Synthesis of Online Handwriting in Indian Languages

C. V. Jawahar, A. Balasubramanian
Center for Visual Information Technology,
International Institute of Information Technology,
Gachibowli, Hyderabad - 500 032
jawahar@iiit.ac.in

Abstract

Synthesis of handwriting has a variety of applications including generation of personalized documents, study of writing styles, automatic generation of data for training recognizers, and matching of handwritten data for retrieval. Most of the existing algorithms for handwriting synthesis deal with English, where the spatial layout of the components are relatively simple, while the cursiveness of the script introduces many challenges. In this paper, we present a synthesis model for generating handwritten data for Indian languages, where the layout of characters is complex while the script is fundamentally non-cursive. The algorithm can learn from annotated data and improve its representation with feedback.

1. Introduction

Given an input text, the problem of handwriting synthesis is to generate data that is close to how a human would write the text. The characteristics of the generated data could be that of a specific writer or that from a generic model. Even with a given model, the synthesis method should not be deterministic since the variations that are found in human handwriting are inherently stochastic. However, if we need to generate data that is similar to a particular writer's handwriting, we need to identify, model, and preserve the basic characteristics of his/her handwriting. The problem of maintaining the writing style while introducing variability [1] makes the problem of synthesis very difficult. A handwriting synthesis solution has variety of applications including automatic creation of personalized documents [2], generation of large quantities of annotated handwritten data for training recognizers [3, 4], and writer-independent matching and retrieval of handwritten documents.

Traditionally, handwriting synthesis has been dealt within the realm of offline handwriting [5], where the handwritten data is a scanned image of a paper document. With the popularity of hand-held devices and Tablet PCs, electronic capture of handwriting is emerging as an appealing alternative to the traditional pen and paper-based handwriting. Handwritten data collected by these devices incorporate temporal information about the writing process, in addition to the spatial information present in traditional handwritten data. Hence the data from such devices are referred to as *online handwriting* or *digital ink*. Online handwriting is stored as a sequence of *strokes*, where each stroke is defined as the trace of the pen tip from a pen-down to the next pen-up. Devices with pen-based interfaces facilitate storing of the handwritten ink in the digital format and thus enabling a variety of applications such as search and retrieval of large sets of handwritten notes [6] as well as efficient communication across the Internet. In the context of digital ink, the technique of handwriting synthesis is extremely useful as it leads to applications that preserve the compactness of online data, while being natural.

2. Characteristics of Indian Scripts

The problems associated with handwriting synthesis are different depending upon the nature of the script that one is trying to synthesize. Languages that use the Roman script contain a small set of symbols that are arranged in a linear fashion to create a word. The complexity of these scripts arises due to the cursive nature of the script, where the individual characters are connected together. In fact real-world handwriting is a mixture of cursive and non-cursive parts, which makes the problems of recognition and synthesis, more difficult.

Indian language scripts are fundamentally non-cursive in nature, where the *aksharas* (characters) are written independently, separated by space or pen-lifts. However, these scripts often contain a large number of characters that have complex spatial layout of strokes. Indian scripts have compound characters, which are combinations of multiple consonants and vowels. The handwriting synthesis process should hence model all the possible variations of characters and their combinations to be able to generate any given text. This makes the problem of synthesis, extremely complex in the case of Indian language scripts. There are many other properties of the Indian scripts that are not seen in Roman. Following is a list of interesting characteristics of Indian scripts that are relevant to the synthesis problem.

- Words in many of the Indian language scripts have *Shirorekha* (horizontal bar on the top). Figure 1 (a) shows how *Bangla* characters are joined on the top with the *Shirorekha* to form a word. After writing the individual characters, *Shirorekha* is drawn

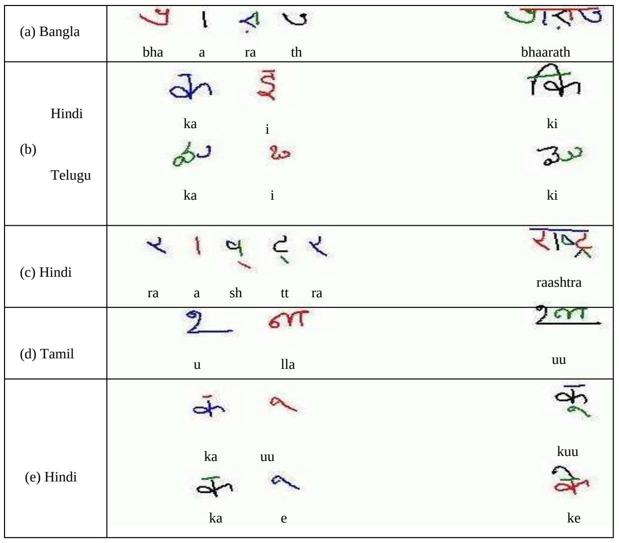
(a) Bangla
bha a ra th
bhaarath
(b)
Hindi
ka i
ki
Telugu
ka i
ki
(c) Hindi
ra a sh tt ra
raashtra
(d) Tamil
u lla
uu
(e) Hindi
ka uu
kuu
ka e
ke

Figure 1. Some of the special cases in Indian language scripts.

to connect them and describe the word boundaries. This need not be considered as a signature of cursiveness. Some of the Indian languages that have *Shirorekha* are Hindi, Bangla, Marathi, and Gurumukhi (Punjabi).

- Figure 1 (b) shows the cast of the vowel modifiers in *Telugu* and *Hindi*. The consonant *ka* when combined with the vowel *e* gives a compounded akshara *ki*. Vowels often get converted as an augmented shape modifiers to consonants in most Indian scripts. Vowels could also appear in isolation.

- Figure 1 (c) is an example from the *Devanagari* script which shows how the word *raastr* is formed from the basic consonants and vowels. Here the variant is further more complex with the last akshara *str* being formed by the following three consonants *ss* , *tt* and *ra*. This is called the *Samyuktakshara* which could contain multiple consonants and a vowel.

- Figure 1 (d) shows an example from the *Tamil* script where we have a vowel *u* and a consonant *lla*. They both join together to give out a new character, the vowel *uu*.

- In Figure 1 (e) we have the consonant *ka*, which when combined with the vowel *uu* results in *kuu* and when combined with the vowel *e* results in *ke*. One has to note that both the vowel modifiers for *uu* and *e* have similar looking strokes, however their positions are different.

One may observe that for Indian scripts, spatial positioning of the strokes is equally important as their shapes.

3. Synthesis of Non-Roman Scripts

Handwriting generation and synthesis has been of immense interest. Broadly there are two basic approaches in synthesizing the handwriting. Earlier attempts employ the motor model based synthesis of handwriting [7, 8]. Singer and Tishby [8] model the handwriting process as modulation of oscillatory motions of the pen. The modulation parameters decide the shape and variations of the generated data. The Delta LogNormal Model by Guerfali and Plamondon [7] is based on movement simulation techniques, and may be defined as a curvilinear stroke generator made up of two parallel neuromuscular networks, which control the agonist and the antagonist [9] activities associated with a specific movement. The second approach is based on the mathematical model of an algebraic curve whose shape is controlled by parameters, typically direction and time [2, 10, 7]. There have been other attempts like the vector-matrix of successive strokes by Kondo [11], and by Schomaker *et al.* [12] who used allograph codes as inputs.

Roman script was the primary focus in the motor model based synthesis techniques [9, 7]. On the other hand there has been no concrete model available for the oriental languages. Some oriental characters need many pen-tip lifting steps due to the presence of large number of short segments. Scripts such as the Korean has been studied before for the purpose of synthesis [13]. The Beta-Velocity model was proposed by Lee and Cho [13] to simulate cursiveness with a letter or a word. This model was an improvised version of the Delta LogNormal model, the main difference being that the Beta-Velocity model uses asymmetric curves, whose skewness can be controlled by variables.

The work presented in this paper is primarily aimed at Indian language scripts, which are different from both western and other oriental scripts. As per our knowledge, this is the first such attempt on the synthesis of Indian language scripts.

As pointed out before, the scripts of Indian languages are more complex than Roman script. The complexity arises due to a variety of factors:

- Alphabets of Indian scripts have far more complex shapes and varied writing styles.

- The size of the alphabets is typically high. In addition, the presence of *samyuktaksharas* (compound characters) makes modeling of Indian scripts more difficult.

- The basic stroke shapes in Roman scripts are often unambiguous in their meaning. However, this is not the case with the Indian scripts, where a single stroke shape can acquire different meanings depending on its position and size.

- Indian scripts are non-cursive in nature and thus the available models need not be the most suitable.

- The spatial location of an akshara is dependent on the previous akshara in some Indian language scripts.

In this paper, we develop a stroke shape and layout model for Indian language scripts that can be learnt from labeled samples. Hence we can use the model for generation of a specific writer or develop a generic writer model. The proposed model captures both the shape and temporal aspects of the strokes as well as their order information. Hence we can generate either online or offline data using the learnt parameters.

4. Modeling of Handwritten Data

Figure 2 gives the outline of the learning and synthesis steps of our method. The handwriting model consists of two parts, a **stroke model**, which captures the shapes and variations in the basic strokes that form the characters, and a **layout model**, which controls the spatial layout of the individual strokes. The individual models can be learnt from online handwritten data that is collected from a writer or multiple writers. The training data needs to be annotated manually. Examples of strokes corresponding to each stroke class is used to learn the stroke model. The spatial distribution of strokes are learnt by the layout model.

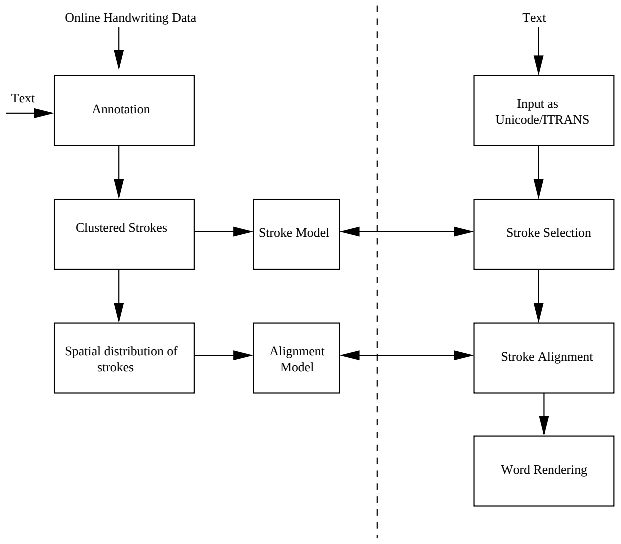

Figure 2. Block diagram for Handwriting Synthesis. The modules to the left of the dotted line forms the training phase, while those to the right form the synthesis.

4.1. Stroke Model

Each script contains a set of basic strokes that are used to form all the characters in the script. For example, Hindi consists of around 200 basic strokes, while Telugu consists of more than 300 basic strokes. The stroke model consists of a representation for each of these basic strokes. We have used two different models for representation of the strokes:

1. **Normalized Template Model:** In this model, we represent each basic stroke class using a set of training strokes, that are normalized in size. The stroke selector will randomly select one of the samples for the required class and scale it to the appropriate size determined by the position of the stroke in the character. This model is appealing in many applications due to its simplicity and is useful in cases where the training data is limited. It can also provide high quality synthesis in presence of limited training samples. However the range of variations that are present in the results could be limited with smaller training samples. The model M, could be represented by a set of k stroke models, each containing a set of stroke samples:

$$M = \{S_1, S_2, .., S_k\}$$
$$S_i = \{s_1, s_2, .., s_{n_i}\}$$
$$s_i = [(x_1, y_1)(x_2, y_2)..(x_m, y_m)],$$

where S_i represents the model of the i^{th} stroke class containing n_i stroke samples, and each s_i is a sample that consists of a sequence of (x, y) coordinate pairs.

2. **Mean Trace Model:** A more complex model that is capable of generating a large set of stroke samples is the mean trace model. Here we compute the mean of all the traces (or strokes) of each of the given stroke classes during the training phase. To compute the mean trace, the strokes are normalized and the points are aligned using an elastic matching technique. The distribution of the samples of the strokes are estimated using the means and covariance matrices of the aligned sample points and are stored in the order in the trace. The stroke selector module will generate random samples from each of the distributions to create a new stroke of a given class, assuming a Gaussian distribution. The alignment and estimation steps make sure that outliers are not included in the training phase, so that the learnt model is close to the supplied handwriting.

$$M = [\bar{X}_1, \bar{X}_2, .., \bar{X}_k]$$
$$\bar{X}_i = \frac{1}{n_i}\sum_{j=1}^{n_i}(x_j, y_j),$$

where $\bar{X}_i$ represents the model of the i^{th} stroke class containing n_i stroke samples, and each sample point of the strokes model are the mean of the corresponding points ((x, y) coordinate pairs) after alignment and outlier removal. The mean model estimate is hence the maximum likelihood estimate of the sample sequence, assuming each sample comes from a multivariate Gaussian distribution.

One could also employ generative models such as HMMs to learn the stroke structure and generate samples of a person's handwriting. However, we noticed that such models tend to allow a lot of variation in the writing style, and the individuality information is lost in the training phase. Hence they are not best suited for synthesis applications in their basic form.

Note that the stroke models that we employ are relatively simple compared to the motor model based approaches employed for Roman scripts. However, since the strokes themselves are only parts of characters in the Indian language scripts, their spatial distribution, captured by the layout model, can generate realistic renderings of handwriting.

4.2. Layout Model

The most important part of our synthesis model is the layout model that is capable of capturing the spatial relationship between stroke classes. As noted before the spatial layout between the strokes in an akshara can be very complex. A straight forward approach is to learn the spatial distribution of strokes within each akshara. However, the number of possible aksharas in Indian languages are very large when considering the samyuktaksharas or compound characters. The total number of aksharas can run into many thousands. Moreover learning such a model does not exploit the redundancy in information present between similar characters. For example, all the modifications of a multi-stroke character will have the basic set of strokes repeated in some form. To exploit this redundancy, we model the spatial layout as a set of pairwise spatial distributions between stroke classes.

Let ω_i and ω_j be two stroke classes that are modeled using the stroke model defined in section 4.1. We describe the layout model as the spatial relationship between the two strokes ω_i and ω_j in terms of distance, r, and direction, θ. The distance and direction could be measured using the centroids of the strokes or based on the bounding boxes. In this experiment we use the top left corner of the bounding boxes to compute the relative distance and direction.

Let $p(r, \theta|\omega_i\omega_j)$ represent the spatial distribution of the class ω_j w.r.t the class ω_i, where r is the radial distance and the θ is the angle between the two classes. We represent the spatial layout $D_{\mathcal{L}}$ of a language $\mathcal{L}$, as a set of such mutual spatial distributions of the strokes:

$$D_{\mathcal{L}} = \{p(r, \theta|\omega_i\omega_j)|\omega_i \text{ and } \omega_j \text{ are neighbors in } \mathcal{L}\}$$

The total number of possible stroke pairs in Indian languages can be very high since the number of stroke classes range from 200 to 350. However, not all stroke pairs will appear in proximity to each other in any character in the language. Hence the typical number of parameters that we need to estimate is around 1000. Here we assume that the parameters r, θ form a distribution for each stroke class pair, which can be modeled as a multivariate Gaussian distribution. The mean and covariance matrices are estimated as the MLE, similar to that in the stroke model.

Each character in the script could compose of multiple strokes and the number can differ based on the writing style. Hence, in addition to the spatial layout, we also need to learn the set of stroke classes that are used to write a particular character by specific writer. This information is identified and stored for each character class during the training phase.

In Figure 2, the estimation of the model is depicted to the left of the dotted line. Once the model parameters are estimated from annotated samples, we can synthesize any given text based on the synthesis procedure (shown to the right of the dotted line in Figure 2).

5. Synthesis from Spatial Layout

Let $\omega_1, \omega_2, ..., \omega_k$ be the k primitive classes which compose the entire script. These primitive classes compose of strokes extracted from the stroke model. Let x_i be a particular stroke in the training samples of ω_j. Conventional algorithms model $P_{\omega_j}(x_i)$ based on a set of parameters that control the shape of x_i [10], and the primitives used are characters. In our model, $P_{\omega_j}(x_i)$ is controlled by the stroke model.

The word is synthesized using the individual primitive classes, and the distribution of samples within the classes. The synthesis proceeds as follows:

- Given a text word, create a sequence of stroke classes that constitute the handwriting equivalent of the input word. This information was learnt during the training of the layout model.
- For each stroke class, we select/generate a sample stroke using the stroke model for the writer under consideration.
- The layout model is used to arrange the sample strokes to generate the final word.
- The word could be rendered in appropriate form, depending on the application or could be passed to the next phase in applications such as retrieval and training of recognizers.

Stroke Selection: The stroke selection module selects a sample stroke from the set of training samples in the case of normalized template model. For the mean trace model, we generate a sample stroke based on the learnt distribution of the sample points within the stroke. One can introduce variations in the synthesis by generating random samples from each sample distribution. Alternately, the mean of the sample points would give the most likely stroke sample for each stroke class.

Stroke Alignment: Given two consecutive primitives, We compute the most likely direction between the two classes using the layout model. To introduce writing variations, one could generate random sample from the spatial relation distribution that was estimated in the training phase.

The probability of the synthesized word with m stroke classes can be computed according to the stroke and layout models as follows:

$$P(word) = p(x_1|\omega_1)\prod_{i=2}^{m} p(r_i, \theta_i|\omega_{i-1}\omega_i)p(x_i|\omega_i),$$

where r_i and θ_i are the distance and direction, generated by the layout model for the i^{th} stroke sample.

6. Experiments and Discussions

We verified our algorithm in various Indian scripts such as: i) Telugu, which has one of the most complex layouts among all of the Indian languages, ii) Hindi, which has shirorekha, iii) Malayalam, which has long and complex strokes, and iv) Bangla, which is similar to Hindi and v) Tamil, which has the smallest set of alphabets in Indian languages. The primary dataset for the experiments were collected in Telugu. We collected 15 pages of data from 5 different writers. The writers were chosen from varied backgrounds, like script familiarity, educational qualification, age and gender. Each of these writers wrote Telugu text on an IBM Crosspad that was used to collect the online handwriting data. The data contains approximately 14000 strokes and over 2000 words.

The experimental framework consists of i) an annotation tool that can annotate the handwritten data at the stroke level, ii) a handwriting model and synthesis module and iii) an ITRANS-based encoding module for processing the text input. The data was annotated manually using the annotation toolkit [14] in ITRANS [15]. In this toolkit, handwritten data can be displayed simultaneously with the annotation, which is dynamically updated as the user types the annotation.

Modeling and Synthesis of Handwriting: Figure 3 shows the handwritten Telugu word EdainA (means *anything*) written by two different writers. As it is evident from Figure 3, the second writer's handwriting (Figure 3 (b)) is readable and neat when compared to that of the first writer (Figure 3 (a)) whose handwriting sample is composed of several pen lifting movements, stroke discontinuity, slant and many deformities. Also note the fact that the stroke order is not unique for every writer for a given word and sometimes even the same writer has different stroke order when writing the same word.

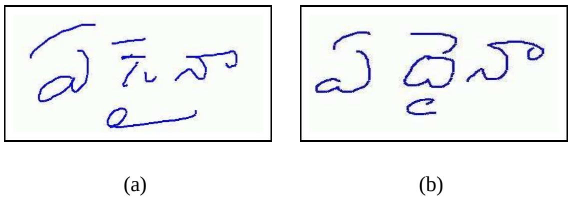

Figure 3. Telugu Word EdainA written by (a) writer 1, (b) writer 2.

Figure 4 shows the original samples of the Telugu word EdainA from two different writers. As can be seen, the synthesized word for two different writers (Figure 4 (b) and (d)) is very similar when compared to their original forms (Figure 4 (a) and (c)). So our model is able to generate natural and realistic words that are very close to the original ones.

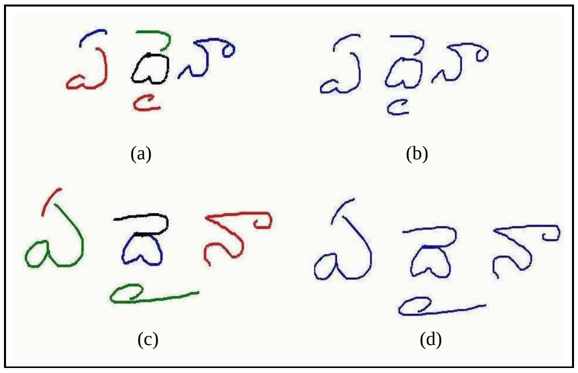

Figure 4. Telugu Word EdainA written by (a) writer 1, and (b) synthesized for writer 1. Same word EdainA written by (c) writer 2, and (d) synthesized for writer 2

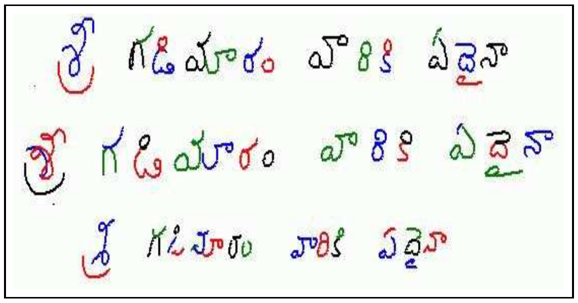

Figure 5. Sample words synthesized for three different writers

Figure 5 shows the synthesis results of the same set of words written by three different writers. Characteristics such as the inter-character spacing, relative size of loops and other matras (diacritical marks) of the three writers vary greatly. The mean trace model allows us to generate variations in handwriting of a specific writer, while maintaining the characteristics traits of the individual.

During synthesis the user input is typically in the form Unicode or ITRANS [15] encoded text, which are the two popular encodings for Indian language scripts. The input encoder converts the ITRANS or Unicode text string into a sequence of stroke classes. The specific stroke classes that are generated depend on the character in the input as well as the writing style of the writer under consideration. The stroke selector module generates a sequence of strokes using the stroke model that are needed to generate the input text. Stroke alignment module is then used to compute the spatial positioning of the generated strokes based on the information from the alignment model and finally the word is rendered.

Multi-Script Synthesis Our synthesis model allows for generation of handwriting in multiple scripts of Indian languages for a given input word. Indian language scripts are phonetic-alphabetic in nature and they share a common set of alphabets. Hence the ITRANS representation of a word, especially proper names, are identical in all the Indian scripts. Given an input word in ITRANS, one can generate the corresponding strokes in any of the Indian scripts and use the corresponding language model to generate the handwriting for a particular user. This capability is essential for applications such as cross-lingual search, where one would like to search for a word in different scripts, simultaneously.

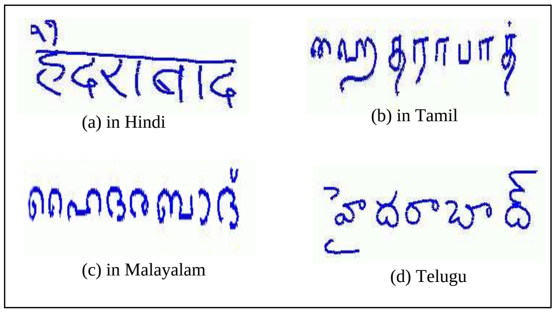

Figure 6. The word *haidaraabaad* synthesized in various Indian languages

In spite of the success of the approach described in this paper, we need to investigate a set of issues in order to develop the proposed framework into a complete synthesizer for Indian languages. They include:

1. **Writer Independence:** The current framework supports a writer specific synthesis scheme. To extend this to a writer independent synthesis module, one needs to model an average writer. This could be learnt using samples from a large set of writers.

2. **Stroke Order:** The order of strokes can change depending on the writer as well as within a writers samples. The current framework assumes a single mapping from an ITRANS encoded text to the stroke sequence. This needs to be extended in order to handle multiple stroke orders.

3. **Complex Stroke Model:** The stroke model as described before is a simple one. A more complex stroke model can be used that encapsulate the variations in a writers samples more closely.

4. **Quantifying Performance:** Quantitative evaluation of the performance is important in addition to visual similarity of synthesized handwriting. Currently we are not aware of any specific evaluation metric for this purpose. Hence the derivation of a quantifying metric will help in comparing the various synthesis algorithms.

7. Conclusions and Future Work

In this paper, we have looked into various issues associated with the Indian scripts. Synthesis results using the model learnt from training samples produces natural looking words. The generation model was also verified using retrieval experiments. The model also allows for multiple script synthesis for Indian languages for a given input word. The individual modules such as the stroke model and the layout model could be further enhanced by incorporating more complex deformation models. One of the interesting directions that we are currently pursuing is the study of stroke shape variations when it is in the proximity of other strokes or when the position of the stroke in the word changes.

References

[1] S. Srihari, H. Cha, S.-H. Arora, and S. Lee, “Individuality of handwriting,” *Journal of Forensic Sciences*, vol. 47, no. 4, pp. 1–17, 2002.

[2] Isabelle Guyon, “Handwriting synthesis from handwritten glyphs,” in *Proc. of IWFHR*, pp. 140–153, 1996.

[3] Tamas Varga and Horst Bunke, “Generation of synthetic training data for an HMM-based handwriting recognition system,” in *Proc. of ICDAR*, pp. 618–622, 2003.

[4] Muriel Helmers and Horst Bunke, “Generation and use of synthetic training data in cursive handwriting recognition,” in *Proc. 1st Iberian Conf. on Pattern Recognition and Image Analysis*, pp. 336–345, 2003.

[5] Y. Zheng and D. Doermann, “Handwriting matching and its application to handwriting synthesis,” in *Proceedings of ICDAR*, pp. 861–865, 2005.

[6] D. Lopresti and A. Tomkins, “On the searchability of electronic ink,” in *Proc. of IWFHR*, pp. 156–65, 1994.

[7] Wacef Guerfali and R. Plamondon, “The delta lognormal theory for the generation and modeling of cursive characters,” in *Proc. of ICDAR*, pp. 495–498, 1995.

[8] Y. Singer and N. Tishby, “Dynamical encoding of cursive handwriting,” in *Proc. IEEE Conf. CVPR*, pp. 341–346, 1993.

[9] R. Plamondon, “A Kinematic theory of rapid human movements, Part I. Movement representation and generation,” in *Biological Cybernetics*, vol. 72, pp. 295–307, 1995.

[10] Jue Wang, Chenyu Wu, Ying-Qing Xu, Heung-Yeung Shum, and Liang Ji, “Learning-based cursive handwriting synthesis,” in *Proc. of IWFHR*, pp. 157–162, 2002.

[11] S. Kondo, “A model of handwriting process and stroke structure of character fi gures,” in *Computer Recognition and Human Production of Handwriting*, pp. 119–130, 1989.

[12] L.R.B. Schomaker, A.J.W.M Thomassen, and H.L. Teulings, “A computational model of cursive handwriting,” in *Computer Recognition and Human Production of Handwriting*, pp. 119–130, 1989.

[13] D.-H. Lee and H.-G. Cho, “A new synthesizing method for handwriting Korean scripts.,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 12, no. 1, pp. 46–61, 1998.

[14] A. Bhaskarbhatla, S. Madhavanath, M. Pavan Kumar, A. Balasubramanian, and C. V. Jawahar, “Representation and Annotation of Online Handwritten Data,” in *Proc. of IWFHR*, pp. 136–141, 2004.

[15] “http://www.aczoom.com/itrans/,”