprint style to fully cursive style, in a natural way.

In this paper, we present a novel and practical approach to English handwriting synthesis. It generates different handwriting even for the same ASCII text and supports different handwriting styles. The synthesized handwriting looks natural and is similar to the user's original handwriting as shown by the experimental results. Compared with the existing methods, our system requires less user involvement in the process of collecting handwriting samples.

It is observed that the visual appearance of English handwriting is affected by many factors [12,13]. In our system we extract features, such as character glyph, size, slant, and pressure, special connection style, letter spacing, cursiveness, as the user's writing style. The feature extraction is efficiently done with the aid of the specially designed sample collecting user interface. In particular, the user is only required to input each distinct character three times, several special pairs of letters, and several multi-letter words.

After extracting the writing style, our system synthesizes handwriting hierarchically. It firstly selects appropriate character glyphs after deciding the connection states between lowercase letters in a word. Then each glyph is geometrically perturbed and aligned on the baseline with appropriate horizontal distance between neighboring glyphs and vertical offsets from the baseline. Next, the adjacent letters are connected to each other using high-order polynomial interpolation, if they are decided to be connected according to the connection states. The heads or tails of the glyphs may be trimmed in order to avoid server overlap and to ease connection. Then the pressure is assigned to the ligature, and words are rendered one by one to form a line. The lines are further stacked into paragraphs.

The rest of this paper is organized as follows. Section 2 reviews related work. Then the following three sections introduce the factors that depict the handwriting style, the extraction of user handwriting style, and the handwriting synthesis process, respectively. Next, we present the experimental results in Section 6. Finally, we give conclusions and

future work in Section 7.

2 Related work

Since the late 1980s, people have tried to interpret handwriting from an underlying physical scheme [1,3]. Different computational models have been proposed for relating velocity and force to the handwriting trajectory. A typical example is the delta log-normal model [2,19]. Li et al. [4] employed this model to represent the velocity of handwriting trajectory and encode the trajectory by a group of parameters. Bezine et al. [5] proposed a beta-elliptic model to estimate the correlation between geometry and kinematics in fast handwriting generation. These models prove to be successful for representing, compressing, and reconstructing captured handwriting (e.g. [6,7,4]), but not synthesizing novel handwriting. Schomaker et al. [1] introduced another computational model for the production of handwriting. Assuming that the handwriting is ballistic and fluently cursive, the handwriting is segmented into compound strokes that are modeled by a group of parameters in the velocity domain. Given an input text, a grammar for the connection of cursive allographs determines abstract codes for connecting strokes, then symbols are translated into a sequence of parameterized strokes. Their method requires that the handwriting samples be *non-hesitant* and written by experienced adult writers.

On the other hand, recording the user's handwriting directly with a digital capturing device and "redrawing" it faithfully on the (receiver's) computer with the recorded information, such as pen-tip position, pressure, and brush style, may be the simplest way to produce personal handwriting. A pen-based system, such as a Tablet PC, provides such a functionality, with which users are able to write on the screen using a digital pen and save the handwritten document. It turns out to be laborious for users in case of long-time writing, such as preparing lengthy emails or numerous e-greeting cards. In addition, whenever

handwriting is required the user has to write him/herself.

Personal font design provides a more automatic way to produce handwriting. There have been many commercial font design services, such as Personal Font and ParaType. Customers are usually required to fill a form and send it to a font design company [21]. Font experts will select good handwriting samples and make sophisticated adjustments before creating a TrueType® or a PostScript® font for the customer. Then the user uses the personal font as a system font. However, users may feel inconvenient as the font creation requires the involvement of font designers. Furthermore, the output handwriting has no variation in character glyphs or word appearance that natural handwriting is supposed to have. In addition, without careful writing and font tuning, cursive handwriting cannot be generated because the characters in system fonts can only be rendered side by side without generating the ligature on the fly.

Handwriting synthesis can combine the advantages of the above two approaches. Like personal font, it lets users type on the keyboard or simply copy text, then the system will generate the handwritten script. Moreover, it enables users to produce more natural handwriting without depending on font experts. In the following, we review some important work on handwriting synthesis.

In 1996 Guyon [8] introduced a straightforward approach to synthesize handwritten words. The system collects handwritten glyphs of single characters and letter groups that most frequently appear in English text, such as “tion” and “ing”. When synthesizing a word, the system splits the word into letter groups or characters. For example, “believe” may be partitioned into “be”, “li”, and “eve”. Then the corresponding glyphs are placed side by side without additional effort to connect them into a fluent handwriting. This method does not handle glyph variation (although a global transform is tried). As a result, the synthesized handwriting has a regular appearance, and possible connections exist

within each glyph only. In addition, the system requires users to write more than *a thousand* letter groups in order to provide complete samples, which is tedious and impractical.

In 2002 Wang et al. [9] proposed a learning-based cursive handwriting synthesis system. The trajectory is represented by a set of sparse control points and B-spline interpolation is used to reconstruct it. They employ a tri-unit letter model in which a letter is segmented into the head, body, and tail parts. The letter glyphs and ligatures (e.g., parts connecting neighboring letters) of the cursive words are extracted by template matching. The distribution of the control points of each character is learnt via PCA. For each ligature, the segmented samples also form a generative distribution. During synthesis, the letters and ligatures are randomly sampled from the generative distributions. Then a geometric deformable model is applied to smooth the ligature part which consists of the tail of the previous letter, the ligature, and the head of the latter letter.

In 2003, Wang et al. [10] proposed an improved algorithm over [9] to achieve better results in letter segmentation and ligature generation. Given a handwritten sample, they extracted features at two levels: coordinates of trajectory points and script codes that depict the shape of letter glyph at a higher level. Then a two-level framework of level building is applied to optimally segment single letters from cursive handwritten samples, which reaches a correct segmentation rate of about 86%. Finally they adopted the delta log-normal model [2,19] to synthesize smooth cursive handwriting .

The work of Wang et al. [9,10], however, has several drawbacks. First, their systems always require the user to write in fully cursive style. Unfortunately, partially cursive handwriting and handprint styles are also common in real situations. Second, a large handwriting database should be collected to learn the *a priori* distribution of letter glyphs for letter segmentation. Third, during the segmentation process human interaction is demanded to fix the letter segmentation error as automatic segmentation is not always correct [9,10,14].

Such a procedure is not natural to non-technical users. Fourth, the sparse control point representation and PCA learning of character glyphs may result in distortion such that the generated glyphs may be invalid. Finally, their systems do not consider the pressure variation of the strokes, which may make the synthesized handwriting less realistic and less readable because letter spotting, as the first step of reading, becomes more difficult without the help of stroke width variation to indicate the beginning and the end of each letter.

Choi et al. [11] presented a character generation method based on Bayesian networks, which integrates on-line handwriting recognizers. Instead of fluent handwriting, their method generates separated characters only as they did not consider the ligature between letters in the case of cursive handwriting. Furthermore, the method discards the personal handwriting characteristics since the Bayesian networks represent the “average” writing style of all users. Though it could be extended to adapt to personal writing styles, a large amount of handwriting samples may have to be collected for each user.

3 Factors that Contribute to the Handwriting Style

The handwriting of different people differs in many aspects. These aspects actually define the handwriting style of a person. As suggested by handwriting analysis techniques in forensic inspection [13] or character analysis [12], factors that are easily noticeable to ordinary people to distinguish different handwriting styles include: 1. the glyph and the size of characters; 2. the pressure distribution and the slant of handwriting; 3. the relative sizes of the middle, the upper, and the lower zones of letters; 4. the existence and the shape of lead-in, connecting, and ending parts; 5. the letter, the word, and the line spacings; 6. the embellishment; and 7. the simplified or neglected strokes.

In our system, we choose features that depict the first five factors since they are relatively easy to be computed by computers (as will be shown in Section 4), or simply be provided by the user as samples. The synthesis of embellishment and the simplified or neglected strokes may require a thorough understanding of handwriting dynamics [1,3,2] or even psychology, which is still not fully available. Experimental results (in Section 6) show that our system is capable of characterizing many aspects of handwriting style and adapting to various handwriting styles.

4 Computing the Features of Handwriting Style from Samples

To make the system practical for ordinary users, we need a natural and intuitive way by which a user "tells" the computer what his/her writing style is. In our first approach, we requested users to write a paragraph of text and segment the samples from the handwritten document. However, automatic segmentation does not always produce correct results [14,9,10]. Manually fixing the segmentation errors is unnatural and inconvenient for a non-technical user. In our current system, we let users write isolated characters (with ligature parts if users prefer cursive writing) instead. Though there might be mismatch between the ligatures exhibited in isolated characters and those that appear when writing words, writing isolated characters makes the glyphs of individual characters easily available to the computer. Specifically, we carefully design a sample collecting UI (Fig. 1) which facilitates extracting features of the handwriting style. The UI consists of three parts for collecting samples of individual characters, special letter pairs, and multi-letter words, respectively. The user is requested to follow three stages (corresponding to each part of the sample collecting interface), during which certain characters and character pairs are collected.

Fig. 1.

At the first stage, the user is asked to write all individual characters that appear on a QWERTY keyboard (94 characters in total) three times (Fig.s 1(a)∼(d)). The three samples for each character serve for glyph variation during synthesis. If the user prefers cursive writing (which can be detected later when the user writes multi-letter words), the three samples of lowercase letters will be viewed as their appearance at the beginning, the middle, and the end of words, respectively. The UI also reminds the user to provide the head and/or tail parts of the lowercase letters for possible connection (Fig. 1(a)). Note that three samples per character may not capture all possible variants. To avoid asking the user to input a large number of samples, we choose to add glyph variation during synthesis instead. Based on the samples, the following features are computed for each character:

(1) *Character glyph*. We use a dense sequence of control points to represent the character glyph. To do so, Sklansky's algorithm [15] is adopted to approximate each stroke, i.e., the trajectory of pen from pen-down to pen-up, by a polyline. Intermediate points may be inserted to the sequence of polyline vertices if successive vertices are farther than the average length of the polyline segments or the polyline has high curvature at those points. These polyline vertices are taken as the control points of the character glyph. This representation supports easy control on the character glyph in two aspects compared to the wave function approximation [2,19]. First, glyph variation can be efficiently realized by simply moving control points around. Second, ligatures between neighboring letters can be conveniently generated by adding control points. In comparison to Wang et al.'s sparse control point representation [9,10], our dense control point representation better preserves the character glyph and eases the letter head/tail trimming (Section 5.2.2) and ligature generation (Section 5.2.3).

(2) *Character size*. The size of character glyph may differ significantly when the characters are written separately in our UI. As a result, the relative sizes among characters may appear unbalanced when synthesizing a line or a paragraph. Therefore size normalization is necessary. However, making the height or the width of characters identical may remove natural size variation. Instead, we develop a scaling algorithm so that the abnormal size variance among the characters is minimized while the natural size variation is preserved. Readers may refer to Appendix for the details.

(3) *Pressure*. It is the measure of how heavily the user presses the pen against the screen. It is physically captured when the user writes on a Tablet PC.

(4) *Slant*. The letter slant is estimated as the average direction of letter strokes [16]. The global writing slant is taken as the average of all the letter slants. Then each letter is de-slanted with its own slant so that during synthesis the global writing slant can be applied to generate handwriting with a more uniform slant. It is possible to add small slant variation to every character so that the synthesized handwriting looks more casual.

(5) *Average height of capital letters, middle zone letters, and descendent letters* (Please refer to Appendix for their classification). They are estimated from the normalized letters. In particular, the average height of middle zone letters is very useful for aligning letters and punctuations that do not lie on the baseline.

(6) *Existence and shape of lead-in, connecting, and ending pieces*. They are provided by the user, e.g., the head and/or tail parts of lowercase letters. However, whether the user prefers cursive writing and how a lowercase letter connects to others are still unknown. Such information will be further probed by asking the user to write special letter pairs and multi-letter words.

Our current system assumes that only lowercase letters may be connected to each other. We further separate lowercase letters into uni-stroke letters and multi-stroke letters. The multi-stroke letters include ‘f’, ‘i’, ‘j’, ‘t’, ‘x’, and ‘z’, which are probably written in multiple strokes. Here ‘z’ is considered as a multi-stroke letter because some people like to add a dot to the middle of it (Fig. 14(b)). The rest lowercase letters are uni-stroke letters. Note that multi-stroke letters have more than one way of connecting to other lowercase letters. For example, when writing “ta” (Fig.s 2(7)∼(12)), the user may write the t-stem first and then connect ‘a’ to the t-bar (Fig. 2(9)), or write the t-bar first and then connect ‘a’ to the t-stem (Fig. 2(8)). Though uni-stroke letters may be written in multiple strokes, we assume they have only one connection type, i.e., connection happens at the beginning point or the end point of the letter.

The second stage of sample collection is designed to identify the special connection style of multi-stroke lowercase letters, i.e., how they are connected to other lowercase letters. For simplicity, the connection between two multi-stroke letters are not considered in the current system. The user is asked to write special letter pairs, “af”, “fa’, “ai”, “ia”, “aj”, “ja”, “ta”, “at”, “ax”, “xa”, “az”, and “za” (Fig. 1(e)) once. We pair multi-stroke lowercase letters with ‘a’ to determine their connection styles because: 1. ‘a’ is easy to connect when the user prefers cursive writing; 2. ‘a’ is usually written in a single stroke, which greatly reduces the complexity of analysis; and 3. its shape and stroke length make it robust for the bounding box test and the length test described in Algorithm 1.

Fig. 2.

Fig. 2 shows possible ways of writing “at” and “ta”. The examples show that the connection style of multi-stroke letters can be very complex. The connection style includes the head connection type and the tail connection type. Let us start from the tail connec-

tion, using ‘t’ as an example. In Fig. 2, cases 7∼9 illustrate three possible ways in which the end point of a ‘t’ glyph is the connection point. This case is defined as the NORMAL type. Note that ‘t’ may be written as a uni-stroke in case 7, or two strokes in cases 8 & 9. In cases 11 & 12, ‘t’ has no connection to ‘a’. This is defined as the NO_CONNECTION type. Case 10 shows a special situation where the last point of the first stroke of ‘t’ is the connection point, i.e., the t-bar is a late stroke. This is defined as the SPECIAL_TAIL type. Similarly, for the head connection, cases 1∼4 have letter ‘a’ connected to the beginning point of a ‘t’ glyph. Therefore, they belong to the NORMAL type. And cases 5 & 6 are of NO_CONNECTION type.

The following pseudo code (Algorithm 1) shows the heuristic rules of determining the tail connection type of letter ‘t’ by checking the number, the bounding boxes, and the lengths of the strokes in the letter pair “ta”. The head connection type of letter ‘t’ can be determined in a similar manner. Although the algorithm is presented to deal with letter ‘t’, it is applicable to other multi-stroke lowercase letters ‘f’, ‘i’, ‘j’, ‘x’, and ‘z’.

4.3 Features from Multiple-Letter Words

In this stage, the user is asked to write several multiple-letter words (Fig. 1(f)) in order to get information of spacing and cursiveness.

(1) *Letter spacing*. It is defined as the distance between the central lines of neighboring letters. We estimate it as the average letter width of multi-letter words after deslanting the words. Each multi-letter word provides an estimate of the letter spacing. For simplicity, we model the distribution of the letter spacing by a Gaussian.

(2) *Cursiveness*. Cursiveness is a measure of how much the user prefers cursive writing. It is between 0 and 1, where 0 represents that the user prefers handprint writing while 1 denotes that the user likes completely cursive writing. During synthesis, the sys-

Algorithm 1. Determine the tail connection type of letter 't'.

input: Strokes S for letter pair "ta"

output: Tail connection type $tailType$

if S contains 1 stroke

$tailType$ = NORMAL;

else if S contains equal or more than 3 strokes **then**

$tailType$ = NO_CONNECTION;

else

Compute the bounding boxes, B_1 and B_2, of the two strokes;

Compute the lengths, L_1 and L_2, of the two strokes;

if overlap between B_1 and B_2 is small **then**

$tailType$ = NO_CONNECTION;

else if $L_1 < c \cdot L_2$ //$c = 1.0$ for letter 't'

$tailType$ = NORMAL;

else

$tailType$ = SPECIAL_TAIL;

end

end

tem has to determine which pair of adjacent letters in a word is connected. It would be ideal if we compute the connection probability from handwritten samples. However, it is impractical in real practice due to the large amount of letter pairs. Because writing all the pairs once is laborious, and writing each letter pair only once cannot provide an accurate estimate of the connection probability. Instead, we decouple the pairwise connection probability into two components: the writer-independent *a priori* connection probability, which defines the easiness of connecting a pair of letters and is estimated by counting the frequencies of their connection in handwriting sam-

ples (not those samples provided by the user), and the writer-dependent cursiveness, which measures how much the user prefers cursive writing. The two components jointly estimate the connection probability of any letter pair during synthesis (Please refer to Section 5.1.1). The user cursiveness can be estimated as follows.

In the UI the user is asked to write a particular set of words. Given the i-th word, let n_i be the number of letters, m_i be the number of expected strokes when the word is written in handprint style, and k_i be the number of detected strokes. The cursiveness p_i of the i-th word is defined by:

$$p_i = \min\left(\max\left(\frac{m_i - k_i}{n_i - 1}, 0\right), 1\right).$$

For example, "table" has 5 letters ($n_i = 5$), and 6 expected strokes ($m_i = 6$) assuming 't' is written in two strokes and each of the rest letters written in a single stroke. A user may write the whole word in one stroke only ($k_i = 1$). Then we have $p_i = 1$. The user cursiveness is calculated as:

$$P_{user} = \frac{1}{M}\sum_{i}^{M} p_i,$$

where M is the number of multi-letter words that the user writes. It is easy to check that, the fully cursive writing style yields $P_{user} = 1$; the complete handprint style yields $P_{user} = 0$; and the mixed style (partial cursive and partial handprint) yields $P_{user} \in (0, 1)$. A similar definition of cursivity index is introduced in [20].

5 Handwriting Synthesis Process

Based on the features of handwriting style, our system synthesizes handwriting in a hierarchical way. For an input ASCII text, the glyphs of characters are first generated. Then the characters are aligned on the baseline and are connected when needed to form a word. Finally, the words are aligned into lines and further paragraphs. During synthesis,

the extracted handwriting style features will be used in the subsequent processing steps described in Sections 5.1, 5.2, and 5.3.

5.1 Character Generation

Fig. 3 shows the flowchart of character generation and the required information of the handwriting style. As the glyphs of a lowercase letter when connected or disconnected to other letters may differ significantly, we generate the letter glyph based on the knowledge of its connection state, i.e., whether it is connected to its previous or subsequent letters, so that appropriate samples can be chosen (Please refer to the first paragraph of Section 4.1.). For the remaining characters, the three samples are randomly selected. Then a geometric deformation is applied to perturb the character glyph. In the following, we will present these steps in more details.

Fig. 3.

5.1.1 Connection state sampling

With the user cursiveness P_{user}, we can estimate the probability of connecting the i-th and the j-th letters as:

$$P_{ij} = \begin{cases} P_{user}, & \text{if } P_{user} = 0, 1, \\ \min(\alpha P_{user} p_{ij}, 1), & \text{otherwise,} \end{cases} \tag{1}$$

where $\alpha^{-1} = \dfrac{1}{26 \times 26} \sum_{i,j} p_{ij}$, and p_{ij} is the writer-independent ***a priori*** connection probability (i.e., the relative easiness of connecting the i-th and the j-th letters, see Section 4.3).

For an input ASCII word, the connection probability P_{ij} of every pair of neighboring lowercase letters is approximated by Eq. (1). Our system then generates a random number r that is uniformly distributed on [0, 1]. If $r \leq P_{ij}$, this letter pair is decided to be

connected. Otherwise, they will not be connected. After each adjacent pair is processed, the states for whether a letter connects with its previous or next one are determined.

5.1.2 Glyph initialization

For each letter in the word, we choose one of its three samples as the initial glyph. Recall that we assume only lowercase letters might be connected to each other, and that the three samples are supposed to appear at the beginning, middle, and end of words, respectively. The initial glyph of a lowercase letter will be selected according to its position in the word and its sampled connection state. More specifically, the “beginning” sample is chosen when the lowercase letter is at the beginning of the word, or at a middle position but not connected with the previous letter. The “end” sample is chosen if the letter is at the end position or at a middle position but not connected with the subsequent letter. The “middle” sample is chosen only when the letter is at a middle position and connected to both its previous and subsequent letters. For example, given a word “hello” if the connection state sampling decides that every two neighboring letters will be connected except for ‘e’ and the first ‘l’ (Fig. 4). Then we will chose the “beginning” samples for ‘h’ and the first ‘l’, the “middle” sample for the second ‘l’, and the “end” samples for ‘e’ and ‘o’, as shown in Fig. 4. If the word has only one letter, the “beginning” sample is chosen. For capital letters, digits, or punctuations, the three samples are selected randomly.

Fig. 4.

5.1.3 Geometric deformation

We apply geometric deformation to simulate handwriting variation in real situations. This method brings the advantage of avoiding the collection of a large number of handwritten samples. As illustrated in Fig. 5, for stroke pieces delimited by high curvature points, we sequentially apply local random rotation and random scaling to them, where

the starting point of the current piece is fixed at the ending point of last piece that has undergone perturbation. The deformation is at a small scale so that the perturbed glyph looks similar to, but still different from, the original one.

Fig. 5.

5.2 Word Composition

To compose the glyph of a word (Fig. 6), we first align the letter glyphs, vertically and horizontally, against the baseline. The heads or tails of the glyphs may be trimmed to avoid severe overlap and to facilitate smooth connection. Then the ligatures between neighboring glyphs are generated by utilizing high-order polynomial interpolation. Finally the ligatures are assigned with pressure values.

Fig. 6.

5.2.1 Vertical alignment

Vertical alignment places letter glyphs vertically with respect to a horizontal baseline. More specifically, for middle-zone letters, ascendent letters, capital letters, and digits (Please refer to Appendix), the bottom of their bounding boxes is expected to meet the baseline. For descendent letters, the top of their bounding boxes is expected to meet the top of the middle zone, which is determined by the height of middle zone letters. For all-zone letters (such as ‘j’) and punctuations, we assign the vertical offsets from the baseline as scales of the middle zone height. The scales are class-dependent so we choose not to spread out the empirical formulae due to the page limit.

5.2.2 Horizontal alignment

Horizontal alignment places letter glyphs horizontally along the baseline. We expect the distance between the central lines of the *bodies* of two neighboring letters to be d, which

is sampled from the letter spacing distribution (Please refer to Section 4.3). However, the letter samples often have the head and the tail parts that are useful for connection but may interfere in accurate central line computation. The letter glyphs may also severely overlap each other when the head or tail parts are rather long. As a result, the synthesized handwriting may look weird or it may be hard to produce smooth ligatures. In the following, we design a head/tail trimming scheme to remove redundant portions of heads/tails.

Fig. 7.

To detect the head and the tail, the end of the head part and the beginning of the tail are first roughly estimated at the first cusp and the last cusp of the letter (Fig. 7), respectively. At these cusps, the turning angles exceed a threshold. Such an estimation may not be accurate for letters without salient head or tail part. We may refine the head/tail positions by detecting the points that have maximum or minimum values in either horizontal or vertical coordinates within the roughly estimated head/tail part (Fig. 7). The index of the refined head/tail position is the minimum/maximum of the indices of these points, in which the beginning or end points of the stroke are not taken into account.

After detecting the head and the tail, the remaining parts form the body of the letter. If a part of the head/tail is *outside* the bounding box of the body and *inside* the bounding box of its neighboring letter, this part is clipped as a redundant part (Fig. 8).

Fig. 8.

Up to now, letter glyphs still have vertical central lines since the letter samples are deslanted when extracting the writing style (Please refer to Section 4.1). After cutting the head/tail parts of all letter pairs in the word, we may shear these letter glyphs with the global writing slant.

5.2.3 Ligature generation

When two neighboring letters are connected according to the sampled connection states, a smooth ligature is expected to occur between them. We propose using a high-order polynomial to fit the ligature part, which consists of the tail part T of the first letter, the head part H of the second letter, and the line segment L linking the end point of T and the beginning point of H (Fig. 9(a)). In particular, for multi-stroke lowercase letters, their head/tail connection types tell which stroke may contribute to the head/tail part. For example, let the t-bar and the t-stem be the first and second stroke of 't' separately. If the tail connection type of 't' is NORMAL, the t-stem will be selected to connect 't' and the next lowercase letter (Fig. 2(8)). If the tail connection type of 't' is SPECIAL_TAIL, the t-bar instead will be selected for connection (Fig. 2(9)).

Fig. 9.

Assume the ligature to be parameterized by

$$\mathbf{P}(t) = \sum_{k=0}^{N} \mathbf{P}_k t^k, \qquad t \in [0,\ 1]\,,$$

where P_k are the control points of the ligature to be determined, and N is the number of control points. We impose three constraints on the ligature: similarity to the original ligature, deformation energy from the original ligature, and smoothness of the ligature.

The similarity requires that the new ligature should be close to the original ligature. It is defined as

$$E_1 = \int_0^1 \xi_1(s) \left\|\mathbf{O}(s) - \mathbf{P}(s)\right\|^2 ds,$$

where $\xi_1(s)$ is a weighting function and $\mathbf{O}(s)$ is the parametric function for the original ligature interpolated from the control points by cubic splines. In order to allow larger

deviation at the part of L, $\xi_1(s)$ should be smaller when s is parameterizing L.

The deformation energy requires that, conceiving the ligature as a spring, the new ligature should be deformed from the old one with least energy. The deformation energy is defined by

$$E_2 = \int_0^1 \|\mathbf{O}'(s) - \mathbf{P}'(s)\|^2 \, ds.$$

The smoothness requires that the resulting ligature is smooth, defined as

$$E_3 = \int_0^1 \xi_3(s) \, \|\mathbf{P}''(s)\|^2 \, ds,$$

where $\xi_3(s)$ is a weighting function. Because the non-smoothness occurs around the end points of L, $\xi_3(s)$ should be larger when s is parameterizing the parts around the end points of L.

Based on the above energy functions, the control points $\mathbf{P}_k$, $k = 1, \cdots, N$, should minimize the following function:

$$E\left(\{\mathbf{P}_k\}_{k=1}^N\right) = \lambda_1 E_1 + \lambda_2 E_2 + \lambda_3 E_3,$$

with boundary conditions:

$$\mathbf{P}_0 = \mathbf{O}(0), \qquad \sum_{k=0}^{N} \mathbf{P}_k = \mathbf{O}(1), \qquad \mathbf{P}_1 = \mathbf{O}'(0), \qquad \sum_{k=0}^{N} k\mathbf{P}_k = \mathbf{O}'(1),$$

where λ_1, λ_2, and λ_3 are constants. They are chosen as $\lambda_1 = 0.89$, $\lambda_2 = 0.085$, and $\lambda_3 = 0.025$, respectively, by trial and error. The above problem turns out to solve a linear equation for $\mathbf{P}_k$, $k = 1, \cdots, N$.

In implementation, some details should be considered when preparing the ligature from

neighboring letter glyphs. First additional control points may be inserted to ensure that both T and H have at least three control points (could be replicant). Second, if L is close to vertical, it may be difficult to connect two glyphs smoothly. In this case, the first point of H is dropped and L is updated as the line segment linking the end point of T and the second point of H (Fig. 9(b)), so that the slope of L can be smaller.

5.2.4 Pressure assignment

The variation in stroke width not only adds liveliness to handwriting, but also helps reading as the letter spotting becomes easier when the strokes of the letter end with diminishing width. The stroke width variation can be fulfilled by introducing pen pressure to the stroke. Recall that the pressure on the letter samples has been captured during handwriting sample collection, so the pressure of points at the body part simply inherits its original value. For the ligature, we assign the pressure by "transferring" the pressure from the head and the tail parts. Assume that there are m points on the head and the tail parts, and $p_1, \cdots, p_m$ are their pressure values captured in letter samples. If there are n points on the new ligature, the pressure value of the i-th point is assigned by

$$\tilde{p}_i = p_k, \quad \text{where } k \text{ is the integer part of } i \cdot \frac{m}{n}.$$

Given the stroke points and pen pressure, the rendering APIs provided by Microsoft® Tablet PC Ink SDK will automatically render the strokes, where low-pressure parts have small width and high-pressure parts have large width.

In our system, multiple words are rendered one by one with the inter-word spacing, i.e., the distance between the right of the bounding box of the first word and the left of the bounding box of the second word, being assumed as half of the letter spacing. Should the handwriting appear in multiple lines, we have to choose an appropriate line spacing. We have observed that users often have the top of the second line meet the bottom of the first line. Then we may take the spacing as follows:

$$d_{line} = H_{cap} + H_{des} - H_{mid} + \Delta h,$$

where H_{cap}, H_{des}, and H_{mid} are the height of capital letters, descendent letters, and middle-zone letters (Please refer to Section 4.1), respectively, and Δh is a small positive value to ensure that the handwriting on two lines does not overlap, so that the synthesized handwriting is more readable. In our system, Δh is empirically chosen as 10. Randomness can also be added to Δh to enrich the naturalness of synthesized handwriting.

6 Experimental Results

We build the handwriting synthesis system on a Tablet PC with which the users can write directly on the screen with a digital pen. Eight testers are invited to test our system. They are from China, USA, and Japan, respectively. Their handwriting styles vary from handprint to completely cursive, as shown in the left column of Fig. 10. Four testers have no experience of writing on a Tablet PC and they are allowed to practise to get accustomed to writing on the screen. Usually, a user can finish inputting his/her handwriting samples at his/her normal writing speed within twenty minutes. The sample collecting process can

be much faster if the writer is experienced of using Tablet PCs, as tested by the authors.

Fig. 10.

Fig. 11.

Fig. 12.

Fig. 10 shows some handwriting samples of the eight users and the synthesized glyphs. One can see that most of the synthesized words are quite similar to the original samples. Note that the synthesized words vary from handprint to completely cursive. Therefore, the cursiveness of the writers is well preserved. Fig. 11 shows handwriting paragraphs synthesized by our system using the writing styles of the eight users, respectively. On a Pentium 2.8GHz PC, it takes about one second to synthesize this paragraph of text for each user using our unoptimized codes. The computation of horizontal alignment and ligature generation accounts for the majority of time. Table 1 is the cross rating among the testers, i.e., each tester evaluates whether the synthesized handwriting of every writer is similar to its corresponding real handwriting. The evaluation shows that the performance of our system is rather satisfactory.

Fig. 12 shows the paragraphs generated by the approach proposed in [9] in the styles of the second and the eighth writers (Fig. 10(b1)(b2)(h1)(h2) and Fig. 11(b)(h)). One can see that our approach produces more natural, readable, and user-dependent handwriting, and the difference in visual appearance among different writers is much larger than that in Wang's results.

Fig. 13.

We also have our system integrated with Microsoft® Office® Outlook®. Fig. 13 shows an example of the communication via emails. Although the sender sends a text email, what the receiver reads is a handwriting email. The handwritten script is sent as an image so that the requirement on the receiver's system is minimal. We choose the image format as

Table 1

Cross rating among the testers. The score at the cross of row i and column j is the rating of the tester i on the similarity between the synthesized handwriting and the real handwriting of the tester j . The scores are between 1 and 5, with 1 being completely dissimilar and 5 being very similar.

	Tester 1	Tester 2	Tester 3	Tester 4	Tester 5	Tester 6	Tester 7	Tester 8
Tester 1	4	4	5	5	4	4	4	2
Tester 2	5	3	4	5	3	5	3	2
Tester 3	4	5	4	5	5	4	4	2
Tester 4	4	4	5	4	4	5	5	3
Tester 5	5	5	5	5	3	5	5	3
Tester 6	5	4	4	4	3	4	4	2
Tester 7	5	4	5	5	4	4	4	2
Tester 8	4	4	5	5	4	4	4	1

TIFF which ensures a small image size while preserving the visual quality of thin strokes. For the given example, the image of the handwriting is about 22KB.

7 Conclusions and Future Work

In this paper, we presented a novel handwriting synthesis system which extracts the user's handwriting style and synthesizes new handwritten scripts according to the user's writing style. Particularly, our system respects the cursiveness that varies from completely handprint to completely cursive, as well as the special connection styles of multi-stroke lowercase letters. The experimental results demonstrate that our system can produce personal handwriting with pleasing visual quality.

The proposed system, however, does not capture all aspects of the handwriting style. For example, we only provide connection between lowercase letters and the variance of letter glyphs is simply approximated by geometric deformation. Moreover, our system assumes that the users write at constant speed. But users may write more quickly and

less patiently after some time in real situations such that the handwriting may become crabbed. We may incorporate such an effect by introducing the impact of time and speed on the handwriting appearance. Other handwriting psychology should also be understood to make our system more robust. As shown in Fig. 10, some synthesized words, such as the synthesized glyphs of “people” and “little” for the eighth writer, look different from their counterparts. It is mainly because the isolated letter samples were written with quite long head/tail parts that actually do not appear when the user writes words. Finally, although currently our system only supports English handwriting, it is possible to be extended to support other western languages with some modifications. Considering general handwriting synthesis, incorporating part of our techniques, e.g., the treatment on handprint and partial cursive writing styles and the multi-stroke letters, with the computational model proposed by Schomaker et al. [1] may be a possible way. In this case, the ligature insertion method described in [1] might be adapted to make the ligature generation process simpler.

As argued above, there are opportunities to improve our current system. However, in this paper we have discussed various advantages of our approach. First, due to the pragmatic procedure of collecting a relatively small amount of samples, the required involvement of the user is minimal compared to other systems. Second, our results appear visually acceptable (for both cursive and handprint handwriting), which was sustained by a user study presented in this paper. Third, we believe that compared to the commercial font design services, our approach offers a valuable and more personal alternative, which mimics true handwriting in a better way.

Appendix: Size Normalization

In paragraph writing, the relative sizes among characters usually appears uniform. But they may differ significantly when the characters are written separately in our input UI.

Therefore, size normalization should be done among the same class of characters or among the samples of a character. The characters can be classified into seven classes:

(1) Middle zone letters: a, c, e, m, n, o, r, s, u, v, w, x;

(2) Ascendent letters: b, d, h, i, k, l, t;

(3) Descendent letters: g, p, q, y;

(4) All-zone letter: f, j;

(5) Capital letters: A∼Z;

(6) Digits: 0∼9;

(7) Others: z and the rest characters.

For the first six classes of characters, the size normalization is applied so that the heights of the normalized characters in the same class are almost the same. For the last class of characters, the size normalization is done among the three samples of each character only. Note that 'z' is singled out because it has at least three kinds of glyphs in handwriting (Fig. 14), and one of the glyphs is descendent (Fig. 14(c)).

Fig. 14.

We wish not to make the heights of the characters in the same class identical in order to preserve the natural size variation. Therefore, we propose a scaling algorithm so that the size variance among the characters is minimized, and on the other hand the scaling factor for each sample is also close to 1. These constraints try to preserve the natural size variation while suppressing abnormal size variation. Suppose the scaling factor for each character is s_i, and their optimal width is W_{opt}. We have to find W_{opt} and $\mathbf{s} = (s_1, \cdots, s_N)$ to minimize both functions:

$$\begin{aligned} g(\mathbf{s}, W_{opt}) &= \frac{1}{2}\sum_{i=1}^{N}(s_i w_i - W_{opt})^2 + \frac{1}{2}\sum_{i=1}^{N}\left(s_i h_i - \frac{1}{N}\sum_{j=1}^{N} s_j h_j\right)^2, \\ \phi(\mathbf{s}) &= \frac{1}{2}\sum_{i=1}^{N}(s_i - 1)^2, \end{aligned} \tag{2}$$

where w_i and h_i are the width and height of the i-th sample, and N is the number of samples in a given class. The minimization of g aims at making the size of the scaled characters be as uniform as possible, while the minimization of ϕ requires the scaling factors to be as close to 1 as possible. We do not replace W_{opt} with $\frac{1}{N}\sum_{j=1}^{N} s_j w_j$ because we want the width of the characters to be more uniform so that the horizontal alignment can be easier. The solution to (2) is:

$$W_{opt} = \frac{\mathbf{1}^T \mathbf{A}\mathbf{w}}{\|\mathbf{A}\mathbf{w}\|^2}, \quad \mathbf{s} = W_{opt}\mathbf{A}\mathbf{w},$$

where $\mathbf{1} = (1, \cdots, 1)^T$, $\mathbf{w} = (w_1, \cdots, w_N)^T$, and $\mathbf{A} = (\mathbf{\Lambda} - N^{-1}\mathbf{h}\mathbf{h}^T)^{-1}$, in which $\mathbf{\Lambda} = \text{diag}(h_1^2 + w_1^2, \cdots, h_N^2 + w_N^2)$ and $\mathbf{h} = (h_1, \cdots, h_N)^T$. After normalization, the average width and height of each character are recorded for later use.

References

[1] L.R.B. Schomaker, Simulation and Recognition of Handwriting Movements, Doctoral Dissertation/PhD Thesis (NICI TR-91-03). Nijmegen University, The Netherlands, 1991.

[2] R. Plamondon, A kinematics theory of rapid human movements, Part I: movement representation and generation, Biological Cybernetics 72 (1995) 295-307.

[3] R. Plamondon, and F. Maarse, An evaluation of motor models of handwriting, IEEE Trans. Pattern Analysis and Machine Intelligence 19(5) (1989) 1060-1072.

[4] X. Li, M. Parizeau, and R. Plamondon, Segmentation and reconstruction of on-line handwritten scripts, Pattern Recogtion 31(6) (1998) 675-684.

[5] H. Bezine, A.M. Alimi, and N. Derbel, Handwriting trajectory movements controlled by a beta-elliptical model, Proc. Seventh Int'l Conf. Document Analysis and Recognition, Edinburgh, Scotland (2003) 1228-1232.

[6] H. Chen, O. Agazzi, and C. Suen, Piecewise linear modulation model of handwriting, Proc. Fourth Int'l Conf. on Document Analysis and Recognition, Ulm, Germany (1997) 363-367.

[7] H. S.M. Beigi, Processing, modeling and parameter estimation of the dynamic on-line handwriting signal, Proc. World Congress on Automation, Montpellier, France (1996).

[8] I. Guyon, Handwriting synthesis from handwritten glyphs, Proc. Fifth Int'l Workshop on Frontiers of Handwriting Recognition, Colchester, England (1996) 309-312.

[9] J. Wang, C.Y. Wu, Y.Q. Xu, H.-Y. Shum, and L. Ji, Learning based cursive handwriting synthesis, Proc. Eighth Int'l Workshop on Frontiers of Handwriting Recognition, Ontario, Canada (2002) 157-162.

[10] J. Wang, C.Y. Wu, Y.Q. Xu, and H.-Y. Shum, Combining shape and physical models for on-line cursive handwriting synthesis, Int'l J. Document Analysis and Recognition, to appear.

[11] H. Choi, S.-J. Cho and J.H. Kim, Generation of handwriting characters with Bayesian network based on-line handwriting recognizers, Proc. Seventh Int'l Conf. Document Analysis and Recognition, Edinburgh, Scotland (2003) 995-999.

[12] K.K. Amend and M.S. Ruiz, Achieving Compatibility with Handwriting Analysis, Newcastle Publishing Co., Inc., North Hollywood, California, 1992.

[13] S. Srihari, S. Cha, H. Arora, and S. Lee, Individuality of handwriting, J. Forensic Science 47(4) (2002) 856-872.

[14] R. Plamondon and W. Guerfali, Why handwriting segmentation can be misleading? Proc. Int'l Conf. on Pattern Recognition, Vienna, Austria (1996) 396-400.

[15] J. Sklansky and V. Gonzalez, Fast polygonal approximation of digitized curves, Pattern Recognition 12 (1980) 327-331.

[16] R. Powalka, Experiments with applying slant counteraction to script recognition, Department of Computing, Technical Report of The Nottingham Trent University (1993).

[17] A.K. Jain, A. Namboodiri, and J. Subrahmonia, Structure in on-line documents, Proc. Int'l Conf. on Document Analysis and Recognition (2001) 844-848.

[18] K. Hinckley et al., Design and analysis of delimiters for selection-action pen gesture phrases in scriboli, Proc. SIGCHI Conf. on Human Factors in Computing Systems (2005) 451-460.

[19] R. Plamondon and S.N. Srihari, On-line and off-line handwriting recognition: a comprehensive survey, IEEE Trans. Pattern Analysis and Machine Intelligence 22(1) (2000) 63-82.

[20] L. Vuurpijl and L. Schomaker, Coarse writing-style clustering based on simple stroke-related features. Proceedings of the 5th International Workshop on Frontiers in Handwriting Recognition (1996) 29-34.

[21] FontGod Corperation, http://www.fontgod.com/

[22] C.C. Tappert, Handwriting synthesis of a particular writer's style. http://csis.pace.edu/ ctappert/research/researchtopics.htm

[23] T. Varga and H. Bunke, Generation of synthetic training data for an HMM-based handwriting recognition system. Proceedings of International Conference on Document Analysis and Recognition, pages 618-22, 2003.

About the author—ZHOUCHEN LIN received the Ph.D. degree in applied mathematics from Peking University in 2000. He is currently a researcher in Visual Computing

Group, Microsoft Research, Asia. His research interests include computer vision, computer graphics, pattern recognition, statistical learning, document processing, and human computer interaction. He is a member of the IEEE.

About the author—LIANG WAN is a Doctoral candidate of Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. She was a visiting student at Microsoft Research, Asia.

Figure legends:

(1) Figure 1. The user interface (UI) to collect user handwriting samples. (a) The overall appearance of the UI when collecting the samples of lowercase letters. (b)∼(f) The sample collection boxes when collecting the samples of capital letters, digits, punctuations, special letter pairs, multi-letter words, respectively.

(2) Figure 2. Different connections between ‘t’ and ‘a’. (1)∼(6) Different ways of writing “at”. (7)∼(12) Different ways of writing “ta”. Note that in (8) the t-bar is the first stroke of ‘t’, while in (10) the t-bar is the second stroke of ‘t’.

(3) Figure 3. The flowchart of character generation.

(4) Figure 4. Selecting lowercase letter samples according to their positions in the text word and the connection states.

(5) Figure 5. Adding geometric deformation to each stroke piece sequentially. With small random scaling and rotation of each stroke piece (delimited by the high-curvature points indicated by the dots), the perturbed stroke (solid stroke) may be different, but still similar, to the original stroke (dashed stroke).

(6) Figure 6. The flowchart of word composition.

(7) Figure 7. Detecting the head and the tail. The end point of the head part and the beginning point of the tail part are first detected as the cusps (round dots) that are close to the ends of the stroke, and are then refined with the x-min-max or y-min-max points (square and diamond dots) in the estimated head or tail parts. In this example, the head part is detected as the part before the hollow dot because it is also the x-max and y-max point, and the tail part is the part after the diamond dot.

(8) Figure 8. The rule of trimming head/tail parts of a letter. If part of the head or tail part is outside the bounding box of the body and inside the bounding box of its neighboring letter, this part is clipped. In this example, the bounding boxes of ‘a’, its body part, and the subsequent letter ‘d’ are the thin solid, the thick solid, and the

dashed rectangle, respectively. Therefore, the tail part of ‘a’ between lines C and D is clipped.

(9) Figure 9. (a) The original ligature between two letters. The dots are the control points on the ligature. (b) The first control point of the head part of the second letter may be dropped so that the slant of the linking line segment is smaller. This will be a better initial shape for ligature.

(10) Figure 10. Comparison of the captured handwriting samples (left column) of eight writers and the synthesized handwriting (right column).

(11) Figure 11. Synthesized handwriting paragraphs in the style of the eight writers.

(12) Figure 12. Examples of the synthesized handwriting by Wang’s system [9] in the styles of the second and the eighth writers, respectively. Note that they look similar although the actual handwritings are quite dissimilar. Moreover, the completely cursive writing style required by the system causes severe deformation in letter glyphs.

(13) Figure 13. Integration of our handwriting synthesis system with Microsoft® Office Outlook®. (a) The text email composed by the sender. (b) The synthesized handwriting email read by the receiver. The handwriting is sent as an image.

(14) Figure 14. Different ways of writing ‘z’.