6 Conclusions

A segmentation method suitable for application to unconstrained discretely written words of variable length has been outlined. Word length estimation and the control structure utilize digit and character recognition results to help accurately segment word images. On discretely written words composed of digits, word length estimation is very accurate and a high satisfactory segmentation rate is achieved. Results on a wider class of automatically located words are understandably lower but, further research in character recognition and cursive script techniques will improve performance. The study on interaction with the recognition system will continue and larger tests will be conducted.

7 Acknowledgements

Xin Zhao contributed various recognition algorithms. The author would like to gratefully acknowledge the support os the Office of Advanced Technology of the United States Postal Service (USPS). Dr. Sargur Srihari and Dr. Jon Hull have provided valuable advice.

References

[1] L. Lam and C.Y. Suen, Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers, *Pattern Recognition*, 21:19-31.

[2] M. Sridhar and A. Badreldin, Context-directed segmentation algorithm for handwritten numeral strings, *Image and Vision Computing*, 5(1):3-9, February 1987.

[3] A. Pervez and C.Y. Suen, Segmentation of unconstrained handwritten numeric postal zip codes, *Proceedings of the 6th International Conference on Pattern Recognition*, 545-547.

[4] R. Fenrich and S. Krishnamoorthy, Segmenting diverse quality handwritten digit strings in near real-time, *Fourth USPS Advanced Technology Conference*, November 1990, 523-537.

[5] E. Lecolinet and J. Moreau, A New System for Automatic Segmentation & Recognition of Unconstrained Handwritten Zip Codes, *The 6th Scandanavian Conference on Image Analysis*, June 1989.

[6] E. Cohen, J.J. Hull, and S.N. Srihari, Reading and Understanding Handwritten Addresses, *Fourth USPS Advanced Technology Conference*, November 1990, 821-836.

[7] M. Sridhar and A. Badreldin, Recognition of isolated and simply connected handwritten numerals, *Pattern Recognition*, 19(1):1-12, 1986.

[8] J. Schürmann, A multifont word recognition system for postal address reading, *IEEE Transactions on Computers*, C-27(8):721-732, August 1978.

[9] R. G. Casey, Moment Normalization of Handprinted Characters, *IBM Journal of Research and Development*, 548-557, September 1970.

[10] Y. S. Chen and W. H. Hsu, A New Parallel Thinning Algorithm for Binary Image, *Proceedings of National Computer Symposium*, 295-299, 1985.

[11] T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville MD, 1982.

Teulings, H.-L., & Schomaker, L.R.B. (1992).
Un-supervised learning of prototype allographs in cursive script recognition using invariant handwriting features.
In J.-C. Simon & S. Impedovo (Ed.)
From Pixels to Features III (pp. 61-73), Amsterdam: North-Holland.
#

Unsupervised learning of prototype allographs in cursive-script recognition using invariant handwriting features.

Hans-Leo Teulings & Lambert R.B. Schomaker

Nijmegen Institute for Cognition and Information (NICI), University of Nijmegen, P.O.Box 9104, 6500 HE Nijmegen, The Netherlands.
E-mail: hlt@nici.kun.nl & Schomaker@nici.kun.nl; Fax: +31-80-615938.

Abstract

A cursive-script recognizer has to be trained using an extensive data base of a person's own handwriting. In order to train the recognizer, separate letters in their context of cursive script have to be identified first. An automatic procedure, which segments cursive script into letters before the recognizer has a chance to learn the letters, would facilitate learning of a new person's handwriting by the recognizer. In this paper various methods are presented to segment on-line recorded cursive script into letters. One of the methods is a stochastic method inspired by simulated annealing. This method is elaborated more extensively. According to this method begin and end strokes of the letters are randomly varied. Only those variations are accepted that would make identical letters have more similar begin and end strokes. Results show that the pure simulated-annealing procedure can be improved significantly by selecting mainly the worst outliers. The results suggest that this method may yield a valuable tool for isolating individual cursive-script letters, excluding occasional connection strokes, such that these letters can be learned by the recognizer.

Introduction

Two approaches exist for designing a handwriting recognizer allowing recognition of any kind of handwriting. The first approach is to implement a-priori knowledge about how each handwritten letter should look like, occasionally using a trial and error method (e.g., Simon & Baret, 1990; Ouladj et al., 1990). The second approach is to train the recognizer with handwriting produced by specific writers (e.g., Schomaker & Teulings, 1990). The latter training procedure would be relatively straight forward if the writers produce handprint or separate numbers (Guyon et al., 1991), but laborious if they are used to produce cursive script. The difficulty is caused by the fact that cursive script has to be segmented into letters before they can be learnt by the recognizer. This paper presents a number of methods to identify individual letters in cursive script in their natural context, prior to learning that particular handwriting.

Cursive script represents a less restricted domain than handprint. Therefore variability within and between writers is greatest in cursive script. In order to cope with individual differences of cursive script, a recognizer needs to be trained, using very large numbers of prototype letters per writer. However, these prototype letters should be produced in their usual context, rather than in isolation like in handprint, in order to obtain representative samples.

Of course, the problem of letter segmentation can be omitted by whole-word recognition, where the complete pattern of the word is matched with a finite number of patterns of each word in the recognizer's data base (Simon & Baret, 1990). A further advantage is that especially longer words may be recognized remarkably well. However, a disadvantage is the less efficient storage of large numbers of word prototypes because for each combination of letter variations a complete prototype word has to be stored. Furthermore, words not stored cannot be recognized at all even if it is written very clearly. Therefore, this approach is complementary to our procedure.

The procedure to follow is that of individual letter recognition. The advantage is that only a few prototypes per letter have to be stored. Any sequence of letters may be written and each letter may be written in various versions (allographs). The procedure of letter recognition in cursive script allows for more efficient storage of the writing patterns in terms of prototype letters and word lists, and allows recognition of clearly and unambiguously written patterns. However, a major disadvantage is that the cursive-script recognizer will identify many letters in overlapping stroke sequences. This causes an explosion of word hypotheses, which remains to be sorted out by a linguistic postprocessor (e.g., Wells et al., 1990). A linguistic postprocessor may be as simple as verifying which combination of letter hypotheses yields a word occuring in a word list or may echo the 'closest' word in a word list. Linguistic processing may even take place in parallel to the recognition and segmentation of the letters (Ouladj et al., 1990). Linguistic processing may resemble whole-word recognition but that is not true: also words not occuring in the word list may, in principle, be recognized, depending upon the reliance upon the word list.

Because we prefer the approach of letter recognition in cursive script, and still allow writers to produce their own handwriting, it is desirable to train the recognizer with many prototype letters in their context, occuring in the writers' handwriting. This does not turn out to be trivial. In order to understand methods to obtain prototype letters a small summary will be presented of the low-level processing of on-line cursive script.

Normalization and feature extraction

In most recognizers at least five phases can be discerned: acquisition, normalization, feature extraction, symbolic pattern recognition, and linguistic postprocessing. Of importance to this paper are normalization and feature extraction.

Normalization concerns global transformations of position, orientation, size, and slant which serves removing the non-essential variations, which are manipulated freely by the writer. Ideally, features should be virtually independent of the non-essential, noisy variations so that identical letters will have similar features. However, not all variations can be removed by normalization. For example, identical letters may be produced with topologically different shapes. Topologically distinct versions of the same letter are called allographs. The recognizer treats different allographs as different letters having the same ASCII representation.

Feature extraction is based on the notion that on-line handwriting is produced by a sequence of ballistic strokes. Thus a stroke can be defined by two consecutive absolute-velocity minima. Each stroke is characterized by a set of features: the feature vector. So a cursive-script word

consists of a sequence of feature vectors and an allograph consists of a subset of these vectors. An allograph may count 1 to 8 characteristical strokes. It may be noted that we assume that letter boundaries are only at stroke boundaries, although one can imagine that in cursive script a stroke may, in part, belong to a letter.

The stroke features being used in the present paper have been selected on the basis of their low signal-to-noise ratios across replications within a writer (Schomaker & Teulings, 1990). The success of the methods of automatic letter segmentation depends on the choice of the features (Teulings & Schomaker, in prep.). Features which are little dependent upon motor noise yield good results in the segmentation, and, of course, yield also consistent prototypes for the recognizer. The features used are:

(a) The vertical positions of the begin and end of a stroke and the path length of the stroke all scaled to the average x-height, and relative to the base line.

(b) The directions of the five, straight stroke segments between two subsequent points corresponding with the time moments

$$t = t_1 + (n / 5) * (t_2 - t_1)$$

where t_1 and t_2 are the time moments of begin and end of the stroke and n = [0, 1, ..., 5]. Here we explicitly use dynamic movement information. The rationale is that in equal time intervals the movement direction is changing a relatively constant amount (e.g., Thomassen and Teulings, 1985) such that each new segment adds an equal amount of information.

(c) The relative size of the enclosed area between the a stroke and the subsequent stroke, which is rather a visually salient feature.

(d) A pen up indicator, which shows the proportion of time that the pen is up during a stroke.

Words consisting of sequences of strokes, each one characterized by these feature sets or feature vectors, need to be segmented into allographs. This requires that the correct strokes are assigned to each allograph without using a-priori knowledge of the allographs. Selection of allographs would be trivial in neat handprint, where a-priori segmentation is possible. However, in cursive script segmentation into letters is a problem when the letters are not yet known.

Methods for automatical letter segmentation

Several methods exist to obtain cursive script letters in context before their recognition. We assume that a large corpus of cursive script (e.g., 1000 words of 6 letters on average), segmented into words, and each word's ASCII letter sequence are available. This is relatively easy for the person whose handwriting has to be learned and does not need to cost more than the writing time (plus typing time in some cases). We assume here that only lower case letters are used. The methods are listed from highly interactive and laborious to fully non-interactive.

Manual method

Each letter boundary in the whole corpus is pointed at manually, casually helped by automatic guessing procedures. This method is very laborious, introduces many ambiguities about which connection stroke belongs to which letter, and requires accurate error checking. A data set like this, once created, could train all on-line cursive-script recognizers for the prepared types of handwriting.

Incremental method

The letter boundaries of the first letters are determined manually and stored into the empty recognizer's data base of prototype allographs. Gradually, the recognizer starts identifying (or selecting the letter boundaries) by itself (Morasso et al., in prep.). This trial and error method is repeated until most letter boundaries are correctly identified. It may still be a laborious method depending upon the kind of script and the speed of learning.

Bootstrap method

The recognizer has somehow learned a type of handwriting and is presented with a new writer's handwriting, which is only slightly different. While recognizing or selecting letter boundaries correctly, a new data base of prototypes is built. However, it may be difficult to bridge larger differences between cursive script.

A-priori connection stroke identification

In Western cursive script connection strokes run mostly from the baseline via a cup-shaped stroke in up-forward direction or show local minima in the lower contour (Maier, 1986; Leroux et Salome, 1990). However, many exceptions exist (e.g., u, w, y).

Least-squares method

Assume that each letter has a specific number of strokes, mostly between 1 and 8. Then N (of the order of 1000) words written by M (of the order of 100) different letters yield N equations of M unknown parameters. The equations are normalized with respect to the known coefficients. The least-squares solution requires solving the normal equations. It should be realized that the number of strokes of a specific allograph and the number of connection strokes may vary between instances. Although it may be possible to obtain estimates of the most likely (integer) numbers of strokes for each letter, it is still unlikely that letters can be located in the middle of a word by just counting the numbers of strokes belonging to the previous or following letters.

Deterministic method

If all patterns starting with the same letter are collected the first and the following strokes will match well, or, at least, form a few clusters (one for each allograph). Beyond a certain stroke the feature vectors are inconsistent because they may belong to any connection stroke or letter that follows. Interesting is that the first letter may be written according to a few

distinct allographs so that a few clusters of feature vectors are found. The largest cluster belongs to the most frequently occuring allograph. After the last stroke of the first allograph the number of clusters suddenly increases (though not so dramatical if there is still another, less frequent allograph with more strokes having the same ASCII representation). This yields the number of strokes of the first allograph. Then, the number of strokes belonging to the first letter has been identified. Subsequently, the strokes of the identified allograph are removed from the corpus of strokes, and the corresponding letter from the corpus of words. The procedure is repeated until all data have been used (Teulings et al., 1990). The advantage of this method is that, in principle, different allographs may be identified. Furthermore, it can handle optional initial or connection strokes. In order to skip the connection stroke the patterns are optimally aligned by shifting some patterns by one stroke. However, the major disadvantage of this method is that several parameters and thresholds are involved which dramatically determine the outcome. Furthermore, this method is rather complex and therefore less attractive.

Pattern-frequency method

This method does not require the letter sequences of the words written in order to find regularities. It rather assumes that the feature vectors form discrete classes, representing the topological information of a stroke. Quantization of features may be less appropriate for continuously varying cursive script, though. On the other hand, it would also be a simple and straight forward method which again allows identification of various allographs for one letter.

Imagine that A, B, C, D, E, and F stand for the 6 different, discrete feature vectors occuring in a writing pattern of one word consisting of 16 strokes, e.g.:

A B C D E F B D C D E F A B A B

then for each stroke i the frequencies of occurrence of stroke i can be estimated as well as those of the patterns consisting of strokes i, and i+1, and of the pattern consisting of strokes i, i+1, and i+2, etc. can be estimated. If more words are included, then the patterns cannot be extended across word boundaries. Each stroke where these frequencies increase is the beginning of a new allograph. As can be seen in Table 1, the 2-sequence starting at the first A is (A B) and occurs 3 times, whereas (B C) is infrequent and (C D) occurs 2 times. Therefore at stroke sequence (C D) a new sequence of strokes starts. The length of the latter sequence is 4 as (C D E F) occurs 2 times as well but (C D E F B) is again infrequent. Therefore the units of the sequence are:

(A B) (C D E F) (B) (D) (C D E F) (A B) (A B)

If the stroke sequence represents the ASCII letter sequence "abcdbaa" then post-hoc it follows that:

a = (A B), b = (C D E F), and c d = (B) (D).

Notice that these frequent patterns have been identified without using information on the sequences of letters involved. Therefore, different allographs can be identified easily. Note

that from this limited example it is unclear whether (B) (D) forms, in fact, one pattern (B D) as they are all infrequent. The use of the 'ASCII representation helps to disambiguate. Furthermore, information on the letters helps to keep together the strokes belonging to letters having similar beginning strokes (e.g., a, c, d, g, q). Although this may seem an interesting method, it only works in discrete and relatively noise-free data. It cannot easily be applied in continuous and noisy features as in sloppy handwriting.

Table 1

Frequencies of the patterns of feature vectors as a function of pattern length help to identify relatively frequent stroke patterns, which may represent letters. (. means frequency 1; <blank> means frequency 0)

Feature vector	Pattern frequencies/length 1 2 3 4 5 6 7 8	Frequency	Length	Letter
A	3 3	3	2	a
B	4			
C	2 2 2 2	2	4	b
D	3 2 2			
E	2 2			
F	2			
B	4	1	1	c
D	3	1	1	d
C	2 2 2 2	2	4	b
D	3 2 2			
E	2 2			
F	2			
A	3 3 . .	3	2	a
B	4 . .			
A	3 3	3	2	a
B	4			

Structural-information theory method

Imagine that A, B, C, D, E, and F stand for 6 different, discrete feature vectors occuring in a writing pattern of one word consisting of 12 strokes, such as

A B C D E F E F A B C D,

then the stroke sequence can automatically be rewritten using the three structural-information operators. In the present example only one of the operators is needed:

S [(A B C D) (E F)]

where S [ ] is the append-symmetry operator (Van der Helm & Leeuwenberg, 1991). Of course, if more words are used, patterns cannot extend across word boundaries. Furthermore, only operations are admitted that do not change the within-pattern sequence of strokes. For example, the append-symmetry operation S [A B C D] produces the nonsense stroke sequence A B C D D C B A. Similarly to the previous method, it is sufficient to post-hoc assign letters to stroke sequences which appear as a unit. A side effect with some advantages is that these frequent patterns may be confounded with regularities of spelling. The program by the authors (PISA) allows several hundreds of elements processed at once, which is getting in the direction of sequences of the order of 1000 words of 6 letters of 4 strokes. However, strong discretization of strokes is probably not appropriate in cursive script, just as in the previous method.

Numerical optimization

Begin and end stroke of each letter are set to random initial values. Then a total 'cost' function is calculated, e.g., by the RMS difference between the average feature vectors within all groups of identical letters, or by the distance of their cells of a Kohonen topological stroke map (Schomaker & Teulings, 1990). Then the cost function is optimized by varying the letter boundaries.

Simulated annealing

Begin and end stroke of each letter are set to random initial values. Then an 'energy' measure is calculated for each begin and end stroke. The energy of a begin or end stroke is based upon the consistency of this stroke with the corresponding strokes of the other, identical letters. Then at random a begin and an end stroke are varied and the new total energy is calculated. In the beginning, when the 'temperature' is high, energy increases are accepted at a certain probability which is high for small increases and approaching zero for larger energy increases. Energy decreases are always accepted, of course. When temperature is decreased, jumps to higher energy levels become less and less probable to be accepted. The effect is that many combinations of begin and end strokes may occur but that the ones yielding low energies are the most likely. This method is conceptually not complicated and therefore attractive. It allows finding the lowest energy by adjusting the stroke boundaries but requires considerable computational power. This method resembles the previous method. The difference is that the numerical-optimization method follows a deterministic gradient descent whereas this method follows a stochastic approach. From the previous overview it appears that the simulated-annealing method seems promising and uses most of the information available.

The simulated-annealing method implemented

The simulated-annealing method introduced above will be explained in more detail. The following steps are performed:

Step 1. The sequence of strokes, in terms of feature vectors, is stored in one buffer and the sequence of corresponding letters in another. Initially begin and end strokes of each letter are randomly selected with the border condition that letter strokes do not overlap and do not cross word boundaries. For each different letter the following procedure is repeated.

Step 2. The feature vectors of all strokes representing all replications of a single letter are extracted and stored in a smaller buffer. Z-scores (i.e., values transformed to zero mean and unit variance) are calculated across the replications of all features in the first stroke, the second stroke, etc. Therefore, stroke features which depart from the mean across replications receive absolute values which are much larger than 1. Averaging the absolute z-scores of the feature vectors of the first, second, and following strokes of a single letter yields a distance measure of that letter from the mean sequence of feature vectors. This distance will be called the 'energy' of the begin strokes of the very letter.

In order to advance larger stroke sequences and to balance for the greater probability that the energy of the begin stroke is flattered by an incorrect end stroke, the energy has been multiplied with the inverse of the number of strokes, but this is probably a less relevant detail.

Step 3. Similarly, the 'energy' of the end strokes of each letter is estimated by averaging across the end strokes of the letters by reversing the stroke sequences. The whole procedure is therefore symmetrical for the begin and end strokes. For the sake of simplicity, the procedure will be explained in terms of begin strokes only.

Step 4. Thus, the energy estimates for the begin and end strokes of each replication of a specific letter and for all replications are calculated. The total energy of the begin and end strokes has to be minimized separately.

Step 5. Lowering the minimum energy of the begin strokes is done by randomly selecting a letter and changing its begin stroke. The change is at most one stroke position. Again there are some border conditions: no overlapping letter strokes and no letters across word boundaries. Furthermore the letter boundaries are geared towards increasing the number of strokes and, if not possible, to reducing the number of strokes.

Step 6. After a new begin stroke has been selected the total energy of the begin strokes is calculated again. The update will only be accepted if the total energy decreases or if:

$$\exp (- \text{Energy\_increase} / T) > r$$

where each time r is a random value between 0 and 1. For high values of parameters T, which represents the temperature, large energy increases are accepted. However, when T is lowered (annealed) it is less and less likely that energy increases are accepted.

Step 7. A new begin or end stroke is selected at a lower temperature T. Apart from the normal exponential temperature decrease, several other annealing schedules exist for lowering temperature T (e.g., Segura & Frias, 1990). We found that the annealing schedule is not as important as the starting temperature and the rate of temperature decrease. If the temperature is chosen too high the system rambles through a large part of the solution space. If the temperature drops too quickly, the system freezes at a less optimal configuration. In the present case low starting temperature, decreasing linearly to zero, has been selected.

Thus lower and lower values for the total energy can be reached. However, this standard

procedure could be made work more efficient after changing the procedure yielding the following modifications, which could be combined independently:

(a) Instead of updating any begin or end stroke with equal probability, strokes having larger departures from the mean obtain higher probabilities. Thus, when comparing the stroke patterns of all identical letters, the most outlying stroke boundaries are most likely to be changed.

(b) Instead of comparing the total energy, the energy of the letter stroke being updated is compared. This sounds reasonable if one realizes that the number of strokes is very high and that changing one stroke boundary may cause only a marginal effect upon the total energy. Thus, an updated letter boundary is more likely accepted if it does not yield a worse outlier. It does not seem quiet the same as comparing total energy at a lower temperature, which one would expect.

(c) Instead of lowering the temperature independently of whether the update is accepted or not, the temperature is lowered when an update has been accepted, which may occur after many trials. In order to prevent an infinite loop, the temperature is also lowered if no update could be achieved after so many trials that nearly every stroke must have been selected at least once.

In Figure 1 examples of two processing steps are represented where begin and end strokes are updated. Mainly the outlying strokes obtained high probability for updating and the resulting (relative) energies appear to be much lower.

```
A  exxl      xuu      llxel      ulux      uul
B  e]x-|x-|l]x-]u-|u-|l]l|x-|e]l|u-]l|u-|x-]u-]u-|l]
C  e] x] x|l]x-]u-| u|l]l] x|e]l|u--|l-|u]x|u-]u| l]
D                  ←↑
E  0  3  3 2 8  2   9 2 2  3 0 2 1   9  6 3 2  3  2
F  0  2  2 2 5  1   1 2 2  2 0 2 0   7  9 2 1  1  2
G  e| x| x|l]x-]u-|u-|l]l| x|e]l|u--] 1 u]x|u-]u] l]
H                                                ↑→
I   0  0  0 1  0  1  3 1 1  0 0 1   2 9  3 0  1 4  1
J   0  0  0 0  0  0  0 0 0  0 0 0   2 9  2 0  0  0 0
```

Figure 1. Representation of two processing steps where the begin and end strokes are updated. (A: The input words; B: The true begin and end strokes. Each column occupies one stroke. The begin stroke of a certain letter is indicated by the name of that letter. The end stroke, if different from the begin stroke, is indicated by] and the within-letter strokes are indicated by -; C: The current begin and end strokes; D: The currently selected begin stroke, indicated by ↑ and the accepted update, indicated by ← or →; E and F: The current and the new relative energies of the begin strokes, where 9 and 0 represent maximum and zero energy; G - J: The same as C - F but then for the end strokes.)

Results

Results are based on finding the optimal solution of simulated stroke data of 5 words of 3 to 5 letters using a 4-letter alphabet. The letters consisted of 2 or 3 strokes and no noise was added such that equal begin and end strokes have zero distance. The computational efforts of the procedure to achieve the perfect solution as a function of the proposed Modifications (a) - (c) are listed in Table 2. It can be seen that especially Modification (a) (select the outlying stroke before trying to update) saves a great deal of the computational resources. The other modifications improve efficiency if performed alone, but do not seem to help much in combination with Modification (a).

Table 2

Required VAX3100Station CPU seconds for perfect solution as a function of modifications (*) relative to the standard simulated-annealing procedure.

Modification (c)		*		*		*		*
Modification (b)			*	*			*	*
Modification (a)					*	*	*	*
CPU seconds	232	59	123	152	5.8	5.2	3.9	5.2

Suggested Improvements

When dealing with real-life cursive script, the results do not seem encouraging. This may be caused by the fact that cursive script still contains many symmetries. It seems difficult for the system to jump two strokes, by one at a time, in order to obtain a best fitting stroke. Runs where the possibility of jumping two strokes at a time was included, indicated that the system starts rambling around dramatically. Also the border condition of non-overlapping strokes causes the requirement that first one letter has to 'withdraw' before the adjacent letter can claim 'its' stroke. Runs, were the possibility of overlapping strokes was allowed showed again dramatical departures from the desired solution. Finally, it may be noted that knowledge has not at all been used to select the initial configuration of letter boundaries. The tendency emerges that the system is able to find only a minimum energy solution if the initial configuration was already nearly correct. An initial setting can be guessed with higher probability of correctness by using specific numbers of strokes and connection strokes for each cursive letter.

References

Guyon, I., Albrecht, P., le Cun, Y., Denker, J., Hubbard, W. (1991). Design of a neural network character recognizer for a touch terminal. Pattern Recognition, 24, 105-119.

Leroux, M., Salome, J.C. (1990). Reconnaissance de l'ecriture cursive: Une methode de segmentation en "lettre". In Bigre nr 68, Reconnaissance automatique de l'ecrit. Rennes: IRISA.

Maier, M. (1986). Separating characters in scripted documents. Proceedings of the Eight International Conference on Pattern Recognition, 2, 1056-1058.

Morasso, P. & Di Marco S., (in prep.). Interactive segmentation of cursive script into letters.

Ouladj, H., Petit, E., Lemoine, J., Gaudaire, M., & Lorette, G. (1990). A prediction-verification strategy for automatic recognition. In R. Plamondon, & G. Leedham (Eds.), Computer Processing of Handwriting (pp. 187-206). Singapore: World Scientific.

Schomaker, L.R.B., & Teulings, H.L. (1990). A handwriting recognition system based on the properties and architectures of the human motor system. In C.Y. Suen (Ed.), Frontiers in Handwriting Recognition (195-209). Montreal: CENPARMI, 1990. ISBN: 1-895193-00-1.

Segura, E.C., & Frias, B.C. (1990). Self-learning simulated annealing. International Joint Conference on Neural Networks, Vol. I, 463-467, Hillsdale NJ: Erlbaum.

Simon, J.C., & Baret, O. (1990). Handwriting recognition as an application of regularities and singularities in line pictures. In C.Y. Suen (Ed.), Frontiers in Handwriting Recognition (23-38). Montreal: CENPARMI, 1990. ISBN: 1-895193-00-1.

Teulings, H.L., & Schomaker, L.R.B. (in prep.), Invariant properties of handwriting motor programs to be employed in automatic cursive-script recognition. Paper submitted at 5th Handwriting Conference of the IGS. Motor control of handwriting. 1991, Tempe, Arizona.

Teulings, H.L., Schomaker, L.R.B. Gerritsen, J., Drexler, H., & Albers, M. (1990). An on-line handwriting-recognition system based on unreliable modules. In R. Plamondon, & G. Leedham (Eds.), Computer Processing of Handwriting (pp. 221-234). Singapore: World Scientific.

Thomassen, A.J.W.M., & Teulings, H.L. (1985). Time, size, and shape in handwriting: Exploring spatio-temporal relationships at different levels. In J.A. Michon & J.B. Jackson (Eds.), Time, mind, and behavior (pp. 253-263). Heidelberg: Springer.

Van der Helm, P., & Leeuwenberg, E. (1991). Accessibility: A criterion for regularity and hierarchy in visual pattern codes Mathematical Psychology, 35, 151-213.

Wells, C.J., Evett, L.J., Whitby, P.E., & Whitrow, R.J. (1990). The use of orthographic information for script recognition. In R. Plamondon, & C.G. Leedham (Eds.), Computer Processing of Handwriting (pp. 273-289). Singapore: World Scientific.

Acknowledgements

This research was supported by Esprit Project 5204, Pen And Paper Input Recognition Using Script ("Papyrus"). The authors wish to thank Peter Alfonso and Gerben Abbink for valuable comments.