Speeding up On-line Recognition of Handwritten Characters by Pruning the Prototype Set

Vuokko Vuori, Jorma Laaksonen, and Erkki Oja
Helsinki University of Technology
Laboratory of Computer and Information Science
P.O Box 5400, FIN-02015 HUT, Finland
{vuokko.vuori,jorma.laaksonen,erkki.oja}@hut.fi

Jari Kangas
Nokia Research Center
P.O. Box 100
FIN-33721 Tampere, Finland
jari.a.kangas@nokia.com

Abstract

This work describes a prototype-based online handwritten character recognition system and a two-phase recognition scheme aimed to speed up the recognition. In the first phase, the prototype set is pruned and ordered on the basis of preclassification performed with heavily down-sampled characters and prototypes. In the second phase, the final classification is performed without down-sampling by using the reduced set of prototypes. Two down-sampling methods, a linear and nonlinear one, have been analyzed to see their properties regarding the recognition time and accuracy.

1. Introduction

On-line handwriting recognition has become an important alternative for textual input since palm-sized computers and advanced mobile phones, in which a keyboard cannot conveniently be integrated, have become more common. Among the key factors determining the usability and acceptability of a handwriting recognition system are recognition accuracy and speed.

The recognition should be faster than writing so that the user does not need to wait for the results and go back several characters in order to correct recognition mistakes. Humans are able to write from 12 (block printing) to 33 (fast cursive writing) words per minute. Assuming that the average number of characters in a word is five, recognition time for a single character should be approximately 300 ms or less [7]. Also, the recognition accuracy should be high. An acceptable recognition error rate depends on the nature of the task. For example, for writing down personal notes and information meant for another person it is 97% and 99%, respectively [1, 3, 6].

The recognition accuracy can be improved during the normal use of a device by adaptation. For example, a prototype-based system can very easily and quickly be adapted to new writing styles just by adding new character samples to the prototype set or modifying and inactivating existing prototypes [10, 11]. In on-line recognition systems, user adaptation must be concerned too. It is essential that the users comprehend how the recognition system is working in principle. Only then the users can have some insight into the recognition errors and are able to adapt their writing styles so that the recognition performance of the system improves. In addition, well-understood recognition errors might be less irritating than those which seem more or less random. In this respect, a prototype-based recognizer with an intuitive matching method is convenient for on-line applications [12].

A drawback of the prototype-based classifier is that the recognition time depends on the size of the prototype set. The initial prototype set should cover at least the most common writing styles. Adaptation has to be planned so that in the end there will not be many similar prototypes of the same class and all the prototypes will be useful. Recognition time increases when new prototypes are added into the prototype set due to adaptation. On the other hand, it can be decreased by inactivating prototypes which have never been used. Recognition time can be decreased also by clever ordering and pruning of the prototypes prior the final matching. In this work, this approach has been taken. In the experiments, we study how much computation time can be saved by ordering and pruning the prototype set on the basis of preclassification performed with down-sampled characters and prototypes.

The rest of this paper is organized as follows. Section 2 describes the prototype-based recognition system. The decision rule, dissimilarity measure, forming of prototype set and speedup methods are explained too. Section 3 introduces the character database used in the experiments and the normalization and preprocessing methods of the character samples. The experiments are explained and their results are given in Section 4. The concluding remarks are drawn in Section 5.

2. Prototype-based classifier

In our recognition system, classification is carried out by evaluating the dissimilarity measures between the normalized and preprocessed input character and all the prototypes and then applying the nearest neighbor rule [2]. This simple rule has nice asymptotic properties, recognition error rate approaches twice the Bayesian error rate as the number of prototypes becomes large and it has yielded good results in various pattern recognition applications. Its major drawback is that it is computationally heavy, especially with large prototype sets and complicated dissimilarity measures.

2.1. DTW-based dissimilarity measure

Characters are compared with each other by using the same dissimilarity measure based on the Dynamic Time Warping (DTW) algorithm [8] during both prototype selection and classification. Characters are represented as sequences of data points, see Figure 1. The DTW-algorithm matches two characters so that the sum of the squared Euclidean distances between the matched data points is minimized. This can be seen as an elastic matching of the curves: the dissimilarity measure tells how much bending, stretching, and shrinking is required in order to make the two curves similar.

The matching is constrained by boundary and continuity conditions. Boundary conditions ensure that the first and last data points of the two curves are matched against each other. The continuity condition requires that all the data points are matched at least once and in the same order in which they have been produced. Connected parts of the drawn curve where the pressure between the pen and writing surface exceeds a given value are considered as strokes. The characters are compared stroke wise – the dissimilarity between two characters is the sum of the dissimilarities between the stroke pairs. Dissimilarity between characters of different numbers of strokes is set to infinity. The dissimilarity measure is described in more detail in [9].

2.2. Forming of the prototype set

A simple clustering algorithm was used for forming a tree-like clustering for each character class and stroke-number variation. In the beginning of the algorithm, there were as many clusters as there were character samples. Next, the two clusters whose middle items were the most similar were merged and the middle item of the new cluster was searched for. Then again, two clusters were merged into one in a similar manner. The algorithm continued until there was only one cluster left. The benefit of this algorithm is that it does not require any prior information on the number of writing styles. It also has a tendency to keep malformed, rare or even erroneous samples in their own clusters.

All the cluster trees were examined and the prototypes for the next phase were manually selected. Some of the very rare styles (i.e. only one or two samples from one writer) were omitted. All the selected prototypes differed from each other by their shape or by the drawing order or direction of the strokes. Next, the selected prototypes were fine-tuned with a modified LVQ-algorithm using the rest of the characters as training samples [4, 5]. The LVQ-algorithm is able to reshape the prototypes gradually so that they are more general representatives of a group of similar learning samples. Finally, those prototypes which were used by only a single writer in the training set were omitted from the final prototype set.

2.3. Speedup methods

In our system, recognition time can be reduced in two ways: 1) by ordering the prototypes according to the rough shape of their first strokes [9], and 2) by pruning and ordering the prototypes on the basis of preclassification performed with considerably down-sampled characters. The ordering of the prototypes saves computation time as the input character does not need to be completely matched against all the prototypes. If the distance measure between the unknown characters and a prototype exceeds the distance to the kth nearest prototype among the preceding ones, the matching procedure can be interrupted, even though all the points or strokes have not been matched. Parameter k can be the number of prototypes the classification decision is based on or the number of the candidate prototypes selected for the final classification phase. Ordering of prototypes has no effect on the recognition accuracy. Down-sampling is an efficient way to reduce recognition time as the complexity of DTW-matching depends quadratically on the number of data points in a stroke. However, down-sampling increases error rates as characters are represented less accurately.

3. Data

The data used in the experiments consists of isolated handwritten characters ('a'-'z', 'å', 'ä', 'ö', 'A'-'Z', 'Å', 'Ä', 'Ö', '0'-'9') collected from 45 subjects. The writing equipment consisted of a special pressure sensitive tablet attached to a Unix workstation. The resolution of the tablet is 100 lines per millimeter and the sampling rate is at maximum 205 data points per second. A character is presented as a series of the x- and y-coordinates of the moving pen point.

The total number of characters was approximately 40 000. Half of the subjects wrote the characters after a dictation of a short story and without any visual feedback. The other half wrote characters in random order. This time, the characters were shown on the computer screen and were recognized on-line. All the characters were written without any constraints on the style. There seems to be no significant difference between the quality of the characters collected with the two setups.

The locations of the characters were normalized by moving their mass centers to the origin of the coordinate system. The size variations were normalized by scaling the characters so that the lengths of the longer sides of their bounding boxes are set to 100 units and their aspect ratios remain unchanged. After these simple normalizations, the number of data points per each stroke can be reduced by two alternative preprocessing methods. *Decimate*(n) performs a linear down-sampling as it keeps every $(n+1)$th data point and abandons the intermediate ones. The same number of data points (or one less) are abandoned from the beginning of the stroke as from the end. If the number of original data points is less than $(n+1)$, only the first and last data points are kept. *ExtremePoints*(d) is a nonlinear down-sampling method. It keeps the first and last data point of the stroke and the original extreme points in which the x- or y-component of the pen point velocity changes its sign, ceases to be zero, or becomes zero. However, if the Euclidean distance between two successive inflection points is less than d units the latter point is abandoned. The distance between the inflection points is measured along the trace of the pen.

The effects of the two preprocessing methods are illustrated in Figure 1. An unpreprocessed character is shown in subfigure 1(a). From that subfigure, it can be seen that the sampling rate is high and the pen point trajectory can faithfully be represented with the data points as no peculiar corners appear when adjacent data points are connected with lines. The data points are unevenly distributed along the trace of the pen. The data point density is high in those parts where the pen has moved slowly or momentarily stopped. When the character is preprocessed with *Decimate*(n) method, as shown in subfigures 1(b), 1(d) and 1(f), the data point density, and thus some implicit dynamic information on the pen point movements is preserved. However, all the data points are treated equally and some of the original corners and cusps might be cut away and some new ones introduced in rather arbitrary locations. *ExtremePoints*(d) method, see subfigures 1(c) and 1(e), does not preserve the data point density but is able to keep the corners in their original places and introduces new ones only in some geometrically meaningful locations. Therefore, it is not sensitive to variations of writing dynamics between

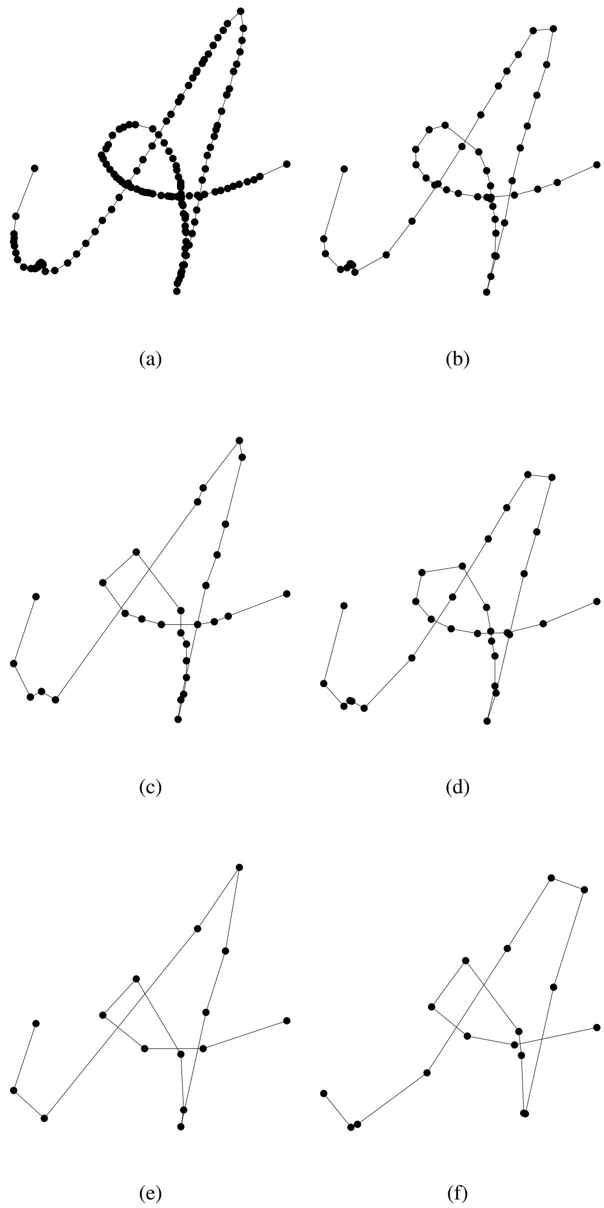

Figure 1. (a) An example character and its preprocessed versions when (b) Decimate(2), (c) ExtremePoints(5), (d) Decimate(4), (e) ExtremePoints(20), or (f) Decimate(8) has been applied.

different subjects. In addition, character samples collected using various sampling methods and devices can be used together when preprocessed with *ExtremePoints*(d) method. This is a desirable property when using databases which have several contributors.

4. Experiments

In all the experiments, the same character database was used both for creating the prototype set and for testing. However, due to the fine-tuning of the prototypes, none of the prototypes was exactly similar with any of the test samples. The number of prototypes selected for fine-tuning was 327 and the final number of prototypes was 254. In the first set of experiments, *Decimate*(n) and *ExtremePoints*(d) methods were compared with each other in respect to their effects on the recognition time and accuracy. All the character samples in the database were classified according to the nearest neighbor rule by using the whole prototype set. Prototypes were ordered according to the shape of their first stroke. The results of these experiments are given in Table 1. From there, it can be seen that both preprocessing methods are efficient in reducing the average recognition time t_{ave}. In the case of *Decimate*(n), the recognition accuracy clearly deteriorates and its variation between the writers increases if the value of the decimation parameter n is increased. When *ExtremePoints*(d) is used as a preprocessing method, recognition accuracy is more or less the same with the different values of the distance parameter d. If the parameter values are set so that the recognition time is approximately the same with the two preprocessing methods, namely $(n = 8, d = 15)$ or $(n = 9, d = 20)$, *ExtremePoints*(d) seems to be better than *Decimate*(n) in respect to the total error rate $E_t ot$, the average error rate calculated for the error rates of different writers E_{ave} and their standard deviation E_{std}.

Next, pruning and ordering of the prototypes on the basis of preclassification was experimented with. In the preclassification phase, both the prototypes and the character samples were heavily down-sampled in order to find the N best matching prototypes fast. In addition, prototypes were ordered according to the shape of their first stroke. The final classification was carried out with pruned prototype set without preprocessing. Again, the classification decision was based on the nearest neighbor rule. The values of the parameters of the preprocessing methods were $n = 8$ and $d = 20$. The results of the experiments are shown in Table 2. In respect to the recognition time and accuracy, the differences between the result obtained with *ExtremePoints*(20) and *Decimate*(8) preprocessing methods are insignificant. Clearly, the most important factor is the number of candidates selected in the preclassification phase. The average recognition time for a single character can be reduced by

Table 1. Effects of Decimate(n) (Dec) and ExtremePoints(d) (ExtrP) preprocessing on the average recognition time of a single character t_{ave} (ms) and recognition error rates. E_{tot} is the total error rate calculated for all character samples. E_{ave} and E_{std} are the average and standard deviation calculated for the error rates of different writers.

Preproc.	t_{ave}	E_{tot}	E_{ave}	E_{std}	$E_{\text{std}}/E_{\text{ave}}$
None	2472	17.4	19.0	5.5	0.29
Dec(1)	685	18.9	20.8	6.0	0.29
Dec(2)	338	18.6	19.9	4.9	0.25
Dec(5)	120	21.3	22.8	5.1	0.22
Dec(6)	98	21.9	23.2	5.2	0.22
Dec(7)	84	24.0	26.1	6.2	0.24
Dec(8)	74	24.9	26.4	6.4	0.24
Dec(9)	66	27.6	29.8	6.7	0.23
ExtrP(5)	168	24.5	25.9	5.6	0.22
ExtrP(10)	102	24.8	26.1	5.3	0.20
ExtrP(15)	78	24.1	25.0	4.4	0.18
ExtrP(20)	65	24.2	25.0	4.8	0.19
ExtrP(25)	57	25.3	26.4	4.8	0.18

76% without compromising the recognition accuracy by using either one of the preprocessing operations for ordering and selecting $N = 10$ candidate prototypes for the final classification. As can seen from Figure 2 there is not much use in founding more candidate prototypes than that. The probability that the best-matching prototype will be among the ten candidates is over 99% with both preprocessing operations.

5. Conclusions

In this work, we have showed that the recognition time of a prototype-based system can be reduced considerably by performing the classification in two phases. First, the prototypes and the unknown sample are heavily down-sampled so that the matching can be performed fast. The down-sampled prototypes are ordered and the N best ones are selected for the second classification phase. The final classification is carried out with the pruned and ordered prototype set but without performing any down-sampling. In that way, the recognition time could be decreased by 76% while the error rate remained the same. We compared two alternative down-sampling methods, a linear and a nonlinear one. According to the experiments, there are no significant differences between the two methods in two-phase classifica-

Table 2. The average recognition time of a single character t_{ave} (ms) and the error rates when prototypes are pruned down to N candidates by using Decimate(8) (Dec) or ExtremePoints(20) (ExtrP) preprocessing. E_{tot} is the total error rate calculated for all character samples. E_{ave} and E_{std} are the average and standard deviation calculated for the error rates of different writers.

Prepr.	N	t_{ave}	E_{tot}	E_{ave}	E_{std}	$E_{\mathrm{std}}/E_{\mathrm{ave}}$
None	-	2472	17.4	19.0	5.5	0.29
Dec	2	253	19.6	21.4	5.8	0.27
ExtrP	2	245	19.8	21.4	5.4	0.25
Dec	3	315	18.6	20.4	5.8	0.28
ExtrP	3	304	18.3	20.0	5.7	0.28
Dec	4	363	18.1	19.8	5.8	0.29
ExtrP	4	355	18.0	19.7	5.6	0.28
Dec	5	412	17.7	19.4	5.6	0.29
ExtrP	5	401	17.8	19.5	5.6	0.29
Dec	10	591	17.4	19.1	5.6	0.29
ExtrP	10	594	17.4	19.0	5.5	0.29
Dec	15	740	17.4	19.1	5.6	0.29
ExtrP	15	784	17.5	19.0	5.6	0.29

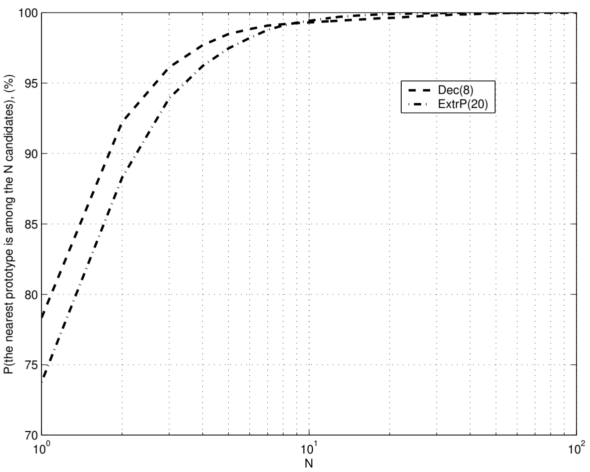

Figure 2. Probability that the best matching nonpreprocessed prototype can be found among the N candidates selected on the basis of the preprocessed prototypes. Dashed and dash-doted lines correspond to cases in which preprocessing is performed with Decimate(8) or ExtremePoints(20), respectively.

tion. However, in one-phase classification, the error rates obtained with the nonlinear method were less dependent on the value of down-sampling parameter and there was less variation between different writers. In addition, the nonlinear down-sampling method can make character samples collected with different devices comparable with each other.

References

[1] Survey of the State of the Art in Human Language Technology. http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html, November 1995.

[2] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, Jan. 1967.

[3] C. Frankish, R. Hull, and P. Morgan. Recognition accuracy and user acceptance of pen interfaces. In *Proceedings ACM CHI'95 Conference on Human Factors in Computing Systems*, 1995.

[4] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, 1997. Second Extended Edition.

[5] J. Laaksonen, J. Hurri, E. Oja, and J. Kangas. Comparison of adaptive strategies for on-line character recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pages 245–250, 1998.

[6] M. J. LaLomia. User acceptance of handwritten recognition accuracy. In *Proceeding of ACM CHI'94 Human Factors in Computing Systems Conference*, page 107, April 1994.

[7] I. S. MacKenzie, B. Nonnecke, S. Riddersma, C. McQueen, and M. Meltz. Alphanumeric entry on pen-based computers. *International Journal of Human-Computer Studies*, 41:775–792, 1994.

[8] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley, 1983.

[9] V. Vuori. Adaptation in on-line recognition of handwriting. Master's thesis, Helsinki University of Technology, January 1999. http://www.cis.hut.fi/ vuokkov/hcr/dtyo.ps.

[10] V. Vuori, M. Aksela, J. Laaksonen, E. Oja, and J. Kangas. Adaptive character recognizer for a hand-held device: implementation and evaluation setup. In L. R. B. Schomaker and L. G. Vuurpijl, editors, *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 13–22, Amsterdam, September 2000. Nijmegen: International Unipen Foundation. ISBN 90-76942-01-3.

[11] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. On-line adaptation in recognition of handwritten alphanumeric characters. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 792–795, 1999.

[12] R. G. Webster and M. Nakagawa. An interface-oriented approach to character recognition based on a dynamical model. *Pattern Recognition*, 31(2):193–203, 1998.