Allograph extraction of isolated handwritten characters

Miguel L. BOTE-LORENZO, Yannis A. DIMITRIADIS and Eduardo GÓMEZ-SÁNCHEZ
School of Telecommunication Engineering, University of Valladolid
Camino del Cementerio s/n, 47011 Valladolid, Spain
mbotlor@pireo.tel.uva.es, yannis@tel.uva.es, edugom@tel.uva.es

Abstract. This paper presents a new allograph extraction method. The proposed technique is based on two clustering phases. First, a rough clustering of handwritten data is made taking into account both characters' global and local information. Next, the clustering is refined in order to obtain clusters of characters belonging to the same allograph that is finally computed. Experiments on allograph extraction made using characters from UNIPEN international database yield an allographs per character rate up to 7,2 in upper-case characters. The quality of allographs is also tested using them to initialize a handwriting recognizer achieving an average recognition rate of 90,15%.

1. Introduction

Occidental written languages use alphabets consisting of small sets of characters (e.g. the English alphabet has 26 letters). However, most characters may be written using more than one different shape or model. These models are called *allographs* (Parizeau & Plamondon, 1995) (Duneau & Dorizzi, 1994). For example, the letter {a} can be written in several different ways, such as an upper-case, a block printed, or a cursive variant. Those variants are different character allographs for the character concept {a}. The writer can generate different *instances* of a given allograph. These can be considered as a degradation of the allograph that resembles more or less accurately to the original model depending on the writer's skill and assiduity.

The aim of allograph extraction methods is to retrieve allographs from a collection of character instances, i.e. to identify the different models of a *character concept*. In terms of pattern recognition, allograph extraction can also be seen as the detection of cluster prototypes (Jain et al., 2000). On the contrary, handwriting recognition consists in labeling character instances in order to get the character concept from the instance by directly or indirectly identifying its allograph. Thus, allograph extraction and handwriting recognition are closely related. Figure 1 illustrates all these ideas.

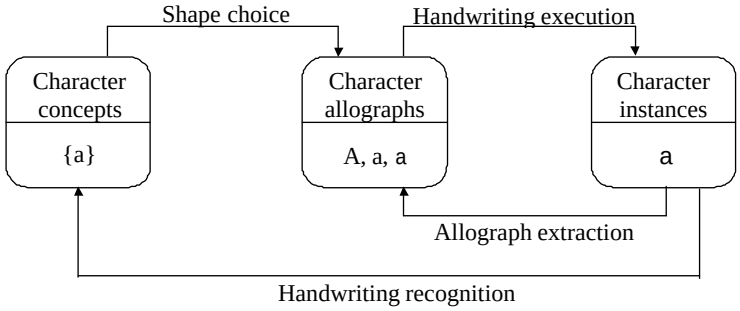

Figure 1: A conceptual model that relates generation and recognition processes through allographs.

Within this framework, allograph extraction can be used for several purposes. First, it can help to formalize the study of the different handwriting styles used by humans according to several factors like age or culture (Plamondon & Srihari, 2000). In addition, the relation between allographs and character instances is an important point for the research on handwriting generation (Parizeau et al., 1995) (Duneau et al., 1994). Finally, the construction of a small size dictionary of allographs that may also be human understandable can be useful for the development of new on-line handwriting recognition architectures (Teulings & Schomaker, 1992).

Given this background, a new allograph extraction method is proposed in this paper. This method tries to retrieve allographs by first identifying clusters of instances, and then extracting a cluster prototype. This process is performed in two stages. First, neural networks are employed to carry out a supervised clustering made according to classification criteria. Then, clustering is refined grouping character instances that were possibly generated by the same allograph. In addition, a new on-line handwriting recognition system is proposed. This system uses the extracted allographs as initial knowledge in order to compare the allographs to other sources of initialization for a recognizer.

The organization of this paper is as follows. Section 2 first describes the system used to carry out the allograph extraction procedure. Next, Section 3 presents the numeric results achieved by the allograph extraction method using isolated letters from a multi-writer database; the reconstruction of the allographs extracted for the

character {Ø} is also shown and discussed as an example. Section 4 tries to validate the obtained allographs as a source of knowledge by comparing the recognition rates achieved by a system initialized with the already mentioned allographs and some other well-known initialization methods. Finally, in Section 5 conclusions and current research are discussed.

2. Description of the allograph extraction method

If characters instances are represented using feature vectors, and these features are discriminant, these vectors will spread along the feature space forming separate clouds. Each cloud can be seen as a cluster of character instances around a prototype that in turn can be considered the generating allograph. Thus, the more the character instance resembles to the allograph, the smaller the distance between the instance and the allograph vectors will be. If the allograph is the description of an average character, it can be represented by the mean of the instance vectors belonging to the cluster. The use of the mean to compute the allograph assures the minimization of the distance between the model vector and the whole of instance vectors employed for its calculation (Devijver & Kitler , 1982).

Allograph extraction can therefore be performed in two stages: first, instances are grouped in clusters that serve for classification purposes, i.e. all instances within a cluster represent the same character concept. The number of groups created depends only on classification needs, i.e. the proximity of instances of a different character concept, and therefore each of these dusters may contain instances generated following one or more different character allographs. A second clustering is thus performed, starting from the existing clusters, but based on the search of clouds of instances instead of on classification criteria. These two steps are shown in Figure 2.

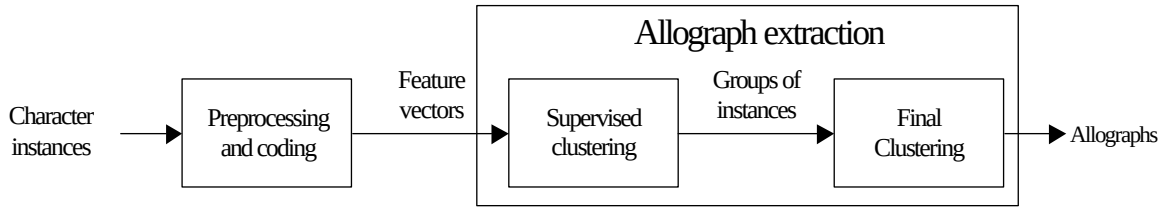

Figure 2: Block diagram of the allograph extraction method.

Prior to allograph extraction, raw handwriting data are first preprocessed and segmented into strokes according to the methods described in (Gómez-Sánchez et al., 1998) and then coded into feature vectors. The character feature vectors are made up by seven features of each character's stroke (mean x-coordinates value, mean y-coordinates value, sine and cosine of the three phases defined by three segments approaching the stroke, length and a sequential feature showing whether the stroke starts and/or ends a component or is inside the component) and a character's global feature (character's height and width ratio).

2.1 Supervised clustering stage

A rough division of the feature vectors is made in the initial clustering stage using Fuzzy ARTMAP (Carpenter et al., 1992) neural networks. The training is made setting design parameters r =0 and b =1 and carried out until null prediction error on the training data is reached. Since the number of strokes is not fixed, feature vectors are processed by a different neural network depending on their number of strokes.

After training, the *n*-stroke network will have created a number of hyperboxes in the *n*-stroke feature space. All instances within a given hyperbox have the same label, unless also contained by a smaller hyperbox. Sometimes zero volume hyperboxes containing just one feature vector are found. These hyperboxes are usually produced by noisy or extraneous samples, and thus are rejected prior to finding out the final allographs in the next stage. The output groups are thus made by the feature vectors activating the same hyperbox.

2.2 Final clustering stage

Previous stage produces groups of instance vectors according to classification criteria. However, these groups may enclose instances stemming from different allographs. To correct this, another clustering stage proceeds redistributing the instances within previously determined hyperboxes and identifying the generating allographs.

The second clustering stage incrementally builds the allograph lexicon from the existing hyperboxes. For each iteration, a number of *permanent allograph* exists, as well as a *candidate allograph* that can vary until it becomes permanent.

Initially, only one permanent allograph exists: it is calculated as the mean of the vectors clustered in the smallest hyperbox, i.e. this will be the most specific allograph.

Afterwards, the next hyperbox in ascending order of size is selected, so that its patterns can be used to create new allographs. In order to do this, one vector among them is randomly selected as a candidate allograph. Then, all the patterns closer to this candidate than to any other existing allograph can be said to have been generated by the current candidate allograph, that in turn can be recalculated again as the mean of all these vectors. Because of the random choice of the initial candidate, its new value can differ significantly. Therefore, all patterns in the processed hyperbox are newly assigned to the closest allograph, permanent or candidate, and another more precise value for the candidate allograph is calculated. Eventually, this process stops when the variation between two successive values for the candidate allograph is small. When this occurs, the allograph becomes permanent and it is added to the allograph lexicon.

However, it may well happen that not all vectors in the processed hyperbox select (are closer to) the newly generated allograph. If some vectors remain, they are closer to another already existing allograph that may have the correct label associated. If however they are closer to an allograph with a different class label, it means that a new allograph should be created. This new allograph can be seen as the generator of another "cloud" of character instances that significantly differ from others describing the same character concept.

To calculate this new allograph, a vector is randomly selected among the remaining patterns of the hyperbox being processed. The allograph is then refined through the same iterative process described above, until it becomes a permanent allograph. The process continues, so that even more than two allographs could actually be extracted from a given hyperbox.

When all the patterns in a given hyperbox are finally processed, the next larger hyperbox is then taken. The procedure continues until all existing hyperboxes have been processed. Therefore, this algorithm starts from very specific allographs, continuing to more general ones, while also increasing the generality of those already computed.

In addition, processing the groups of feature vectors in ascending order of the size of their associated hyperboxes allows to deal first with compact hyperboxes that contain only one cloud of vectors (i.e. one allograph). Since larger hyperboxes are processed later, they may contain some of the previous hyperboxes, and thus already generated allographs. The detection of instances belonging to the same group and laying in both sides of a previously computed allograph lets us determine the existence of different allographs within the group.

A step-by-step description of this algorithm can be found in Appendix A.

3 Allograph extraction results

The proposed allograph extraction method has been evaluated on the UNIPEN (Guyon et al., 1994) data sets, using versions 2 and 7. This ensures a large amount of data, author-independence and comparability to other systems. The test was carried out independently using three different sets of isolated characters of both versions: digits, upper-case letters and lower case letters. The number of labels used in each set is 10, 26 and 26 respectively according to the English alphabet. Every set was divided in two subsets of the same size warranting the presence of samples by any writer in both of them. The first subset was used to perform the allograph extraction experiments and to train the character recognition systems. The second was only used in the recognition experiments.

The number of networks employed in the system is set to 6 for digits and upper-case letters experiments and to 7 in lower-case letter. Characters having a larger number of strokes are considered as segmentation errors due to the preprocessing stage. However, the number of bad-segmented characters was never more than the 0,2% of the total amount of data used for the test.

Figure 3 shows an example of the allographs extracted for the character {Ø} (zero) from UNIPEN version 2 data set. Allographs in Figure 3 are sorted by the number of strokes; first row contains one-stroke allographs, second contains two-stroke allographs, etc. The system extracted 16 allographs including the two allographs expected *a priori*: the normal zero (0) and the slashed zero (Ø). Most of the allographs extracted are perfectly recognizable at first glance.

It is noteworthy that a number of allographs correspond to similar graphical representations of the characters, but to different ways of generation. For example, the first and the last two-stroke zeros have been made following the same model {Ø}; however, they have been drawn clockwise and counterclockwise respectively. This fact clearly relates allographs to the studies on handwriting generation.

It can be seen that the same execution of the same allograph is found for different number of strokes. Therefore, the segmentation method should be revised, since ideally all instances generated from a given allograph should have been segmented into the same number of strokes. On the other hand, very similar allographs are sometimes identified having the same *n*-stroke segmentation. The improvement of the proposed clustering methods could overcome this problem. Finally, it must be said that the example shown and discussed here is one of the worst cases found for the version 2 digits data set experiments results.

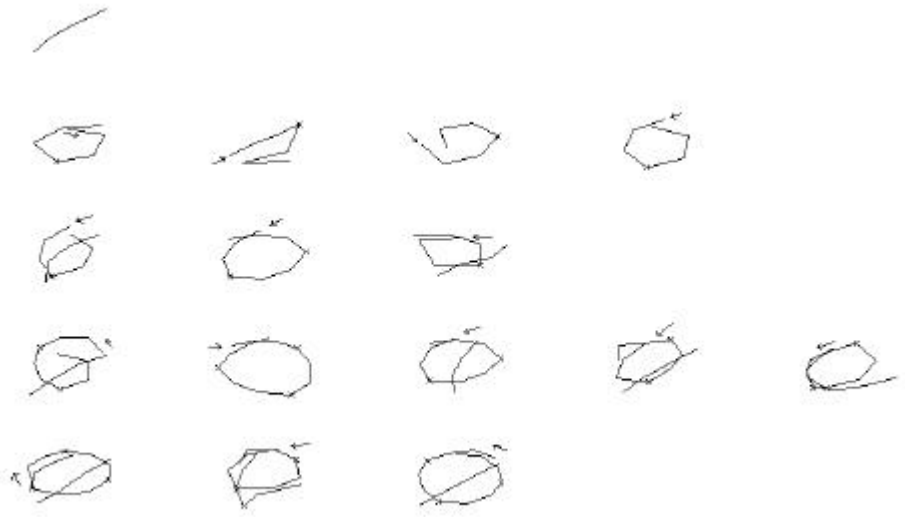

Figure 3: Reconstruction of the allographs extracted for the '0' digit of the version 2 UNIPEN data set. The arrows mark the beginning of the characters' first stroke. Segmentation points are marked using a cross.

The numeric results of the allograph extraction tests are shown in Table 1. The number of allographs per character found for version 2 digits and upper-case letters seems to be reasonable, although it could be improved by the elimination of the repeated allographs, as discussed in previous paragraph. This number increases for lower-case letters in version 2 given their greater variability. Allographs per character rates also increase for version 7 tests. This is due to the larger amount of data. It can also be observed that the total number of allographs extracted for version 7 sets is larger than for those of version 2. Ideally, the system should extract the same number of allographs regardless of the number of instance vectors used. However, the number of extracted allographs achieved is low enough to deal with, and may allow the definition of an allographs' dictionary.

	Version 2			Version 7		
	Digits	Upper-case letters	Lower-case letters	Digits	Upper-case letters	Lower-case letters
Number of instance vectors	1916	2109	6100	7245	12105	23710
Number of allographs	108	186	558	278	723	1577
Allographs per character ratio	10,8	7,2	21,5	27,8	27,8	60,7

Table 1: Results of the allograph extraction test.

4 Recognition results using extracted allographs

In the previous section, a qualitative discussion on the quality of the extracted allographs was made. However, if the extracted allographs are used as knowledge employed to initialize a handwriting recognizer, the achieved classification rates can be considered a quantitative approach on the evaluation of the allographs' quality.

Since the allographs, as constructed here, are prototypes close to training instances, it seems reasonable to apply recognizers based on the comparison of distances between prototypes and test instances. For this purpose, LVQ codebooks (Kohonen et al., 1995) can be used, existing other initialization methods that can be used for comparison purposes.

The handwriting recognizer thus employed consists of a series of LVQ codebooks (Kohonen et al., 1995). A different codebook is employed to classify the characters according to their number of strokes. Three different methods were used to initialize the LVQ codebooks. The first one simply takes the extracted allographs as the initial codebook vectors to be trained later. The other two methods, called *propinit* and *eveninit* (Kohonen et al., 1995), choose randomly the initial codebook entries from the training data set, making the number of entries allocated to each class be proportional or equal, respectively. The number of initial entries must be set *a priori* for the *propinit* and *eveninit* initialization methods. In order to make the comparisons as fair as possible the number of initial vectors for each codebook and data set was fixed to the number of allographs with the corresponding number of strokes extracted from the same data set. In all cases, the OLVQ1 algorithm (Kohonen et al., 1995) was employed to carry out the training.

Two different experiments have been made using the three initialization methods already mentioned with the different data sets. First, the system was tested without any kind of training. Second, the test was carried out after training the recognizer. The chosen training lengths were always 40 times the total number of codebook vectors.

The initial value of the parameter α was set to 0,3 for all codebook entries. The achieved results are shown in Table 2.

	Version 2			Version 7		
	Digits	Upper-case letters	Lower-case letters	Digits	Upper-case letters	Lower-case letters
Number of training vectors	1916	2109	6100	7245	12105	23710
Number of test vectors	1917	2109	6101	7242	12104	23714
No train. allograph init. recog. Rate	**92,80**	**86,96**	**83,87**	**91,12**	**87,28**	**83,53**
No train. *propinit* init. recog. Rate	75,85	70,65	67,30	83,97	75,78	75,50
No train. *eveninit* init. recog. Rate	75,74	58,04	65,25	79,51	70,86	70,63
Allograph initialization recog. Rate	**93,84**	**87,81**	**86,76**	**95,04**	**89,68**	**87,76**
Propinit initialization recog. Rate	88,47	78,38	76,71	89,23	80,92	83,49
Eveninit initialization recog. Rate	85,08	73,11	75,40	89,42	80,02	82,28

Table 2: Results of recognition experiments using different initialization methods.

It is noteworthy the fact that achieved recognition rates employing the allograph initialization *before* training the system are significantly higher than using the *propinit* and *eveninit* methods in all cases. This is because the codebook entries set by the allograph initialization method are found in the middle of every cluster. On the contrary, the *propinit* and *eveninit* methods, given its random nature, do not assure the existence of an entry in every cluster found nor the placement in the middle of the cloud.

This conclusion is supported by the results of the second experiment. The recognition rates using allograph initialization increase slightly after carrying out the training. Since the allographs are computed as the mean of the cluster vectors, the initial codebook vectors are already quite well placed from the classification point of view. The training just contributes to refine the allograph positions in order to minimize the classification error. The increase of recognition rates after training using *propinit* and *eveninit* initialization methods is much higher. In this case, the training phase moves the codebook entries towards more suitable positions in the future space according to classification criteria. However, the obtained recognition rates are still lower than using the allograph initialization. This is again because there will not always be an entry placed in every cluster found in the training data.

A new experiment can be made in order to both have a measure of the allographs' quality and try to decrease the proliferation of allographs. The experiment consists in successively removing the allographs having the smaller number of character instances related to them. The recognition system is initialized using the remaining allographs and then trained following criteria mentioned previously. A comparison with the *propinit* and *eveninit* methods can be made too if we use these methods to initialize the system setting the number of initial codebook vectors to the number of remaining allographs.

This experiment has only been made for version 7 lower-case letters, the most difficult case from the classification point of view. Removing the allographs representing ten or less instances reduces the number of models from 1577 to 297 while the recognition rate decreases from 87,76% to 81,66%. If the test is carried out using 297 initial codebook entries generated by the *propinit* and *eveninit* algorithm, the achieved recognition rates are 70,22% and 62,22% respectively. The decrease observed is thus more significant.

Finally, it must be remarked that the proposed handwriting recognition system based on allograph initialization shows a good performance. This point can be confirmed by comparing the recognition rates with those achieved with the recognition system presented in (Gómez-Sánchez et al., 2001). The best recognition rates on similar version 2 data sets using a neuro-fuzzy classifier are 85,5%, 76,39% and 59,57% for digits, upper-case letters and lower case letters respectively. All these marks are exceeded by the system presented in this paper.

5 Conclusions

The study of allograph extraction methods may be considered of interest for three main reasons. First, it may help to tackle the study of handwriting styles. Second, establishing the relationship between character instances and allographs may also help to advance in the field of handwriting generation comprehension. Third, the construction of a small size allographs' dictionary is a desirable objective

This paper has presented a new method for the extraction of allographs from isolated handwritten characters. This is achieved by identifying clusters of character instances and then extracting a prototype for them. This prototype is calculated as the mean of the instances. With this approach, instances can be seen as a degradation of the allograph. The proposed system identifies the clusters, and thus the allographs, in two stages. In the first one, clustering is performed according to a classification method. In the second, a refinement of previous clusters is made, and allographs are finally extracted.

This method has been validated on the UNIPEN handwriting database, showing that a reasonable number of

allographs can be extracted from a large multi-writer amount of samples. Furthermore, the number of extracted allographs is affordable to build a lexicon, though reducing this number would be a desirable objective. The observation of reconstructed allographs shows that repeated allographs are found thus producing a proliferation problem. The improvement of the preprocessing stage and of the clustering algorithm should eventually overcome this problem. The quality of the extracted allographs has also been validated by their use for the initialization of a LVQ classifier. Experiments show that allographs provide much more initial knowledge than other existing initialization methods. Besides, the character recognizer yields better recognition results than others previously reported in literature.

Appendix A

Let R_1, R_2, ..., R_N be the hyperboxes created in the supervised clustering stage sorted in size so that $|R_1|<|R_2|,...,<|R_N|$. Let r be an index indicating the hyperbox being processed. The vectors associated to hyperbox R_r are denoted $\iota_r=\{\boldsymbol{v}_1^r,\boldsymbol{v}_2^r,...,\boldsymbol{v}_{Mr}^r\}$. Let T denote the collection of allographs, initially empty, i.e. T=∅. Then,

Step 1- Take r=1 (the smallest hyperbox is processed first), and compute the first allograph $\boldsymbol{a}_1$ as the mean of the vectors related to R_1.

Step 2- Select next category in size, i.e. $r=r+1$. Mark all vectors in ι_r as not related to any allograph.

Step 3- Select any random vector from those in ι_r not related to any allograph, and let it be the candidate allograph $\boldsymbol{a}_c$.

Step 4- Compute the distances from every vector $\boldsymbol{v}_i^r$ to $\boldsymbol{a}_c$ and all other allographs in T.

Case a- Vector $\boldsymbol{v}_i^r$ is closer to $\boldsymbol{a}_c$ than to any other allograph; or vector $\boldsymbol{v}_i^r$ is closer to some permanent allograph $\boldsymbol{a}_j$ than to $\boldsymbol{a}_c$ but $\|\boldsymbol{a}_c-\boldsymbol{v}_i^r\|<\|\boldsymbol{a}_c-\boldsymbol{a}_j\|$ and $\boldsymbol{a}_j$ *has different label as* hyperbox R_r, then mark vector $\boldsymbol{v}_i^r$ as related to allograph $\boldsymbol{a}_c$.

Case b- Vector $\boldsymbol{v}_i^r$ is closer to some permanent allograph $\boldsymbol{a}_j$ than to $\boldsymbol{a}_c$ and allograph $\boldsymbol{a}_j$ has the same associated label as hyperbox R_r then mark vector $\boldsymbol{v}_i^r$ as related to allograph $\boldsymbol{a}_j$.

Case c- Vector $\boldsymbol{v}_i^r$ is closer to some permanent allograph $\boldsymbol{a}_k$ than to $\boldsymbol{a}_c$, allograph $\boldsymbol{a}_k$ has different associated label as hyperbox R_r and $\|\boldsymbol{a}_c-\boldsymbol{v}_i^r\|>\|\boldsymbol{a}_c-\boldsymbol{a}_j\|$, then do not relate vector $\boldsymbol{v}_i^r$ to any allograph.

Step 5- Compute a new value for allograph $\boldsymbol{a}_c$ as the mean of all vectors in ι_r associated to it. If $\boldsymbol{a}_c^{old}$ is the previous value of this allograph, then

Case a- $\|\boldsymbol{a}_c-\boldsymbol{a}_c^{old}\|>\theta$ mark all vectors in ι_r as not associated to any allograph, and go to **Step 4**.

Case b- $\|\boldsymbol{a}_c-\boldsymbol{a}_c^{old}\|<\theta$ remove all vectors associated to $\boldsymbol{a}_c$ from ι_r. If ι_r still contains vectors, and they are not related to any allograph, go to **Step 2** to process next category. Otherwise, go to **Step 3** to process the remaining vectors in ι_r.

References

Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., & Rosen, D.B. (1992, September). Fuzzy ARTMAP: A Neural Network Architecture for Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks, 3,* (5), 698-713.

Devijver, P.A., & Kittler, J. (1982). *Pattern Recognition: A Statistical Approach*. London: Prentice-Hall International.

Duneau, L., & Dorizzi, B. (1994). Incremental Building of an Allograph Lexicon. In C. Fanre, P. Kenss, G. Lorette, & A. Vinter (Eds.), Advances *in Handwriting and Drawing: a Multidisciplinary Approach*. (pp. 39-53). Paris: Europia.

Gómez-Sánchez, E., Dimitriadis, Y.A., Sánchez-Reyes Mas, M., Sánchez García, P., Cano Izquierdo, J.M., & López Coronado, J. (2001). On-Line Character Analysis and Recognition with Fuzzy Neural Networks. *Intelligent Automation and Soft Computing, 7,* (3). In Press.

Gómez-Sánchez, E., Gago González, J.Á., Dimitriadis, Y.A., Cano Izquierdo, J.M., & López Coronado, J. (1998, March). Experimental Study of a Novel Neuro-Fuzzy System for On-Line Handwritten UNIPEN Digit Recognition. *Pattern Recognition Letters, 19,* (3), 357-364.

Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., & Janet, S. (1994, October). UNIPEN Project of On-Line Data Exchange and Recognizer Benchmarks. In *Proceedings of the 12th International Conference on Pattern Recognition* (pp. 9-13). Jerusalem, Israel.

Jain, A.K., Duin, R.P.W., & Mao, J. (2000, January). Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern* Analysis *and Machine Intelligence, 22,* (1), 4-37.

Kohonen, T., Kangas, J., Laaksonen, J., & Torkkola, K. (1995). LVQ-PAK: The Learning Vector Quantization Program *Package*. Helsinki University of Technology, Finland, http://www.cis.hut.fi/research/som_lvq_pak.shtml.

Parizeau, M., & Plamondon, R. (1995, July). A Fuzzy-Syntactic Approach to Allograph Modeling for Cursive Script *Recognition*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17,* (7), 702-712.

Plamondon, R., & Srihari, S.N. (2000, January). On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22,* (1), 63-84.

Teulings, H.L., & Schomaker, L. (1992). Unsupervised Learning of Prototype Allographs in Cursive Script Recognition. In S. *Impedovo* & J. C. Simon (Eds.), *From Pixels to Features III: Frontiers in Handwriting Recognition*. (pp. 61-75). Elsevier Science Publishers B.V.