1. Introduction

While human beings are alike at birth and have common points in their culture, their personality, their emotions and their instantaneous state-of-mind make their handwritings significantly dissimilar. Because of this extreme variability, recognition of cursive script is a very difficult recognition problem. Paradoxically, cursive handwriting remains a universal means for exchange and communication. By what means can we recognize the scripts of our fellow-men ? Experimental psychology shows that for reading, we proceed by *saccades* (fixations followed by fast translations). Therefore, the reading texts should be segmented; but how can we perform this segmentation ? and at what level ? Should we segment for recognition, or recognize for segmentation ?

To answer these questions, two main classes of methods for the recognition of cursive script have been proposed : *analytical* and *global* methods. The first ones attempt to segment words into letters to recognize them, while the others attempt to recognize the words globally.

In what follows we outline the main properties of these two main classes of approaches.

1.1 Analytical Methods

There are basically two kinds of analytical methods :

-- Analytical methods with **explicit** (or "external") segmentation, where segmentation is performed explicitly *in a previous stage* taking place before recognition as in [Harmon62], [Sayre73], [Maier86], [Bozinovic89].

-- Analytical methods with **implicit** (or "internal") segmentation, where segmentation and recognition are performed *at the same time* (segmentation in letters is then a "result" of recognition as in [Eden61], [Berthod80], [Tappert82], [Ouladj89], [Higgins85]).

As they are designed to recognize letters, the main advantage of all these techniques is that most of them do not need a preliminary training phase on words. Some of these techniques are even able to recognize an unlimited vocabulary (no lexicon is provided). Other methods need an arbitrary lexicon (typically the ASCII list of possible words), but not specific learning on word data samples. But both of these two kinds of techniques have important drawbacks:

-1- segmenting word before recognizing them can be found paradoxical. What kind of criteria can be used by *explicit methods* to avoid the numerous errors that would be the result of an arbitrary segmentation process.

-2- basing segmentation on recognition only is also paradoxical: words containing illegible letter can not be identified because word segmentation will fail if some letters can not be recognized. This limits the use of *implicit methods* to well-written cursive handwriting.

1.2 Global Methods

Global methods do not attempt to distinguish the different letters of the word. They are generally based on an analysis of the global shape of the word (for instance chain codes) or on an analysis of a set of features extracted from the whole word. Such methods have been experimented by [Earnest62], [Miller71], [Farag79], [Wong85], [Lery89].

The main advantage of these methods is that they overcome the difficult problem of segmentation of words into separated characters. So, badly written words should be recognized even if some letters are omitted or badly written.

Their main drawback is that they are only able to deal with a predefined vocabulary: a training phase (or a special coding) is therefore necessary to recognize any new vocabulary or any new word added to the predefined lexicon (by opposition, some analytical methods would need a lexicon, but only in an ASCII form: no training is required). Besides, computation time can be prohibitive if the lexicon is not of a very small size because it is necessary to compare an unknown word with all the possible reference words of the lexicon.

Moreover, on a conceptual point of view, it can be said that the required training phase on words implies a kind of dependence on the writer style: there is a huge number of possible stylistic variations to write words and it has never been proved that any training phase could take into account all these variations. This makes these methods more suited for mono- or multi-scriptor recognition (a few predefined scriptors) than omni-scriptor recognition (any writers).

2. Principles of the Method: towards the Concept of Grapheme

2.1 A Few Hints

One of the most interesting/important property of cursive handwriting is that all the components of the "written signal" do not have the same level of meaning:

-- *ligatures* are only non-significant stylistic artifacts,

-- while *characters* can be considered as the informative part of the words.

Moreover, some characters can be much more significant than others for a given context as shown in **Figure 1** which contains two samples of the word "cinquante" (fifty in French). Although the second word is rather badly written, it would be easily recognized by a French reader (at least in a given context, like literal amounts of checks). This can be explained as follows: as the total number of possible words is small for that given context, only a few letters (for instance letters "C", "q", "t" and "e") need to be recognized to identify the word.

This means that an omni-scriptor identification system must be able to **focus recognition** on the *most significant parts* of the word † and eliminate non-significant data like ligatures. It must also perform some kind of **global interpretation** from all these meaningful entities in order to allow word identification even if some letters are omitted, erroneous or badly written.

This leads us to the concept of a **hybrid identification approach**, that is to say a method which is both:

-- **analytical**, as it is able to extract meaningful entities from the words (during a segmentation phase), and,

-- **global**, as the final identification process is done in an overall fashion.

Such a concept emphasizes the crucial role of a clever segmentation process. This segmentation phase will take place *before* the actual recognition of the extracted entities. Finally, the word will be *identified globally* by means of a contextual matching stage comparing the whole list of graphemes to the possible likely words of the lexicon. The method will not try to recognize accurately all the letters but only to *collect enough information* from the whole word.

The rest of this paper focuses on the segmentation problem only (other phases are described in [Lecolinet90]) and tries to solve the following two basic problems:

-1- what kind of *meaningful entities* are we going to extract ?

-2- what kind of *segmentation criteria* can we use to distinguish between non-significant stylistic dependent features and informative data.

2.2 A Definition of the Graphemes

By definition, the **graphemes** are the output of the segmentation process. Graphemes are those parts of the word which *"do not look like"* a ligature or *"look like"* a character. In practice, graphemes are high-level entities close to characters, but not actual characters, as explained further. That close relationship with characters avoid the method to be dependent on a specific vocabulary: no word learning phase is required but only an ASCII lexicon).

† see also the interesting work of Pr. Simon about regular and singular features and how these concepts are applied to handwriting recognition by extracting singular features by complementarity to identify the words [Simon90].

An important property is that because these entities are *high-level* they are more *invariant* than low-level features descriptions (like loops, curves, crossing points, etc.), and that recognition is less sensitive to the noise brought by ligatures. This is illustrated in **Figure 2** which shows the instability and the inaccuracy of an uniform representation based on low-level primitives (loops, oriented concavities, ascenders and descenders in this case):

-- in the first case (**Fig. 2a**), two different words (neuf = nine & sept = seven in french) have very similar representations: as the ligatures remain undetected the information that would enable to differentiate the words is lost

-- in the second case (**Fig. 2b**) the same word (trente = thirty) written in different styles (cursive in one case, script in the other) results in very different representations because the ligatures between letters cause a *"background noise"* which becomes more important than the *"written message"*.

On the other hand, segmentation and recognition of high-level entities decrease the influence of ligatures because ligatures can be considered as low-level perturbations of the shape of the graphemes (they become part of higher level entities which can generally still be recognized). Moreover, segmentation points constitute a kind of *"pauses"* in the writing which make its recognition easier by decreasing the number of possible combinations (the "complexity" of recognition).

2.3 PreRecognition and Grapheme Extraction

Because graphemes are supposed not to *"look like"* ligatures but to *"look like"* characters, **grapheme extraction** (or "segmentation") is based on two competitive processes:

-- *ligature detection*, and,

-- *character detection* (or **preRecognition**).

The function of this second module is to *detect* the parts of the writing which are *likely* to be characters in order to prevent detection of features that could be considered as ligatures by the segmentation algorithm but are *actual parts* of some character. It is interesting to notice that it *is not* a character recognition algorithm, in the sense that it is not able to *decide* which character is written, but only to *detect* if a given part of writing is likely to be a character. As it performs some kind of basic recognition although it is not actual character recognition, it has been called **"preRecognition"**†. In one way, such a scheme (that is to say *recognizing without actually recognizing*) is a way to solve the usual paradox according to which one should both have to segment to recognize and to recognize to segment !

2.4 A Typology of Ligatures

Experience shows that most of the ligatures between cursive letters are the local minima (or "valleys") of the upper outline of the word (as shown in **Figure 3**). However, unfortunately, there are several exceptions to this rule:

-1- all the minima (upper valleys) are not actual ligatures:

a/ some valleys can be *artifacts* due to bad writing or binarisation, for instance when loops are not actually closed (**Fig. 4a**),

b/ some valleys are *intra-letter* links. This occurs mainly with letters "u", "v", "w" ... where a part of the shape looks exactly like a ligature, but is an actual part of the letter (**Fig. 4b**),

-2- some ligatures are not upper valleys:

c/ some letters are *"stuck together"* and/or very difficult to distinguish (**Fig. 4c**)

d/ some ligatures correspond to a *ridge of the lower outline*. This case occurs mainly after the letters "o", "v" and "w" (for instance "oi" etc..) (**Fig. 4d**).

† A similar concept has yet been used in a previous study we carried out for the segmentation of handwritten zip codes (see ref. [Lecolinet89]).

These cases are dealt with in the following way:

-- *PreRecognition* will prevent valleys detected in **case -a-** to be considered as actual ligatures because they would take place inside a detected character,

-- **cases -b- and -c-** will produce respectively *over-* and *mis-segmentation* because there is no way to prevent this without the help of context: such is the structure of the character than it can be confused with two characters linked together (for instance a "u" looks like two "i" without a point and linked together). As it is a contextual problem which generally requires the whole word to be identified, it is solved at the final contextual matching stage between the list of extracted graphemes and the reference ascii words of the lexicon. During this stage, graphemes can be combined by means of *contextual re-association rules*. (For instance, one of these rules will allow to match two consecutive graphemes recognized as "small vertical stroke" with a single letter like "u" or "n").

-- *case -d-* requires a special complementary procedure to detect and segment properly this kind of ligatures. The principle is the same as for the detection and the segmentation of the valleys of the upper profile.

3 Segmentation Techniques

3.1 The detection of Ligatures

As mentioned before, the detection of ligatures is mainly based on the detection of the *valleys of the upper outline*. But dealing with outlines presents several drawbacks because they cannot be described by a monotonic function on the x-axis (the "temporal" axis). For this reason, we prefer to deal with *profiles* and consider two different types of valleys:

-1- the *"projective" valleys* of the upper profile, which can be reached directly by vertical projection,

-2- the *lateral concavities* of the upper profile, that cannot be reached directly because of the presence of an horizontal stroke or because the writing is slanted (see example on **Figure 5**).

a). Projective valleys of the upper profile are:
-- local minima,
-- surrounded by 2 local maxima (one on each side),
-- with a sufficient relative depth between this minimum and the two surrounding maxima.

This algorithm (which is explained more precisely in [Lecolinet90]),
-- is in only one pass,
-- does not take into account deformations along the x-axis (the only criterion is relative depth),
-- remains robust in presence of noise and always returns the deepest valley when several possibilities could be chosen.

b). Lateral concavities are detected when there is an empty space between two successive points of the upper profile like in example on **Figure 5** (between the "d" and the "e"). These concavities are systematically examined by means of a *mobile cursor* algorithm (an automaton) which tries to find the lowest point of the concavity, while avoiding non-significant local minima. An example of what would be the path of that mobile cursor is also shown in **Figure 5**. The final position of the cursor will be the "anchorage point" (that is to say a hypothetical starting point) of the actual segmentation process. More details about this technique can be found in [Lecolinet90]. A similar algorithm is also explained in [Lecolinet89].

3.2 The PreRecognition Process

As explained before, preRecognition is used to avoid segmenting certain parts of the characters which could be erroneously identified as ligatures. The preRecognition process is based on concavity analysis and some complementary procedures outlined in the next paragraphs.

a). Concavity analysis is performed to detect loops which are not actually closed (because the word has been written hastily or with bad ink ...). This procedure both permits to extract some relevant information (the presence of loops) and avoid some erroneous segmentations.

A new specific algorithm has been developed to solve this difficult problem (such an algorithm must avoid to detect concavities which are not *real* ligatures, but, for obvious efficiency reasons, it must also be able to detect most of the "open loops"). This algorithm works in a complementary way:

-- first, the *"outside"* of the characters is estimated (that is to say the background area that surrounds them),

-- then, the *"concavities"* (that is to say the possible loop areas) are obtained by subtraction (each of them is a set of connected pixels which belong to the background but not to the "outside"),

-- last, these concavities are *classified* by considering their area and their outline:

  -- if its area is too small, the concavity is only an insignificant *artifact*,

  -- if Nc, the number of point of the outline which are connected to a pixel of the "outside" is null, the concavity is a *closed loop*,

  -- otherwise, Nc is compared to Nt (the total number of points of the outline) to determine if the concavity is an *"open loop"* or a mere *"concavity"*.

More precisely, this comparison is improved by taking into account the *direction* of the outline points. This is shown in **Figure 6**, where the ratio Nc/Nt is almost the same for the concavities of the two characters, but quite different if the direction (for instance the horizontal and vertical directions) is taken into account. This last criterion takes better account of the topological properties of concavities and we think it corresponds to the intuitive notion of "loop".

b). Extension analysis permits to invalidate certain segmentation hypothesis because of their y-location (ligatures are supposed to be close to the *"central part"* of the cursive writing. The central part of the word (i.e. the horizontal part between upper and lower extensions) is determined by calculating an upper and a lower baseline by means of a histogram technique. Then ascenders and descenders are extracted, and as they are supposed to be part of some characters, segmentation is avoided on such features.

c). Shape and Size Analysis procedure (which has yet been used for the detection of numerical characters in [Lecolinet89]) takes into account the numbers of loops, the relative size, etc ... to estimate if the component analyzed is likely to be a "real" character or a part of character or a combination of merged characters. In this case is it used to refine the results of the previous stages and to perform useful regrouping when some characters are broken into several parts.

3.3 One-Pass and Hierarchical Segmentation

The **one-pass segmentation process** considers all the upper valleys or the anchorage points of the lateral concavities that have been detected and tries to find a separation path going from this point to the lower bottom of the image. It is important to notice that valleys and anchorage points are only segmentation hypothesis : actual segmentation will take place only if the separation path verify several coherence criteria. (A few segmentation examples are shown in **Figure 7**).

In a few specific cases, the algorithm described above cannot be efficient because of the presence of a large horizontal stroke (as the crossbar of a large "t"). In such a case, as the profile cannot be analyzed properly, a specific algorithm is used to perform the **hierarchical segmentation** of the subpart of the word which cannot be dealt with normaly. An example of that is shown in **Figure 8** (all the letters beneath the "t" are properly segmented).

4. Experiments

This algorithm has been used for two application projects carried out by the Matra-MS2i company for the French Postal Services:

* the automatic reading of city names on mail,
* the recognition of literal amounts of checks.

In both cases, the word identification process consists in three stages:

-1- grapheme extraction (segmentation of the word),

-2- grapheme recognition,

-3- contextual analysis (matching of the whole list of graphemes to the possible likely words of the lexicon).

In both studies, a linear discrimination technique was used for grapheme recognition and a dynamic programming method for contextual matching (the complete system for city name recognition is outlined in [Lecolinet90b]. The linear discrimination technique has previously been used for zip code recognition (see [Lecolinet89]). The contextual matching phase attempts to match the list of recognized graphemes with the most likely words of the lexicon (a first selection is done by estimating the actual length of the written word). As the lexicon is in an ASCII form, no special training is needed. This allows to build the lexicon dynamically, as for the city name application where the lexicon is the list of city names which correspond to the previously recognized zip code. In this case, the total vocabulary is about 36000 words, but the lexicon is only a small subpart of it (less than 30 words), which depends on the zip code.

The second study (check recognition) includes more sophisticated features as re-association contextual rules for global matching (to deal with over- and mis-segmentation problems) that permit a more efficient use of contextual information, as explained in [Lecolinet90]. The vocabulary is fixed and contains about 30 words (the "root" numbers and a few special words like the currency name). The size of the ASCII lexicon is larger (about 70 words) because several variations are allowed (words beginning by a lower-case or an upper-case letter, usual abbreviations).

The evaluation was carried out by using the detailed analysis of about 200 words on checks taken at random from an important test base. In order to obtain sufficiently general results that can be compared to other existing methods, the data have been divided into four homogeneous categories (see a few typical examples in **Figure 9**):

-a- good quality: these words more or less respect the rules of school writing,

-b- correct quality:
some letters are distorted, stuck together or missing,

-c- poor quality: informative letters are missing or very badly written or several consecutives letters are stuck together or illegible,

-d- bad quality:
these words are difficult to read without the help of context.

For each of these four categories, the table shown in **Figure 10** gives the percentage of properly identified words belonging to one of the N first positions, for N = 1,2,3,4,5,10. It also indicates the proportion of words which were rejected by the algorithm and the distribution of the words into the four categories.

Subjective as it may be, this classification has the advantage of emphasizing to what extent any evaluation is dependent on the quality of the test bases. It should be remembered that in our case the data are really omni-scriptor and unconstrained. On the other hand, the vocabulary is of reduced size. It appears that:

-- The method gives good results when the writing is "correct" (the good solution comes first in 3 cases out of 4 and among the first 3 choices in nearly 9 cases out of 10).

-- The results remain encouraging with poor quality writing if we keep the first five choices (the percentage of good recognition is then 70%). This result is interesting because for many applications (such as the reading of checks) the syntax of the whole sentence makes it possible to eliminate wrong solutions.

-- Lastly, this methods makes it also possible to assess the reliability of recognition by analyzing the distribution of the matching scores associated with each possible choice: the better the writing (and the recognition), the bigger the difference between the scores and vice-versa. This property is interesting for further contextual processing at the sentence level.

5. Conclusion

We have developed and tested an hybrid method for the recognition of cursive words which has been designed for:

-- off-line and omni-scriptor recognition,

-- with poor quality handwriting,

-- and an undifferentiated vocabulary but of a limited size (given as an ASCII list of words).

In this paper we tried to show that interesting results can be reached in recognition of poor quality cursive handwriting by using an hybrid method based on the extraction and identification of high-level menaningful entities called *graphemes*. The paper focuses on the most critical phase of this approach which is the segmentation phase of the image words into graphemes. For that purpose, a new segmentation technique which is based on competitive processes and introduces the concept of *preRecognition* is proposed. PreRecognition is a kind of primitive recognition which is only used to *"detect"* the presence of characters without actually recognizing them. It works in competition with ligature detection procedures in order to avoid erroneous segmentation. So, the accuracy of the segmentation process is increased by means of contradictory procedures which achieve a kind of *"balance"*.

The results obtained even on data of poor quality, are quite encouraging: this shows that omni-scriptor cursive script recognition is now a realistic goal and that efficient systems can be designed for the recognition of ordinary data. Further research will be necessary to deal with larger vocabularies and, more particularly, to improve the global matching stage. Because of the general character of the method, such an extension seems natural.

6. Acknowledgments

We would like to thank Mr. J.V. Moreau and Ph. Antier and all the team of Matra MS2i and of the French Postal Services.

7. References

[BER80] M. Berthod & S. Ahyan, *On Line Cursive Script Recognition : A Structural Approach with Learning*, 5th Int. Conf. on Pattern Recognition, pp723-725, 1980.

[BER82] M. Berthod, *On-line Analysis of Cursive Writing*, in Computer Analysis and perception. Vol. I. Visual Signals. Ch. IV, pp.55-80, 1982 CRC Press Inc.

[BOZ89] R.M. Bozinovic & S.N. Srihari, *Off-Line Cursive Script Recognition*, IEEE Trans. on Pattern Analysis and Machine Intel., Vol PAMI-11, No1, pp68-83, 1989.

[EAR62] L.D. Earnest, *Machine Recognition of Cursive Writing*, C.Cherry editor, Information Processing, 462-466, 1962.

[EDE61] M. Eden, *Handwriting and Pattern Recognition*, IRE Trans. on Inf. Theory, Feb 1961.

[FAR79] R.F.H. Farag, *Word-Level Recognition of Cursive Script*, IEEE Trans. on Computers, Vol C-28, No 2, 172-175, Feb. 1979.

[HAR62] L.D. Harmon, *Handwriting Reader Recognizes Whole Words*, Electronics 29-31, Aug 1962.

[HAR72] L.D. Harmon, *Automatic Recognition of Print and Script*, Proc. of the IEEE, Vol60, No10, pp1165-1177, Oct 72.

[HIG85] C.A. Higgins & R. Whitrow, *On-Line Cursive Script Recognition*, Human-Computer Intreraction - INTERACT'84, 1985.

[LEC89] E. Lecolinet & J-V. Moreau, *A New Sytem for Automatic Segmentation & Recognition of Unconstrained Handwritten Zip Codes*, The 6th Scandinavian Conference on Image Analysis, Finland, June 1989, Oulu.

[LEC90] E. Lecolinet, *Segmentation d'images de mots manuscrits: Application a la lecture de chaines de caracteres majuscules alphanumeriques et a la lecture de l'ecriture cursive*, Doctorat de l'universite Paris 6, March 1990.

[LEC90b] E. Lecolinet & J-V. Moreau, *Off-Line Recognition of Handwritten Cursive Script for the Automatic Reading of City Names on Real Mail*, 10th Int. Conf. on Pattern Recognition, Atlantic City, June 1990.

[LER89] J-M. Lery, *Reconnaissance de l'écriture manuscrite mono-scripteur : description symbolique des mots par un graphe de primitives*, Doctorat de l'universite Paris 6, June 1989.

[LIN65] N. Lindgren, *Machine recognition of human language; Part III-Cursive script recognition*, IEEE Spectrum, pp104-116, May 1965.

[MAI86] M. Maier, *Separating Characters in Scripted Documents*, 8th Int. Conf. on Pattern Recognition, Paris, pp1056-58, 1986.

[OUL89] H. Ouladj, G. Lorette, E. Petit, J. Lemoine & M. Gaudaire, *From Primitives to Letters A Structural Method to Automatic Cursive Handwritting Recognition*, The 6th Scandinavian Conference on Image Analysis, Finland, pp593-599, June 1989, Oulu.

[SAY73] K.M. Sayre, *Machine Recognition of Handwrittten Words : A Project Report*, Pattern Recognition 5, 213-228, 1973.

[SIM90] J.C Simon & O. Baret, *Handwriting Recognition as an Application of Regularities and Singularities in Line Images*, in Frontiers in Handwriting Recognition by C.Y. Suen, 1990.

[TAP82] C.C. Tappert, *Cursive Script Recognition by Elastic Matching*, IBM J. Res. Develop. 26, 765-771, Nov. 1982.

[TAP88] C.C. Tappert, C.Y. Suen & T. Wakahara, *On-line Handwriting Recognition - A Survey*, 9th Int. Conf. on Pattern Recognition, Roma, pp1123-1127, Nov 1988.

[WON85] K.H. Wong & F. Fallside, *Dynamic Programming in the recognition of connected handwritten script*, 2nd Conf. on Artificial Intelligence Applications, 666-670, Dec. 1985.