Chapter 7

Conclusion

There exists a great variety of methods for the recognition of on-line handwritten mathematical expressions. Most of them share a common approach: recognition of mathematical expressions is divided into two subproblems, namely recognition of characters and structural analysis of the mathematical expression. However, authors concentrate only on one of the subproblems or on the development of a user interface, offering only a partial solution to the whole problem.

The objective of this thesis was to develop a system for the recognition of on-line handwritten mathematical expressions. Instead of giving partial solutions to the problem, we offered an integrated solution in a real environment. Our system handles on-line handwritten mathematical expressions with a minimum of restrictions: the expression can be written using the usual mathematical conventions.

The main contributions of this work are the following:

Classification of On-Line Symbols

Our experiments indicate that Support-Vector Machines are best suited for the recognition of isolated on-line handwritten symbols.

We propose a system for the recognition of isolated on-line handwritten characters which is based on support vector classification. We also propose a suitable representation for strokes and symbols which is used to improve the classification rates of the classifier.

We realize a variety of experiments using user-dependent and user-independent data. In addition to Support-Vector Machines, in our experiments we used other popular classification techniques: nearest neighbors, naive Bayes, classification trees, and artificial neural networks. Among all these classifiers, the one we developed

achieved the best classification rates with the data. This could be accomplished by extensive preprocessing of the data and by parameter selection for the support vector classification. From experiments, we can also conclude that our classifier was superior, in terms of classification rates to others classifiers found in the literature. Our experiments suggest that the Support-Vector approach optimizes the trade-off between training time and classification rates.

Structural Analysis of Mathematical Expressions

We propose a new structural analysis method for the recognition of on-line handwritten mathematical expressions based on a minimum spanning tree construction and symbol dominance. Our method addresses important layout problems frequently encountered in on-line handwritten formula-recognition systems. Our method also aims to handle input as naturally as possible, i.e. using the usual mathematical conventions, without restrictions in the order the symbols are written.

We introduce a technique to locate fraction lines in expressions to overcome horizontal irregularities. Irregular horizontal layouts are normally caused by neighboring symbols of fractions. Locating fraction lines also helps to find the correct association of superindices taking the form $b^{\frac{1}{2}}$ when the arguments of the fraction overlaps the horizontal region of the base. Our method also addresses the problem of argument association to sum-like operators. These arguments are symbols located above and below the sum, integral, product operators, etc.

Our method handles symbols with non-standard layout, like $\prod_{*}^{*}$, as well as tabular layouts, e.g. matrices. To our knowledge, solutions to handle these important layout structures cannot be found in the literature. This novel solution can be easily extended to recognize other important mathematical layouts, for example the ones defined by the LaTeX commands `\overbrace` and `\begin{cases}`–`\end{cases}`, among others.

A User Interface for the Recognition of Mathematical Expressions.

We developed a prototype system for editing and recognition of on-line handwritten mathematical expressions, which has the following characteristics:

- **Near-natural handwriting recognition.** Since the editor uses our recognition engine for mathematical expressions, the user writes following the usual mathematical conventions.

- **User-Friendly input interface.** Our editor integrates more editing and visualization capabilities than other free programs aimed to the same purpose, providing a user-friendly user interface.

- **Use of gestures.** The editor allows manipulation and correction of expressions via gestures. This new tool minimizes the use of menus, buttons and other GUI components, although they are also available to the user.

- **Manipulation of output.** Our editor is unique in the way it handles the output it generates. Its output is suitable not only for word processing but also for symbolic computation. The output of the editor generates a LaTeX expression which is used for the visualization of the recognition results. If the output is interpreted as an algebraic expression and Mathematica is available, the expression can be manipulated algebraically as well as used for function plotting.

Integration with E-Chalk

We integrated our recognition engine with the E-Chalk System.

The recognition stage in E-chalk starts when the lecturer puts the system in recognition mode by selecting a reserved color to draw the strokes. In this way, when a set of strokes is grouped into symbols, they are preprocessed and classified. If the classifier gives the `\end` symbol as output, the system analyzes the complete list of recognized symbols and the expression is constructed. It is translated into a Mathematica or Maple expression and then evaluated. The recognition capabilities in the E-chalk are similar to the ones of our editor, except for editing via GUI components.

The user engine can be also used by animation applets, which can be constructed using the animation libraries of the E-Chalk Team. Developers can use the recognition engine in their own animations. Examples of the application of our recognition engine to other scenarios are the E-Chalk Applets for algorithm visualization by Esponda [25] and the applets for simulation of biological neurons by Krupina [51]. Our recognition engine can potentially be used by any E-Chalk developer.