Word Shape Analysis for a Hybrid Recognition System

R.K. Powalka, N. Sherkat, R.J. Whitrow

Department of Computing, The Nottingham Trent University

Abstract - *This paper describes two wholistic recognizers developed for use in a hybrid recognition system. The recognizers use information about the word shape. This information is strongly related to word zoning. One of the recognizers is explicitly limited by the accuracy of the zoning information extraction. The other recognizer is designed so as to avoid this limitation. The recognizers use very simple sets of features and fuzzy set based pattern matching techniques. This aims to increase their robustness, but also causes problems with disambiguation of the results. A verification mechanism, using letter alternatives as compound features, is introduced. Letter alternatives are obtained from a segmentation based recognizer coexisting in the hybrid system. Despite some remaining disambiguation problems, wholistic recognizers are found capable of outperforming the segmentation based recognizer. When working together in a hybrid system, the results are significantly higher than that of the individual recognizers. Recognition results are reported and compared.*

1 Introduction

There appears to be no single, uniform approach to the problem of handwriting recognition. Various algorithms achieve considerable success with certain handwriting styles, but cannot maintain their high recognition rates for other styles. An on-line cursive script recognition system developed by the authors [1, 2] is no exception. It is capable of recognition rates exceeding 90%, as well as rates lower than 30%, depending on the writer.

It is found that various recognition methods produce different results and errors. Their decisions are frequently complementary. Combining results of multiple recognizers provides almost universal improvement [2, 3, 4, 5]. This is even the case when identical vectors of features are used [3].

The recognition system developed by the authors uses multiple interactive segmentation [1], a segment and recognize approach. The method works well in cases where correct segmentation is possible, however it inherently fails for sloppy[1] writing. Word ending postulation [2] has been introduced to cope with words becoming sloppy towards their endings. A wholistic recognition approach can be adopted as an alternative solution. Information about the word shape considerably limits the number of word alternatives which may represent a handwritten word. It is often possible to identify the word correctly without having to locate and recognize all the letters composing it (e.g. [6, 7, 8]). This can be an enormous help in recognition of cursive handwriting, where sloppy or difficult to segment words are rather frequent. Where correct segmentation is possible, the multiple interactive segmentation recognizer provides an answer.

1. Handwriting is considered sloppy if it is impossible to locate, segment out and recognize all the letters composing words.

This answer can be further verified by a wholistic recognizer. Where segmentation fails, the wholistic recognizer is relied upon.

This paper discusses two new on-line wholistic recognizers designed for use in a hybrid recognition system, beside the multiple interactive segmentation recognizer. The first recognizer uses the sequence and position of ascenders and descenders detected within the word. It is referred to as the *ascender/descender recognizer*. The second recognizer uses downwards directed vertical parts of strokes. It is referred to as the *vertical bars recognizer*. Both ascenders/descenders and vertical bars are features which determine the word shape. Ascenders and descenders need to be correctly classified. This is achieved using word zoning information, extraction of which can be difficult and prone to errors. Vertical bars do not require any further classification, hence their detection is less prone to errors.

Furthermore, *letter verification* is introduced to improve the disambiguation among the resulting word alternatives. Letter alternatives located within the word are used by the letter verification as compound features. Letter alternatives are provided by the multiple interactive segmentation [1] allowing reuse of partial results.

Figure 1 presents the relationship between the algorithms presented in this paper. The abbreviations introduced in the figure are further used in Section 8, when discussing the results.

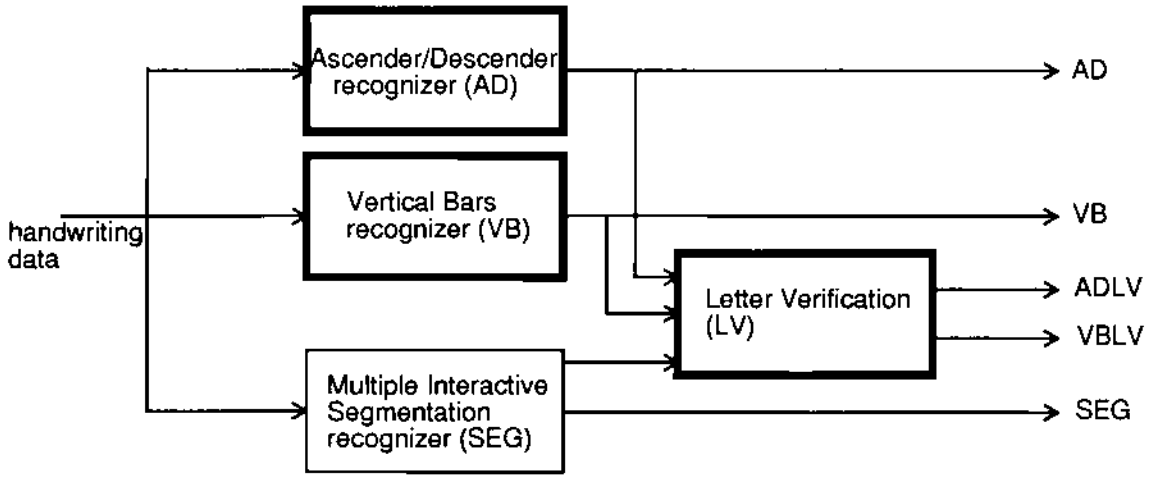


Figure 1. Relationship between the recognizers used by the authors. Algorithms discussed in this paper are marked with thick line.

2 Word shape extraction

Throughout this paper the term *word shape* is used to mean the information on presence, sequence and relative position of ascenders and descenders within a word, and the approximate word length. In case of the vertical bars recognizer the word shape will also include the information extracted from the middle zone of the word. The ascenders and descenders are detected using the word zoning information. The word length is estimated based on the number of times an imaginary horizontal line intersects the trace of the pen in its most dense area [9]. *Information in the middle zone of the word is available in the form of vertical bars.*

Typically two approaches to zoning information extraction are applied [10]:

- histogram method;
- extrema method.

The histogram method is more efficient in detecting the presence of various zones, but has difficulties with reliably locating *zone boundaries*[1]. The extrema method has more difficulties with detecting distinct zones, but is better at locating precise zone boundaries.

A combination of the two approaches has been used in order to exploit the advantages of both. The histogram method is used to determine the type of a word (i.e. whether it is composed of middle zone letters[2] only, or it contains some letters with ascenders[3] or/and descenders[4]). This is referred to as the *zoning class*. Four zoning classes can be distinguished: middle zone only, middle/upper, middle/lower and upper/middle/lower zones. Once the zoning class is determined, the extrema method is used to decide the zone boundaries.

2.1 Word zoning classification

A horizontal word density histogram is built and analysed for each word. The histogram is created by counting the number of intersections of the pen path with imaginary horizontal lines drawn through the word at different heights. Horizontal parts of the pen path count as one intersection. This alleviates problems with peaks caused by 't' crosses. A zone of high density is then identified within the histogram. A threshold derived from the average value of the histogram is used. The extracted zone is expected to contain most of the middle zone letters.

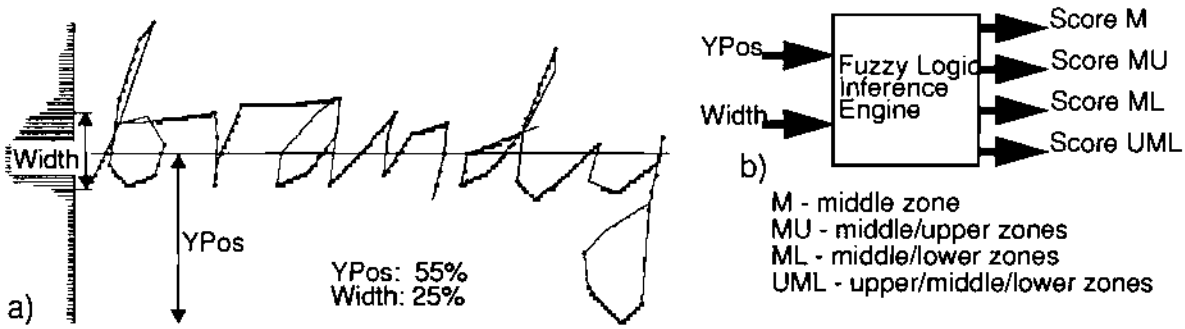


Figure 2. Word zoning classification:
a) choice of decision parameters; b) diagram of the fuzzy logic inference engine.

In order to describe the extracted zone two parameters are required. Zone width (*Width*) and vertical position within the word (*YPos*) have been chosen, as presented in Figure 2a. Alternatively, position of the upper and lower boundary of the identified zone can be used. However, the choice of *Width* and *YPos* is more intuitive and allows to easily set expectations for particular types of words (see Figure 3a). The analysis of presence of various zones within

1. Zone boundaries are understood as horizontal lines delimiting tops and bottoms of middle zone letters from all the ascenders and descenders, respectively. An example of the zone boundaries can be seen in Figure 6a.
2. Letters: a, c, e, i, m, n, o, r, s, u, v, w, x, z.
3. Letters: b, d, f, h, k, l, t.
4. Letters: f, g, j, q, p, y.

This answer can be further verified by a wholistic recognizer. Where segmentation fails, the wholistic recognizer is relied upon.

This paper discusses two new on-line wholistic recognizers designed for use in a hybrid recognition system, beside the multiple interactive segmentation recognizer. The first recognizer uses the sequence and position of ascenders and descenders detected within the word. It is referred to as the *ascender/descender recognizer*. The second recognizer uses downwards directed vertical parts of strokes. It is referred to as the *vertical bars recognizer*. Both ascenders/descenders and vertical bars are features which determine the word shape. Ascenders and descenders need to be correctly classified. This is achieved using word zoning information, extraction of which can be difficult and prone to errors. Vertical bars do not require any further classification, hence their detection is less prone to errors.

Furthermore, *letter verification* is introduced to improve the disambiguation among the resulting word alternatives. Letter alternatives located within the word are used by the letter verification as compound features. Letter alternatives are provided by the multiple interactive segmentation [1] allowing reuse of partial results.

Figure 1 presents the relationship between the algorithms presented in this paper. The abbreviations introduced in the figure are further used in Section 8, when discussing the results.

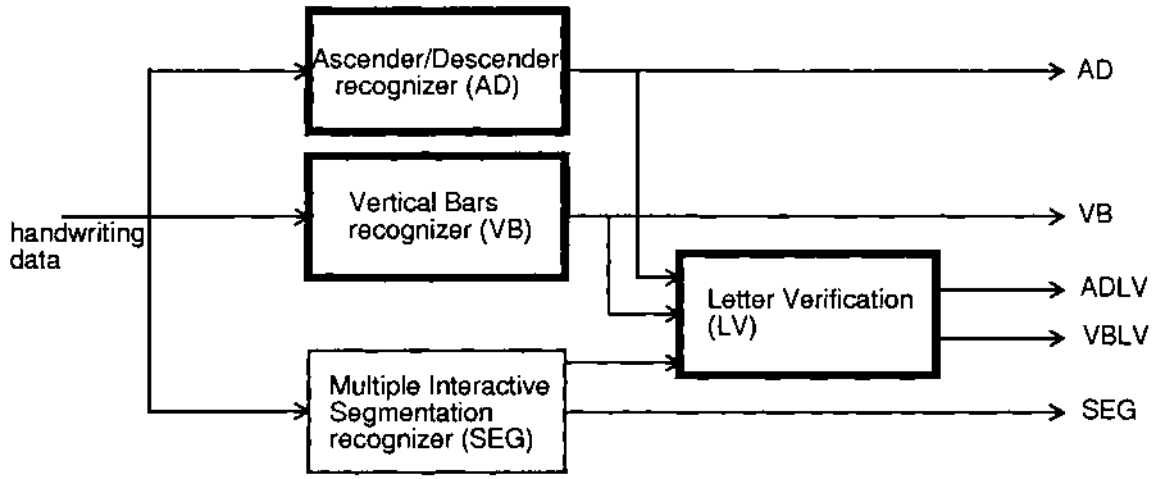


Figure 1. Relationship between the recognizers used by the authors. Algorithms discussed in this paper are marked with thick line.

2 Word shape extraction

Throughout this paper the term *word shape* is used to mean the information on presence, sequence and relative position of ascenders and descenders within a word, and the approximate word length. In case of the vertical bars recognizer the word shape will also include the information extracted from the middle zone of the word. The ascenders and descenders are detected using the word zoning information. The word length is estimated based on the number of times an imaginary horizontal line intersects the trace of the pen in its most dense area [9]. Information in the middle zone of the word is available in the form of vertical bars.

Typically two approaches to zoning information extraction are applied [10]:

- histogram method;
- extrema method.

The histogram method is more efficient in detecting the presence of various zones, but has difficulties with reliably locating *zone boundaries*[1]. The extrema method has more difficulties with detecting distinct zones, but is better at locating precise zone boundaries.

A combination of the two approaches has been used in order to exploit the advantages of both. The histogram method is used to determine the type of a word (i.e. whether it is composed of middle zone letters[2] only, or it contains some letters with ascenders[3] or/and descenders[4]). This is referred to as the *zoning class*. Four zoning classes can be distinguished: middle zone only, middle/upper, middle/lower and upper/middle/lower zones. Once the zoning class is determined, the extrema method is used to decide the zone boundaries.

2.1 Word zoning classification

A horizontal word density histogram is built and analysed for each word. The histogram is created by counting the number of intersections of the pen path with imaginary horizontal lines drawn through the word at different heights. Horizontal parts of the pen path count as one intersection. This alleviates problems with peaks caused by 't' crosses. A zone of high density is then identified within the histogram. A threshold derived from the average value of the histogram is used. The extracted zone is expected to contain most of the middle zone letters.

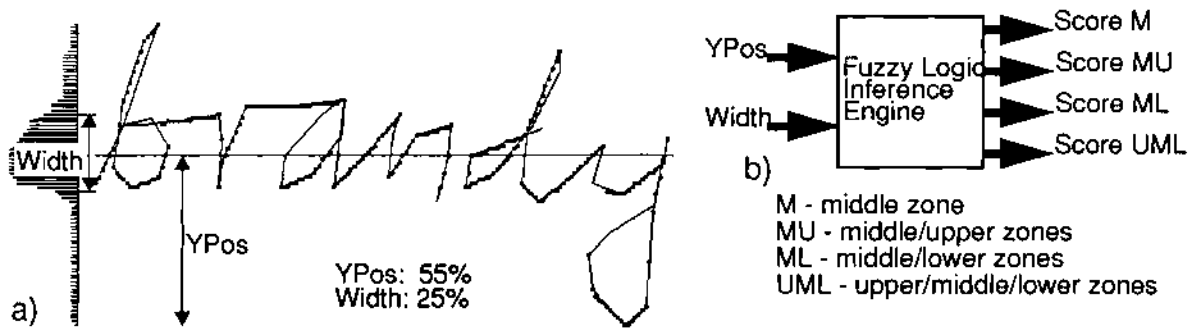


Figure 2. Word zoning classification:
a) choice of decision parameters; b) diagram of the fuzzy logic inference engine.

In order to describe the extracted zone two parameters are required. Zone width (*Width*) and vertical position within the word (*YPos*) have been chosen, as presented in Figure 2a. Alternatively, position of the upper and lower boundary of the identified zone can be used. However, the choice of *Width* and *YPos* is more intuitive and allows to easily set expectations for particular types of words (see Figure 3a). The analysis of presence of various zones within

1. Zone boundaries are understood as horizontal lines delimiting tops and bottoms of middle zone letters from all the ascenders and descenders, respectively. An example of the zone boundaries can be seen in Figure 6a.
2. Letters: a, c, e, i, m, n, o, r, s, u, v, w, x, z.
3. Letters: b, d, f, h, k, l, t.
4. Letters: f, g, j, q, p, y.

a word is referred to as *word zoning classification.* Fuzzy logic is applied to this task (Figure 2b).

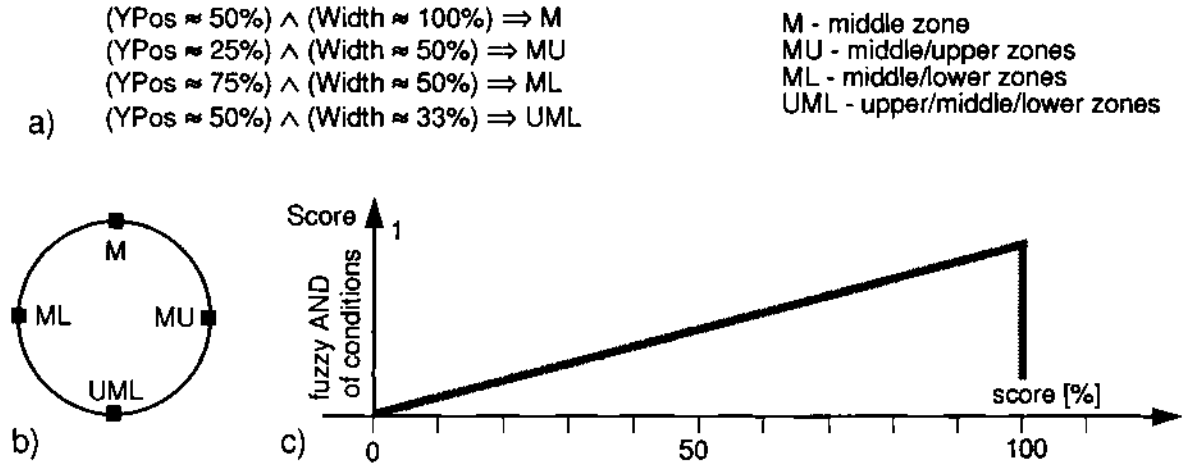


Figure 3. Fuzzy word zoning classification:
a) initial rules; b) neighbouring zoning classes; c) defuzzification function.

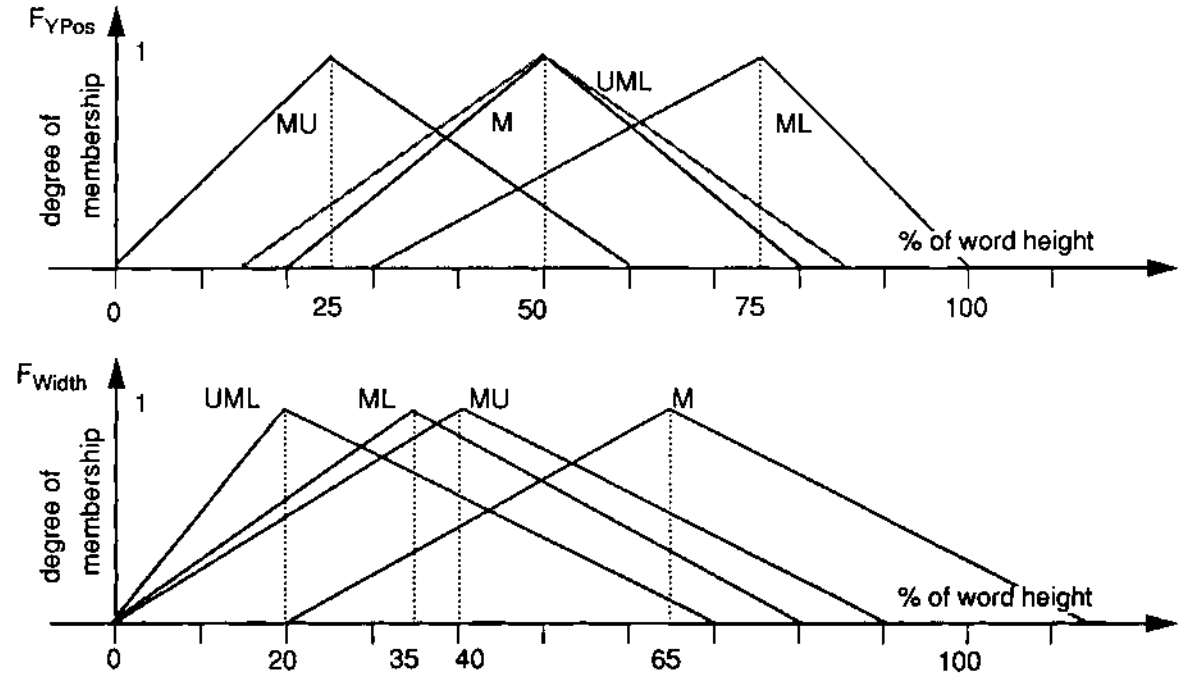


Figure 4. Experimental fuzzy word zoning classification rules for parameters: YPos and Width. Figure represents true parameters of the fuzzy sets.

Figure 3a presents the initial conditions for the word zoning classification. The initial conditions are based on the knowledge about "ideal" writing (i.e. writing conforming to the handwriting rules). The numerical values of the initial conditions are experimentally refined to result in the

set of conditions presented in Figure 4. This is normal practice when applying fuzzy logic. The refined conditions for parameter *Width* differ significantly from the initial values (Figure 4 versus Figure 3a). This is due to the fact that letters within the word often do not line up well. This results in a less crisp histogram than for a perfectly written word. Consequently the zone of highest density within the word is narrower than expected in the ideal writing.

The fuzzy inference engine produces individual scores for each possible word zoning classification (M, MU, ML, UML - as in Figure 4):

$$Score = min(F_{YPos}(YPos), F_{Width}(Width)) \cdot 100\% \quad (1)$$

Function *min*() implements fuzzy operator AND [11], functions $F_{YPos}()$ and F_{Width} are presented in Figure 4.

A very simple defuzzification function is used (Figure 3c). The highest score indicates the most likely zoning classification. The difference between the highest and second highest score indicates the ambiguity of classification. No attempt is made to produce a single answer. Effectively, the defuzzification only re-scales the results of fuzzy AND of the initial conditions (Figure 3c). A more classical defuzzification process, which would merge the four obtained scores (Figure 2b) into one, is less straightforward. This is because each possible zoning class can be confused with two others, resulting in a circular relationship (Figure 3b). The defuzzification would have to be performed in such a circular manner. This could be achieved, however it would not change the interpretation of the results. The final choice of one of the zoning classes would still have to be made.

As an example, values of the parameters *YPos* and *Width* for the word "brandy" in Figure 2a are 55% and 25%, respectively. Table 1 presents calculated membership function values and final scores for various possible zoning classes. It can be observed that the upper/middle/lower zone class obtained the highest score. This score is significantly higher than the second highest score, which indicates low ambiguity of the zoning classification process in this particular case. Indeed, the analysed word undoubtedly belongs to the upper/middle/lower zone class.

Table 1: Fuzzy word zoning classification of word "brandy" in Figure 2a.

Zoning class	F_{YPos}	F_{Width}	Score
M	0.834	0.111	11.1%
MU	0.143	0.625	14.3%
ML	0.555	0.714	55.5%
UML	0.857	0.900	85.7%

Table 2 presents the results of word zoning classification of the data used in the recognition experiments. The handwriting data are further described in Section 8. It can be seen that nearly 20% of words were not correctly classified. This rate is similar to results reported by other researchers [10]. However, when considering the two best choices of the word zoning classifier, the majority of words are classified correctly.

Table 2: Effectiveness of word zoning classification.

Position of the correct classification	Percentage correctly classified
1.	81.2%
2.	16.2%
3.	2.2%

2.2 Detection of ascenders and descenders

The zoning classification is used as a guide for detecting ascenders and descenders. They are detected using local extrema. A simple threshold method is used to decide whether particular extrema are ascenders/descenders or belong to the middle zone (Figure 5).

Currently, threshold lines are always horizontal. This may cause problems for some handwriting styles, where letter size is inconsistent. Non-horizontal threshold lines need to be considered. One of the possibilities is to allow tiered threshold lines [8]. This would further increase the flexibility.

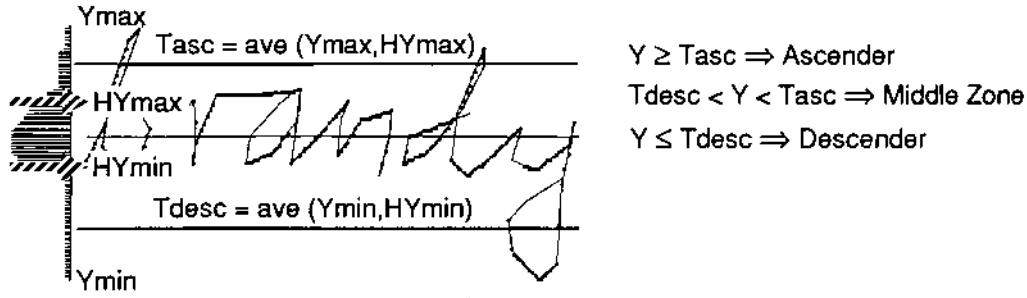


Figure 5. Detection of ascenders and descenders.

Table 3 presents the effectiveness of detecting the ascenders and descenders for different choices of word zoning classification. It can be observed that the results are lower than those of word zoning classification (Table 2). This indicates problems with the method chosen. Irrespective of the word zoning classification choice, the detection of ascenders and descenders failed in 25.1% of cases. However, it has to be remembered that the test did not take into account any individual variations of handwriting style (e.g. middle versus middle/lower zone "z," upper/ middle/lower versus middle/upper zone "f" and others, like "r" with an ascender). Also, not all "i" dots and "t" crosses were eliminated successfully.

Table 3: Effectiveness of ascenders/descenders detection.

Word zoning classification choice	Percentage correctly classified
1.	65.6%
2.	7.4%
Other	1.9%

Other clustering methods, based on vertical proximity of the local extrema points, were applied. These provided similar results.

3 Ascender/descender recognizer

The ascender/descender recognizer uses the sequence and horizontal position of ascenders and descenders located within the word. A number of lexicons were analysed in order to determine the potential of the sequence of ascenders and descenders for word recognition. The analysed lexicons contained between 200 and 15436 words. They were created of the most frequent words of the Oxford Advanced Learners Dictionary [12]. Word frequencies were obtained using the LOB corpus [13] and methods developed by Keenan in his work [14]. The 200 words lexicon also satisfies extra requirements described in Section 8.

The results of the analysis indicate that the sequence of ascenders/descenders alone is not a strong criterion for the recognition of handwritten words. There are relatively few word shape patterns and some of them represent numbers of words. Table 4 presents the distribution of word shape patterns representing various numbers of words. The number of possible word shape patterns varies from 30 to 144, depending on the lexicon size. Few very frequent patterns can be observed.

Table 4: Number of word shape patterns representing given number of word alternatives in lexicons of different sizes.

Number of word alternatives	Number of words in the lexicon				
	200	520	4107	8541	15436
1	12	10	10	28	35
2-10	10	13	31	32	54
11-25	7	3	7	10	13
26-50	1	4	6	9	12
51-100		2	4	6	7
101-250		1	3	5	8
251-500			5	5	5
501-1000			2	3	6
1000+				2	4
Total	30	33	68	100	144

Table 5 presents ten most frequent word shape patterns identified within the analysed lexicons. These ten patterns represent a considerable majority of the entire lexicon. It is clear that other criteria are necessary in order to limit the recognition domain to a small enough size. In this work positions of ascenders/descenders within the word are relative to the word length.

The number and sequence of the ascenders/descenders are used to limit the number of database word patterns which are matched with the unknown word pattern. Most common patterns, which are the worst case, will require around 20% of the lexicon to be considered (see Table 5).

Only exactly matching word patterns are considered. As a result, the recognition rate of the ascender/descender recognizer is limited by the accuracy of detection of ascenders/descenders (see Table 3).

Table 5: Most frequent word shape patterns identified in various size lexicons. Meaning of symbols: 'd' - ascender, 'q' - descender, 'm' - middle zone only.

Word shape pattern	Number of words in the lexicon				
	200	520	4107	8541	15436
d	28 / 14.0%	108 / 20.8%	776 / 18.9%	1413 / 16.5%	2164 / 14.0%
dd	19 / 9.5%	80 / 15.4%	636 / 15.5%	1389 / 16.3%	2336 / 15.1%
ddd	15 / 7.5%	32 / 6.2%	252 / 6.1%	645 / 7.6%	1276 / 8.3%
dq	13 / 6.5%	42 / 8.1%	305 / 7.4%	557 / 6.5%	1004 / 6.5%
qd	22 / 11.0%	33 / 6.3%	301 / 7.3%	640 / 7.5%	964 / 6.3%
m	25 / 12.5%	67 / 12.9%	300 / 7.3%	489 / 5.7%	698 / 4.5%
q	12 / 6.0%	40 / 7.7%	262 / 6.4%	491 / 5.7%	786 / 5.1%
ddq	13 / 6.5%	23 / 4.4%	183 / 4.5%	367 / 4.3%	720 / 4.7%
qdd	8 / 4.0%	11 / 2.1%	120 / 2.9%	330 / 3.8%	656 / 3.0%
dqd	8 / 4.0%	18 / 3.4%	158 / 3.8%	327 / 3.8%	572 / 3.7%
Total	163 / 81.5%	454 / 87.3%	3293 / 80.2%	6648 / 77.8%	11176 / 72.4%

3.1 Word shape encoding

Obtaining word shape from the zoning information is presented in Figure 6. Calculating the zone boundaries (Figure 6a) allows the identification of ascenders and descenders within the word. Their horizontal position within the word provides the word shape information. Details of the middle zone are not taken into account (Figure 6b).

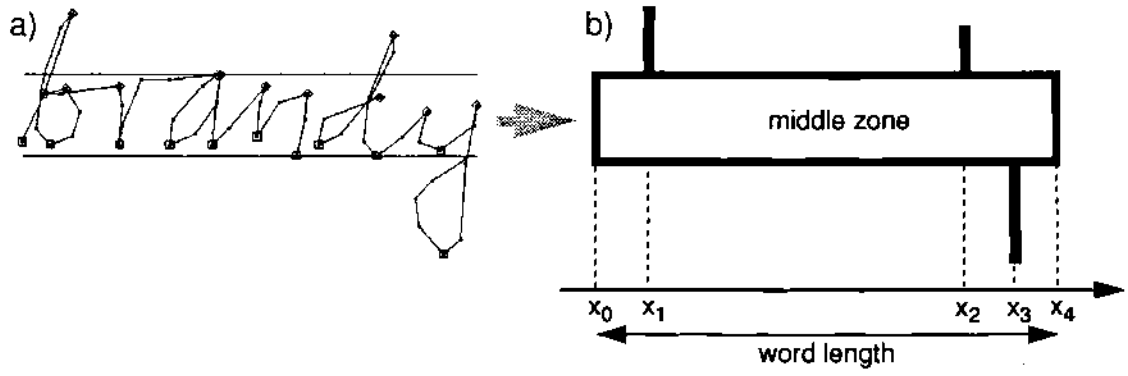


Figure 6. Word shape encoding for the ascenders/descenders wholistic recognizer: a) zone boundaries; b) word shape encoding.

The use of the zoning information introduces errors, as the extraction of the zoning information is not always accurate. In the tests performed, the accuracy of the word shape extraction is below 70% (Table 3), although it can be considerably higher for individual writers.

3.2 Word shape matching

Word shape matching is performed in two stages:

- matching the sequence of ascenders/descenders obtained from zoning information (Figure 7a);
- assessing the similarity of the unknown pattern to all the patterns with identical sequence of ascenders/descenders (Figure 7b).

Matching of the sequence of ascenders/descenders is performed in order to limit the number of words to be considered in greater detail. It is performed in a strict manner. A database of word coding is created using information about the "ideal" shape of perfectly written words. Each letter has certain shape attributes associated with it. Word shape is obtained by concatenating shape attributes of letters composing the word. Details are presented in Section 5.

Once the words with identical sequences of ascenders/descenders have been identified, it is necessary to choose the one most similar to the unknown word. To this purpose a degree of similarity is assessed for each of the identified words. The word comparison takes into account the position of ascenders/descenders within the word. In order to compare different words, their lengths are scaled to be equal.

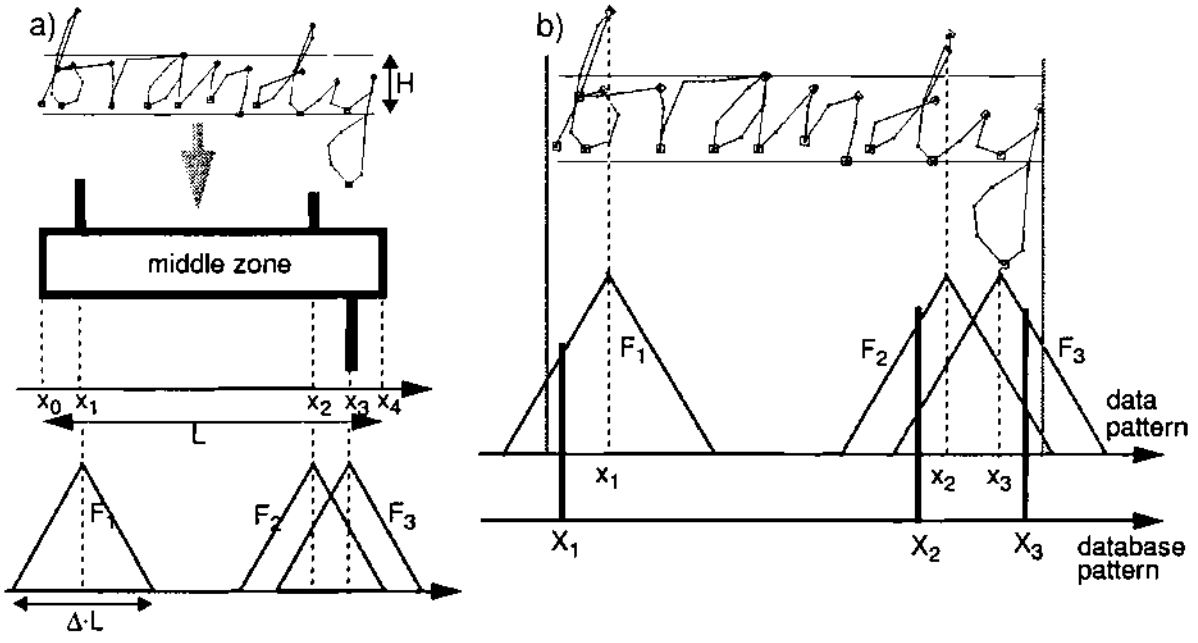


Figure 7. Word shape matching: a) deriving word shape data pattern; b) calculating similarity of the data and database patterns.

Fuzzy logic is applied to the word shape comparison [11]. Positions of ascenders/descenders within the matched word are represented as fuzzy sets (Figure 7a). Symmetric triangular fuzzy sets are used. Their tops correspond to positions of the ascenders/descenders and their widths

are determined using the length of the word (L). Such an approach results in fuzzy sets which are more crisp for short words and less crisp for longer words. This characteristic is desired, as in the case of long words positions of ascenders/descenders can differ from those represented in the database to a greater degree than in the case of short words. The coefficient Δ is chosen experimentally. Its choice determines the strictness of the matching process. The lower it is, the more matches will obtain zero score and will therefore be rejected. However, the value of Δ does not affect the order of the matching result alternatives. In the experiments presented in this paper Δ is set to 0.5.

For each of the fuzzy sets the degree of membership of the position of appropriate ascender/descender within the database is calculated (Figure 7b). When more than one ascender/descender is present, several partial scores are obtained. Two ways of using them are proposed:

- use the worst case, the minimum score;
- use the average score.

The use of the worst case is a severe approach, however it captures well the requirement that *all* the ascenders/descenders are in the expected places. The use of the average score takes into account all the partial scores. However, good scores can still be obtained for some significantly wrong partial scores. The final score is thus calculated using the following formula:

$$ShScore = min(F_1(X_1), F_2(X_2), ..., F_n(X_n)) \tag{2}$$

where $ShScore$ is the final shape similarity score in percent, n is the number of ascenders/descenders, F_i is a degree of membership of the position of i-th ascender/descender of the database pattern within the fuzzy set constructed around the i-th ascender/descender of the unknown pattern, and X_i is the position of the i-th ascender/descender in the database pattern.

The minimum function implements AND in fuzzy logic [11]. This implies that all the ascenders/descenders within the compared words should be in their expected positions in order to obtain a good similarity score.

3.3 Word length matching

After the word shape is matched, the word length is further taken into account in order to distinguish between words of similar shape but different in length. This criterion is particularly important for words with no ascenders/descenders. Since word length is an absolute measure which depends on the size of writing, it is normalised into the relative word length (RWL):

$$RWL = \frac{L}{H} \tag{3}$$

where L is the absolute length of the word and H is the height of the middle zone identified within the word, as in Figure 7a.

The length matching score is calculated as follows:

$$LenScore = \begin{cases} \dfrac{RWL}{DBWL} & \text{for} \quad RWL \le DBWL \\ \dfrac{DBWL}{RWL} & \text{for} \quad RWL > DBWL \end{cases} \tag{4}$$

where *LenScore* is the length matching score, *RWL* is the relative word length and *DBWL* is the length of the word in the word shape database.

The length matching score is used to modify the similarity shape score to produce the final word matching score:

$$Score = ShScore \cdot LenScore \tag{5}$$

3.4 Use of middle zone information

The similarity scoring methods currently used consider the *relative* position of ascenders/ descenders within the word. This makes comparison of words of very different sizes possible. It can also produce high similarity scores for words which are clearly different to human readers.

Figure 8 presents such an example. Original words (a) are normalised, so that their lengths are identical (b). It can be observed, that relative positions of ascenders in the word "terrible" are similar to those in the word "hello." When comparing the word "hello" with the word shape database (Figure 8c) the pattern for the word "terrible" may obtain a higher similarity score than the correct pattern "hello" as is the case in the example.

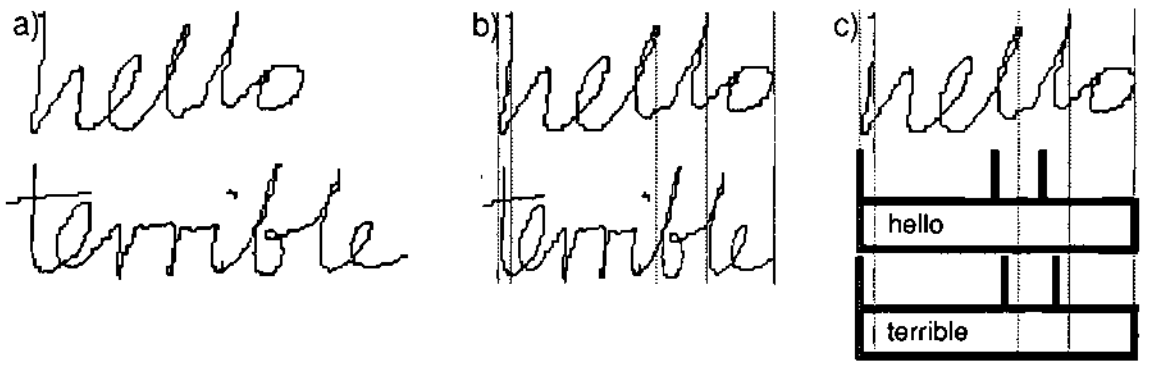


Figure 8. Words considered similar by the word similarity scoring method: a) original words; b) normalised lengths; c) word "hello" versus database patterns for "hello" and "terrible."

The comparison of word length is performed in order to choose the right word. However, the estimate of word length has to be normalised in order to be useful. Height of the middle zone is used as the normalising factor. Unfortunately estimation of the middle zone height can be imprecise and variable for different handwriting styles. This strongly affects the robustness of the word length criterion.

On the other hand, for human readers, the two words in Figure 8a are clearly different. In the context of word shape they are distinguished by the presence of a "t" cross and the information in the middle zone.

Information about dots and dashes is currently not used in the word shape recognition. However, attempts have been made to use the information in the middle zone. The objective has been to determine the presence and possibly the number of middle zone letters between consecutive ascenders/descenders. At the same time no segmentation or letter recognition has been attempted.

It appears that this information can be derived from the horizontal distance between consecutive ascenders/descenders. However, the use of these distances has been discarded for two reasons:

- widths of different letter vary considerably. It is therefore extremely difficult to tell the number of letters present within any area of a word. Also, the letter width is different for different letters;
- the word shape similarity scoring methods currently used already employ distances between ascenders/descenders. This is achieved implicitly, as the relative horizontal position of ascenders/descenders within the word is taken into account.

Another criterion has been found which is expected to be less variable than the letter width. It is the number of times the pen path intersects an imaginary horizontal line drawn through the middle of the word. The criterion is unambiguous and independent of size of the word. It is also used for the word length comparison. The middle of the word is identified as the area of highest horizontal ink density.

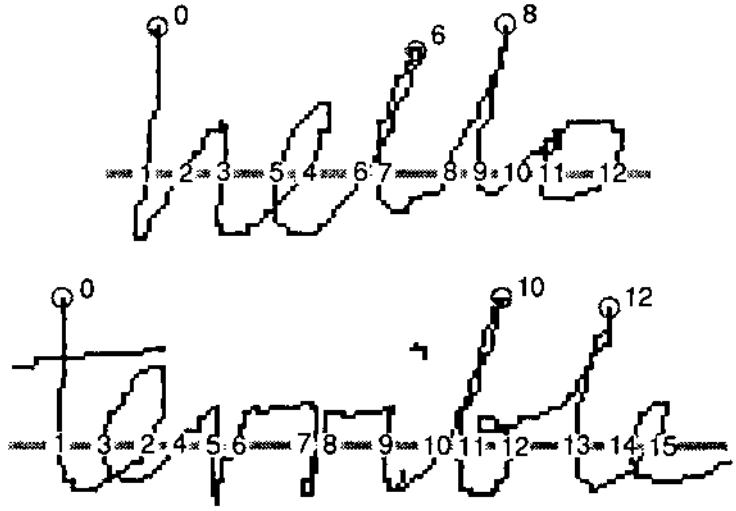


Figure 9. Position of ascenders/descenders in terms of the number of intersections of the pen path with an imaginary horizontal line.

Estimation of the number of letters cannot be very precise, due to different widths of various letters. Different combinations of letters can produce identical number of intersections. On the other hand, a position within a word can be expressed in terms of the number of intersections of the pen path up to that position. This measure is expected to be fairly consistent, as it does not depend on the letter width. Figure 9 presents the two sample words which are described using

the intersection position of ascenders/descenders. The number of intersections between the consecutive ascenders/descenders is also available. A *dissimilarity* score is calculated as an absolute difference between the positions of appropriate ascenders/descenders. This score is further converted into a *similarity* score and normalised, so it can be compared with others.

Conversion from dissimilarity to the similarity score is achieved as a complement of the total number of intersections in the word being recognized. Further normalisation is also performed with respect to the number of intersections in the word being recognized.

The obtained similarity score based on the intersections is taken into account when calculating the total similarity score. It is combined with the score obtained from the relative ascender/ descender position comparison. If either of these scores is zero, the final score is zero. Otherwise, the final score is the average of the two shape similarity scores. Experiments show that the use of the middle zone information improves the results.

4 Vertical bars recognizer

The vertical bars recognizer uses downwards directed and approximately vertical parts of the pen trajectory. Such parts are referred to as *vertical bars*. Their number, sequence, height and vertical position are the features used for the word shape matching. Zoning information is not required. The pattern of vertical bars representing an unknown word is compared to all the appropriate patterns within the database. As a result the vertical bars recognizer provides better recognition rates than the ascender/descender recognizer, especially when dealing with data difficult to zone reliably. By matching all the detected bars, including those in the middle zone, the recognizer implicitly makes use of the information present in the middle zone.

A number of lexicons have been analysed in order to evaluate the potential of the vertical bars recognizer for the wholistic recognition. The same lexicons were used as in Section 3.

Table 6 presents the distribution of vertical bar patterns representing various number of words, where bar size and vertical position are taken into account. An ideal situation is assumed when the zoning classification of each bar is well defined and correct. Special cases for letters "f" and "z" are also taken into account (Section 4.1). It can be observed that the majority of vertical bar patterns produce very limited number of word alternatives. Thus, the size and vertical position of bars appear to be a strong criterion for the wholistic recognition.

Table 6 presents an ideal case. In reality the zoning classification of the vertical bars is not determined. This results in a confusion between some shapes, particularly for less carefully written words. The vertical bars recognizer depends on the detection of the vertical bars. These features are usually highly invariant within the word. However, superfluous vertical bars (Figure 13) were commonly detected in the course of analysis of real handwriting data. This is due to a particular handwriting style and very simple vertical bars identification method used. It is also possible to fail to detect appropriate number of bars (Figure 14). This is mostly due to skipping letters or their parts in sloppy writing. Missing vertical bars were much rarer than superfluous bars in the analysed handwriting data. As a result patterns of similar number of vertical bars also need to be considered. This increases the necessary amount of work to be performed and the chance of confusing word shapes.

Table 6: Number of vertical bar patterns representing given number of word alternatives in lexicons of different sizes. Bar size and vertical position are taken into account.

Number of word alternatives per coding sequence	Number of words in the lexicon				
	200	520	4107	8541	15436
1	165	280	1441	2699	4840
2-10	12	69	542	1069	1917
11-25		2	36	66	129
26-50			7	26	39
51-100			1	4	12
101-250					1
Total	177	351	2027	3864	6938

4.1 Word shape encoding

Vertical bars are identified using local vertical extrema. Preprocessed ink data points [15] are analysed and local vertical extrema are detected. Each ordered pair of a maximum and a minimum within a single stroke results in a vertical bar (Max_n-Min_n in Figure 10).

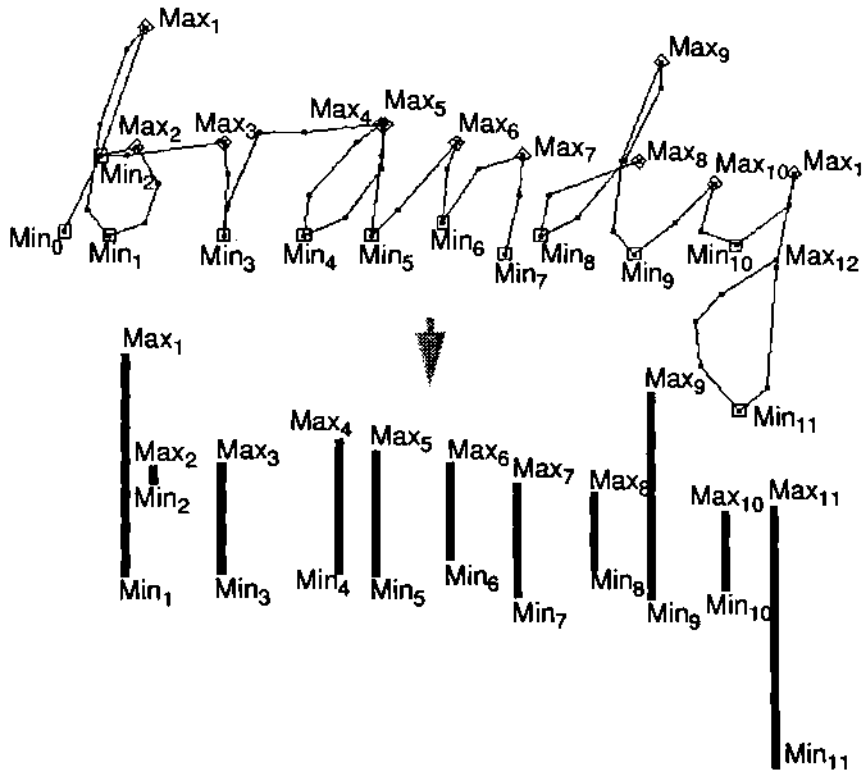


Figure 10. Vertical bars detection.

The order of the extrema is significant as the encoding is designed to represent only downwards directed pen trajectories. The horizontal position of each bar is calculated as the average of horizontal positions of the pair of local extrema defining the bar. Word shape encoding using vertical bars is presented in Figure 10.

The encoding process produces a number of bars of different height and position (Figure 11b), usually different from the ideal encoding (Figure 11c). The better a word is written, the more its bar encoding resembles the ideal.

The definition of vertical bars is based on the way letters are written. Therefore dynamic handwriting information is necessary in order to identify vertical bars. However, because of the way the vertical bars are defined, they are expected to be highly invariant features. Unless letters are written in an entirely unusual manner, they are expected to contain the same set of vertical bars (see Figure 16, Section 5). This expectation fails when letters, or their parts, are omitted within words (Figure 14). It can also fail due to the very simple vertical bars identification method adopted. This can be seen for the extrema pair Max_2-Min_2 in Figure 10 and for examples in Figure 13. Methods of dealing with both cases are discussed in the following sections.

4.2 Word shape matching principles

Word shape matching is performed by calculating the distance between the pattern derived from the data and all the relevant database patterns. Relevant database patterns are identified according to the number of vertical bars. The database pattern with the smallest distance represents the input pattern best. Distances are converted into scores and normalised into the range [0..100].

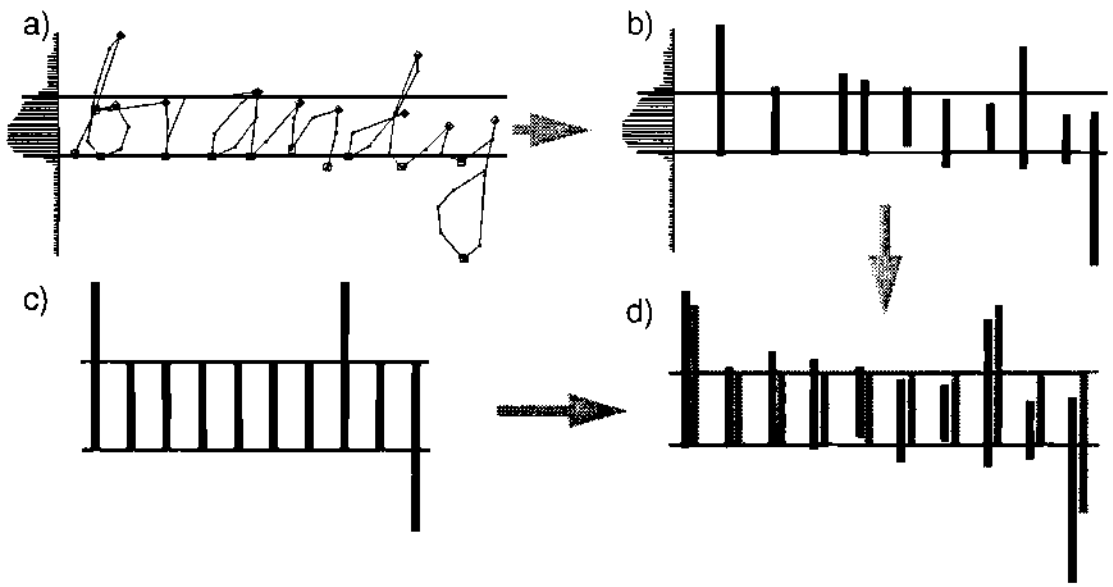


Figure 11. Word shape matching using vertical bars encoding: a) original word with horizontal density histogram; b) vertical bar encoded word with the zone of highest word density; c) database pattern for the word; d) comparison of data and database patterns.

Figure 11 presents word shape matching using vertical bar coding. Area of the highest ink density within the word is used as encoding base for the detected vertical bars (Figure 12). The original word (a) is encoded into the vertical bars (b). Area of the highest density is detected within the original word using the histogram of horizontal word density (a). This area is maintained in the bar encoding (b). The obtained vertical bar pattern (b) is compared with relevant database patterns (c). Figure 11d visualises the comparison process. Note that areas of highest density of both patterns are scaled to be equal. Vertical bars within the data pattern (b) are moved horizontally in order to better depict the comparison. The horizontal bar position is not taken into account in the comparison.

The comparison process takes into account bar sizes and vertical positions. These two parameters are equivalent to the vertical position of top and bottom of each bar (Y_{top} and Y_{btm} in Figure 12).

The position of endpoints of each vertical bar are encoded using two fuzzy sets [11], as depicted in Figure 12. The use of fuzzy sets allows encoding of the distance of each endpoint from the appropriate boundary of the highest density zone. Fuzzy set F_{top} is used for the top of the vertical bar and F_{btm} is used for the bottom of the vertical bar. The obtained fuzzy membership values can be directly compared to values stored in the database. Since the fuzzy sets are defined relative to the width of the highest density zone (H), no further normalisation is required.

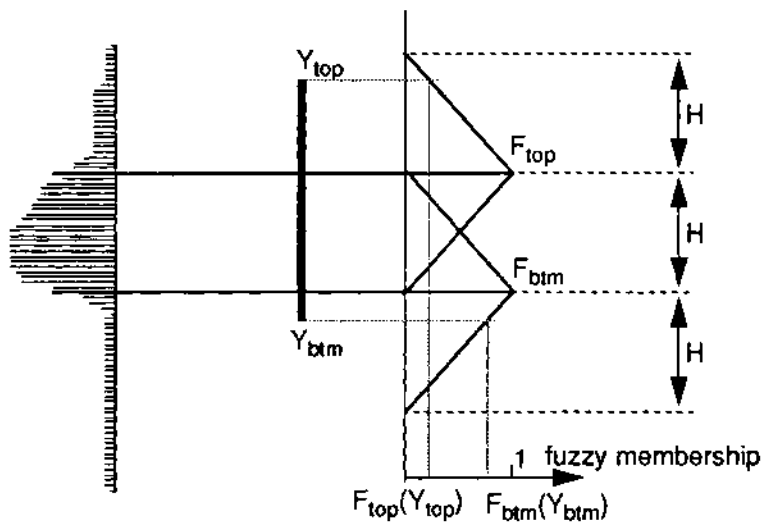


Figure 12. Principles of fuzzy set based encoding of vertical position of bar endpoints.

The distance between the compared patterns is defined as the sum of all the distances between appropriate individual bars:

$$Dist = \sum_{i=1}^{n} D_i \tag{6}$$

where $Dist$ is the distance between patterns, D_i is the distance between i-th bars and n is the number of bars within the pattern.

Distance between individual bars is calculated as follows:

$$D = max(D_{top}, D_{btm}) \tag{7}$$

where *max* function implements AND of differences of fuzzy conditions, D is the distance between individual bars, D_{top} and D_{btm} are differences between vertical positions of the compared patterns (top and bottom of bars, respectively):

$$D_* = |F_*(Y_*) - Fdb_*| \tag{8}$$

$F_*(Y_*)$ is the fuzzy position of bar endpoints in the data pattern (Figure 12), and Fdb_* is the fuzzy position of bar endpoints in the database pattern (top and bottom, respectively), as depicted in Figure 16.

The calculation of distance in equation (6) requires the number of vertical bars in the compared patterns to be identical. This is the simplest, ideal case. In reality however, the detection of vertical bars can fail. This is mostly due to variations in handwriting style or sloppiness. Also, some letters can be written in a number of ways, each of which can have a different bar encoding (e.g. Figure 17a).

Variations in handwriting style may result in superfluous vertical bars (see Figure 13). In such cases the bar encoding derived from the data contains too many bars, which need to be dealt with. An attempt may be made to remove the superfluous bars. These can however be often confused with significant bars, resulting in removing important information. Another possibility is to make the matching algorithm tolerant towards superfluous vertical bars. This is the basis of the vertical bars matching algorithm presented in this paper.

Observation of the collected handwriting data indicates that the presence of downwards directed parts of pen trajectory is highly invariant. However, some writers have been observed to occasionally skip downwards pen movements in their handwriting (see Figure 14). An extension to the matching algorithm is necessary to cope with such cases.

Two new bar patterns comparison algorithms are introduced. The first algorithm allows extra bars only in the data pattern. The extended algorithm allows extra bars both in the data and database patterns. Both algorithms are comparable to dynamic programming [16]. The main difference is that the described algorithms allow only one-to-one assignment of bars within patterns, i.e. a bar which has already been used in the distance calculation cannot be used again.

4.3 Word shape matching algorithm

Superfluous bars have been observed to directly precede ligatures in majority of cases within the collected handwriting data. This allows to reduce the complexity of the comparison. Database patterns contain information about the position of ligatures (see Section 5). During the comparison process, ligature positions in the database pattern may be assigned bars from the data pattern. This way the data pattern bars are ignored. For each ligature position two possibilities are investigated: the ligature is simply skipped, or it absorbs one (presumably superfluous) bar of the data pattern. The match resulting in the lowest distance between patterns is retained.

Figure 13 presents examples of the vertical bar matching process where data samples contain superfluous bars (circled and indicated in the figure). Figure 13a presents the expected outcome of the matching process. The first ligature is omitted and the second absorbs the small superfluous bar. Figure 13b presents a difficult case. Two consecutive superfluous bars are present. In such a case the first superfluous bar is absorbed by the ligature in the database pattern. However, the second superfluous bar cannot be absorbed. It has to be used for the matching. As a result, a valid bar is ignored in order to allow the patterns to be matched. The obtained distance between patterns is higher than desired.

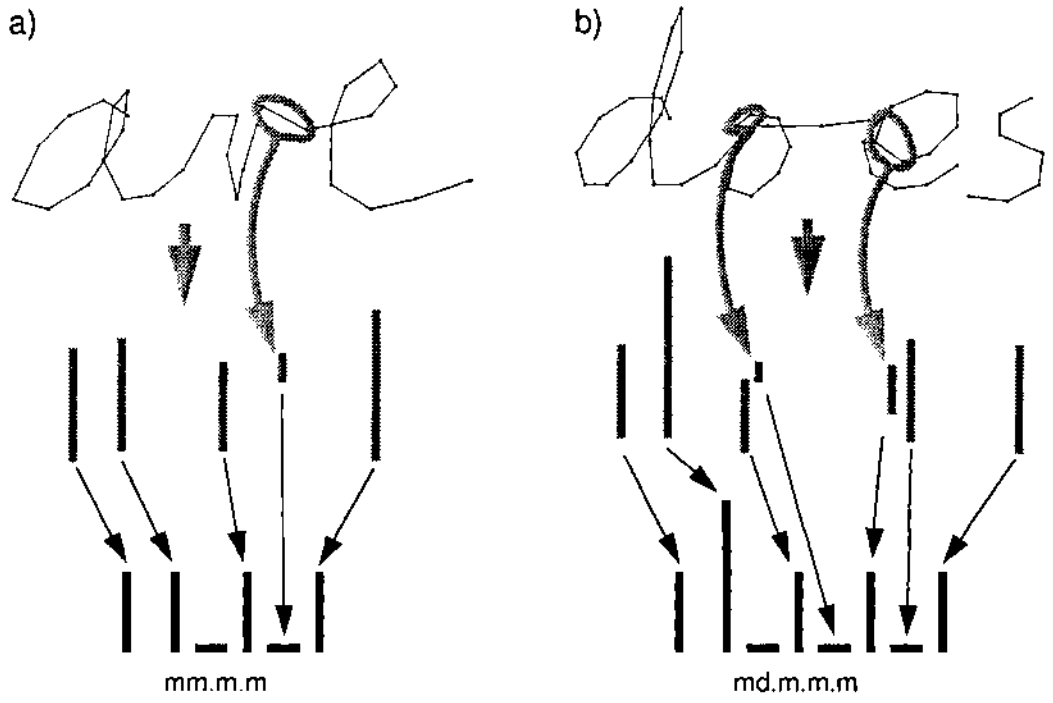


Figure 13. Vertical bar pattern matching with tolerance for superfluous bars in the data pattern: a) regular situation; b) irregular situation: two consecutive superfluous bars. Arrows represent the best match, horizontal bars within database patterns represent

Comparisons where no bars are ignored are preferred, as all the information within patterns is used. Hence each ignored bar increases the distance between patterns. The distance function from equation (6) is extended in order to accommodate this:

$$Dist = \sum_{i=1}^{n} D_i + \sum_{j=1}^{m} Pd_j \tag{9}$$

where Pd_j is the distance penalty incurred by the j-th ignored bar and m is the total number of ignored bars within the data pattern. The distance penalty for vertical bars in the data pattern is calculated as follows:

$$Pd = \frac{h}{H} \tag{10}$$

where h is the height of the bar and H is the height of the zone of the highest ink density within the word (see Figure 12). The function is designed to be proportional to the size of the discarded bar. Superfluous vertical bars are usually small. Hence the penalty for ignoring them ought to be small. In case of larger vertical bars, it may be difficult to judge whether they are superfluous or not. The distance penalty for ignoring them is thus larger.

When matching data and database patterns the objective is to minimise the distance function (see equation (9)). This is achieved by an algorithm akin to dynamic programming:

$$f(\text{no more data bars, no more database bars}) = 0$$

$$f(i,j) = \begin{cases} D(i,j) + f(i+1, j+1) & \text{database bar not followed by ligature} \\ min\begin{pmatrix} D(i,j) + f(i+1, j+1) \\ Pd(i) + f(i+1, j) \end{pmatrix} & \text{database bar followed by ligature} \end{cases} \quad (11)$$

Where f is a recursive function calculating the minimum distance between the patterns. The algorithm starts from the beginning of the patterns. Vertical bars in the data and database patterns are indexed by i and j, respectively. Functions D and Pd are defined in equations 7 and 10, respectively.

4.4 The extended matching algorithm

Some writers omit parts of letters in their writing. Words can often be read (guessed), although it is impossible to locate all the individual letters (Figure 12). Since parts of letters may be missing, the vertical bar encoding of a word will contain fewer patterns than necessary. An extended matching algorithm is introduced for this purpose.

Unfortunately, no regularity has been observed on the position of the missing data bars. Writers tend to miss out parts of middle zone letters. No missed out ascenders or descenders have so far been observed. The matching algorithm thus allows to skip every middle zone vertical bar in the database pattern. The match with the lowest distance is retained. Figure 14 presents an example of the matching process discarding superfluous bars in the database pattern. The superfluous bars are encircled. As the middle zone bars in the database patterns are identical, any of two bars in the middle area of the word can be discarded without affecting the distance between compared patterns.

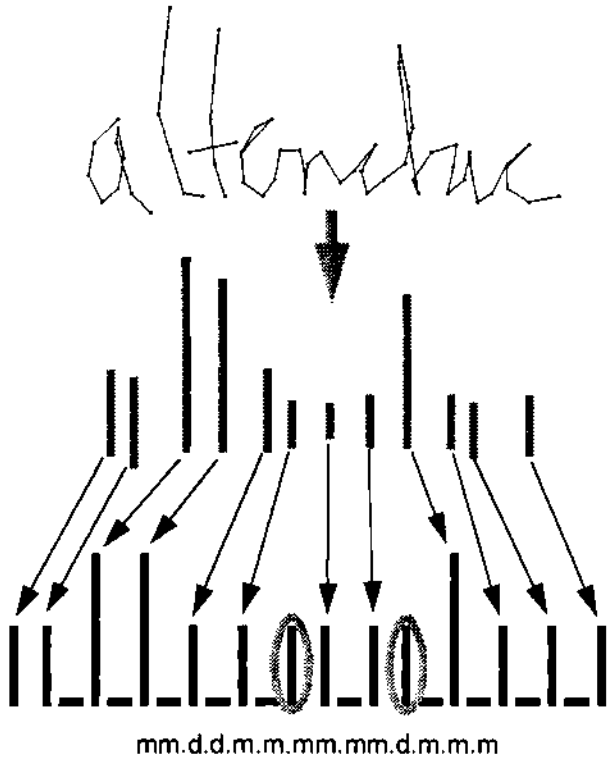


Figure 14. Vertical bar pattern matching with tolerance for superfluous bars in the database pattern. Arrows represent the best match, horizontal bars within database patterns represent ligatures.

Superfluous bars in the data pattern are still likely to occur. Hence the extended comparison method allows them, as well as the extra bars in the database patterns. Again, any discarded bar incurs a penalty. The distance function from equation (9) is further extended and becomes:

$$Dist = \sum_{i=1}^{n} D_i + \sum_{j=1}^{m} Pd_j + \sum_{k=1}^{l} Pdb_k \tag{12}$$

where Pdb_k is the distance penalty incurred by the k-th ignored bar and l is the total number of ignored bars within the database pattern. The distance penalty for vertical bars in the database pattern is calculated as follows:

$$Pdb = 3FuzzyTop - Fdb_{top} - Fdb_{btm} \tag{13}$$

where *FuzzyTop* is the maximum membership value of fuzzy sets used to define the position of bar endpoints, Fdb_{top} and Fdb_{btm} are fuzzy positions of the top and bottom endpoints of the database pattern bar, respectively (see Figure 16c).

As it was the case with the matching algorithm presented in the previous section, the objective is to minimise the distance function (see equation (12)). This is achieved in a manner similar to that previously used (see equation (11)):

$$f(\text{no more data bars, no more database bars}) = 0$$

$$f(i,j) = \begin{cases} \min\begin{pmatrix} D(i,j) + f(i+1,j+1) \\ Pdb(j) + f(i,j+1) \end{pmatrix} & \text{database bar not followed by ligature} \\ \min\begin{pmatrix} D(i,j) + f(i+1,j+1) \\ Pd(i) + f(i+1,j) \\ Pdb(j) + f(i,j+1) \end{pmatrix} & \text{database bar followed by ligature} \end{cases} \quad (14)$$

where f is a recursive function calculating the minimum distance between the patterns. The algorithm starts from the beginning of the patterns. Vertical bars in the data and database patterns are indexed by i and j, respectively. Functions D, Pd and Pdb are defined in equations 7, 10 and 13, respectively.

As previously mentioned, the matching process is performed for all relevant patterns in the word shape database. These are selected using the number of vertical bars in the word pattern. Due to possible superfluous or missing bars in the data pattern, patterns shorter and longer (only shorter for the algorithm described in Section 4.3) than the unknown one also have to be considered. A threshold is set to limit the allowed number of superfluous or missing bars. This threshold is experimentally set to three bars. Obviously, the algorithm which only allows superfluous bars in the data pattern does not consider any database patterns shorter than the data pattern matched.

5 Generation of word shape database

A database of word shape information is necessary for the word shape recognition. Such a database can be created either by analysing real handwriting data, or by using assumed information about ideal letters and words.

Ascenders/descenders, as well as the vertical parts of letters, are relatively invariant features of letters. Unless some writing errors occur, they are present within letters. For this reason information about letters and words is used for creation of the word shape database. Hence, no large amounts of handwriting data are needed to create the database. New words can be easily added to the database. Furthermore, it is possible to create word shape patterns from the text representation of words dynamically, during the recognition process.

Each letter has certain information associated with it. For instance, letter "d" has two vertical bars, one ascender, no descenders, its height is about twice its width, the ascender is positioned on the right edge of the letter and the pen path crosses an imaginary horizontal line three times.

It is clear that such information is generic and may differ for a particular handwriting style. Sometimes the differences may be large enough to make correct recognition difficult (e.g. "z" with a descender when no descender is expected - see Figure 17c). Also actual letter metrics

(width, position of ascenders/descenders, etc.) may differ from the assumed values. The model can greatly benefit from adjusting to a particular handwriting style.

a)

a	b	c	d	e	f	g	h	i	j	k	l	m
1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	0.5	0.5	1.0	0.5	1.5
n	o	p	q	r	s	t	u	v	w	x	y	z
1.0	1.0	1.0	1.0	0.5	1.0	0.5	1.0	0.7	1.3	1.0	1.0	1.0

b)

a	b	c	d	e	f	g	h	i	j	k	l	m
0.0	0.0	0.0	1.0	0.0	0.2	0.5	0.0	0.0	0.0	0.0	0.0	0.0
n	o	p	q	r	s	t	u	v	w	x	y	z
0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0

c)

Zone	Letters
middle	aceimnorsuvwxz
middle/upper	bdhklt
middle/lower	gjpqy
all	f

d)

a	b	c	d	e	f	g	h	i	j	k	l	m
3.0	2.5	1.0	3.0	2.0	2.0	3.0	3.0	1.0	1.0	3.0	1.0	5.0
n	o	p	q	r	s	t	u	v	w	x	y	z
3.0	2.0	3.0	3.0	2.0	1.0	1.0	3.0	2.0	3.0	2.5	3.0	1.0

e)

ligature width	0.1
ligature number of crossings	1.0

f)

a	b	c	d	e	f	g	h	i	j	k	l	m
0.0	0.0	0.0	2.0	0.0	1.5	3.0	0.0	0.0	1.0	0.0	0.0	0.0
n	o	p	q	r	s	t	u	v	w	x	y	z
0.0	0.0	1.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0

Figure 15. Assumed letter characteristics for the ascender/descender recognizer: a) width; b) position of the ascender/descender; c) zoning classification; d) number of crossings; e) ligature parameters; f) intersections position of ascender/descender within the letter.

The letter characteristics are assumed using general writing rules. The assumptions used for the ascender/descender recognizer are presented in Figure 15. Widths of letters are expressed in units of the middle zone height. Positions of ascenders/descenders are expressed relative to particular letter widths. The number of crossings of the pen path with an imaginary horizontal line takes into account different ways of writing letters. Hence it can also be fractional. For instance, letter "b" can be written in two ways, with two or three crossings. Thus the number of

crossings assumed for 'b' is 2.5. Figure 15f presents metrics for extra information necessary to utilise the information in the middle zone of words. It describes the number of intersections *within* a particular letter before the ascender/descender is reached.

Figure 16 presents the encoding of word shape database for the vertical bars recognizer. Five types of vertical bars are identified (Figure 16c). Three of them represent vertical bars in the middle, middle and upper, and middle and lower zones (symbols m, d and q in Figure 16c, respectively). Two other types of vertical bars are introduced to deal with special cases: letters "f" and "z" (symbols f and z in Figure 16c, respectively). Letters are represented by the combination of appropriate bars, as shown in Figure 16a. Bar endpoints assume fuzzy values presented in Figure 16d. Two letters of the alphabet, "f" and "z" are treated differently. Bottom endpoints of vertical bars representing these letters are mid-way between middle and lower zone position. These bars reflect two different variations for each letter (see Figure 17b, c). Bar f nearly reaches the lower zone line. It will provide better match for letters "f" which span all three zones. Bar z ends near the middle zone. It will provide better match for letters "z" which appear in the middle zone only. Values in the shaded cells of the table in Figure 16d are experimental and reflect only the *preferences* for the individual variations of letters "f" and "z." Letter width metrics (Figure 16b) are identical to those used by the ascender/descender recognizer (see Figure 15a).

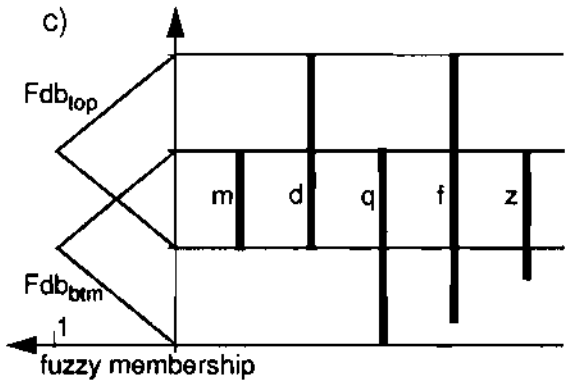
a)

a	b	c	d	e	f	g	h	i	j	k	l	m
mm	d	m	md	m	f	mq	dm	m	q	dm	d	mmm
n	o	p	q	r	s	t	u	v	w	x	y	z
mm	m	qm	mq	m	m	d	mm	m	mm	m	mq	z

b)

a	b	c	d	e	f	g	h	i	j	k	l	m
1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	0.5	0.5	1.0	1.0	1.5
n	o	p	q	r	s	t	u	v	w	x	y	z
1.0	1.0	1.0	1.0	0.5	1.0	0.5	1.0	0.7	1.3	1.0	1.0	1.0

c)



d)

	Fdb$_{top}$	Fdb$_{btm}$	
m	1.0	1.0	middle
d	0.0	1.0	mid/up
q	1.0	0.0	mid/low
f	0.0	0.2	letter "f"
z	1.0	0.8	letter "z"

Figure 16. Encoding of word shape database for the vertical bars recognizer: a) vertical bar coding symbols assigned to letters of the alphabet; b) letter width metrics; c) vertical bars used in the encoding; d) values of fuzzy positions of endpoints for the vertical bars.

The mechanism for dealing with superfluous bars in data patterns allows extra bars only at the end of each letter. Thus the database encoding must contain information about the beginning and end of each letter. This can be easily achieved by inserting a special "ligature" symbol between bar patterns for each letter.

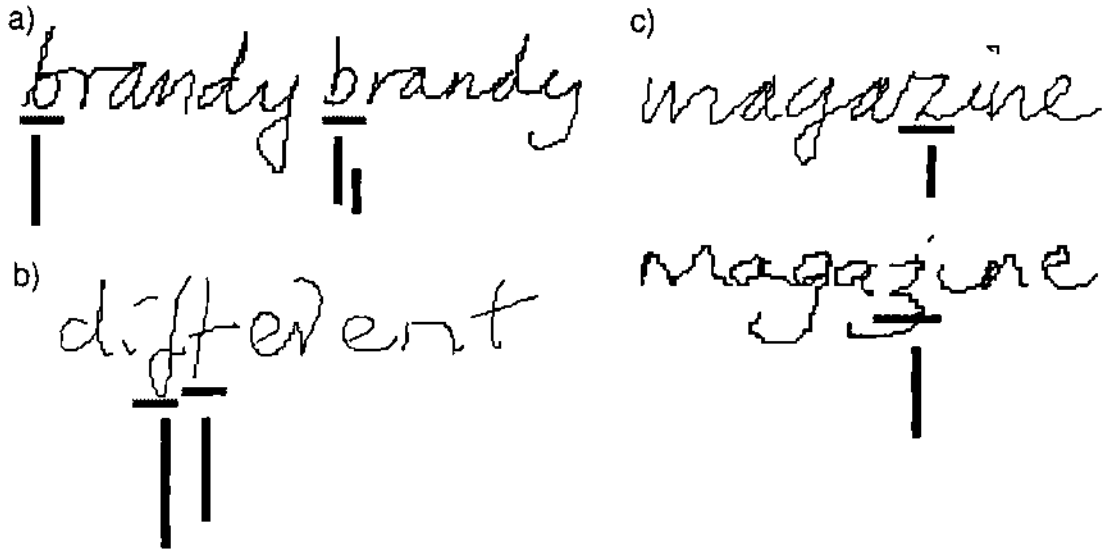


Figure 17. Examples of problems with generating letter templates: a) different ways of writing letter "b" resulting in different number of vertical bars; b, c) different ways of writing letters "f" and "z" resulting in bars of significantly different heights.

Database patterns generated using the described model are not expected to ideally match vertical bar patterns obtained from the real data. The objective is to assess the degree of similarity. For closer modelling of the real data the extraction of information from handwriting data may be necessary.

Letters which are written differently from the assumed model can cause problems. Figure 17 presents some examples. Two classes of problems can be distinguished:

- different ways of writing a particular letter, which results in a completely different encoding (e.g. letters "b" and "z" in Figure 17a, c);
- different ways of writing a letter, which results in a distorted encoding (e.g. letters "f" and "z" in Figure 17b, c).

The ascender/descender recognizer inherently fails when ascenders/descenders are detected incorrectly. Hence different ways of writing some letters (e.g. Figure 17b, c) can significantly influence the results. When the ascenders/descenders are detected as expected, differences in the way letters (e.g. Figure 17a) are written are less significant. They affect the recognition only if the middle zone information is taken into account.

In the case of the vertical bars recognizer both classes of problems are addressed by the matching algorithm. The matching algorithm is capable of comparing patterns of different number of vertical bars. Thus the first class of problems is addressed. The second class of problems is addressed by the use of fuzzy sets for representing positions of bars endpoints.

6 Letter verification

Since a very simple set of features and a relaxed matching algorithm are used, wholistic recognizers described in this paper are frequently capable of correct recognition, but have difficulties choosing the correct alternative as the best answer amongst other words of similar shape. Additional features like loops, arcs, cusps, concavities and convexities (e.g. a set of features described in [17]) could be used to make the recognizers more discriminative. These extra features could be applied to the results of the wholistic recognition in order to modify their recognition scores, in a fashion similar to that of feature tools described in [18].

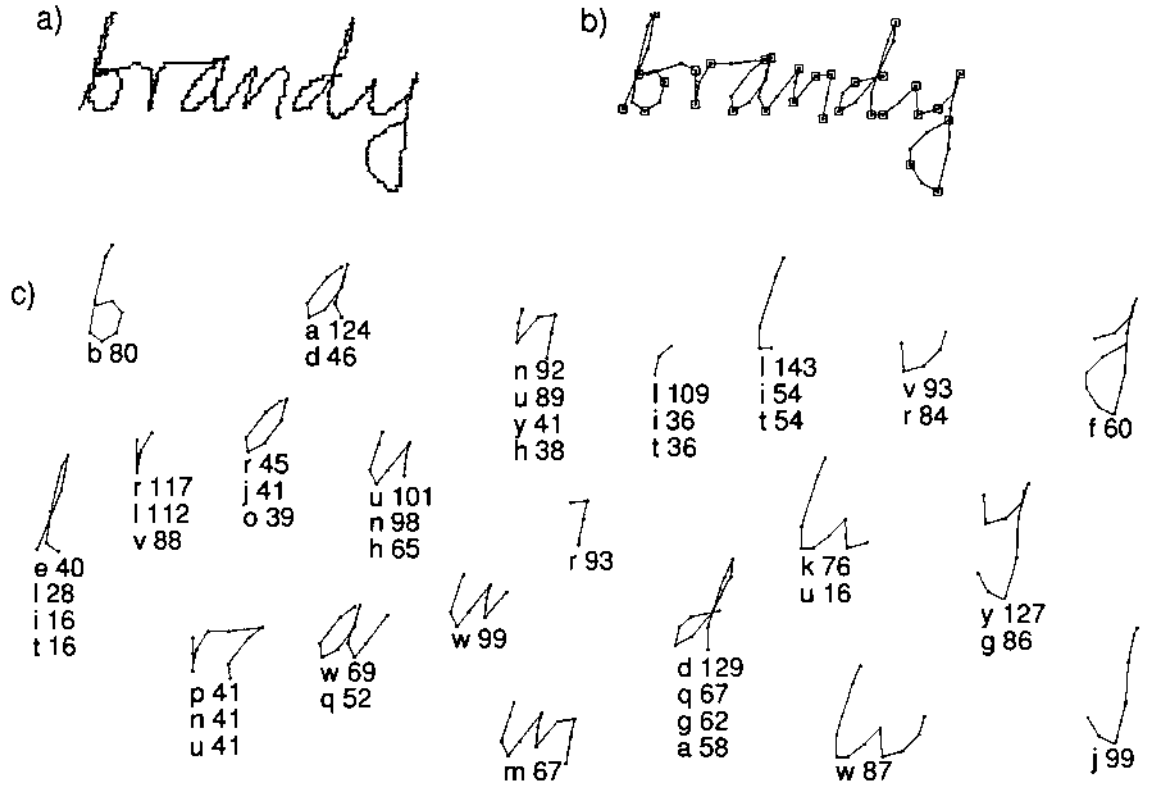


Figure 18. Working of the multiple interactive segmentation recognizer: a) input data; b) identified potential segmentation points; c) some of the identified letter alternatives. Letter recognition results are provided together with the obtained scores. Observe that

Letter alternatives located and recognized by the multiple interactive segmentation recognizer [1] can also be used for verification. Letters are combinations of the previously mentioned features, hence they could be used as compound features. The number and confidence scores of located letters can be used to judge to what extent the proposed word is similar to the input data. For each word alternative proposed by the word shape matching, the letter alternatives provided by the multiple interactive segmentation (see Figure 18) can be searched for the best combination composing the proposed word. This is referred to as the *letter verification.*

6.1 Verifying wholistic recognition results with letter alternatives

The multiple interactive segmentation process identifies parts of the input data which can potentially be letters. Such potential letters can overlap, as letters can frequently be identified within other, larger letters. Each potential letter is recognized (see [2]), resulting usually in a number of alternative results. Typical outcome of the described process is depicted in Figure 18c.

Obtained letter alternatives can be put together into strings. These strings represent possible recognition results. Two rules are followed in concatenating letter alternatives. No two different letter alternatives may share any part of their data with each other. Secondarily, the letter alternatives have to be merged in the direction of writing, from left to right. Finally, a lexicon is applied to eliminate invalid words [1].

In effect, the rules controlling the merging of the letter alternatives impose an order on these alternatives. The set of the letter alternatives produced by the multiple interactive segmentation can be viewed as a directed acyclic graph. The process of identifying word alternatives is equivalent to finding all the valid paths through this graph.

The letter verification works differently. Its essence is to find the best path through the provided acyclic graph of the letter alternatives, which would correspond to the word being verified. This process is repeated for every word alternative produced by a wholistic recognizer.

Figure 19 presents the overview of the letter verification. The input data is processed independently by a wholistic recognizer and the multiple interactive segmentation recognizer. The wholistic recognizer produces a list of word alternatives and their recognition scores. The multiple interactive segmentation provides the letter alternatives it identified[1]. Both word alternatives produced by the wholistic recognizer and letter alternatives produced by the multiple interactive segmentation constitute the input for the letter verification. The letter verification produces as a result a new list of word alternatives. The list can contain only the words which were provided by the wholistic recognizer. Their scores, however, are updated by the letter verification, resulting in the correct alternative moved closer to the top of the list.

The mapping of a word onto the letter graph is performed recursively, starting with the first letter of the word alternative. Any number of letters of the word are allowed to be missing within the graph. After the path through the letter graph is found, scores of the located letter alternatives are averaged in order to provide the score of the letter verification process. The mean is computed over the number of letters present in the word alternative being verified. In effect, the more letters are located, and the higher their scores are, the better is the letter verification score.

1. It can, of course, also provide its own recognition results, however this is of no concern here.

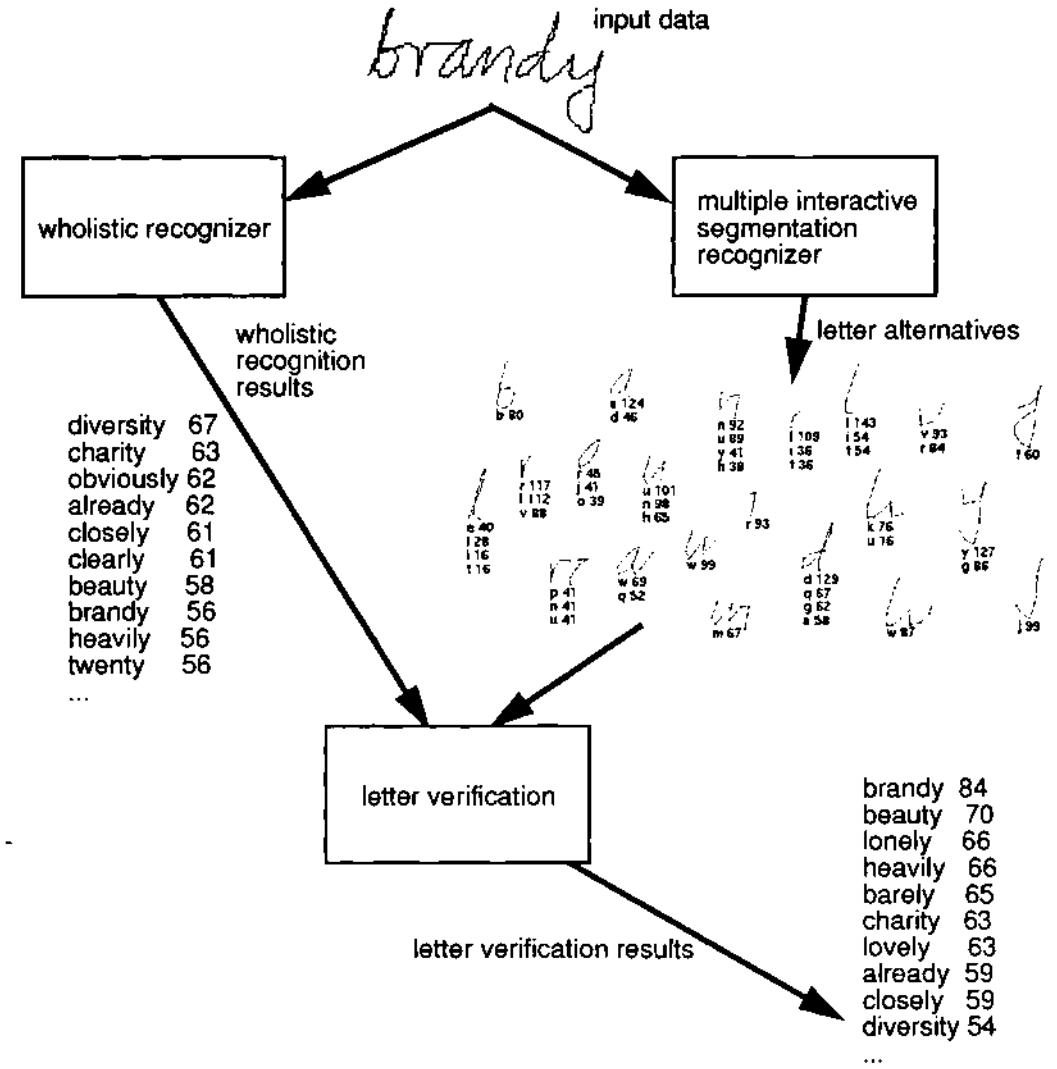


Figure 19. Overview of the letter verification.

Figure 20 presents results of the letter verification process for two of the alternatives produced by the example in Figure 19: "diversity" and "brandy". The letter verification process attempted to map each of these alternatives onto the provided letter graph. When mapping the alternative "diversity", letters "d", "e" and "s" could not be found in the letter graph (see Figure 20a), resulting in a match "#iv#r#ity" (where # denotes a wildcard). The score of such a match is the sum of scores of letters "i", "v", "r", "i", "t" and "y" divided by nine (number of letters in the word alternative). It can be seen that the score will be low, because not all the letters have been found within the graph and those which are present, have low recognition scores. On the other hand, the alternative "brandy" can be mapped onto the letter graph, resulting in a match not requiring any wildcards (see Figure 20b). The resulting score will be higher, as there is no penalty brought by the wildcards. The ordered list of word alternatives, after the letter verification, can be seen in Figure 19.

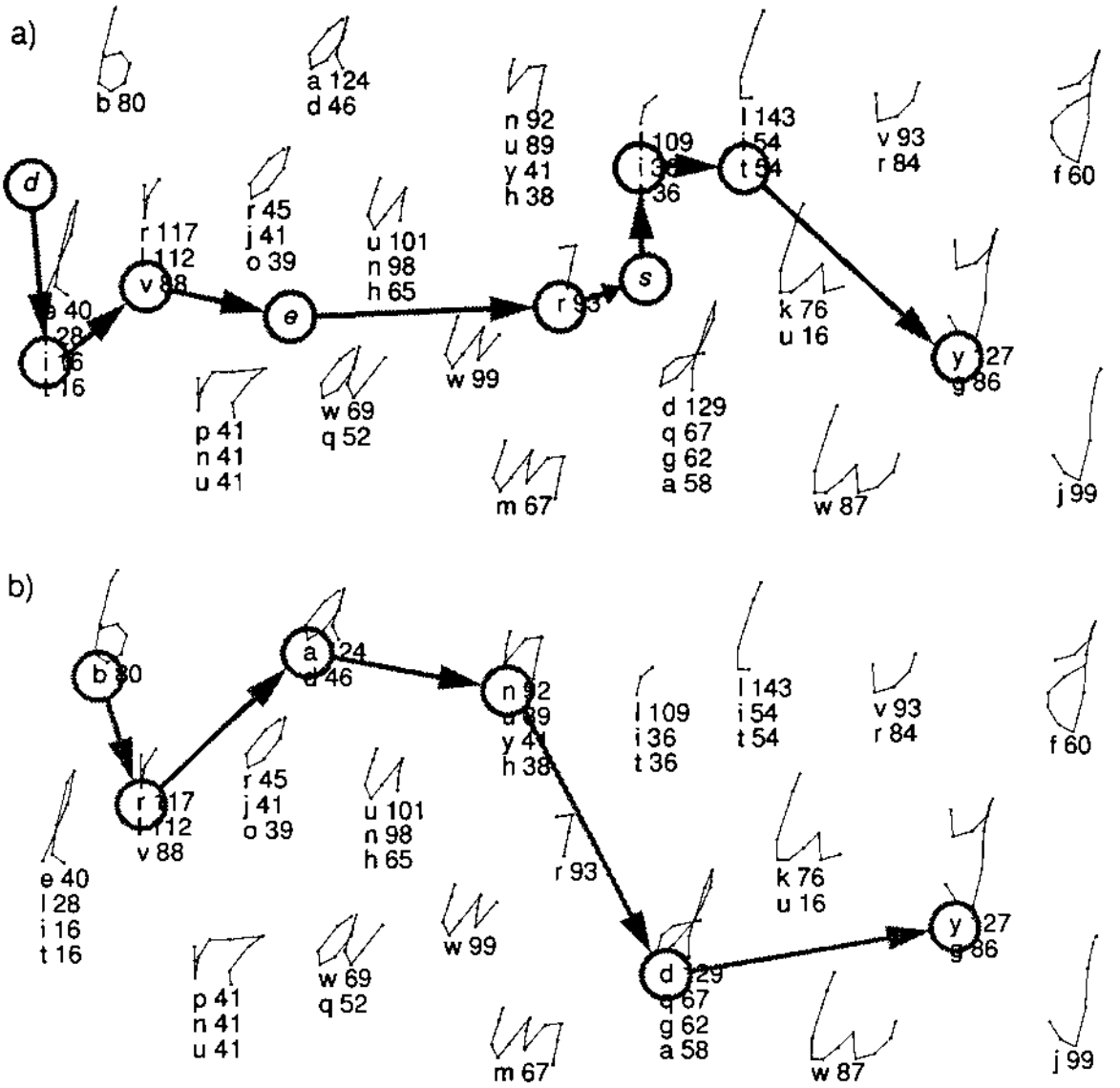


Figure 20. Using letter alternatives provided by the multiple interactive segmentation recognizer for the letter verification of two word alternatives produced by a wholistic recognizer: a) "diversity"; b) "brandy". Letters which are italicised were skipped were

Scores obtained both by the wholistic recognizer and the letter verification are integrated into a single score using a weighted arithmetic average. Changing the weights allows to adjust influences of input recognition scores on the final score. Currently equal weighting factors are assumed.

6.2 Formulating the letter verification expectations

In order to avoid considering letter alternatives irrelevant in the context of a particular word alternative, a set of expectations is formulated for the positions of each letter within the word being verified. The expected letter position is calculated using letter width metrics (Figure 15a).

Figure 21b presents the set of letter position expectations formulated for the word alternative "brandy" (which is the correct recognition result). The required letter width metrics are presented in Figure 21a. Assumed letter widths are scaled such that their sum (including ligatures) is equal to the length of the word. Zones of the width Δ are positioned in the middle of the expected position of each letter. These zones define letter position expectations. Letter alternatives considered in the letter verification process have to intersect with the relevant zone. Otherwise they are not considered. The width of each zone limiting letter positions (Δ) is derived from the width of the middle zone (*H*) of the word. It is identical for all the word alternatives considered for particular data.

a)

Letter	Width
b	1.0
r	0.5
a	1.0
n	1.0
d	1.0
y	1.0
ligature	0.1

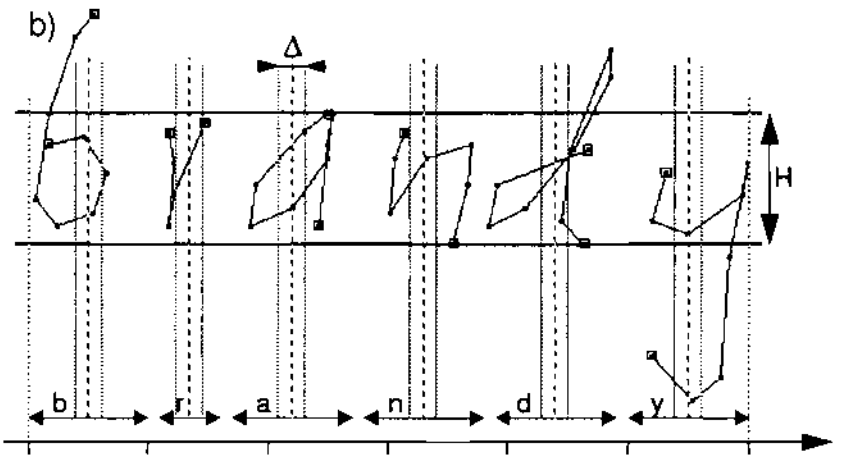


Figure 21. Limiting positions of letters in the letter verification process. The verified word alternative is "brandy." a) letter width metrics (as in Figure 16b); b) derivation of letter position limits.

6.3 Limiting computational intensity of the letter verification

Due to the nature of the multiple interactive segmentation process and the letter recognizer used by the multiple interactive segmentation [2], the generated graphs of letter alternatives can often be large. As any number of letters can be allowed to be missing, analysing large letter graphs can be computationally intensive. Two ways of limiting the computational intensity of the letter verification are introduced:

- limiting the position at which each letter alternative is allowed within the word;
- keeping track of previously considered letter alternatives at each position within the word.

The first approach is straightforward. Any number of letters are allowed to be missing from the path throughout the letter graph. If the position of letters is not limited, it is possible for the analysis to lift the score of any approximate substrings of the word being matched. It is also possible to assume the existence of a number of letters that are not recognized within a very confined space where it is impossible for them to be located. The limitation is imposed by the set of expectations for the letter positions within particular word alternatives. Adjusting the width of each zone limiting letter positions (Δ) allows to control the tolerance of the letter position limits and the amount of work necessary for the letter verification process.

The second limitation stems from the analysis of the generated letter alternatives. Multiple interactive segmentation frequently produces a number of segmentations for each letter. Each of the obtained segmentations can result in a number of letter alternatives produced by the letter recognizer. For letter verification purposes letter alternatives with best scores are most significant. It is therefore possible to limit the computational intensity by considering at each position within the word only letter alternatives which have better scores than the previously considered ones. If a newly considered letter alternative has lower score than one previously encountered at the same position within the word, this branch of the letter graph analysis is abandoned.

Tests show that the imposed limitations significantly reduce the processing requirements of the letter verification. The saving depends on the word length. For the word "brandy" (Figure 21b) limiting the position at which each letter alternative is allowed within the word reduced the computations by a factor of 20. Keeping track of previously considered letter alternatives at each position within the word reduced the computations by a factor of 7. Applying both limitations reduced the computational intensity by a factor of 42. Savings are still larger for longer words.

7 Hybrid system

The segment and recognize approach represented by the multiple interactive segmentation [1] and the wholistic recognition approaches, presented in this paper, both have certain advantages and disadvantages.

The multiple interactive segmentation based recognizer always locates and recognises single letters. Hence it is expected to produce more reliable results. No time is spent searching the lexicon or verifying all the word alternatives with similar shapes. However, this method will fail on words in which individual letters cannot be located and/or recognized.

Wholistic recognizers, on the other hand, are expected to cope with less clearly written words. The success of the word shape recognition depends on the extraction of the zoning information. Also, as the lexicon size grows, several words can be found to have very similar shapes. As a result, the word shape recognition has difficulty in choosing the correct answer as the best alternative, even though it is likely to find the correct word more often.

In order to maximise advantages of both approaches, they are combined into a *hybrid system* [2]. Both multiple interactive segmentation and word shape recognizers are applied, independently and in parallel. Their results are further combined. The objective is to produce the correct word alternative even for difficult to recognize words and to increase the ranking of the correct word alternative proportional to the number of letters that can be identified and recognized.

When the recognition process is complete, lists of word alternatives obtained from different recognizers are merged. When a word alternative is produced only by a single recognizer its score remains unaffected. When a word alternative is produced by more than one recognizer, new scores are calculated using the recognition scores provided by individual recognizers:

$$sc = \alpha \cdot fn(sc_1, sc_2), \alpha > 1 \qquad (15)$$

where sc is the combination score, sc_1 and sc_2 are the scores provided by individual recognizers and α is the confirmation factor indicating how much the confirmation of results by another recognizer should improve the score. fn is the function used for merging the individual recognition scores. Recognition experiments show that arithmetic average performs better than the geometric average and maximum functions. The results combination is biased towards the multiple interactive segmentation based recognizer. The biasing factor has been arrived at experimentally.

8 Results

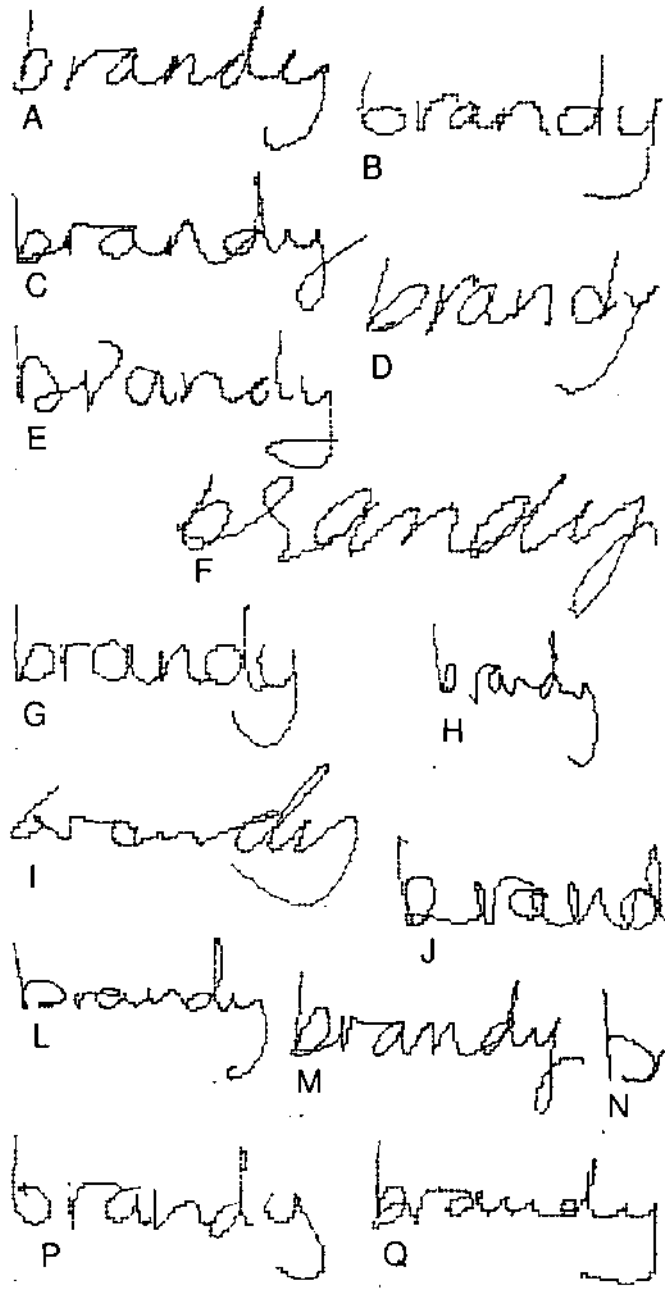
Tests were performed to assess the performance of the various recognizers presented in this paper. Handwriting data were collected from eighteen writers. Each writer wrote the same 200 words, one to sixteen letters long. The 200 words list contains words frequently used in English language. Also, care was taken so that the words in the list contain all the letters of the alphabet as well as have similar distributions of letters and lengths as it is found in a list of 70502 words extracted from the Oxford Advanced Learners Dictionary [12]. The correlation coefficients[1] *are: $r = 0.95$ ($t = 14.72$, $df = 24$) for the letter distribution, and $r = 0.82$ ($t = 6.66$, $df = 21$) for the* word length distribution.

An NCR 3125 pen computer was used as an input device. Sixteen writers are right-handed, two are left-handed. Thirteen writers have had no previous experience with the "electronic paper." Details of the selection of the word list, the data and the data collection process can be found in [19]. Examples of each writer's handwriting can be seen in Figure 22. As the recognizers do not require any training, all eighteen data sets were used for testing. A medium size lexicon of 4107 words was used. The 200 words list is a subset of the 4107 words lexicon. Both the 200 words list and the 4107 words lexicon are the same as those used for the analysis of the wholistic recognizers potential. Their origins are described in Section 3.

Figure 23 presents results obtained for various recognizers:

- multiple interactive segmentation (SEG). Results are provided for comparison only;
- ascender/descender (AD);
- vertical bars (VB);
- ascender/descender with letter verification (ADLV);
- vertical bars with letter verification (VBLV);
- hybrid SEG x AD;
- hybrid SEG x VB;
- hybrid SEG x ADLV;
- hybrid SEG x VBLV.

1. The Pearson correlation coefficient was used.



Writer	SEG	AD	VB	ADLV	VBLV
A	-	2	10	1	1
B	1	1	2	1	1
C	1	14	2	7	8
D	-	-	12	-	18
E	1	-	4	-	1
F	-	-	-	-	-
G	1	1	3	1	1
H	-	13	-	13	-
I	-	8	3	1	1
J	-	6	12	2	2
K	-	-	3	-	9
L	-	1	17	1	3
M	1	1	2	1	1
N	1	2	1	1	1
O	1	8	2	1	1
P	1	6	2	2	1
Q	-	1	2	1	2
R	1	5	2	1	1

Figure 22. Samples of data used in the recognition test: the word "brandy" written by eighteen subjects. Table shows the position of the correct result on the list of alternatives produced by each wholistic recognizer: "1" is the top alternative and "-" means the correct result was not produced. Results of the SEG recognizer are provided for comparison.

Recognition results are averaged over all writers. The results are provided as functions of the number of word alternatives produced by recognizers, which are taken into account. From a practical point of view only the recognition rate obtained for the best alternative is important. This is what the user experiences. However, considering other result alternatives is important

from research point of view. It may help assess the potential of individual recognizers and suggest possible improvements.

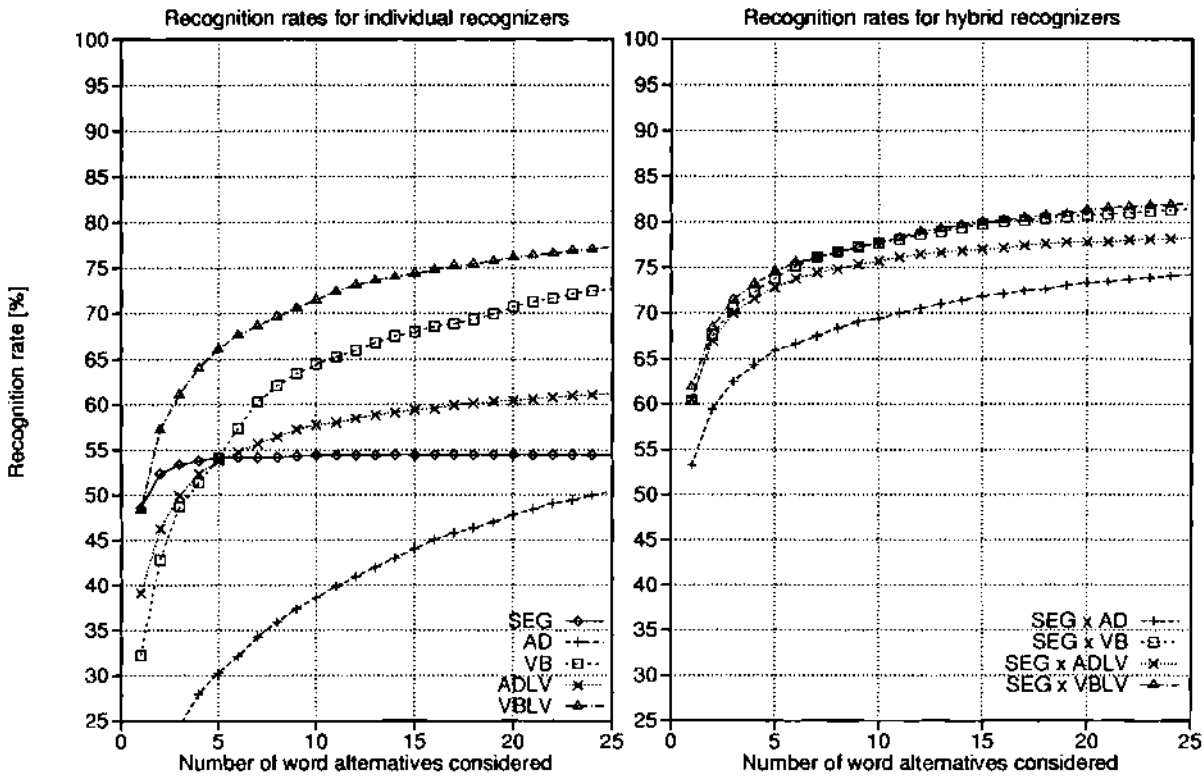


Figure 23. Recognition rates obtained for various recognizers. Up to 25 word alternatives taken into account.

When only the best word alternative is taken into consideration, the multiple interactive segmentation recognizer (SEG) performs best amongst the individual recognizers. Vertical bars recognizer with letter verification (VBLV) performs marginally worse. Other discussed recognizers (AD, ADLV and VB) perform significantly worse.

As the number of results alternatives taken into consideration grows, a significant difference can be observed between the performance of the segmentation based and wholistic recognizers. Wholistic recognizers are capable of finding the correct result when the SEG recognizer fails. Examples can be seen in Figure 22 - words A, D, H, I, J, K, L, Q. The recognition rate of SEG increases by about 5% for top five word alternatives (observe the results in Figure 22 - the SEG recognizer either gets the answer right, or not at all). The recognition rate does not increase further as the number of considered alternatives grows. On the other hand, recognition rates of the wholistic recognizers increase very significantly (again, observe the results in Figure 22 - the wholistic recognizers get the correct answer, but tend to rank it lower amongst other alternatives). The increase in recognition rate reaches 20% when top five word alternatives are taken into consideration. Recognizers VB and ADLV match the SEG, and recognizer VBLV outperforms SEG by around 12%. The increase in the recognition rates for the wholistic recognizers continues, albeit more slowly, as the number of word alternatives taken into consideration grows.

It can be observed that the ascender/descender approach (both AD and ADLV) provides results worse than the vertical bars approach (VB and VBLV, respectively). The ascender/descender recognizer does not use the information in the middle zone of the word. This results in worse results disambiguation. The results curve for the AD is significantly less steep than the curve for the VB (see Figure 23). When letter verification is introduced, both recognition curves become similar (ADLV and VBLV in Figure 23). This is where the influence of the other limitation can be observed - the accuracy of the zoning classification and detection of ascenders/descenders. Examples of failure in the zoning classification can be seen in Figure 22 - words D, F and K. Word E in Figure 22 is an example of incorrect detection of ascenders/descenders (the algorithm identified two consecutive ascenders in this word).

Hybrid recognizers provide a universal improvement. An improvement of 5 to 12 percent is observed for the top word alternative. As the number of word alternatives taken into account increases, the recognition rate of the hybrid recognizers is 3 to 10 percent better than that of VBLV. When compared to SEG, a 10 to 25 percent improvement in the recognition rate is observed, depending on the number of alternatives taken into account.

All the recognizers show disambiguation problems. The problems are strongest for the wholistic recognizers (AD and VB) and weakest for the multiple interactive segmentation recognizer (SEG). Introduction of letter verification improves disambiguation of alternatives (ADLB and VBLV). Hybrid recognizers disambiguate between alternatives better than wholistic recognizers, but still worse than individual recognizers.

Significant increase in the recognition rate for a larger number of alternatives indicates that the results of the wholistic and hybrid recognizers can be further improved.

9 Discussion

The results obtained so far are encouraging. The system was tested with a reasonably sized lexicon and a significant number of writers, including many unknown to it. The word shape templates used were constructed using general knowledge about writing. No training is performed. Fuzzy logic is used to deal with the handwriting variability.

The word shape recognition is limited by word shape calculation capability. It is extremely difficult to extract reliable zoning information for some handwriting styles. Such styles are currently not isolated and consequently treated like all other cases. A better approach might be to identify handwriting styles difficult to zone and lower the influence of the word shape recognizer [18]. Some individual handwriting variations strongly affect the word shape. The system currently lacks any mechanisms to adjust to such variations.

The ascender/descender recognizer uses information about the sequence and horizontal position of ascenders and descenders located within the word. The sequence of the ascenders/descenders is used to focus the recognition process on words of similar shape. This allows limitation of the recognition task to likely candidates only. However, when any errors in the ascender/descender detection occur, the correct solution is automatically excluded. The recognition rate of the ascender/descender recognizer is thus inherently limited by the accuracy of the word shape extraction. One idea to alleviate this limitation is to extract and process ascenders and descenders separately. It has been found that descenders are correctly identified more frequently [19]. The ascender/descender recognizer can be readily adapted to use information about upper

or lower half of the word. However, the disambiguation problems in such a case become even larger [19].

Horizontal position of the ascenders/descenders allows assessment of the degree of similarity between the data and particular database patterns. Fuzzy sets are used in this process. As the position of ascenders/descenders is expressed relatively in respect to the word length, a measure of the word length has to be taken into account in order to distinguish between words of similar proportions.

The ascender/descender recognizer does not take into account any information present in the middle zone. The use of this information can further improve the recognition rate of the recognizer.

The vertical bars recognizer uses information about downwards directed parts of the pen trajectory. Fuzzy sets are used to represent the position of the ends of vertical bars. All the detected vertical bars are used in the matching process. No zoning classification of vertical bars is necessary. The use of fuzzy sets also allows the special treatment of letters with different forms, like "f" or "z." No explicit classification of ascenders and descenders is required, hence the recognition performance is not limited by the accuracy of the location of the zoning information. Smaller vertical bars, present in the middle zone, are also used in the matching process. This presents an advantage over the ascender/descender recognizer.

Larger vertical bars have been observed to be relatively invariant within words. Smaller bars, which occur in the middle zone, tend to have higher variability (see Section 5). The matching algorithm introduced to cope with this problem is akin to dynamic programming [16]. Comparing patterns of vertical bars can also be viewed in the context of string matching. Ignoring superfluous and missing bars can be interpreted as dealing with symbol deletions and insertions. The matching algorithms presented in this paper (equations 11 and 14) are comparable to the string matching algorithm presented in [20]. The main difference between the matching algorithms developed by the authors and the classical dynamic programming solution is the presence of specific limitations when ignoring superfluous patterns and the fact that the search for the best solution is performed forward, not backwards.

Two versions of the algorithm are implemented. The original version of the matching algorithm allows superfluous vertical bars only in the data pattern. This version is significantly less computation intensive. The extended version of the matching algorithm allows superfluous bars both in the data and the database patterns. Where handwriting contains all the expected parts of letters, both matching algorithms provide similar results. The extended matching algorithm provides significantly better results only for particular styles of writing.

The presented wholistic recognizers use small sets of features. Ascenders/descenders and vertical bars are related to local vertical extrema used by other researchers (e.g. [4, 6, 9]). The use of vertical bars is also reported in an off-line recognition system [21]. A conscious decision was made to strictly limit the number of features used to give greater generality; the resulting system has greater difficulty choosing the correct word alternative as the best answer, however it is expected to reject the correct alternative due to errors in feature extraction less frequently.

The alternatives disambiguation problem is addressed by letter verification. The treatment of letter alternatives as higher level handwriting features is also reported in the results combination literature [22]. Letter alternatives themselves are obtained from the multiple interactive segmentation recognizer, which allows the reuse of partial results. The letter verification can be

compared to word ending postulation [2]. However, distinct differences are observed. The word ending postulation is designed to suggest the endings of words. It is capable of introducing new alternatives. The letter verification can only filter the results already obtained.

Recognition rates of wholistic recognizers are low. Figures of 13.9% and 32.2% for the best word alternative are obtained for recognizers AD and VB, respectively. The use of the letter verification improves the results. Recognition rates of 39.1% and 48.3% for the best word alternative are obtained for recognizers ADLV and VBLV, respectively. These are the results for eighteen writers and 4107 words lexicon. Farag reports recognition rate of 100% for ten words and one writer [7]. Bertille and Yacoubi report recognition rates between 23% and 63% for the recognition of cursive postal codes [23]. Brown and Ganapathy reach between 63% and 80.3% for three writers, 22 words and 47 words lexicon [9]. Earnest reports 18% for unique identification and 60% for the correct word appearing somewhere on the list of alternatives [6]. Tests were performed for five writers, 107 words and a lexicon of 10397 words. Simon reports 76% for top five alternatives, eight writers, 875 words and 25 words lexicon [8]. Overall, recognition rates obtained by the authors appear to be comparable with those cited by other researchers. It ought to be noted that wholistic recognizers are introduced here as auxiliary recognizers and are not meant to be used stand-alone. However, given better methods of results disambiguation, wholistic recognizers have the potential of outperforming the multiple interactive segmentation recognizer (SEG). The recognition rate of VBLV already nearly matches that of the SEG (48.5%). Wholistic recognizers reject correct solutions less frequently than the multiple interactive segmentation recognizer.

The use of wholistic recognizers together with the multiple interactive segmentation recognizer in a hybrid system provides a significant improvement of the recognition results. Recognition rates of 53.3, 60.4, 60.5 and 61.9 percent are obtained for the best word alternative for recognizers SEG x AD, SEG x VB, SEG x ADLV and SEG x VBLV, respectively. This is a clear improvement over 48.5% provided by the SEG. The resulting hybrid recognizers inherit some of the disambiguation problems from the wholistic recognizers. The difference in recognition rates for top word alternative and larger numbers of alternatives is significant, reaching up to 7.1% for top two alternatives only (SEG x VB). This indicates that the hybrid recognition system is frequently capable of recognizing the word, but it still has difficulties choosing the correct word alternative as the best answer.

The results combination process is currently very simple. Other approaches have been attempted, including the characterisation of recognizers and combination based on the assessment of handwriting characteristics. This has however been found to provide no significant improvement so far [24].

Recognition performance can be improved by providing better disambiguation capabilities. This can be achieved through using a recognition toolbox [18]. Work is currently in progress to provide a number of feature tools that would be used to disambiguate between alternatives produced by the recognizers. The objective is a system of small feature tools that push the correct word alternative to the top of the list. The feature tools are to be selected dynamically, so that they reflect a particular handwriting style. Average recognition rates of around 80% for the first word alternative are within reach for the hybrid recognizer, as can be seen in Figure 23.

References

1. R.K. Powalka, N. Sherkat, L.J. Evett, R.J. Whitrow. Multiple word segmentation with interactive look-up for cursive script recognition. *Second Int. Conf. on Document Analysis and Recognition* ICDAR'93, Tsukuba Science City, Japan, 196-199 (Oct. 1993).

2. R.K. Powalka, N. Sherkat, L.J. Evett, R.J. Whitrow. Dynamic cursive script recognition. A hybrid approach. *Advances in Handwriting and Drawing: A multidisciplinary approach*, C. Faure, P. Keuss, G. Lorette, A. Vinter (Eds.), 137-154, Europia, ISBN 2-909285-02-2 (1994).

3. T.K. Ho, J.J. Hull, S.N. Srihari. A regression approach to combination of decisions by multiple character recognition algorithms. *Machine Vision Applications in Character Recognition and Industrial Inspection*, Proc. SPIE 1661, 137-145 (1992).

4. B. Plessis, A. Sicsu, L. Heutte, E. Menu, E. Lecolinet, O. Debon, J.-V. Moreau. A multi-classifier combination strategy for the recognition of handwritten cursive words. *Second Int. Conf. on Document Analysis and Recognition* ICDAR'93, Tsukuba Science City, Japan, 642-645 (Oct. 1993).

5. L. Xu, A. Krzyzak, C.Y. Suen. Methods for combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, **23**, 3, 418-435 (1992).

6. L.D. Earnest. Machine recognition of cursive writing. Information Processing, Proc. of IFTP Congress 62, C.M. Popplewell (ed.), Munich, Germany (Aug.-Sept. 1962).

7. R.F.H. Farag. Word-level recognition of cursive script. *IEEE Trans. on Computers*, **C-28**, 2, 172-175 (1979).

8. J.-C. Simon. Off-line cursive word recognition. *Proc. of the IEEE*, **80**, 7, 1150-1161 (1992).

9. M.K. Brown, S. Ganapathy. Cursive script recognition. *Fifth Int. Conf. on Cybern. and Society*, Boston, 47-51 (1980).

10. W. Guerfali, R. Plamondon. Normalizing and restoring on-line handwriting. *Pattern Recognition*, **26**, 3, 419-431 (1993).

11. L.A. Zadeh. Fuzzy sets. *Inform. Contr.*, **8**, 574-591 (1965).

12. R. Mitton. The machine usable form of the Oxford Advanced Learners Dictionary. *Oxford Text Archives* (1986).

13. Stig Johansson. The LOB corpus of British English texts: presentation and comments. *ALLC Journal*, **1**, 1 (1980).

14. F.G. Keenan. Large vocabulary syntactic analysis for text recognition. *PhD Thesis*, The Nottingham Trent University (1993).

15. J.S. Lipscomb. A trainable gesture recognizer. *Pattern Recognition*, **24**, 9, 895-907 (1991).

16. C.C. Tappert. Cursive script recognition by elastic matching. *IBM J. Res. Develop.*, **26**, 6, 765-771 (1982).

17. Sh.A. Guberman, V.V. Rozentsveig. Algorithms for reading of handwritten texts (in Russian). *Avtomatika i Telemekhanika*, 5, 122-129 (1976).

18. R.K. Powalka, N. Sherkat, R.J. Whitrow. A toolbox for recognition of varied handwritten script. *First European Conf. on Postal Technology* JET POSTE 93, Nantes, France, 140-147 (June 1993).

19. R.K. Powalka. An algorithm toolbox for on-line cursive script recognition. *PhD Thesis*, The Nottingham Trent University (1995).

20. P.A.V. Hall, G.R. Dowling. Approximate string matching. *ACM Computing Surveys*, **12**, 4, 381-402 (1980).

21. J.J. Hull. Hypothesis generation in a computational model for visual word recognition. *IEEE Expert*, **1**, 3, 63-70 (1986).

22. J. Franke. Statistical combination of multiple classifiers adapted on image parts. *First European Conf. on Postal Technology* JET POSTE 93, Nantes, France, 566-572 (June 1993).

23. J.-M. Bertille, A.E. Yacoubi. Global cursive postal code recognition using Hidden Markov Models. *First European Conf. on Postal Technology* JET POSTE 93, Nantes, France, 129-138 (June 1993).

24. R.K. Powalka, N. Sherkat, R.J. Whitrow. Recognizer characterisation for combining handwriting recognition results at word level. *Third Int. Conf. on Document Analysis and Recognition* ICDAR'95, Montreal, Canada. 68-73 (Aug. 1995).

Acknowledgments

The paper has been accepted for publication in the Pattern Recognition journal.

The authors would like to gratefully acknowledge the permission of the Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington OX5 1GB, UK, to publish this paper in the Working Papers Series.