Científica Vol. 10 Núm. 3 pp.131-141
© 2006 ESIME-IPN. *ISSN 1665-0654*. Impreso en México

Associative Memories Applied to Printed Word Recognition

Benjamín Cruz
Humberto Sossa
Ricardo Barrón

Centro de Investigación en Computación
Instituto Politécnico Nacional.
Av. Juan de Dios Batis y M. Othón de Mendizábal
Col. Nueva Industrial Vallejo, México, DF. CP 07738
MÉXICO.

Tel. 5729 6000 ext. 56512. Fax 5729 6000 ext. 56607

email: benjamincruz@sagitario.cic.ipn.mx
hsossa@cic.ipn.mx
rbarron@cic.ipn.mx

Recibido el 2 de mayo de 2005; aceptado el 27 de enero de 2006.

1. Abstract

We describe a simple but effective methodology for the recognition of printed words from incomplete versions of them. The proposed methodology incorporates two main stages, one of word learning and one of word recognition. During word learning sets of different words with different number of letters are used to build a bank of associative memories, one associative memory for each group of words with the same number of letters. During word identification, a word is recognized as follows: First its available letters are recognized by means of a recently proposed recognition scheme based on pattern decomposition. Spaces between available letters are next detected by appropriate segmentation techniques. The whole pattern composed of the identified letters and detected spaces is presented to the corresponding trained associative memory that in turn responds with the appropriate restored word. Formal conditions for correct reconstruction of a word in the presence of alterations are provided. Numerical and real examples are also given to show the efficiency of the proposal.

Key words: Pattern reconstruction; Pattern restoration; Pattern decomposition; Associative processing.

2. Resumen (Memorias asociativas aplicadas al reconocimiento de palabras impresas)

Se describe una metodología simple pero efectiva para el reconocimiento de palabras impresas a partir de versiones incompletes de ellas. La metodología propuesta incorpora dos etapas principales, una de aprendizaje de palabras y otra de reconocimiento. Durante el aprendizaje, conjuntos de palabras diferentes con números de letras distintos son usados para construir un banco de memorias asociativas, una memoria asociativa para grupo de palabras con el mismo número de letras. Durante la identificación de palabras, una palabra es reconocida como sigue: primero sus letras disponibles son reconocidas por medio de una técnica recientemente propuesta, basada en la descomposición de patrones. Los espacios entre letras son detectados por medio de técnicas apropiadas de segmentación. El patrón completo compuesto de letras identificadas y espacios detectados es presentado a la memoria entrenada correspondiente, que en turno responde con la salida deseada. Se dan las condiciones formales de reconstrucción de una palabra en presencia de letras faltantes. Se dan además ejemplos numéricos y con ejemplos reales para demostrar la eficiencia de la propuesta.

Palabras clave: Reconstrucción de patrones, restauración de patrones, descomposición de patrones, procesamiento asociativo.

3. Introduction

An important problem in automatic document analysis is the identification of printed or written words. In some cases the printed words will appear complete. Unfortunately, due to acquisition procedures and other factors such time, words will appear noisy, and in some cases, like for example in ancient documents, they might appear incomplete, even with letters missing. In this work, we concentrate on the problem of identifying printed words from altered versions of them when letters are missing.

One possible solution to the previously described problem could be to iteratively replace the spaces between available letters of an incomplete word, visit the list of possible words in a word dictionary and make guesses until obtaining the desired result. By following appropriate construction and

orthographic rules and depending on the size of the word and the number of words in the dictionary, we should arrive sooner or latter to the same result in a shorter time.

Instead of using any traditional searching scheme, we prefer to use in this work an associative memory to give a solution to this problem. Associative memories have demonstrated their usefulness in many applications, refer for example to [1-13]. One of these applications is pattern restoration or pattern reconstruction, *i.e*. given a distorted version a of a pattern a, the goal is to restore (in one step) by applying to the previously constructed memory.

Lots of models of associative memories have been emerged in the last 40 years, since the simple Lermatrix of Steinbuch [1], then the Linear Associator of Anderson [2] and Kohonen [3], then the well-known model proposed by Hopfield in 1982, the Hopfield Memory [4], and finally the morphological associative models of Ritter and their variations [5-11].

3. Problem statement

Before stating the problem to be solved in this research, let us have the following definitions:

Definition 1.
A word s is a chain of letters belonging to an alphabet. Examples of words are the words in this phrase.

Definition 2.
An incomplete version (word) $\bar{s}$ is an incomplete version of a word s with missing letters.

Incomplete versions of the word ELEPHANT are, for example, the following three: EL∅PH∅NT, ∅LEPH∅∅T, and ELE∅∅∅NT, with ∅ meaning an empty space. It is clear that depending on the number of available letters the identification of the word would be more or less difficult to achieve.

Problem statement: Given an incomplete version $\bar{s}$ (noisy version), of a word s belonging to a dictionary D of words of different sizes, reconstruct s under the following restrictions:

R-1. The words $s_1, s_2, \ldots, s_p$ belonging to D are all different and can be composed of different number of letters.
R-2. Letters are supposed to be isolated from each other.
R-3. Spaces between available letters are supposed to have more or less the same size. Their size is comparable to the size of a letter.
R-4. The first and last letters of the word are supposed to be present.

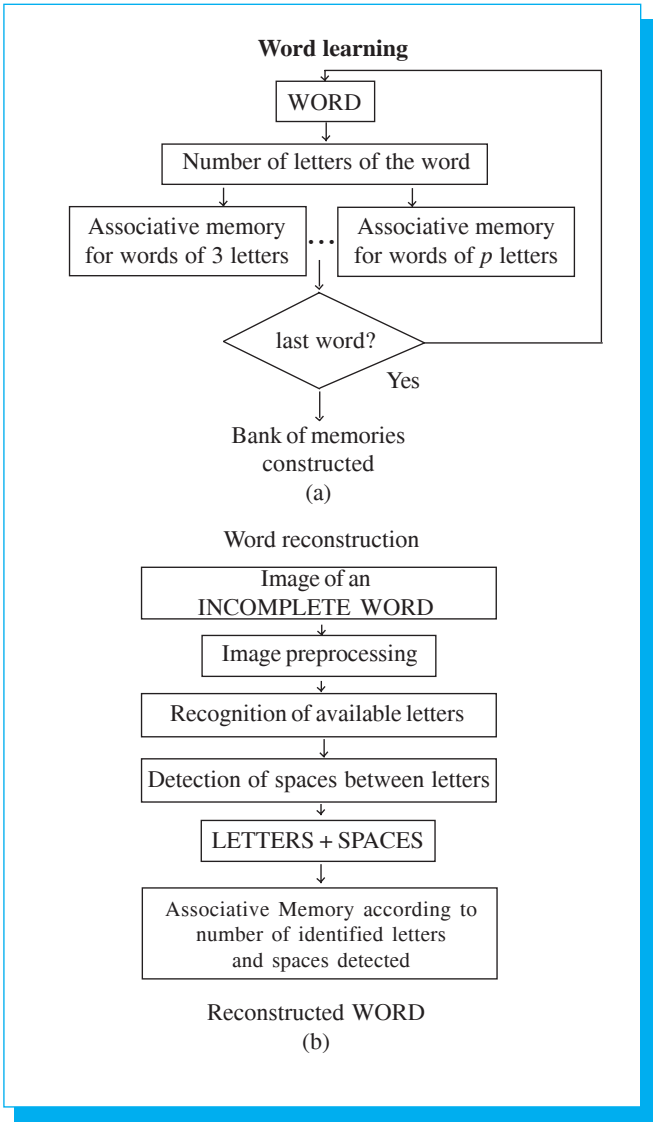

Fig. 1. The main steps of the proposed methodology.

4. Development

4.1 Basics on associative memories

An associative memory M is a special kind of neural network with just one layer. Two main advantages of associative memories over other recalling mechanisms are 1) training of the memory is done in one epoch, 2) pattern recalling is done in one step, mining this that the desired pattern is restored in one step after presenting to the associative memory its corresponding key-pattern.

As defined in [8], an associative memory can be seen as an input-output system by which we can associate input patterns with outputs pattern as: $a \rightarrow M \rightarrow b$, with a and b, respectively the input and the output patterns vectors. Each input vector forms an association with its corresponding output vector. An association between input pattern a and output pattern b is denoted by (a,b). For k a positive integer, the correspondent association will be (a^k,b^k). Usually, an associative memory M is represented by a matrix whose ij-th component is m_{ij} [8]. M is generated from a finite and a priori set of known associations, known as *fundamental set of association or simply fundamental set* (FS) [8]. If k is an index, this FS is represented as: $\{(a^k,b^k), k = 1,\ldots,p\}$, with p the cardinality of the set. The patterns integrating a FS are called *fundamental patterns* [8]. If for $k = 1,p$, it holds that $(a^k = b^k)$, then that memory is auto-associative, otherwise it is hetero-associative [8].

Fundamental patterns could appear distorted with noise. A distorted version of a pattern a will be denoted as $\tilde{a}$. If when presenting to an associative memory M a fundamental pattern, M responds with the correct pattern, we say that M presents correct recall. If for all patterns of a given FS, correct recall is obtained, M is said to presents correct recall.

4.2 The proposal

The stages of the proposal are next explained in detail in this section. The methodology to automatically restore a given word from an incomplete version of it has two main stages, one of learning the set of words, and one of restoring a given word from an incomplete version of it. Figure 1 shows the main steps of each of these two stages.

4.2.1 Word learning stage

Words to be further identified from incomplete versions of them are firstly stocked into a bank of associative memories. One memory is used for words of three letters; one more memory is used for words of four letters, and so on. Each set of words (according to the number of elements of each word) is called *base set*. Before each word is stoked into its corresponding associative memory, each letter is firstly converted to a *binary vector letter* as shown in Table 1. Each given word is then converted to a *vector word* as shown in the example depicted in Figure 2.

As shown in Figure 1 (a), for each transformed base set, a memory is constructed. This ends word memorization.

Table 1. Representation of each letter for training and recognition purpose.

Letter	Position	Representation
A	1	00001
B	2	00010
⋮	⋮	⋮
Y	25	11001
Z	26	11010

4.2.2 Memory model used

As described before, words are going to be recognized from incomplete versions of them. Spaces between available letters of a word can be considered as subtractive noise added to the pattern. Thus any associative memory able to cope with subtractive noise would efficiently do the job. In this work we decided to use the morphological αβ associative memory of type Λ, first introduced in [7] to first memorize and then restore words from incomplete versions of them. Just to remember, Λ-associative memories, during training, make use of alpha (α) operation [7] between elements of the patterns to be learned followed by **min** operator of the partial constructed memories. Given a set of p patterns $a^k = (a_1^k\, a_2^k \ldots a_n^k)$, $k = 1,\ldots,p$, the elements are binary numbers:

1. Build p matrices of size $n \times n$ as follows:

$$W^k = \begin{pmatrix} w_{11}^k = \alpha(a_1^k, a_1^k) & w_{12}^k = \alpha(a_1^k, a_2^k) & \cdots & w_{1n}^k = \alpha(a_1^k, a_n^k) \\ w_{21}^k = \alpha(a_2^k, a_1^k) & w_{22}^k = \alpha(a_2^k, a_2^k) & \cdots & w_{2n}^k = \alpha(a_2^k, a_n^k) \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1}^k = \alpha(a_n^k, a_1^k) & w_{n2}^k = \alpha(a_n^k, a_1^k) & \cdots & w_{nn}^k = \alpha(a_n^k, a_n^k) \end{pmatrix}$$

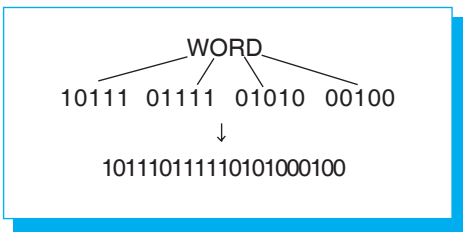

Fig. 2. Composition of a vector word in terms of its vector letters.

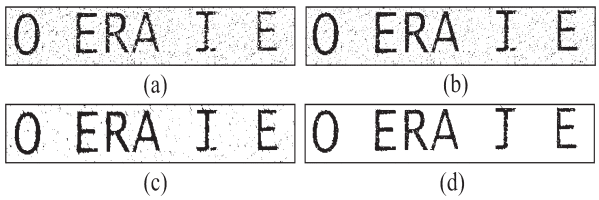

Fig. 3. Preprocessing of an image. (a) Gray-level image. (b) Corresponding binary version. (c) Filtered image by morphological opening-closing, and (d) Spurious component removal.

2. Perform **min** ($\wedge$) operation between the elements of the matrices W^1, W^2,..., W^p as follows to get the final associative matrix for this set of words:

$$W = W^1 \wedge W^2 \wedge \cdots \wedge W^p =$$

$$\begin{pmatrix} w^1_{11} \wedge \cdots \wedge w^p_{11} & w^1_{12} \wedge \cdots \wedge w^p_{12} & \cdots & w^1_{1n} \wedge \cdots \wedge w^p_{1n} \\ w^1_{21} \wedge \cdots \wedge w^p_{21} & w^1_{22} \wedge \cdots \wedge w^p_{22} & \cdots & w^1_{2n} \wedge \cdots \wedge w^p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w^1_{n1} \wedge \cdots \wedge w^p_{n1} & w^1_{n2} \wedge \cdots \wedge w^p_{n2} & \cdots & w^1_{nn} \wedge \cdots \wedge w^p_{nn} \end{pmatrix}$$

3. This ends the learning stage.

During pattern restoration Λ-associative memories make use beta (β) and **max** operations for the reconstruction of the pattern. For the details about the functioning of $\alpha\beta$ memories refer to [7].

4.2.3 Word identification stage

A word is identified in five steps as follows:

Step 1. Word preprocessing. During this step, a gray-level image of a word is firstly preprocessed to get a binary image. A standard thresholder is used for this purpose [14]. Spurious components sometimes due to bad image thresholding are next removed from this binary image by a combination of standard morphological filters and small-component-removing techniques to get a filtered image [15]. In Figure 3 (a), a gray level image of a word with some missing letters is shown. Its corresponding binary version is shown in Figure 3 (b) when Otsu's thresholder [14] is used. Figure 3 (c) shows the corresponding filtered image after morphological opening followed by a morphological closing. Finally, Figure 3 (d) shows the preprocessed image after using a size filter to eliminate components sized in less than 10 pixels.

Fig. 4. Segmentation by connected component labeling of letters of a word from its image.

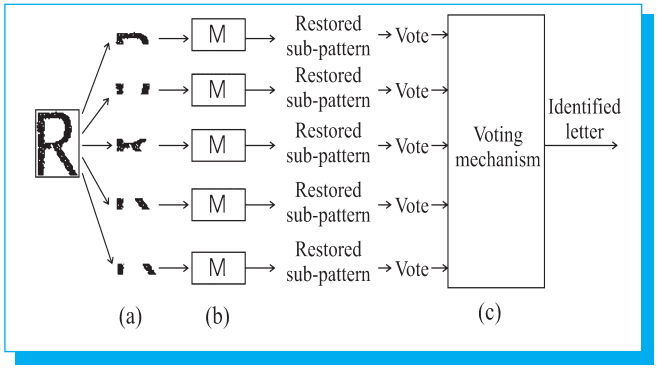

Fig. 5. Recognition of an individual letter by means of subpattern decomposition approach described in [13].

Step 2. Letter segmentation. Normally if step 1 is performed correctly, letters of the word would appear isolated. A standard connected-component-labeling algorithm is then used to automatically segment (by component labeling) the letters in the image [16]. Figure 4 shows the labeled (segmented) letters of the image of Figure 3 (d) after connected component labeling.

Step 3. Individual letter recognition. A segmented letter is identified by means of the sub-pattern decomposition and recognition approach recently introduced in [13]. In short, the pattern (a letter in this case) is first decomposed into a set of sub-patterns as shown in Figure 5 (a). This decomposition into sub-patterns is justified due to the possible distortions introduced to the whole pattern. In [13], it is shown that this decomposition provides, in general, much better identification results than using the whole pattern for learning and identification. According to [13], each individual sub-pattern is next presented to an associative memory for its restoration (Figure 5(b)). The letter is identified by the number of votes assigned to its class in terms of the number of correctly restored patterns (Figure 5(c)). For the details of the functioning of this technique, refer to [13].

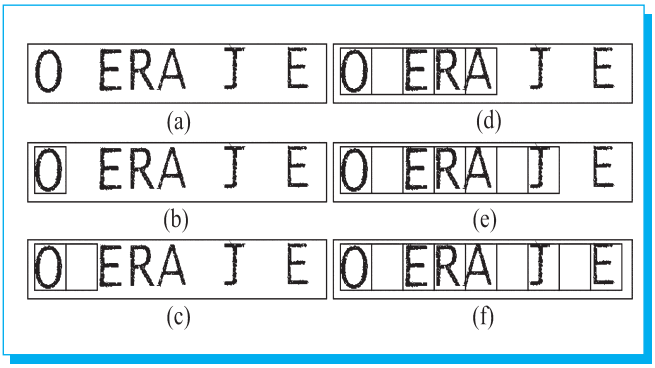

Fig. 6. Detection of spaces between letters of a word. Three spaces between letters have been detected. (a) Binary version of image of Figure 3. (b) Mask. (c) First space detected. (e) Second spaced detected. (f) Third space detected.

Step 4. Detection of space between letters. Due to restriction R-3, we first build a blank mask (a mask filled with 0's). Its size is chosen as the size of the first letter in the word. In the horizontal direction the size of the scanning mask is a little bit longer (for an example, refer to Figure 6 (b)): the horizontal size of the letter plus a small space representing the small empty space between two adjacent letters. To detect thus the blank spaces between letters we swap the previously generated blank mask and perform an AND operation between its elements and the elements under it. When a logical «0» is obtained we mark the swapped space as an empty space. The swapping distance is always the size of the generated mask. Each time a logical «0» is obtained an empty space is marked. We apply this process until completely swapping the whole image. Figure 6 shows the detected spaces on the segmented image of Figure 4 after applying the described procedure described. In the example three empty spaces have been detected by applying the described procedure.

Step 5. Word identification. The identity of the recognized letters, their positions and the blank spaces and positions detected in Step 4 are used to completely identify the desired word. The number of recognized letters plus the number of detected blank spaces automatically tells the proposal which associative memory to use. This information is arranged in a vector as: $\bar{a} = (\bar{a}_1\ \bar{a}_2\ \ldots\ \bar{a}_n)$, where, of course, some $\bar{a}_i$'s represent recognized letters and some $\bar{a}_i$'s represent blank spaces between them. Each $\bar{a}_i$'s is either a binary vector of Table 1 or a 00000 representing a blank space. For the incomplete version: O∅ERA∅I∅E of the word OPERATIVE, vector $\bar{a} = (\bar{a}_1\ \bar{a}_2\ \ldots\ \bar{a}_n)$, according to Table 1, would be:

$\bar{a}$ = (01111 00000 00101 10010 00001 00000 01000 00000 00101)

This is the information presented to the associative memory for the reconstruction of word OPERATIVE.

4.2.4 Conditions for perfect recall of a word

In this section we present formal conditions under which a given word already memorized by an associative memory can be perfectly restored from a version of it with missing letters.

At the end of the restoration process we can have two cases: 1) the desired word correctly restored, or 2) the desired word not correctly restored. If it is not restored this is mainly due to the restoration response of the adopted auto-associative memories. For these memories works correctly in the presence of noise several conditions must hold [7]:

a) For $k = 1,\ldots, p$, let $\tilde{x}^k$ an altered version of pattern x^k and M an morphological associative memory of kind **min**. Thus $M\Delta\tilde{x}^k = y^k$ (we have perfect recall) if and only if:

$$\tilde{x}_j^k \geq x_j^k \wedge \bigvee_{i=1}^{q}\left(\bigwedge_{l\neq k}\left[y_i^k - y_i^l + x_i^l\right]\right)\forall\, j = 1,\ldots,n$$

and if for each row index $i\in\{1,\ldots,q\}$ there exists a column index $j_i\in\{1,\ldots,n\}$ such that:

$$\tilde{x}_j^k = x_j^k \wedge \left(\bigwedge_{l\neq k}\left[y_i^k - y_i^l + x_{j_i}^l\right]\right)$$

b) Let $\{(a^k,b^k),\ k = 1,\ldots,q\}$ the fundamental set of an αβ associative memory of kind $\wedge$ represented by M, and let $\tilde{x}\in\{1,0\}^n$ an altered pattern with noise. Then, it holds that

$$M \wedge_\beta \tilde{x} = y^k \text{ (we have perfect recall)}$$

if for each $i\in\{1,\ldots,q\}$ it holds that: $\exists j = j_0 \in \{1,\ldots,n\}$, which depends on k and i such that $M_{ij0} \leq \alpha(y^k_i, \tilde{x}_{j0})$.

To reach these conditions, at the moment of forming vector words, it is very difficult; by adding more words, it is much more difficult.

Also, if to a word a letter is removed, and the word can not correctly recalled with that alteration, at the moment of removing one more letter, it sounds logic that the word would not be correctly restores. From now on, these letters will be called *key-letters*.

Definition 1. Let P an unaltered word (an chain of characters), and let $V = [v_1, v_2,\ldots, v_i, \ldots, v_n]$ (where $v_i \in B^m$, $V \in B^{nxm}$ *and* $B = \{1,0\}$) the *word vector* of P. v_i is letter vector of some letter of P. Also, let

$$S = \{\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_j, \ldots, \tilde{V}_k\}$$

a set, where $\tilde{V}_j$ is an altered version of V where v_i is the zero vector. If $\forall_{j=1,\ldots,k}, \tilde{V}_j$ does not satisfy the conditions for recalling in the presence of noise of one of the associative memories previously mentioned, then the letter that represents the *letter vector* v_i it is a *key-letter*.

More precisely, a key-letter is that letter that if it removed form the word, the word vector does not satisfies the conditions for recalling in the presence of noise of the selected associative memories.

The ideal is that the key-letters be the first and/or the last of each word. In consequence (as we will see in the section of experiments) perfect recall will be always obtained. Unfortunately, this is not case for all the words; for some words, more that one key-letter can be found.

One more thing that is worth mentioning is that the key-letters change from word to the other; this is, that for a word a key-letter could be let say a «b», and for other word containing a «b», the key-letter can be for example a «d». The following propositions gives the conditions under which a given word can be perfectly recalled from altered versions of it.

Theorem 1. Let $\tilde{P}$ an incomplete version of P, and let

$\tilde{V} = [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_i, \ldots, \tilde{v}_n]$ and $V = [v_1, v_2, \ldots, v_i, \ldots, v_n]$

where $\tilde{v}_i, v_i \in B^5$, $\tilde{V}, V \in B^{n\times 5}$ and B = {1,0},

the incomplete and complete vector word of word P, respectively. Besides, $\tilde{v}_i, v_i$ are the letter vector of the letters of $\tilde{P}$ and P. If $\forall_{j=1,\ldots,n}$ the letter representing letter vector v_i is a key-letter and $\tilde{v}_i$ is not the zero vector, then the restoration is always perfect.

Proof. By the definition of key-letter, this theorem always holds. ■

(a) (b)

Fig. 7. Example of test image. (a) original image. (b) original image after image pre-processing.

Table 2. Sets of words in Spanish from 3 to 10 letters used in the experiments.

Number of letters per word

3	4	5	6	7
DOS	CAMA	PERRO	MEXICO	COMEDOR
CIC	GATO	SABER	VECTOR	PALABRA
IPN	REZO	LLAVE	FIESTA	RESUMEN
ARO	MIRO	GESTO	FUERTE	PLANETA
CAL	SUBE	ILESO	GRANDE	ESPACIO
PAZ	VALE	MICRO	ESPADA	LEYENDA
---	SOPA	PEDAL	HOMBRE	GENERAL
---	HUGO	LIBRO	CARTON	CABALLO
---	---	MARCO	---	---
---	---	LEGAL	---	---

8	9	10
CONTADOR	RESFRIADO	EXTRANJERO
ELEFANTE	AUTOMOVIL	TRANSPORTE
PROFESOR	REPUBLICA	PARTICULAR
MANIOBRA	DEFENSIVA	CIENTIFICO
RESPIRAR	ELECTRODO	---
FANTASIA	COCODRILO	---
---	LAGARTIJA	---
---	---	---
---	---	---
---	---	---

With Theorem 1 we get a necessary and sufficient condition for the perfect restoration of incomplete words. This is to correctly restore a word, it must contain all its key-letters, at the moment of word restoration. In short, if a word contains all its key-letters, by Theorem 1, we always have perfect recall.

One way to get the key-letters is to take out one letter at the time and to test the methodology, if at a given moment, correct restoration is not obtained, then that letter is a key-letter.

4.2.5. Experimental results

In this section the performance of the described methodology is tested. For this the set of words of different sizes shown in Table 2 is used. For word learning and word identification purposes, letters are represented as explained as shown in Table 1. Also, the empty space representing a missing letter in the word is represented as «$0_{10} = 00000_2$».

Fig. 8. Different altered versions of the word GENERAL

Table 3. Set of words for test number 1.

Set	Number of words	Number of altered versions
Words of 3 letters	6	6
Words of 4 letters	8	16
Words of 5 letters	10	30
Words of 6 letters	8	32
Words of 7 letters	8	40
Words of 8 letters	6	36
Words of 9 letters	6	49
Words of 10 letters	4	32

Table 4. Set of words for test number 2.

Set	Number of words	Number of altered versions
Words of 3 letters	-	-
Words of 4 letters	8	8
Words of 5 letters	10	30
Words of 6 letters	8	48
Words of 7 letters	8	80
Words of 8 letters	6	90
Words of 9 letters	6	126
Words of 10 letters	4	112

We took eight sets of images of words of different number of letters, each one with a determined number of words. Each image was obtained by means of a digital camera attached to the main computer. The procedure was first to print the words on a sheet of paper; the images were next obtained with the camera. An example of these images is shown in Figure 7 (a). Figure 7 (b) shows the preprocessed image after the application of the procedures described in section 4. Table 2 shows the set of words, from 3 to 10 letters, used in the experiments. It is worth mentioning that the adopted letter style facilitates the detection of the missing spaces.

To test the efficiency of the proposal, several letters were removed from each word, obtaining new versions of them. Figure 8 shows several altered versions of the word GENERAL. In the first case one letter was removed out; in the second case two letters were removed; in the third three letters were removed.

Tests were done as follows: For the first experiment, for each set of words we took a word and generated all the possible altered versions by removing one letter. For the second experiment, we generated all the possible versions of each word by removing now two letters, and so on. This procedure was applied to all the words of each word set.

One last experiment was performed without removing the so-called key-letters (section 4.4). In total nine experiments were performed.

Test No. 1. Taking out one letter. In this case, we took the sets of words according to Table 3. Column two shows the number of words of each set. As mentioned these words were altered by removing one letter by knowing that the first and last letters must always appear. Third column shows the number of altered words for each set. Recognizing results are

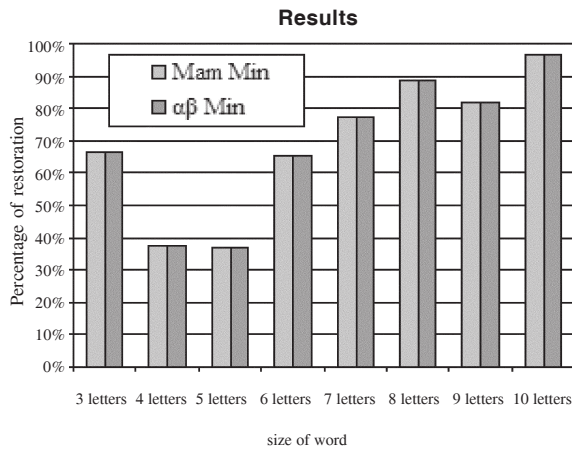

Fig. 9. Restoration results of the set of words of test number 1.

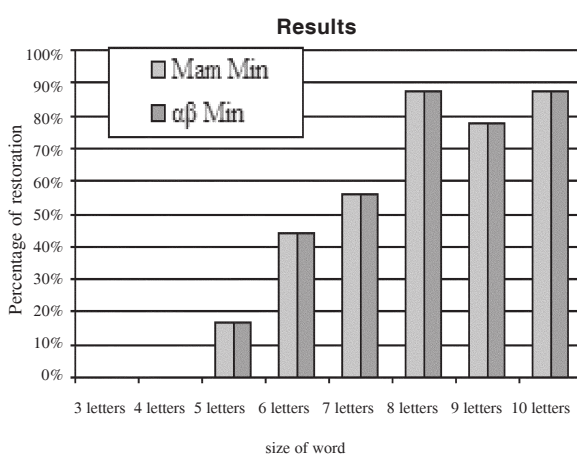

Fig. 10. Restoration results of the set of words of test number 2.

Table 5. Set of words for test number 3.

Set	Number of words	Number of altered versions
Words of 3 letters	-	-
Words of 4 letters	-	-
Words of 5 letters	10	10
Words of 6 letters	8	32
Words of 7 letters	8	80
Words of 8 letters	6	120
Words of 9 letters	6	210
Words of 10 letters	4	224

shown in Figure 9. In Figures 9 to 16, Mam means Morphological associative memory.

Test No. 2. Taking out two letters. Second test was done with the same sets of words, but instead of taking out one letter, two letters were removed. Table 4 shows the number of words per set. Note that in this experiment, the first set of words of three letters is not used due to if two letters were removed, restriction R-4 would be violated. Restoration results for this experiment are shown in Figure 10.

Test No. 3. Taking out three letters. This test was done with the same sets of words, but three letters were removed. Table 5 shows the number of words per set. Note that in this experiment, the first and second sets of words of three and four letters were not used for the same reasons as explained. Restoration results for this experiment are shown in Figure 11.

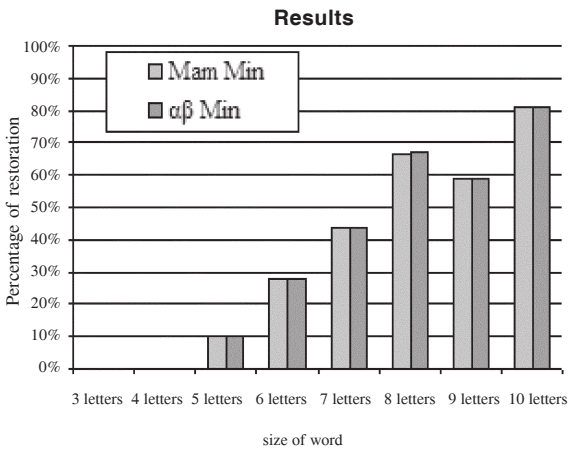

Fig. 11. Restoration results of the set of words of test number 3.

Table 6. Set of words for test number 4.

Set	Number of words	Number of altered versions
Words of 3 letters	-	-
Words of 4 letters	-	-
Words of 5 letters	-	-
Words of 6 letters	8	8
Words of 7 letters	8	40
Words of 8 letters	6	90
Words of 9 letters	6	210
Words of 10 letters	4	280

Test No. 4. Taking out four letters. This test was done with the same sets of words, but now four letters were removed. Table 6 shows the number of words per set. Note that in this experiment, the first, second and third sets of words were not used. Restoration results for this experiment are shown in Figure 12.

Test No. 5. Taking out five letters. This test was done with the same sets of words, but now five letters were removed. Table 7 shows the number of words per set. Note that in this experiment, the first to fourth sets of words were not used. Restoration results for this experiment are shown in Figure 13.

Test No. 6. Taking out six letters. This test was done with the same sets of words, but now six letters were removed. Table 8 shows the number of words per set. Note that in this experiment, the first to fifth sets of words were not

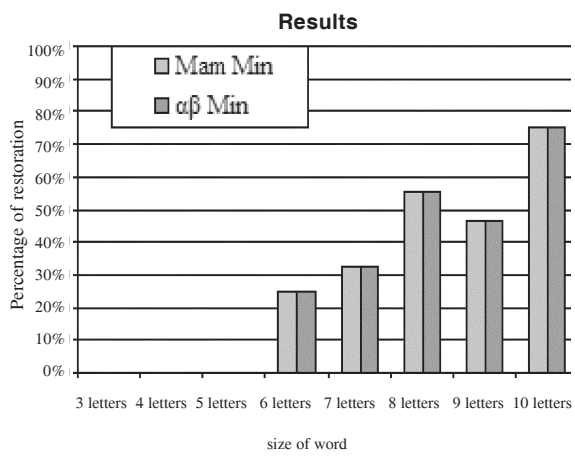

Fig. 12. Restoration results of the set of words of test number 4.

Table 7. Set of words for test number 5.

Set	Number of words	Number of altered versions
Words of 3 letters	-	-
Words of 4 letters	-	-
Words of 5 letters	-	-
Words of 6 letters	-	-
Words of 7 letters	8	9
Words of 8 letters	6	36
Words of 9 letters	6	126
Words of 10 letters	4	224

used. Restoration results for this experiment are shown in Figure 14.

Tests No. 7 and 8. Taking out seven and eight letters. This test was done with the same sets of words, but now seven and eight letters were removed. Table 9 shows the number of words per set. Note that in this experiment, the first to sixth sets of words were not used. Last row of this table shows the only set of words with eight letters missing. Restoration results for this experiment are shown in Figure 15.

Test No. 9. Key-letter not removed. One way to detect the key letters for a given word is by removing one by one its letters and test the methodology; if in one case perfect restoration is not obtained, then that letter is a key-letter. Once the key-letters are identified, we again performed the eight experiments already done, without eliminating the detected key-letters. By proceeding this way, it reduced a little the

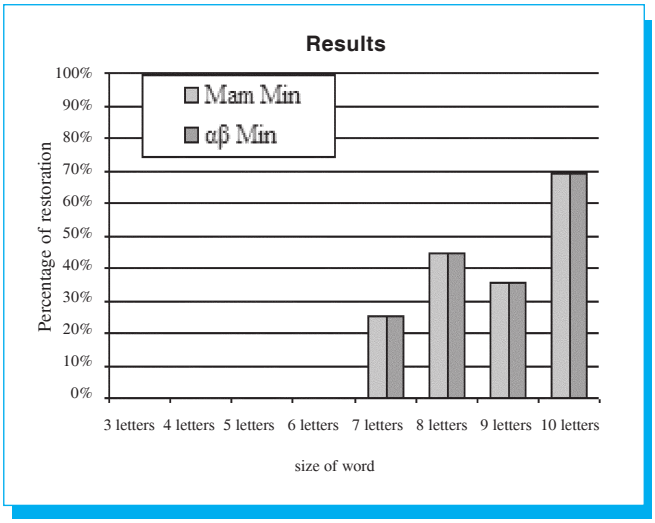

Fig. 13. Restoration results of the set of words of test number 5.

Table 8. Set of words for test number 6.

Set	Number of words	Number of altered versions
Words of 3 letters	-	-
Words of 4 letters	-	-
Words of 5 letters	-	-
Words of 6 letters	-	-
Words of 7 letters	-	-
Words of 8 letters	6	6
Words of 9 letters	6	42
Words of 10 letters	4	112

number of altered versions of each set. The results of this test are shown in Figure 16. As can be appreciated, in all cases patters restoration was of 100%.

Discussion. From the experiments, we can observe that as the number of removed letters increases, the efficiency of the proposal reduces. By making an analysis of the word vectors, we arrived to the conclusion that restoration was not correctly achieved mainly to the reconstruction capacity of the adopted auto-associative memory model.

One more thing that it is worth mentioning is that the percentages of error are carried from one test to the other due to fact that the words that were not correctly restored in a past test, were not also restored in subsequent tests.

Finally, as we can see from Figure 16, in all cases restoration was of 100%. Theorem 1 assures pattern perfect restoration if key-letters are not removed from the words.

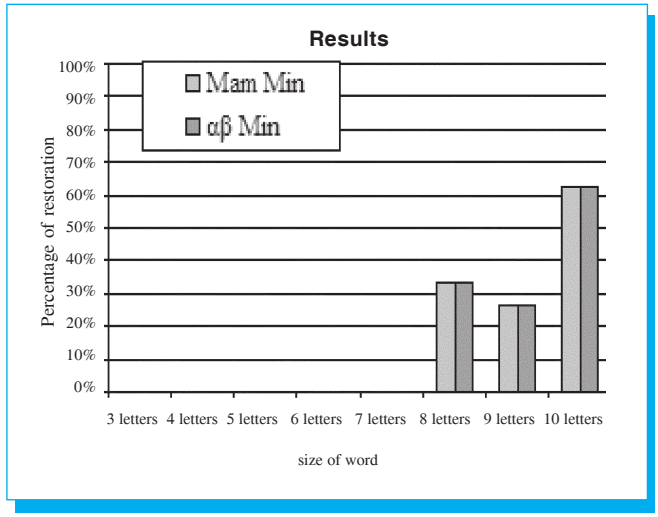

Fig. 14. Restoration results of the set of words of test number 6.

Table 9. Set of words for test number 7 and 8.

Set	Number of words	Number of altered versions
Words of 3 letters	-	-
Words of 4 letters	-	-
Words of 5 letters	-	-
Words of 6 letters	-	-
Words of 7 letters	-	-
Words of 8 letters	-	-
Words of 9 letters	6	4
Words of 10 letters	4	32
Words of 10 letters	4	4

5. Conclusion

In this paper, we have described a very simple but effective methodology for the reconstruction of printed words when some letters of the word are missing. A combination of classical image processing and modern information techniques has been used for this goal. From the set of patterns presented, the performance of the proposal has been shown to be very effective and promising. We have shown for the first time in this paper how image processing and associative memories can together contribute to the solution of an important pattern recognition problem.

Nowadays, we are testing the proposal with bigger sets of words with different kind of letters. We are also investigating the possibility to employ the proposal to the important problem of restoring documents from partial information of them. Finally, we are searching for an automatic way to find, given a set of words with the same number of letters, the key-words. Until now we do this manually.

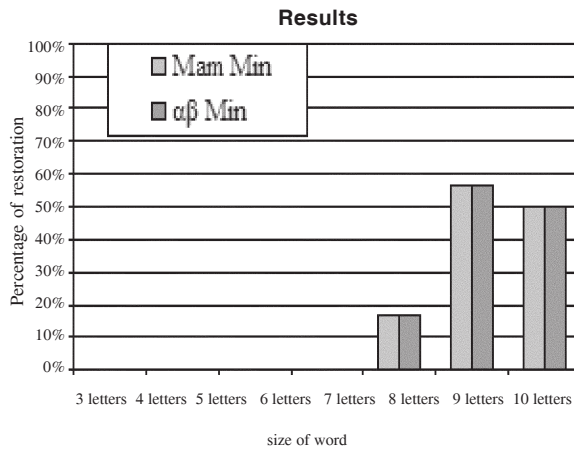

Fig. 15. Restoration results of the set of words of test number 7 and 8.

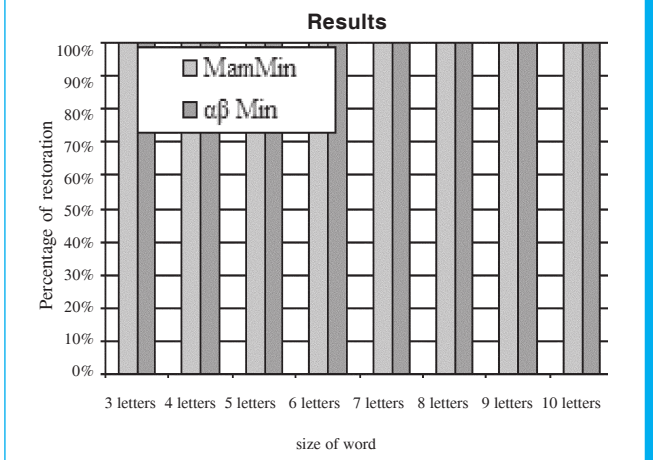

Fig. 16. Restoration results of the set of words of test number 9.

Acknowledgements

This work was economically supported by SIP-IPN under grants 20050156 and 2006517 and CONACYT by means of grant 46805. Benjamín Cruz specially acknowledges CONACYT for the economical support during his postgraduate studies at CIC-IPN.

6. References

[1] K. Steinbuch (1961). *Die Lernmatrix*, *Kyberneitk* **1**:26-45.
[2] J. Anderson (1972). A simple neural network generating an interactive memory, *Mathematical Biosciences,* **14**: 197-220.
[3] T. Kohonen (1972). Correlation matrix memories, *IEEE Transactions Computers,* **21(4)**: 444-445.
[4] J. Hopfield (1982). Neural Network and Physicals systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, **79**: 2554-2558.
[5] G. X. Ritter *et al*. (1998). Morphological associative memories, *IEEE Transactions on Neural Networks*, **9**: 281-293.
[6] G. X. Ritter *et al*. (1999). Morphological bi-directional associative memories, *Neural Networks*, **12**:851-867.

[7] C. Yáñez (2002). *Associative Memories based on Order Relations and Binary Operators* (In Spanish), PhD Thesis, Center for Computing Research-IPN.
[8] M. H. Hassouon (1993). *Associated neural memories. Theory and implementation*. Oxford University Press.
[9] H. Sossa *et al*. (2004). *Extended Associative Memories for Recalling Gray Level Patterns*. LNCS 3287. Springer Verlag. pp. 187-194.
[10] H. Sossa *et al*. (2004). *New Associative Memories to Recall Real-Valued Patterns*. LNCS 3287. Springer Verlag. pp. 195-202.
[11] H. Sossa *et al*. (2004). *Real-Valued Pattern Recall by Associative Memory*. LNAI 3315. pp. 646-655.
[12] H. Sossa *et al*. (2005). Associative gray-level pattern processing using binary decomposition and a-b memories. *Neural Processing Letters*. **22**:85-111.
[13] B. Cruz, H. Sossa and R. Barrón. Associative processing and pattern decomposition for pattern reconstruction. Submitted to *Neural Processing Letters*.
[14] N. Otsu (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on SMC*, **9(1)**:62-66.
[15] R. Jain *et al*. (1995). *Machine Vision*. McGraw-Hill. pp. 47-48.
[16] R. C. Gonzalez and R. E. Woods (2002). *Digital Image Processing*. Second edition. Prentice Hall, Inc. 2002.