ADAPTIVE CHARACTER RECOGNIZER FOR A HAND-HELD DEVICE: IMPLEMENTATION AND EVALUATION SETUP

VUOKKO VUORI, MATTI AKSELA, JORMA LAAKSONEN, ERKKI OJA
Helsinki University of Technology
Laboratory of Computer and Information Science
P.O.Box 5400
FIN-02015 HUT, Finland
{vuokko.vuori,matti.aksela,jorma.laaksonen}@hut.fi

JARI KANGAS
Nokia Research Center
P.O.Box 100
FIN-33701 Tampere, Finland
jari.a.kangas@nokia.com

In this work, we describe a character recognition system we have implemented for experimenting with self-supervised adaptation method. The Dynamic Time Warping algorithm is used for matching input characters to prototypes and recognition is carried out according to the k-nearest neighbor rule. The prototype set is adapted by adding new prototypes into the prototype set and reshaping existing ones with a method based on the Learning Vector Quantization. The adaptation process is supervised by the user's reactions to the recognition results and other indirect information obtained from the user interface of text input. We also discuss the practical problems encountered in the implementation of a computationally heavy recognition method into a device with limited resources.

1 Introduction

The user interface of a hand-held device cannot be directly adopted from normal-sized computers. Naturally, all the output has to be presentable with a small display of lower resolution. However, the traditional input methods cannot be just scaled down. A miniature keyboard is not convenient to use if the keys are smaller than finger tips and they are placed too near to each other. One solution to the problem is to use a pen for both text input and pointing purposes. Keys can be selected more precisely with a sharp pen than with a finger. A more sophisticated solution is to equip the device with a handwriting recognizer and write the desired text with the pen. Our work concentrates on such input methods for isolated Latin characters.

The main problem in recognition of handwriting is the vast number of personal writing styles. Even if a recognizer is trained with data from several writers and therefore accepts various writing styles, it will certainly perform poorly with writers whose styles are not covered in the learning set. In order to obtain satisfactory recognition accuracy with all users without constraining the allowed style of writing, the recognizer has to be adaptive, i.e., it has to be able to learn new writing styles.

Adaptation of a recognizer can be performed before or during the normal use of the device. The latter way of adaptation is the more appealing one for the user. With such a system, there is no need to carry out some enrollment program first but the device can be used for its designed purpose straight away. However, the supervision of the learning process has to be planned very carefully. In this work, we describe a self-supervised adaptation method suitable for a prototype-based character recognition system. We also discuss the practical problems encountered in the implementation of a computationally heavy recognition method into a device with limited resources.

2 Recognition system

The recognition system used in our work is based on Dynamic Time Warping (DTW) matching and therefore it can be easily adapted by adding new prototypes and reshaping or inactivating existing ones. Characters are input by writing them one by one on a pressure sensitive surface, for example the display of a hand-held device. Recognition is carried out by comparing an input character with all the prototypes and classifying it according to the majority of the k best matching prototypes by the k-NN rule [1]. If an input character cannot be matched with any of the prototypes due to the properties of the applied DTW-algorithm or prototype pruning, it will be rejected.

2.1 Preprocessing and normalization methods

Characters collected with a hand-held device contain some spurious points which have to be filtered out. Filtering is performed by abandoning the ith data point p_i if the following condition is satisfied:

$$fD(p_{i-1}, p_{i+1}) \leq D(p_i, p_{i-1}) + D(p_i, p_{i+1}), \tag{1}$$

where D is the squared Euclidean distance between two data points and f is a constant. On the basis of both visual inspection and classification experiments the value of the filtering parameter f was set to 1.5.

All the characters are preprocessed with a decimation operation which keeps every $(n+1)$th data point and abandons the intermediate ones. For characters collected with a tablet, $n = 2$. Prior to matching, input character and prototypes are moved into the same location. This is carried out by moving their mass centers to the origin of the coordinate system. Size variations are normalized by scaling characters so that the length of the longer side of the bounding box drawn around a character is constant and its aspect ratio remains unchanged. These operations are justified by the results of some previous experiments.[2]

2.2 DTW-matching

The DTW-algorithm[3] matches two curves represented by sequences of data points so that the sum of the squared Euclidean distances between the matched data points is minimized. 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 first continuity condition requires that all the data points are matched at least once and in the same order that they have been produced. The second continuity condition regulates the relative amount that the matching is allowed to differ from linear matching. The strictness of this condition can be presented with a single parameter c. The continuity condition is formulated as follows: Let N_1 and N_2 be the total number of data points in curves 1 and 2, respectively. The ith data point of curve 1 and the jth data point of curve 2 can be matched if

$$\frac{N_2}{N_1} i - cN_2 \leq j \leq \frac{N_2}{N_1} i + cN_2. \tag{2}$$

This condition is symmetric which means that the roles of the curves can be interchanged. The number of feasible matchings and therefore the recognition time can be reduced by decreasing the value of parameter c. If $c = 1$, the continuity condition has no effect. If $c = 0$, linear matching is the only feasible solution.

2.3 Pruning and ordering of the prototypes

The connected parts of a drawn curve in which the pressure between the pen and writing surface exceeds a given value are considered as strokes. The DTW-matching is performed on stroke basis which means that the prototypes are always pruned according to their number of strokes. Prototypes can be further pruned by requiring that the strokes to be matched have to be of somewhat similar length. We have experimented with the following pruning rule: If N_1 and N_2 are the lengths of strokes 1 and 2, respectively, and

$$(N_2 \geq \alpha N_1 + \beta) \quad \text{or} \quad (N_1 \geq \alpha N_2 + \beta), \tag{3}$$

strokes cannot be matched.

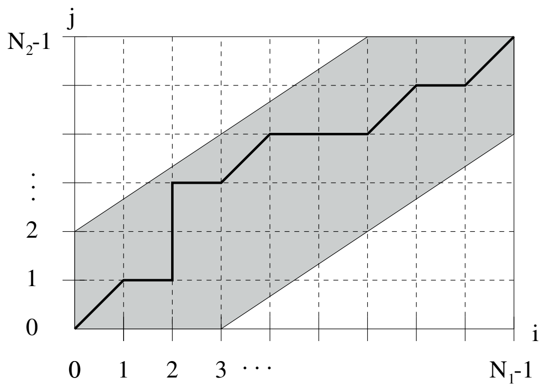

Figure 1: The second continuity condition of DTW-algorithm restricts the amount that solutions can differ from linear matching. In the example, the strictness parameter c is $1/3$ and the allowed matchings are inside the shaded area. Also, an example of feasible solution is plotted with a bold line.

Prior to matching, prototypes are ordered on the basis of the rough shape of their first stroke. When matching the input stroke and a prototype, the procedure is interrupted if the dissimilarity measure between the input character and the current prototype exceeds the kth smallest dissimilarity measures evaluated so far. As a result, significant savings in the computation time can be achieved.

2.4 Prototype selection

First, in order to form an initial prototype set for a hand-held device, character samples written by several subjects were clustered.[4] The character data available to us in this phase of the work was collected with a tablet the resolution and sampling frequency of which are much higher than those of a pressure sensitive display. Even though the cluster centers could not be used directly as a prototype set, the clustering provided valuable insight into the different writing styles. The prototypes were written by a single writer on the basis of the center-most items of the clusters. In this way, the initial prototype set is at least in some sense writer independent as it covers various writing styles. Unfortunately, it is biased to the writing style of that single writer who wrote the prototypes.

2.5 *Adaptation*

Adaptation of the prototype set is performed after a whole text sequence, for example a line, has been submitted. The k nearest prototypes of each input character are examined and if any one of them belongs to the correct class, the nearest prototype is reshaped with an algorithm based on the Learning Vector Quantization (LVQ).[5,6] Otherwise, the input character is added into the prototype set as such. The input characters are handled in their order of writing. Suitable parameter values, $k = 3$ and the learning rate of the LVQ algorithm $\eta = 0.3$, were found in some earlier experiments.[7] The methods for labeling the input characters on the basis of the recognition results and the user's actions will be explained in section 3.

3 User interface

For evaluation purposes, the character recognition system was implemented in a hand-held device, namely a Philips Nino or Everex Freestyle using Windows CE. It is used as a text input method in a hypothetical questionnaire program. The user interface of the program is illustrated in Figure 2. The program asks the user questions which cannot be answered with a single word. The user inputs one character at a time in either of the two writing areas and the recognition results are shown in the text area above. Successive characters input in the same writing area are separated by a time threshold. The system is able to recognize lower and upper case letters and digits. In addition, a single horizontal line drawn from left to right or vice versa is recognized as a space or backspace, respectively. These two symbols can also be inserted with special buttons. If the system is not able to classify the input character (no prototype can be matched due to prototype pruning on continuity constraints), the user is asked to choose the correct class from a table of all the available character classes.

The text cursor can be relocated by pointing the text with the pen. Text partitions can be selected by drawing a horizontal line over the text. These simple functions enable the edition of the text. For example, the user can correct the recognition results by selecting a character and rewriting it. So that the system would not make the same mistake twice in a row, the rewritten characters will not be recognized into the same class as its predecessor. Alternatively, user can make the correction by picking the out the correct class from the table prompted by the *Set*-button. Special symbols, such as '?', '% ', '(', ',' etc., can be inserted in a similar manner with the *Sym*-button. More than one character can be deleted at the same time by first selecting the characters and then replacing them with a single character, space, or backspace. When

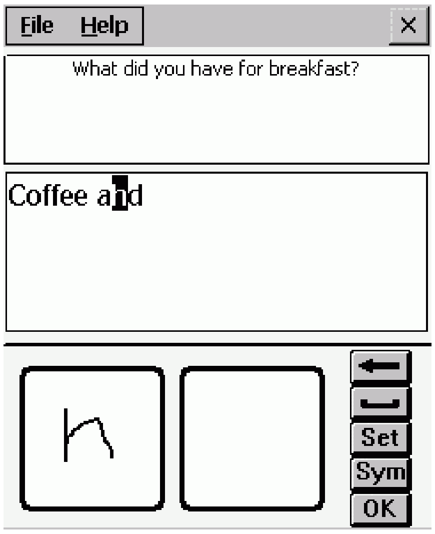

Figure 2: User interface of the questionnaire program. Questions to be answered are shown in the uppermost part of the window. Recognition results are shown in the middle part. Location of the text cursor is indicated with a blinking vertical line and selected text partitions are highlighted with black. Input characters are written into the two rectangular areas in the lower part of the window.

the answer is ready, it is submitted with the *OK*-button.

The recognition system is adapted between questions. Learning samples are labeled according to the recognition results of the latest characters written in the same positions or the class set by the user. All the letters and digits of the submitted answers are used as learning samples. Deleted characters and characters labeled according to them are abandoned from the learning set. However, if a single character is deleted and replaced with a new character immediately, both characters are kept in the learning set. With this policy, the user can change his mind on what he is writing without confusing the adaptation process.

4 Performance evaluation

The performance evaluation of a recognition system implemented in a real-world application, such as the questionnaire program described in section 3, is not a straightforward task. Recognition rate cannot be calculated directly as the true classes of the input characters are not known in advance. Manual examination and labeling of the characters afterwards is laborious and some of the characters are ambiguous also for human readers and their classes can only be concluded from the context.

Instead of the recognition rate, the performance of the recognition system can be evaluated by using such indices as recognition time, rejection rate, submitted characters vs. all input characters, characters submitted with one attempt vs. all input characters, or characters submitted with one attempt vs. all submitted characters. In addition, valuable information on the recognition and adaptation can be gained by simply questioning the users. Relevant questions are, for example, were the initial and final accuracies satisfactory, were the recognition errors understandable, and was the adaptation only beneficial or did it introduce some new errors.

5 Data

Three databases were used in the experiments. Database 1 and Database 2 were collected with a tablet. They consist of approximately 10 000 and 30 000 characters written by 22 and 24 subjects, respectively. The characters were written without any constrains on the writing style. The distribution of characters classes (0-9, a-Z, å, ä, ö, Å, Ä, Ö) is nearly even. Database 3 was collected with the hand-held device and it contains about 1 000 characters written by two persons. The first writer contributed approximately 300 characters and she was advised to imitate the different writing styles found from the Database 1 by the clustering algorithm. The second subject run the questionnaire program and wrote about 700 characters. The distribution of the characters is similar to that of the English language.

The resolution of Wacom ArtPad II tablet used for the first two databases is 100 lines per millimeter and the sampling rate is at maximum 205 data points per second. The pressure sensitive display of the Windows CE device is less accurate than the tablet: its resolution is approximately 4 lines per millimeter and the sampling rate is on average 65 points per second. In both cases, the data points consist of the x- and y-coordinates of the pen point.

6 Experiments and results

The first experiments were performed in order to evaluate what kind of effects the pruning of the prototypes by their stroke lengths has on the recognition and rejection rates. Characters of Database 2 were used as a test set and were matched against prototypes which were selected from Database 1 by a clustering algorithm. The number of prototypes per each class was seven. The second continuity condition of the DTW-algorithm was not applied. The pairs

Table 1: The effects of the pruning of the prototypes by their stroke lengths. α and β are the pruning parameters, p, E, R, and Δt stand for the share of the stroke pairs which satisfy the pruning condition, error rate, rejection rate, and change in recognition time in percentages, respectively.

α	β	p (%)	E (%)	R (%)	Δt (%)
∞	∞	0.00	23.11	0.00	0.00
3	10	0.17	23.09	0.00	-3.46
2	10	1.54	23.27	0.01	-11.40
1.5	10	5.12	24.23	0.11	-23.83

Table 2: The effects of the second continuity condition of the DTW-algorithm. c is the pruning parameter, E, R, and Δt stand for the error rate, rejection rate, and change in recognition time in percentages, respectively.

c	E (%)	R (%)	Δt (%)
1	23.11	0.00	0.00
0.5	23.09	0.11	-13.91
0.3	23.12	0.29	-27.34
0.2	23.25	0.55	-35.72
0.1	25.00	1.37	-44.24
0.05	34.94	5.71	-44.31

of stroke lengths of the best-matching prototypes and test characters were recorded and plotted. Pruning parameters α and β were selected so that most of these pairs would not satisfy the pruning condition of (3). The results of these experiments are summarized in Table 1. According to them, recognition time can be reduced approximately 11% while error and rejection rates remain practically unchanged.

Next, the effects of the second continuity condition of the DTW-algorithm were examined. Characters of Database 2 were classified according to 1-NN rule and using different values for the strictness parameter c of (2). The prototype set was the same as in the previous experiments but this time prototypes were not pruned on the basis of their stroke lengths. From Table 2, it can be seen that the second continuity condition is useful for decreasing the recognition time. When $c = 0.3$, recognition time is decreased nearly 30%, change in the error rate is insignificant, and rejection rate is less than 1%. Rejection rates are high with small values of c because the second continuity condition allows matching of strokes the lengths of which vary only a little.

The R4000 processor of the hand-held device has no floating point unit and the recognition is unbearable slow unless all the calculation are performed using only integer numbers. The recognition time was evaluated for the two

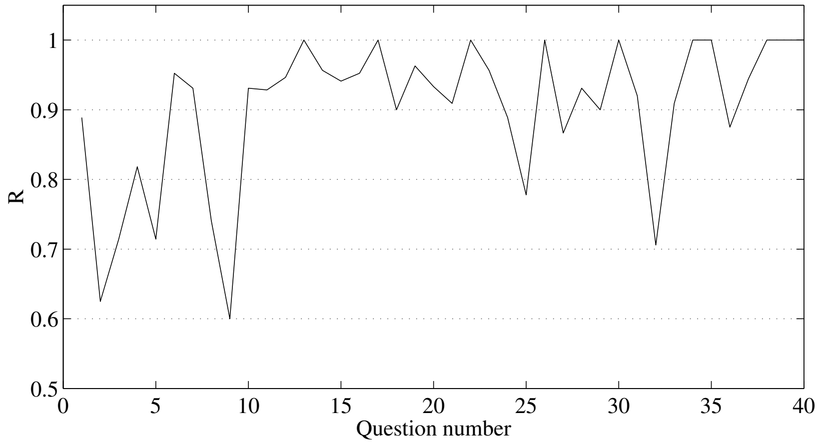

Figure 3: Evolution of the ratio of characters submitted with a single attempt to all submitted characters (R) during the run of the questionnaire program.

versions of the DTW-algorithm by classifying the characters of the second writer of Database 3 according to 1-NN rule. Characters of the first writer were used as a prototype set. In this case, the two prototype pruning rules, (2) and (3), were not applied. According to this experiment, the recognition time is reduced by 85% and the error rate is increased by 16% if floating point operations are replaced by integer operations.

Changing over to integer operations did not speed up the recognition enough. Therefore, it became important to examine how much data could be abandoned. The next experiments were performed with Database 3 and using different values for the decimation parameter n (see subsection 2.1). The integer version of the DTW-algorithm was applied. Otherwise, the experiments were similar to those described above. The most promising value for n was 2: recognition time deceased by 49% while the error rate increased only by 12%.

The performance of the recognition system was evaluated by calculating the average recognition time and the ratio R of characters submitted with a single attempt to all submitted characters for every answer in the questionnaire program. At the beginning of collection, average recognition time was about 300 ms. Due to adaptation, it increased during the first 10 questions and then remained around 480 ms for the last 30 questions. The ratio R, see Figure 3, was on the average better for the last questions than for the first ones.

7 Conclusions

The experiments showed that pruning of the prototypes on the basis of the stroke lengths and the additional continuity condition of the DTW-algorithm are useful: they both decrease the recognition time significantly without affecting the recognition accuracy. The other speedup methods, namely changing from floating point to integer operations and decimation of the data points, are also successful in reducing the recognition time but they deteriorate the recognition accuracy more. According to the results of the genuine on-line experiment, the initial accuracy is rather poor: on the average, every 5th submitted character had to be input at least twice. However, the proposed adaptation scheme was able to improve the accuracy so that at the end of the experiment only every 16th submitted character was input more than once.

References

1. E. Fix and J. L. Hodges. Discriminatory analysis—nonparametric discrimination: Consistency properties. Technical Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.
2. Vuokko Vuori. Adaptation in on-line recognition of handwriting. Master's thesis, Helsinki University of Technology, 1999.
3. D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison.* Addison-Wesley, 1983.
4. Jorma Laaksonen, Vuokko Vuori, Erkki Oja, and Jari Kangas. Adaptation of prototype sets in on-line recognition of isolated handwritten latin characters. In Seong-Whan Lee, editor, *Advances in Handwriting Recognition*, pages 489–497. World Scientific Publishing, 1999.
5. Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, 1997. Second Extended Edition.
6. Jorma Laaksonen, Jarmo Hurri, Erkki Oja, and Jari Kangas. Comparison of adaptive strategies for on-line character recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pages 245–250, 1998.
7. Vuokko Vuori, Jorma Laaksonen, Erkki Oja, and Jari Kangas. On-line adaptation in recognition of handwritten alphanumeric characters. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 792–795, 1999.