hdisk:[hlt.text]trondls3.tex 30-mar-1990 15:05:38
Paper presented at the 4th International Graphonomics Society Conference, Trondheim, 24-26 July 1989.

AN ONLINE HANDWRITING-RECOGNITION SYSTEM BASED ON UNRELIABLE MODULES

HANS-LEO TEULINGS, LAMBERT R.B. SCHOMAKER, JAN GERRITSEN, HANS DREXLER, and MARC ALBERS
Nijmegen Institute for Cognition Research and Information Technology (NICI)
P.O. Box 9104, 6500 HE Nijmegen, The Netherlands

ABSTRACT

In automatic recognition of unrestricted handwriting the ambiguities can be solved by top-down processing. However, automatic systems never have access to the extended background knowledge available to human readers. In order to replace this higher-level information we need to improve the reliability of the bottom-up processing. A handwriting-recognition system can be split up into six discrete blocks: (1) digitizing, word segmentation, pre-processing, and segmentation into strokes, (2) normalization of global handwriting parameters, (3) extraction of features per stroke, (4) allograph recognition, (5) optional word hypothesization, and, in order to allow recognition (6) a learning phase. The present paper discusses the design of three of these processing blocks: normalization, allograph recognition, and learning and briefly specifies feature extraction. Normalization concerns orientation, size, and slant. However, various alternative algorithms can be chosen and some algorithms yield more reliable results than others. A mechanism is proposed that will, sooner or later, find the most appropriate normalization algorithms. Consequently, the features extracted from each stroke in the handwriting pattern will be more uniform within a writer and even between writers. In the recognition phase, handwriting patterns are segmented into allographs using an algorithm that can handle allographs with various numbers of strokes and with optional connection strokes between them. In order to teach the recognizer the allographs a method has been designed that builts non-interactively a lexicon of allographs by automatically discovering the allographs in a large corpus of cursive script.

1. Introduction

Handwriting cannot be neglected as a universal tool to store a wide variety of graphical information, such as personal notes containing non-standardized characters together with graphs. Although this material may sooner or later be entered into a computer for further processing, people might prefer a pen as it cannot replace mouse interaction or voice input. In order to develop methods that optimally employ the movement information of the pen we investigate the possibilities of on-line handwriting.

Thirty years ago automatic recognition of handwriting was considered an 'excercise' for automatic speech recognition (Harmon, quoted by Lindgren, 1965). Still, no cursive-script recognition system appeared appropriate in a sufficiently wide range of applications. The need for automatic handwriting recognition is high, though, also for entering non-western script into computers (e.g., Chinese or Kanji, Hiragana, Katakana, Korean, Brahmi, Devanagari, Tamil, Indian, Hebrew, Arabic). A useful recognizer should be able to deal also with digits, arbitrary character sequences in handprint and cursive script where the identification has to

be based upon the bottom-up processing and in more redundant character sequences where higher-level information is required to disambiguate the stroke pattern. User friendliness requires recognition of all kinds of rapid and slow, cursive writing and handprint recognition by a wide variety of subjects.

Two methods to enter handwriting data into the computer exist. The first is to optically scan the picture of a sample of handwriting that was produced off-line. The second is to record the movements of the pen tip on-line by means of a digitizer. We employ the latter method because it is a richer source of information which allows us to exploit our basic knowledge about the motor system. The lack of practical results from traditional approaches is probably caused by the limitations of off-line recognition. Recent developments in the field of digitizer-annex-displays and microcomputers enable us to employ on-line recognition methods. This approach allows us to exploit dynamic movement information. Also promising off-line cursive-script recognition methods are under development (e.g., Srihari & Božinović, 1987; Aoki & Yamaya, 1986), but we discuss here only on-line recognition of handprint and cursive script.

A recent overview of on-line handwriting recognition is presented by Tappert et al. (1988). They list four experimental cursive-script recognition systems, 10 experimental handprint recognition systems, and 15 commercial hand-print recognition systems. Most systems act in a restricted domain such as boxed handprint characters or allowing specific words only.

The usefulness of a handwriting interface is stressed in Brocklehurst (1988). To demonstrate the idea of "electronic paper" they provide a prototype handwriting-recognition system. In the frame of the Esprit Project 295, Wright (1988) designed a cursive-script recognizer with efficient low-level processing but at the cost of reliance upon a dictionary and higher-level linguistic processing. 94% of the characters of a data set of 112 people was correctly recognized, which compares well with the rate of human recognition without context. Higgins & Whitrow (1985) achieved good results using only a quadgram table based on a dictionary. In the same project Doster & Oed (1984) and Mandler (1989) developed a recognizer for handprint. Within the frame of Esprit Project 419 a recognizer for normal cursive script, based on knowledge of the motor system is under development (Teulings et al., 1987; Thomassen et al., 1988a; 1988b; Schomaker and Teulings, 1990; Morasso et al., 1990). The system consists of a parallel structure of redundant modules and includes an non-interactive learning phase. Ouladj et al. (1989) developed a cursive-script recognizer where characters are hypothesized from left to right and are verified in parallel using a tree-like representation of a word list. Using a word list of 100 different words 90% of the words by 5 writers were recognized correctly.

Leedham and Downton (1986) developed a shorthand recognizer but state that the performance is still poor due to the simplified nature of the algorithms without knowledge base and due to ergonomic deficiencies of the digitizer. Recognition of online Arabic script is discussed by Amin, who states that the bottom-up analysis is still insufficient such that complicated grammatical analyses are required due to the complexity and non-uniformity of Arab grammar. In USA, Ward & Blesser (1985) developed a commercial handprint recognizer. Tappert (1986) at IBM Watson Research Center is developing a cursive-script recognizer using elastic matching techniques for IBM PC/AT. Plamondon & Baron (1986) used a handwriting recognizer to input computer programs. Skrzypek & Hoffman (1989) presented a neural network for cursive-script recognition. In Japan, Kondo (1989) realized a recognizer of handprinted numerals based on a noisy character generator.

Our aim is to design a system based on knowledge of the motor system which is able to recognize unrestricted handwriting (Schomaker & Teulings, 1990). In such a manner a computer can be operated without using a keyboard, nor a mouse, or voice input (e.g.,

Higgins & Duckworth, 1989; Welbourn & Whitrow, 1989) as everything we want to do can be done using a pen.

In Teulings et al. (1987) we introduced a modular architecture for the low-level bottom-up analysis of handwriting of the so-called Virtual Handwriting System (VHS). A handwriting recognition system can be split up into six discrete modules, in accordance with Srihari, S.N., & Božinović (1987):

1. Digitizing, preprocessing and segmentation of cursive script into words and subsequently into sequences of discrete strokes, which are defined here as the segments of pen movement between two successive minima of the absolute velocity

2. Normalizing handwriting patterns

3. Stroke feature extraction

4. Allograph recognition

5. Optional word hypothesization using higher-level information

6. Learning phase in order to build the allograph lexicon required for recognition

Of course, the higher-level subsystems of linguistic processing are essential in cursive-script recognition (e.g., Higgins & Ford, 1989; Wells et al., 1989). However we want to emphasize the importance of the optimal low-level handwriting processing because we doubt whether the higher-level linguistic processing can restore the information lost by unreliable lower level processing as it does not have available the extended knowledge of an educated reader of a handwritten message. This holds especially if character sequences are used that do not at all occur in the lexicon of words.

The present paper discusses the current status of four of these processing blocks: normalization, feature extraction, allograph recognition, and learning.

2. Normalizing Handwriting Patterns

In order to extract the feature vector of the handwriting patterns, several normalization steps have to be performed (See Thomassen et al., 1988b, for an overview). The reason is that a sample of a person's handwriting contains various subject-specific parameters, like size or slant (e.g., Maarse et al., 1988). On top of that, the motor system is able to transform handwriting deliberately such as changing orientation, size or slant (e.g., Pick & Teulings, 1983). But all these parameters do not contain any information about the identity of the characters. Therefore, the handwriting patterns have to be normalized first.

A particular problem is that different writers may also have different cursive-script styles, and one writer may even employ various cursive-script versions of the same character. For that purpose we use the term "allograph" which is a subject-specific and context-specific topological structure of a character. By "character" we mean one symbol out of the series of upper and lower case keys that can be typed using a keyboard. We suggest that normalizing the handwriting patterns may reduce the between-writer differences to the extent that they are not caused by differences in topological structure.

The sequence of levels of normalization we find the most appropriate are: orientation, vertical position and size, and slant (Thomassen et al., 1988b). The rationale is that orientation needs to be normalized in order to estimate vertical size and position, and that vertical size and position, in turn, should be normalized in order to estimate slant. However, various algorithms for each normalization step are available and not every algorithm may be appropriate in all conditions. For instance, orientation of the word currently being processed

Figure 1: Figure 1. An example of an incorrectly estimated orientation of a word, which causes subsequently an incorrect estimate of the slant.

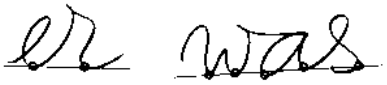

Figure 2: Figure 2. The pride of the baseline estimate of the second word is higher than that of the first word because of the additional evidence of several segmentation points consistent with the base line in the second word.

can be estimated by a global algorithm based on a whole page or by a local algorithm based on the very word only. Although the latter algorithm bases its estimate on less data, it may yield a more relevant orientation estimate especially in writers who produce words with varying orientations. Although several estimates for the orientation are available just one orientation parameter is needed in the above example. In general, each normalization module consists of three types of submodules: Several "estimators" of a normalization parameter (e.g., the orientation of a word), a "selector" of the optimal normalization parameter, and an "operator", which finally performs the normalization using the parameter selected. The selector is actually part of one larger unit ("Scheduler") which surveys all levels and chooses the most confident hypothesis to be evaluated next.

The problem is thus which estimate to adopt if they yield different values. Choosing the wrong estimate of the orientation, for instance, may have dramatic effects upon the subsequent slant estimation (See Figure 1). Taking the average of the estimates is probably not a good choice because one of the estimators may be totally wrong. Evaluating all possible combinations of estimates forms no solution either because it would yield an explosion of computations. The solution we propose is evaluating only the most confident, non-evaluated hypothesis.

In order to quantify the compound confidence of a certain estimator in a particular case, two sources of confidence are employed. The first source of confidence is a measure for the consistency of a specific instance of an estimate ("Pride"). For instance, the pride of a specific baseline estimate increases with the number segmentation points that are on a straight line. (See Figure 2 for an example). The second source of confidence is the cumulated reliability demonstrated from the past ("Prejudice"). In previous cases, the prejudice of an estimator was increased if its estimate has contributed to a successful final solution and was decreased otherwise. The extent of the increase or decrease of the prejudice depends on the value of the pride. For instance, if an estimator reports a high pride, but turns out to produce an incorrect estimate the prejudice needs to be reduced badly.

In order to obtain the compound confidence of a specific estimate, the prejudice and pride are multiplied (yielding the "optimism"). The rationale is that the confidence should only be high if both prejudice and pride are high. In case of several alternative estimator algorithms, the algorithm producing the estimate with the highest optimism is selected to

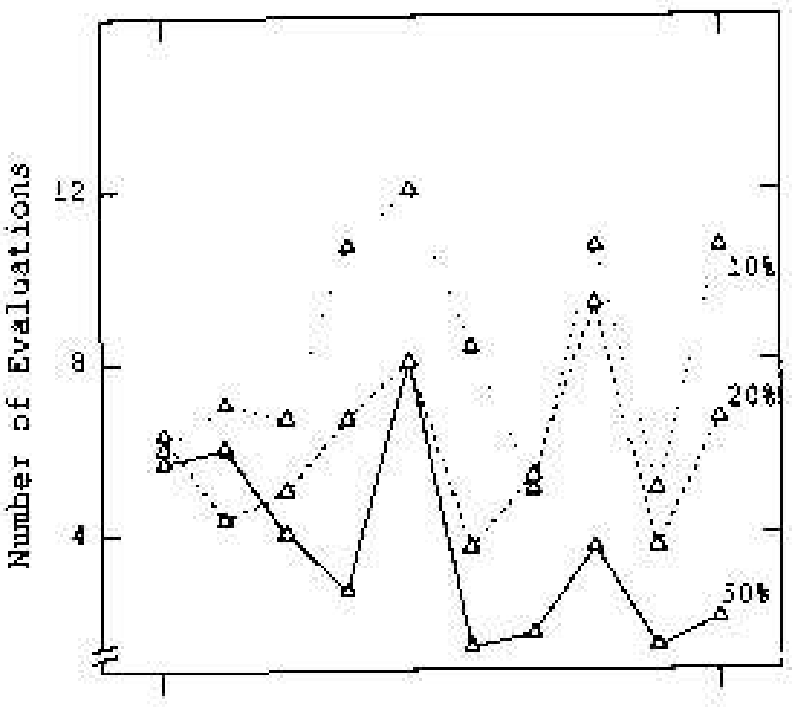

Figure 3: Figure 3. Simulation of the number of evaluations of an estimator (averaged across three sequential trials) as a function of number of trials. The estimators were perturbated with noise (the standard deviations of the added noise were 10, 20, and 50% of the exact estimate values, respectively). The more noise added, the less frequently an estimator was evaluated.

transform the data. If two or more estimators produced compatible estimates this provides additional evidence. The estimators are then combined in a group (called a "committee") and the estimators' optimisms are added up. In this way a committee may outrule a single estimator which happens to have an exceptionally high optimism.

Finally, the combined confidence score of a particular sequence of algorithms from the first processing step to the current processing step is needed. A sequence of estimates receives a confidence score which is equal to the product of all optimism scores of the individual estimates. At any time, the estimator, with the highest confidence score will be evaluated further. In this way, the system can backtrack to earlier levels where an erroneous decision was made and finally arrive at the most appropriate sequence of choices of the estimator algorithms. In this fashion, the system learns which estimators are in general reliable and which are not. Gradually, the most reliable estimators are adopted by default and the system gains efficiency by making fewer wrong choices of estimators (See Figure 3).

A problem of the present design is that it is rather conservative. If an estimator happens to be acceptable, its prejudice keeps on increasing. If at a later stage another kind of handwriting needs to be processed and if an alternative estimator yields better results, its pride may be insufficient to compete with the high prejudice of the first estimator. In fact it would have been better to require still some competition between estimators.

3. Feature extraction

Various sets of features per handwriting stroke have been used for different purposes. In the present application strokes are characterized by the following set of 14 features (Schomaker & Teulings, 1990) which make up the "feature vector":

(a) The vertical positions of the beginning (Y_b) and end of a stroke (Y_e) relative to the base line and the path length of the stroke (S) all scaled to the average body height, also called x-height, referring to the lower case x.

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

$$t = t1 + (n/5) * (t2 - t1),$$

where $t1$ and $t2$ are the time moments of beginning and end of the stroke and $n = [0, 1, ..., 5]$, i.e., $(\phi_1, \phi_2, \phi_3, \phi_4, \phi_5)$. 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 stroke segment adds an equal amount of new information. The two previous and the two following stroke segments (respectively, ϕ_{b4}, ϕ_{b5}, and ϕ_{e1}, ϕ_{e2}) are included as well in order to capture the stroke's context.

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

(d) A pen up indicator (P), which shows whether the pen is predominantly up or down during a stroke. It may be noted that strokes above the paper also count as strokes.

Feature vectors can also be used to define a distance measure between two strokes, called the "stroke distance". This distance is similar to the Euclidian distance of the normalized feature vector with the exception that the features referring to directions are not entered as the square difference but as the tangent of half of the difference. This is most consistent with our notion that strokes with opposite directions should have maximal distance. Finally, feature vectors are useful to define the average stroke across a set of strokes with small distances between them.

The number of features is high, indeed. This number can be reduced by building a two-dimensional self-organizing Kohonen network and then replacing the feature vector by the two network coordinates of the cell closest to the feature vector (Morasso, 1989; Schomaker & Teulings, 1990). This speeds up and improves the performance. Both representations are still under consideration.

4. Allograph Recognition

In the recognition phase the handwriting patterns in terms of sequences of stroke features are translated into characters using an "allograph lexicon". This allograph lexicon relates specific stroke-vector sequences to the corresponding character. The building of this allograph lexicon will be described in the section on the learning phase.

Two ways to do this job are under investigation. The first method is based on a neural-network approach (E.g., see Schomaker and Teulings, 1990). The second method, which is the one described here, is an algorithmic approach. The advantage of the latter method is that it optimally splits up the input sequence of strokes into more or less adjacent sub-sequences of strokes, where each sub-sequence forms one entry in the allograph lexicon. An established procedure to segment a sequence into adjacent sub-sequences is the Viterbi algorithm. However, the standard Viterbi algorithm is not applicable in the present case because it was designed for equal-length sub-sequences. In cursive script, however, the allographs have different numbers of strokes. Moreover, the algorithm has to detect which of the strokes are the connecting strokes between the allographs. The algorithm we developed to detect the allographs, is a modified version of the Viterbi algorithm.

Apart from the stroke distance introduced earlier, the algorithm requires two other distance measures, In order to quantify the resemblance of a piece of handwriting and an allograph of the allograph lexicon, we define the "allograph distance" as the average stroke

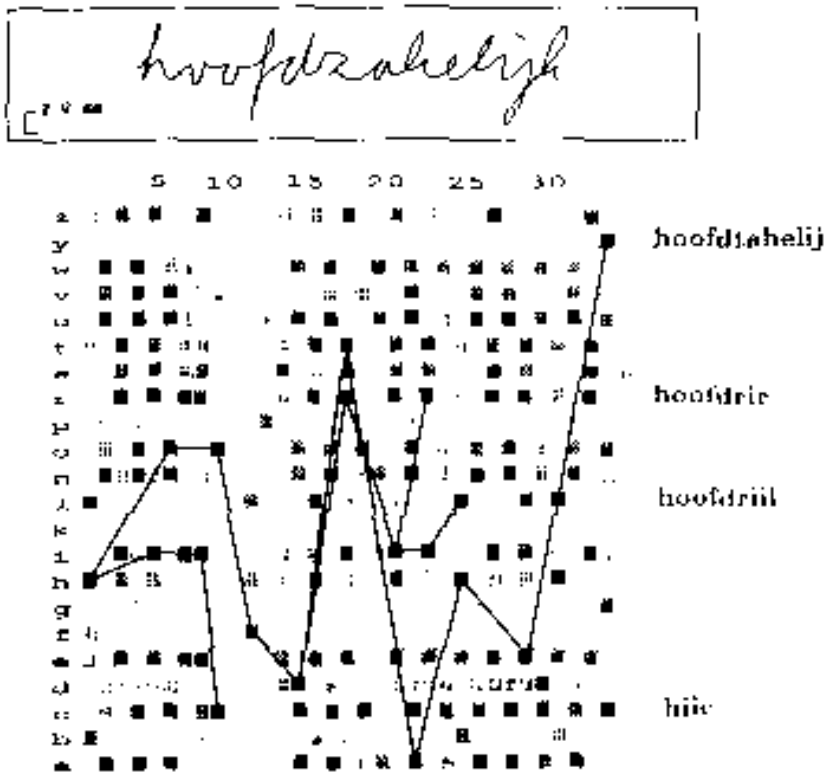

Figure 4: Figure 4. A representation of the allograph-distance table and a few examples of discontinued survivors and the most confident survivor. (Horizontal are all strokes of the Dutch word 'hoofdzakelijk' and vertical are all characters in the allograph lexicon. For the sake of simplicity, the distances of the allographs representing the same character have been collapsed. The darker the entry the lower the distance.)

distance between the corresponding strokes. In order to quantify the confidence of a sequence of hypothesized characters, we define the "word distance" as the average allograph distances. The connection strokes are not rated in the distance scores.

The algorithm estimates first the allograph distances between each allograph in the lexicon and each part of the to-be-recognized handwritten word. By storing only the allographs having relatively small distances in a table, reduces the number of possible allograph sequences dramatically (see Figure 4). Now, the modified Viterbi algorithm is started. During the first iteration, each allograph starting at the first or at the second stroke of the input word is selected. Namely, the first stroke of the input word may be either an irrelevant connection stroke or the allograph's initial connection stroke and is not necessarily included in the allographic lexicon, because they vary with context.

The allographs selected constitute the "survivors" of length one. In subsequent iterations, each survivor is extended by the best fitting allograph which fits either immediately after the survivor or with one intermediate, connection stroke. If an expanded survivor covers the whole handwritten word, it is added to a list of word hypotheses. The iterative process continues until all survivors are added to this list.

A problem in the present design is that the Viterbi algorithm is not optimal for cursive handwriting patterns where allographs have varying numbers of strokes whereas the initial strokes may be highly similar. The consequence is that the system cannot decide between

two survivors of similar quality without incorporating the between-word context. Too many solutions are competing at any serial position.

5. Learning Phase

Before a cursive-script recognition system is able to work, it has to learn the name of specific stroke sequences If the written text is available, the learning module could just assign each allograph to a character in the text. However, in cursive handwriting it appears to be a rather cumbersome task to teach a system each allograph, that may occur in a person's handwriting. The reason is that the allograph boundaries in cursive script have to be specified somehow. The simplest solution would be to set the number of strokes of the characters manually. For instance, characters 'a' and 'b' count three strokes and character 'c' counts one stroke apart from optional connection strokes, etc. However, this may not work in natural, sloppy handwriting because the number of strokes may vary. Another method is to segment cursive script into allographs using a-priori assumptions about the shape of the connection strokes between allographs (e.g., Maier, 1986). However, such a method produces persistent errors (e.g., segmenting allographs like cursive b, v, w, u, or y into two parts). A third and promising method has been proposed by Morasso et al. (1990). First a few representative prototype allographs are identified manually. If a stroke sequence is presented together with the character sequence it is representing, the sequence of strokes can be split into sub-sequences where each sub-sequence matches the prototype allograph with minimum distance. In this way, more prototype allographs can be identified. Although the set of prototypes may form the allograph lexicon, it is wise to condense the set of prototypes. For that purpose they built "allographotopic maps" in an array of Kohonen networks for sequences of two, three, ... strokes, respectively.

In the present paper we describe an algorithmic approach which may achieve the segmentation of cursive script into allographs completely automatically, but without using prototypes, and also without specific knowledge about the structure of the allographs used, nor their numbers of strokes, nor assumptions about the properties of connection strokes. The method assumes that handwriting recordings have been collected from a specific text, containing words with all kinds of character sequences. The corpus of handwriting is segmented into words and these words, in turn, into strokes. Subsequently, the feature vector of each stroke is determined. The method is based The first step is to select the largest group of words starting with the same character. Those words starting also with the same allograph will show small mutual stroke distances for the first few strokes. In order to find out which words show relatively small stroke distances a nearest neighbor clustering method is employed. However, an allograph may or may not start with an arbitrary connection stroke. Therefore, the stroke sequences have to be aligned first. This can be done by clustering the stroke distances of the first and the second strokes and selecting the strokes in the largest cluster.

After aligning the selected words all first strokes will be very similar. The group of similarity of the first strokes implies that the cluster analysis on the first stroke's feature vector forms at least one large cluster. The next step is to estimate the average inter-stroke distance of the second stroke, etc. As an example, consider the selected words starting with the character 'b', which has three strokes, apart from one optional connection stroke at the beginning or one at the end. Because all kinds of characters may follow the first character in the selected group of words, the cluster size will decrease markedly beyond the third stroke (See Figure 5). Using an appropriate threshold, the number of strokes that actually belong to character 'b' can be estimated. Finally, the average stroke pattern belonging to this character will be appended to the allograph lexicon.

In order to identify the second and following allographs in the corpus of handwriting, the initial character is removed from the words that were part of the resulting cluster and

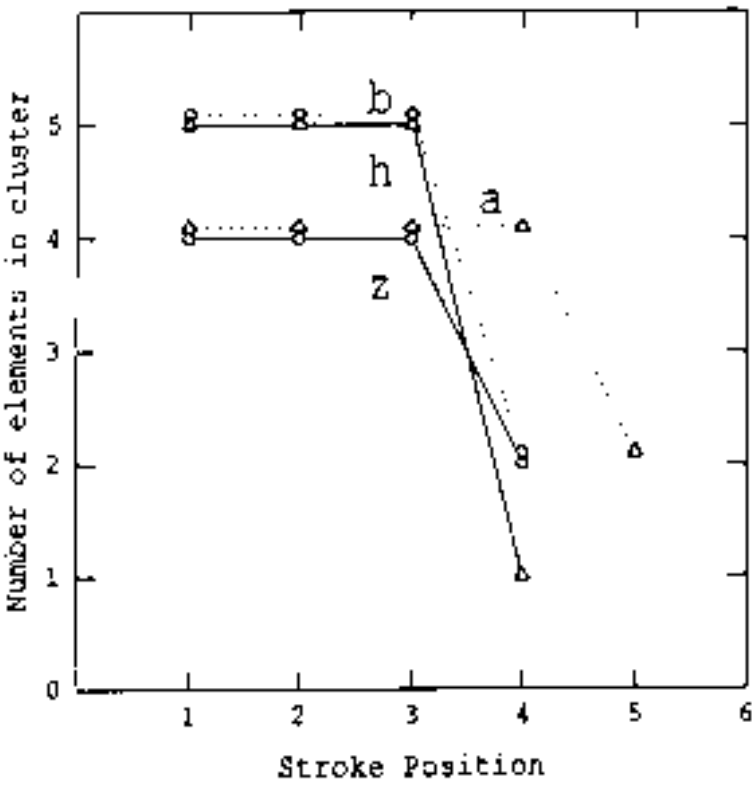

Figure 5: Figure 5. The cluster size of the feature vectors of the first six strokes from four subgroups of aligned words (The groups of words are words starting with an allograph 'b' (N=5), 'a' (N=4), 'h' (N=5), and 'z' (N=7). The cluster size in the words starting with 'a' decreases significantly after the fourth stroke, indicating that this allograph counts four strokes. Similarly, 'b', 'h', and 'z' count three strokes.)

the corresponding feature vectors are removed from the sequence of feature vectors of the words in the corpus. Then the previous step is repeated. Processing all data in this way may yield a condensed list of prototypes, which forms the lexicon of allographs. In fact it may be possible to identify even various allographic versions of the same character. As the condensed prototypes found on the basis of the writings of different writers do not need to be inconsistent their lists of allographs may be merged. The number of times that a certain allograph is used in the recognition phase indicates how universal each prototype allograph is. These numbers enable the system to discriminate between useful and redundant allographs.

Specific problems in the present design are still the necessity of tuning parameters for deciding when the cluster size drops significantly. Another problem is introduced by the artificial left-to-right scanning of the words, making the algorithm less reliable for the final characters of the words. In both respects the approach by Morasso et al. (1990) seems more appropriate.

6. Conclusion

The present paper discussed some of the modules designed for on-line cursive script recognition. However, the modules are still fragile and slow. But this is not the major problem we are dealing with: How to quantify the performance of each of the modules such that they can be compared with the performance of other designs. The four modules described above do their task but what does this tell us? In the present paper the modules were processing natural, purely cursive script of arbitrary sequences of only lower case allographs with omission of all diacritical signs such as dots on i and j. In fact the performance varies more with the properties of the sample of cursive script than with the appropriateness of the algorithms? This implies that the modules should be tested in a large number of subjects, while each subject should produce a considerable amount of cursive script. It would be wise to distinguish several "qualities" of handwriting ranging from extremely regular handwriting till the normal, sloppy handwriting, and several "character sets" ranging from only some lower case allographs till all kinds of allographs. Even if the modules are tested with large amounts of handwriting it may be hard to tell whether it performs well. The least subjective quality criterion of, for instance, normalization of the orientation, may be requiring the subjects to write horizontally on lined paper and to look for the closest-to-zero and the least-variance orientation estimator. The best known quality measure is probably still the overall percentage-of-characters correctly recognized, but this implies that the modules of a complete recognition system have been realized and run in a realistic time scale.

The second problem is the sensitivity of some modules' parameters. It would be ideal if a module could operate using only selection criteria like "the largest", or "the smallest", or "the closest to zero". But very often discrete parameters are required. They may be as harmless as "the maximum number of strokes per character is eight" till more suspect parameters as "select only strokes longer than a fixed stroke size". We feel that the micro-architecture design rules of a complex system of a cursive-script recognizer are important.

7. Acknowledgement

This research was supported by ESPRIT Project 419, Image and Movement Understanding.

8. References

Amin, A. *IRAC: Recognition and understanding systems.* Internal Report.

Aoki, K., & Yamaya, Y. (1986). Recognizer with learning mechanism for hand-written script english words. *Proceedings of the Eight International Conference on Pattern Recognition, 690-692.*

Brocklehurst, E.R. (1988). *The NPL electronic paper project.* NPL Report DITC 133/88.

Doster, W., & Oed, R. (1984). Textbearbeitung auf Personal-Computern mit handschriftliche Direkteingabe. *PC-Praxis.*

Higgins, C.A., & Ford, D.M. (1989). A comparison of N-grams and tree structure for letter graph reduction *Fourth IGS Conference* (p. 26), Trondheim, July.

Higgins, C.A., & Whitrow, R. (1989). The pad (pen and display): A demonstrator for the electronic paper project. *Fourth IGS Conference* (p. 45), Trondheim, July.

Higgins, C.A., & Whitrow, R. (1985). On-line cursive script recognition. In B. Shakel (Ed.), *Human-Computer Interaction - INTERACT '84* (pp. 139-143).

Kondo, S. (1989). A model of the handwriting process and stroke-structure of character-figures. In R. Plamondon, C.Y. Suen, M. Simner (Eds.), *Computer recognition and human production of handwriting* (pp. 103-118). Singapore: World Scientific.

Leedham, C.G., & Downton, A.C. (1986). On-line recognition of Pitman's handwritten shorthand: an evaluation of potential. *International Journal of Man-Machine Studies, 24,* 375-393.

Lindgren, N. (1965). Machine recognition of human language: Part III - Cursive script recognition. *IEEE Spectrum, 2,* 104-116.

Maarse, F.J., Schomaker, L.R.B, & Teulings, H.L., (1988). Automatic identification of writers. In G.C. van der Veer & G. Mulder (Eds.), *Human-Computer Interaction: Psychonomic Aspects* (pp. 353-360). New York: Springer.

Maier, M. (1986). Separating characters in scripted documents. *Eight International Conference on Pattern recognition* (ISBN: 0-8186-0742-4), 1056-1058.

Mandler E. (1989). Advance preprocessing technique for on-line recognition of handprinted symbols. In R. Plamondon, C.Y. Suen, M. Simner (Eds.), *Computer recognition and human production of handwriting* (pp. 19-36). Singapore: World Scientific.

Morasso, P. (1989). Neural models of cursive script handwriting *International Joint Conference on Neural Networks,* Washington, DC, June.

Morasso, P., Kennedy, J., Antonj, E., Di Marco, S., & Dordoni, M. (1990). Self-organisation of an allograph lexicon. *International Joint Conference on Neural Networks,* Lisbon, March.

Ouladj, H., Lorette, G., Petit, E., Lemoine, J., & Gaudaire (1989). From primitives to letters: A structural method to automatic cursive handwriting recognition. In Pietikainen, M. & Roning, J. (Eds.), *Proceedings of the Sixth Scandinavian Conference on Image analysis,* 593-598.

Pick, H.L., Jr., & Teulings, H.L. (1983). Geometric transformations of handwriting as a function of instruction and feedback. *Acta Psychologica,* 54, 327-340.

Plamondon, R., & Baron, R. (1986). On-line recognition of handprinted schematic pseudocode for automatic fortran code generation. *Proceedings of the Eight International Conference on Pattern Recognition,* 741-744.

Schomaker, L.R.B., & Teulings, H.L. (1990). A handwriting recognition system based on properties of the human motor system. *International Workshop on Frontiers in Handwriting Recognition,* Montreal, April.

Skrzypek, J., & Hoffman, J. (1989). *Visual Recognition of Script Characters; neural network architectures.* Technical report UCLA MPL TR 89-10, Computer Science Department University of California, Los Angeles

Srihari, S.N., & Božinović, R.M. (1987). A multi-level perception approach to reading cursive script. *Artificial Intelligence,* 33, 217-255.

Tappert, C.C. (1986). An adaptive system for handwriting recognition. In H.S.R. Kao, G.P. van Galen, & R. Hoosain (Eds.), *Graphonomics: Contemporary research in handwriting* (pp. 185-198). Amsterdam: North-Holland.

Tappert, C.C., Suen, C.Y., & Wakahara, T. (1988). On-line handwriting recognition: A

survey. *Proceedings of the 9th International Conference on Pattern Recognition,* 2, 1123-1132, Rome, November 1988.

Teulings, H.L., Schomaker, L.R.B., Morasso, P., & Thomassen, A.J.W.M. (1987). Handwriting-analysis system. In R. Plamondon, C.Y. Suen, J.-G. Deschenes, & G. Poulin (Eds.), *Proceedings of the Third International Symposium on Handwriting and Computer Applications* (pp. 181-183). Montreal: Ecole Polytechnique.

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.

Thomassen, A.J.W.M., Teulings, H.L., & Schomaker, L.R.B (1988a). Real-time processing of cursive writing and sketched graphics. In G.C. van der Veer & G. Mulder (Eds.), *Human-Computer Interaction: Psychonomic Aspects* (pp. 334-352). New York: Springer.

Thomassen, A.J.W.M., Teulings, H.-L., Schomaker, L.R.B., Morasso, P., & Kennedy, J. (1988b). Towards the implementation of cursive-script understanding in an online handwriting-recognition system. In Commission of the European Communities: D.G. XIII (Ed.), *ESPRIT '88: Putting the technology to use. Part 1* (pp. 628-639). Amsterdam: North-Holland.

Ward, J.R., & Blesser, B. (1985). Interactive recognition of handprinted characters for computer input. *IEEE, Computer Graphics and Applications, 5* (9), 24-37.

Welbourn, L.K. & Whitrow, R.J. (1989). A gesture based text and diagram editor. *Fourth IGS Conference* (p. 48), Trondheim, July.

Wells, C.J., Evett, L.J., & Whitrow, R.J. (1989) The use of orthographic information for script recognition. *Fourth IGS Conference* (p. 27), Trondheim, July.

Wright, Ph.T. (1988). Design and implementation of a handwriting recognizer. *Plessey New technology, 2,* 1988.