Fusing High- and Low-Level Features for Handwritten Word Recognition

Alessandro L. Koerich, Alceu S. Britto Jr., Luiz E. S. De Oliveira, Robert Sabourin

Fusing High- and Low-Level Features for Handwritten Word Recognition

Alessandro L. Koerich[1], *Alceu S. Britto Jr.*[1], *Luiz E. S. de Oliveira*[1] *and Robert Sabourin*[2]
[1]Pontifical Catholic University of Paraná (PUCPR), Curitiba, Brazil
[2]École de Technologie Supérieure (ÉTS), Montréal, Canada
{alekoe,alceu,soares}@ppgia.pucpr.br, robert.sabourin@etsmtl.ca

Abstract

This paper presents a novel approach that combines high level and low level features for the recognition of handwritten words. Given a word image, high level features are extracted from loosely segmented words. Such features are used with an HMM word classifier in a lexicon-driven approach. This classifier produces at the output a ranked list of the N-best recognition hypotheses consisting of text transcripts, segmentation boundaries of the word hypotheses into characters, and recognition scores. Given the segmentation boundaries produced by the HMM classifier, low level features are extracted from the character hypotheses. Such features are used with a segmental neural network classifier (SNN) in a hypothesis-driven approach. At combination level, the confidence scores produced by both HMM and SNN classifier are combined through simple combination rules. Experimental results on a large database have shown that the combination of high-level and low-level features reduces the word error rate in almost 71%.

Keywords: Feature extraction, Combination, Word recognition.

1. Introduction

Recognition of handwritten words has been a subject of intensive research in the last years [8, 13]. Significant improvements in the performance of recognition systems have been achieved. Current systems are capable of transcribing handwriting with average recognition rates of 50-99%, depending on the constraints imposed (e.g. size of vocabulary, writer-dependence, writing style, etc.) and also on the experimental conditions. The improvements in performance have been achieved by different means. Some researchers have combined different feature sets or used optimized feature sets [3, 8]. Better modeling of reference patterns and adaptation have also contributed to improve the performance [8]. However, one of the most successful approaches to achieve better performance is the combination of classifiers. Combination of classifiers relies on the assumption that different classification approaches have different strengths and weaknesses which can compensate each other through the combination. However, most of the combination approaches in handwriting recognition uses single feature vectors and different classifiers [2, 10, 12].

This paper is focused on the combination of features extracted from different representation spaces, namely, words and characters with the aim of improving the handwritten word recognition performance. To achieve such a goal, two categories of features are considered: high level features including, global, histogram and segmentation features are extracted from word segments and generate variable-length feature vectors; low features that represent different discriminant characteristics of the handwriting are extracted from character hypotheses and generate fixed-length feature vectors. Both feature vectors are combined at classification level through the combination of the confidence scores produced by the classifiers. The classifier that accounts for the high level feature vectors is based on hidden Markov models (HMM) and it provides a list of the *N*-best word hypotheses, their *a posteriori* probabilities and character hypotheses segmented from such word hypotheses. The low level features are extracted at the character level and used together with a segmental neural network (SNN) character classifier. An overview of the proposed approach is shown in Fig. 1. The details of the preprocessing and segmentation steps are omitted in this paper but can be found in [1].

The novelty of this approach relies on the fusion of high-level, long-range features and low-level local features. Through this combination more information is represented in the features. The proposed approach hold the promise of improving recognition by adding complementary information and possibly robustness to discriminate very similar words. The novelty also relies on the combination of classifiers that operates into two different representation spaces, i.e. word and character.

The rest of this paper is organized as follows: Section 2 describes the high-level long-range features and the low-level local features. Section 3 gives a brief description of the HMM and SNN classifiers that accounts for the high and low-level feature vectors respectively. The fusion of the high-level and low-level features through classifier combination is presented in Section 4. Section 5 reports experiments and results obtained. Conclusions are drawn in the last section.

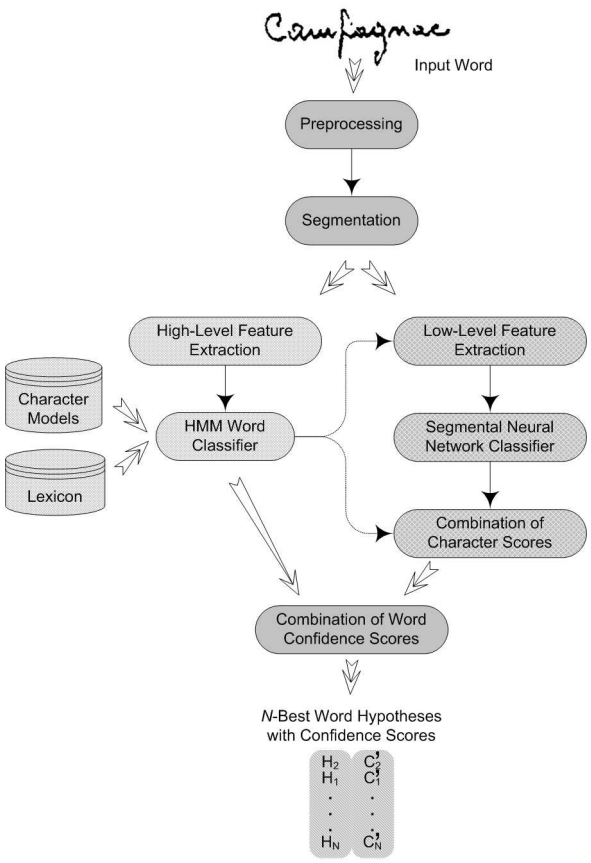

Figure 1. An overview of the fusion of high-level and low-level feature vectors

2 Feature Extraction

High-level features are global features such as ascenders, descenders and loops. They are usually powerful but less robust. These features do not work very well on unconstrained handwriting, because the results of high-level feature extraction tend to be erroneous due to the large shape variations in natural cursive handwriting, especially among different writers [5]. On the other hand, low-level features are local features such as profiles, projections, directional contour, concavities, etc. They are usually less informative but more reliable [5].

It is clear that to further improve word recognition, combination of high-level long-range features and low-level local features is necessary.

2.1 High Level Features

The aim of high-level feature extraction is to extract, in an ordered way, a set of relevant features that reduce redundancy in the word image while preserving the discriminative information for recognition. The main philosophy in this step is that lexicon-driven word recognition approaches do not require features to be very discriminative at the character or pseudo character level because other information, such as context, word length, etc., are available and permit high discrimination of words. Thus, we consider features at the grapheme level with the aim of

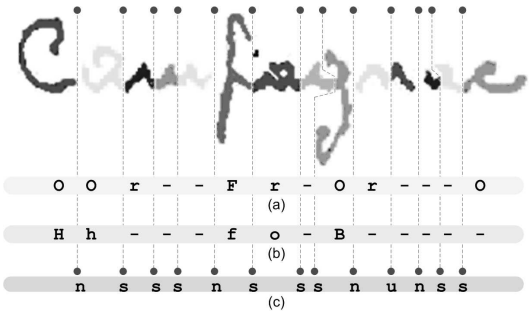

Figure 2. High level features extracted from a handwritten word: (a) sequence of global features; (b) sequence of bidimensional transition histogram features; (c) sequence of segmentation features

clustering letters into classes. A grapheme may consist of a full character, a fragment of a character or more than a character.

The sequence of segments obtained by the segmentation process is transformed into a sequence of symbols by considering two sets of features where the first feature set is based on global features, namely loops, ascenders, and descenders [1]. Ascenders (descenders) are encoded in two ways according to their relative size compared to the height of the upper (lower) writing zone. Loops are encoded in various ways according to their membership in each of the three writing zones and their relative size compared to the sizes of these zones. The horizontal order of the median loop and the ascender (or descender) within a segment are also taken into account to ensure a better discrimination between letters. Each combination of these features within a segment is encoded by a distinct symbol, leading in this way to an alphabet of 27 symbols [1].

The second high-level feature set is based on the analysis of the bidimensional contour transition histogram of each segment in the horizontal and vertical directions. After a filtering phase consisting of averaging each column (row) histogram value over a five pixels-wide window centered in this column (row) and rounding the result, the histogram values may be equal to 2, 4, or 6. In each histogram it is focused only on the median part, representing the stable area of the segment, and it is determined the dominant transition number for which the number of columns (rows) with a histogram value equal to the dominant transition number is maximum. Each different pair of dominant transition numbers is then encoded by a different symbol or class. After having created some further subclasses by a finer analysis of the segments, this coding leads to a set of 14 symbols.

Five high-level segmentation features that try to reflect the way segments are linked together are also extracted. For connected segments, two configurations are distinguished: If the space width is less than a third of the average segment width, it is considered that there is no space. Otherwise, the space is validated and encoded in two ways, depending on whether the space width is

smaller than average segment width. If the two segments are connected, the considered feature is the segmentation point vertical position which is encoded in two ways depending on whether the segmentation point is close to or far from the writing baseline.

Given an input word image, the output of the high-level feature extraction process is a pair of symbolic descriptions of equal length, each consisting of an alternating sequence of segment shape symbols and associated segmentation point symbols. Figure 2 shows an example of high-level feature sequences generated from a handwritten word. Such a high-level feature vector has a variable length which depends on the word length.

2.2 Low Level Features

The main assumption in extracting low-level features is that the segmentation of the words into characters carried out by the HMM classifier is reliable. This assumption is based on previous studies that have shown that the segmentation is reliable in most of the cases. We rely on the segmentation of the word hypotheses into characters and on the labels provided by the HMM classifier to extract low-level features at the character level in an attempt to better discriminate characters and reduce the ambiguity between similar words. Given the output of the HMM classifier, character alternatives are located within the word hypotheses by using the character boundaries.

The aim of low-level feature extraction is to extract from isolated characters a set of relevant features that provides highly discriminative information between character classes. The main philosophy in this step is to extract very discriminative features at the character level [6, 14, 15]. After an empirical evaluation of many different features types combined into different feature sets, three types of features were chosen: profiles and projection histogram from whole characters and contour-directional histogram from six zones of the characters.

The profile counts the number of pixels (distance) between the bounding box of the character image and the edge of the character. The profile of a character can be taken at any position, but usually profiles are taken at four positions: top, bottom, left and right hand sides as illustrated in Figure 3(a). The profiles describe well the external shapes of characters and allow to distinguish between a great number of letters, such as ”p” and ”q”. Since the profiles depend on the image dimension, the features are made scale independent by using a fixed number of bins on each axis that is obtained by merging neighboring pixels and dividing by the total number of black pixels in the character image. We have normalized the profiles to ten bins at each axis to have an equal number of elements for all characters[1].

Projection histograms count the number of pixels in each column and row of a character image [4, 15]. The projection can also be taken at any position, but usually

[1]The number of bins was determined empirically by exploratory experiments on the validation dataset where the character recognition rate and the dimensionality were used as evaluation criteria.

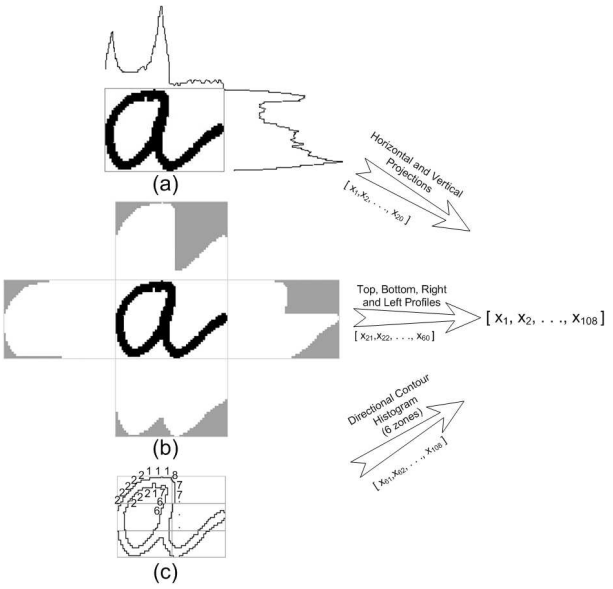

Figure 3. An example of low-level features for the letter ”a”: (a) Top, bottom, left and right hand side profiles; (b) Vertical and horizontal projection histograms; (c) The contour of the letter ”a” split in 6 zones and the directional-contour histogram

projection are taken at the vertical and horizontal axis. For a horizontal projection, $Ph(x_i)$ is the number of pixels with $x = x_i$ while for a vertical histogram, $Ph(y_i)$ is the number of pixels with $y = y_i$. The features are made scale independent by using a fixed number of bins on each axis that is obtained by merging neighboring pixels and dividing by the total number of black pixels in the character image. Projection histograms can separate characters such as ”m” and ”n” (3 and 2 peaks in vertical projection respectively) or ”E” and ”F” (3 and 2 peaks in horizontal projection respectively). The projection histograms are normalized to ten bins at each axis to have an equal number of elements for all characters. Figure 3(b) shows the vertical and horizontal projection histograms for the letter ”a”.

The contour of the character image is given by the outer and inner boundary pixels that can be easily found by examining each pixel within a 3×3 window. Considering a binary character image, if the center pixel is black and at least one of its neighborhood pixels is white, then the center pixel is a contour pixel and it is set as such (set to black). All other pixels are set to white. Figure 3(c) shows the contour for the letter ”a”. Next, the resulting contour is divided into $n \times m$ zones by superimposing an $n \times m$ grid as show in Figure 3(c) [15]. For each of these zones the contour is followed and a directional histogram is obtained by analyzing the adjacent pixels in the 3×3 neighborhood of each contour's pixel. The goal of the zoning is to obtain local characteristics, instead of global characteristics. The zones provide information on the character contours and the direction of the strokes that form them.

These three features types are combined into a single 108-dimensional feature vector and are extracted from each character hypotheses provided by the HMM classifier.

3. Classifiers

3.1. HMM Classifier

The HMM classifier uses discrete observations produced by transitions rather than by states. Transitions with no output are also incorporated into the model. The assumptions to be made are related to the behavior of the segmentation process. As the segmentation process may produce either a correct segmentation of a letter, a letter omission, or an oversegmentation of a letter into two or three segments, a 10-state left-right HMM having three paths to take into account these configurations is adopted. In this model, observations are emitted along transitions. In addition, there is a special model for inter-word space, in the case where the input image contains more than one word. This model simply consists of two states linked by two transitions, modeling a space or no space between a pair of words.

The recognition process consists of determining the word maximizing the *a posteriori* probability that a word w (modeled by a concatenation of character HMMs) has generated an unknown observation sequence O, that is:

$$\hat{w} \ni P(\hat{w}|O) = \max_{w \in \mathcal{R}} P(w|O) \tag{1}$$

where $\mathcal{R}$ is the recognition vocabulary. Using Bayes' Rule and assuming that $P(O)$ does not depend on w, and equal *a priori* probabilities of words $P(w)$, the MAP decoding rule can be approximate by:

$$\hat{w} = \arg\max_{w \in \mathcal{R}} P(O|w) \tag{2}$$

The estimation of $P(O|w)$ requires a probabilistic model that accounts for the shape variations O of a handwritten word w. Such a model consists of a global Markov model created by concatenating character HMMs at state level. The architecture of this model remains the same as in training. However, as the writing style is unknown during the recognition, a character model here actually consists of two models in parallel, associated with the uppercase and lowercase representations of a character.

The HMM classifier uses a variant of the Viterbi algorithm that accounts for the specifics of the HMMs, that is, the concatenation of character HMMs, observations generated along transitions, the presence of null transitions, left-right transitions, and well-defined initial and final states [9]. The decoding procedure is applied repeatedly for each word in the lexicon and at the end we pick up the n words that give the highest probabilities which are denoted as H_n.

3.2. SNN Classifier

The architecture of the SNN resembles a standard MLP character classifier. The choice of such a classifier to perform the character recognition task was determined by several constraints such as: recognition speed, capacity of dealing with unbalanced distribution of samples per class, and mainly because, if properly configured, an MLP classifier estimates Bayesian *a posteriori* probabilities [11]. The last characteristic is very important, because later we want to combine the output of the character recognizer with that of the HMM classifier.

The task of the SNN is to assign an *a posteriori* probability and a character class to each segment representing a character. We define x_l as the feature vector corresponding to the l-th word segment and c_l as the character class of the l-th word segment. The output of the SNN is $P(c_l|x_l)$ which is *a posteriori* probability of the character class c_l given the feature vector x_l.

To build the SNN we have considered a 26-class problem, where uppercase and lowercase representations of characters are merged into a unique class called *metaclass* (e.g. "A" and "a" form the metaclass "Aa"). The main reason for such a choice is related to the irregular distribution of samples per class. The network takes a 108-dimensional feature vector as input and it has 100 units in the hidden layer and 26 outputs, one for each character class. The isolated characters are represented in the feature space by 108-dimensional low-level feature vectors which are formed by combining three different types of features.

Having a word hypothesis and the probability of each character that form such a word given by the SNN classifier, it is possible to combine such probabilities to obtain a probability for the word hypothesis. Assuming that the representations of each character are conditionally statistically independent, character estimations can be combined by a product rule to obtain word probabilities. However, in practice, this is a severe rule of fusing the character probabilities as it is sufficient for a single character to exhibit a low probability (close to zero) to flaw the word probability estimation [7]. Since we have equal prior word probabilities, an alternative to the product rule is the median rule which computes the average probability as:

$$P(H_n|S_n) = \frac{1}{L} \sum_{l=1}^{L} P_{SNN}(c_l|x_l) \tag{3}$$

where $P(H_n|S_n)$ is the probability of the word hypothesis H_n given a sequence of segments S_n, $P_{SNN}(c_l|x_l)$ is the *a posteriori* probability estimated by the SNN to each segment x_l, and L is the number of characters at the word hypothesis H_n.

4. Fusion of High-Level and Low-Level Features

The HMM classifier is regarded as a sophisticated and executes a huge task of assigning a class (a label from $\mathcal{R}$) to the input pattern. On the other hand, the SNN uses quite

different features and classification strategy and it works at a different feature and decision space. Improvements in the word recognition rate are expected by combining such different approaches and feature sets. However, before the fusion of the classifier outputs, both the HMM and SNN classifier output scores are normalized by Equation 4 so that the scores assigned to the *N*-best word hypotheses sum up to one.

$$C_{\cdot}(H_n) = \frac{P(.|.)}{\sum_{n=1}^{N} P(.|.)} \quad (4)$$

where $C_{\cdot}(H_n)$ denotes the normalized confidence score for the word hypothesis, $P(.|.)$ corresponds to either $P(w_n|o_1^T)$ or $P(H_n|x_1^L)$ depending on whether the output of the HMM classifier or SNN classifier is being normalized respectively, and N is the number of best word hypothesis take into account.

The measures produced by the classifiers are confidence scores for each word hypothesis in the *N*-best word hypothesis list. From the combination point of view we have two classifiers each producing N consistent measurements, one for each word hypothesis in the *N*-best word hypothesis list. The simplest means of combining them to obtain a composite confidence score is by *Max*, *Sum*, and *Product* rules. However, in our case, the classifiers are not conditionally statistically independent since the space of the SNN classifier is a subspace of the HMM classifier. It is logical to introduce weights to the output of the classifiers to indicate the performance of each classifier. Changing the weights allows us to adjust the influence of the individual recognition scores on the final score. During the experiments we have found that the weighted sum rule given by Eq. 5 performs better than the others.

$$C'(H_n) = \alpha C_{HMM}(H_n) + \beta C_{SNN}(H_n) \quad (5)$$

where $C_{HMM}(H_n)$ and $C_{SNN}(H_n)$ denote the confidence scores produced by the HMM and the SNN classifier to the n best word hypothesis and α and β are the weights associated with the HMM and the SNN classifier respectively. The composite score resulting from the combination is denoted by $C'(H_n)$.

5. Experiments and Results

During the development phase of our research we have used the SRTP database which is a proprietary database composed of more than 40,000 binary images of real postal envelopes digitized at 200 dpi. From these images, three datasets that contain city names manually located on the envelopes were generated: a training set with 12,023 words, a validation set with 3,475 words and a test set with 4,674 words.

The performance of the HMM classifier together with the high-level features was evaluated on a very-large vocabulary of 85,092 city names. Table 1 shows the absolute word error rate achieved by the HMM classifier on five different dynamically generated lexicons. In Table 1 TOP1, TOP5 and TOP10 denote that the truth word is the best word hypothesis or it is among the five best or the ten best word hypotheses respectively.

Table 1. Average word error rate on 4,674 words for the HMM classifier alone, the SNN classifier alone and the fusion of both by the weighted sum rule

Lexicon Size	Feature Set	Word Error Rate (%) TOP 1	TOP 5	TOP 10
10	High	1.16	0.04	0.0
	Low	5.96	1.24	—
	High+Low	0.71	0.02	—
1,000	High	8.99	3.68	2.29
	Low	11.79	4.16	—
	High+Low	5.37	2.70	—
10,000	High	18.94	9.42	7.64
	Low	17.47	8.88	—
	High+Low	12.47	7.68	—
40,000	High	26.77	15.36	12.09
	Low	24.03	14.02	—
	High+Low	18.98	12.82	—
80,000	High	31.35	18.68	14.9
	Low	27.61	16.91	—
	High+Low	22.38	15.73	—

The SNN classifier was developed on a database of isolated handwritten characters derived from the SRTP database. A training set contains 84,760 isolated characters and a validation set of 36,170 characters were used during the training of the SNN and a test set of 46,700 characters was used to evaluate the performance of the SNN in the recognition of isolated characters.

The performance of the SNN classifier together with the low-level features to recognize handwritten words is shown in Table 1. In this experiment, given the ten best word hypotheses and segmentation boundaries generated by the HMM classifier, the SNN classifier is invoked to produce confidence scores to each character hypothesis. The confidence scores for word hypotheses are obtained through Eq. 3 and the word error rate is computed based only on such scores. There is a relative increase in recognition errors compared to the HMM classifier for lexicons with less than 10,000 words. On the other hand, for lexicons with 10,000, 40,000 and 80,000 words, the error rates achieved by the SNN classifier are lower than those of the HMM classifier.

The results of the combination of high-level and low-level feature vectors through the weighted sum rule show a significant reduction of almost 71% in the error rate relative to the high-level feature vector alone (80,000-word lexicon). More moderate improvements were achieved for smaller lexicon sizes. Notice that the effects of the combination are gradually reduced as the size of the lexicon decreases, but it is still able to reduce the error rate by about 61% for a 10-word lexicon.

In spite of the reduction in error rate brought about by the combination of the high-level and low-level features there is still a significant difference between the er-

ror rates of the top one and top ten word hypotheses which ranges from 0.71% to 7.48% for a 10-word and an 80,000-word lexicon respectively. To understand better the role of the low-level features in the recognition of unconstrained handwritten words we analyze the situations where the SNN classifier succeeds in re-scoring and re-ranking the truth word hypothesis (shifting it up to the top of the *N*-best word hypothesis list). In summary, the combination of the high-level and the low-level features was able to re-rank correctly 10.45% of the word hypotheses, shifting them up to the top of the list, but it also re-ranked wrongly 1.48% of the word hypotheses, shifting them down from the top of the list. This represents an overall reduction of 8.97% in the word error rate for an 80,000-word lexicon. Considering that the upper bound for improvement is the difference in error rate between the top one and the top ten, that is, 17%, and that the truth word hypothesis was not present in 9.56% of the ten-best word hypothesis lists, the improvement brought about by the combination of the high-level and the low-level features is very significant (more than 50%). The error analysis on other sizes of lexicons were also carried out and the proportion of the errors found was similar to those presented above.

6. Conclusion

In this paper we have presented a approach that relies on the combination of high-level long-range features and low-level local features through the fusion of classifiers output. The proposed approach combines features extracted from different representation spaces (word and character). The error rate resulting from the combination of the two feature types at classification level is significantly better than that achieved by single feature types. For instance, on an 80,000-word lexicon the HMM classifier which uses high-level features achieves word error rates of about 32%. Combining it with the SNN classifier which uses low-level features, it is possible to achieve word error rates of about 22%. This implies in a relative error reduction of about 71%.

In spite of the good results achieved, there are some shortcomings related to the proposed approach. The first shortcoming is that extraction of low-level features depends on the word hypotheses and segmentation hypotheses provided by the HMM classifier. If the truth word hypothesis is not present in the *N*-best word hypothesis list provided by the HMM classifier, the use of the low-level features becomes useless. However, this problem can be alleviated by using a great number of word hypotheses instead of only ten.

In summary, the main contribution of this paper is a novel approach that combines high-level features extracted from word segments and low-level features extracted from isolated characters to reduce the word error rate. The proposed combination is effective and computational efficient. The improvements reported in this paper are significant. Hence it is logical to conclude that a fusion of different feature types at classification level is a promising research direction in handwriting recognition.

Acknowledgments

The authors would like to acknowledge the CNPq-Brazil for the financial support (Grants 476275/2004-0 300878/2004-4), and the SRTP-France for providing us the database.

References

[1] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C. Y. Suen. Unconstrained handwritten word recognition using hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.

[2] P. D. Gader, M. A. Mohamed, and J. M. Keller. Fusion of handwritten word classifiers. *Pattern Recognition Letters*, 17:577–584, 1996.

[3] F. Grandidier, R. Sabourin, and C. Y. Suen. Integration of contextual information in handwriting recognition systems. In *7th International Conference on Document Analysis and Recognition*, pages 1252–1256, Edinburg, UK, 2003.

[4] L. Heutte, T. Paquet, J. V. Moreau, Y. Lecourtier, and C. Olivier. A structural/statistical feature based vector for handwritten character recognition. *Pattern Recognition Letters*, 19:629–641, 1998.

[5] J. Hu, A. S. Rosenthal, and M. K. Brown. Combining high-level features with sequential local features for online handwriting recognition. In *Proc. International Conference on Image Analysis and Processing*, pages 647–654, Florence, Italy, 1997.

[6] F. Kimura, N. Kayahara, Y. Miyake, and M. Shridhar. Machine and human recognition of segmented characters from handwritten words. In *Proc. 4th International Conference on Document Analysis and Recognition*, pages 866–869, Ulm, Germany, 1997.

[7] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[8] A. L. Koerich, R. Sabourin, and C. Y. Suen. Large vocabulary off–line handwriting recognition: A survey. *Pattern Analysis and Applications*, 6(2):97–121, 2003.

[9] A. L. Koerich, R. Sabourin, and C. Y. Suen. Fast two–level hmm decoding algorithm for large vocabulary handwriting recognition. In *Proc. 9th International Workshop on Frontiers in Handwriting Recognition*, pages 232–237, Tokyo, Japan, 2004.

[10] R. K. Powalka, N. Sherkat, and R. J. Whitrow. Word shape analysis for a hybrid recognition system. *Pattern Recognition*, 30(3):412–445, 1997.

[11] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, 3:461–483, 1991.

[12] S. Srihari. A survey of sequential combination of word recognizers in handwritten phrase recognition at cedar. In *Proc. 1st International Workshop on Multiple Classifi er Systems*, pages 45–51, Cagliari, Italy, 2000.

[13] T. Steinherz, E. Rivlin, and N. Intrator. Offline cursive script word recognition – a survey. *International Journal on Document Analysis and Recognition*, 2:90–110, 1999.

[14] H. Takahashi and T. D. Griffin. Recognition enhancement by linear tournament verification. In *Proc. International Conference on Document Analysis and Recognition*, pages 585–588, Tsukuba, Japan, 1993.

[15] O. Trier, A. K. Jain, and T. Taxt. Feature extraction methods for character recognition. *Pattern Recognition*, 29(4):641–662, 1996.