Segmenting Handwritten Text Using Supervised Classification Techniques

Yi Sun*, Timothy S. Butler†, Alex Shafarenko‡, Rod Adams§, Martin Loomes¶, Neil Davey‖
Department of Computer Science
Faulty of Engineering and Information Sciences
University of Hertfordshire
College Lane, Hatfield
Hertfordshire AL10 9AB
*Y.2.Sun@herts.ac.uk
†t.s.butler@herts.ac.uk
‡a.shaferenko@herts.ac.uk
§r.g.adams@herts.ac.uk
¶m.j.loomes@herts.ac.uk
‖N.Davey@herts.ac.uk

***Abstract*— Recent work on extracting features of gaps in handwritten text allows a classification into *inter-word* and *intra-word* classes using suitable classification techniques. In this paper, we apply 5 different supervised classification algorithms from the machine learning field on both the original dataset and a dataset with the best features selected using mutual information. The classifiers are compared by employing McNemar's test. We find that SVMs and MLPs outperform the other classifiers and that preprocessing to select features works well.**

I. Introduction

In this paper, we address the problem of identifying word boundarys in handwritten text: a process known as word segmentation. We make use of a selection of contemporary classification algorithms, such as multi-layer perceptrons, support vector machines, and Gaussian mixture models.

Surprisingly, little attention has been paid to the word segmentation problem by the neural net community. Nevertheless, recent work on extracting features of gaps between pieces of handwritten text allows for attaining segmented words by classifying gaps to *inter-word* and *intra-word* classes directly [3]. In this paper we try to find a suitable classifier to automatically segment so-called *digital ink*, i.e. graphically enhanced fragments of pen trace representing handwritten words, shapes and symbols of the sort that usually appear on paper when real ink is used for writing. Further details about the problem domain can be found in the next section. The previous work was done by using statistical methods to classify gaps into two classes based on one significant feature, named *river*, which is described in more detail in the following section. Each stroke involves an array of time-stamped sample points. However, as indicated in [12], exceptions are commonplace because of flourishes in writing styles with leading and trailing ligatures in handwriting. It is important to consider other possible features, as combinations of variables can provide significant information which is not available in any of the individual variables separately. The task is therefore to propose a classifier which can make as few errors as possible, based solely on the set of features.

In this work, we test 5 different supervised classification learning algorithms from the machine learning field to categorise gaps. We are also interested in selecting the most significant features. Since there is a proportion of gaps which can be classified with 99 percent accuracy in terms of the value of *river* directly, we apply these classification techniques for those patterns which cannot be judged easily by the feature *river*.

We expound the problem domain in the next section. In section III, we introduce the datasets used in this paper. We explain how we select a subset of features in terms of mutual information. In addition, a fuzzy dataset is obtained by using thresholds of *river*. Section IV briefly lists the classifiers used in our experiments and gives all the experimental results. We analyse the classification results by applying McNemar's test as well. The paper ends in section V with a discussion.

II. Problem Domain

Despite the widespread use of office computers, handwriting has been and remains an important mode of capturing and annotating textual information. Computer-assisted handwriting is an increasingly important part of the general interface between the electronic media and the business world. Indeed, apart from the niche market of Personal Digital Assistants (including mainly smart phones and palmtop PCs), where the use of pen input devices is motivated primarily by their greater compactness, the mainstream computing technology now includes so called Tablet PCs. A tablet PC is a portable computer with a sensitive screen and a digital stylus, which is used as the main, or even the only, input device. The OS of a tablet PC is augmented with components that can handle *digital ink*. It is important to understand the difference between the digital ink and character-recognition interfaces. While the latter is merely a form of machine intelligence

capable of recognising letters of an alphabet so that a keyboard can be replaced by an equivalent, but more compact, tablet and pen, the digital ink represents a separate form of input. It persists in documents as long as desirable for the author or/and readers. More importantly though, it is *processed* in its native form, i.e. as a graphical object. Words may be inserted, deleted or replaced at will without first being converted into a semantically focused form, such as an ASCII string. Such a conversion may happen eventually, when the final copy is produced.

There is therefore a fine balance for digital ink applications, namely one between the graphical form and semantic substance. One would like to benefit from the immediacy of pen input, its highly informal nature and potentially unlimited alphabet of letters, features and symbols, while at the same time having the computer penetrate the *structure* of the ink to the extent that it is necessary to be able to edit distinct parts of it. The depth of such penetration needs to be no more than superficial, down to a level of large self-contained units, such as lines and words, where the structuring is fairly well (albeit informally) defined. On the other hand, if no analysis is done of the ink input, then it is not really treated as handwriting, but as a general freehand graphical input. Consequently computer assistance (in the form of automatic placement, formatting and linkage with the rest of the document environment) would be very limited.

In this paper we focus on one level of the semantic penetration of pen input: the level of words. By 'word' we mean a group of pen strokes that have lexical significance, i.e. one that represents a word in a human language or a distinct symbol that can be used as a word. We wish to automatically segment digital ink represented as a *sampled pen trace* into word fragments purely on the basis of spatiotemporal relations between consecutive strokes, ignoring any meaning that may be represented by each such stroke. This has been a known problem in handwriting recognition research as well, although in this area of technology, word segmentation is seen merely as a precursor to full character recognition. In their recent comprehensive survey of handwriting recognition research, Plamondon and Srihari state that "prior to any recognition the acquired data is generally preprocessed to... segment the signal into meaningful units" [12].

The history of word segmentation research is delimited by the survey [12] and the one 10 years earlier [16], which is also referenced in [12]. The significant achievements reported in [16] for this area are confined to straightforward geometric segmentation using convex shells [8] with some consideration given to stroke timing. It is noteworthy that these early proposals have not been developed any further as is evidenced by [12]. One can only speculate about the reason why no further progress has been reported. Our experience shows that simple segmentation methods are prone to error due to an individual writer's idiosyncrasies as well as the fact that these methods fail to capture more subtle structural and temporal signals which would strengthen the basis for segmentation. More recent work is attempting to improve structure recognition by introducing hierarchical agglomerative clustering, see [14], [9] in a broader context of automatic structural analysis of handwritten document. These in our opinion are interesting approaches, though they are susceptible to writing idiosyncrasies while being insensitive to any recurrent features of the language (or symbolic system) used by the writer.

The variability of one's writing style as well as the inherent diversity of writers would strongly advocate an adaptive solution. The solution would not be confined to any specific *ad hoc* metric of the pen trace as the basis of segmentation, but would accommodate a reasonably large set of these metric, taking into account both prime features (such as the size and duration of inter-stroke gaps) as well as any secondary ones which may be significant. Such features are still proposed on the basis of their plausibility, without much formal basis or a priori evidence. However, we have been guided by [13] where a thorough geometric and temporal classification was provided for a pen gesture recogniser. To give an idea of the sort of features that were being used there, we illustrate some of them in figure 1. It presents a single pen stroke with its bounding box. The features x and y as shown give the dimensions of the bounding box and the angle α is linked with its aspect ratio. The distance s is between the end points of the stroke, and β is the angle between the line connecting those points and the vertical. Finally, if θ_i is the angle between two consecutive pen segments of the stroke, i and $i+1$ then one can use the feature

$$\sigma = \sum_{i=1}^{n-1} \theta_i$$

as a measure of curvature. The proposed features were not all purely geometric; there were a few related to the time interval of the stroke and the speed of the pen tip. Note that most of these features are inapplicable to inter-stroke gaps, but some still make sense, e.g., x, y β, etc. We have introduced a gap feature which has proven especially useful for our purposes. We call it *river width* or *river* for short, following Fox and Tappert [8]. The river of a gap is the shortest distance between two consecutive strokes, i.e. the length of the shortest chord drawn between pen position samples from neighbouring strokes, as shown in figure 2. Two rivers are indicated there by double-headed arrows.

We have expanded the set proposed therein by our own form factors, see [3] for each pen stroke. The pen trace has thus been abstracted to a sequence of stroke and gap, where each gap is represented by 14 variables. In this work, we are interested in classifying gaps, so we ignore the strokes. A human reader has annotated the gaps in our experimental traces as either intra-word or inter-word by recognising the words in the language. Thus the task is to search for a classification method which can produce the same annotations with as few errors as possible.

III. The Description of The Datasets

A. Gaps Datasets

In this paper, we present experimental results on the gap datasets. The *original* gap dataset includes 2482 data points

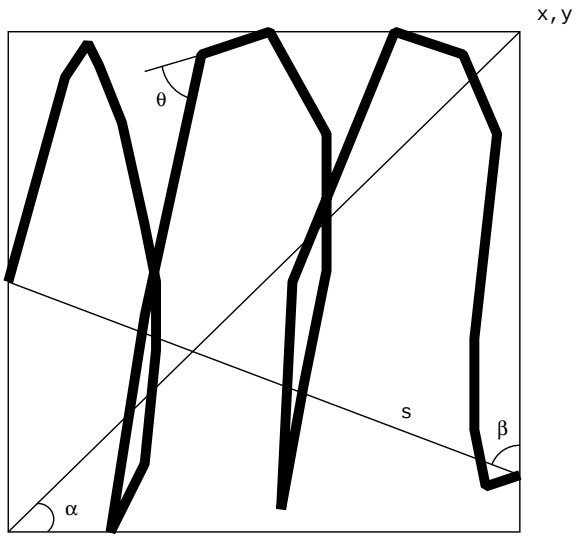

Fig. 1. An illustration: the sort of features of a single pen stroke with its bounding box.

Fig. 2. An illustration: two rivers of gaps are shown by double-headed arrows.

labeled by *inter-word* and 4980 *intra-word*. In the experiments, $2/3$ of the data points from the dataset are used for training, while $1/3$ for test. We do experiments with all 14 features and *reduced* features involving the 8 most significant to the classification found by analysing mutual information as discussed in section III-B.

B. Feature Extraction by Using Mutual Information

The features associated with gaps are reduced by employing mutual information. The mutual information of two variables is a measure of the common information shared between them [10]. In this work, the two variables are the class variable c and the feature variable x. c may take one of two values and x one of 14 features. The bigger the value of the mutual information, the more common information is shared. If two variables are independent, their mutual information is zero. An advanced treatment of feature extraction using mutual information maximization can be found in [5].

Assuming data points are generated from C classes (In this paper, $C = 2$). Mutual information, denoted by MI, is given by [4]

$$MI = H(c) - H(c|x), \tag{1}$$

where $H(c)$ is the entropy of the classes prior probability $P(c_i)$ given by

$$H(c) = -\sum_{i=1}^{C} P(c_i) \log P(c_i), \tag{2}$$

and $H(c|x)$ is conditional entropy having the form, as follows

$$H(c|x) = -\sum_{i=1}^{C} P(c_i, x) \log P(c_i|x), \tag{3}$$

where $P(c_i, x)$ are the joint probability distributions, and $P(c_i|x)$ are posterior probabilities. Equation (3) can be further written as

$$H(c|x) = -\sum_{i=1}^{C} P(c_i) \int p(x|c_i) \log P(c_i|x)\ dx. \tag{4}$$

Note that $\int p(x|c_i) \log P(c_i|x) dx$ is the expectation of $\log P(c_i|x)$ given the probability density $p(x|c_i)$.

Empirically the conditional entropy $H(c|x)$, which is based on the probability density function of the variable x, can be approximated as follows, when considering two classes:

$$\begin{aligned} H(c|x) \quad \approx \quad & -\frac{1}{N_1} P(c_1) \sum_{k=1}^{N_1} \log P(c_1|x^k) \\ & -\frac{1}{N_2} P(c_2) \sum_{l=1}^{N_2} \log P(c_2|x^l), \end{aligned} \tag{5}$$

where x^k and x^l denote the feature values given that the data points are generated from two densities $p(x|c_1)$ and $p(x|c_2)$, respectively. N_1 and N_2 are the number of samples from the two distributions, respectively. Here we use the arithmetic average to approximate the expectation of $\log P(c_i|x)$. As the sample size N tends to infinity, then the arithmetic average tends to equal the common mean of each variable (weak law of large numbers).

To sample a large number of independent points from two probability density functions, we need to estimated them first, and then a sufficient number of data points: N_1 plus N_2, are sampled from the two estimated distributions. We employ two Gaussian mixture models to model the distributions of the gaps data collected from the class *inter-word* denoted by c_1 and *intra-word* denoted by c_2. The *expectation-maximisation* (EM) algorithm [6] is used for finding parameters of each model. A mixture distribution having M components (in this work, $M = 5$) can be calculated using:

$$p(x|c_i) = \sum_{j=1}^{M} p(x|j, c_i) P(j), \tag{6}$$

where $P(j)$ are mixing coefficients and

$$p(x|j, c_i) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{-\frac{1}{2\sigma_j^2}(x - \mu_j)^2\right\}, \quad (7)$$

where μ_j and β_j are mean and variance of each component j respectively. More details about Gaussian mixture models can be found in [1]. Then $500,000$ data points were sampled from these two distributions. Finally, the posterior probability can be computed using Bayes' theorem

$$P(c_i|x) = \frac{P(c_i)p(x|c_i)}{\sum_{i=1}^{C} P(c_i)p(x|c_i)}. \quad (8)$$

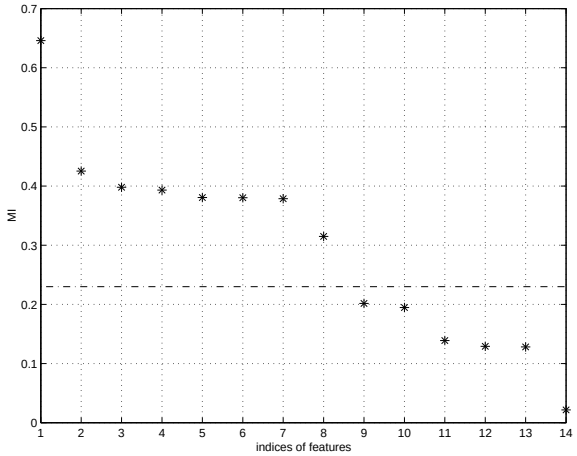

Fig. 3. Mutual information of class variable and each feature of gaps: each value is shown as a star sign. The background are dashed lines that are major grid lines to the current axes. The horizontal dash-dot line denotes the cut off value.

Figure 3 shows the mutual information of each feature with the class variable sorted by their values. As shown, there is a reasonable "jump" from the ninth value to the eighth. We ignore those features indexed from 9 to 14. Thus 8 features with mutual information values more than 0.3, a subset of all of the features, can be obtained.

C. The Fuzzy Dataset with Thresholds

As seen in Figure 3, there is one feature which is the most significant to classification, named *river*. Since it measures the shortest distance between samples in adjacent strokes, gaps between words usually have a larger value than gaps within words. One can expect to benefit from this variable as much as possible, though exceptions often occur with variety in writing styles as mentioned in the introduction section. Two boundaries of the values of *river* can be determined as displayed in figure 4. In this figure, the river values increase from left to right. Boundary 1 specifies a *river* value, on the left of which one can ensure that the probability that the gap belongs to class *intra-word* is not less than 99 percent; while boundary 2 specifies another value of *river*, on the right of which the probability that the gap belongs to class *inter-word* is not less than 99 percent. Then the whole dataset is filtered by means of these two thresholds. In this way, a sub-dataset called *fuzzy*, whose values of the *river* feature are within these two boundaries, is obtained. This subset therefore consists of 3361 gaps that cannot easily be classified by the *river* feature.

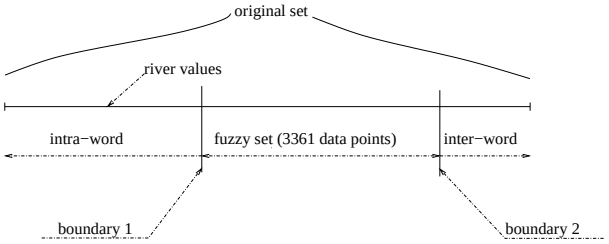

Fig. 4. A diagram: explaining how the fuzzy dataset is generated with two boundaries.

IV. Experimental Results with Supervised Classifiers

A. Supervised Classifiers

In this section, we first list the supervised classifiers used in our experiments. Readers who are interested in those classification techniques can follow the references to learn more.

- Logistic discrimination analysis (LDA) [1];
- K-nearest neighbor classification (KNN) [11];
- Guassian mixture model (GMM) [1];
- Multi-layer perceptron (MLP) using scaled conjugate gradients algorithm [1];
- Support vector machine (SVM) using Gaussian kernel [15].

Parameters of each class-condition density were estimated from the training dataset in the GMM. For the MLP, a two-layer architecture was set up, since it has been proved for classification tasks that the MLP with sigmoidal activation function and two layers of weights can approximate any decision boundary to arbitrary accuracy [2].

B. Experiments

Experiments were performed on both the original dataset and the fuzzy dataset with all 14 features and the selected set of 8 features. The user-chosen parameters for each classifier were selected by cross-validation, where the training set was divided into 10 partitions. 9 partitions were used to train the model and the other one was used as a validation set. The SVM experiments were completed using LIBSVM, which is available from the URL
`http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
The others were implemented using the NETLAB toolbox, which is available from the URL
`http://www.ncrg.aston.ac.uk/netlab/`.

In Table I, we present all the user-chosen parameters attained by using cross-validation.

TABLE I

USER-CHOSEN PARAMETERS FROM CROSS-VALIDATION. K DENOTES THE NUMBER OF NEIGHBOURS; $nc1$ AND $nc2$ ARE THE NUMBER OF GAUSSIAN MODELS IN EACH MIXTURE; j SIGNIFIES THE NUMBER OF HIDDEN UNITS IN THE MLP; A IS THE UPPER BOUND OF COEFFICIENTS α_i IN THE SVM; AND σ IS WIDTH OF RADIAL BASIS FUNCTION.

	KNN (K)	GMM ($nc1$, $nc2$)	MLP (j)	SVM (A, σ^2)
fuzzy 8	9	6, 6	8	25, 0.16
fuzzy 14	9	6, 4	5	20, 0.1
orig. 8	5	8, 9	15	25, 0.25
orig. 14	5	9, 9	5	5, 0.16

C. Classification Results

Classification results for each test dataset with different supervised classifiers are displayed in Table II. The accuracy is defined as the number of correct classified patterns over the number of total patterns in the test set. The results to the GMM and MLP shown in Table II are average of 10 repetitions with different random initial conditions.

TABLE II

RESULTS ON GAP DATASETS: ACCURACY RATE %

	LDA	KNN	GMM	MLP	SVM
fuzzy 8	86.0	90.1	86.6	92.1	92.2
fuzzy 14	87.5	89.2	84.5	91.5	92.5
orig. 8	92.7	93.8	92.0	95.3	95.8
orig. 14	93.2	93.8	90.4	96.1	96.2

Table II shows that using the reduced features as found by mutual information one can obtain a result as good as using all 14 features when employing the KNN, MLP and SVM. In addition, it also suggests that the MLP and SVM provide more accurate classification than the LDA, KNN and GMM classifiers. Interestingly, one can work on the fuzzy dataset and still achieve comparable results. The values given in the first two rows of Table II for the dataset are the accuracy rate for just the fuzzy gaps. Since the rest of original dataset has already been classified with 99% accuracy, the classification for the whole dataset achieved by this quicker method can be calculated. For instance, the SVM classifier gives a full classification rate for the whole dataset, when processing a dataset involving 3361 fuzzy gaps among all 7462 gaps, as follows,

$$\frac{3361}{7462} \times 92.5\% + \frac{7462 - 3361}{7462} \times 99\% = 96.1\%.$$

D. Statistical Test for Comparing Supervised Classification Learning Algorithms

Our primary goal is to choose the best learning algorithm for recognising the two class gaps. Looking at Table II, it can be seen that there is no big difference between the MLP and SVM algorithms. As addressed in [7], McNemar's test can be used for determining whether one learning algorithm is better than another on a special task with acceptable the probability of incorrectly detecting a difference when no difference exists. Thus we apply McNemar's test for comparing these two algorithms. In addition, we provide results of McNemar's test on the KNN and SVM as a comparison.

We first calculate the contingency table assuming there are two algorithms I and II, illustrated in Table III [7], where

TABLE III

2×2 CONTINGENCY TABLE

n_{00}	n_{01}
n_{10}	n_{11}

n_{00} is number of samples misclassified by both algorithms; n_{01} number of samples misclassified by algorithm I but not II; n_{10} number of samples misclassified by algorithm II but not I; n_{11} are correctly classified by both algorithms.

McNemar's test has a chi-square distribution with 1 degree of freedom [7]. Quantity χ^2 is computed as follows:

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}. \tag{9}$$

The null hypothesis assumes that the performance of two different learning algorithms is the same, i.e. $n_{10} = n_{01}$. The P-value from a chi-square value is computed with McNemar's test. Since small P-values suggest that the null hypothesis is unlikely to be true, we may reject the null hypothesis if the probability that $\chi^2 \geq 3.84$ is less than 0.05 [7].

TABLE IV

RESULTS OF MCNEMAR'S TEST FOR COMPARING THE MLP WITH THE SVM AND THE KNN WITH THE SVM ALGORITHMS.

	mlp-svm		knn-svm	
dataset	χ^2	P-value	χ^2	P-value
fuzzy 8	0.77	0.38	7.22	0.0072
fuzzy 14	1.47	0.23	13.28	0.0003
orig. 8	1.80	0.18	22.52	0.0001
orig. 14	0.62	0.43	31.65	0.0001

Table IV displays results for comparing the MLP with the SVM and the KNN with the SVM algorithms. The χ^2 for the MLP and SVM is an average calculated over the 10 runs. Looking at the third column, since all P-values are greater than 0.05, we cannot reject the null hypothesis which suggests that applying the MLP and SVM learning algorithms to construct classifiers for this application can achieve the same classification results. In addition, since the SVM outperforms the LDA, KNN and GMM, as seen in Table II, one can expect that the P-value should be smaller than 0.05 when comparing

them with the SVM. This is illustrated in the last column where the KNN and SVM classifier results are compared.

V. Discussion

In this paper, we apply a variety of contemporary classification algorithms to the word segmentation problem. We report classification results obtained by using 5 different supervised classifiers: LDA, KNN, GMM, MLP and SVM. The various classifiers are compared by McNemar's test. The results show the best result can be achieved by using non-linear classification techniques: the MLP and SVM algorithms. Mutual information is employed to select the most significant subset of features.

The results show the smaller set of features characterises the data as well as the full set. One of the features allows for 99% correct classification of roughly half the data by simple thresholding. Removing these data points leaves a reduced dataset which can then be classified using the more sophisticated non-linear techniques. The results show that this work well and it is faster than using the full dataset.

References

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*, New York: Oxford University Press, 1995.

[2] E. K. Blum and L. K. Li, "Approximation theory and feedforward networks," *Neural Networks*, vol. 4, no. 4, pp. 511–515, 1991.

[3] T. Butler, *Department of computer Science*. Ph.D. thesis, University of Hertfordshire, Department of Computer Science, University of Hertfordshire, Hertfordshire, UK.

[4] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., 1991.

[5] K. Torkkola, "Feature extraction by non-parametric mutual information maximization", *Journal of machine learning Reseach*, vol. 3, pp. 1415-1438, 2003.

[6] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *J. Roy. Stat. Soc. B*, vol. 39, pp. 1–38, 1977.

[7] T. G. Dietterich, "Approximate statistical tests for comparing supervised classifi cation learning algorithms", *Neural Computation*, vol. 10, no. 7, pp. 1895-1923, 1998.

[8] A. S. Fox and C. C. Tappert, "On-line external word segmentation for handwriting recognition", *Proceedings of the Third International Symposium on Handwriting and Computer Applications*, 1987, pp. 53–55.

[9] Y. Li, Z. Guan, H. G. Wang, G. Z. Dai and X. S. Ren, "Structuraizing freeform notes by implicit sketch understanding", *AAAI Spring Symposium on Sketch Understanding*, 2002.

[10] D. Michie, D. J. Spiegelhalter and C. C. Taylor, (Eds), *Machine Learning, Neural and Statistical Classifi cation*, Ellis Horwood, 1994.

[11] I. T. Nabney, *Netlab Algorithm for Pattern Recognition*, Springer, 2001.

[12] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey", *IEEE Transactions on Pattern Analysis and machine Intelligence*, V 22, no. 1, pp. 63-84, 2000.

[13] D. Rubine, "Specifying gestures by example", *Computer Graphics*, vol. 25, no. 4, pp. 329-337, 1991.

[14] E. Saund, J. Mahoney, D. Fleet, D. Larner and E. Lank, "Perceptual organization as a foundation for intelligent sketch editing", *AAAI Spring Symposium on Sketch Understanding*, 2002.

[15] B. Scholköpf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, 2002.

[16] C. C. Tappert, C. Y. Suen and T. Wakahara, "The State of the art in on-line handwriting recognition", *IEEE Transactions on Pattern Analysis and machine Intelligence*, vol. 12, no. 8, pp. 787-808, 1990.