Learning to Segment Cursive Words using Isolated Characters*

Jean-François Hébert, Marc Parizeau
Laboratoire de vision et systèmes numériques
Université Laval
Ste-Foy (Qc), Canada, G1K 7P4
{jfhebert,parizeau}@gel.ulaval.ca

Nadia Ghazzali
Dept. de mathématiques et de statistique
Université Laval
Ste-Foy (Qc), Canada, G1K 7P4
ghazzali@mat.ulaval.ca

Abstract

This paper presents a new strategy for isolating handwritten characters in cursive words without making an explicit a priori segmentation of the script, and without imposing any lexical or other linguistic constraints. Furthermore, this approach can be completely trained using data sets of isolated characters only.

The main idea behind this strategy is to have a window of attention moving around in the cursive word, searching for instances of known characters. If one assumes that the current window contains some significant part of a character, then the problem is to translate and scale the window of attention in such a way that it converges to the bounding box of that character. This process is implemented using both a detector *network and a set of* locator *networks. The detector network is responsible for recognizing whole characters of any class and thus for stopping the iterative process, whereas a locator network is assigned the task of recognizing the crucial parts of a given character class and producing the corresponding transformation parameters for the window. The feasibility of this process is shown through experiments using the UNIPEN database of on-line scripts.*

1 Introduction

The automatic recognition of on-line cursive script is a difficult pattern recognition problem[1, 2], mainly because of the great variations encountered in different handwriting styles, but also because of the so-called segmentation/recognition dilemma where characters need both to be segmented before they can be recognized, and recognized before they can be segmented. Many recognition systems have been designed to tackle these difficulties using various methods and techniques. Some of the more recent works include the use of neural networks [3, 4, 5], hidden markov models [6, 7] and formal grammars [8].

*This work was supported in part by NSERC and FCAR grants to M. Parizeau and in part by an NSERC grant to N. Ghazzali.

However, most of these systems deal with the segmentation/recognition dilemma by, on the one hand, oversegmentation of the scripts in smaller unit than characters (usually graphemes) and analysis of all possible segmentation paths, and, on the second hand, by imposing lexical constraints in order to limit combinatorial explosion. The object of this paper is to present a somewhat different strategy where no explicit a priori segmentation is made and where no lexical constraints are a priori imposed.

The main idea behind this strategy, is to be able to initialize a window of attention somewhere near a character in a cursive word, and have a segmentation process fine tune the position and size of this window in such a way that it searches for the bounding box of this character. By repeating this process on a sufficiently large set of initial points, one could then build a segmentation graph for the cursive word which could be interpreted using adjacency constraints [10], lexical knowledge, or both. It is hoped that using such a strategy, a complete and flexible cursive script recognition system can be built, although no claims to such a system is made in this paper (work on this is still under way). What is claimed in this paper, however, is the feasibility of this new segmentation strategy and that the underlying process can be trained automatically using only isolated characters.

In previous work, Suen et al. [9] have already demonstrated that handwritten alphanumeric characters are made up of *crucial parts that tend to preserve the invariant characteristics and exhibit the distinctive features of characters*. It is our objective to exploit these crucial parts in order to create a more flexible character segmentation/recognition algorithm.

The rest of this paper is organized as follows. First, Section 2 presents a global overview of our segmentation process. Then, Section 3 to Section 6 describe in details each of its four components. Finally, experimental results are presented and discussed in Section 7.

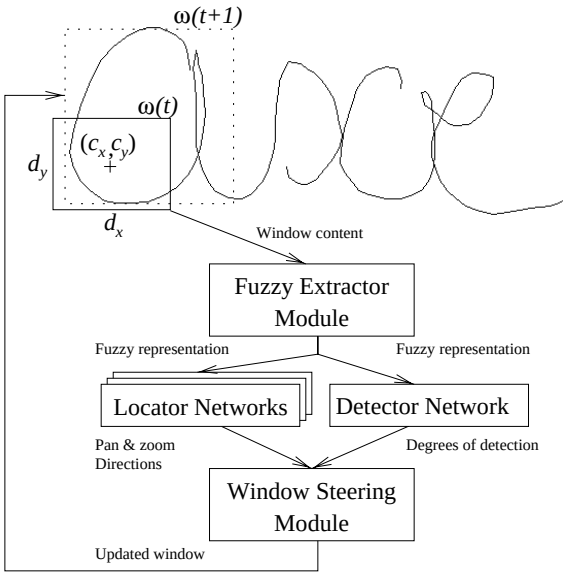

Figure 1: *Character segmentation process.*

2 Segmentation process overview

The objective of our segmentation process is to find all instances of character classes in a cursive word. This search, conducted on a class by class basis, is mainly based on an iterative positioning process that controls the panning and zooming of a window of attention. As illustrated in Figure 1, four distinct components form this character segmentation process. An overview of each component is presented in this section, and a more detailed description can be found in the following sections.

The first component of this segmentation process is the *fuzzy extractor module* (c.f. section 3) that computes a representation that maps the handwriting content of the window of attention into a fixed dimension fuzzy vector space. The window of attention is characterized by a quadruple $\omega(t) = [c_x(t), c_y(t), d_x(t), d_y(t)]$, where at iteration $t \geq 0$, $c_x(t)$ and $c_y(t)$ represent respectively the X and Y center coordinates of $\omega(t)$, and where $d_x(t)$ and $d_y(t)$ are its horizontal and vertical sizes.

Once extracted, the fuzzy vector which describes the handwriting content of $\omega(t)$ is passed on to both a *detector network* (c.f. section 4) and a set of *locator networks* (c.f. section 5). Each locator network is assigned to a specific character class, whereas only one detector network is used for all character classes.

The detector and the locator networks perform very distinct tasks. The goal of the detector network is to assess whether or not the current window of attention $\omega(t)$ encompasses an instance of any character class. The locator network assumes that the window of attention encompasses at least part of a given character, and using the content of $\omega(t)$, it then determines what should be the best transformation to apply to $\omega(t)$ in order to produce an $\omega(t+1)$ that better segments this character instance. This transformation may consist of both pan and zoom operations. If the window of attention indeed encompasses part of a character, then we expect at the next iteration to increase the degree of detection of its character class. On the contrary, for the other character classes, we expect to decrease the degrees of detection.

At the end of an iteration, the *window steering module* (c.f. section 6) considers the output of both preceding networks to decide if the locating process should be carried on with additional iterations. This iterative process is interrupted either if an instance of the searched character class is considered as being correctly segmented, or if nothing is found after a sufficient number of iterations. In both cases, the segmentation process is started again with a new initial window of attention. However, if the steering module considers that the process should continue, the parameters of the window of attention are updated using the outputs of both the locator and detector networks.

3 Fuzzy extractor module

A fuzzy geometric representation is used to map the handwriting content of window $\omega(t)$ into a fixed dimension fuzzy vector space. This fuzzy representation is obtained by first segmenting the original handwritten script into a sequence of elementary "strokes" $\mathcal{S} = s_1, s_2, \ldots, s_q$, where a stroke s_i, $i = 1, 2, \ldots, q$, is modeled as a circular arc [11]. Thereafter, window $\omega(t)$ is positioned over the segmented script and decomposed into a 3×2 grid of rectangular regions[1], as shown in Figure 2. Each stroke (or sub-stroke) of $\mathcal{S}$ found in each region is then fuzzified according to its orientation and curvature using some predefined fuzzy sets. Fuzzified strokes are combined using fuzzy operators to form regional fuzzy vectors, and these vectors are simply concatenated to produce the final representation denoted $\tilde{\mathcal{S}}(\omega(t))$.

For more details about the fuzzy sets and operators, the reader is referred to Hébert et al. [12], where this fuzzy representation was used to recognize isolated digits.

4 Detector network

Any classifier that can be trained can be used as the detector network for the character segmentation process. Indeed, since the task of the detector network is to determine if the current content of the window of attention looks like an instance of any character class, the only constraint is to be

[1] Other configurations are obviously possible, but this one has already shown good results [12].

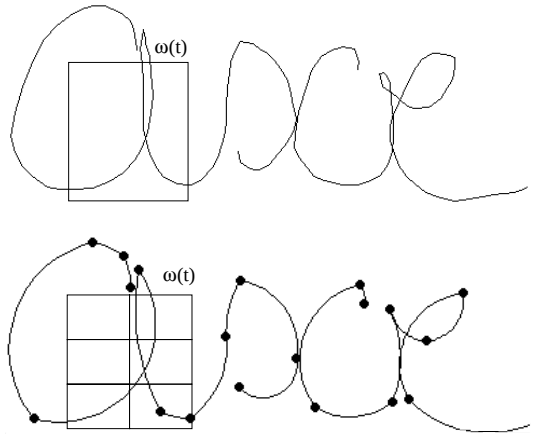

Figure 2: *Original (up) and segmented word (down) "axe". The window of attention $\omega(t)$ is shown on the segmented word with its 3×2 grid of regions*

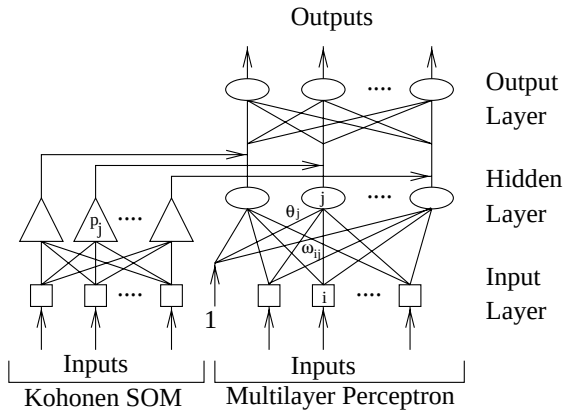

Figure 3: *KP network architecture.*

able to train the classifier on a data set of isolated characters. For the experiment described in this paper (see Section 7), we have used the KP neural network [13] that combines (see Figure 3) a Kohonen self-organizing feature map (the K-net) with a multilayer feedforward Perceptron (the P-net). The idea behind this hybrid configuration is to benefit from the well-known modeling capacities of the K-net to help determine which cluster of the P-net should contribute more than others when producing the network outputs. Another interesting feature of the KP network is its ability to learn incrementally [13]. Moreover, high recognition rates (96.3%) were reported on isolated digits recognition [12] using the international UNIPEN database of on-line scripts [14]. More results for the case of isolated lowercase letters are also given in Section 7.

Let $\mathbf{C} = \{C_1, C_2, \ldots, C_p\}$ denote the set of p character classes, where C_i, $i = 1, 2, \ldots, p$, designates the set of character samples for class i. During training, the fuzzy extractor module is applied on the content of a window of attention placed over the bounding box of each sample character. After training, given a certain window content $\widetilde{\mathcal{S}}(\omega(t))$, the output of the detector network is a set $\Delta(\widetilde{\mathcal{S}}(\omega(t))) = \{\delta_1, \delta_2, \ldots, \delta_p\}$ of detection degrees associated to each character class, with $\delta_i \in [0, 1]$, $i = 1, 2, \ldots, p$.

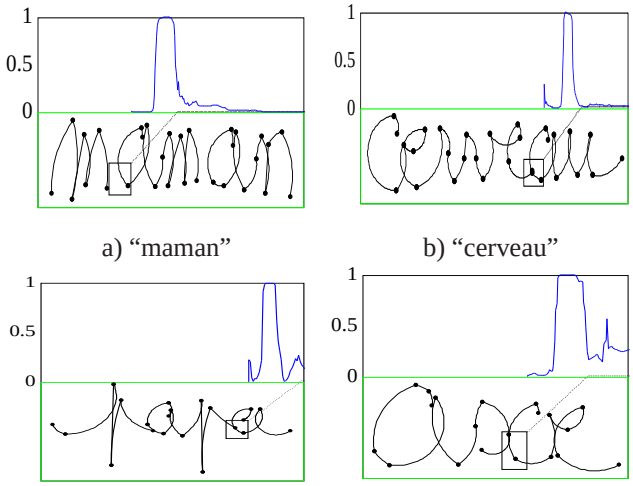

Figure 4: *Detection measures for character class 'a' using windows of increasing sizes.*

In order to illustrate the ability of an already trained KP network to detect instances of a character class, Figure 4 shows some detection results for letter 'a' in several French words. In each example, the initial window of attention was manually positioned and then enlarged rightward and upward (keeping the lower left corner of the window fixed) along the diagonal dotted lines in the figures. Above each word, the graph of the detection degree associated with letter 'a' is shown according to the position of the upper right corner of the window.

In Figures 4a, 4b, and 4c, the peaks in the detection signal correctly identify the character instances when the window of attention effectively surrounds them. One should observed that in all these examples, the KP network succeeded in detecting the 'a' instances even though ligatures link letters together. But of course, the detector network might also find false instances of a character class if the content of the window of attention looks like the searched character class. That case occurs in Figure 4d where an 'a' letter is detected even though the word does not include such a letter at that position. However, this should not be considered an error since the corresponding window content can indeed be interpreted as an instance of letter 'a' if one does not consider any other contextual or lexical informations.

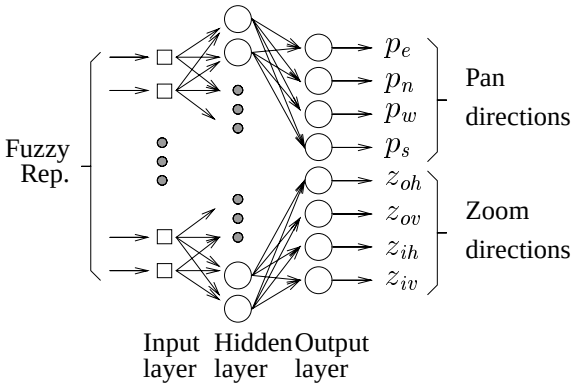

Figure 5: *Structure of the locator network.*

5 Locator networks

In our current implementation, the locator networks consist of simple multilayer perceptrons [15], as illustrated in Figure 5. The number of neurons on the hidden layer may vary, but the output layer is always formed of eight neurons. Among the output neurons, the first four are related to desired pan directions (p_e = east, p_n = north, p_w = west and p_s = south), while the last four correspond to zoom directions (z_{oh} = out-horizontal, z_{ov} = out-vertical, z_{ih} = in-horizontal, z_{iv} = in-vertical).

For training a locator network, different windows of attention are positioned over distinct parts of isolated characters. The objective for the locator network is to learn how to pan and zoom each window of attention in such a way that it can tend to encompass the full character. The fuzzy representation $\widetilde{\mathcal{S}}(\omega(t))$ forms an input datum for which the desired outputs of pan and zoom are restricted to take only binary values (the magnitude of the pan and zoom operations will be determined by the *window steering module*).

A standard set of n training attention windows $\Omega = \{\omega_1(t), \omega_2(t), \ldots, \omega_n(t)\}$ is defined for all locator networks. In a given segmented character $\mathcal{S}$ belonging to character class k, the set of fuzzy vectors $\{\widetilde{\mathcal{S}}(\omega_1(t)), \widetilde{\mathcal{S}}(\omega_2(t)), \ldots, \widetilde{\mathcal{S}}(\omega_n(t))\}$ constitutes n training data for the locator network assigned to class k. Thus, a training database composed of $|C_k|$ character samples for each class k will lead to $n \times |C_k|$ training data for the corresponding locator network.

Our choice of Ω is based on the work of Suen & al. [9] who defined the concept of "crucial parts". The first subset of Ω is thus based directly on the three partitions that were found to contain the most crucial parts of alphanumeric characters. These windows, denoted $\omega_1(t), \omega_2(t), \ldots, \omega_8(t)$, are illustrated in Figure 6: a 2×2 partition (6a), a 1×2 partition (6b) and a 2×1 partition (6c).

The second subset of Ω stems from the first but is designed to also include some blank areas around the characters. These additional windows ($\omega_9(t), \omega_{10}(t), \ldots, \omega_{23}(t)$) are illustrated in Figure 7.

6 Window steering module

Once the detector and the locator networks are trained over their respective training sets, the segmentation process may start to search for instances of characters. The steering module is responsible for updating the parameters of the current window of attention using the outputs of both the locator and detector networks. The output of the locator network specifies the type of transformation that should be applied to the window of attention. However, it only indicates a panning and zooming direction. It is the steering module that modulates this information using the output of the detector network.

For each character class k, the updating equations for the parameters of window $\omega(t) = [c_x(t), c_y(t), d_x(t), d_y(t)]$ are defined as follows :

$$\omega(t{+}1): \begin{bmatrix} c_x(t) + (p_e{-}p_w)[1{-}\delta_k(\omega(t))] \exp(\frac{-d_x(t)}{w_k}) d_x(t), \\ c_y(t) + (p_n{-}p_s)[1{-}\delta_k(\omega(t))] \exp(\frac{-d_y(t)}{h_k}) d_y(t), \\ d_x(t) + (z_{oh}{-}z_{ih})[1{-}\delta_k(\omega(t))] \exp(\frac{-d_x(t)}{w_k}) d_x(t), \\ d_y(t) + (z_{ov}{-}z_{iv})[1{-}\delta_k(\omega(t))] \exp(\frac{-d_y(t)}{h_k}) d_y(t) \end{bmatrix} \quad (1)$$

where p_e, p_w, p_n and p_s are the panning outputs of the locator network, z_{oh}, z_{ih}, z_{ov} and z_{iv} are its zooming outputs, δ_k is the output of the detector network for class k, and where h_k and w_k are the expected height and width of characters in class k. The value of these last two parameters can be estimated from the cursive word using a standard baseline extraction method [16]. Their exact value is not critical. In the above equations, three multiplicative factors determine the magnitude of the window transformation: the first is the direction of change indicated by the locator network, the second takes into account the output of the detector network, and the third weights the current window size relative to its expected size.

The search process stops either when $\delta_k(\omega(t)) \geq \delta_{min}$ and $\delta_k(\omega(t)) - \delta_k(\omega(t-1) < 0$, or when $t > t_{max}$. Threshold δ_{min} is the minimum degree of detection for considering that $\omega(t)$ as converge, whereas t_{max} is the maximum number of iterations allowed before considering that no characters can be found.

7 Experiments and results

The experiments presented in this section are divided in two sets. In the first set, recognition experiments are conducted on isolated lower case characters in order to evaluate the

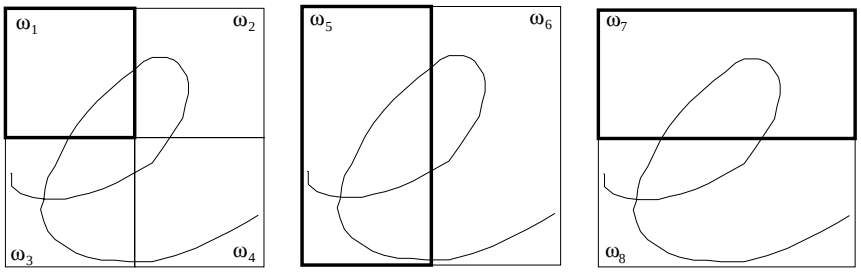

Figure 6: *Subset* $\{\omega_1(t), \omega_2(t), \ldots, \omega_8(t)\}$ *of* Ω.

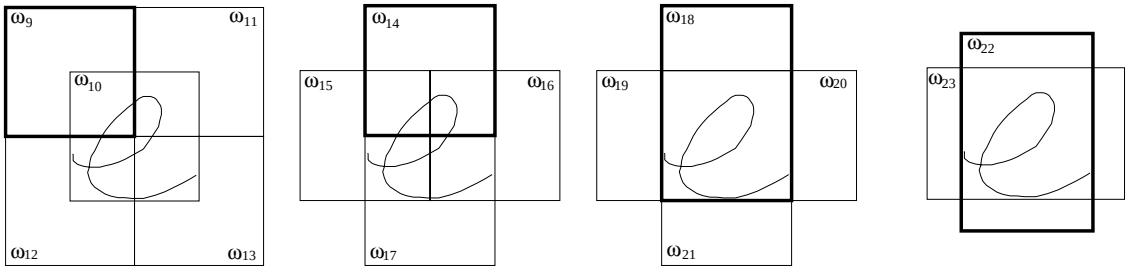

Figure 7: *Subset* $\{\omega_9(t), \omega_{10}(t), \ldots, \omega_{23}(t)\}$ *of* Ω.

nominal performance of the detector network. Indeed, the segmentation process directly depends on the ability of the detector network, in our case a KP network, to recognize character instances. In the second set of experiments, the performances of the segmentation process itself are measured in regards to its ability to efficiently segment characters in cursive words under different initial conditions.

7.1 Isolated character recognition

The detector network was trained using the international UNIPEN database for on-line handwritten scripts [14]. More precisely, Section 1c of data set Train-R01/V07 was used. This section contains $61\,539$ lowercase characters. Then, the detector network was tested on the $37\,470$ lowercase characters of data set DevTest-R02/V02. The reader should note that, except for some empty segments, none of the characters found in these data sets were removed, even though many of them are either mislabeled, missegmented or very badly written. Table 1 gives the recognition rates obtained by the KP network according to the first five hypotheses. For these experiments, each character was decomposed into a 3×2 grid of rectangular regions in order to compute a 49-dimension fuzzy vector using the fuzzy sets already defined in Hébert et al. [12]. A $49 \times (20 \times 20) \times 26$ KP neural network was trained using $3\,000\,000$ iterations for the K-net and 200 epochs for the P-net.

Table 1: *Recognition rates for the isolated lower case characters of DevTest-R02/V02 (Section 1c). Rates are given for Top-1 to Top-5 hypotheses.*

Top-1	Top-2	Top-3	Top-4	Top-5
81.2%	89.6%	92.4%	94.0%	94.9%

Overall the results given in Table 1 are good, especially when one considers that 19% of all the confusions that arise for the Top-1 hypothesis come from only 10 pairs of letter classes among the 650 possible pairs. For example, letter pairs ‘i’ and ‘l’, ‘m’ and ‘n’, ‘t’ and ‘f’, and ‘y’ and ‘g’ are often very hard to distinguish because of very similar shapes. They produce the highest number of confusions for the network. Some confused letters are shown in Figure 8.

7.2 Character segmentation

In order to quantify the performance of our segmentation process, data sets of words with known character positions and size are needed. Unfortunately, these informations are in general not available in existing word test sets such as the UNIPEN databases. Therefore, synthetic words have been constructed by concatenating randomly selected characters from the UNIPEN DevTest-R02/V02 (section 1c). A cur-

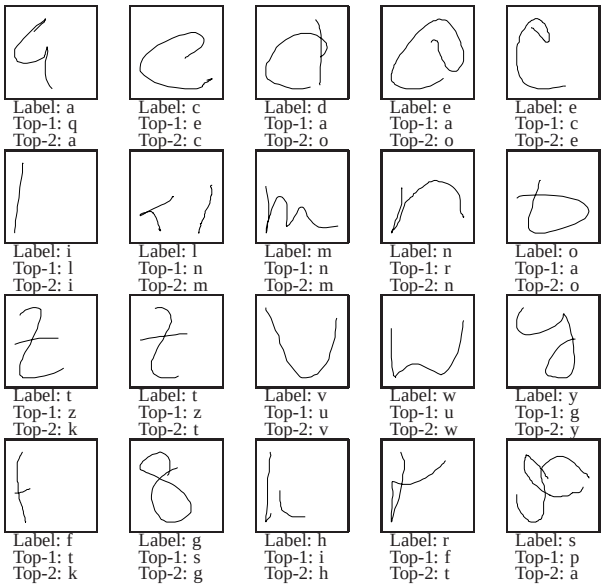

Figure 8: *Examples of confused letters with their label and the Top-1 and Top-2 hypotheses produced by the detector.*

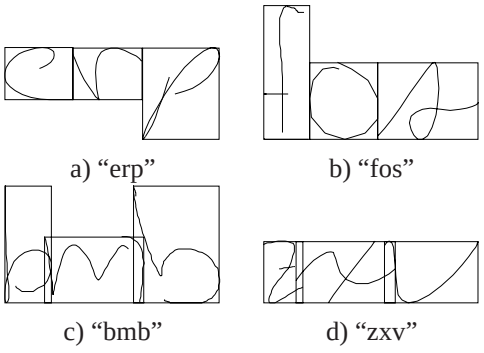

Figure 9: *Examples of synthetic words with a), b) no overlap and c), d) a 5% overlapping.*

sive effect is also simulated by overlapping the characters slightly. Figure 9 gives some examples with and without overlapping.

For each class k, a test set $\Gamma_k = \{\gamma_{k1}, \gamma_{k2}, \ldots, \gamma_{k100}\}$ of 100 synthetic words is constructed according to the following pattern. For $\gamma_{kj} \in \Gamma_k$, the middle letter is always a randomly selected instance of C_k, whereas the first and third characters are randomly selected instances of any other classes. We call the middle letter of γ_{kj} the reference letter. To introduce a cursive effect within γ_{kj}, a 5% overlapping factor is forced between each pair of adjacent characters. The segmentation experiments which are then conducted on each word $\gamma_{kj} \in \Gamma_k$ concern the segmentation of the reference letter, that is, the only instance of C_k in γ_{kj}.

In order to measure the ability of the segmentation process to converge towards the reference letter of a synthetic word for various situations, nine non-overlapping initial windows $\omega_1(0), \omega_2(0), \ldots, \omega_9(0)$ are positioned in the neighborhood of the reference letter as illustrated in Figure 10. The dimensions of each initial window are the same and correspond to half of the dimensions of the reference letter along both the X and Y axes. Thus, these initial windows of attention cover all together a total area which is 2.25 times greater than the area covered by the reference letter. Moreover, only one initial window ($\omega_5(0)$) is completely included inside the reference letter, the other ones being included only for either 50% ($\omega_2(0), \omega_4(0), \omega_6(0), \omega_8(0)$) or 25% (($\omega_1(0), \omega_3(0), \omega_7(0), \omega_9(0)$)) of their respective area.

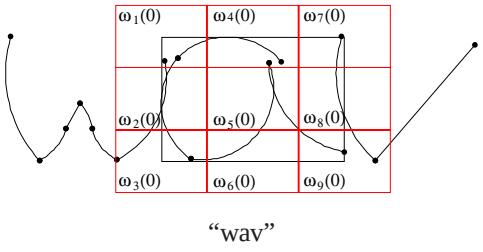

Figure 10: *Initial windows of attention for the reference letter of synthetic word “wav”.*

For word $\gamma_{kj} \in \Gamma_k$, nine segmentation processes are thus started sequentially for each initial window $\omega_1(0), \omega_2(0), \ldots, \omega_9(0)$. The same parameters are used for each process: $t_{max} = 12$ and $\delta_{min} = 0.8$. As for parameters h_k and w_k, the expected height and width of characters in class k, the height of the main body of the word is first estimated using standard histogram analysis. Then the values of the two parameters are computed from the height of the main body using constant factors.

In Figure 11, intermediate results for synthetic word “wav” are illustrated for the nine windows $\omega_1(0), \omega_2(0), \ldots, \omega_9(0)$. Among these, eight have rapidly converged toward the reference letter with a very high degree of detection at the end of the process. This result is very good considering that among the nine initial windows, five ($\omega_2(0), \omega_3(0), \omega_7(0), \omega_8(0), \omega_9(0)$) contain some parts of the surrounding ‘w’ or ‘v’ letters. Even so, the segmentation processes succeeded in correctly locating the reference letter for four of these more ambiguous cases. Moreover, the only initial window for which the process did not converge ($\omega_9(0)$) could hardly be expected to do so since it contains none of the “crucial” information of the reference letter.

Table 2 presents the results of more exhaustive tests conducted over our complete synthetic word data sets. The first and second rows of this table give the average degrees of detection achieved when the window of attention matches

Table 2: *Average degree of detection per character class (in %) ; the first two rows are reference performances while the next four describe the performance of the segmentation process.*

	a	b	c	d	e	f	g	h	i	j	k	l	m
isolated	97.0	99.2	97.2	98.6	97.1	85.6	97.2	95.8	98.9	98.7	96.9	90.6	95.4
word	81.0	83.8	50.0	95.4	53.3	78.8	89.9	66.6	54.7	80.8	81.4	89.3	64.7
1st max	96.5	97.1	84.2	99.9	94.8	96.8	99.1	96.6	99.9	98.6	98.1	90.8	98.9
2nd max	93.6	92.9	73.3	99.7	85.8	93.6	97.0	93.9	95.8	96.2	95.8	81.5	97.7
3th max	88.8	87.2	65.1	99.2	78.1	86.7	94.2	89.3	86.3	93.1	94.3	71.5	95.6
4th max	81.0	81.2	56.2	97.6	65.4	78.1	87.6	83.3	66.3	83.2	88.8	56.5	92.9
	n	o	p	q	r	s	t	u	v	w	x	y	z
isolated	95.7	97.6	98.7	96.4	97.3	97.7	95.4	94.8	91.9	94.7	98.5	93.4	96.9
word	64.7	70.9	90.0	90.9	65.7	68.1	72.8	67.8	55.8	82.8	86.6	71.9	53.8
1st max	93.5	88.1	99.0	98.6	99.6	95.9	98.0	91.6	87.6	96.7	99.5	97.9	88.9
2nd max	88.0	80.8	97.1	95.6	98.1	91.2	95.9	85.0	79.4	91.5	98.7	96.0	79.1
3th max	79.5	69.5	95.4	90.7	96.7	85.3	91.8	77.3	70.1	84.8	96.6	93.8	69.4
4th max	68.7	56.7	93.3	82.3	92.0	78.7	85.6	65.5	58.1	74.7	95.0	89.2	52.8

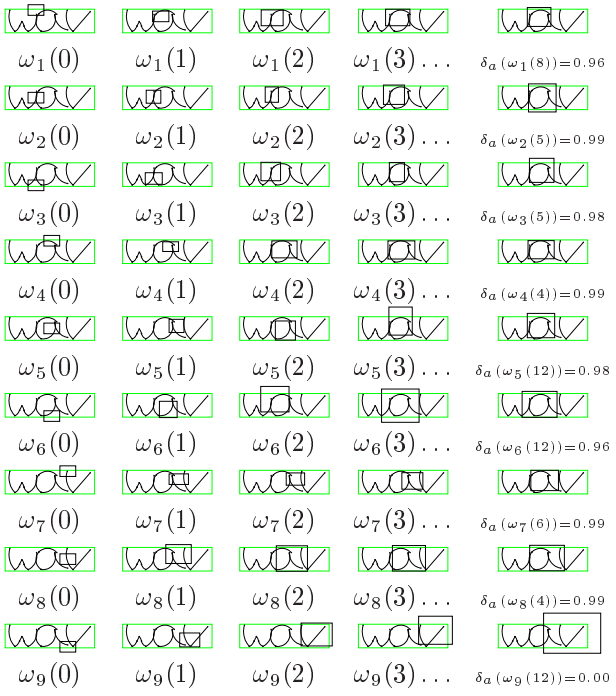

Figure 11: *Illustration of the segmentation process for nine distinct initial windows over synthetic word "wav".*

exactly the known bounding box of the reference letter. The first row is when the reference letter is taken completely out of context (i.e. is isolated), while the second row is when the reference letter is in a word context. These results clearly indicate that the 5% of character overlapping in the synthetic words induces a significant drop in the average degree of detection, especially for simple characters like letter 'c'. The content of these two rows are important because they will serve as reference performances. Indeed they can be interpreted respectively as the maximum and minimum expected performances of the segmentation process.

The next four rows of the table show the results of the proposed segmentation process when considering respectively the averages of the first, the second, the third and the forth maximum detection degrees[2] obtained from the nine initial windows. If these results are compared with those of row two, one can observe that, except for the case of letter 'l' and letter 'o', these maximum degrees of detection are all higher than those obtained with the real bounding boxes of the reference letters in a word context. Thus the segmentation process was able to strongly segment the reference letters even though their representations were significantly degraded by the overlap between characters. Furthermore, if the best results of the segmentation process are now compared to those of the first row, it can be seen that in many cases, they come very close to the degree of detection of the isolated reference letter, and sometimes they can even surpass them slightly. This phenomenon can be explained by the fact that the segmentation process is sometimes able to zoom into a character and reject some of its more bulky ligatures that can be found in the isolated characters of the UNIPEN database.

Overall, the averages of the rows of Table 2 are respectively 96.0, 73.5, 95.6, 91.0, 85.8 and 77.3, which demonstrates that on average, at least 4 times out of 9, our segmentation process can converge on the reference letter and produce a better detection degree than the one that stems from its own bounding box. Thus, we conclude to the feasibility of our proposed segmentation strategy.

[2] A window that has converged with a high degree of detection but with its center point outside of the real bounding box of the reference letter is automatically rejected.

8 Conclusion

This paper has presented a new strategy for tackling the segmentation/recognition dilemma in cursive handwriting. This strategy is based on a moving window of attention, initially positioned somewhere within the cursive word, that searches for surrounding character hypotheses using the learned crucial parts of each character class. The main advantage of the proposed approach is that it does not rely on, nor require, any of the usual ill-defined preprocessing steps like dehooking, slant removal, precise baseline extraction, etc, nor does it assume sequential processing of strokes, thus eliminating the problems caused by delayed strokes. Another of its interesting characteristics is that it can be completely trained using data sets of isolated characters only.

Although it has yet to be applied to the recognition of real cursive script, its feasibility was however shown by constructing synthetic three letter words using overlapping isolated characters, and by showing that when the window of attention is positioned near the middle letter of these synthetic words, the proposed segmentation process is capable of converging rapidly to the bounding box of the middle letter with a high degree of detection at least 4 times out of 9 (on average). Thus by defining a simple scanning policy of the whole word, and by using adjacency constraints [10], one could create a segmentation graph of all found character hypotheses, and then search this graph for coherent character strings.

References

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

[2] T. Wakahara, H. Murase, K. Odaka, "On-Line Handwriting Recognition", *Proc. of the IEEE*, vol. 80, no. 7, pp. 1181-1194, 1992.

[3] L. Schomaker, "Using Stroke- or Character-based Self-Organizing Maps in the Recognition of On-Line, Connected Cursive Script", *Pattern Recognition*, vol. 26, no. 3, pp. 443-450, 1993.

[4] P. Morasso, L. Barberis, S. Pagliano, D. Vergano, "Recognition Experiments of Cursive Dynamic Handwriting with Self-Organizing Networks", *Pattern Recognition*, vol. 26, no. 3, pp. 451-460, 1993.

[5] G. Seni, R. K. Srihari, N. Nasrabadi, "Large Vocabulary Recognition of On-Line Handwritten Cursive Words", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 757-762, 1996.

[6] Y. Bengio, Y. LeCun, C. Nohl, C. Burges, "LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition", *Neural Computation*, vol. 7, no. 5, 1995.

[7] M. Mohamed, P. Gader, "Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 5, pp. 548-554, 1996.

[8] M. Parizeau, R. Plamondon, "A Fuzzy-Syntactic Approach to Allograph Modeling for Cursive Script Recognition", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, No. 7, pp. 702-712, July 1995.

[9] C. Y. Suen, J. Guo, Z. C. Li, "Analysis and Recognition of Alphanumeric Handprints by Parts", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 614-631, 1994.

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

[11] X. Li, M. Parizeau, R. Plamondon, "Segmentation and Reconstruction of On-Line Handwritten Scripts", *Pattern Recognition*, vol. 31, no. 6, pp. 675-684, 1998.

[12] J.-F. Hébert, M. Parizeau, N. Ghazzali, "A New Fuzzy Geometric Representation for On-Line Isolated Character Recognition", *Proc. of the 14th International Conference on Pattern Recognition*, pp. 1121-1123, 1998.

[13] M. Guillot, R. Azouzi, "Improving On-Line Adaptation in Neurocontrol using a Combination of Self-organizing Map and Multilayer Feedforward Network", appears in *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 4, ASME Press, pp. 915–922, 1995.

[14] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, S. Janet, "UNIPEN Project of On-Line Data Exchange and Recognizer Benchmarks", *Proc. of the 12th International Conference on Pattern Recognition*, vol. 2, pp. 29-33, 1994.

[15] S. Haykin, *Neural Networks — A Comprehensive Foundation*, chap. 6, IEEE Press, 1995.

[16] M. Côté, E. Lecolinet, M. Cheriet, C. Y. Suen, "Automatic Reading of Cursive Script Using a Reading Model and Perceptual Concepts: the PERCEPTO System", to appear in the first issue of *International Journal on Document Analysis and Rec.*, 1998.