Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Pattern Recognition Letters 24 (2003) 1023–1031

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Impacts of verification on a numeral string recognition system

L.S. Oliveira [a,b,*], R. Sabourin [a,b,c], F. Bortolozzi [c], C.Y. Suen [b]

[a] *École de Technologie Supérieure, Département de Génie de la Production Automatisée, Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), 1100 rue Notre-Dame Ouest, Montreal, Quebec, Canada H3C 1K3*

[b] *Centre for Pattern Recognition and Machine Intelligence (CENPARMI), 1455 de Maisonneuve Blvd. West, Suite GM 606, Montreal, Canada H3G 1M8*

[c] *Pontíficia Universidade Católica do Paraná (PUCPR), Rua Imaculada Conceição 1155, Prado Velho, 80215-901 Curitiba Pr, Brazil*

Abstract

In this paper we discuss the use of high-level verification on handwritten numeral strings. First of all, we introduce the concept of levels of verification and present the baseline system used to carry out the experiments. Two different strategies were developed: absolute and one-to-one verifiers. A thorough error analysis is also presented in order to identify under which conditions high-level verification is more appropriate. Experimental results are presented on NIST SD19 database.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: High-level verification; Handwritten digit recognition; Neural networks

1. Introduction

Significant improvements have been made in the recognition rate of handwritten numeral string recognition systems in the last decade. One of the most compelling reasons is that researchers have combined different feature sets and classifiers in order to achieve better recognition rates (Xu et al., 1992). However, when considering real business constraints, there may be some difficulties such as run time inefficiency, system complexity, and co-ordination of different organizations. Another strategy that can increase the recognition rate in a relatively easy way with a small additional cost is through the use of high-level verification. Such a scheme consists of refining the top few candidates in order to enhance the recognition rate economically. The focus of this work is to show the implementation of this strategy in an existing handwritten numeral string recognition system.

For this purpose, we will consider the system presented in (Oliveira et al., 2001). It takes a segmentation-based recognition approach where an explicit segmentation algorithm determines the cut regions and provides a multiple spatial representation. Such a system considers a strategy based on recognition and verification where the recognition function takes into account only a general-purpose recognizer while the verifiers evaluate the result

* Corresponding author.
E-mail address: soares@cenparmi.concordia.ca (L.S. Oliveira).

supplied by the recognizer. The integration of all modules is done through a probabilistic model inspired by information theory.

In this work we start by introducing the concept of levels of verification and present a brief overview of the system. Afterwards, we describe two different strategies of high-level verification as well as the results achieved by them. All experiments reported in this work were carried out on NIST SD19 database. In order to get a better comprehension of the system, we show a strong error analysis. Finally, we outline the perspectives for future works and present our conclusions.

2. Levels of verification

The recognition and verification scheme looks straightforward, with a verification module embedded in the traditional classification system, which has a general-purpose recognizer only. The goal of the general-purpose recognizer is to attribute a given input to one of the *n* existing classes of the system, while the pattern verifier assumes the role of an expert to evaluate precisely the result of the recognizer in order to compensate for its weakness due to particular training, and consequently to make the whole system more reliable. Usually, a pattern verifier is applied after a general-purpose recognizer and it is designed to "plug and play", i.e., it is used without knowing the implementation details of the recognition modules.

Takahashi and Griffin (1993) define three kinds of verification: absolute verification for each class (Is it a "0" ?), one-to-one verification between two categories (Is it a "4" or a "9" ?) and verification in clustered, visually similar, categories (Is it a "0", "6" or "8" ?).

In addition to these definitions, we introduce the concept of levels of verification, where two levels are considered: high-level and low-level. We define as high-level verifiers those that deal with a sub-set of the classes considered by the general-purpose recognizer. The goal of the verifiers at this level is to confirm or deny the hypotheses produced by the general-purpose recognizer by recognizing them (Takahashi and Griffin, 1993; Zhou et al., 2000). We define as low-level verifiers those that deal with meta-classes of the system such as characters and part of them. The purpose of a low-level verifier is not to recognize a character, but rather to determine whether a hypothesis generated by the general-purpose recognizer is valid or not (Cho et al., 2000).

3. System overview

Basically the baseline system used in this work (Oliveira et al., 2001) is composed of four parts: component detection and segmentation, feature extraction, recognition and verification and post-processing. Due to the restricted space we have here, we will discuss just the recognition and verification module.

The feature extraction is composed of three different feature sets. The first one, which feeds the general-purpose recognizer, uses a mixture of concavity and contour based features. The second one, which feeds the over-segmentation verifier, is based on a multi-level concavity analysis. The last feature set, which feeds the under-segmentation verifier, takes into account the same concavity analysis used by the general-purpose recognizer. The baseline system is composed of all modules depicted in Fig. 1 except the gray ones, which represent the feature set and the high-level verifier.

Although many types of neural networks can be used for classification purposes, we opted for a multi-layer perceptron (MLP) which is the most widely studied and used neural network classifier. Therefore, all classifiers presented in this work are MLPs trained with the backpropagation algorithm (Rumelhart et al., 1995). The training and validation sets were composed of 195,000 and 28,000 samples from hsf_{0,1,2,3} series respectively while the test set was composed of 60,089 samples from hsf_7 series. The recognition rates (zero-rejection level) achieved by the general-purpose recognizer were 99.66%, 99.45%, and 99.13% on the training, validation, and test sets respectively.

The recognition system also considers two verifiers to cope with the over-segmentation and under-segmentation problems. The objective of these verifiers is to validate the general-purpose recognizer hypotheses by using the following

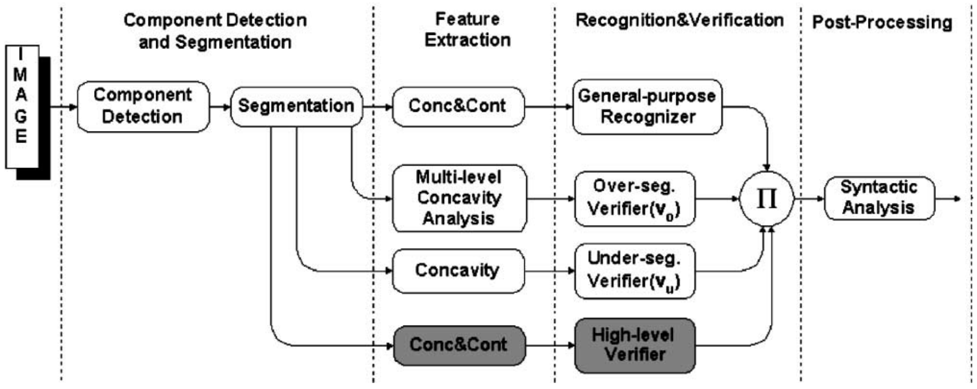

Fig. 1. Block diagram of a numeral string recognition system.

meta-classes: characters, part of characters and under-segmented characters. The over-segmentation verifier was trained with 28,000 samples and it reached a recognition rate of 99.40% on the test set (14,000 samples) while the under-segmentation verifier was trained with 9000 samples and it reached a recognition rate of 99.17% on the test set (4000 samples). We have shown that such verifiers (v_o and v_u in Fig. 1) provided an improvement of about 7% in the recognition rate for numeral strings. However, it is important to emphasize that these verifiers are low-level verifiers and hence we will not discuss how they work in this paper.

4. High-level verification

As mentioned previously, the goal of the high-level verification is to confirm or deny the hypotheses produced by the general-purpose recognizer by recognizing them. In this section we discuss two different strategies of high-level verification: absolute and one-to-one. We will describe how such verifiers were implemented as well as why they contribute to improve the recognition rate of the system in some cases and why they fail in others. We can see this in Fig. 1, where the gray boxes represent the feature set and the absolute verifiers.

All verifiers presented in this work are MLPs trained with the gradient descent applied to a sum-of-squares error function (Bishop, 1995). The transfer function employed is the familiar sigmoid function. In order to monitor the generalization performance during learning and terminate the algorithm when there is no longer an improvement, we have used the method of cross-validation. Such a method takes into account a validation set, which is not used for learning, to measure the generalization performance of the network. During learning, the performance of the network on the training set will continue to improve, but its performance on the validation set will only improve to a point, where the network starts to overfit the training set, that the learning algorithm is terminated. The databases used were the same that we described in the previous section. All networks have one hidden layer where the units of input and output are fully connected with units of the hidden layer.

4.1. Absolute high-level verifier

In this experiment 10 absolute verifiers (one for each numerical class) were considered. Each verifier was trained with two classes: digit and noise. For example, for the verifier of the digit class 0, we have used all zeros of the training set (19,500 samples) for the digit class and the same number of other digits for the noise class. We have tried different feature sets such as concavity analysis in 8-Freeman directions and moments (Hu, 1962). The feature set that produced better results was the same one used by the general-purpose recognizer. Table 1 shows the recognition rates reached by each absolute high-level verifier.

Table 1
Recognition rates achieved by the absolute high-level verifiers

Class	RR (%)
0	99.66
1	99.08
2	99.58
3	99.20
4	99.80
5	99.66
6	99.50
7	99.84
8	99.28
9	99.10

According to the probabilistic model used by the baseline system, the output of the recognition module will be the product of four probabilities: the probability supplied by the general-purpose recognizer, the probability supplied by its respective high-level verifier and the probabilities yielded by the two low-level verifiers (Fig. 1).

We have observed that this strategy of verification produces an improvement to naturally isolated digits, however, when the system faces problems such as touching and fragmentation, it does not seem very appropriate. Table 2 presents the results on different string lengths on NIST SD19 (hsf_7 series). It is worth of remark that 1-digit string (Table 2) means that isolated digits are submitted to the system modules without any a priori knowledge of the image. In addition to recognition errors, segmentation and fragmentation errors are also considered and consequently the system reaches lower recognition rates than when the isolated digit is submitted directly to feature extraction and recognition. As we can see, the overall performance achieved for numeral strings by the system that considers the absolute verifiers is worse than the baseline system. This can be explained by the fact that strings with more than one digit present several cases of fragmentation and touching.

Table 2
Recognition rates (%—zero-rejection level) for numeral strings: (A) system without verifiers, (B) system with low-level verifiers, (C) system B with absolute verification and (D) system B with one-to-one verification

String length	Number of strings	System			
		A	B	C	D
1	60,089	93.71	98.06	98.72	98.02
2	2370	91.56	96.88	96.65	96.10
3	2385	87.98	95.38	95.03	94.98
4	2345	84.91	93.38	92.97	92.91
5	2316	82.00	92.40	92.01	91.03
6	2169	85.66	93.12	92.60	91.77
10	1215	78.97	90.24	89.51	89.00

This strategy seems to be suitable for systems that have a weak general-purpose recognizer or systems that do not face problems such as touching and fragmentation very often. In such cases, the verifier can re-rank the list of hypotheses in order to get the correct answer.

In Table 2 we can also see the importance of the low-level verifiers. For details about them, see (Oliveira et al., 2001).

5. One-to-one high-level verifier

The second strategy of high-level verification that we have implemented was the one-to-one verifier. Such a strategy is straightforward and makes it easy to concentrate on the local difference between two classes. In order to determine the main confusions of the baseline system, we carried out an error analysis on the validation set of isolated digits and we observed 39 different confusions (theoretically, the number of possible confusing digit pairs is $10 \times 9/2 = 45$). We can solve 75.05% and 62.76% of all errors focusing on the top-39 and top-20 confusions respectively. Therefore, it seems more cost effective focusing on top-20 confusions, since we have to deal with about 50% of the confusions produced by the system. Table 3 presents top-20 confusions with their respective frequencies.

We trained each verifier with 40,000 samples (20,000 for each class of digit involved) using the same feature set that we have used for our general-purpose recognizer. For these verifiers we have tried different feature sets such as Edge Maps (Chim et al., 1998) and histograms, but the one that brought better results was the same one used by the general-purpose recognizer. Thereafter, we used the one-to-one verifiers in the same manner we have used the absolute verifiers. Thus, for this experiment the gray module “high-level verifier” in

Table 3
Top-20 digit confusion with frequencies

Confusion	Frequency
8–0	48
3–2	40
2–1	29
4–0	28
7–3	28
9–7	28
9–4	27
5–3	22
6–0	22
8–3	22
7–1	17
9–5	17
7–2	16
9–8	15
8–2	15
6–4	14
6–5	12
8–5	11
9–0	11
8–4	10

Fig. 1 means the one-to-one verifiers. Table 2 (System D) presents the results achieved by this strategy for different string lengths.

As we can notice, this strategy gave worse results than the previous one (System C). We expected better results at least for 1-digit string problem, once the confusions were detected on the isolated digit database. But even for this case the results were unsatisfactory. In order to enhance the results supplied by this strategy, it is necessary to improve the verifier training set by including misrecognized samples. The difficulties of implementing such a solution lie in two points: (i) Lack of samples to improve the database. If we consider our most frequent confusion (8-0), we have just 48 cases, and (ii) if we include a few misrecognized samples in the training set of the verifier, probably we will introduce noise to our models. We can visualize this problem from Fig. 2.

In order to identify the different sources of error of the system and find why the high-level verification schemes achieved unsatisfactory results, we decided to carry out an error analysis on numeral strings instead of isolated digits.

6. Error analysis on numeral strings

In order to gain a better insight of the system, we divided the errors into four classes: confusions generated by the general-purpose recognizer, confusions generated by the low-level verifiers, errors caused by segmentation and errors caused by fragmentation. This analysis was carried out considering the baseline system. Table 4 describes all sources of errors as well as its frequency per string size.

We can read Table 4 in the following way. For 2-digit strings, we have detected 74 errors which

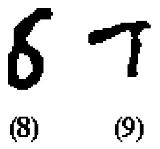

Fig. 2. Misrecognized samples: 8 confused with 6 and 9 confused with 7.

Table 4
Distribuition of the system errors

String length	G-P recognizer				Verifier		Segmentation		Fragmentation		Total	
	Errors	%	Top-10	%	Errors	%	Errors	%	Errors	%	Errors	%
2	43	1.81	31	1.31	15	0.64	9	0.38	7	0.29	74	3.12
3	78	3.26	40	1.67	15	0.64	9	0.38	8	0.33	110	4.61
4	104	4.46	53	2.27	20	0.86	17	0.73	13	0.56	154	6.61
5	126	5.43	72	3.10	11	0.48	14	0.60	25	1.08	176	7.59
6	98	4.49	65	2.98	31	1.42	13	0.59	8	0.37	150	6.87
10	89	6.73	55	4.16	8	0.60	9	0.68	23	1.74	129	9.75

correspond to a global error rate of 3.12%. This error is divided into the following: 43 due to the general-purpose recognizer, 15 to low-level verifiers, 9 to segmentation and 7 to fragmentation. By focusing on the top-10 confusions of the general-purpose recognizer, we can correct 31 of the 43 errors found. In the following subsections, we will give more details about each source of error.

6.1. General-purpose recognizer

In order to generate the total number of confusions for numeral strings, we carried out the analysis for each string length independently and afterwards we summarized the results to compare with the confusions obtained for isolated digits. The first interesting fact we observed was that we have different confusions for different string lengths. This means that a verification strategy based on one-to-one verifier could generate improvements for some string lengths but will be difficult to optimize globally the system with this strategy. Table 5 summarizes the top-10 confusions for each string length.

If we compare the top-10 confusions found on numeral strings (Table 5) with the top-10 confusions found on isolated digits (Table 3), we can observe that the confusions do not respect the same order, and some of the numeral string confusions even do not exist for isolated digits. We can cite for example, 7–2, 4–1 and 7–1 pair confusions. We observed that the effects generated by the segmentation algorithm such as ligatures produce several confusions. Fig. 3 shows the problem of the confusion between 9–7 (top-1 confusion).

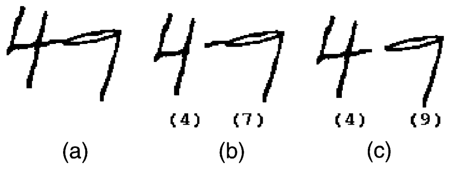

Fig. 3. Confusion between 9 and 7: (a) original image, (b) best hypothesis of segmentation–recognition and (c) correct hypothesis.

6.2. Low-level verifiers

We divided the error of the low-level verifiers into two classes: confusion generated by the over-segmentation verifier and confusion generated by the under-segmentation verifier. We have observed that the latter is responsible for 87.7% of the errors generated at this level, while the former generates just 12.3% of the errors. About the second low-level verifier, the confusions are generated when isolated digits are classified as under-segmented digits. The classes where this kind of confusion occurs are the digit classes 6 (44%), 0 (22%), 8 (17%), 4 (12%) and 2 (5%). Fig. 4 shows some examples of these classes of digits.

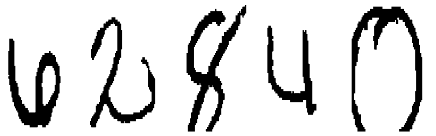

Fig. 4. Digits confused with under-segmented class by the second low-level verifier. This kind of 8 and 0 are sometimes misverified by the first verifier.

Table 5
Top-10 digit confusion with frequencies per string length

2-Digit	3-Digit	4-Digit	5-Digit	6-Digit	10-Digit	Total
7–2: 6	4–1: 5	5–3: 7	7–3: 11	9–7: 11	9–7: 9	9–7: 41
9–4: 5	7–1: 5	9–7: 7	7–9: 9	7–3: 9	6–2: 8	8–0: 39
8–0: 4	4–1: 5	4–1: 6	8–0: 9	8–0: 9	8–0: 7	3–2: 31
3–2: 4	3–2: 5	3–2: 5	7–2: 8	3–2: 8	2–1: 7	9–4: 30
6–5: 2	2–1: 5	2–1: 5	9–4: 7	7–2: 7	7–1: 7	7–2: 27
4–1: 2	6–0: 3	6–0: 5	9–8: 6	9–4: 6	9–4: 5	7–3: 24
8–2: 2	9–4: 3	9–4: 5	3–2: 6	8–3: 6	3–2: 3	5–3: 20
8–5: 2	8–0: 3	8–0: 5	5–3: 6	7–1: 6	4–2: 3	4–1: 20
9–0: 2	7–3: 3	7–3: 4	9–5: 5	5–3: 5	6–5: 3	7–1: 20
9–7: 2	2–0: 3	2–0: 4	6–4: 5	8–2: 5	9–1: 3	2–1: 18

About the first low-level verifier (v_o), the confusion is generated when the verifier fails to detect the over-segmented parts. Such a fact usually happens with digit classes 0 (61%) and 8 (39%). The over-segmentation in these two classes is generated when the digits are opened (Fig. 4). This kind of effect is caused usually by pre-processing.

6.3. Segmentation

The segmentation errors can be caused either by under-segmentation, which is due to a lack of basic points in the neighbourhood of the connection stroke, or wrong segmentation. More details about segmentation errors can be found in (Oliveira et al., 2000).

6.4. Fragmentation

The confusions produced by fragmentation are found basically when the algorithm groups the fragmented part with the wrong neighbor. Usually, it fails for images that have neighbors (left and right) with similar distances to the fragmented part (Fig. 5b) and for images with poor quality (Fig. 5a).

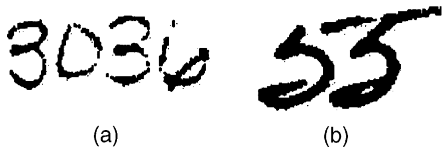

Fig. 5. Fragmentation problems.

7. Discussion

So far, we have described two different strategies of high-level verification in order to improve the recognition rate of the system. We also presented a strong error analysis carried out on numeral strings. As we can notice, both strategies (absolute and one-to-one) do not achieve satisfactory results on numeral strings. Such strategies become interesting either when there is a diversity of samples (confusions) to train the verifiers or when the system has a weak general-purpose recognizer, e.g., the system presented by Britto Jr et al. (2001).

The strategy based on absolute verifiers has brought an improvement for 1-digit string. Such a fact emphasizes that high-level verifiers should be built in order to cope with more complex problems, e.g., all sources of errors presented in the last section. However, we have seen that it is not a trivial problem. As described in Table 5, the confusions for numeral strings are not concentrated in a few classes and for this reason a different strategy of optimization should be adopted in our case. One strategy could be the finding of different feature sets to feed the high-level verifiers. But in this case, the system should overcome the same kind of problems faced by multi-classifier systems, e.g., run time inefficiency and system complexity.

In spite of the fact the current system can be optimized in some respects, we already have recognition rates comparable or better than those reported in the literature. Table 6 summarizes the recognition rates claimed by different authors on NIST SD3/SD19 (hsf_7). Ha et al. (1998) used about 5000 strings of the NIST SD3. Lee and Kim

Table 6
Recognition rates on NIST SD3/SD19—hst_7, zero-rejection level

String length	Ha et al. (1998)		Lee and Kim (1999)		Fujisawa et al. (1992)		Britto Jr et al. (2001)		System B	
	Strings	RR%	Strings	RR%	Strings	RR%	Strings	RR%	Strings	RR%
2	981	96.2	1000	95.23	1000	89.79	2370	95.23	2370	96.88
3	986	92.7	1000	88.01	1000	84.64	2385	92.62	2385	95.38
4	988	93.2	1000	80.69	1000	80.63	2345	92.11	2345	93.38
5	988	91.1	1000	78.61	1000	76.05	2316	90.00	2316	92.40
6	982	90.3	1000	70.49	1000	74.54	2169	90.09	2169	93.12
10							1215	86.94	1215	90.24

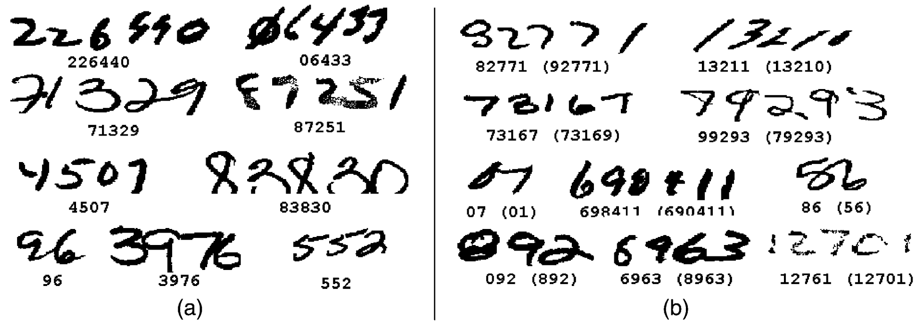

Fig. 6. Examples of (a) correctly recognized fields and (b) misclassified fields.

(1999) used 5000 strings but they did not specify the data used. Britto Jr et al. (2001) used the same database we have used. It is important to remark that the results achieved in Fujisawa's system were published in (Lee and Kim, 1999). As we can see, even considering a larger number of strings we reach better results than the other systems. Fig. 6a shows the examples of fields containing touching or broken characters that were correctly recognized by the baseline system while Fig. 6b shows the examples of misclassified fields.

8. Conclusion

We have presented in this paper some experiments considering strategies of high-level verification. Two different schemes were developed, namely, absolute verifiers and one-to-one verifiers. We have introduced the concept of levels of verification and described the baseline system, which takes into account a segmentation-based recognition approach with an explicit segmentation algorithm. We have seen that the absolute verifier strategy brought an improvement in the recognition rate for 1-digit string but it reached worse results on strings with more than one digit due to the different kinds of errors between isolated digits and string of digits.

Based on the experiments described in this work, we can conclude that one of the best ways to optimize a system with a good overall performance lies in the optimization (deletion, addition, and modification) of the feature sets. Therefore, our next efforts will be focused on the optimization of the feature sets employed in the system. Finally, some results claimed by different authors on NIST SD19 database have been compared.

References

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford University Press, Oxford, UK.

Britto Jr., A., Sabourin, R., Bortolozzi, F., Suen, C.Y., 2001. A two-stage HMM-based system for recognizing handwritten numeral strings. In: Proc. 6th ICDAR, Seattle-USA, pp. 396–400.

Chim, Y.C., Kassim, A.A., Ibrahim, Y., 1998. Dual classifier system for handprinted alphanumeric character recognition. Pattern Anal. Applicat. 1, 155–162.

Cho, S.J., Kim, J., Kim, J.H., 2000. Verification of graphemes using neural networks in an HMM-based on-line Korean handwritting recognition system. In: Proc. 7th IWFHR. Amsterdam, Netherlands, pp. 219–228.

Fujisawa, H., Nakano, Y., Kurino, K., 1992. Segmentation methods for character recognition: from segmentation to document structure analysis. Proc. IEEE 80, 1079–1092.

Ha, T.M., Zimmermann, M., Bunke, H., 1998. Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. Pattern Recognition 31 (3), 257–272.

Hu, M.K., 1962. Visual pattern recognition by moment invariant. IEEE Trans. Inform. Theory 8, 179–187.

Lee, S.W., Kim, S.Y., 1999. Integrated segmentation and recognition of handwritten numerals with cascade neural networks. IEEE Trans. Systems Man, Cybernet. Part C: Applicat. Rev. 29 (2), 285–290.

Oliveira, L.S., Lethelier, E., Bortolozzi, F., Sabourin, R., 2000. A new segmentation approach for handwritten digits. In: Proc. 15th ICPR, Barcelona-Spain, vol. 2, pp. 323–326.

Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y., 2001. A modular system to recognize numerical amounts on Brazilian bank cheques. In: Proc. 6th ICDAR, Seattle, USA, pp. 389–394.

Rumelhart, D.E., Durbin, R., Golden, R., Chauvin, Y., 1995. Backpropagation: the basic theory. In: Chauvin, Y., Rumelhart, D.E. (Eds.), Back Propagation: Theory, Architectures and Applications. Lawrence Erlbaum, Hillsdale, NJ, pp. 1–34.

Takahashi, H., Griffin, T., 1993. Recognition enhancement by linear tournament verification. In: Proc. 2nd ICDAR, Tsukuba, Japan, pp. 585–588.

Xu, L., Krzyzak, A., Suen, C.Y., 1992. Methods of combining multiple classifiers and their applications to handwritten recognition. IEEE Trans. Systems Man Cybernet. 22 (3), 418–435.

Zhou, J., Gan, Q., Krzyzak, A., Suen, C.Y., 2000. Recognition and verification of touching handwritten numerals. In: Proc. 7th IWFHR. Amsterdam, Netherlands, pp. 179–188.