A Fuzzy-Syntactic Approach to Allograph Modeling for Cursive Script Recognition*

Marc PARIZEAU† and Réjean PLAMONDON‡

printed July 8, 1997

Abstract

On-line cursive script recognition has recently become the focus of renewed interest because of the development of notebook computers that incorporate a digitizing tablet over a high-resolution graphics display. This new hardware, incorporating the electronic penpad concept, aims at the elimination of the keyboard and mouse for many interactive applications. However, although recognition algorithms for isolated characters are widely used for this purpose, cursive script recognition is just starting to be proposed in commercial systems and still remains a stumbling block.

The object of this paper is to present an original method for creating allograph models and recognizing them within cursive handwriting. This method concentrates on the morphological aspect of cursive script recognition. It uses fuzzy-shape grammars to define the morphological characteristics of conventional allographs which can be viewed as basic (a priori) knowledge for developing a multi-writer recognition system. The system developed uses no linguistic knowledge to output character sequences that possibly correspond to an unknown cursive word input.

The recognition method is tested using multi-writer cursive random letter sequences. For a test dataset containing a handwritten cursive text 600 characters in length written by ten different writers, average character recognition rates of 84.4% to 91.6% are obtained, depending on whether only the first (best) character sequence output of the system is considered or if the best of the top ten is accepted. These results are achieved without any writer-dependent tuning. The same dataset is used to evaluate the performance of human readers. An average recognition rate of 96.0% was reached, using ten different readers, presented with randomized samples of each writer. The worst reader-writer performance was 78.3%. Moreover, results show that system performances are highly correlated with human performances.

*This work was published in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 7, pp. 702-712, 1995. It was supported in part by NSERC Canada under grant OGP000915, and in part by FCAR Québec under grant CRP2667. Marc Parizeau received scholarships from NSERC and FCAR.

†Marc Parizeau was with the Département de génie électrique et de génie informatique, École Polytechnique de Montréal. He is now with the Département de génie électrique et de génie informatique, Université Laval, St-Foy (PQ), Canada, G1K 7P4. E-mail: *parizeau@gel.ulaval.ca*

‡Réjean Plamondon is with the Département de génie électrique et de génie informatique, École Polytechnique de Montréal, C.P. 6079 – Succ. "A", Montréal, Canada, H3C 3A7.

I Introduction

Computer recognition of handwriting in general, and of cursive script in particular, has been a research topic for more than thirty years [1, 2, 3]. In the beginning, cursive script recognition was mostly seen as a way to get some insight into what researchers at the time considered to be a tougher problem: speech recognition. This lasted through the 1960s and ebbed in the 1970s. Then, in the 1980s, new accurate digitizing tablets [4] and powerful microcomputers became available, which created a renewed interest in handwriting recognition research and in pen-based man-machine interfaces [5, 6, 7]. Now, in the 1990s, this interest is booming with the introduction of notebook computers based on the electronic ink concept.

Electronic ink is the simulation of the trace of a pentip on a high resolution graphics display. The position of the pentip on the display is sampled at constant frequency by the computer which lights up the trace of its motion in real time. By overlaying a digitizing tablet and a flat panel display, very compact computers, without keyboard or mouse, are now available. But to take full advantage of this new hardware, powerful handwriting recognition algorithms are needed, particularly for cursive script.

We refer to cursive script as natural handwriting, that is, a style of handwriting that a human uses naturally when no constraints are imposed, except, perhaps, minimum legibility. Thus, for cursive script, successive letters in a word are sometimes discrete, in the sense that they start at the beginning of a handwriting *component*[1] and end at the end of a compo-

[1]The term *component* [8] is used to designate the portion of the written trace between a pendown and a penlift (while the

nent (not necessarily the same one), and are sometimes linked by a common component.

There are three main problems that must be tackled by on-line cursive script recognition systems. First, cursive letters within cursive words cannot be segmented unambiguously before recognition [2]. Thus, the recognition processes found in the literature always integrate some kind of segmentation scheme that ultimately leads to several concurrent letter hypotheses. The second problem concerns the ways of writing each letter of the alphabet, or the models. These models are called *allographs*. For example, lower- and upper-case letters are usually different allographs. But, even for a particular lower-case letter, there can be several different allographs used by different individuals or even by the same writer. Indeed, in cursive script, a person can trace different allographs of the same letter, even in the same word, depending on the context of neighboring letters. And, of course, these allographs can be more or less degraded depending on the writer's skill and assiduity. Finally, the third problem revolves around the use of linguistic context. Since several concurrent hypotheses can be generated by the segmentation/recognition processes, linguistic context is often required to disambiguate otherwise perfectly compatible and coherent combinations of hypotheses. But to use linguistic context, especially at higher levels, is a difficult problem and for this reason current recognition systems are usually restricted to a limited vocabulary. The methods developed in this paper concentrate on the first two problems in an effort to avoid reliance on linguistic constraints.

There are basically three types of knowledge available for solving these problems: morphological, pragmatic and linguistic. *Morphological* knowledge refers to everything that is known about the shape of allographs. *Pragmatic*[2] knowledge refers to what is known about how to spatially arrange allographs into words, phrases and paragraphs. *Linguistic* knowledge concerns the language that is used to convey the message represented in handwriting (i.e. English, French, etc.). This last type is divided into three categories[3]: lexical, grammatical and semantic.

Early work by Mermelstein and Eden [9] used points of zero Y-velocity to decompose handwriting into segments ("strokes"). For their system, morphological knowledge is built statistically, from letter prototypes, on parameter sets associated with "strokes". Pragmatic knowledge is confined to processing sequential (in terms of components) letters, which prohibits i-dots and t-crossings which are often traced at the end (last components) of the word. Linguistic knowledge is limited to using a short dictionary (59 words). More recent work by Tappert [10] has circumvented the segmentation problem by using dynamic programming to evaluate all possible presegmentation points. But here again morphological knowledge takes the form of letter prototypes that are compared with elastic matching. Pragmatic knowledge is also restricted to sequence information. However, preprocessing algorithms are proposed for "delayed strokes", like i-dots and t-crossings. Linguistic knowledge is limited to bigram search and analysis. Another system, by Higgins and Whitrow [11], segments handwriting at different "turning points" corresponding to local extrema of position signals. Morphological knowledge for this system consists of a hierarchical description of prototypes, while pragmatic knowledge is again restricted to sequence, and linguistic knowledge is confined to a combination of quadgrams and limited vocabulary. Very recent work is now focusing on self-organizing networks [12, 13]. These systems extract feature vectors from motoric strokes and build morphological knowledge statistically (like the early work of Mermelstein), but using modern classifiers like, for example, Kohonen's Self-Organizing Maps (SOM). Recognition is still constrained by sequence information, although delayed strokes are reordered by preprocessing algorithms, and is intimately linked to a given vocabulary.

The work described in this paper [14] is part of a larger project that aims toward the proposal of an intelligent electronic penpad [15]. It deals with an aspect of cursive script recognition that has not been addressed in the field: the creation of writer-independent allograph models and their recognition within cursive script. The idea behind these *intrinsic* (writer-independent) models is to give basic morphological knowledge of cursive handwriting that can enable recognition of well-written script when linguistic knowledge is sparse or even unavailable.

Basic morphological knowledge needs to be writer-independent for building true multi-writer recognition systems. In that sense, letter prototypes are not the best foundations on which basic morphological knowl-

pen is in contact with the paper).

[2]This definition of pragmatic knowledge should not be confused with the definition used by linguists, who refer to knowledge induced by common sense.

[3]Linguists would probably add pragmatic as a fourth category.

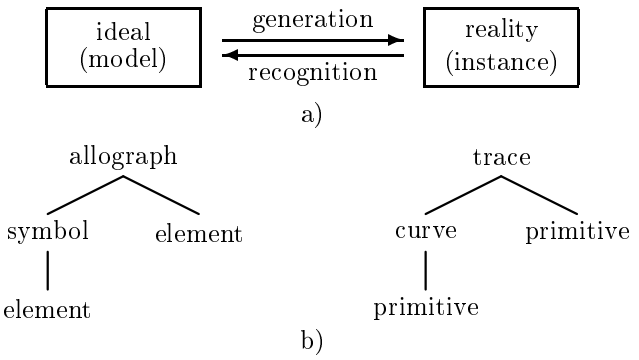


Figure 1: a) *Motor-perceptual interaction,* b) *Hierarchical nomenclatures.*

edge should be built. This is not to say, however, that they are useless, but rather that they form a complementary source of knowledge (most useful when a particular writer deviates from standard models) biased toward characteristics that are writer-dependent. Basic morphological knowledge should be based on what is characteristic of allographs, not what is characteristic of writers. Moreover, intrinsic allograph models exist since they are taught to schoolchildren. Several different models can be found in different schools or regions [16], but their number is still finite.

I.1 Organization of the Paper

The remainder of this paper is subdivided as follows. In section II, preliminary definitions and a system overview are provided. Sections III to V give details of various aspects of the system. Section III presents the primitive extraction phase. Section IV describes the general allograph modeling strategy, together with the allograph segmentation process. Character sequence construction from adjacent segmented allographs is then described in section V. Experimental results are presented and discussed in section VI, and concluding remarks are made in section VII.

II System Overview

The system that has been developed enables the creation of sophisticated writer-independent allograph models and their recognition within cursive handwriting. The system design philosophy is illustrated in Figure 1. The approach relies on the concept that handwriting is generated from an ideal representation of the letters and that the recognition of specific instances is performed with reference to these ideal models. In other words, our fundamental design hypothesis is that neuronal mechanisms of handwriting generation and perception interact and share a "common memory", where some basic motor-perceptual interactions are represented. The allograph models, or simply *allographs*, should thus be seen as expressions of ideal concepts. Handwriting generation is a process that transforms allographs into *traces* which correspond to instances produced by a writer at a specific time and place. Handwriting recognition involves the reverse transformation.

In general, allographs are represented by a set of *symbols* which can be constructed from other (simpler) symbols or *elements*. An allograph is thus the root of a hierarchy of symbols where leaves are elements. Similarly, a trace is represented by a set of *curves* which can be assembled from other (simpler) curves or *primitives*. A trace is thus the root of a hierarchy of curves where leaves are primitives.

Given this nomenclature, the system's block diagram shown in Figure 2 can now be described. The input of the system is a set of *handwriting components* sampled by a digitizing tablet. A (handwriting) component consists of a pair of vectors $C(t) = [x(t), y(t)]$, $t \in \{0, 1, \cdots, i, \cdots, n\}$ that define the tablet's pen position along the two orthogonal axes of its writing surface, sampled at fixed time intervals. $C(0)$ is thus the position of the pen at time t_0 (i.e. when the pen is put into contact with the writing surface), $C(i)$ is the position at time t_i (after the i^{th} sampling period) and $C(n)$ is the last sampled point before the pen is lifted up from the surface. A cursive word can be composed of one or more handwriting components.

The output of the system is a set of *character sequences* that can possibly correspond to the input components, given the system's three knowledge sources: a handwriting model, a set of allograph models and adjacency constraints. Each of these knowledge sources is used by one of three system processes.

The *handwriting model* [17] refers to the representation of handwriting used for decomposing components into primitives. It is part of the morphological knowledge of the system. That model defines characteristic points of components, each of which is associated with a *cursive primitive*. A (cursive) primitive consists of attributes that represent the portion of a component around one of its characteristic points. The corresponding *primitive extraction* process is described in Section III.

The *allograph models* define the main morpholog-

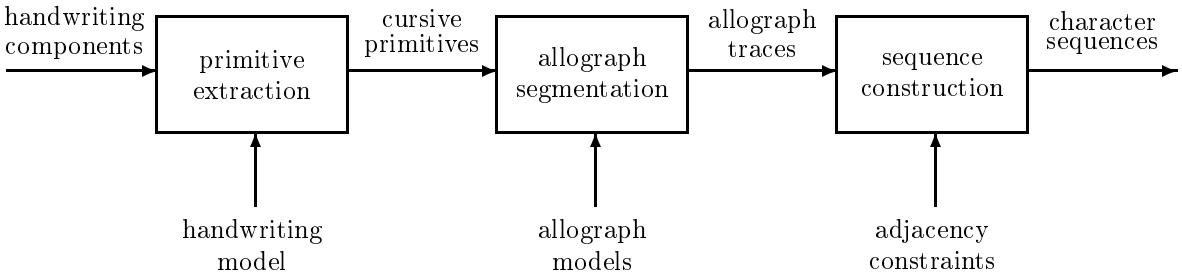


Figure 2: *Block diagram of system.*

ical knowledge of the system. They are represented by fuzzy-shape grammars [18] which are context-free attributed grammars with applicability conditions expressed in fuzzy-logic [19, 20]. They are used by the *allograph segmentation* process which generates *allograph traces* consisting in subsets of primitives that respect the conditions of corresponding allograph models. This process, which corresponds to a parser for fuzzy-shape grammars, is described in Section IV together with the general modeling strategy for building allographs.

The *adjacency constraints* [21] define the pragmatic knowledge of the system. They enable the construction of an allograph trace segmentation graph which describes adjacent traces. A path in this graph represents a candidate *character sequence* output that constitutes a morphologically and pragmatically coherent interpretation of the input components. The corresponding *sequence construction* process is described in section V.

At this stage, the recognition system does not use linguistic knowledge and can be viewed as a morphological expert in the context of a multi-expert system.

III Primitive Extraction

The primitive extraction process is based on a representation of handwriting that preserves only the useful information for recognition purposes. We call this information *morphologically pertinent* because it is characteristic of the shape of the symbols that must be recognized.

III.1 Handwriting Model

Many handwriting models have been proposed for analyzing or generating pieces of handwriting [8, 22]. Most of these are concerned mainly with temporal simulation of handwriting, however, and not directly with recognition (although some could be used for this purpose). The model that we describe here is an operational model in the sense that, while incorporating some of the features of a general handwriting model [23, 24, 25], it is aimed directly at the recognition problem and thus at making some pragmatic simplifications.

The proposed model defines a handwriting component as a sequence of characteristic points linked together by segments of *constant curvature*. Characteristic points are morphologically pertinent points of the component. The underlying hypothesis is that changes in curvature are only pertinent for recognition when they coincide with characteristic points. Otherwise, they are considered as an artefact, either of the handwriting process itself or of the data acquisition process.

The chosen characteristic points are local extrema of vectors $x(t)$ and $y(t)$, and local inflexion points of $C(t)$. Two hypotheses are made with this choice of characteristic points: first, it is assumed that the baseline of the word is approximately oriented along the X axis of the writing plane, and second, that the inclinations of the ascenders and descenders in the letters are approximately oriented along the Y axis of the writing plane. If these hypotheses are not met, it is assumed that preprocessing techniques [26] can detect and correct these orientations.

Under these hypotheses, the chosen characteristic points correspond to an approximation of the segmentation scheme proposed by Plamondon [23, 24], where *components* are made up of *strings*, that is, portions of components between two angular discontinuities; each string is made up of a combination of curvilinear and angular *strokes*, that is, displacements resulting from an impulse applied as input to curvilinear and angular velocity generators; and strokes are characterized by log-normal velocity profiles. The strings are always delimited by local extrema of either $x(t)$

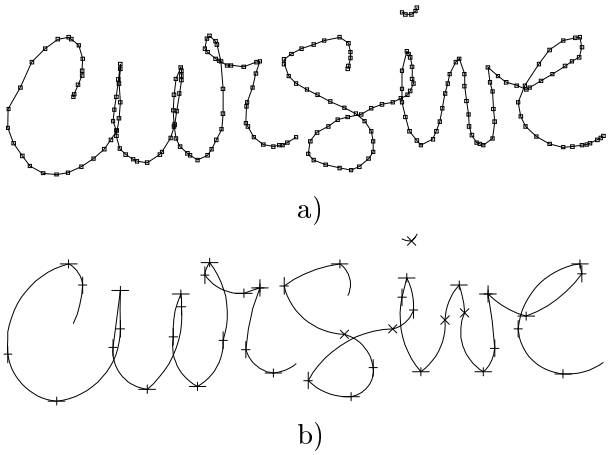


Figure 3: *Example of the word* cursive. a) *original components with their digitized points,* b) *reconstructed components with their characteristic points.*

or $y(t)$. The other local extrema and local inflexion points are rough estimates of stroke boundaries which are, in fact, hidden in the signal due to a superimposition phenomenon.

Figure 3 shows an example of the word *cursive* with its digitized and characteristic points. Figure 3a gives the original handwriting components with the digitized points linked by line segments, and Figure 3b shows the model-based reconstruction of the components, where the characteristic points are linked by circular arcs (local extrema are marked with + and local inflexion points with $\times$). This example shows that the proposed handwriting model keeps the major part of the morphologically pertinent information, while effecting efficient data compression. In fact, in an experiment with human readers [17], no significant increase in recognition errors was observed between the original and reconstructed handwriting samples.

III.2 Cursive Primitive

The cursive primitive stems directly from this handwriting model. A primitive is associated with each characteristic point and is defined by a set of attributes. Two types of attributes are used: attachment points for testing the syntactic arrangements of the primitives and properties for testing their semantic[4] coherence.

Three attachment points are defined: a *starting* point, a *characteristic* point and an *ending* point. Because only characteristic points of the handwriting model are considered morphologically pertinent, the starting and ending points are chosen as the previous and next characteristic points respectively.

This choice of attachment points implies that two successive primitives of the same component will always share two attachment points and a common segment of the cursive trace. This is consistent with the proposition that all non-characteristic points are only a consequence of the relation that unites characteristic points. It is also consistent with the Plamondon handwriting model where strokes usually overlap [25].

Seven properties are considered: a measure of angular *discontinuity* of the cursive trace at the characteristic point, a measure of *tilt* at each of the three attachment points, a measure of *curveness*[5] for each of the two segments between the three attachment points and a unique *index* number that identifies the primitive within the sequence of primitives.

A cursive primitive can thus correspond to any portion of a component that spans three characteristic points. Details of the algorithms for extracting the characteristic points, or for computing the seven properties, can be found in [17].

IV Allograph Segmentation

The allograph segmentation process consists in finding traces (i.e. subsets of cursive primitives) that respect the conditions of the allograph models. These models define the morphological characteristics of the allograph using hand-generated fuzzy-shape grammars coded in a specially designed programming language named $\mathcal{HAD}$ ($\mathcal{H}$ierarchical $\mathcal{A}$llograph $\mathcal{D}$escription) [14]. $\mathcal{HAD}$ was inspired from the work by Davis and Henderson [27].

IV.1 Modeling Strategy

The strategy designed for modeling the allographs can be divided into three phases. The first phase consists in creating several classes of handwriting elements. The basic handwriting element represents a general piece of a component that spans three characteristic points. It is a defined by three attachment points and seven properties (just like its instance counterpart:

[4]The reader should note that although the meanings of the words *syntactic* and *semantic* are the same as in the introduction, the context has changed. In the introduction we were referring to the syntax (grammar) and semantics of natural languages like English. From now on, these words should be interpreted in the context of syntactic pattern recognition and attributed languages.

[5]The term *curveness* is used as a heuristic curvature.

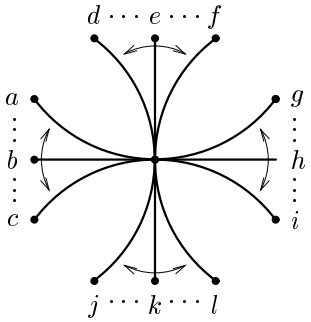


Figure 4: *Illustration of "continuous" primitives.*

the cursive primitive) and can have many shapes. The idea is to build classes of elements that represent more specific elementary shapes. This is done by simply defining new symbols with fuzzy-shape grammars. For example, a "continuous" symbol can be created for modeling smooth primitives (i.e. those with low angular discontinuity at the characteristic point). In this case, the grammar just defines a simple production rule that evaluates a membership function on the discontinuity property of the handwriting element. This membership function acts as a *fuzzy-threshold* in the sence that, like a threshold that produces a set of acceptable values, a fuzzy-threshold produces a fuzzy-set [19, 20] of acceptable values. Similarly, a "discontinuous" symbol can be created and these two new symbols can be used in other production rules to define even more specific classes of elements. For example, "horizontal" and "vertical" symbols can be created from the "continuous" symbol to represent elementary horizontal and vertical displacements. This time, the fuzzy-thresholds would be applied on the tilt property at the characteristic point. Figure 4 illustrates different "continuous" primitives. They correspond to any path from one point in $\{a, b, c\}$ to one point in $\{g, h, i\}$ (the "horizontal" ones) or from one point in $\{d, e, f\}$ to one point in $\{j, k, l\}$ (the "vertical" ones). It must be emphasized here that these classes of elements, created by fuzzy-thresholds, are in fact fuzzy-sets, which implies that any cursive primitive can be assigned to several classes of elements, for example to the "continuous" and "discontinuous" classes if its discontinuity property is borderline, although not necessarily with the same grade of membership. It would all depend on the membership function[6] used to define the "*low* angular discontinuity" concept.

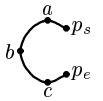
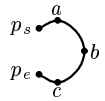
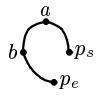
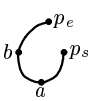
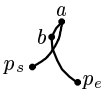
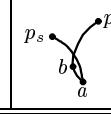
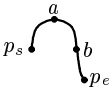
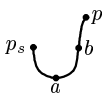
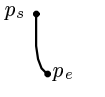
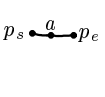
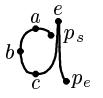
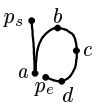
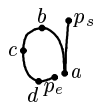
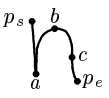
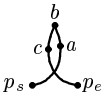
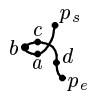
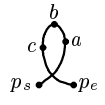
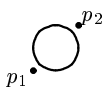
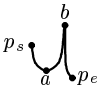
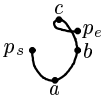
The second phase of the modeling strategy is to create, from the different element classes, symbols that model sub-allograph shapes like *c-shapes*, *i-shapes*, *loops*, *dots*, *t-crossings*, etc. These new symbols, created from combinations of one or more elements, usually involving several production rules, correspond roughly to the "basic shapes" of other structural systems (for example, [7, 9]). However, the sub-allographs described here have two fundamental advantages. First, these symbols again define fuzzy-sets, which implies that a given subset of primitives may be used to construct curves of different symbols with different grades of membership. For instance, the existence of a loop curve does not override the existence of the c-shape curve that is embedded into it. Both interpretations coexist and can be used to assemble different traces. Second, these symbols are attributed just like the elements. Thus they are themselves powerful models. The creation of such symbols also serves to reduce complexity. Indeed, since these symbols model shapes that are to some extent common to several allographs, it makes sense to process them separately. Table 1 illustrates the sub-allograph shapes that were modeled. Most of them are defined with similar attributes: starting and ending attachment points (p_s and p_e) plus a varying number of other attachments $(a, b, c, \ldots)$; starting and ending *curveness* properties, starting and ending *tilt* properties and starting and ending *index* numbers.

The shapes of Table 1 are synthetic and do not illustrate the variations allowed by the production rules of the models. They are shown only to give an idea of the type of sub-allographs that were modeled. The top two lines of the table show the simple sub-allographs while the two bottom lines show the more complex ones. The symbol names are formed by two or three letters that identifies the shape. The first character is always an `s` for *shape*. The second character identifies the letter that resembles the most to the sub-allograph. For instance, symbol `sc` corresponds to a c-shape sub-allograph. It should be noted that sub-allograph symbols like `sc` model very loosely their corresponding letter (in fact, they usually model only part of a letter) in the sense that their curves can be very crooked or disproportionate. Thus, although the trace of a letter `c` will always be a curve of symbol `sc`, the opposite is false. When the third letter of the symbol name is an *i*, it means that it is a horizontally or vertically "inverted" version of the symbol. For instance, symbol `sci` is an inverted c-shape. Symbol `stc` corresponds to a *t-crossing*.

To illustrate how these shapes are coded in the $\mathcal{HAD}$ language, Figure 5 gives the example of the

[6] See Section IV.2 for details.

Table 1: *Sub-allograph symbols.*

name	shape	name	shape	name	shape	name	shape	name	shape
sc		sci		se		sei		sj	
sji		sn		sni		st		stc	
sa		sb		sd		sh		si	
sk		sl		so		su		sv	

```
symbol sa[ps,a,b,c,d,e,f,pe][ns,ne,ts,te,cs,ce]{} :=
   (sc,si) if(sequ(@1.ne,@2.ns))
      {ps = @1.ps; a = @1.a; b = @1.b; c = @1.c; d = @2.a;
       e = @2.b; f = @2.c; pe = @2.pe; ns = @1.ns; ne = @2.ne;
       ts = @1.ts; te = @2.te; cs = @1.cs; ce = @2.ce;},
   (sc,sl) if(sequ(@1.ne,@2.ns) &
              pcent(crx(@2.a-@2.c),crx(@2.b-@1.b))[0,70,0,20])
      {ps = @1.ps; a = @1.a; b = @1.b; c = @1.c; d = @2.a;
       e = @2.b; f = @2.c; pe = @2.pe; ns = @1.ns; ne = @2.ne;
       ts = @1.ts; te = @2.te; cs = @1.cs; ce = @2.ce;},
 1:(ecvp,sa) if(sequ(@1.ne,@2.ns))
      {ps = pxy(crx(@2.ps),cry(@1.ps)); a = @2.a; b = @2.b; c = @2.c;
       d = @2.d; e = @2.e; f = @2.f; pe = @2.pe;
       ns = @1.ns; ne = @2.ne; ts = @2.ts; te = @2.te;
       cs = @2.cs; ce = @2.ce;};
```

Figure 5: *Grammar of symbol* `sa`.

grammar for symbol `sa` which is composed of symbol `sc` followed by either symbol `si` or `sl`. The reserved token `symbol` starts a grammar definition for the symbol name that follows. Here, symbol `sa` is defined with eight attachment points (i.e. `ps`, `a`, `b`, `c`, `d`, `e`, `f` and `pe`) and six properties (i.e. `ns`, `ne`, `ts`, `te`, `cs` and `ce`). The attachment points are illustrated in Table 1 (except for points `d` and `f` that are not shown). The properties are respectively starting and ending index, starting and ending tilt and starting and ending curveness. Three production rules are given for symbol `sa`. The first production rule states that symbol `sa` is constructed from two constituent symbols `sc` and `si` if the predicate `sequ` is true for the ending and starting indexes of `sc` and `si` respectively. In other words, the production rule is applied if and only if the two symbols are in sequence. The @ operator is used in an expression to specify an attribute of a particular constituent. For instance, the notation `@1.ne` means "attribute `ne` of the first constituent". The second production rule states that symbol `sa` may also be constructed from symbols `sc` and `sl` if they are in sequence and if the width of symbol `sl` (i.e. `crx(@2.a-@2.c)`) relative to the total width of symbols `sc` and `sl` (i.e `crx(@2.b-@1.b)`) is not too large. Operators `&` and `|` correspond respectively to fuzzy *and* and *or* operators, functions `crx` and `cry` respectively extract coordinates x and y from the attachment point given as an argument, function `pcent` computes the ratio of its two arguments expressed as a percentage, and an expression of the form $x[a, b, \delta_a, \delta_b]$ (here x represents the result of function `pcent`) is a call to the membership function that sets a fuzzy-threshold (see Section IV.2). Finally, the third production rule states that symbol `sa` may be constructed recursively from itself (i.e. the previous production rules) and symbol `ecvp` (an upward *continuous* and *vertical* element) if the later immediately precedes the former in the sequence. Here, however, the statement `1:(ecvp,sa)` limits the depth of the recursion process to one. The generation rules for the attributes of symbol `sa` are given between a pair of brackets for each production rule. Function `pxy` is used in one case to construct a point from two coordinates given as arguments.

The final phase is the modeling of the allographs themselves. Table 2 illustrate the 54 allograph models that were developed. Each of these is assembled from symbols defined thus far and several *morphological characteristics* are used to discriminate among them. Morphological characteristics (MC) are relative measures of size which characterize each allograph. Consider, for example, the principal allograph of the lower-case letter a shown in Table 2 with its different attachment points. Such an allograph corresponds to a `sa` shape. However, without changing this configuration, simply by moving point d upwards sufficiently relative to point a, then can we get a trace of an allograph of the letter d (see the first line of Figure 6). A first MC expressed by the ratio $\frac{a_y - d_y}{a_y - c_y}$ is thus defined to specify (with a fuzzy-threshold) the height of the i-part relative to the height of the c-part (a_y means coordinate y of point c). Likewise, if point f (and point e with it) is moved up to approximately point d, we could get a trace of an allograph of the letter o (second line of Figure 6). Similarly, if point f is moved down sufficiently, we can get a trace of an allograph of the letter q (third line of Figure 6). The second MC is thus defined as the ratio $\frac{c_y - f_y}{d_y - c_y}$ and is used to specify the height of the final ligature relative to the vertical position of the i-part. Finally, as the horizontal gap within the c-part and the i-part of the symbol widens, the a allograph would become more and more invalid (forth line of Figure 6). Hence, a third MC $= \frac{e_x - a_x}{e_x - b_x}$ is used to specify that gap. The expressions of all MC were obtained by trial and error.

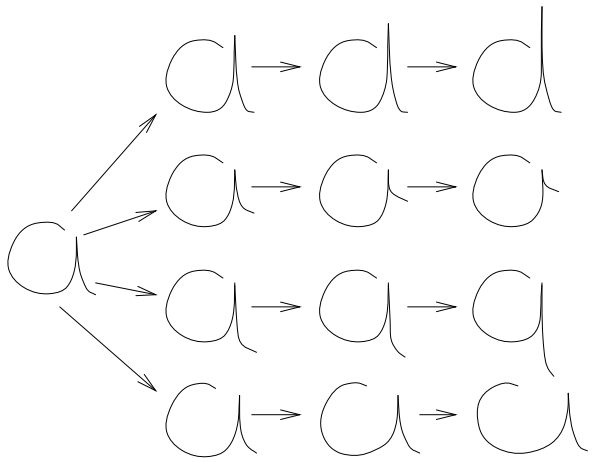
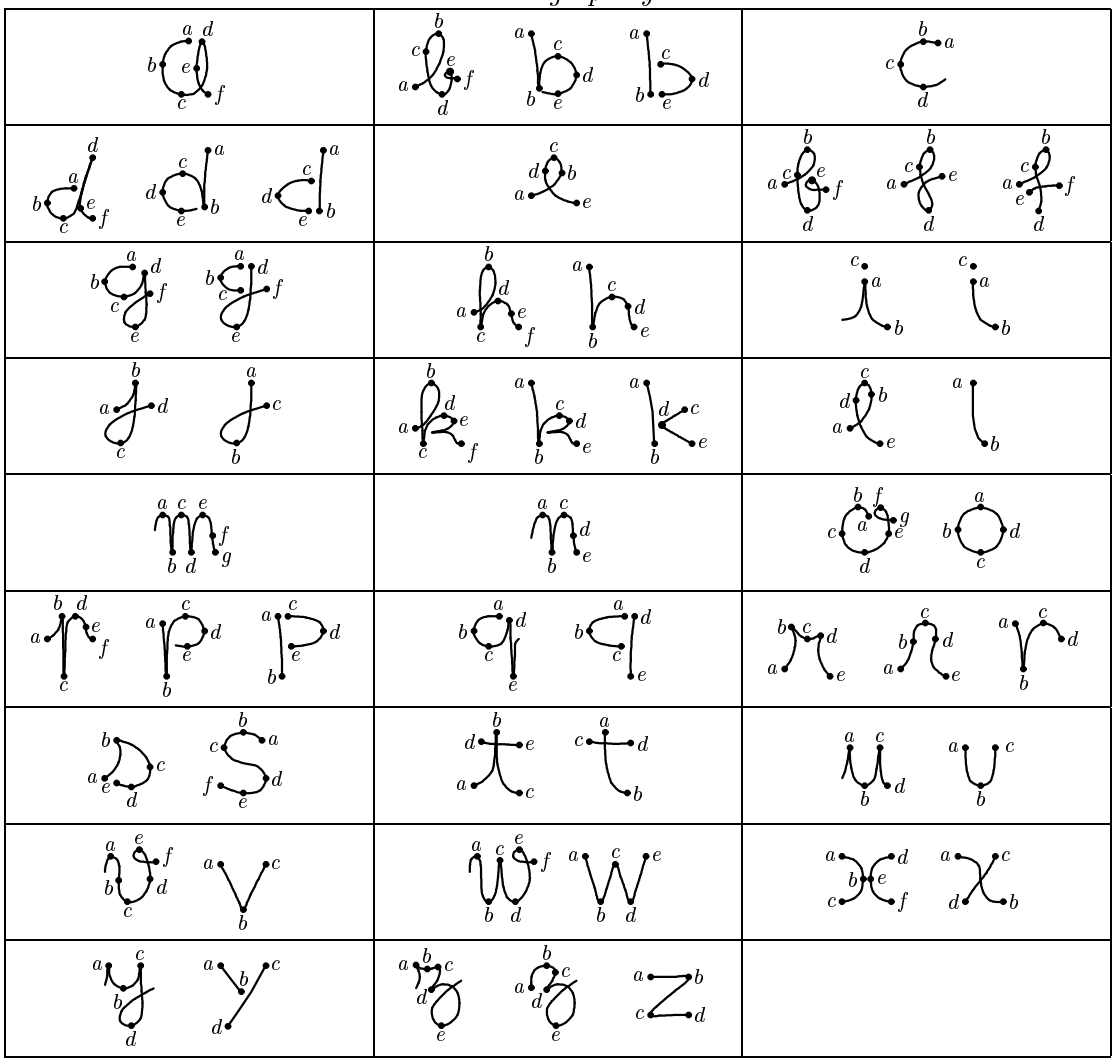


Figure 6: *Examples of traces for the a allograph.*

The grammar for the a allograph is given in Figure 7. Symbol `a`, like all other allograph symbols, is defined with four attachment points and two properties. The attachment points are used in a generic representation of the allograph (see Section V for details). Points `p1` and `p2` correspond respectively to the lower left and upper right corners of a bounding box that encloses the main body of the allograph. Points `p1` and `pu`, when applicable, are the lower and

Table 2: *Allograph symbols.*



```
symbol* a[p1,p2,pl,pu][l,m] {
  symbaux a1[a,b,c,d,e,f][] :=
    (sa) if((crec(@1.ce) | cpos(@1.ce)) & trange(@1.te,-135,45))
      {a = pxy(crx(@1.ps),cry(@1.a)); b = @1.b; c = @1.c; f = @1.pe;
       d = pxy(crx(pxmax(@1.d,@1.f)),cry(@1.e));
       e = pxy(tprd(crx(@1.d)>crx(@1.e),crx(@1.f),crx(@1.e)),cry(@1.e));};
  symbaux ca1[p1,p2][r1,r2,r3] :=
    (a1) if(tdy(a,c) & tdy(d,c) & tdx(e,b))
      {p1 = xypnt(b,c); p2 = xypnt(d,a);
       r1 = yyratio(a,d,a,c); r3 = xxratio(e,a,e,b);
       r2 = tprd(cry(@1.c-@1.f)>0,yyratio(c,f,a,c),yyratio(c,f,d,c));};
} :=
  (ca1) if(@1.r1[-30,50,20,30] & @1.r2[-30,30,20] & @1.r3[-50,30,20])
    {p1 = @1.p1; p2 = @1.p2+(margin/2); pl = @1.p1; pu = @1.p2;
     l = 97; m = 1;};
```

Figure 7: *Grammar of the a allograph.*

upper points of its descender and ascender. Properties `l` and `m` correspond respectively to the ASCII code of its associated character and to its model number. The reserved token `symbaux` starts the definition of an auxiliary (non terminal) symbol. All auxiliary symbols are defined within the scope of two brackets before the production rules of symbol `a` can be enumerated. Here, auxiliary symbol `a1` correspond to the first and only developed allograph model for letter *a*. Its attachment points are illustrated in Table 2. It is defined as an `sa` symbol that has either a rectilinear (null) or counter clockwise ending curveness (i.e. `crec(@1.ce) | cpos(@1.ce)`), and an ending tilt in the 1h30 to 7h30 o'clock range (i.e. `trange(@1.te,-135,45)`). Functions `crec`, `cpos` and `trange` all correspond to fuzzy-thresholds. Auxiliary symbol `ca1` is an intermediate symbol used for debugging purposes. It represents all candidates for this first allograph model. Attachment points `p1` and `p2` are the corners of the bounding box that surround the allograph and properties `r1`, `r2` and `r3` are the three MC described in the previous paragraph. Functions `tdx`, `tdy`, `xypnt`, `xxratio` and `yyratio` are in fact macro definitions defined by the following **C** language preprocessor directives[7] :

```
#define tdx(p1,p2) (crx(@1.p1) > crx(@1.p2))
#define tdy(p1,p2) (cry(@1.p1) > cry(@1.p2))
#define xypnt(x,y) pxy(crx(@1.x),cry(@1.y))
#define xxratio(n1,n2,d1,d2) pcent(crx(@1.n1-@1.n2),\\
                                   crx(@1.d1-@1.d2))
#define yyratio(n1,n2,d1,d2) pcent(cry(@1.n1-@1.n2),\\
                                   cry(@1.d1-@1.d2))
```

and `tprd` is a function that returns its second argument when the first one evaluates to a true value, or its third argument otherwise. Finally, the production rule for symbol `a` states that it is a `ca1` symbol with MCs that respect three fuzzy-thresholds (i.e. `r1[-30,50,20,30]`, `r2[-30,30,20]` and `r3[-50,30,20]`).

Using this strategy, some 18 classes of elements, 20 sub-allograph symbols and 54 allograph models, for lower-case letters of the Roman alphabet, were thus created [14] ($\approx$ 2 000 lines of $\mathcal{HAD}$). Experimental results obtained with these models are presented in section VI. The next subsection discusses the use of fuzzy-thresholds in this modeling strategy. The two subsequent subsections then present the grammatical formalism of $\mathcal{HAD}$ and its associated parsing process.

[7] The preprocessor of the **C** compiler is used by the $\mathcal{HAD}$ compiler.

IV.2 Fuzzy-Thresholds

In the above modeling strategy, allograph grammars are constructed in a hierarchical fashion from simpler symbols and elements (see Figure 1) on which constraints are imposed by the production rules. Parsing these grammars (see Section IV.4) consists in seeking combinations of primitives that respect the conditions of their production rules. The role of fuzzy-thresholds in this process is to limit the combinatorial explosion by filtering in a stepwise manner impossible combinations of existing primitives. The thresholds are fuzzy to avoid having to take binary decision that often leads to unrecoverable errors. Consider for example a proximity condition that we might want to impose on two symbols in order to assemble a third one. Obviously, it is possible to select two curves of the two symbols and arrange them in such a way that this proximity condition is absolutely satisfied. Reciprocally, these curves can be moved away from each other in such a manner that the condition becomes completely unacceptable. But in between the two arrangements, there exists a number of other arrangements where it would be dubious to take an unambiguous binary decision.

Fuzzy-logic is thus used to manage this ambiguity [19, 20]. A fuzzy-threshold applied on some non-fuzzy universe U is a function α_A, called a membership function, that associates a grade of membership $\alpha_A(x)$ to any element $x \in U$ of this universe and generates a fuzzy-set $A \subset U$ such that:

$$A = \{(x, \alpha_A(x)) | x \in U\}, \quad 0 \leq \alpha_A(x) \leq 1 \qquad (1)$$

where interesting element of A are those for which $\alpha_A(x) \neq 0$.

The generic membership function used in all the fuzzy-thresholds of the grammars has the general shape of the trapezoid[8] shown in Figure 8, where a, b, δ_a and δ_b are the four parameters of every fuzzy-threshold (when δ_b is not specified explicitly, it should be assumed equal to δ_a).

IV.3 Grammatical Formalism

Let $\mathcal{F} = \{S_1, \ldots, S_i, \ldots, S_n\}$ be the set of n pertinent symbols for a segmentation problem. Then, a grammar $G_i(T_i, N_i, P_i, S_i)$ is associated with all S_i symbols (except for elements), where T_i is the set of terminal

[8] It should be noted that other types of membership functions can be found in the literature but we have not tried them so far. This one was chosen for its simplicity.

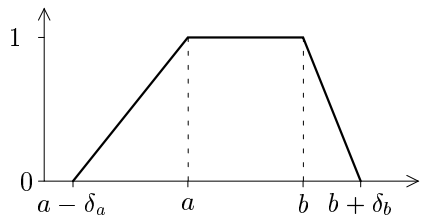


Figure 8: *Generic membership function used for fuzzy-thresholding.*

symbols, N_i is the set of nonterminals, P_i is the set of production rules and $S_i \in N_i$ is the start symbol. Let $V_i = N_i \cup T_i$ denote the set of all vocabulary symbols for grammar G_i; and associate $\forall v \in V_i$ a level number $\nu(v) \in \{0, 1, \ldots, k_i\}$ with $\nu(S_i) = k_i$ and $\forall v \in T_i$, $\nu(v) = 0$.

The set T_i of terminal symbols contains either elements or start symbols defined by other grammars. The possibility of using start symbols as terminal symbols of other grammars justifies the hierarchical epithet of the $\mathcal{HAD}$ language. However, to avoid circular definitions, if $S_j \in T_i$ then the constraint $j < i$ is imposed to make the grammars *stratified*.

The nonterminal symbols of N_i are intermediate patterns that permit the passage from the start symbol to terminal symbols. Unlike string grammars, all symbols $v \in V_i$ possess a non trivial structure consisting of a name and two sets of attributes:

$$\text{v} = \textit{name}[\textit{attachment points}][\textit{properties}]$$

where *name* is a token that identifies uniquely the symbol, *attachment points* is a set of points used to specify the spatial arrangement of the symbol, and *properties* is a set of measures used to characterize the symbol.

The set P_i contains all production rules for symbols of N_i. Every production rule has the form $(v := \lambda, C, G)$, which indicates that symbol $v \in N_i$ can be rewritten as the group of constituent symbols $\lambda = \{c_1, \ldots, c_j, \ldots, c_m\}$ if condition C is verified, with $c_j \in V_i$ and $\nu(c_j) \leq \nu(v)$, $\forall j \in \{1, \ldots, m\}$. The case where $\nu(c_j) = \nu(v)$ is not, however, permitted for the first production rule of v. The applicability condition C can mix both the attachment points and the properties of any symbol $c_j \in \lambda$ to test for acceptable spatial arrangement and semantic coherence. It is expressed in fuzzy-logic. G describes the set of generation rules for computing the attributes of v. Again, these rules are functions of any attribute of any symbol $c_j \in \lambda$. No production rules are associated with terminal symbols.

IV.4 Parsing Process

The only role of the parser process is to assemble curves, starting initially with primitives, according to the set of production rules P_i of a grammar G_i. Let o denote a particular curve. Then, curve o is a structure that contains certain informations:

1. constituent curves $\mathcal{C}(o) = \{c_1, \ldots, c_m\}$
2. attachment points $\mathcal{A}(o) = \{a_1, \ldots, a_u\}$
3. properties $\mathcal{P}(o) = \{p_1, \ldots, p_v\}$
4. grade of membership $\alpha(o)$

The constituents are the curves that were assembled to form curve o. The attachment points and properties are computed from the attributes of the constituent curves using the generation rules of the production rule that created the curve.

Each curve is associated with a particular symbol, that is, it belongs to a particular symbol class which corresponds to a fuzzy-set. Let C denote the applicability condition of the rule that created a curve o. Then, the *grade of membership* α of that curve is defined by:

$$\alpha(o) = \min\left[C(o), \min_i \alpha(c_i)\right], \quad 1 \leq i \leq m \qquad (2)$$

where $\alpha(c_i)$ is the grade of membership (in its own class) of the i^{th} constituent of o.

The *domain* $\mathcal{D}$ of a curve o is defined by the set of primitives that are either direct constituents of o or, recursively, constituents of its constituents:

$$\mathcal{D}(o) = \sum_{i=1}^{m} \mathcal{D}(c_i) \qquad (3)$$

A curve o is said to be *redundant* relative to another curve o' of the same class if and only if its domain is completely included in the domain of o' and its grade of membership is lower than or equal to the grade of membership of o':

$$\exists\, o' \neq o \;:\; \left\{ \begin{array}{c} \mathcal{D}(o) \subseteq \mathcal{D}(o') \\ \text{and} \\ \alpha(o') \geq \alpha(o) \end{array} \right. \iff \text{redundant}(o) \qquad (4)$$

Two curves o and o' are said to be *consistent* if and only if the intersection of their domains is empty:

$$\mathcal{D}(o) \cap \mathcal{D}(o') = \emptyset \iff \text{consistent}(o, o') \qquad (5)$$

Two curves o and o' are said to be *adjacent* if and only if the distance between their respective bounding box along the X axis[9] is smaller than a certain threshold. The bounding box of a curve corresponds to the smallest imaginary rectangle that completely encloses its attachment points.

We now consider without loss of generality — because of the stratified formalism — the problem of parsing grammar $G_i(T_i, N_i, P_i, S_i)$ associated with symbol S_i. Let $T_i = \{T_i^1, \ldots, T_i^j, \ldots, T_i^p\}$ be the set of p terminal symbols. For each symbol T_i^j, a set of curves D^j is known and constitutes the starting data:

$$D^j = \{d_1^j, d_2^j, \ldots, d_{p_j}^j\}, \qquad 1 \leq j \leq p \tag{6}$$

It is subsets of these curves that can, depending on the set of production rules P_i, correspond to constituent curves of symbol S_i. In fact, the objective of the parsing process is to identify these subsets that at step $i+1$ will possibly serve to construct the curves of S_{i+1}.

Let $N_i = \{N_i^1, \ldots, N_i^j, \ldots, N_i^q\}$ be the set of q nonterminals. Again, without loss of generality, we consider the problem of parsing symbol N_i^j. Then we seek to find the set O^j of curves that respect the production rules of N_i^j:

$$O^j = \{o_1^j, o_2^j, \ldots, o_{q_j}^j\}, \qquad 1 \leq j \leq q \tag{7}$$

Ultimately, it is the set O^q — the curves associated with $N_i^q = S_i$ — that we seek.

Let $\lambda = \{c_i^{j.1}, \ldots, c_i^{j.l}, \ldots, c_i^{j.m}\}$ be the set of m constituents of N_i^j for a particular production rule, where $c_i^{j.l}$ represents the symbol of the l^{th} constituent of symbol N_i^j for which its associated curve set $O^{j.l}$ is known (i.e. either $c_i^{j.l}$ corresponds to a terminal symbol, say T_i^k, and thus $O^{j.l} = D^k$, or $c_i^{j.l}$ is a nonterminal that can and must be parsed before N_i^j). Then, the set $\mathcal{O}$ of possible assemblies for curves of N_i^j is constructed in the following manner:

$$\mathcal{O} = O^{j.1} \otimes O^{j.2} \otimes \cdots \otimes O^{j.m} \tag{8}$$

with the product $A \otimes B$ defined as follows:

$$A \otimes B = \{x \cup y |\ x \in A,\ y \in B,\ x \bowtie y\} \tag{9}$$

and where the notation $x \bowtie y$ should be interpreted as: x is *consistent* with and *adjacent* to y. Product $A \otimes B$ is commutative and associative.

[9] This definition is used to limit combinatorial explosion. Obviously, it is relevant to the parsing of the Roman alphabet in which letters are aligned on a common baseline. For other types of script, it might not be justified.

Then, the set O^j is given by:

$$O^j = \left\{ o \in \mathcal{O} \;\middle|\; C\left(\sum\{\mathcal{A}(c) \cup \mathcal{P}(c) |\ c \in \mathcal{C}(o)\}\right) > 0 \right\} \tag{10}$$

where C is the applicability condition of the production rule, which is a function of the sum of the attributes ($\mathcal{A}$ and $\mathcal{P}$) of the set of constituent curves $\mathcal{C}(o)$ of o.

The attributes of each curve o_w^j of set O^j are then computed using the set of generation rules G of the production rule:

$$\begin{aligned} &\forall a_k \in \mathcal{A}(o_w^j), \quad 1 \leq w \leq q_j, \\ &\quad a_k = G_{a_k}\left(\sum\{\mathcal{A}(c) \cup \mathcal{P}(c) |\ c \in \mathcal{C}(o_w^j)\}\right) \end{aligned} \tag{11}$$

$$\begin{aligned} &\forall p_k \in \mathcal{P}(o_w^j), \quad 1 \leq w \leq q_j, \\ &\quad p_k = G_{p_k}\left(\sum\{\mathcal{A}(c) \cup \mathcal{P}(c) |\ c \in \mathcal{C}(o_w^j)\}\right), \end{aligned} \tag{12}$$

where G_{a_k} and G_{p_k} represent respectively the generation rules for attachment point a_k and property p_k of the production rule.

When symbol N_i^j possesses multiple production rules, they are applied sequentially. *Non-redundant* curves are appended to set O^j:

$$O^j = \left\{ o \in \sum_r O_r^j \;\middle|\; \neg\text{redundant}(o) \right\} \tag{13}$$

where O_r^j denotes the parse result for the r^{th} production rule.

Figure 9 summarizes the parsing process for a given production rule $R = (\lambda, C, G)$ of symbol N_i^j and the superset $\Omega = \{O^{j.1}, \ldots, O^{j.m}\}$ of constituent symbol curve sets.

V Sequence Construction

The sequence construction process consists in analyzing the spatial adjacency of segmented allograph traces for generating morphologically and pragmatically coherent character sequence outputs. Consider, for example, a trace of the letter *l*. When taken out of context, such a trace can usually be interpreted either as an *e* or an *l*. Only by looking at the context of adjacent traces, can we discriminate between these two interpretations.

Traditionally, recognition systems detect, prior to recognition, the two frontiers that separate the three zones of handwriting: upper, middle and lower. The middle zone contains the main body of letters, while the upper and lower zones contains respectively their

$\mathcal{HAD}$**_parser**(R, Ω)
/* $R = (\lambda, C, G)$ is the production rule of symbol N_i^j and $\Omega = \{O^{j.1}, \ldots, O^{j.m}\}$ is the superset of constituent curve sets */
{
/* construct consistent and adjacent groups */
$\mathcal{O} = \{\ \}$;
For $l = 1$ **to** m **{** $\mathcal{O} = \mathcal{O} \otimes O^{j.l}$; **}**
/* select groups with coherent attributes */
$O^j = \{o \in \mathcal{O} \mid C\,(\sum\{\mathcal{A}(c) \cup \mathcal{P}(c) \mid c \in \mathcal{C}(o)\}) > 0\}$;
For $w = 1$ **to** q_j **{** /* generate attributes of o_w^j */
For $k = 1$ **to** u **{** /* generate attachment pts */
$a_k = G_{a_k}\,(\sum\{\mathcal{A}(c) \cup \mathcal{P}(c) \mid c \in \mathcal{C}(o_w^j)\})$; **}**
For $k = 1$ **to** v **{** /* generate properties */
$p_k = G_{p_k}\,(\sum\{\mathcal{A}(c) \cup \mathcal{P}(c) \mid c \in \mathcal{C}(o_w^j)\})$; **}**
}
return O^j ;
}

Figure 9: *Algorithm for parsing a production rule.*

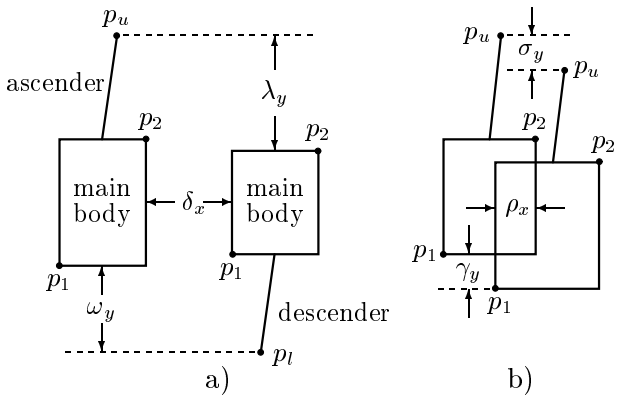


Figure 10: *Spatial adjacency of allograph traces.* a) *case of horizontally disjoint main bodies,* b) *case of overlapping main bodies.*

ascenders and descenders. They use these frontiers to invalidate any incoherent recognition hypothesis. The problem with such an approach is that the position of these frontiers often fluctuates within a cursive word, thus trying to approximate them with two horizontal lines is a difficult problem and can induce many recognition errors.

The approach presented in this section uses local criteria to accept or reject combinations of segmented traces.

V.1 Adjacency Constraints

Allographs are designed in such a way that each trace is segmented with 4 attachment points $p_1(x_1, y_1)$, $p_2(x_2, y_2)$, $p_l(x_l, y_l)$ and $p_u(x_u, y_u)$ corresponding respectively to the lower left and upper right corners of the bounding box enclosing its main body, and to its lower and upper points (see Figure 10a). When the allograph doesn't possess an ascender (or a descender), point p_u (or p_l) then has the value $p_u = p_2$ (or $p_l = p_1$). The coordinates of these points are used to specify the adjacency constraints between pairs of traces. Two traces are considered adjacent if, and only if, they respect both a *horizontal constraint* that measures the coherence of their horizontal spacing, and a *vertical constraint* that measures the quality of their vertical alignment.

For the horizontal constraint, the two cases in Figure 10 must be distinguished. When the two main bodies of the traces are disjoint (Figure 10a), the size δ_x of the gap between them should not be too large in proportion to their heights. The horizontal constraint for that case is thus defined by a fuzzy-threshold on the ratio of δ_x over the height of the tallest main body. When the two main bodies overlap (Figure 10b), the size ρ_x of this overlap should not be too large in proportion of their widths. The horizontal constraint for that case is defined by another fuzzy-threshold on the ratio of ρ_x over the width of the thinnest main body.

The vertical constraint is composed of two parts: one for the alignment of the upper zones and one for the alignment of the lower zones. These two alignments are handled in essentially the same way by considering three cases that must be dealt with. First, when one trace has an ascender or a descender and the other doesn't (Figure 10a), then the size λ_y (or ω_y) should be large enough in proportion to the height of the ascender (or descender). The vertical constraint for that case is thus defined by a fuzzy-threshold on the ratio of λ_y (or ω_y) over the height of the ascender (or descender). Second, when both traces have ascenders (which is the case in the upper zones in Figure 10b) or both have descenders (not illustrated), then the size σ_y should be large enough in proportion to the height of the ascender (or descender). The vertical constraint for that case is thus defined by a fuzzy-threshold on the ratio of σ_y over the height of the tallest ascender (or descender). Third, when no trace possesses an ascender (not illustrated) or no trace possesses a descender (as is the case in the lower zones in Figure 10b), then the size γ_y should be large enough in proportion to the heights of the main bodies. The vertical constraint for that case is thus defined by still another fuzzy-threshold on the ratio of γ_y over the height of the tallest main body.

These constraints are used to model an adjacency

relation between allographs defined as a symbol in $\mathcal{HAD}$ called *edge* (for details consult [21]). This symbol enables the construction of an allograph segmentation graph.

V.2 Segmentation Graph

An allograph segmentation graph is defined by a set of nodes N and a set of edges E. Each node $n \in N$ is defined by a couple $n = (\mu, \alpha_\mu)$, where μ represents a trace and α_μ its grade of membership. The grade of membership α_μ is a measure of how this particular trace fits the corresponding allograph. Let Ω be the set of traces segmented in a particular cursive word, then N is defined by the following equation:

$$N = \{(\mu, \alpha_\mu) \mid \mu \in \Omega, \alpha_\mu > 0\} \tag{14}$$

Each edge $e_{ij} \in E$ is defined by a triplet $e_{ij} = (n_i, n_j, \chi_{ij})$ where $(n_i, n_j) \in N^2$, $i \neq j$ are two nodes of the graph and χ_{ij} is the weight of the edge, which corresponds to the grade of membership of edge (n_i, n_j) to the fuzzy-set of adjacent traces:

$$\chi_{ij} = \min[\chi_h(\mu, \nu), \chi_v(\mu, \nu)] \tag{15}$$

with $n_i = (\mu, \alpha_\mu)$ and $n_j = (\nu, \alpha_\nu)$ and where χ_h and χ_v are respectively the horizontal and vertical constraints. Then, E is defined by the following equation:

$$E = \{(n_i, n_j, \chi_{ij}) \mid (n_i, n_j) \in N^2, i \neq j, \chi_{ij} > 0\} \tag{16}$$

A path in the segmentation graph from the node associated with trace μ to the node associated with trace ν is defined by a sequence of k nodes $n_1, n_2, \ldots, n_k$ such that $n_1 = (\mu, \alpha_\mu)$, $n_k = (\nu, \alpha_\nu)$ and n_i and n_{i+1} are adjacent nodes $\forall\, i \in \{1, \ldots, k-1\}$.

A character sequence can be associated with every path in the graph. These sequences then correspond to different, possibly incomplete, interpretations of the unknown cursive word. Three criteria are used to sort these interpretations. The first concerns the quality of the segmented traces in the path. The higher their degree of membership, the more likely the interpretation. The second deals with adjacency between traces. Again, the better their adjacency relations are, the more likely the interpretation. Finally, the third criterion takes into account the total number of primitives associated with the traces in the path. Obviously, the greater that number is, the more representative the resulting path.

These criteria are combined using the following ranking coefficient Γ:

$$\Gamma = \sum_{i=1}^{k} \mathrm{card}(n_i) \left[\frac{1}{k} \sum_{i=1}^{k} \alpha_i + \frac{1}{k-1} \sum_{i=1}^{k-1} \chi_{i,i+1} \right] \tag{17}$$

where n_i is the i^{th} node in the path, α_i is the grade of membership for the trace associated with n_i, $\chi_{i,i+1}$ is the grade of membership of edge $e_{i,i+1}$, and $\mathrm{card}(n_i)$ represents the number of primitives contained in the domain of the trace associated with node n_i.

VI Experimental Results

Allograph models for all 26 lower-case letters of the Roman alphabet have been hand-generated and optimized using a multi-writer isolated cursive letter dataset containing 10 samples of each letter written by 13 different writers (total: 3,380 cursive letters). These writers were given sheets of paper and were instructed to reflect on their own writing styles or on any other style that they knew of. Then, they were asked to summarize the fruit of their reflections by writing 10 samples of each letter representing the widest range of variants, including ligatures that can precede or follow the letter.

For recognition experiments, two test datasets were constructed with the help of 10 volunteers (6 of them participated in the construction of the learning datasets and 4 did not). They were instructed to use their natural, although clean, handwriting style. The first dataset contains 10 isolated cursive samples/writer for each of the 26 letters (total: 2,600 cursive letters). This dataset will be used to evaluate the performances of the allograph models without the need for adjacency constraints. The second dataset contains 100 cursive letter sequences/writer (total: 1,000 letter sequences). Writers were asked to perform random letter sequences instead of dictionary words to compare system performance with human performance on a common dataset using only morphological and pragmatic knowledge. Hence, 100 character sequences were generated randomly, each containing from 5 to 7 lower-case characters, while respecting the letter frequencies of a 32,000-word French dictionary. Figure 11 gives a few samples taken from this test dataset.

Data acquisition was conducted using a *PenPad 300* digitizing tablet from Pencept[10] Inc. with a reso-

[10] Now a division of Numonics Inc.

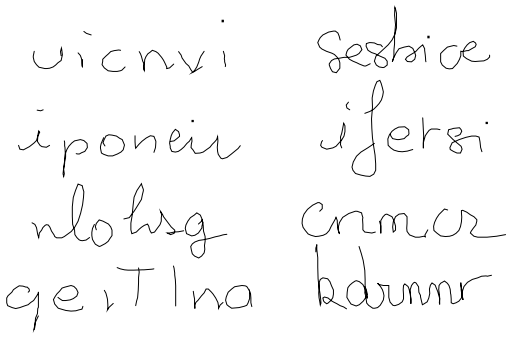


Figure 11: *Samples of cursive random letter sequences.*

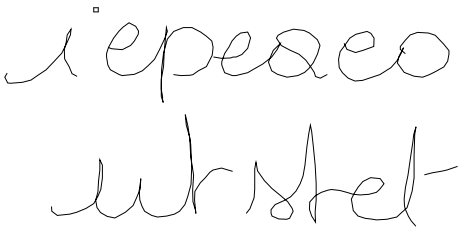


Figure 12: *Samples of letter sequences* `iepeaeo` *and* `utstet` *with unrecognized traces of letters a and t.*

lution of $0.001''$, a sampling frequency of 100 Hz and a precision (as specified by the manufacturer) of $0.005''$.

Average recognition rates for the isolated letters are given in Table 3 for each individual writer. For this experiment, the *sequence construction* process has been slightly modified to permit only paths of unit length in the segmentation graph. In this case, equation 17 becomes:

$$\Gamma = \mathrm{card}(\mathcal{D}(\mu)) \times \alpha_\mu \tag{18}$$

and the recognized letter is the one associated with the trace μ for which Γ is maximum. On average, 91.7% of the samples were correctly recognized. The results for each writer show that the allograph models are quite robust over different handwriting styles. Especially considering the fact that system parameters are writer-independent and that the 4 writers who didn't participate in the construction of the learning dataset (#4, 6, 8 and 10) all obtained rates over 90%.

For the second experiment, the cursive letter sequences were used to evaluate the combined performance of the allograph models and the adjacency constraints. Table 4 gives recognition rates by writer for the cursive letter sequence dataset. They were obtained by computing error rates using the Wagner-Fischer string-to-string editing distance [28] with unit cost for insertion, deletion and substitution. These recognition rates should be interpreted as similarity measures between the correct character sequence and the system's sequence outputs. The first line of Table 4 gives the results when considering only the character sequence output ranked first by the system. The next four lines of this table list the results when the best of the first 2, 3, 5 and 10 sequence outputs are considered respectively.

The average recognition rate thus varies from 84.4% to 91.6%, depending on whether only the sequence ranked first is considered, or if the best of the top 10 sequences is accepted. These performances are very encouraging, considering once again that the recognition system is multi-writer, in the sense that system parameters are the same for all writers, and the fact that no linguistic knowledge is used for recognition. However, contrary to the results obtained for the first experiment, the writers who didn't participate in the construction of the learning dataset didn't perform as well as the others, except for writer #10. This can be explained in part by the fact that writers can sometimes modify their handwriting style when they switch from isolated letters to attached letters, as was especially the case for writers #4 and #6 who used several allographs that weren't modeled. For instance, Figure 12 shows an example of letter sequences `iepeaeo` and `utstet`. In the first case, the trace for the letter *a* clearly does not match the only developed model for that letter and thus cannot be recognized. In the second case, the first two traces for the letter *t* have no clear t-crossings and cannot be recognized since the system does not try any "guessing" nor does it proceed by elimination like a human would probably do.

Table 4 also gives the recognition rates obtained by human readers on the same dataset. Because the goal of 100% recognition is probably not realistic, this experiment was conducted to improve the performance evaluation of the system relative to that of the best available cursive script recognizers: humans. A group of 10 volunteers (different from the writers) was chosen from a pool of graduate students and research staff, some of them experts in pattern recognition and image processing. The 1,000-letter sequences in the dataset were assigned randomly to the volunteers (100 each) with the constraint that each reader would be presented with 10 samples from each writer. The samples of the dataset were printed on sheets of paper (25 samples/sheet) and the volunteers were asked simply to transcribe into an ASCII file what they could read

Table 3: *Recognition rates for isolated letters (%).*

writer										average
#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	rate
89.2	96.2	89.6	91.9	92.3	95.4	95.0	92.3	84.2	90.4	91.7

Table 4: *Recognition rates for cursive random letter sequences (%).*

accepted ranks	writer										average rate
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	
1	91.0	91.7	84.5	78.5	92.2	68.2	82.7	75.5	89.6	89.9	84.4
1-2	93.1	95.8	88.9	81.9	94.8	73.4	85.5	80.1	94.1	93.8	88.1
1-3	93.1	96.7	90.5	83.4	95.8	75.2	85.8	83.0	95.3	94.6	89.3
1-5	93.5	97.7	93.1	85.3	95.9	77.5	86.9	85.5	96.1	95.4	90.7
1-10	93.5	98.2	93.8	87.1	96.2	78.6	87.3	88.4	96.7	96.1	91.6
humans	98.1	99.2	94.6	95.1	97.1	91.1	97.4	92.7	97.5	97.2	96.0
residues	4.6	1.0	0.8	8.0	0.9	12.5	10.1	4.3	0.8	1.1	4.4

for each sample. The only hint that they were given was that the samples contained only lower case-letters of the Roman alphabet.

The results show that on average, the human readers correctly recognized only 96% of the letters and that human performance is highly correlated (0.92) with system performance. Indeed, the most illegible writers according to the system (#4, 6 and 8) are the same for humans. Furthermore, the worst reader/writer combination produced up to 21.7% of errors for writer #6, which is comparable to the system's performance for that writer. Finally, if average human performance is considered the new goal and the best of the top ten sequence outputs is acceptable, then the residual error of the system is given on the last line of Table 4. It can be observed that for writers #2, 3, 5, 9 and 10, this residual error is less than 1.1%.

The choice of selecting the top ten results is of course arbitrary. The important thing is that most of the correct characters be contained in these first few outputs so that a list of morphologically and pragmatically coherent alternatives of limited length can always be produced. Then, any available linguistic knowledge can be used by higher decision processes. For instance, in a typical limited vocabulary application, efficient and well-known search techniques can be used to scan a dictionary to find words that approximately match the list of character sequences produced by the system.

The recognition system presented in this paper is implemented by a C program of approximately $5,000$ lines. It was run on a SUN SPARKstation 2 under the UNIX operating system. Average processing time for the cursive letter sequences (5 to 7 letters long) of the dataset was around 2.4 seconds. Cursive words of up to 25 letters were processed in less than 10 seconds. For processing the test dataset, the program used at most 1.1 Mbytes of memory including approximately 600K for the recognition program itself, the $\mathcal{HAD}$ parser, the allograph grammars and 25 samples of the dataset maintained in memory at any given time.

VII Conclusion

This paper has presented an approach for modeling cursive script allographs and recognizing them using exclusively morphological and pragmatic knowledge. The main contributions of this work are the following:

1. A model of handwriting that enables the extraction of attributed cursive primitives without loss of morphologically pertinent information.
2. A dedicated programming language ($\mathcal{HAD}$) for specifying allographs using a fuzzy-shape grammar formalism. This language enables modular development of allographs.
3. A parser for fuzzy-shape grammars. Attributed shape-grammars enable the use of both structural and statistical features for disambiguating conflicting hypotheses. The use of fuzzy-logic by the parser avoids the need to make binary decisions concerning the presence of structural fea-

tures.

4. Adjacency constraints for building an allograph segmentation graph representing possible sequence outputs. These local constraints allow the pruning of many incoherent recognition hypotheses. Also, they eliminate the need for finding in a preprocessing step, the two frontiers that separate a cursive word into upper, middle and lower zones of handwriting.

The idea behind the whole approach is to separate two distinct, though complementary, problems: the *recognition* of a set of graphics symbols (i.e. a given alphabet), and the *reading* of a message coded with these symbols (i.e. a text written in a given language). The advantage of such a separation is that solutions to these two fundamental and difficult problems can be optimized independently and, eventually, be integrated into a global handwriting reading system.

Of course, this is not to say that linguistic knowledge cannot or should not be integrated into recognition processes, but rather that recognition should not be restricted to the linguistic knowledge. In that sense, systems that restrict recognition to a limited and fixed vocabulary are useful only for special applications. The use of letter n-grams is less restrictive and could obviously benefit the sequence construction process described in Section V. But the object of this work was to demonstrate the feasibility of recognizing cursive script, at least partially, without using any linguistic context.

The allograph models developed for testing the proposed approach have demonstrated this feasibility even though they are far from perfect. Indeed, for half the writers, when considering the best of the top ten sequence outputs of the system, the results obtained were almost as good as those of human readers, while for the other half they were somewhat inferior. After examining some of the recognition errors, and without changing the modeling strategy, it is clear that some allographs could be recoded to make them more robust and that some new allographs should be added. This is, however, a time-consuming process even though it has to be done only once.

Furthermore, it can be argued that the human readers were very proficient with cursive script considering their long academic careers. An open question is how would, for instance, high school students perform ? Still more interesting, by replicating the human reading experiment with different age groups, could we determine to what level of human training does a system perform ?

Finally, we have not addressed the problem of adapting the allograph models to a particular writer because our objective was to develop basic writer-independent intrinsic models. However, once built, the allographs can be fine-tuned for specific writers by automatically changing the parameters of the membership functions that specify the fuzzy-thresholds on their morphological characteristics. Any statistical learning approach can be used for this (including neural nets and genetic algorithms) since these parameters can be assembled into a vector of fixed dimension. The adjacency constraints can also be fine-tuned in a similar fashion. As for adding new allograph models, it requires first to manually create new grammars. The difficulty of this task ranges from relatively simple to moderately complex depending on whether or not the sub-allograph shapes of the allographs that we want to create are already adequately modeled. Thus in the best of cases, the task is to create and experiment with two or three production rules per allograph and, in the worst of cases, it may require to deal with up to approximately ten production rules.

VIII Acknowledgements

The authors would like to thank all the volunteers who contributed samples of cursive handwriting, as well as those who participated in the human reading experience. They would also like to thank the referees for their helpful comments.

References

[1] **Plamondon** R., **Leedham** G., *Computer Processing of Handwriting*, World Scientific Publishing, 413p, 1990.

[2] **Tappert** C.C., **Suen** C.Y., **Wakahara** T., "The State of the Art in On-Line Handwriting Recognition", *IEEE trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-12, no. 8, pp. 787-808, 1990.

[3] **Nouboud** F., **Plamondon** R., "On-Line Recognition of Handprinted Characters: Survey and Beta Tests", *Pattern Recognition*, vol. 23, no. 9, pp. 1031-1044, 1990.

[4] **Ward** J.R., **Phillips** M.J., "Digitizer Technology: Performance Characteristics and Effects on the User Interface", *IEEE Computer Graphics and Applications*, pp. 31-44, April 1987.

[5] **Ward** J.R., **Blesser** B., "Interactive Recognition of Handprinted Characters for Computer Input",

IEEE Computer Graphics and Applications, pp. 24-37, September 1985.

[6] **Tappert** C.C., **Fox** A.J., **Kim** J., **Levy** S.E., **Zimmerman** L.L., "Handwriting Recognition on Transparent Tablet Over Flat Display", *Proc. 1986 SID International Symposium Digest of Technical Papers*, pp. 308-312, May 1986.

[7] **Brocklehurst** E.R., "The NPL Electronic Paper Project", *International Journal of Man-Machine Studies*, vol. 34, pp. 69-95, 1991.

[8] **Plamondon** R., **Maarse** F.J., "An Evaluation of Motor Models of Handwriting", *IEEE trans. on Systems, Man and Cybernetics*, vol. SMC-19, no. 5, pp. 1060-1072, 1989.

[9] **Mermelstein** P., **Eden** M., "Experiments on Computer Recognition of Connected Handwritten Words", *Information Control*, vol. 7, no. 2, pp. 255-270, 1964.

[10] **Tappert** C.C., "Cursive Script by Elastic Matching", *IBM Journal of Research and Development*, vol. 26, pp. 765-771, 1982.

[11] **Higgins** C.A., **Whitrow** R., "On-Line Cursive Script Recognition", *Proc. Interact'84,* 1st *IFIP Conference on Human-Computer Interaction*, vol. 2, pp. 140-144, 1984.

[12] **Schomaker** L., "Using Stroke- or Character-Based Self-Organizing Maps in the Recognition of On-Line, Connected Cursive Script", *Pattern Recognition* (special issue on Handwriting Analysis and Recognition), vol. 26., no. 3, pp. 443-450, 1993.

[13] **Morasso** P., **Barberis** L., **Pagliano** S., **Vergano** D., "Recognition Experiments of Cursive Dynamic Handwriting with Self-Organizing Networks", *Pattern Recognition* (special issue on Handwriting Analysis and Recognition), vol. 26., no. 3, pp. 451-460, 1993.

[14] **Parizeau** M., "Reconnaissance d'écriture cursive par grammaires floues avec attributs: étape vers la conception d'un bloc-notes électronique", Ph.D. Thesis, Ecole Polytechnique de Montréal, 1992.

[15] **Plamondon** R., "Steps Towards the Production of an Electronic Pen Pad", *Proc. International Conference on Document Analysis and Recognition*, St-Malo, September 30 to October 2, pp. 361-371, 1991.

[16] **Suen** C.Y., "Handwriting Education — A Bibliography of Contemporary Publications", *Visible Language*, vol. 9, p. 145-158, 1975.

[17] **Parizeau** M., **Plamondon** R., "A Handwriting Model for Syntactic Recognition of Cursive Script", *Proc. of the* 11th *International Conference on Pattern Recognition*, The Hague, August 31 to September 3, vol. II, pp. 308-312, 1992.

[18] **Parizeau** M., **Plamondon** R., **Lorette** G., "Fuzzy-Shape Grammars for Cursive Script Recognition", *Advances in Structural and Syntactic Pattern Recognition*, H. Bunke (editor), World Scientific Publishing, pp. 320-332, 1993.

[19] **Zadeh** L.A., "Fuzzy Sets as a Basis for a Theory of Possibility", *Int. Journal for Fuzzy sets and Systems*, vol. 1, no. 1, pp. 3-28, 1978.

[20] **Dubois** D., **Prade** H., *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, 1980.

[21] **Parizeau** M., **Plamondon** R., "Allograph Adjacency Constraints for Cursive Script Recognition", *Proc. of the Third International Workshop on Frontiers in Handwriting Recognition*, Buffalo, May 25-27, pp. 252-261, 1993.

[22] **Plamondon** R., **Alimi** A., **Yergeau** P., **Leclerc** F., "Modeling Velocity Profiles of Rapid Movements: a Comparative Study", *Biological Cybernetics*, in press, 1993.

[23] **Plamondon** R., "A Handwriting Model Based on Differential Geometry", in *Computer Recognition and Human Production of Handwriting*, R. Plamondon, C.Y. Suen and M. Simner (editors), World Scientific Publishing, pp. 179-192, 1989.

[24] **Plamondon** R., "A Model-Based Segmentation Framework for Computer Processing of Handwriting", *Proc. of the* 11th *International Conference on Pattern Recognition*, The Hague, August 30 to September 3, vol. II, pp. 303-307, 1992.

[25] **Plamondon** R., "Looking at Handwriting Generation from a Velocity Control Perspective", *Acta Psychologica*, vol. 82, pp. 89-101, 1993.

[26] **Guerfali** W., **Plamondon** R., "Normalizing and Restoring On-Line Handwriting", *Pattern Recognition* (special issue on Handwriting Analysis and Recognition), vol. 26, no. 3, pp. 419-431, 1993.

[27] **Davis** L.S., **Henderson** T.C., "Hierarchical Constraint Processes for Shape Analysis", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 3, pp. 265-277, 1981.

[28] **Wagner** R.A., **Fischer** M.J., "The String-to-String Correction Problem", *Journal of the ACM*, vol. 21, no. 1, pp. 168-173, 1974.