Pergamon

Pattern Recognition, Vol. 28, No. 6, pp. 807–822, 1995
Elsevier Science Ltd

0031-3203/95 $9.50 + .00

0031-3203(94)00160-X

TOWARDS AN ART BASED MATHEMATICAL EDITOR, THAT USES ON-LINE HANDWRITTEN SYMBOL RECOGNITION

YANNIS A. DIMITRIADIS*† and JUAN LÓPEZ CORONADO‡

* Department of Signal Theory, Communications and Telematics Engineering, School of Telecommunications Engineering, University of Valladolid, Calle Real de Burgos S/N, 47011 Valladolid, Spain

‡ Department of Automatic Control and Systems, School of Industrial Engineering, University of Valladolid, Paseo del Cauce S/N, 47011 Valladolid, Spain

(*Received* 9 *November* 1993; *in revised form* 8 *November* 1994; *received for publication* 14 *December* 1994)

Abstract—A new mathematical editor, based on the recognition of run-on discrete handwritten symbols, is proposed. The tested laboratory prototype of the system, modular and adaptable to the user habits and site requirements, uses a natural handwriting interface as well as human gestures. Two methods were used for symbol recognition, namely the state-of-the-art elastic matching algorithm and an Adaptive Resonance Theory neural architecture. The neural solution is proved to be better adapted to the cognitive nature of the problem and faster in both learning and test phases. Finally a novel attribute grammar permits the detection and subsequent correction of errors in the mathematical expressions.

Adaptive resonance theory Mathematical editor Handwritten symbol recognition
Attribute grammar Self-organized neural networks Elastic matching

1. INTRODUCTION

Although a better user interface was always a central point in the design and development of an information system, recently major emphasis is observed for a natural user interface in both research and commercial levels. The use of speech and handwriting are the reasonable candidates for such a natural, human-like interface. The systems of speech synthesis and recognition have made a substancial advance, especially in the case of isolated words.[1,2] On the other hand, the use of handwriting has not yet presented a feasible solution for a general system, despite its enormous scientific and economic interest. Nevertheless, the development of software tools based on handwriting is currently under way, in order to follow the recent hardware and software achievements, that are globally known by the name *pen-based computing*.[3]

The proposed mathematical editor is an example of such tools. It is based on the recognition of the introduced run-on discrete-time handwritten symbols and on the use of gestures that perform typical tasks of text processing, such as insertion, modification or deletion. Both aspects contribute to a system that follows the pen–paper paradigm.

The main drawback for the expansion of systems, using handwriting as a means of interaction with the computer, is the difficulty of providing a system able to recognize close to unrestricted handwriting. In the past several methods were proposed, based on the classical statistical or syntactical pattern recognition.[4] Although these methods can present a relative success for limited cases, the heuristic definition of the feature vectors and the non-cognitive nature of the classification techniques has not led to a satisfactory solution. Recently, several techniques based on neural network theory were proposed in order to overcome the afore mentioned drawbacks. The well-known general features of the neural networks such as parallelism, adaptability etc. make them good candidates for the problem of handwritten symbol recognition that has a close relationship with the corresponding human task. The majority of the proposed methods use the feedforward neural networks with the Backpropagation learning algorithm,[5,6] while some of them intend to provide models inspired from biological neural networks.[7] In any case, no major models were proposed for the case of on-line run-on discrete handwritten symbols.

In this paper a new neural network architecture is proposed for the recognition of on-line handwritten symbols. The underlying theory of the neural models is the Adaptive Resonance Theory,[8] that was developed by Gail Carpenter and Stephen Grossberg of Boston University. It forms one of the most appealing theories for unsupervised learning, based on a solid mathematical and biological background. Several models were proposed for the case of unsupervised learning, such as ART2[9] and Fuzzy-ART,[10] while new architectures based on the same principles were

† Author to whom correspondence should be addressed.

proposed for supervised learning (ARTMAP[11] and Fuzzy-ARTMAP[12]). On the other hand, since a symbol may be considered as a sequence of components, modules that can represent a sequence by a spatial pattern are necessary. At the proposed architecture, a module that follows the same basic neural theory is used, namely the STORE (Sustained Temporal Order REcurrent) proposed by Bradski *et al.*[13,14]

The vast bibliography on handwriting recognition confirms that no single recognition module can have an acceptable performance rate in a realistic system. Therefore, the context is normally used for the detection and correction of committed errors,[15] as is the case also in the human performance. In the mathematical editor a modified architecture is proposed for the alphanumeric text, based on the classical approaches of text recognition systems. The mathematical expressions form an especially interesting part, because of their structured nature and the bidimensional distribution of their elements. Few solutions were proposed in the literature for the processing of the mathematical expressions,[16,17] and to the best of our knowledge never in the context of detection and correction of errors. In this paper a new attribute grammar[18] is proposed that can efficiently and flexibly represent the mathematical expressions, for its subsequent use in error detection and correction.

Finally, the importance of the specifications in the mathematical editor should be emphasized if we consider that a useable system should be adaptable to the user habits and site requirements. The use of these specifications conditioned the design and development of the final laboratory prototype that was implemented. It should be noted that the proposed system still lacks features of a complete fully tested commercial prototype, although a considerable effort is being currently done in this direction.

In Section 2, the design of the mathematical editor is explained and the general hardware and software structure is presented. In Section 3, we describe the problem of on-line recognition of handwritten symbols, presenting the two approaches that were followed. A brief introduction to ART is made and the neural architecture is described. On the other hand, the main points of the elastic matching approach are given in order to serve for comparison with the neural architecture. The modules used for error detection and correction for both alphanumeric text and mathematical expressions are described in Section 4. Experimental results for the distinct modules of the editor are presented in Section 5, while the main points are discussed in the concluding section.

2. DESIGN OF THE MATHEMATICAL EDITOR

2.1. *The design criteria*

Personal productivity and office automation tools have shown an impressive expansion since the last decade, especially due to the enhanced ratio of price/performance and the wide use of personal computers and workstations. The preparation of documents, that may vary from simple letters to complex books, constitute one of the principal examples of this field. Specifically, the preparation and formatting of technical documents, that include alphanumeric text and mathematical expressions, are some of the main concerns for the engineering and academic communities.

A computer system, able to implement such a mathematical editor, should have the following features:[19]

- Interactivity, where the most important aspect is the man-machine communication, i.e. offering an easy way to handle the program, and a real-time, high quality visualization of the resulting document.
- An efficient internal structure of the information, for its further processing by the corresponding algorithms.

The solutions that have been given up to now mainly focus on just one of the above criteria. In concrete the text processors follow the WYSIWYG (What You See Is What You Get) paradigm, thus treating the mathematical symbols in a geometrical form, while the editors of the mathematical expressions are not integrated in the whole system. On the other hand, the text formatters, like T_EX,[20] have a strong symbolic internal structure and provide a high quality typesetting, but they do not possess the necessary interactive characteristics.

The lack of an adequate solution and the recent developments in hardware and software components that treat handwriting[3] enforce the need for a mathematical editor, whose main interaction would be by a natural means, such as handwriting.

The basic design criteria followed in the present editor are:

- User-friendly interface and full interactivity with the editor, as far as the data introduction, graphical representation and formatting are concerned.
- Use of the electronic pen as the principal means for communication with the editor. Human gestures are to be used for typical text processing operations, such as insertion, deletion or modification.
- Flexibility in the adaptation of the system to special site requirements concerning the processing capacity, the minimum error rate required as well as the minimum time requirements for a real-time processing.
- Flexibility in the decision, whether the system should be adapted to the user or inversely. Systems that cannot be adapted to some of the user habits result being unusable.[21] It should be pointed out that although the restriction of run-on discrete symbols may be important for the generic use of alphanumeric text, it does not constitute a serious drawback for the mathematical expressions.
- Use of established standards that contribute to an increased continuity and user familiarity.

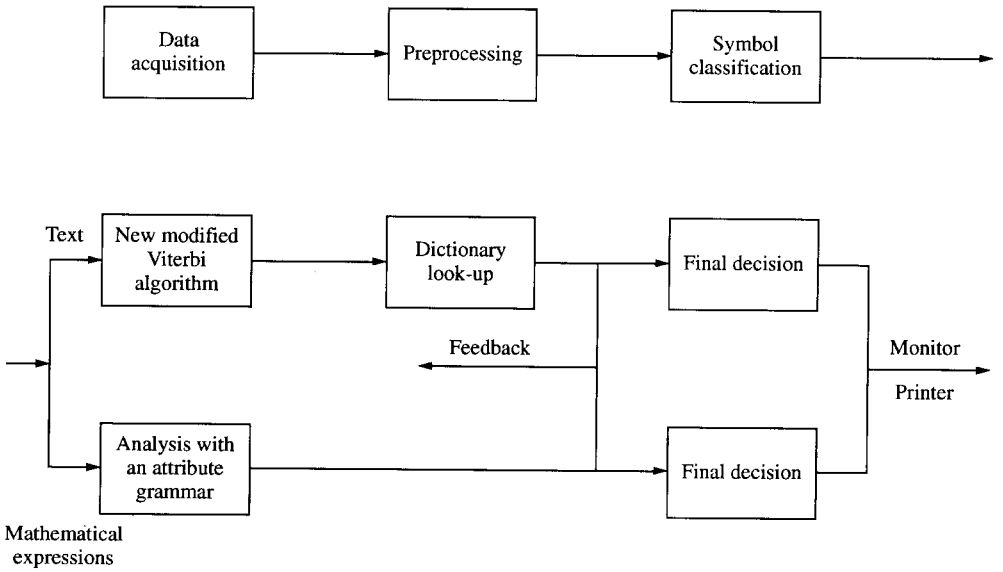


Fig. 1. The software structure of the mathematical editor.

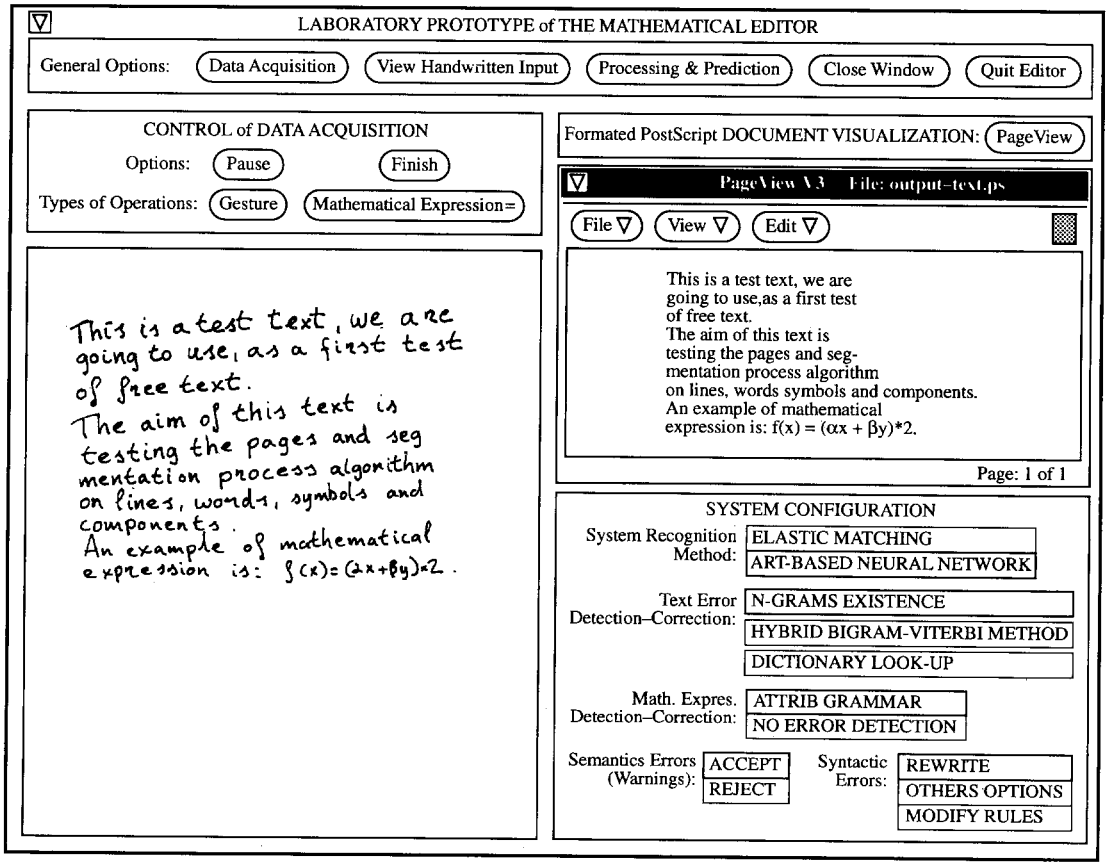


Fig. 2. A typical instance of the mathematical editor's interface.

2.2. *The structure of the mathematical editor*

According to the design criteria described in the previous subsection, the hardware system consists of a digitizing tablet, a workstation and a laser printer. All these devices should have enough power for graphics processing. Also all other input devices should have a marginal use.

The software structure of the mathematical editor is shown in Fig. 1.

It consists of a data acquisition and preprocessing module, followed by symbol recognition and error detection and correction. The final document is converted into the adequate format and processed by the T_EX package.

The aspect of the general graphical interface is based on the X-Windows system and is comprised of the following regions, shown in the Fig. 2.

- The data acquisition zone, where the data input is controlled and the digitized data are visualized in real-time, thus providing a digitized form of the symbols that are printed on the paper that covers the digitizing tablet. This region will become auxiliary, when the new technology of transparent digitizers is incorporated in the system.[22]
- The zone, where the final recognized and formatted document is presented.
- The configuration and messages zone, where the user may have an interactive control of the system.

A final issue to the design and implementation of the interface of the mathematical editor is the use of human gestures[23–25] as an integral part of it. Its use provides a path to a generalized interface based on the pen–paper paradigm. Besides the introduction of the data (text and alphanumeric expressions), most of the typical text processing operations such as addition, deletion or modification are performed using human-like gestures. This gesture-based interface, that was extensively tested is presented in detail in another paper.[26]

3. ON-LINE RECOGNITION OF THE HANDWRITTEN SYMBOLS

3.1. *Introduction and the elastic matching approach*

The module of symbol recognition described in this paper treats on-line run-on discrete handwritten symbols. The diversity of experimental conditions such as type of symbols, mode of data acquisition or type of processed documents give rise to a difficult task of performing a comparative analysis of the different techniques that were proposed in the literature.[27,28] Also, although in off-line character recognition common databases are being created for benchmarks, a similar procedure is much more difficult in the field of on-line recognition. Therefore in this work, a widely used technique, known as elastic matching[4,28,29] was chosen in order to be able to provide a comparative measure with the neural architecture.

The elastic matching approach is a method of pattern matching, that uses dynamic programming as a means to calculate the minimum possible distance between the test pattern and each of the *a priori* defined reference patterns. The nonlinear way of estimating the minimum distance compensates for any local changes and focuses on the general characteristics of the patterns. The same technique was widely used in the field of speech recognition and is known as Dynamic Time Warping (DTW).[30]

Although the elastic matching technique makes an efficient use of dynamic programming as an optimization technique, it still has many problems with the time required for a comparison of real complex patterns. In order to compensate for this problem, elastic matching was used as a part of a decomposed pattern recognition system or in other cases special hardware and VLSI was proposed for its wide-spread applications.[31,32]

3.2. *The ART-based neural network architecture*

3.2.1. *Overview of the architecture.* ART emerged as a theory in the process of understanding the self-organizing natural code formation.[8,33,34] It is based on the idea of competitive learning and follows the development of several basic modules, incorporated in all of the ART-based models. Each one of them performs a generic task and there is sufficient evidence that they are embedded in different neural architectures in the brain.

These basic modules and their corresponding functions are:

- *Instar*, that performs a codification of the input pattern to an output node, through a set of adaptive weights (many-to-one relationship).
- *Outstar*, the reverse scheme from the instar. It permits the weights, that come out of the source node, to learn the presented pattern.
- *On-center off-surround field*, where the nodes compete among them. Each node excites itself (on-center) while inhibiting the surrounding ones (off-surround). Thus, a competition takes place resulting in a winner node at the simplest case.

The design and characteristics of the architectures based on these modules permits them comply with the following two principles.

- The stability–plasticity dillema, i.e. how a network can be stable, while at the same time being able to learn new incoming information.
- The noise–saturation dillema, i.e. how the noise can be suppressed without allowing large input signals to saturate the neurons.

Within the ART framework, several models have been proposed for the unsupervised or supervised classification of binary or analog patterns, or even time-varying patterns.[33] Recently, the concepts of

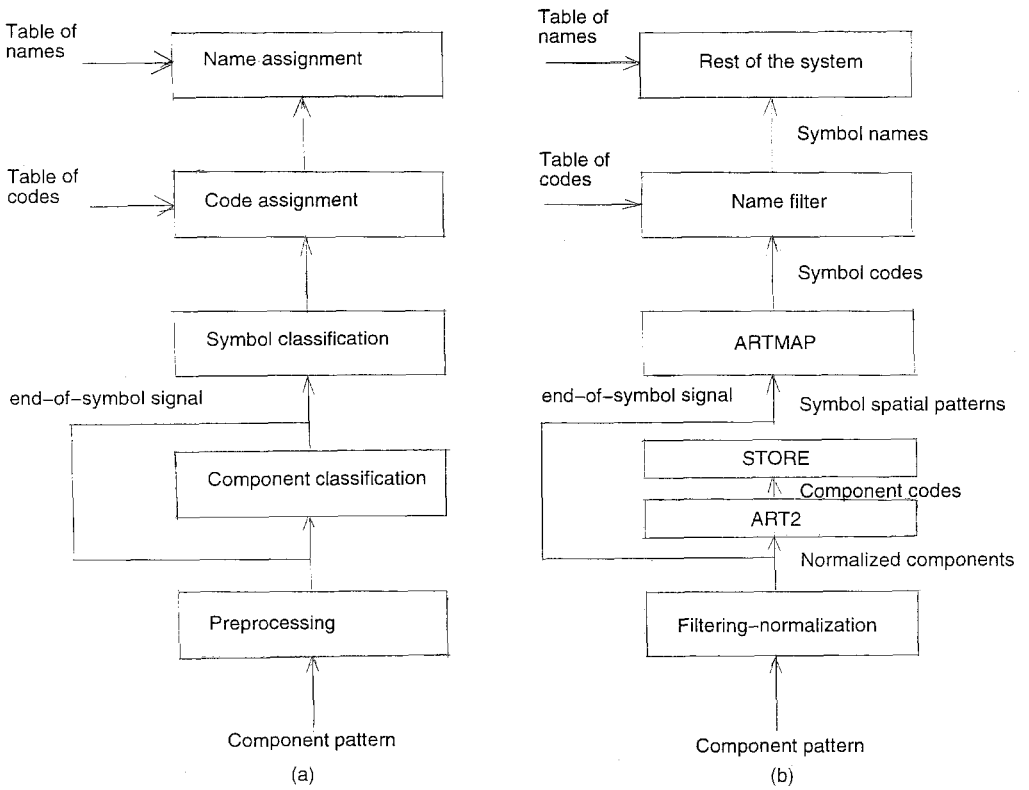


Fig. 3. (a) A block diagram of the neural architecture. (b) The implemented neural architecture.

fuzzy sets were incorporated in new architectures, resulting in a new powerful tool.[10,12]

In the proposed architecture a handwritten symbol is represented by the sequence of its components, that are extracted using the pen-lift signals. Each component is represented by a length-normalized vector of the equidistant samples of the angular velocity, i.e. for any discrete time instant nT the samples will be: $v(nT) = arctan(\Delta y/\Delta x(nT)$, where $\Delta y = y(nT) - y(nT - T)$ and $\Delta x = x(nT) - x(nT - T)$. This representation avoids a heuristic feature detection, which is present in most of the already proposed character recognition schemes and approximates a handwriting generation model proposed by Plamondon.[35] Presently, it represents an engineering approach oriented to a future exploitation of handwriting generation models in the recognition process.

Based on the previous representation of the handwritten symbols, the recognition problem can be decomposed to the following functions, that are shown in Fig. 3.

(1) Data acquisition and preprocessing, where the handwritten input is enhanced and segmented to symbols and components, the digitized representations of the symbols are calculated, and a normalization of the component vector length is performed.

(2) Unsupervised classification of the components.

(3) Representation of the sequence of component categories of each symbol by a spatial pattern.

(4) Assignment of codes and names to the symbols, through a supervised classification module.

As shown in Fig. 3, all the fields that are embedded in the implemented neural hierarchy come from the set of models that have been proposed in the framework of ART, thus giving an architectural consistency to the proposed solution.

3.2.2 *Unsupervised classification of the symbol components.* The ART2 neural network[9] treats analogue inputs of fixed length. Therefore a linear elastic matching technique was necessary to normalize the length of the patterns which represent the components. The main functions of ART2 that follow the general guidelines of ART are:

- Suppression of the noise that is below the quenching threshold θ.
- Contrast enhancement and normalization of the input pattern.
- Codification of the pattern in a node of F_2 level through an *instar* and an *on-center off-surround* competitive network.
- Learning of the expectancies through an *outstar*.
- Reset in the case that the distance between the input pattern and the learned expectancy is above the vigilance parameter ρ.
- Choice of one of the existing or non-committed nodes, using a winner-take-all rule thus concluding to a resonant state.

In our system, a run-on discrete symbol is considered as a sequence of components, i.e. a series of traces between two successive pen lifts. After the codification of a component in a node of the F_2 field of ART2, the rest of the components of a symbol enter the ART2 module and get classified similarly at a node of the same field. The symbol is composed of the ordered sequence of the component category nodes, as indicated by the end-of-symbol signal that derives from the segmentation preprocessing stage.

3.2.3. *Representation of the component sequence by a spatial pattern.* It is obvious from the above formulation, that the problem of representation and processing of a temporal sequence arises. Although several syntactical, engineering or dynamic systems solutions were proposed,[36] it seems psychologically more valid to adopt a neural theory proposed by S. Grossberg that considers the problem as one of parallel processing of STM spatial patterns.[37,38] According to Grossberg's theory, the formation of the spatial patterns has to comply to the following principles:

- Invariance, i.e. when the new sequentially presented patterns are stored, the temporal order codes of the previous patterns should remain invariable.
- Partial normalization, i.e. the limited capacity of the STM should be reflected in the spatial pattern.

This way, psychological data of free recall can be explained.[39] In these experiments subjects recall the first items (primacy), final ones (recency) or both of them (bowing), when presented with a sufficiently long list of items.

Recently, a series of neural models have been proposed to implement the above principles[13,14] or incorporate them in more complex architectures.[40,41] In our architecture the most recent version of the STORE model proposed by Bradski[14] was implemented and used.

According to its two-level structure, the bottom layer registers the inputs and due to its competitive nature it represents the later entries with larger STM activations. On the other hand, the top-layer nodes track the bottom-layer activities, preventing an erosive influence during the presentation of new inputs. In order to take into account the repeated items, a winner-take-all preprocessor and an array structure were proposed. We should note that a similar solution was proposed in.[40,41]

3.2.4. *Supervised code and name assignment to the symbols.* In the STORE network a spatial pattern was obtained, which stores the item and order information of the sequence of components that represents the handwritten symbol. The final step of the neural hierarchy is the codification of this spatial pattern. An unsupervised scheme proposed in[13] and composed by a series of *outstars* could not be used in the present scheme, since the correspondence between sequence of components and symbol codes is not one-to-one, thus giving rise to a conflict between spatial STORE patterns and symbol codes. The solution to the above problem is comprised of an analogue ARTMAP network, which combines the unsupervised nature of an ART2 model with a prediction error tracking procedure.

The original ARTMAP network[11] is composed of two ART1 networks, that are connected through an inter-ART associative memory module and a controlling scheme that regulates the learning and data stream. In ARTMAP the pattern to be classified is presented to the ART_a module, while the corresponding code is presented to the ART_b. After the usual unsupervised codification at both ART modules, the resulting patterns are matched at the inter-ART module, resulting to an increase of the adaptive weights, when a correct prediction is made. On the other hand, if a mismatch is produced, an inter-ART reset is triggered that increases the vigilance parameter ρ_a of ART_a just until a new inter-ART match is produced. This process of match tracking permits the creation of finer categories that correspond to rare but important inputs.

In our architecture the analogue version of ARTMAP is used, as proposed in,[42] in order to account for analogue inputs. In that network the ART1 modules were substituted by those of ART2. The patterns introduced to ART_a are spatial, produced by STORE, while the inputs to ART_b correspond to all symbol codes considered. In the learning phase pairs of component sequences and symbol codes are presented, while at the prediction phase the patterns that come out from the ART_b module are considered as the codes generated by the unknown symbols. Finally a name is assigned to the resulting code for its posterior processing.

4. ERROR DETECTION AND CORRECTION

In the mathematical editor two different types of data are treated, i.e. alphanumeric text and mathematical expressions. The distribution of these two types of information is fundamentally different, because on the one hand the typical Latin-based languages, that are treated here, are written from left to right, while on the other hand the mathematical expressions may be written in any form in the two-dimensional (2D) space. This property leads to a different approach in the segmentation phase, as well as at the error detection-correction phase, that is described below.

4.1. *A hybrid architecture for alphanumeric text*

The information used in the detection and correction module for the alphanumeric text is mainly of low level and consists of the n-gram frequencies and the dictionary-based one. The higher level information, i.e. the syntactic and semantic ones, cannot yet be efficiently used because of the problems encountered in the natural language field, although recently some attempts

were made for the incorporation of statistical measures of them.[43,44] We should also note that the existing techniques for error detection and correction are computationally costly in terms of processing time and storage space. Therefore and according to the design specifications of the mathematical editor, the final user can configure the system and use some, if any, of these modules, depending on the site requirements.

The first module consists of an algorithm that detects the errors, based on the information of the n-grams.[45] Although normally only the information of the n-gram validity of a specified language is used, the appearance frequency of the n-grams was exploited in our system. We should note, that the n-grams with $n>3$ do not contribute significantly to the error detection, while raising exponentially the complexity, and therefore were not included in the present implementation.

In the next module a Modified Viterbi Algorithm (MVA)[46] was implemented. This technique, widely used in the communications field, computes the best estimate of the characters of a word, based on the probability of correct recognition of a character and the transition or bigram frequencies. The MVA follows the general VA algorithm, with the difference that it only considers the d most probable candidates for each character. The simplicity as well as the existing hardware implementations of MVA make it a useful module in any recognition system.

In the complete configuration of our system, the n-gram validation module precedes the MVA thus reducing significantly the computational cost. Another enhancement consists in the use of the Raviv parameter F[47] that regulates the contributions of the classification and the context information in MVA. A good choice of the parameter F, depending on the characteristics of the data and the experiment conditions, may enhance significantly the error detection–correction capacity of this module.

In the final step a simple dictionary look-up was implemented, in order to check the validity of the recognized words, resulting in a feedback to previous modules and a consideration of other character candidates. The final user can configure this module, by choosing the appropriate size and type of dictionary, according to the compromise between computation time and efficiency. Currently, a more compact and storage efficient dictionary look-up technique is being implemented, following the one proposed by Wells *et al.*[48]

4.2. *An attribute grammar for mathematical expressions*

The mathematical expressions, contrary to the alphanumeric text of natural language, can be qualified as well-defined and structured, permitting variations that are easily interpreted by the context. The definition of the mathematical "language" and the nature of its users allowed a small margin of non-structured evolution. On the other hand, a large amount of different mathematical expression can be produced, thus having to reach a compromise between their free construction and the use of complementary symbols, such as parentheses, brackets etc.

The analysis of mathematical expressions has been treated in different occasions,[16,49,51] but always within the framework of handwriting recognition. In our system, their description is made as a complement to a general recognition scheme and in concrete for the error detection.

Traditionally, knowledge had been represented by rule-based systems[52] that have been proven inflexible. Especially in the context of the mathematical editor, a user follows certain habits that have to be taken into account for efficient tool use. Therefore a flexible but structured representation of the mathematical expressions has to be employed. The attribute grammars[18] constitute a solution to the above requirements, since they have been proven equivalent to the rule-based systems also demonstrating advantages in the procedural aspects as well as in their construction.[53,54]

The attribute grammars form a class of context-free grammars and are defined by the following 4-tuple: $G=(V_N, V_T, P, S)$, where V_N is the set of the non-terminal symbols, V_T is the set of the terminal symbols, S is the start symbol and P is the set of production rules.

Its peculiarity consists in the use of semantic rules α_i in addition to the syntactic ones that handle a set of attributes, synthesized $A_0(X)$ and inherited $A_1(X)$, that are associated to each symbol $X\in(V_N\cup V_T)$. Therefore characteristics associated with the grammar symbols can be used, providing a much more flexible and robust use of the grammar.

In the case of the mathematical expressions, the non-terminal set V_N consists of higher or more elemental structures, such as ⟨expression⟩, ⟨sumterm⟩ or ⟨integterm⟩. The set of terminal symbols V_T contains all considered symbols. The attributes $A(X)$ of the grammar symbols have to determine their position with respect to the rest.

The coordinates $(x_{min}, x_{max}, x_{med}, y_{min}, y_{max}, y_{med})$ of the structures are extracted during the segmentation process and completely characterize the symbols.

In the set of production rules P, the syntactic rules represent the decomposition of the higher level expressions, while the semantic rules describe the relative position of the structures in the (x, y) coordinate space. Finally, the rules of generation of the synthesized attributes have to be provided in terms of the attributes of the lower levels.

As an example of the rules of the attribute grammar, a division term is presented:

Syntactic rule:

$$\langle divterm\rangle \Rightarrow \overbrace{numerator\ \langle expression\rangle}^{S_1}\ \overbrace{\langle divop\rangle}^{S_2}\ \overbrace{\langle expression\rangle}^{S_3}\ denominator.$$

Semantic rule:

$$x_{min}(S_1) > x_{min}(S_2),\ x_{max}(S_1) < x_{max}(S_2),\ y_{min}(S_1) > y_{max}(S_2)$$

$$x_{min}(S_3) > x_{min}(S_2),\ x_{max}(S_3) < x_{max}(S_2),\ y_{max}(S_3) < y_{min}(S_2).$$

Synthesized attributes:

$$x_{min} = \min(x_{min}(S_1), x_{min}(S_2), x_{min}(S_3)),$$

$$y_{min} = \min(y_{min}(S_1), y_{min}(S_2), y_{min}(S_3))$$

$$x_{max} = \max(x_{max}(S_1), x_{max}(S_2), x_{max}(S_3)),$$

$$y_{max} = \max(y_{max}(S_1), y_{max}(S_2), y_{max}(S_3))$$

$$x_{med} = 0.5*(x_{min} + x_{max}),\ y_{med} = y_{med}(S_2).$$

A lexical analysis of the input and a parsing phase, where both types of rules are checked, takes place. When an inconsistency is encountered between the input and the decomposition rules, the system sends a message to the user notifying him about the error. Two types of error are used in the system:

- A warning about the inconvenience of the form of entering the mathematical expression. In this case, the user may accept the recommendation and adapt himself to the system or on the contrary maintain his habits and indicate to the system his preferred way of writing the mathematical expression. In the latter case, the system has to adapt its rules (mainly the semantic ones), when there are not fundamental violations of the grammar. The implementation of the grammar, through the standard tools *lex* and *yacc* of Unix, permits a comfortable modification of the rules, although with an increase of the system complexity.
- A message about a basic error. In this case the source of the error has to be identified, among an eventual negligence of the user or an error in the preprocessing and recognition modules.

The formulation of the problem offers the flexibility required by the design specifications, that refer to the extension or modification of the writing norms. Also it offers a high capacity of error detection and eventual correction. On the other hand, the high computational cost should be mentioned, which in turn can be handled by a medium-range scientific workstation.

5. EXPERIMENTAL RESULTS

5.1. *The experimental environment*

The laboratory prototype was developed in a Unix-based environment of a network of workstations, using the following standard software and hardware modules, in order to comply with the design requirements.

- The main workstation was a Sun 4/370, with a monochrome monitor of 19″ and 1152 × 900 resolution. The same system was tested in other Sun-compatible workstations or X-terminals with equivalent performance.
- An electrostatic digitizing tablet Calcomp 23120, with an active area of 12 × 12″ and a resolution of 1000 points per inch, connected to a serial port of the workstation while the sampling velocity was set to its maximum value of 125 points/s. A laser printer HP Laserjet IIIp was used as output device.
- The operating system was the Unix BSD derived SunOS 4.1.1, the programming language was ANSI C, and the windowing environment was X-Windows with the Open Look style, although working versions using Motif were also developed.
- The formatting package was L^AT_EX, and the visualized and printed output was obtained in a PostScript format, using the PageView, gv and NeWSprint packages.

The chosen users were engineering undergraduate and graduate students and were required to write on different types of paper, attached to the digitizing tablet. No special requirements were imposed on the handwriting style, besides the restrictions of the divisions of each page and the run-on discrete symbols. In a first phase, a user was required to fill a page for each symbol, that was going to be used, and later he had to write a predetermined text that was validating the segmentation rules. Finally, the user could proceed to input a document of his own choice.

The acquired data were stored in a database, that was indexed according to the type of paper, the type of text and the username, for its posterior processing and analysis. The interactive use of the mathematical editor was also tested.

5.2. *Preprocessing and segmentation*

The main preprocessing techniques involve the bottom-up segmentation, and the normalization of the length of the vectors that represent the components of each symbol.

Each symbol component is detected by two consecutive pen lifts. The nature of the ART2 neural network requires a fixed length of the input vector. Therefore the linear elastic matching algorithm was employed, that corresponds to a linear interpolation of the points that form a component. A length of 25 was found experimentally sufficient to represent any type of component.

In the segmentation phase, two different techniques were used for alphanumeric text and mathematical expressions. Both techniques are heuristic and have to be adapted to every particular user.

For the alphanumeric case the fundamental units are components, symbols, words and lines. Using a bottom-up approach, each symbol component has to be recursively assigned to the superior level units or equivalently a change has to be detected for each symbol, word or line. The following criteria were used:

- A combined change of symbol, word and line is detected when: $d_x < -D_x^{same-line}$ and $d_y < -D_y^{same-line}$

- A combined change of symbol and word occurs when the previous criterion is not fulfilled and additionally: $d_x > -D_x^{same-word}$.
- Finally a change of symbol occurs, when the previous tests fail and additionally: $d_x > D_x^{same-symbol}$ and $d_x \leq D_x^{same-word}$.

In the above criteria, the following variables for the distance between two successive symbol components were used:

$$d_x = x_{min}^{present-component} - x_{max}^{previous-component}$$
$$d_y = y_{med}^{present-component} - y_{med}^{previous-component}$$

The various threshold variables were estimated from the corresponding histograms of the pre-determined texts, that the users were required to enter. Some indicative values for a typical user in our experimental environment are: $D_x^{same-symbol} = 200$, $D_x^{same-word} = 200$, $D_x^{same-line} = 1000$, $D_y^{same-line} = 500$.

The mathematical expressions were treated in a way similar to the one proposed by Wang *et al.*[17] since the relation of the symbols or expression blocks has to be detected, i.e. whether they belong to the same line or one of them is a subindex or superindex of another one. The employed technique was proved to be sufficiently robust, although its statistical and heuristic nature should be emphasized.

5.3. *Recognition of the handwritten symbols*

5.3.1. *Implementation of the elastic matching approach.* As it was explained in Section 3, the elastic matching algorithm was implemented, in order to serve as a comparison reference for the novel neural architecture. Therefore a brief description of the method and the results are given below.

In the initial phase, the contour of each symbol was obtained, since our intention was to provide a representation that could be used even for off-line recognition, and thus expand the comparison scope. The contour of each symbol was then represented in the same way as in the neural approach.

Since elastic matching belongs to the general category of pattern matching techniques, the reference patterns for each class should first be defined. The data set for the estimation of the reference patterns contained a number of 100–150 symbols that were introduced by each user in the first phase of the experiments. Using a standard method employed in Dynamic Time Warping, the distance d_{ij} of any two symbols i and j was calculated and the number of neighbors n_i that have $d_{ij} < T$ was estimated, where T is a threshold that reflects the distribution of the patterns. The reference pattern for each class is chosen as the one with the greatest n_i.

In the classification phase, the distance between the test pattern and each reference pattern is computed and the nearest neighbour rule is applied for the final categorization. The distances reflect the confidence of classification of the test symbol to each candidate class.

The results of the elastic matching approach confirmed those reported in the literature for similar experiments[4] and were in the range of 90–99%, depending on the quality of the input data. We should remind here the increased complexity of both learning and test phases, especially when the number of classes is large enough, as is the case in the mathematical editor.

5.4. *Analysis of the performance of the neural hierarchy*

In order to better evaluate the performance of the neural architecture, a limited number of symbols was used in the first stage of the experiments. The symbol samples were chosen in order to form a statistically sufficient data set, while incorporating the same or similar types of components in different symbols. In concrete the data set was composed of 132 "f" (two components), 130 "n" (1 component), 193 "," (1 component) and 110 "ñ" (2 components). The digitized symbols used in this phase are shown in the Fig. 4.

When the vectors that represent the symbol components were introduced to the ART2 network, with 25 F_2 nodes, a vigilance parameter $\rho = 0.8$ and a quenching threshold $\theta = 0.1$, the symbols formed the corresponding clusters in the F_2 field. The resulting errors were 8.55% in the strict sense and 5.08% in the wide sense. The strict sense of the errors refers to the cases where conceptually different components as "~" and "—" were classified in the same node, while these errors were discarded in the wide sense. We should note that only three presentation cycles of the input patterns were necessary for the stabilization of the network, obtaining later direct access to the category nodes. An example for the different phases of processing in ART2 is shown in Fig. 5.

In the next step the influence of the choice of the network parameters was studied, by changing their values to $\rho = 0.97$ and $\theta = 0.0$. Although the vigilance parameter ρ, that regulates the coarseness of the categories, was significantly increased, the number of the category nodes remained constant. This fact indicates that the input patterns are sufficiently separated. On the other hand, the diminishing of the quenching threshold θ resulted in a more compact distribution of the input patterns in the category nodes. The meaning of this result is that significant signal and not the noise was removed at the level F_0. In general, the optimum selection of the parameters ρ and θ, being very important for the fine tuning of the ART2 network, can be made from a previous statistical study of the input patterns.

The implementation of the conversion of the component sequences to symbol spatial patterns was made using the recency version of the STORE model.

The spatial patterns, that represent the sequences of components, are fed to an analogue ARTMAP network. The first 100 samples of each symbol were used for learning, while the rest of them were used for prediction. The results, presented in Table 1, show

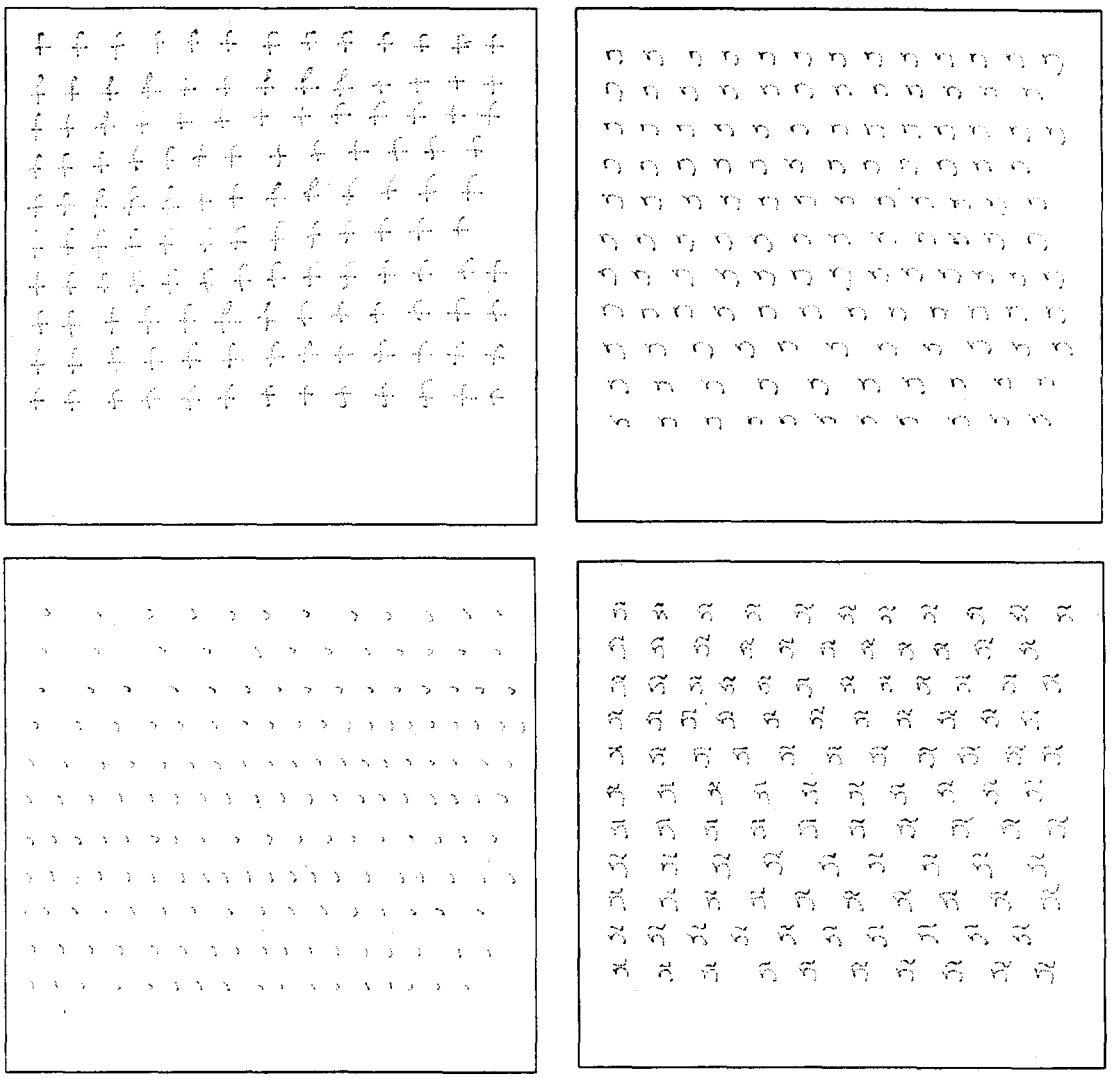


Fig. 4. A sample of the handwritten run-on discrete-time symbols.

Table 1. Results of the symbol classification for $\rho = 0.97$ and $\theta = 0.0$

Symbol	Correct	Erroneous	Uncertain
f	29	0	2
n	31	0	0
'	78	0	6
ñ	12	0	3

93.17% correct classification, 6.83% undecided cases and no incorrect predictions. The parameters of the implemented analogue ARTMAP networks were: $\rho_a = 0.985$, $\rho_b = 0.99$ and $\theta = 0.0$.

If we attempt a qualitative analysis of the noncorrect cases, two main reasons are detected. In the first group of cases, the symbol components were erroneously classified at the F_2 nodes of the ART2 network, which mainly grouped components of other symbols. This ART2 erroneous classification was due to the following factors:

- The bad quality of several introduced symbols, because of user negligence or problems of the digitizing process.
- The inefficient representation of the components by the measure of angular velocity. It is obvious, that in order to account for the enormous variability of handwriting, other features have to be incorporated in the input vector that represents the symbol components. In such a case, the input vector to the ART2 network should contain subvectors of different nature, such as range and sensitivity. It was proved,[55] that the ART2 network is not able to handle properly patterns of this type. Therefore a Multi-ART architecture[55] should be used, instead of the ART2 network, thus handling properly the heterogeneous input patterns, although there is an additional cost of memory storage and processing time.

The other class of undecided cases was produced by the fact that new sequences of symbol components were presented at the prediction phase, which were not

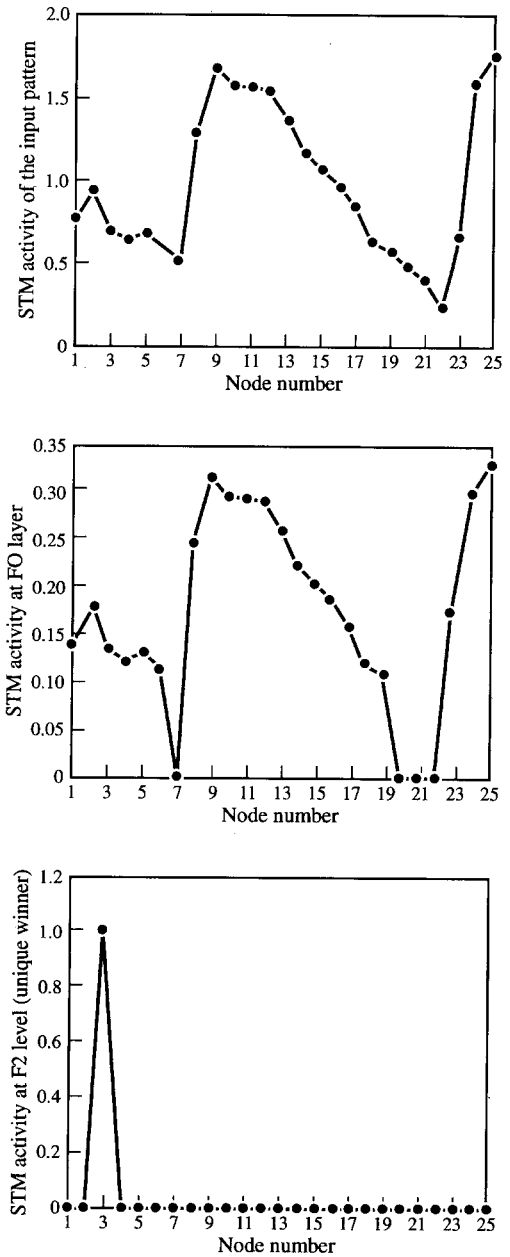


Fig. 5. STM activities at the different fields of ART2, for a symbol "n"

learned before. An obvious solution to this problem is to increase the data set that is used in the learning phase, in order to incorporate more representative cases of the symbols used.

In order to measure the generalization capability of the neural hierarchy, the same symbols written by another user were introduced to it, although no data from the second user were included in the learning set. It was confirmed that symbols, written in a similar way by the second user, had an equivalent prediction rate (140 correct predictions, 1 erroneous and 45 uncertain cases for the symbol "n"). On the contrary, symbols

Table 2. Results for two users and $\rho = 0.97$, $\theta = 0.0$, $\rho_a = 0.99$, $\rho_b = 0.99$

User	Symbol	Correct	Erroneous	Uncertain
1	f	23	3	5
1	n	31	0	0
1	˙	78	0	6
1	ñ	14	0	1
2	f	65	1	2
2	n	35	1	0
2	˙	132	2	14
2	ñ	62	1	2

Table 3. Results of the symbol classification for $\rho = 0.97$ and $\theta = 0.0$, for the new data set

Symbol	Correct	Erroneous	Uncertain
+	15	35	18
Σ	27	9	3
0	43	1	0
8	34	8	1
H	22	5	1
P	12	14	3
X	13	26	0
h	32	1	0
p	6	19	2
x	16	13	0

like "f", written in a totally different way, were classified incorrectly in their majority. This lack of generalization is confirmed by the fact, that an increase of ρ_a to 0.99 converted the incorrect cases to uncertain ones.

If we repeat the same experiment, including the same amount of symbols written by the second user in the learning set, the results shown in Table 2 are obtained:

The results shown in Table 2 confirm the robustness and flexibility of the neural architecture.

The last experiment was made in one of the most difficult sets of symbols, since it includes:

- Symbols, that scan many different types, like lowercase characters (h, p, and x), uppercase characters (H, P, and X), digits (0 and 8) and mathematical symbols (Σ).
- Symbols of the same form but of a different size (x and X, or p and P).
- Symbols, that contain repeated components in the representation of the angular velocity (+ and x).

The results, shown in Table 3, indicate that the neural architecture is still robust, although the defects mentioned above are clearer in this worst-case experiment. In any case, errors due to size differences could be seen as indicators of a robust system that is invariant in size. Errors of this type are detected and corrected by the use of the context information, as is the case with the human readers.

5.5. *Discussion on the symbol recognition problem*

The two techniques of symbol recognition, employed in this work, result in similar recognition rates, al-

though they come from different disciplines. If we look closer at both techniques, we can observe that both are based on pattern matching and use some kind of nearest neighbour rule for the final decision. Besides that, the neural architecture can be thought of as a combination of unsupervised clustering (ART2), representation of sequences (STORE) and supervised classification (analogue ARTMAP).

The main differences between the elastic matching approach and the ART-based hierarchy lie in their different nature. The fact that ART has strong biological and psychological foundations makes the neural architecture more plausible for a cognitive task such as the handwritten symbol recognition. On the other hand, all the general advantages of the neural-based techniques, like their high parallelism, apply here.

The experimental study of both techniques reveals that the processing time and the storage requirements are better for the neural architecture, in both learning and prediction phases. The direct access to the category nodes of the ART networks greatly reduce the processing and storage requirements.

With respect to the comparison of our architecture with other neural networks:

- The design of the architecture includes three separate stages of unsupervised classification, conversion of temporal to spatial patterns, and supervised classification. To the best of our knowledge, there are no other models that can perform the same functions and at the same time belong to a unique class of models. Therefore it is not possible to perform such a direct comparison.
- There are not any neural architectures that attack the problem of on-line run-on discrete characters. Only two SOM-based (Self Organizing Maps) solutions of the dynamic cursive handwriting[56,57] were reported, that are not directly comparable to our architecture that treats run-on discrete characters. In any case, the SOM approaches cover only the unsupervised phase of the allograph clustering, being comparable to our ART2 solution. The main disadvantage of SOM with respect to ART2 is the artificial way of reducing the learning rate, thus partially violating the plasticity-stability dilemma.
- As far as the supervised part of the architecture is concerned, an equivalent solution could include a Multilayer Perceptron with the backpropagation learning algorithm.[5] As cited, previously[58] the corresponding ARTMAP module requires much fewer cycles for similar performance in off-line character recognition, being closer to the neurobiological data and without showing problems of local minima. Our partial experiments confirmed the above comparison.

5.6. *Error detection and correction*

5.6.1. *Results for the alphanumeric text.* A non-exhaustive experimental study was performed for the error detection and correction in the case of alphanumeric text. The statistical information was extracted from a large corpus of the greek language that was available at the National Technical University of Athens, in the context of the European Program Eurotra. Similar experiments were performed for the Spanish language, using data from a reduced corpus, with equivalent results.

In the first phase, texts from journals were used for the study of the performance of the module that detects errors, based on the trigram frequencies. New texts were created, that incorporated randomly produced errors using the confusion matrix of the recognition system.

When erroneous text, considered in the creation of the corpus, was introduced to the trigram frequencies module, the following ranges of detection rates were obtained:

- Detection of correct words: 97.47–98.87%.
- Detection of erroneous words: 70.52–89.39%.
- Average of correct detection: 89.10–96.40%.

When the tested texts had not been considered in the creation of the corpus the corresponding ranges were: 75.10–75.66%, 91.32–95.95% and 77.45–78.46%.

From the above results, we can observe that the trigram module is very efficient in error detection, although its performance depends widely on the corpus used. We should also remember that the processing time is rather low while the required storage is high.

Although both the trigram frequency and the VA modules can be configured to work independently, it was found that a combination of both modules is much more efficient than each one separately. The hybrid system, called here NMVA (New Modified Viterbi Algorithm), consists of the trigram module followed by an MVA one. A variation of the NMVA that uses the parameter F of Raviv was also tested.

It is known that the VA has a rather good error correction rate, when using the bigram frequencies, but on the other hand it favours excessively the most common bigrams, thus introducing new errors. A good measure of the efficiency of VA is defined by:

$\Delta = L_1 - L_2/L_1 * 100$, where L_1 is the percentage of the erroneous words after the classifier, and $L_2 = L_{21} + L_{22}$, where L_{22} is the percentage of erroneous words corrected and L_{22} is the percentage of correct words converted to erroneous.

The range of the index Δ for an MVA module with $d = 5$, depending on the quality of the texts varied among 56.60% for words with a single error, 26.60% for words with three errors and 0% for words with more than three errors. Also, the index L_{22} of the errors introduced by MVA varied between 0.50 and 8.00%.

In NMVA just the words, considered as erroneous by the trigram module, were fed to the MVA. Therefore, NMVA considers much less cases, thus reducing greatly the processing time. On the other hand, the index L_{22} was much less than in MVA, as expected. A

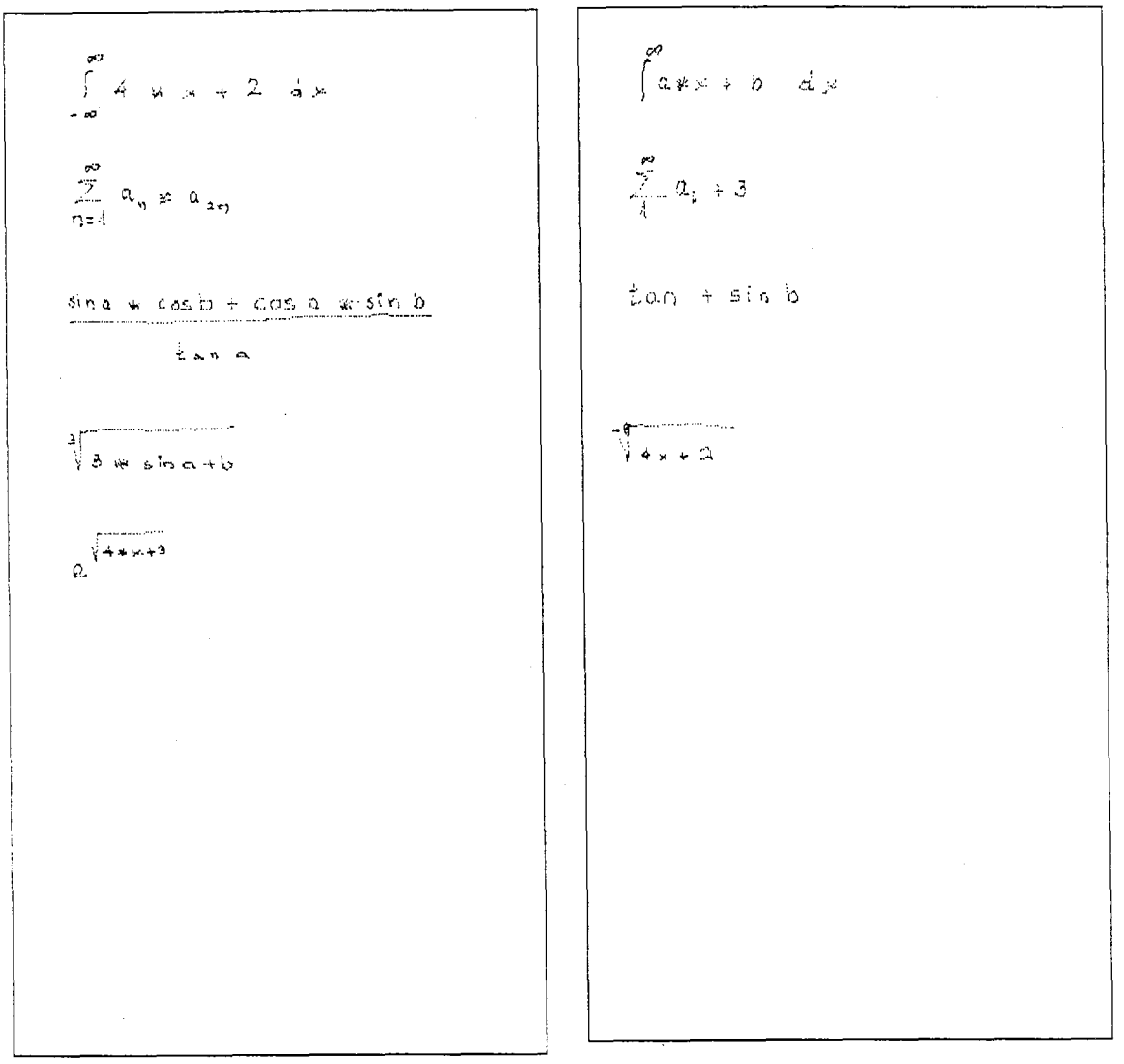
general enhancement of 25 to 35% was observed, compared to the MVA module.

Finally, the parameter F of Raviv, that regulates the influence of the classifier and the statistical knowledge in MVA, was studied. Its optimal selection depends on the quality of the symbols to be recognized, the completeness of the corpus and the efficiency of the classifier. Based on the analysis of the above factors, the introduction of the parameter F increased the efficiency of error correction in a range of 20 to 30%, as compared to the MVA.

5.6.2. *Results for the mathematical expressions.* Since the alphanumeric text and the mathematical expressions are treated in a different way in the preprocessing and the error detection modules, the user has to provide a specific gesture that indicates, when a mathematical expression begins or ends. Also, in order to cover the case of the mathematical expressions within a text, like in T_EX, another gesture is used to indicate this case.

The attribute grammar, used in the implemented laboratory prototype, consists of 72 rules, that treat an extensive range of mathematical structures, such as algebraic and trigonometric expressions, or integrals. In any case, its extension in order to cover other operations, like matrix, sets etc. is not complicated, due to the flexible way of representation of the knowledge.

The recognized symbols together with their synthesized attributes are fed to the system, for its posterior lexical analysis and parsing. As an example of the performance of this module, two groups of mathematical expressions with semantic or syntactic errors were introduced, as shown in the Fig. 6.



(a) (b)

Fig. 6. Mathematical expressions with (a) semantic errors and (b) syntactic errors.

The system reacts in the predicted way, issuing the following error messages or warnings to the user:

Expression 1: Error in rule 9. Compilation O.K. The inferior limit should be written on the left of the integral.

Expression 2: Error in rule 11. Compilation O.K. The inferior limit of the sum should be written within the range of the coordinate x of the sum symbol.

Expression 3: Error in rule 21. Compilation O.K. The division line should cover both terms (numerator and denominator).

Expression 4: Error in rule 30. Compilation O.K. The root order should be written within the range of the x coordinate of the root.

Expression 5: Error in rule 31. Compilation O.K. The root symbol should cover all of its terms.

Expression 6: Compilation error while processing *liminf*. The inferior limit does not exist.

Expression 7: Compilation error while processing *suminf*. The inferior limit should have the form ⟨variable⟩ = ⟨expression⟩.

Expression 8: Compilation error while processing *tan*. The function tan does not have arguments.

Expression 9: Compilation error while processing *rootord*. The root order is a negative number.

In the first group of five expressions, a semantic rule was violated and therefore a recommendation warning was sent to the user. If the user does not accept the recommendation, the system modifies the corresponding semantic rule in order to adapt itself to the user habits. On the other hand, the message issued for the group of expressions 6–9 is mandatory, since a syntactic rule was violated. In that case, either the user has to rewrite the expression or the system has to locate its internal error.

6. CONCLUSIONS

A novel mathematical editor, that uses the pen-paper paradigm was introduced in this paper. The inexistence of such tools that focus on all aspects of graphical representation, flexible internal structure and natural interface, makes the proposed mathematical editor an important tool in the field of pen-computing. Special emphasis was put on the flexibility of its configuration in order to adapt to the site requirements and the user habits. Also an important element is the inclusion of human gestures that permit the user to perform naturally typical operations, such as deletion, modification or insertion of text.

The run-on discrete handwritten symbols are recognized using a new neural architecture, that is based on the mathematically solid and biologically plausible Adaptive Resonance Theory. The experimental results show that the neural architecture is better in terms of processing-time and memory requirements than the state-of-the-art elastic matching algorithm. Besides the general advantages of the neurally inspired methods, the proposed neural scheme is closer to the natural cognitive task of handwriting recognition. In this direction, the on-going research intends to include a model of handwriting generation as a candidate representation of handwriting.

Finally, error detection and correction techniques were proposed for both alphanumeric text and mathematical expressions. In the latter case, a new attribute grammar was proposed that is flexible enough to represent the knowledge of the mathematical structures and efficiently detect errors.

Acknowledgements—We would like to thank the members of the Neural Networks Group (J. M. Cano, J. F. Díez, C. García, J. Maroto, G. Méndez, L. J. Miguel, A. Muñoz, J. Pérez, F. J. Díaz Pernas, E. Sánchez, E. Zalama) for their contributions to the preparation of this paper. Special thanks to José Luis Contreras-Vidal, Department of Cognitive and Neural Systems, Boston University, for his valuable discussions and suggestions during his stay in Valladolid. Also we appreciate the contributions of Cristina de la Maza Pereg, Gregorio Ismael Sainz Palmero of the University of Valladolid and those of Prof. George Carayannis, Athina Petropoulou and Panos Antipas of the National Technical University of Athens. Finally, we thank Simon Jennings for his help in putting this paper into correct English.

REFERENCES

1. L. R. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition, *Proc of IEEE* **77**, 257–285 (1989).
2. R. P. Lippmann, Review of neural networks for speech recognition, *Neural Computation* **1**, 1–38 (1989).
3. C. A. Higgins and D. M. Ford, Stylus driven interfaces—The electronic paper concept, *1st International Conference on Document Analysis and Recognition*, Saint-Malo, France, 853–862 (1991).
4. C. C. Tappert, C. Y. Suen and T. Wakahara, The state of the art in on-line handwriting recognition, *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-12**, 787–808 (1990).
5. Y. le Cun *et al.*, Handwritten digit recognition: Applications of neural network chips and automatic learning, *IEEE Commun. Mag.* **27**, 41–46 (1989).
6. D. E. Rumelhart and J. L. McClelland (eds), *Explorations in the Microstructure of Cognition.* Bradford Books, Cambridge, Massachussetts (1986).
7. K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biol. Cybern.* **36**, 193–202 (1980).
8. S. Grossberg, How does a brain build a cognitive code? *Psychol. Rev.* **87**, 1–51 (1980).
9. G. A. Carpenter and S. Grossberg, ART2: self-organization of stable category recognition codes for analog input patterns. *Appl. Optics* **26**, 4919–4930 (1987).
10. G. A. Carpenter, S. Grossberg and D. B. Rosen, Fuzzy ART: fast stable learning and categorization of analog pattern by an adaptive resonance system, *Neural Networks* **4**, 759–771 (1991).
11. G. A. Carpenter, S. Grossberg and J. H. Reynolds, ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, *Neural Networks* **4**, 565–588 (1991).
12. G. A. Carpenter, S. Grossberg, S. Markuzon, J. H. Reynolds and D. B. Rosen, Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Trans. Neural Networks* **3**, 698–713 (1992).
13. G. Bradski, G. A. Carpenter and S. Grossberg, Working memory networks for learning temporal order with application to three-dimensional visual object recognition, *Neural Comput.* **4**, 270–286 (1992).

14. G. Bradski, G. A. Carpenter and S. Grossberg, Working memories for storage and recall of arbitrary temporal sequences *Int. Joint Conf. Neural Networks*, **II**, 57–62, Baltimore Maryland. (1992).
15. G. T. Toussaint, The use of context in pattern recognition, *Pattern Recognition* **10**, 189–204 (1978).
16. R. H. Anderson, Two-dimensional mathematical notation, *Syntactic Pattern Recognition Applications*. Springer-Verlag, New York (1977).
17. Z. X. Wang and C. Faure, Structural analysis of handwritten mathematical expressions, *9th International Conference on Pattern Recognition*, 32–34, Rome, Italy (1988).
18. W. H. Tsai and K. S. Fu, Attributed grammar—a tool for combining syntactic and statistical approaches to pattern recognition, *IEEE Trans. System, Man Cybern.* **SMC-10**, 873–885 (1980).
19. V. Quint, Une approche de l' edition structurée de documents, Ph.D. thesis, University of Grenoble, Grenoble, France (1987).
20. D. E. Knuth, *The T_EX Book*. Addison-Wesley, Reading Massachussetts (1989).
21. F. Nouboud and R. Plamondon, On-line recognition of handprinted characters: survey and beta tests, *Pattern Recognition* **23**, 1031–1044 (1990).
22. B. L. Mel et al., Tablet: personal computer in the year 2000, *Commun. ACM* **31**, 639–646 (1988).
23. Y. A. Dimitriadis, G. I. Sainz-Palmero and J. López Coronado, A gesture based interface for a mathematical editor, 6th *International Conference on Handwriting and Drawing*, 159–161, Paris, France, July (1993).
24. C. G. Wolf and P. Morrel-Samuels, The use of hand-drawn gestures for text editing, *Int. J. Man-Machine Studies*, 91–102 (1987).
25. K. H. Hanne and J. P. Hoppelman, Combined graphic and natural language interaction, *Graphics Interface* '88 105–112, Edmonton, Canada (1988).
26. A. Kankaanpaa, FIDS-A flat-panel interactive display system, *IEEE Comput. Graphics Applications* 71–82 (1988).
27. V. K. Govindan and A. P. Shivasprased, Character recognition—a review, *Pattern Recognition* **23**, 671–683 (1990).
28. C. C. Tappert, Cursive script recognition by elastic matching, *IBM J. Res. Dev.* **26**, 765–771 (1982).
29. D. J. Burr, Elastic matching of line drawings, *IEEE Trans. Pattern Analysis Mach. Intell.* **PAMI-3**, 708–713 (1981).
30. L. R. Rabiner, A. E. Rosenberg and S. E. Levinson, Considerations in Dynamic Time Warping algorithms for discrete word recognition, *IEEE Trans. Acoustics, Speech Signal Process.* **ASSP-26** (1978).
31. H-D Cheng and K. S. Fu, VLSI architectures for dynamic time-warp recognition of handwritten symbols, *IEEE Trans. Acoustics, Speech Signal Process.* **ASSP-34**, 603–613 (1986).
32. P. Y. Lu and W. Brodersen, Real-time on-line symbol recognition using a DTW processor, *7th International Conference on Pattern Recognition*, 1281–1283 Montreal, Canada (1984).
33. G. A. Carpenter and S. Grossberg (eds), *Pattern Recognition by Self-organizing Neural Networks*. Bradford Books/ MIT Press Cambridge, Massachussetts (1991).
34. S. Grossberg, *Studies of Mind and Brain*. Reidel Press, Boston (1982).
35. R. Plamondon, *Handwriting Control: a Functional Model*, Models of brain function, R. M. J. Cotteril (ed.), 563–574, Cambridge University Press, Cambridge, U.K. (1989).
36. R. F. Port, Representation and recognition of temporal patterns, *Connection Sci.* **2**, 151–176 (1990).
37. S. Grossberg, A theory of human memory: self-organization and performance of sensory-motor codes, maps and plans, *Prog. Theor. Biol.* **5**, 233–374 (1978).
38. S. Grossberg, Behavioral contrast in short-term memory: serial binary memory models or parallel continuous memory models? *J. Math. Psychol.* **17**, 199–219 (1978).
39. R. F. Atkinson and R. M. Shiffrin, The control of short-term memory, *Scientific American*, 82–90, August (1990).
40. J. P. Banquet and J. L. Contreras-Vidal, An integrated network: event related potentials model of temporal and probability context effects of event categorization, *International Joint Conference on Neural Networks* '92, **I**, 541–546, Baltimore, Maryland, June (1992).
41. C. Mannes, A neural network model of spatio-temporal recognition, recall, and timing, *International Joint Conference on Neural Network* '92, **IV**, 109–114, Baltimore, Maryland, June (1992).
42. E. Zalama-Casanova, C. García-Moreno and J. López-Coronado, Adaptive resonance theory architecture for authorization of deficits, 12*th International Conference Avignon* '92, **II**, 649–658, Avignon, France, June (1992).
43. F. G. Keenan, L. J. Evett and R. J. Whitrow, A large vocabulary stochastic syntax analyser for handwriting recognition, 1*st International Conference on Document Analysis and Recognition* **II**, 794–802, Saint-Malo, France, September (1991).
44. T. G. Rose, L. J. Evett and R. J. Whitrow, The use of semantic information as an aid to handwriting recognition, 1*st International Conference on Document Analysis and Recognition*, **II**, 629–637, Saint-Malo, France, September (1991).
45. J. J. Hull and S. N. Srihari, Experiments in text recognition with binary *n*-gram and Viterbi algorithms, *IEEE Trans. Pattern Analysis Mach. Intell.* **PAMI-4**, 520–530 (1982).
46. R. Shinghal and G. T. Toussaint, Experiments in text recognition with the modified Viterbi algorithm, *IEEE Trans. Pattern Analysis Mach. Intell.* **PAMI-2**, 184–193 (1979).
47. J. Raviv, Problem decision making in Markov Chains applied to the problem of pattern recognition, *IEEE Trans. Information Theory* **IT-3**, (1967).
48. C. J. Wells, L. J. Evett, P. E. Whitney and R. J. Whitrow, Fast dictionary look-up for contextual word recognition, *Pattern Recognition* **23**, 501–508 (1990).
49. S. K. Chang, A method for the structural analysis of two-dimensional expressions, *Information Sci.* **2**, 253–272 (1970).
50. P. A. Chou, Recognition of equations using a two-dimensional stochastic context-free grammar, *SPIE Visual Commun. Image Process.* **1199**, 852–863 (1989).
51. M. Bouhier, Generation de la semantique d' une formule mathematique, *Technical Rep. TIM-3*, IMAG, University of Grenoble, June (1989).
52. D. S. Nau, Expert computer systems, *IEEE Comput.* 63–85 (1983).
53. P. Deransart and J. Maluszynski, Relating logic programs and attribute grammars, *J. Logic Program.* **2**, 119–155 (1985).
54. E. Giakoumakis, G. Papakonstantinou and E. Skordalakis, Rule-based systems and pattern recognition, *Pattern Recognition Lett.* **5**, 267–272 (1987).
55. E. Zalama-Casanova, F. J. Díaz-Pernas, Y. A. Dimitriadis and J. López Coronado, A new Adaptive Resonance Theory architecture, able to categorize input patterns that contain information of different nature, *J. Systems Eng.* **3**, 89–109 (1993).
56. L. Schomaker, Using stroke- or character-based self-organizing maps in the recognition of on-line, connected cursive script, *Pattern Recognition* **26**, 443–450 (1993).
57. P. Morasso, L. Barberis, S. Pagliano and D. Vergano, Recognition experiments of cursive dynamic handwriting with self-organizing networks, *Pattern Recognition* **26**, 451–460 (1993).
58. G. A. Carpenter, S. Grossberg and K. Iizuka, Comparative performance measures of Fuzzy ARTMAP, Learned Vector Quantization, and Backpropagation for handwritten character recognition, *International Joint Conference on Neural Networks*, **I**, 794–799, Baltimore, Maryland, June (1992).

About the Author—YANNIS A. DIMITRIADIS was born in Athens, Greece. He received the Engineering Diploma from the National Technical University of Athens, Greece and the M.Sc. degree from the University of Virginia in 1981 and 1983, respectively, all in electrical engineering. He received his Ph.D. degree in engineering from the University of Valladolid, Spain in 1992. From June 1983 to August 1985 he served with the military service in Greece. He worked at the Ministry of Science and Technology of Greece from 1986 to 1988. From 1985 to 1988 he was a teaching assistant in the National Technical University of Athens. Between January 1989 and March 1993 he was a research fellow of the Spanish Government. Currently he is visiting professor at the Telecommunications Engineering School of the University of Valladolid, Spain. He spent two periods of three months, in 1991 and 1993, at the Center for Adaptive Systems, Boston University. His interests include statistical pattern recognition, fuzzy systems, neural networks, analysis of handwriting and document recognition. He is member of the INNS, IEEE, ACM and the National Technical Chamber of Greece. He is also a founder member of the Spanish Regional Interest Group in Neural Networks of IEEE.

About the Author—JUAN LÓPEZ CORONADO was born in 1946 in Barcelona, Spain. He received his Engineering degree from the *Universidad Politécnica de Barcelona* in 1974 and the Ph.D. degree from the *Ecole Nationale Superieur de l'Aeronautique et de l'Espace* in Toulouse, France in 1981. During that period he worked in the field of optimal control of nonlinear processes in industrial and aeronautic applications. In 1985 he received his Ph.D. in industrial engineering at the University of Valladolid and joined the department of systems engineering and control as a professor in 1987. Dr Juan López Coronado is involved in several european research projects. His research interests include computer vision, pattern recognition, 3D image processing, artificial intelligence and neural networks. He has published numerous articles in the fields of automatic control, image processing and neural networks. He is a member of the IEEE, INNS and the Pattern Recognition Society.