

Real-Time Constraint-Free Character Recognition System

Fathallah Nouboud and Réjean Plamondon, Département de Génie Electrique
Ecole Polytechnique de Montréal, C.P. 6079, Succursale "A",
Montréal QC, H3C 3A7

Abstract

In this paper we describe an on-line character recognition system operating in real time (0.07 seconds/character) and with no constraints on the handprinting. The preprocessing step makes it possible to be free of any constraints on either the size of the characters or the speed of the writing. The comparison is performed by a string-comparison processor. Following completion of a learning step, the system can recognize any set of characters defined by the user. The experimental tests show that the system is effective in user-dependent applications. The recognition rate is 96%.

Keywords: On-line, handwriting, character recognition, chain code, dedicated processor, learning, user-dependent.

1. Introduction

A number of problems must be dealt with when designing a character recognition system. Among them, there is the problem of variability. The same character may be written in many different ways by different users or even by the same user, and it is almost impossible to predict all the possible variations of a given character. Many character recognition systems therefore impose constraints on the writing, and these constraints may make users uncomfortable with the process. Other problems are, for example, the slow speed of processing, the use of boxes to separate characters, etc. [1,2,3,8,10].

In the system described here, we use a personal data base for each user, thus the writer does not have to respect any constraint and may use the system with his own natural handprinting style. The use of a specialized

processor for the comparison makes it possible to operate in real time and with a high degree of accuracy.

2. Preprocessing

Characters are digitized with the help of a graphic tablet (PENPAD300, PENCEPT Inc.). The active surface of the tablet is 11" x 11" (27.94 cm x 27.94 cm), the resolution is 1000 points per inch (394 points/cm) and the sampling frequency is 100 points per second.

Segmentation between characters is based on temporal information. The character is considered to be completed when a given time (about 300 ms) has elapsed since the last pen contact with the tablet surface.

A number of preprocessing tasks must be applied to the raw digitized data. These are designed to reduce the amount of information, to eliminate certain imperfections of the trace and to make the process independent of the size of the characters and the speed of the writing. Four preprocessing steps are used in this system.

2.1 Smoothing

The smoothing operation eliminates imperfections caused by the tablet, trembles in the writing, etc. A mobile average filter replaces a point with the average over its neighbours.

The transformation is performed according to the following equation:

$$X_i = (X_{i-3} + 3X_{i-2} + 6X_{i-1} + 7X_i + 6X_{i+1} + 3X_{i+2} + X_{i+3})/27 \quad (1)$$

This equation has been obtained after the convolution:

$$(111 * 111) * 111 = 1367631 \quad (2)$$

The cut-off frequency of this filter is about 8.75 Hz.

2.2 Spatial Filtering

The main objectives of this filtering are to reduce the number of points and to eliminate duplicate points. This is done by forcing a minimum distance between consecutive points. This operation destroys the dynamic information, however this information is not important in on-line character recognition (as compared to signature verification) as long as the time sequence of points is preserved.

The distance between two consecutive points far exceeds the minimum distance when the writing is executed quickly. Therefore, interpolation makes it possible to have equidistant points.

$D(i)$ being the accumulated distance between the last point retained and point i , and δ being a threshold, the filter interpolates the points defined by:

$$X = X_{i-1} + [(\delta - D(i-1) + n\delta) \cdot \cos\phi_i] \quad (3)$$

$$Y = Y_{i-1} + [(\delta - D(i-1) + n\delta) \cdot \sin\phi_i] \quad (4)$$

where $0 < n < d/\delta$, d being the distance between points $i-1$ and i
and $\phi_i = \tan^{-1} ((Y_i - Y_{i-1}) / (X_i - X_{i-1}))$.

In order to save the information about the presence of cusps and corners, more points are retained in regions of greater curvature, i.e. when the following condition is true:

$$\min(360 - |\phi_i - \phi_o|, |\phi_i - \phi_o|) \geq \phi \quad (5)$$

where ϕ_o is the tangent angle for the last point retained

ϕ_i is the tangent angle for point i

ϕ is a threshold.

2.3 Dot Reduction

In this step, the information relating to a dot is reduced to a single pair of coordinates. This is very important in recognition because

there are many ways in which one person may write a point. The writer may simply touch the surface of the tablet, or write small scribbled dots. The spatial filter is then sufficient to obtain a single point. The user may sometimes, however, write larger dots or even small circles. In these cases, spatial filtering is not enough. We have used a method similar to the one proposed by Tappert [9]. The detection of dots is based on three criteria. First, the dimension is smaller than that of the other strokes. Second, there is a high density of points (ratio of the number of points to the dimension). Third, the average change in direction between successive points is higher (scrabbled dots). This process reduces most dots to single points.

2.4 Normalization

In this step, the character is mapped into a box of fixed dimensions. Thus, feature extraction is independent of the size of the character. The user may write very small or very large characters, and the recognition result will be the same.

3. Chain Code Extraction

The character is described by direction codes and position codes. These codes are extracted from the various components of the character. A component is a stroke between two consecutive pen-lifts [5].

A component is represented by a sequence of consecutive points (X_i, Y_i) , $i=1,2,\dots$. The vector spanning the points (X_i, Y_i) and (X_{i+1}, Y_{i+1}) is called an element. Its slope is given by:

$$S_i = \tan^{-1} \frac{Y_{i+1} - Y_i}{X_{i+1} - X_i} \quad (6)$$

This angle is converted to a code from A to L as shown in Figure 1.

The position of the element inside the rectangle surrounding the character is coded from 0 to 9. The normalized distance of the element to the origin of the rectangle (down-left corner) ranges from 0 to 1. This value is converted to a code from 0 to 9. Thus an

element in the down-left corner has code 0 and an element in the upper-right corner has code 9.

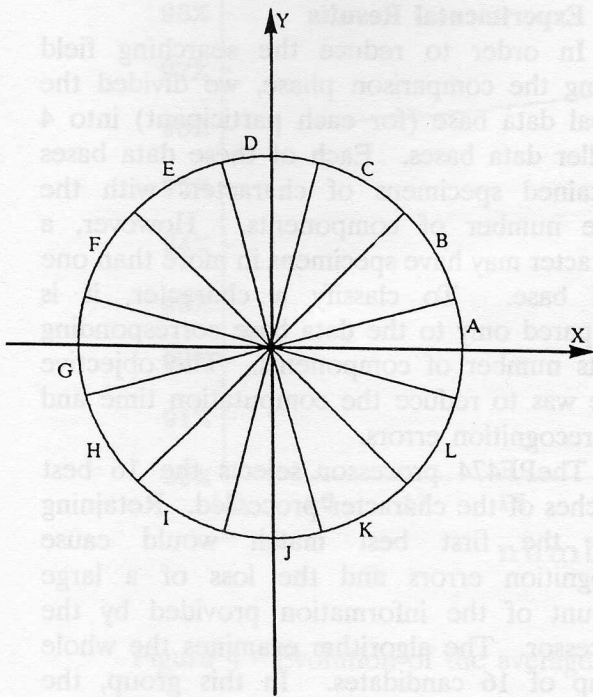


Figure 1 - The twelve direction codes.

Now the component is described by a chain code and the character is represented by the concatenation of its component chains codes (Figure 2).

D

5J4J4J3J3J2J2H

6BL7L7L7L8L8L8K8K7J7I6H5H5H4H3G2G2G1G1G0G

Figure 2 - Character D and its chains codes.

4. Classification

The comparison task is performed by the PF474 board manufactured by Proximity Technology Inc. [6,7]. This processor measures string similarity.

The PF474 processor can compare a string of characters with a long list of strings and

select the best matches. The processor compares the query string with each string in the list and produces a score. Then it performs a ranking over all the scores. The result obtained is a ranked list of the 16 best matches.

The strings in our application contain symbols from the set:

$$CS = \{A, B, C, D, E, F, G, H, I, J, K, L, P, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

The letters from A to L are direction codes, P is the code representing a dot (see I.3 above) and numerals from 0 to 9 are position codes.

Following is a brief description of the comparison algorithm.

Let X be a string. Let us define:

- $|X|$ as the length of X
- $X[i]$ as the i^{th} character ($0 \leq i \leq |X| - 1$)
- X' as the transpose (left-right flip) of X
- $O(c, X)$ as the number of times the character c appears in X
- $R(X, n)$ as the string obtained from X by removing its n first elements.

The comparison value between strings S and T is:

$$\text{comp}(S, T) = \frac{2C_1(S, T) + 2C_2(S, T)}{(|S|^2 + |S|) + (|T|^2 + |T|)} \quad (7)$$

with:

$$C_1(S, T) = \sum_{n=0}^{\infty} \sum_{c \in CS} \min(O(c, R(S, n)), O(c, R(T, n))) \quad (8)$$

$$C_2(S, T) = \sum_{n=0}^{\infty} \sum_{c \in CS} \min(O(c, R(S', n)), O(c, R(T', n))) \quad (9)$$

An intuitive version of the comparison value is given by:

$$\text{comp}(S, T) = \frac{2(S \text{ comp to } T) + 2(S' \text{ comp to } T')}{(S \text{ comp to } S) + (T \text{ comp to } T)} \quad (10)$$

More details and examples are given in [7].

In our application, the strings lengths are less than 60. As a result, the PF474 processor can perform up to 10,000 comparisons and rankings per second. This enabled our system to operate in real time: about 0.07 seconds per character.

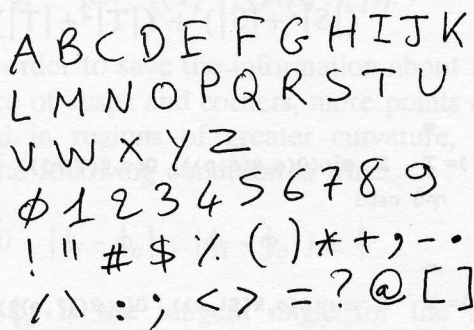
There is however a problem with the PF474 algorithm. During the comparison between two strings, two elements are either identical or different without any measure of their difference. For example position code 0 is considered by the algorithm to have the same difference with code 1 and code 9. Thus some small local variations may be amplified and this may cause recognition errors.

5. Experimental Tests

5.1 Learning

In the learning step, the user defines the set of characters to be recognized by the system. There is no limit to the type or the number of the characters in this set.

The set used in our experiment is shown in Figure 3. The participant writes the characters in his own natural writing style, without any constraints.



A B C D E F G H I J K
L M N O P Q R S T U
V W X Y Z
0 1 2 3 4 5 6 7 8 9
! " # \$ % & ' () * + , - .
/ \ : ; < > = ? @ []

Figure 3 - Set of characters.

In the learning step, the user writes 16 specimens of each character. This operation offers two major advantages. First, the user may define any set of characters suited to his particular application. Second, the system

learns the handprinting style of the user and adapts to it instead of imposing constraints to the user.

5.2 Experimental Results

In order to reduce the searching field during the comparison phase, we divided the global data base (for each participant) into 4 smaller data bases. Each of these data bases contained specimens of characters with the same number of components. However, a character may have specimens in more than one data base. To classify a character, it is compared only to the data base corresponding to its number of components. The objective here was to reduce the computation time and the recognition errors.

The PF474 processor selects the 16 best matches of the character processed. Retaining only the first best match would cause recognition errors and the loss of a large amount of the information provided by the processor. The algorithm examines the whole group of 16 candidates. In this group, the same character may appear several times. Therefore, the system selects the candidate with the highest sum (over its scores) as the classification result. However, if the highest sum is lower than a threshold, the character is rejected.

In this experiment, we asked 15 participants to write the set of characters (Figure 3) 30 times. The total number of characters was 26,550.

The data base of prototypes consists of a subset of the total set of characters written by the participant. During classification, the PF474 processor provides the 16 best matches for the character processed. In order to enhance the chances of retaining several good prototypes among the sixteen, we set the number of prototypes to a value higher than or equal to 16. The best value was determined by calculating the average recognition rate for a number of specimens varying from 16 to 24. The results obtained are shown in Figure 4.

We expected to observe a significant improvement in the recognition rate as the

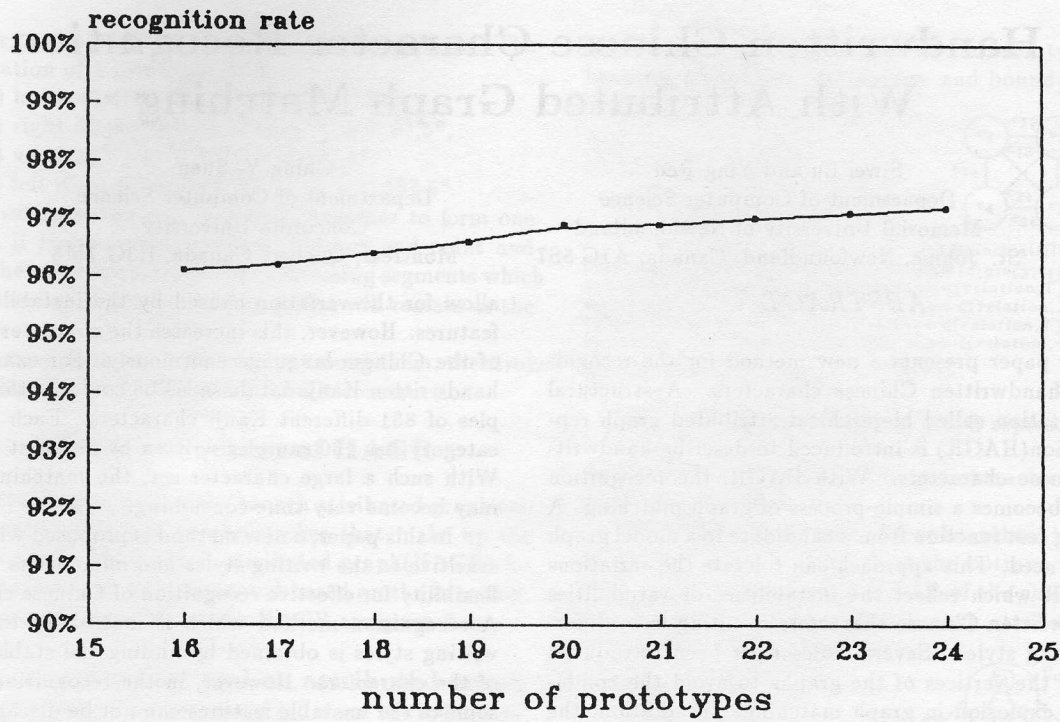


Figure 4 - Evolution of the average recognition rate with the number of prototypes.

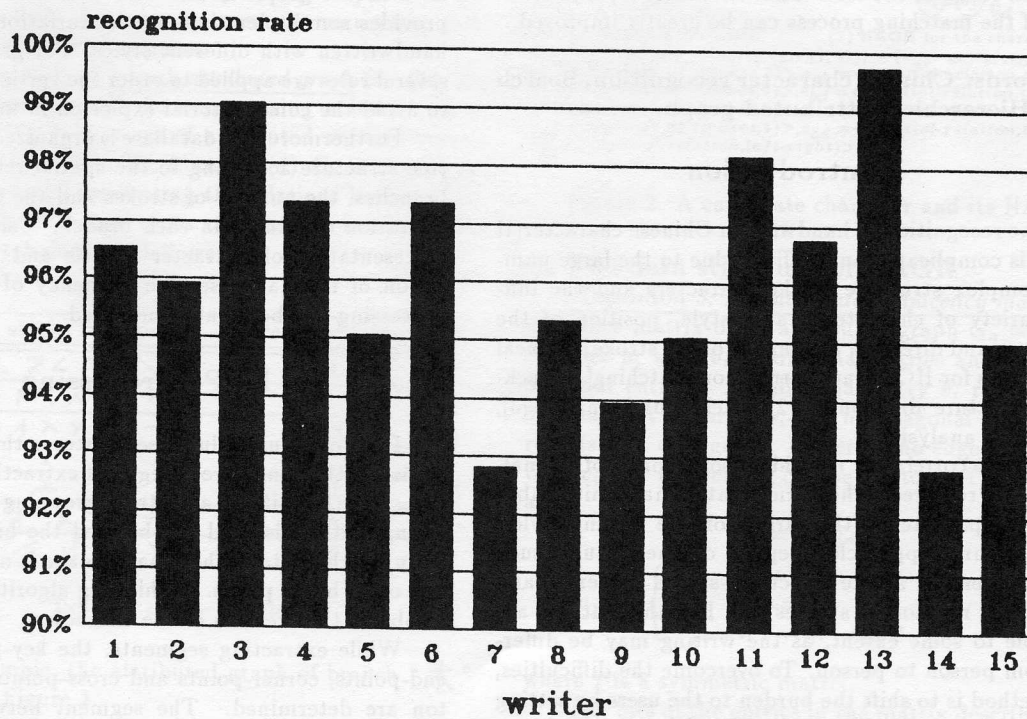


Figure 5 - Individual recognition rates.