# Experimental Evaluation of a Trainable Scribble Recognizer for Calligraphic Interfaces

César F. Pimentel, Manuel J. da Fonseca, Joaquim A. Jorge

Departamento de Engenharia Informática, IST/UTL
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
pimentelcesar@hotmail.com, mjf@inesc.pt, jaj@inesc.pt

**Abstract.** This paper describes a trainable recognizer for hand-drawn sketches using geometric features. We compare three different training algorithms and select the best approach in terms of cost-performance ratio. The algorithms employ classic machine-learning techniques using a clustering approach. Experimental results show competing performance (95.1%) with the non-trainable recognizer (95.8%) previously developed, with obvious gains in flexibility and expandability. In addition, we study both their classification and learning performance with increasing number of examples per class.

## 1 Introduction

The increased interest drawn by pen computers has attracted considerable attention to handwriting and sketch recognition. This paper describes a trainable recognizer for hand-drawn sketches and geometric figures, which develops on our previous work. While the simple approach we had developed earlier achieved fairly high recognition rates, it does not allow users to specify new gestures, requiring hand-coding and careful tuning to add new primitives, which is a tedious and cumbersome task. This considerably limits the flexibility of our previous approach in accommodating new gestures, strokes and interaction primitives. The recognizer we present here is a considerable advance from the one presented in [10]. Instead of limiting ourselves to using the pre-defined fuzzy-logic based method for a fixed number of different stroke classes, it allows us to use one of several learning algorithms to teach the recognizer new classes.

While this paper describes some preliminary work, our trainable recognizer exhibits respectable recognition rates (95%) achieved at a moderate training cost using a simple Naïve Bayes approach. The paper studies three basic tried-and-true techniques, Naïve Bayes, KNN and ID3 and a weighted majority combination of the three. Surprisingly enough, the simplest method outperformed the weighted majority technique not only in training costs but also in recognition performance.

## 2 Related Work

In [10] we presented a fast, simple and compact approach to recognize scribbles (multi-stroke geometric shapes) drawn with a stylus on a digitizing tablet. Regardless of size, rotation and number of strokes, this method identifies the most common shapes used in drawings, allowing for dashed, continuous or overlapping strokes. The method described in [10] combines temporal adjacency, Fuzzy Logic and geometric features to classify scribbles with measured recognition rates over 95.8%. However, the algorithm devised, for its simplicity and robustness, makes it difficult to add new shapes. Each new primitive class added requires hand-coding and testing of new features in a slow, tedious and error-prone process.

Previous work on trainable gesture recognizers can be found in [23]. The author describes a trainable single-stroke gesture recognizer using a classic linear discriminator-training algorithm [7]. The goal of our work is somewhat different than Rubine's. Our purpose is to study the overall performance of several clustering algorithms in distinguishing between a large, dynamic and user-defined set of scribble classes. We have also focused our attention on choosing features that allow the recognizer to maintain a good recognition rate in case the user adds a new "complex" and "unexpected" scribble to the training set.

One notable difference from Rubine's recognizer is that our method allows the user to draw scribbles without any restriction and as close as possible to the intended shapes. This allows us to recognize virtually any complex shape we can think of, and also without the limitation of being single-stroked (i.e., without having to rule out crosses or arrows, contrarily to what happens in [23]). For these reasons, we sometimes say that our recognizer classifies scribbles instead of gestures. In order to accomplish this, we need to choose a richer feature set and more sophisticated training algorithms as compared to the classic linear discriminator with thirteen features presented in [23].

Our work is based on a first approach to recognize schematic drawings developed at the University of Washington [12], which recognized a small number of non-rotated shapes and did not distinguish open/closed, dashed or bold shapes. Tappert [24] provides an excellent survey of the work developed in the area of non-interactive graphics recognition for the introduction of data and diagrams in vectorial CAD systems.

### 2.1 Geometric Features

The main features used by our recognizer are ratios between perimeters and areas of special polygons as the ones presented in Figure 1. To achieve this, we start by computing the convex hull of each scribble points, using Graham's scan [17]. Using this convex hull, we compute three special polygons. The first two are the largest area triangle and quadrilateral inscribed in the convex hull [2]. The third is the smallest area-enclosing rectangle [11]. Finally we compute the area and perimeter for each special polygon to estimate features for the training and recognition process.

## 3 Training Algorithms

We have chosen three general algorithms for their simplicity, flexibility and robustness to implement our trainable recognizer. Each of these algorithms has different training and classification procedures. For training we need to gather a number of gestures as examples, and we need to have a class associated to each of these gestures. The class is, in other words, the name of the type of gesture (for example: `Ellipse`, `Triangle`, `Delete`), which is, of course, defined by the user. Training is therefore, an inductive procedure for it consists in learning a certain concept based on examples. It is obvious that some concepts are easier to learn than others are and that the more training examples we have, the more effective is our training.

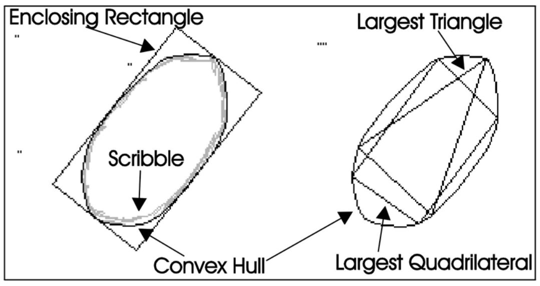

Figure 1: Special polygons

Since the training set is completely user-defined, the algorithms are able to identify, with more or less performance, any type of gesture we can think of. Because the learning algorithms see a gesture as a set of geometric features (and, therefore, continuous numeric features) the training procedure can be seen as supervised clustering.

In the three following sections we present a brief explanation for each of the algorithms used. We also analyze the performance and training costs of a weighted majority combination of the three.

### 3.1 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors is an old and simple machine-learning algorithm, which has proven to be very effective in most situations. Initial theoretical results can be found in [4] and an extensive overview in [7]. Furthermore, Bishop [1], studies on how the algorithm is related to the method of estimating probability densities.

The conceptual idea of the K-Nearest Neighbors algorithm is simple and intuitive. The estimated class for a given gesture instance (feature vector) is the most abundant class within the $k$ nearest neighbor examples of that instance, where $k$ is a set parameter. Neighborliness is measured wrt the vector distance between two instances. This explains why this algorithm works better with numeric features.

An essential step in the KNN algorithm is to normalize all feature values in training and classification instances. This normalization step prevents features with usually large values from having a disproportionately superior weight in the calculation of distances.

This algorithm has good classification efficiency and a fast training response. However, large training sets place considerable demands in both computer memory and classification time required.

### 3.2 Inductive Decision Tree (ID3)

ID3 (Inductive Decision Tree) is a decision tree learning algorithm thoroughly described in [18], [19] and [20]. Later, Quinlan [22] discussed details and implementation issues concerning decision tree learning in general. Our version of ID3 also features reduced-error pruning, as described in [21], to avoid over-fitting. Our approach can also handle continuous-valued attributes, using a method described in [8] and [9], and incorporates a pruning strategy discussed in [16] and [15].

ID3 is one of the most famous algorithms based on decision trees. Once a decision tree has been constructed during the training procedure, it will then be kept in memory and used for classification. Each non-leaf node is labeled with the name of the feature whose value must be tested, and the branches that originate from that node are labeled with the values (or intervals) that the feature can take. Given a gesture instance to be classified, we start from the tree's root, testing the respective feature and moving down through its respective branch. Eventually we will arrive at a leaf node labeled by the class name for that instance.

This algorithm is better suited to handle symbolic features or, at least, discrete-valued ones. Since our features consist mostly of continuous numeric values, it is necessary to transform them all into discrete values before training. This transformation discretizes each feature domain into several intervals and substitutes the feature values for the correspondent interval index. We perform this process also for the Naïve Bayes algorithm, as explained below.

ID3 presents average classification efficiency for continuous features. As is typical of decision-tree classifier methods the recognizer is very fast due to its low computational complexity.

### 3.3 Naïve Bayes (NB)

Documentation on Naïve Bayes' basic notions as well as other Bayesian Classifiers' can be found in [7]. Additionally, Domingos [6] presents an analysis on Naïve Bayes' performance with situations where the independence assumption does not hold (which is the most common situation in real world problems). A discussion concerning the method of estimating probabilities using the *m-estimate* can be found in [3].

The algorithm uses probabilities estimated from the training set. For a given instance (gesture), it computes the probability of each feature value belonging to each class. Then it combines these probabilities, to discover which class is most likely the instance. This procedure assumes that there are no statistic dependencies between features. While generally this is not the case, the simplification is often acceptable and frequently we do not have enough information about the dependencies between features. As a result, the algorithm is very fast in training and classifying, without sacrificing much on classification efficiency.

This algorithm also requires discretizing features in the same way as ID3. Even though the method cannot handle symbolic features, its classification efficiency in recognizing our gestures was proven to be better than any other algorithm we have tried.

### 3.4 Weighted-Majority (WM)

A description of the Weighted-Majority algorithm can be found in [13] and [14]. This algorithm is very commonly used in Machine Learning. It is adopted in situations where our chosen algorithm often performs a wrong classification while most alternative algorithms would succeed. Therefore, Weighted-Majority algorithm combines several algorithms to classify an instance and returns the mostly voted class. Some algorithms' votes carry more weight than others'. These weights are determined by the algorithms' classification efficiency.

We implemented a Weighted-Majority using the three previous algorithms (K-Nearest Neighbors, ID3 and Naïve Bayes). This way, whenever a class receives at least two votes, it will be returned as the result of classification. If, otherwise, each algorithm votes in a different class the result will be the class voted by Naïve Bayes, because it was proven to be the most truthful algorithm in this domain.

## 4 Experimental Assessment of the Algorithms

We will now evaluate the four training algorithms described in the previous section. The first section describes the data collection procedure. Next, we describe the training process and the learning evaluation of each algorithm. Finally we present and discuss the learning evolution of the different algorithms.

### 4.1 Data Collection

To train and evaluate the different learning algorithms described above we used a set of sample data collected using a Wacom PL-300 LCD digitizing tablet and a cordless stylus.

We asked 22 subjects to draw each multi-stroke shape twenty times using solid lines, ten times using dashed and ten times using bold lines. We also asked them to draw 30 times each uni-stroke shape. All these drawings yield a total of 9944 shapes. Figure 2 shows the different classes of shapes drawn.

We gave a brief description of the recognizer to the users, including the set of recognizable shapes. We also told them about the multi-stroke shape recognition capabilities and the independence of changes with rotation or size. Novice subjects had a short practice session in order to become acquainted to the stylus/tablet combination.

### 4.2 Training Process

We performed many test runs using the different algorithms described above. In most cases, we used twelve different gestures classes, namely, `Arrow`, `Circle`, `Cross`, `Copy`, `Delete`, `Diamond`, `Ellipse`, `Line`, `Move`, `Rectangle`, `Triangle` and `WavyLine`, as depicted in Figure 2. In all test runs, none of the gestures used in training were used for evaluating the algorithms. Furthermore, training and evaluation gestures were drawn by different subjects.

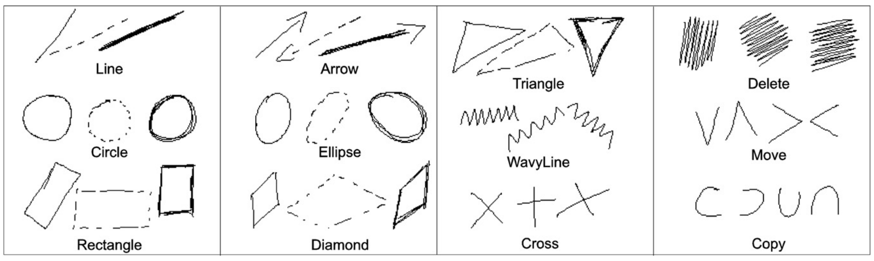

Figure 2: Classes of shapes used in the trainable recognizer.

Every gesture instance is represented by a vector of 24 geometric features, as explained in section 2.1. Since all recognition algorithms used perform clustering techniques, working with twelve classes implies that we will need many examples per class in order to be able to distinguish them well. If, for instance, we only needed to distinguish between three or four classes, the algorithms would be able to perform with better recognition efficiencies at fewer examples per class. For these reasons, the training sets we have created are considerably large (2610 examples).

Each training set includes several instances and their correspondent classes, whilst an evaluation includes only unclassified instances. After a given algorithm classifies an evaluation set, the estimated classes are compared with the real classes that generated the instances (the gesture that the individual intended to draw). This is the method we have used to extract all results presented in this paper.

### 4.3 Learning Evaluation

We quickly noticed that the Naïve Bayes algorithm presents the highest classification efficiency (around 95% when the number of training examples is high). This combined with very acceptable training and classification complexity, makes this algorithm the best suited for the task in hand.

The K-Nearest Neighbors algorithm can also yield good classification efficiency (around 92%), but classification times become unacceptable for large training sets. This is because the algorithm must scan all training examples, to calculate their distances from the given gesture. Much work has been done on preprocessing the training samples so that KNN does not need to scan them all during classification. Some papers on this matter can be found in [5]. This type of optimization, however, is not present in our implementation of KNN.

ID3 was the worst performer yielding classification efficiencies lower than 77%. Also, ID3's training complexity is high and classification complexity, while acceptable is higher than Naïve Bayes'. We think that the reason behind such low classification efficiencies is due to using continuous numeric features, not very suited for decision trees. Although there are more sophisticated ways to coalesce continuous features into intervals (in contrast to the method devised), it is not clear how this algorithm will ever yield high classification efficiency in our domain.

Finally, the three-algorithm Weighted-Majority requires, the sum of their individual training and classification times. As expected, it incurs in fewer mistakes than either algorithm alone, but it is not significantly better than the best of the three (Naïve Bayes). Despite making fewer mistakes, we observed that Weighted-Majority makes some mistakes that Naïve Bayes doesn't, when both KNN and ID3 are wrong.

## 4.4 Learning Evolution

We wanted to know how many examples of a gesture are sufficient to "teach" that gesture. To this end, we plotted recognition performance against training set size for the twelve solid shapes mentioned above, for each algorithm as shown in Figure 3.

We can see that, in order to distinguish between twelve different classes, all of the algorithms show the greatest classification efficiency increase up to 30 examples per class. This is the typical shape of what we call "the learning curve". Relatively speaking, we would say that Naïve Bayes and KNN exhibit acceptable classification efficiency with 50 examples per class, and tend to stabilize around 100 examples per class. We did not find ID3 to perform at an acceptable level for the reasons explained before.

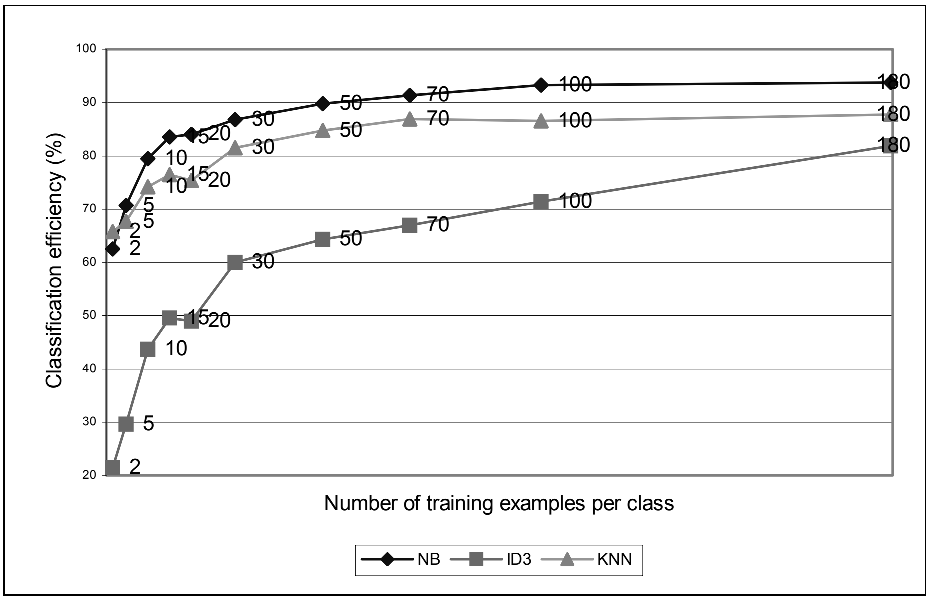

Figure 3: Learning evolution based on the number of examples per class.

We point out that the efficiency in recognizing a given gesture does not solely depend on its training instances. It also depends on how many distinct classes appear in the training set and how the given class differs from them. Based on the observations presented here we conclude that Naïve Bayes algorithm is the better suited for our domain. From now on all the results and recognition rates presented pertain to this algorithm.

## 5 Comparison With The Previous Recognizer

We will now make an experimental evaluation of the trainable recognizer using a set of data collected as described before, comparing its performance to the recognizer described in [10].

### 5.1 Performance Evaluation

We selected a subset of the data described before. The values in the confusion matrix of Figure 4 describe the recognition results by the Naïve Bayes algorithm for the twelve solid shape classes, described in the previous sections. We used 2610 scribbles to train the recognizer under the conditions explained above and 959 other scribbles (from a different set of users) to evaluate it. There were 895 correct classifications out of 959 scribbles, corresponding to a global classification efficiency of 93.3%.

As shown by the confusion matrix, sometimes `arrows` are classified as `crosses` and vice-versa. This happens because these scribbles have two strokes. The number of strokes is one of the most important features for distinguishing arrows and crosses.

| | Arrow | Circle | Cross | Copy | Delete | Diamond | Ellipse | Line | Move | Rectangle | Triangle | WavyLine |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Arrow | 84.8 | | 5.6 | | | 1.6 | | 5.6 | | 0.8 | | 1.6 |
| Circle | | 94.0 | | | 2.0 | | 4.0 | | | | | |
| Cross | 5.9 | | 94.1 | | | | | | | | | |
| Copy | | 1.2 | | 91.7 | | | 6.0 | | | 0.6 | | 0.6 |
| Delete | | | | | 89.1 | | | 2.2 | 2.2 | | | 6.5 |
| Diamond | | | | | | 98.7 | | | | | 1.3 | |
| Ellipse | | 14.0 | | | | | 86.0 | | | | | |
| Line | | | | | | | | 99.2 | | | | 0.8 |
| Move | | | | | | | | | 100 | | | |
| Rectangle | | | 2.0 | | | 2.0 | | 2.0 | | 86.0 | | 8.0 |
| Triangle | | | | | | | | | | | 96.1 | 3.9 |
| WavyLine | | | | | | | | 1.7 | | | | 98.3 |

Figure 4: Confusion matrix for solid shapes (values in percentage).

Arrows are also sometimes misclassified as lines due to their geometrically similar shape. Copies are seen as ellipses when they are drawn as a nearly closed shape. If deletes are made a bit longer than usual, there may be some confusion with wavylines. Many of the ellipses are classified as circles, a very intuitive failure mode. Eight percent of the rectangles are taken as wavylines (perhaps the worst misclassification) because of the similarities between their enclosing rectangles.

Finally, we wish to explain that "strange" misclassifications such as mistaking a rectangle for a wavyline can be easily solved by adding appropriate extra features. Such features should present distant values on each gesture class. The training algorithm will automatically handle the task of using that feature to better distinguish the classes at stake. The features we have chosen have proven useful for these twelve classes, and since these are relatively diverse, we believe the recognizer will behave well for a broad variety of classes. In summary, our recognizer performs considerably well, presenting an overall classification efficiency of 93.3% in this experiment.

### 5.2 Performance Comparison

We will now present the result of a different experiment specifically intended to establish a comparison with the (non-trainable) fuzzy recognizer described in [10]. While the fuzzy recognizer presents 95.8% classification efficiency with all 12 gestures, we must point out that this result was obtained considering that some of the gestures called "shapes" (Arrow, Circle, Diamond, Ellipse, Line, Rectangle and Triangle) may exist in three different versions (Solid, Dashed and Bold). This performance, however, does not consider distinguishing line-style, which make for non-homogenous classes, thus impairing performance. To establish a "realistic" assessment, we have performed a similar experience with our trainable recognizer, described as follows.

We have trained the recognizer with 6022 gesture instances of all 12 classes, this time, considering the three different line-styles (Solid, Dashed and Bold). After, we ran the recognizer through an evaluation set of 3869 gesture instances. Again, this was generated by a different set of people. The recognizer performed with a classification efficiency of 95.1%, higher than the previous case.

## 6 Conclusions

We have presented a simple trainable scribble recognizer for multiple stroke geometric shapes. While some of the work is still preliminary, the Naïve Bayes approach provides for good performance and classification efficiency, making it our method of choice. The classification rates measured under realistic conditions (95.1%) are promising. However, the trainable classifier requires many samples to achieve this value. While this can be perfected, the amount of work required is considerably less than the hand tuning previously required to add new shapes, for a performance comparable to that of the non-trainable recognizer.

## References

1. Bishop C. M. *Neural Networks for pattern recognition*. Oxford, England: Oxford University Press, 1995.
2. Boyce J. E. and Dobkin D. P., *Finding Extremal Polygons,* SIAM Journal on Computing 14 (1), pp.134-147. Feb. 1985.
3. Cestnik B. Estimating probabilities: A crucial task in machine learning. *Proceedings of the Ninth European Conference on Artificial Intelligence* (pp. 147-149). London: Pitman, 1990.
4. Cover T. and Hart P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13, 21-27, 1967.
5. Dasarathy B. V. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, Los Alamitos, CA, IEEE Computer Society Press, 1991.
6. Domingos P. and Pazzani M. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Proceedings of the 13th International Joint Conference on Machine Learning* (pp. 105-112), 1996.
7. Duda R. and Hart P. *Pattern classification and scene analysis*. New York: John Wiley & Sons, 1973.
8. Fayyad U. M., *On the induction of decision trees for multiple concept learning*, (Ph.D. dissertation). EECS Department, University of Michigan, 1991.
9. Fayyad U. M. and Irani K. B. Multi-interval discretization of continuous valued attributes for classification learning. In R. Bajcsy (Ed.), *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (pp.1022-1027). Morgan Kaufmann, 1993.
10. Fonseca M. J. and Jorge J. A. *Experimental Evaluation of an On-line Scribble Recognizer*, Accepted for publication in the Pattern Recognition Letters Special Issue: PRL-RECPAD 2000, 2001.
11. Freeman H. and Shapira R., *Determining the minimum-area encasing rectangle for an arbitrary closed curve,* Communications of the ACM 18 (7), 409-413, July 1975.
12. Apte, A., Vo, V., Kimura, T. D. *Recognizing Multistroke Geometric Shapes: An Experimental Evaluation*. In Proceedings of UIST'93. Atlanta, GA, 1993.
13. Littlestone N. and Warmuth M. *The weighted majority algorithm* (Technical report UCSC-CRL-91-28). Univ. of California Santa Cruz, Computer Engineering and Information Sciences Dept., Santa Cruz, CA, 1991.
14. Littlestone N. and Warmuth M. The weighted majority algorithm. *Information and Computation* (108), 212-261, 1994.
15. Malerba D., Floriana E. and Semeraro G. A further comparison of simplification methods for decision tree induction. In D. Fisher & H. Lenz (Eds.), *Learning from data: AI and statistics*. Springer-Verlag, 1995.
16. Mingers J. An Empirical comparison of pruning methods for decision-tree induction. *Machine Learning*, 4(2), 227-243, 1989.
17. O'Rourke J. *Computational geometry in C*, 2nd edition, Cambridge University Press, 1998.
18. Quinlan J. R. Discovering rules by induction from large collections of examples. In D. Michie (Ed.), *Expert systems in the micro electronic age*. Edinburgh Univ. Press, 1979.
19. Quinlan J. R. Learning efficient classification procedures and their application to chess end games. R. S. Michalski, J. G. Carbonell and T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann, 1983.
20. Quinlan J. R. Induction of decision trees. *Machine Learning*, 1(1), 81-106, 1986.
21. Quinlan J. R. Rule induction with statistical data – a comparison with multiple regression. *Journal of the Operational Research Society*, 38, 347-352, 1987.
22. Quinlan J. R. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
23. Rubine D. H. *Specifying Gestures by Example*, SIGGRAPH'91 Conference Proceedings, ACM, 1991.
24. Tappert, C. C., Suen, C. Y., Wakahara, T. *The state of the art in on-line handwriting recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence 12~(8), 787—807, 1990.