

# Recognition of On-line Handwritten Mathematical Formulas in the E-Chalk System

Ernesto Tapia and Raúl Rojas

Free University of Berlin, Institute of Computer Science  
Takustr. 9, D-14195 Berlin, Germany  
{tapia|rojas}@inf.fu-berlin.de

## Abstract

In this article, we present a system for the recognition of on-line handwritten mathematical formulas which is used in the electronic chalkboard (E-chalk), a multimedia system for distance-teaching. We discuss the classification of symbols and the construction of the tree of spatial relationships among them. The classification is based on support vector machines and the construction of formulas is based on baseline structure analysis.

## 1. Introduction

Systems for the recognition of mathematical formulas have been studied for many years [1, 2]. In *off-line* recognition, formulas written or printed are given in the form of *images* or *bit-maps* (*static representation*). In *on-line* recognition, computers with pen devices (graphics tablets, contact sensitive whiteboards) store the data as “*digital ink*”, a *dynamic representation* which is in essence a sequence of points with temporal information.

Normally, the first step in a formula recognition system is to divide static or dynamic data into groups of strokes (*segmentation*), which are interpreted as single objects. A list of objects and their attributes (location, size, etc.) is returned. The only missing attribute for an object is its identity, which is determined using a classifier. In the second step, some structural analysis technique is applied to obtain a hierarchical structure of the expression which describes the mathematical relationships among the symbols.

With the advent of pen based devices (like PDAs, tablet PCs, etc.) the interest on on-line handwritten symbol recognition has increased. The use of such a devices make more natural the interaction between the user and the computer. In the case of mathematical notation, it is easier for a user to draw a complex mathematical expression than to type the same expression in a string language such as  $\LaTeX$  or to en-

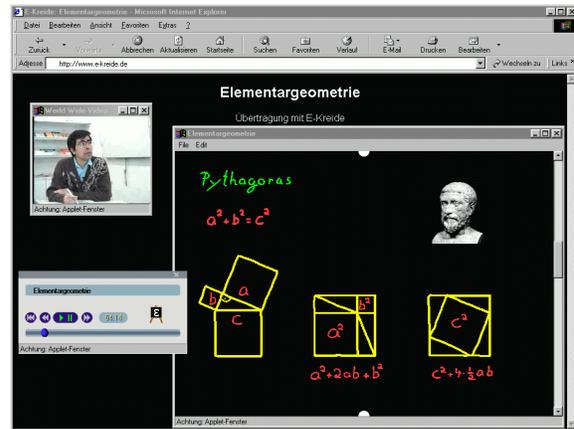


Figure 1. The electronic chalkboard.

tered with a structure-based editor with buttons, menus, etc. In such systems the user is forced to “*parse*” the expression mentally before it is entered. This step is not normally part of the process of writing equations on a paper or on a chalkboard.

### 1.1. The electronic chalkboard (E-chalk)

The electronic chalkboard (E-chalk) combines all the advantages of a traditional chalkboard with the functionality of a modern teleteaching tool [10]. The lecturer can write on the electronic board (via a pen device), integrating seamlessly pictures and interactive programs. E-chalk also includes algebraic formula manipulation, together with function plotting (See Fig. 1). Because of such capabilities, it is of interest to integrate in the E-chalk an automatic on-line handwritten formula recognition system, such that the lecturer can write directly on the electronic chalkboard the formulas to be manipulated without using the keyboard.

In this article we describe our first version of such a system. Section 2 gives an overview of support vector ma-

chines. The recognition of mathematical formulas is described in Sec. 3. Implementation details and experimental results are given in Sec. 4. In the last section we give some final comments about this work.

## 2. Support vector machines

Most of the pattern recognition methods are based on finding a classification function which minimizes the *empirical risk*, the error measure of on the given patterns. The theory developed by Vapnik and Chervonenkis gives upper bounds of the *structural risk*, the error of misclassification of the unseen patterns generated according to an unknown but fixed probability distribution, in terms of the empirical risk and the VC-dimension of the family of classification functions. The minimization of such bounds follows the principle of *Structural Risk Minimization* [13].

Following the mentioned principle for constructing a classification function  $f(x) = \text{sign}(w \cdot x + b)$  based on the patterns  $x_i \in \mathbb{R}^n$  with the corresponding labels  $y_i \in \{-1, 1\}$ ,  $i = 1, \dots, \ell$ , one has to find the solution  $w$  and  $b$  of the following optimization problem

$$\begin{aligned} & \text{minimize} && \frac{1}{2}\|w\| + C \sum_{i=1}^{\ell} \xi_i \\ & \text{subject to} && y_i(w \cdot x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0. \end{aligned} \quad (1)$$

Here the quantity  $1/2\|w\|$ , the *margin*, is related to the VC dimension and is maximized. The parameter  $C$  can be regarded as a regularization parameter. To save some numeric and implementation problems, the problem (1) is transformed using the technique of Lagrange multipliers into the following dual problem

$$\begin{aligned} & \text{maximize} && \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \\ & \text{subject to} && 0 \leq \alpha_i \leq C, \text{ and } \sum_{i=1}^{\ell} y_i \alpha_i = 0. \end{aligned} \quad (2)$$

An important characteristic of SVMs is that  $f$  can be expressed in terms of the training vectors  $x_i$  for which the solution  $\alpha_i$  of (2) are positive, called the *support vectors* (SVs), by setting  $w$  and  $b$  as follows:

$$w = \sum_{\alpha_i \in SV} y_i \alpha_i (x_i \cdot x), \quad b = \frac{1}{2}(w \cdot x_+ + x_-), \quad (3)$$

where  $x_+$  and  $x_-$  are two SVs which belong to the positive and negative class, respectively. The SVs are interpreted as the relevant patterns in the classification process. Frequently, the number of SVs is small with respect to the number of training data, this gives a compact representation and efficient implementation of the classifier. Other important characteristic of SVMs is that non linearity can be introduced by replacing the inner product in (2) by a *kernel function*  $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ , where  $\phi$  maps the patterns into a high (possibly infinity) dimensional inner

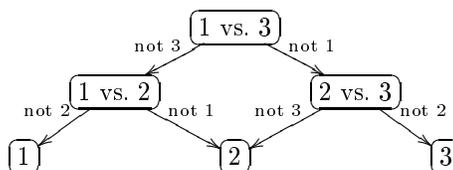


Figure 2. DDAG for three classes.

product space. Some of the best known (and used) kernel functions are:

$$K(x, z) = \exp(-\gamma\|x - z\|), \quad (4)$$

$$K(x, z) = (\gamma((x \cdot z) + 1))^d, \quad (5)$$

$$K(x, z) = \tanh(\gamma(x \cdot z) - \theta), \quad (6)$$

which are known as *radial basis functions (RBFs)*, *polynomial kernels* and *hyperbolic kernels* respectively.

### 2.1. Multi-class support vector machines

For multi-class SVMs one can use *Directed Acyclic Graphs SVM (DAGSVM)* [8]. The classification criterion is made via a rooted binary graph: in the  $k(k-1)/2$  internal nodes of the graph the classifiers are implemented and the  $k$  leaves indicate the class predicted label. The nodes are arranged in a triangle with the root node and the top and the leaves at the bottom, as showed in Fig. 2. Given a new example  $x$ , the classification starts with the classifier in the root node. A node is exited via the left edge if the label in the classification is positive, or the right edge if the label is negative. The next classification node is evaluated. One continues this procedure until a leaf is reached.

## 3. Formula recognition

The hierarchical structure of the expression is constructed by using a modification of the baseline structure analysis method developed by Zanibbi et al [14]. The idea is that mathematical notation can be described as a hierarchical structure of nested baselines. A baseline is a sequence of symbols which represent a horizontal arrangement of symbols in the expression. The symbols have pointers to other baselines, depending on the mathematical relation defined among them and their child baseline based on the symbol layout model. This model contains four symbols classes. *Limit* (sum, product or integral) *square root*, *non-scripted* (operators, open brackets and horizontal line) and *plain* (all other symbols). In the symbol layout model, the centroid of the symbol is also specified. It is calculated using the bounding box coordinates, which takes in account the following types of symbols: *central*, *ascender* or *descender*. Based on the symbols class and type, thresholds are computed, which serve as limits among the centroids to define

the mathematical relationships *up*, *down*, *right*, *superscript*, *subscript* and *subexpression* [5, 14].

The data structure which represents the hierarchical structure of symbols is called *Baseline Structure Tree (BST)*, and it is constructed as follows:

1. Sort the input list  $L$  of symbols by leftmost bounding box coordinate.
2. Calculate symbol class, symbols type and centroids for each symbol in  $L$ .
3. Find  $s_1$  the first dominant symbol in  $L$ . It is found by means of the special function  $s_1 = start(L)$ .
4. Find the rest of the symbols in the baseline by using the function  $s_2 = hor(s_1, L), \dots, s_3 = hor(s_1, L)$  until  $hor()$  returns *null*.  $hor(s, L)$  returns the symbol in  $L$  which is the next at the right of  $s$  and handles irregular layouts.
5. Find for each symbol in the dominant baseline  $D = (s_1, s_2, \dots, s_n)$  the symbols in  $L$  which satisfy the relations *up*, *down*, *left*, *right*, *superscript*, *subscript* and *subexpression*, and add them to the corresponding child lists.
6. Apply recursively steps 1-5 to each of the non-empty child lists of the elements in  $D$ .

The complete description of the functions  $start()$  and  $hor()$  can be found in [14]. After the layout step is completed, it is possible to apply a syntax and semantic steps. The syntax step reorganizes the BST in tokens which consist of many symbols or or many baselines, for example, numbers of fractions. The semantic step reorganizes the the tree according to operator precedence, associativity, etc.

## 4. Implementation

### 4.1. Preprocessing

The data considered are strokes and symbols. A symbol is sequence of strokes and a stroke is a sequence of points stored as coordinate pairs and time information, they are obtained equally spaced in time from a graphic tablet. The following procedures were used to preprocess our on-line data [6, 3, 12]:

**Reversing.** At this step we classify each stroke in the symbol as closed, horizontal, vertical or diagonal, using the ratios  $R_x = |x_l - x_f|/D$  and  $R_y = |y_l - y_f|/D$ , where  $(x_f, y_f)$ ,  $(x_l, y_l)$  are its first and last point and  $D$  the length of diagonal of its bounding box, respectively. We select a threshold  $\delta \in [0, 1]$  and we say that the stroke is closed if  $R_x < \delta$  and  $R_y < \delta$ ; horizontal if  $R_x \geq \delta$  and  $R_y < \delta$ ;

vertical if  $R_x < \delta$  and  $R_y \geq \delta$ ; diagonal if  $R_x \geq \delta$  and  $R_y \geq \delta$ . Horizontal strokes are reversed if  $x_l < x_f$  and vertical and diagonal ones if  $y_l < y_f$ .

**Ordering.** Once the strokes are reversed, they are ordered respect to the angle between the upper segment of the bounding box of the symbol and the segment limited by the upper left corner of the bounding box and the last point of the stroke.

**Filtering.** In this step repeated points of the stroke are eliminated.

**Data reduction.** We take a fixed number of points of the stroke in the following way: we start with the segment formed by first and last point of the stroke, this segment will be the basis the triangle formed with other point of the stroke. Among all the points we select the one which is maximal respect to the height of the triangle (if the point of the basis coincide we use the distance instead the height). We repeat the procedure to the two the sub-strokes formed by “cutting” the original stroke at the maximal point. The procedure finishes when a desired number of points is reached.

**Smoothing.** We average the point coordinates with its previous and next neighbors using the window  $(0.25, 0.5, 0.25)$ .

**Equidistant resampling.** The sequence of captured points is replaced with a sequence of points having the same distance.

**Scaling.** Symbols are scaled and translated such that their bounding box fits the square  $[0, 1] \times [0, 1]$  and the center of the bounding box and the square coincide.

**Segmentation.** A recently drawn stroke belongs to a symbol, if its mathematical dilation with a circle of radius  $\alpha r$  (where  $r$  is the current draw radius and  $\alpha > 1$ ) intersects some of the strokes of the symbol. It is possible that the distance of first point of the new stroke and the last point of some stroke in the symbol is lower than  $\alpha r$ , if that is the case, both strokes are concatenated.

### 4.2. Feature vector

Given a stroke  $s = (p_1, \dots, p_n)$  of length  $\ell$ , we use the following *local features* to construct the input vector for the classifier:

**Coordinates**  $(x_i, y_i)$  of  $p_i$ .

**Turning angle.** The values  $\sin(\theta_i)$  and  $\cos(\theta_i)$  of the turning angle  $\theta_i$  at the point  $p_i$ . See Fig. 3.

**The change of the turning angle.** The values  $\sin(\theta_{i+1} - \theta_i)$  and  $\cos(\theta_{i+1} - \theta_i)$ , where  $\theta_i$  and  $\theta_{i+1}$  represent consecutive turning angles.

**Length position** of  $p_i$  is defined as  $\ell_i = \sum_{k=1}^{i-1} d(p_k, p_{k+1})/\ell$ .

The following global features of  $s$  are also used:

**Center of gravity**  $(\sum_{i=1}^n x_i/n, \sum_{i=1}^n y_i/n)$ .

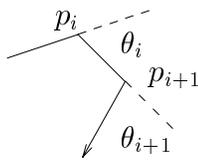


Figure 3. The features of the stroke.

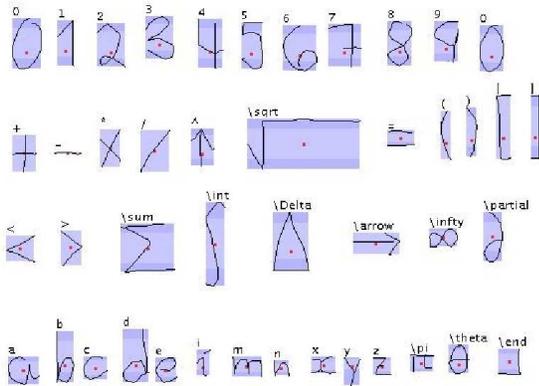


Figure 4. The recognized symbols.

**Length.**

**Relative Length**  $\ell_r = \ell/d$ , where  $d$  is the distant between the first and last point of  $s$ .

**Accumulated angle**  $\theta_a = \sum_{i=1}^n \theta_i / 2\pi$ .

**4.3. Experimental results**

Our data was obtained from a Wacom Graphire 2 tablet connected to a Sony Vaio PCG-FX502 notebook. The symbols were written by a volunteer and each class contains 50 symbols (see Fig. 4). The number of symbols was artificially increased 10 times by means of transformations related to the tangent distance [11]. We select randomly 50% of the data for training, 25% for testing and 25% for validation. We assume that the symbols are formed at most of tree strokes, and tree classifiers were used to classify them, depending of the number of strokes. They were preprocessed as described in Sec. 4.1. The number of points of each stroke was fixed to 16. The features used are the described ones in Sec. 4.2, and if the symbol has more than one stroke, for each one we construct their corresponding feature vector and then concatenate them to obtain the final one.

Our system uses DAG-SVM as multi-class SVM-classifier. The modification one of the *Sequential Minimal Optimization (SMO)* algorithm was used to train the classification nodes [7, 4]. We use the RBF kernel for all the classification nodes of the DAG-SVM. Experiments suggested the value  $\gamma = 0.001$ . For comparison we trained a neu-

Classifier	SVs	Train	Test	Val
SVM1	29.65%	0.31%	0.93%	0.76%
ANN1	-	0.72%	1.17%	1.27%
ANN1*	-	0.98%	1.0%	1.06%
SVM2	36.73%	0.00%	0.62%	0.70%
ANN2	-	0.69%	1.31%	2.09%
ANN2*	-	1.26%	1.78%	2.08%
SVM3	39.41%	0.00%	0.00%	1.17%
ANN3	-	0.00%	1.17%	1.17%
ANN3*	-	0.00%	1.17%	1.17%

Table 1. Classification rates.

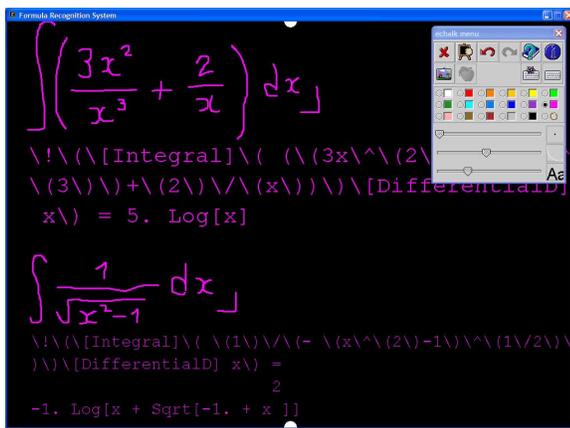
Classifier	SVs	Train	Test	Val
SVM1	45.45%	2.43%	2.86%	3.07%
SVM2	46.64%	1.84%	2.76%	2.27%
SVM3	<b>27.85%</b>	1.00%	1.41%	1.31%
SVM4	29.44%	0.29%	0.86%	1.07%
SVM5	29.23%	<b>0.17%</b>	1.31%	<b>0.76%</b>
SVM6	29.13%	0.24%	0.79%	1.07%
SVM7	29.70%	0.31%	<b>0.75%</b>	0.82%
SVM8	29.65%	0.31%	0.93%	<b>0.76%</b>

Table 2. Classification rates with feature integration.

ral network with the RPROP algorithm with the standard sigmoid and its standard parameter values [9]. The training was finished by using an early stopping criterion with the validation set. The number of hidden units for each one were the same of the dimension of the feature vector.

Table 1 resumes the results obtained for each classifier. The the number indicates the correspondence of the classifier and the number of strokes in the symbols. We trained each ANN five times and selected the best one respect to the error rate on the validation set. We used the support vectors found in the SVM training to train other ANN (marked with an asterisk in the table). Because of size of the support vectors we found that the training time is faster and classification rate improve. When we add more features to the vector, we found some improvement in the classification rates and a reduction in the number of SVs, which is a good result because it increases the speed of the classification. The results for the one-stroke classifiers are showed in Table 2. The features were added in the same order of its description in Sec. 4.2.

The recognition stage in E-chalk starts when the lecturer set the system in the recognition mode by selecting a reserved color to draw the strokes. In this way, when a set of



**Figure 5. Some formulas recognized by the system.**

strokes is grouped into symbols, they are preprocessed and classified. If the classifier gives as output the `\end` symbol, the system analyzes the complete list of recognized symbols and the BST of the expression is constructed. It is translated into a Mathematica expression and then evaluated. Figure 5 shows some recognized formulas in the echalk system.

## 5. Conclusions

In this article we presented a formula recognition system based on support vector classification and baseline structure analysis. Our experiments indicate that SVMs are very suited for on-line handwritten recognition.

Our experiments with ANN also show that SVs could be used as filtered training data which improves training speed and generalization ability of classifiers.

There is a trade-off between the number of features (the dimension of the input vector) and the number of SVs. More research is needed to find a suitable representation of the strokes and a good selection of the features.

At present, E-chalk can convert the lectures into PDF format, and they appear on the paper exactly as written by the lecturer in the E-chalk. Our objective is that in the future the lectures can be converted also into some electronic format like  $\LaTeX$ . Since the winter semester 2002, E-chalk has been used in seminars and lectures at the Free University of Berlin and the Technical University of Berlin, principally in the departments of mathematics and computer science. All the lectures stored at that universities will be our start point of research about the conversion of the documents onto different and useful formats.

## Acknowledgments

Ernesto Tapia thanks the Mexican National Council for Science and Technology (CONACyT) for its support during his Ph.D. studies via the credit-scholarship number 154901.

## References

- [1] R. H. Anderson. *Syntax-Directed Recognition of Hand-Printed Two dimensional Equations*. PhD thesis, Harvard University, Cambridge MA., January 1968.
- [2] B. P. Berman and R. J. Fateman. Optical Character Recognition for Typeset Mathematics. In *International Symposium on Symbolic and Algebraic Computation*, pages 348–353, 1994.
- [3] W. Guerfali and R. Plamondon. Normalizing and Restoring On-Line Handwriting. *Pattern Recognition*, 26(3):419–431, 1993.
- [4] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO Algorithm for SVM Classifier Design. Technical report, Dept of CSA, IISc, Bangalore, India, 1999.
- [5] H. J. Lee and J. S. Wang. Design of a Mathematical Expression Understanding System. In *In Proceedings of IC-DAR'95*, pages 1084–1087, 1995.
- [6] N. Matsakis. Recognition of Handwritten Mathematical Expressions. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1999.
- [7] J. C. Platt. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.
- [8] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large Margin DAGs for Multiclass Classification. *Advances in Neural Information Processing Systems*, 12:547–553, 2000.
- [9] M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA, 1993.
- [10] R. Rojas, G. Friedland, L. Knipping, and W. L. Raffel. Elektronische Kreide: Eine Java-Multimedia-Tafel für den Präsenz- und Fernunterricht. Technical Report B-00-17, FU Berlin, Institut für Informatik, October 2000.
- [11] H. Schwenk and M. Milgram. Constraint Tangent Distance for On-Line Character Recognition. In *International Conference on Pattern Recognition*, pages D–515–519, August 1996.
- [12] 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*, 12(8):787–808, August 1990.
- [13] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.
- [14] R. Zanibbi, D. Blostein, and J. Cordy. Recognizing Mathematical Expressions Using Tree Transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11), November 2002.