A Study on Top-down Word Image Generation for Handwritten Word Recognition

Eiki Ishidera Daisuke Nishiwaki

Multimedia Research Labs. NEC Corporation
4-1-1, Miyazaki, Miyamae-ku, Kawasaki, 216-8555, Japan

Abstract

This paper describes a top-down word image generation model for holistic handwritten word recognition. To generate a word image, it uses likelihoods based, respectively, on a linguistic model, a segmentation model, and a character generation model. In the recognition process with respect to a given input image, it first generates, for each word in a dictionary of possible words, a word image that approximates as closely as possible the input image. The model next calculates distance values between each generated word image and the input image and selects for recognition that generated word image having the smallest distance value. The proposed method has been evaluated in an experiment using handwritten word images, and results show it to be effective for use in handwritten word image recognition.

keyword: holistic handwritten word recognition, word image generation model, level-building-like dynamic programming

1. Introduction

Recognition of handwritten word images is extremely important in document processing. The mail sorting machines used in post offices are, for example, one of the most important applications of handwritten word recognition.

One of the authors of this study has recently proposed a word image generation model for holistic word recognition [1] and applied it to use on typewritten museum archive cards. To generate word images, it uses likelihoods that are based respectively on a linguistic model, a segmentation model, and a character generation model. The method is robust with respect to such problems as faded, broken, or touching characters, but it has not yet been applied to use on cursive script, in which most of the characters in a word may be connected to neighbouring characters, making segmentation especially difficult.

While HMM approaches [2][3] are commonly used for recognizing cursive script, they suffer from the lack of effective evaluation of the potential correctness of the partitioning upon which they depend. Lethelier [4] and Gilloux [5] have proposed HMM-based probabilistic models which do represent the potential correctness of segmentation, but in [4], it is necessary to generate segmentation hypotheses, and in [5] it is necessary to apply the Baum-Welch reestimation algorithm in order to evaluate the potential correctness of segmentation. These requirements are a significant drawback.

In contrast to these HMM methods, all of which are based on bottom-up models, here we apply the top-down method in [1] to handwritten word images. In the recognition process with respect to a given input image, it first generates, for each word in a dictionary of possible words, a word image that approximates as closely as possible the input image. The model next calculates distance values between each generated word image and the input image and selects for recognition that generated word image having the smallest distance value. The proposed method has been evaluated in an experiment using handwritten word images, and results show it to be effective for use in handwritten word image recognition.

2. Word image generation model

The word image generation model consists of four parts: a word generator, a style generator, a character locator, and a character image generator (Fig. 1).

The word generator generates a word to be written, representing it as $\mathbf{\Omega}$. The probability of $\mathbf{\Omega}$ being generated may be set as a constant.

$$P(\mathbf{\Omega}) = P(\omega_1, \omega_2, \cdots, \omega_n) = Const.$$

The word $\mathbf{\Omega}$ consists of n characters, and ω_i indicates the i-th character-category. Since we are considering the case in which the word $\mathbf{\Omega}$ is given, we can assume that $\mathbf{\Omega}$ is independent of any other information, and thus, the probability of $\mathbf{\Omega}$ being generated may be set as a constant. The

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE

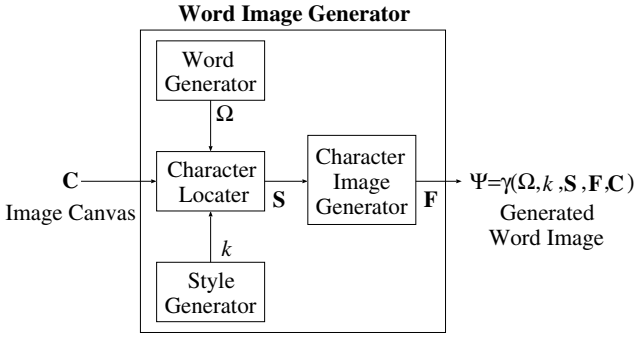

Figure 1. Word Image Generator

style of writing will also be specified. We assume here that style is independent of any other factors, which means that the probability of a style "k" being generated can be set as constant.

$$P(k) = Const.$$

Next, let $\boldsymbol{C}$ be a 2D image canvas of given width and height. When word $\boldsymbol{\Omega}$, style k, and image canvas $\boldsymbol{C}$ have been determined, the character locater is able to represent the location and the size of the characters in $\boldsymbol{\Omega}$ as $\boldsymbol{S} = \{s_1, s_2, \cdots, s_n\}$. Here, s_i is the i-th segment, represented in terms of size and coordinate parameters.

$$s_i = \{w_i, h_i, x_i, y_i\}$$

A segment s_i gives the position and size of the character-image corresponding to the i-th character-category ω_i. Here, w_i and h_i are the width and the height of segment s_i, respectively, and (x_i, y_i) indicates the center of segment s_i.

We assume here that the i-th segment s_i is correlated to five elements: the (i-1)-th segment s_{i-1}, the i-th character-category ω_i, the (i-1)-th character-category ω_{i-1}, style k, and image canvas $\boldsymbol{C}$. We also assume that the size of image canvas $\boldsymbol{C}$ roughly gives the size of the word image to be generated. We can express the probability of $\boldsymbol{S}$ as follows:

$$P(\boldsymbol{S} \mid \boldsymbol{\Omega}, k, \boldsymbol{C}) = P(s_1 \mid \omega_1, k, \boldsymbol{C}) \prod_{i=2}^{n} P(s_i \mid s_{i-1}, \omega_{i-1}, \omega_i, k, \boldsymbol{C})$$

The character image generator fills each segment prepared for the writing of a given word with an appropriate character image in style k. Here, we assume that the character-image is generated independently and individually in accord with the probability density function law alone. Thus, we can write the probability of $\boldsymbol{F} = \{f_1, f_2, \cdots, f_n\}$ as follows:

$$P(\boldsymbol{F} \mid \boldsymbol{\Omega}, k, \boldsymbol{S}, \boldsymbol{C}) = \prod_{i=1}^{n} P(f_i \mid \omega_i, k)$$

Here, f_i is the i-th character-image, corresponding to ω_i and s_i.

We can obtain the probability of a generated word image $\boldsymbol{\Psi}$ as follows:

$$P(\boldsymbol{\Psi} \mid \boldsymbol{C}) = P(\boldsymbol{\Omega})P(k)P(s_1 \mid \omega_1, k, \boldsymbol{C})P(f_1 \mid \omega_1, k) \prod_{i=2}^{n} P(f_i \mid \omega_i, k)P(s_i \mid s_{i-1}, \omega_{i-1}, \omega_i, k, \boldsymbol{C})$$

Figure 2 shows an example of a case in which the word $\boldsymbol{\Omega}$ (=$\{$C,o,r,k$\}$) and image canvas $\boldsymbol{C}$ are given. In this figure, the first character-category ω_1 (= "C"), the first segment s_1, and the first character-image f_1 are all indicated.

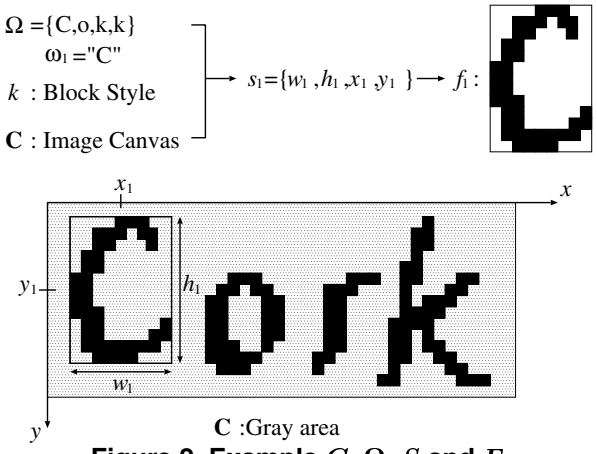

Figure 2. Example C, Ω ,S and F

3 Word recognition algorithm

Let us next consider how to use the proposed word image generation model to recognize an input image $\boldsymbol{\Gamma}$. While Setlur [6] and Mao [7] have proposed a method for generating variant samples of a handwritten cursive string, using generated samples as training samples, here we can directly compare generated word images with input images. In a top-down approach, the recognition process can be carried out by searching for the generated word image $\hat{\boldsymbol{\Psi}}$ on the basis of the equation below:

$$P(\hat{\boldsymbol{\Psi}} \mid \boldsymbol{\Gamma}) = \max_{\boldsymbol{\Psi}} \frac{P(\boldsymbol{\Gamma} \mid \boldsymbol{\Psi})P(\boldsymbol{\Psi})}{P(\boldsymbol{\Gamma})}$$

When maximizing the above equation, we can neglect the term $P(\boldsymbol{\Gamma})$ and obtain the equation below:

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE

$$\max_{\boldsymbol{\Psi}} P(\boldsymbol{\Gamma} \mid \boldsymbol{\Psi})P(\boldsymbol{\Psi}) = \max_{\boldsymbol{\Omega},k,\boldsymbol{S},\boldsymbol{F}} P(\boldsymbol{\Gamma} \mid \boldsymbol{\Psi})P(\boldsymbol{\Psi}) \quad (1)$$

Since word image $\hat{\boldsymbol{\Psi}}$ is generated from the parameter values of $\boldsymbol{\Omega}$, k, $\boldsymbol{S}$ and $\boldsymbol{F}$, the recognition process will be carried out by searching for parameter values that give a maximum value to Equation (1).

To obtain such parameter values, we could generate all possible word images, trying all possible parameter values, and then compare each generated word image to the input image $\boldsymbol{\Gamma}$, but this approach would, of course, be very computationally expensive. Here, we will instead consider a level-building-like dynamic programming (LDP) algorithm to obtain those values.

3.1 Level-Building-like dynamic programming algorithm

Let us consider a situation in which a word $\boldsymbol{\Omega}$ and a style k are specified. The generated word image $\boldsymbol{\Psi}$ for which $\boldsymbol{\Omega}$ and k are specified will consist of segmentation information and character-images. We can represent a generated word image consisting of N characters as follows:

$$\boldsymbol{\Psi} = s_1 f_1 \oplus s_2 f_2 \oplus \cdots \oplus s_N f_N$$

Here, $\oplus$ is the operation of overwriting the black pixels of image f_i onto the portion of image canvas $\boldsymbol{C}$ specified by s_i. We omit overwriting when the pixel value of the template to be overwritten has a lower value than the value already written on image canvas $\boldsymbol{C}$.

Here, we assume that the size of input image $\boldsymbol{\Gamma}$ is equal to the size of image canvas $\boldsymbol{C}$. When we assume the distribution of the term $P(\boldsymbol{\Gamma} \mid \boldsymbol{\Psi})$ to be Gaussian, Equation (1) can be maximized as follows:

$$\max_{\boldsymbol{\Psi}} P(\boldsymbol{\Gamma} \mid \boldsymbol{\Psi}) = \min_{\boldsymbol{S},\boldsymbol{F}} d(s_1 f_1 \oplus s_2 f_2 \oplus \cdots \oplus s_N f_N, \boldsymbol{\Gamma})$$

Here, $d(\alpha, \beta)$ represents the city-block distance between α and β.

Let $\boldsymbol{\psi}_i$ be an image which includes i characters on image canvas $\boldsymbol{C}$. For example, if the word to be written down consists of 5 characters, $\boldsymbol{\psi}_3$ will include the first 3 characters, but it will not yet include the 4th and the 5th characters.

We can express $\boldsymbol{\psi}_1$ as:

$$\boldsymbol{\psi}_1(s_1, f_1) = s_1 f_1 \quad (2)$$

With the level-building-like dynamic programming algorithm, $\boldsymbol{\psi}_i$ $(i \neq 1)$ can be estimated using the equation as follows:

$$\boldsymbol{\psi}_i(s_i, f_i) = \min_{s_{i-1}, f_{i-1}} \{d(\boldsymbol{\psi}_{i-1}(s_{i-1}, f_{i-1}) \oplus s_i f_i, \boldsymbol{\Gamma})\} \quad (3)$$

In this case, $s_{i-1} = (x_{i-1}, y_{i-1}, w_{i-1}, h_{i-1})$ and f_{i-1} satisfy the following limiting conditions:

$$\begin{aligned} (x_{i-1}, y_{i-1}) &\in \boldsymbol{A}_{i-1} \\ (w_{i-1}, h_{i-1}) &\in \boldsymbol{R}_{i-1} \\ f_{i-1} &\in \boldsymbol{T}_{\omega_{i-1}} \end{aligned}$$

Here, $\boldsymbol{A}_{i-1}$ is the area within which segment center (x_{i-1}, y_{i-1}) is to be moved in order to find the parameter values which give the minimum distance value in Equation (3). $\boldsymbol{R}_{i-1}$ is the range over which segment size is to be varied. $\boldsymbol{T}_{\omega_{i-1}}$ is a set of templates corresponding to character-category ω_{i-1}. Limiting the area to $\boldsymbol{A}_{i-1}$ and the range to $\boldsymbol{R}_{i-1}$ allows us to reduce the time required to find the best parameter values. We assume that the probabilities of (x_{i-1}, y_{i-1}) and (w_{i-1}, h_{i-1}) are constant in, respectively, area $\boldsymbol{A}_{i-1}$ and range $\boldsymbol{R}_{i-1}$, and that outside of this area and range they are zero.

To obtain the best f_i, we use multiple templates to represent various possible f_i patterns. Each template image is manually extracted from a training dataset, and the number of templates used varies depending on character-category. About 100 templates are used, for example, for character-category "o", while only 2 are used for "q." We assume that the probabilities for the various f_i character-images are constant. Any given f_i state can be represented in terms of the serial number of the template currently being used.

In word recognition with respect to a given word image, the algorithm will be applied to all possible words in the dictionary and all defined styles.

3.1.1 Estimating $\boldsymbol{A}_1$ and $\boldsymbol{R}_i$

To estimate area $\boldsymbol{A}_1$, we first find the left side edge (x_{min}) of the input image $\boldsymbol{\Gamma}$. Next, we find the uppermost edge of that portion of $\boldsymbol{\Gamma}$ which lies within the area extending from x_{min} to $x_{min} + w_s$, where w_s represents the standard width of a character. This upper edge is referred to as y_{min} (see Fig. 3).

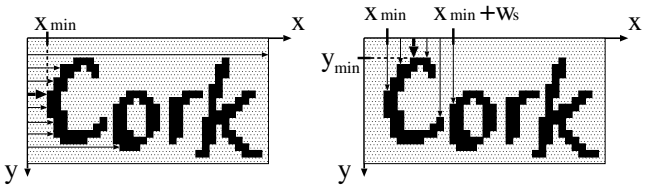

Figure 3. Finding initial location

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE

We can roughly estimate the location of the center of the first character $(\hat{x}_1, \hat{y}_1)$ on the basis of the position (x_{min}, y_{min}) and the standard width and height of a character. Next, we determine the area within which the centers of ω_1 templates are to be moved in searching for best-matching placements. We can consider this area to be $\pm\mu$ pixels horizontally and $\pm\nu$ pixels vertically from the initial position $(\hat{x}_1, \hat{y}_1)$; in this paper, we set (μ, ν) at $(11, 11)$.

With respect to the possible range of segment size $\boldsymbol{R}_i$, we can set the maximum height of segment s_i to be the height of the image canvas, the maximum width to be 150% of image canvas height, and the minimum width and height both to be 20% of image canvas height. We then divide into ζ sections the difference between the maximum and minimum width values, and divide the same difference for height values into η sections. We also search for those width and height values which will yield the minimum distance value in Equation (3). Here, we set (ζ, η) at $(12, 12)$.

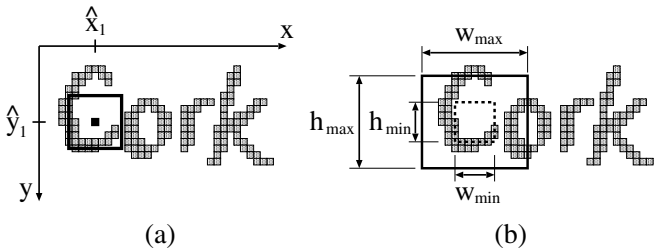

Figure 4. Search area and range

In Figure 4 (a), the black pixel shows the location $(\hat{x}_1, \hat{y}_1)$, and the box surrounding it shows the area $\boldsymbol{A}_1$ within which the centers of templates are to be moved. In Figure 4 (b), the minimum size of segment s_i is shown as a dashed-line box and the maximum as a larger, solid-line box.

3.1.2 Estimating $\boldsymbol{A}_i$ (for $i \neq 1$)

Area $\boldsymbol{A}_i$ (for $i \neq 1$) is determined as follows. First, we calculate an initial-estimate location and an initial-estimate size for the previous character, $\tilde{s}_{i-1} = (\tilde{x}_{i-1}, \tilde{y}_{i-1}, \tilde{w}_{i-1}, \tilde{h}_{i-1})$:

$$\tilde{s}_{i-1} = \underset{s_{i-1}}{\operatorname{argmin}}\{d(\boldsymbol{\psi}_{i-1}(s_{i-1}, f_{i-1}), \boldsymbol{\Gamma})\} \qquad (4)$$

Here, $s_{i-1} = (x_{i-1}, y_{i-1}, w_{i-1}, h_{i-1})$ and f_{i-1} satisfy the limiting conditions described in the previous section.

Next, we roughly estimate the location of the i-th character $(\hat{s}_i)$ on the basis of the initial-estimate location and size of the previous character $(\tilde{s}_{i-1})$, as calculated using Equation (4), and of the character-categories ω_{i-1} and ω_i.

We can roughly classify the font metrics of characters into two classes: the "ascender class" (AC) and the "non-ascender class"(NAC). Upper-case characters are classified as ascender class, as are the lower-case characters "b,d,f,h,i,j,k,l,t". The non-ascender class includes "a,c,e,g,m,n,o,p,q,r,s,u,v,w,x,y,z".

We can roughly predict the location of the i-th character's upper lefthand corner (x'_i, y'_i) from the location of the previous character's upper lefthand corner and the respective categories of previous character ω_{i-1} and character ω_i. The relationship between (x'_{i-1}, y'_{i-1}) and (x'_i, y'_i) is as follows:

$$x'_i = x'_{i-1} + \hat{w}_{i-1}$$

$$y'_i = \begin{cases} y'_{i-1} + \Delta y & \text{if } \omega_{i-1} \text{ is "AC" and } \omega_i \text{ "NAC",} \\ y'_{i-1} - \Delta y & \text{if } \omega_{i-1} \text{ is "NAC" and } \omega_i \text{ "AC",} \\ y'_{i-1} & \text{otherwise.} \end{cases}$$

Here, we set the value of Δy at 5.

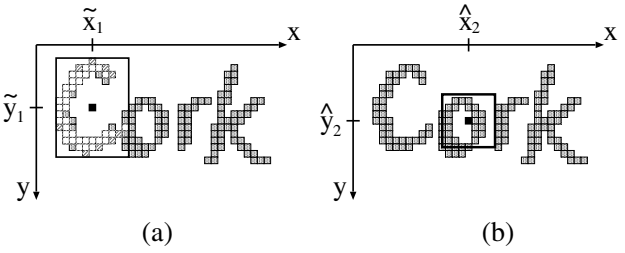

Figure 5. Initial-estimate segment and search area

Figure 5 (a) shows the location and size of the first segment $(\tilde{s}_1)$ as roughly estimated with Equation (4). In Figure 5 (b), the black pixel shows the initial-estimate location $(\hat{x}_2, \hat{y}_2)$, and the box shows area $\boldsymbol{A}_2$.

As we did in the previous section, we can now calculate the parameters which give the minimum distance value in Equation (3), and we can continue the process until reaching the end of the word $(i = n)$.

Here, determing the area of $\boldsymbol{A}_i$ is similar to the process of slant limitation in DP matching. If the various areas of $\boldsymbol{A}_i$ have been determined with sufficient accuracy, i.e. each contains the actual center of its corresponding character, it will be possible to obtain a global minimum distance image. If they have not, however, a global minimum distance value will be impossible to obtain.

4 Evaluation

In order to evaluate the method's effectiveness, it was needed to generate a word image that approximates as closely as possible an input image. For use on 12 handwritten word images, a word dictionary was provided which

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE

contained four words: “Cork,” “Cobh,” “Mallow,” and “Mullen.” Samples of generated word images can be seen in Figure 6.

The proposed method correctly recognized all the images. As may be seen in Figure 6, it is able to generate word images which approximate input images well for both block style and cursive script.

Note that some of the closest-matching templates still contain characters that are significantly different from the originals (e.g., (2) “C,” (5) “C,” (8) “m,” (10) “m,” etc); for the method to be applied successfully in actual practice, a more varied set of templates would be reqired.

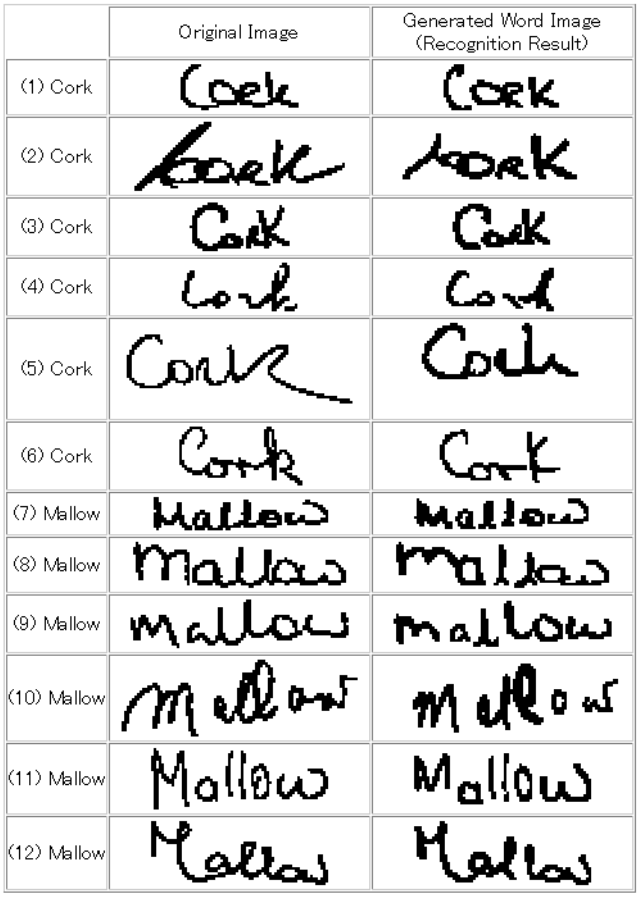

Figure 6. Generated Word Image Samples

5 Conclusion

We have proposed here a top-down word image generation model for use in holistic handwritten word recognition, which is here addressed as a parameterised search problem. Results show that the proposed method can be effectively applied to handwritten word image recognition.

With respect to future work, it will be very important to evaluate the proposed method on a far larger number of images. Further, since the proposed method requires an hour or more to generate a single word, it will also be very important to speed up its processing.

Acknowledgements

We are grateful to Prof. Andrew Downton and Dr. Simon Lucas for their very valuable guidance and encouragement.

References

[1] E. Ishidera et al.:”Top-down Likelihood Word Image Generation Model for Holistic Word Recognition”, DAS2002,2002.

[2] R. Plamondon, S.N. Srihari, “On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, No. 1, pp63-84, 2000.

[3] A. Vinciarelli, “A Survey on Off-line Cursive Word Recognition”, *Pattern Recognition*, Vol. 35, pp.1433-1446, 2002.

[4] E. Lethelier, M. Leroux, M. Gilloux:”An Automatic Reading System for Handwritten Numeral Amounts on French Checks”,ICDAR’95,1995

[5] M. Gilloux, B.Lemaré, M. Leroux:”A Hybrid Radial Basis Function Network/Hidden Markov Model Handwritten Recognition System”,ICDAR’95,1995

[6] S. Setlur, V. Govindaraju, “Generating Manifold Samples from A Handwritten Word”, *Pattern Recognition Letters*, Vol. 15, No. 9, pp.901-905, 1994.

[7] J. Mao, K. M. Mohiuddin, “Improving OCR Performance using Character Degradation Models and Boosting Algorithm”, *Pattern Recognition Letters*, Vol. 18, No. 11-13, pp.1415-1419, 1997.

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 **IEEE**