A Survey on Recognition of On-Line Handwritten Mathematical Notation

Ernesto Tapia, Raúl Rojas

B-07-01
January 2007

A Survey on Recognition of On-Line Handwritten Mathematical Notation[*]

Ernesto Tapia and Raúl Rojas

Technical Report B-07-01

Freie Universität Berlin, Institut für Informatik
Takustr. 9, 14195 Berlin, Germany
tapia,rojas@inf.fu-berlin.de

January 26, 2007

Abstract

This report describes recent advances in the area of the recognition of on-line handwritten mathematical notation. We describe architectures, symbol classification methods, and techniques for the structural analysis of mathematical expressions. We also survey applications specialized for mathematical notation.

1 Introduction

Interest on developing *pen-computing* applications has been growing steadily during the last years, due to, among other factors, the introduction of devices such as *personal digital assistants* (PDAs) or Tablet PCs. The main characteristic of such devices is that they use the stylus as input tool, being a “natural” substitute for keyboards ans mouse, see Fig. 1.

The data generated by users writing with a stylus on an electronic device is known as *digital ink*, and the process of writing is called *on-line handwriting*. The minimal unit which forms digital ink are *strokes*, which are sequences of points generated between *pen-down* and *pen-up* events, at regular time intervals.

The main objective in pen-computing is not only handling digital ink “as is” but also its conversion into another data structure that can be automatically processed by computers. The elemental data structures are alphanumeric characters, since PDAs were originally developed to be personal organizers with a calendar or an address book. Most PDAs use the stroke alphabet *Graffiti* [39] as the standard method for *symbol recognition*. The alphabet consists of a set of pre-defined stroke combinations that allow the recognition of letters and numbers, see Fig. 2.

Figure 1: Pen-based devices.

The development of pen-based systems and applications increased when Microsoft released the Tablet PC version of the Windows operating system. It provides a Software Development Kit (SDK) for processing, storing, and recognizing digital ink. These libraries are specialized towards the recognition of letters and words of western languages and East Asian languages, allowing a more natural writing when compared with Graffiti. One of its most attractive characteristics is that the recognizers are integrated natively in the operating system.

Even though pen-based mathematics has a potential application in scientific document processing, human-computer interaction and mathematical knowledge management, the methods mentioned above are only capable to recognize plain text, but no complex handwritten mathematics.

[*]A revised and extended version of this report is under preparation for submission to a journal. We welcome any comments and suggestions to improve this work.

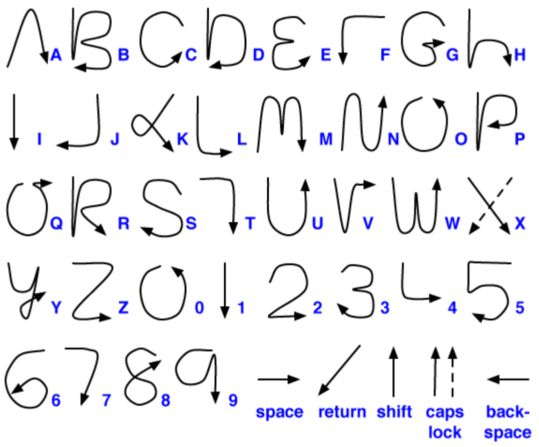
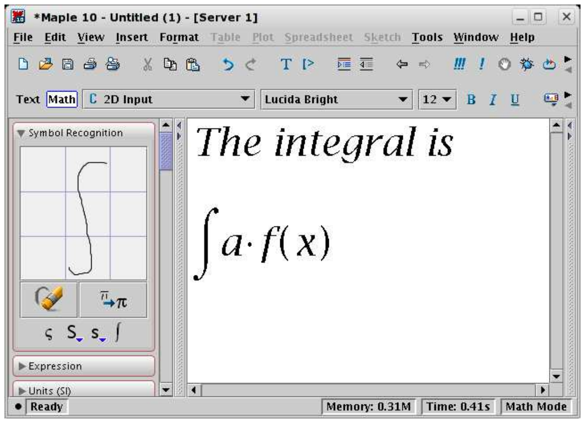


Figure 2: The Graffiti alphabet.


*Maple 10 - Untitled (1) - [Server 1]
File Edit View Insert Format Table Plot Spreadsheet Sketch Tools Window Help
Text Math 2D Input Lucida Bright 12
Symbol Recognition
Expression
Units (SI)
Ready
Memory: 0.31M
Time: 0.41s
Math Mode

The integral is

$$\int a \cdot f(x)$$

Figure 3: Maple's recognition system.

Fortunately, pen-based mathematics has gained attention by the computer science community over the last years. An example of this increasing interest is the development of the symbol recognition tool for the computer algebra system Maple 10. Its pen-based interface allows the recognition of isolated mathematical symbols that are not available in the keyboard, see Fig. 3.

This report describes the advances in the area of pen-based mathematics and on-line mathematical recognition that have during recent years. It is organized as follows. Section 2 describes the main characteristics which distinguishes the recognition of on-line mathematics, as well as architectures for the recognition of mathematical notation. Section 3 describes briefly the symbol recognition task. We give an overview of the most important techniques for structural analysis in Sec. 4. Recent work on user interfaces for on-line handwriting is described in Sec. 5.

2 Recognizing mathematical notation

In comparison with plain handwritten text, recognition of on-line handwritten mathematics requires more elaborated symbol classifiers because mathematical notation uses a big set of characters including Latin letters, Greek letters and mathematical symbols. Other difficulty relies also on the analysis of the intrinsic two-dimensional layout encountered in formulas, arrays of symbols and diagrams, which requires a different treatment in comparison with handwritten plain text. These and other relevant characteristics of mathematical notation are enumerated below, as suggested by Blostein [6], and Chan and Yeung [9].

2.1 Characteristics of Mathematical Notation

2.1.1 Definition of mathematical notation

Mathematical notation is a specialized *two-dimensional notation* which helps to communicate mathematics and to visualize concepts and ideas. Although mathematical notation is a language used in many areas of science, no formal definition in terms of syntax and semantics as a two-dimensional language exists. Actually, this notation is not completely standardized and many *dialects* are used by scientists. Some authors try to describe mathematical notation for solving problems of typesetting [23] and for automatic processing of mathematical notation [29].

2.1.2 Scope of recognition systems

We have to consider which kind of mathematical expressions we will recognize. Researchers normally restrict themselves in the recognition of a subset of mathematical notation, for example notation for a particular area of mathematics, notation needed only for high-school mathematics, etc. In addition, such restrictions also depend on the purpose of the recognition system: recognition of typeset scientific text, an input model for computer algebra systems, etc.

2.1.3 Grouping of basic symbols

When we read an expression, we group single symbols to construct more complex objects, which plays an important role in the interpretation of formulas. For example, the digits '3', '8', and '1' have their own meaning as such, but if they are of the same size and lie in the same line, they can represent the integer value '381'. By varying size and location they can also represent '3^{81}' or '3^81'. Similarly, we can group some letters to represent function names, like 'log' or 'sin'.

2.1.4 Explicit and implicit operators

Variation of size and location can also represent grouping criteria through mathematical relations between symbols. In mathematical notation, we find *explicit* and *implicit* relations between symbols. Subscripts, superscripts and tabular structures are described only by the location of operands; in contrast, addition, subtraction, and division use explicit operator symbols. For example, in the expression '$a + b$' the explicit relation 'addition' between 'a' and 'b' is given by the symbol '+'. An example of an implicit relation would be in the expression 'x^2'. In this case the expression represents the relation 'pow' between 'x' and '2', a variation in the location results in a very different relation, for example 'x_2', representing the relation 'subscript'.

2.1.5 Ambiguity in the role of symbols

We illustrate this characteristic by giving some examples. The horizontal bar can represent the minus sign and also the fraction bar. A dot can represent multiplication or the decimal point. The Greek letter sigma 'Σ' can also represent the operator sum. When grouping the symbols 'dy', they can represent the product of 'd' and 'y' as in '$cx + dy$' or the differential operator when used in '$\int \cos(y)dy$'.

2.1.6 Irregular writing

Irregular handwriting aggravates ambiguities described before and makes it harder to group symbols and to distinguish relations among them. It results in layout problems affecting the recognition of the whole expression. A cause of this is due to unexperienced users, because they normally take excessive freedom with the location and alignment of handwritten symbols. Other kinds of irregular writing arise during the correction, deletion, and insertion of symbols. For example, suppose an expression was entered by the user and he decided to write some super-indices or sub-indices, but there was not enough space available for this correction. Something like this could generate a very complex expression, which could not be recognized even by humans.

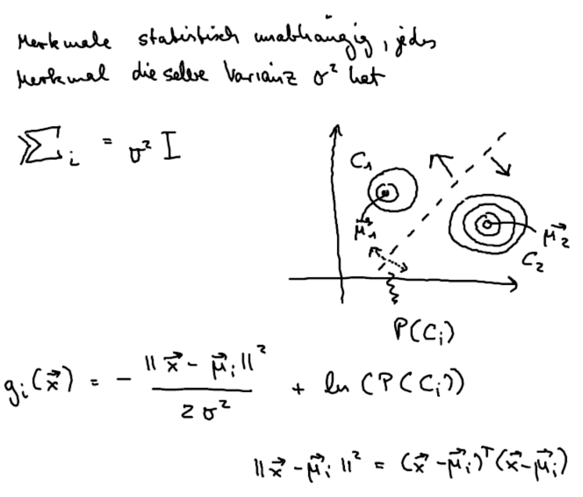

Merkmale statistisch unabhängig, jedes
Merkmal die selbe Varianz σ^2 hat
$\Sigma_i = \sigma^2 I$
C_1
$\vec{\mu}_1$
$\vec{\mu}_2$
C_2
$P(C_i)$
$g_i(\vec{x}) = -\frac{\|\vec{x} - \vec{\mu}_i\|^2}{2\sigma^2} + \ln(P(C_i))$
$\|\vec{x} - \vec{\mu}_i\|^2 = (\vec{x} - \vec{\mu}_i)^T(\vec{x} - \vec{\mu}_i)$

Figure 5: A typical on-line document.

2.2 Architectures and frameworks

This sections describes three architectures which are important for the development of pen-based mathematics. From our point of view, they offer solutions for the same problem which can be considered as mutually complementary. The architectures consider three core building blocks for pen-based mathematical system: *recognition*, *development*, and *user-system interaction.*

2.2.1 Recognition

Most of the working systems and prototype programs for the recognition of pen-mathematics follow, in essence, the steps proposed by Lee and Wang [36]. Although his system is specialized for *off-line* mathematical equations, i.e. expressions in scanned documents, almost all of the procedures can be used as well for the recognition of pen-based mathematics. We have only to substitute the "Optical Scanning" step for a "Ink Input" step in the flow diagram shown in Fig. 4.

The recognition of mathematical notation begins by considering how the data is presented. Optical scanning or ink input follows *digitalization* step that transforms the given expression into a static (off-line) or dynamic (on-line) representation. In the *segmentation* step, mathematical expressions are extracted and isolated from text lines [36, 26]. When dealing with on-line data, it is normally supposed that the data only contain mathematical expressions. However, it is not always the case, because modern on-line documents are also formed by a mix of plain text, graphics and mathematics, see Fig. 5

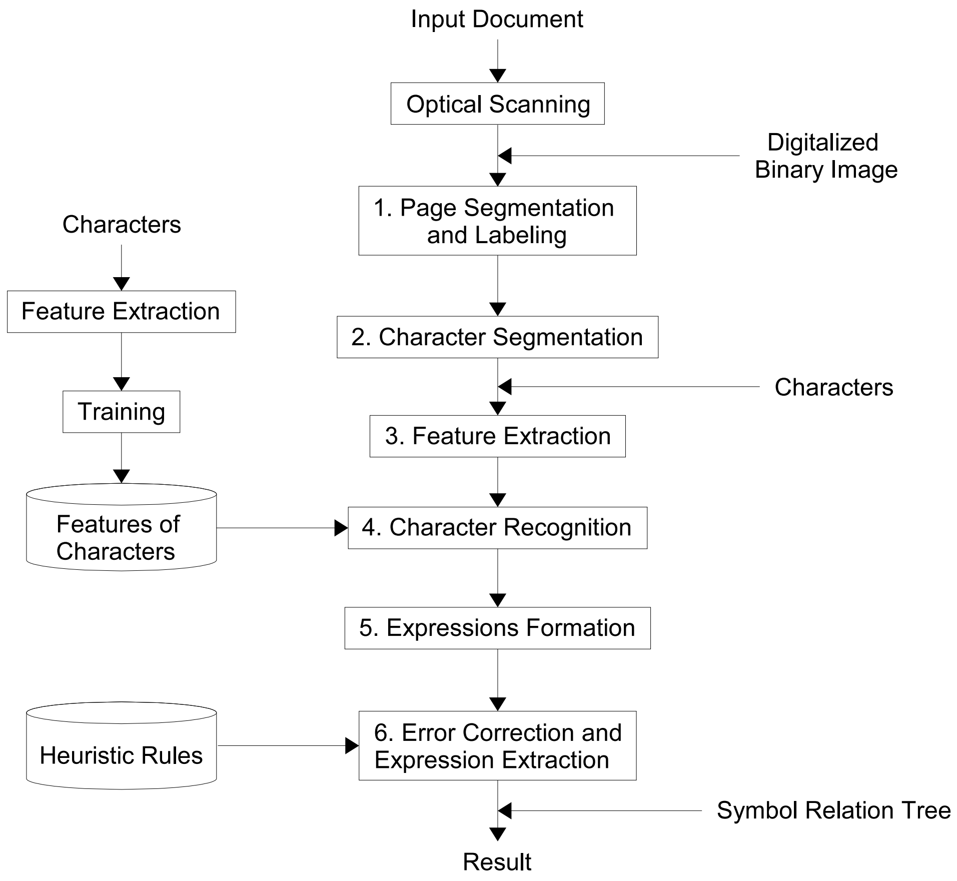


Figure 4: Digram flow for the recognition of mathematical expressions in off-line documents.

Once the document is segmented, the next step is *symbol recognition*. In this step, the label of symbols is established by means of a classifier. During the expression formation step, the structure of the recognized symbols is analyzed to form a hierarchical structure that represents a mathematical expression.

In the last step, the internal hierarchical structure is processed and interpreted to obtain a final result. This result can be a character string which represents the expression in LaTeX, another structure used by a computer algebra system, or an image representing the plot of a function given by the expression, among others.

Their procedural framework can be divided in three main modules:

1. Document Segmentation.
2. Symbol Recognition.
3. Structural Analysis.

Differences between recognition systems found in the literature are generated by variations and improvements of these modules.

2.2.2 Development

Most of the systems created for the recognition of mathematical notation have for objective the conversion of the handwritten expression into a data structure, which can be used mainly by a text editor or for some computer algebra system. Ideally, such a general system should be used for a wide range of applications such as the elaboration of mathematical databases, text processing, symbolic computation, elaboration of mathematical diagrams, etc. It should also be easily extended and provide a practical GUI.

Smirnova and Watt [54] propose an architectural framework for pen-based mathematics from the perspective of software engineering. Their target deployment is for document processing and mathematical computing. Their objective is to define a platform-independent pen-based framework for mathematical

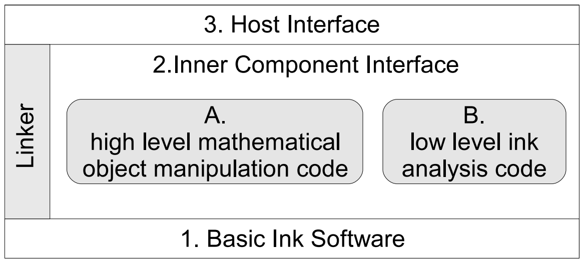

3. Host Interface
2.Inner Component Interface
Linker
A.
high level mathematical object manipulation code
B.
low level ink analysis code
1. Basic Ink Software

Figure 6: Framework components for mathematical notation.

entry, edition and calculation. These objectives lead them to define two portability criteria for the framework. *Longitudinal portability* is the software life and usability given a platform, and *lateral portability* is the software life across platforms.

They refer to the Microsoft's Tablet PC SDK as an example for longitudinal portability: this API evolve with the time having a direct impact for the development of the framework. Thus, they suggest two main components of their framework that have to remain the same along longitudinal evolution: a) high level manipulation of mathematical objects and b) low level analysis of digital ink, see Fig. 6.

An abstract layer with platform-specific wrappers allows the lateral portability of the software. The wrappers access natively resources and recognition engines already contained in the platform (Palm Graffiti or Microsoft Tablet PC, for example). The components of their framework which shall vary, depending of the platform, are 1) basic software to collect digital ink, 2) low level processing module to interconnect the components (a) and (b) mentioned above, and 3) use of the framework in some hosting application. Figure 7 illustrates the complete architecture that includes recognition modules.

2.2.3 User-System Interaction

LaViola [33] proposed a new paradigm for gestural interaction in pen-based mathematics. He consider pen-based mathematical documents to be constituted from mathematical expressions and supporting diagrams. His work concentrates in the development of a gestural user interface, which allows a seamless interaction between these document constituting parts by generating actions from gestures. Figure 8 shows the architecture he proposed.

The most relevant from this architecture are the *User Interface* and *Sketch Preparation* modules. The User Interface Module has a Gesture analyzer component which decides whether the last written strokes should be interpreted as a gesture. If it is decided so, the corresponding action generates a connection with others modules. In the other case, the ink data is stored by the *Data Entry* component. The Correction UI allows the user to correct the recognized mathematical expression at the symbol level or at the parsing level.

The Sketch Preparation module has an *Association Inference* module, which associate label drawing elements with written mathematical expressions. These labels describe some property of supporting diagrams, such as its location expressed as Cartesian coordinates that are modifiable by means of the written expressions. After the analysis of the expression by the *Math Code Generation* module, the diagram can be animated. The Drawing Rectification module works in a similar way as association, but it is used to correct diagram parameters such as angles, lengths and location. As an example for rectification, imagine that a user draws a triangle to illustrate the Pytagora's Theorem, whose cathetus are not orthogonal. This can be corrected by labeling the angle with a variable a to the angle, writing a formula $a = 90$ and using the corresponding gesture which activates the rectification module to correct the drawing. The Dimension analysis component us used to calculate local coordinates systems of suporting diagrams, which are useful to generate animations in the ink document.

3 Symbol Recognition

3.1 Segmentation

Segmentation is the process of decomposing, grouping, or isolating the data into classifiable units which represent single symbols or words. In the case of online documents, the segmentation problem consists of properly classifying strokes as text, graphics.

An example of a simple heuristic-based segmentation is the method used by Mandler [40]. He associates the input strokes with a predetermined thickness by dilating them with a rectangle. With this criterion, two strokes belong to the same symbol if their dilated versions intersect.

Sometimes, segmentation methods use the information given by the classifier. Winkler et al. [30, 65] generate a symbol hypothesis net from the given strokes. The hypothesis net generates all possible combinations of strokes in order to handle ambiguity during symbol grouping. The different combinations of strokes in the net, which represent potential symbols, are classified using a Hidden Markov Model (HMM). The probabilistic output of the classi-

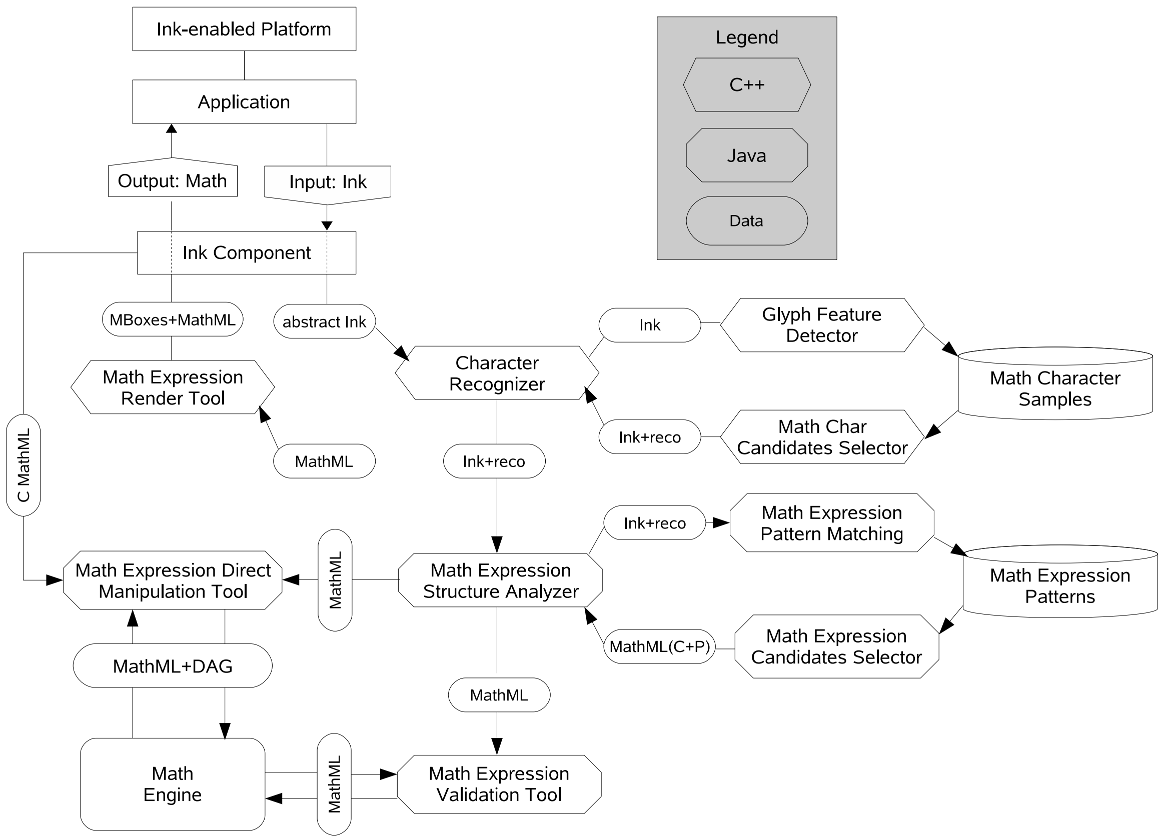


Figure 7: The complete architecture for pen-based mathematics.

fier serves to associate a likelihood value to each combination of strokes, and the one with the highest likelihood value is selected as the final segmentation result. This method is sensitive to the order the strokes are written. If they are not written from left to right and upside-down, segmentation and classification deteriorate dramatically. Examples of classifier-based segmentation are used by some of the systems we describe in Sect. 5.

Shilman et al. [51] present a method for the segmentation of digital ink based on a neighborhood graph and symbol classification. Strokes are regarded as nodes of a graph, where stroke are connected with an edge when they are "close" to each other. Once the graph is constructed, it is segmented into its connected components with a number of nodes up to a previously selected number k. Each connected component is regarded as an isolated symbol, which is classified using a previously trained model. A negative log likelihood cost is assigned to every partition based on the classification results. Finally, dynamic programming is used to efficiently find the optimal stroke partition. The main advantage of this method is that it is insensible to stroke order.

Ao et al. [2] present a framework based also in a neighborhood graph to find structures in digital ink documents. Their method is aimed to find baselines structures and to segment the document into textual and graphical parts. The main steps in their method are patches creation, baseline extraction, separation into text and graph, and word segmentation. The patches are the node subsets which are obtained by eliminating long edges from the neighborhood graph. They are grouped into text baselines my maximizing their *line strength*. The line strength is based on edge lengths (closely positioned), angle differences (linearly positioned) and sizes of neighboring blocks (comparable size). The classification of patches as text of graph is based in the supposition that graphical parts do not present a "reasonable" layout structure. The structure can be quantified by considering width, height, stroke count, and density features on the patch and their neighboring patches. This features are used to train a binary classifier which serves as model for the text/graph segmentation. Similar methods to find text baselines and to classify digital ink as text or graph can be found in [12, 25, 52, 5, 67]

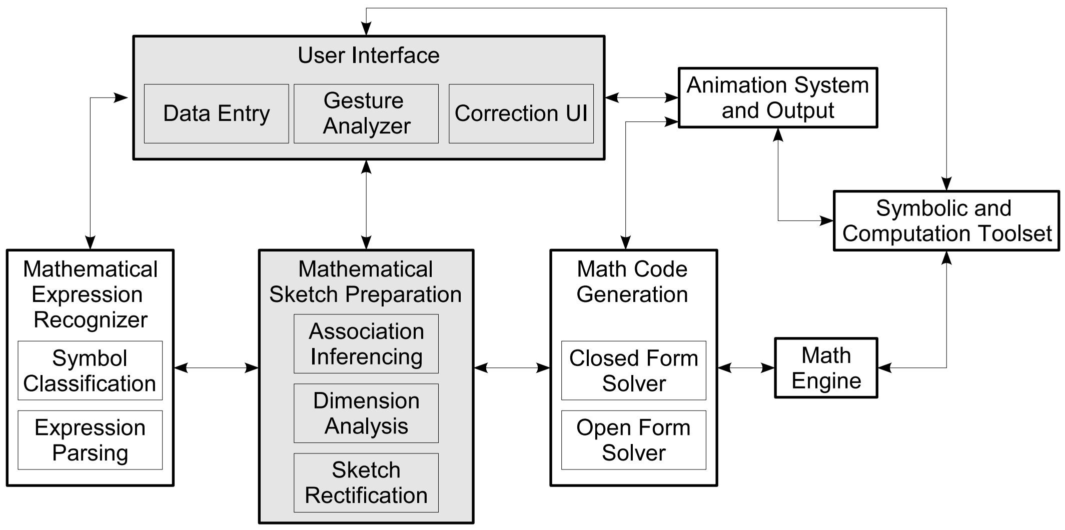


Figure 8: Architecture for mathematical sketching.

3.2 Preprocessing

Preprocessing is necessary to eliminate noise, to reduce the amount of information, and to normalize handwriting [59, 19].

An example of preprocessing to eliminate noise is simply the application of a Gaussian filter to a handwritten stroke. In this case, processed points in the stroke are the result of a weighted average of itself and its neighbors. Dehooking, point clustering, filtering, and stroke connection are useful preprocessing procedures to eliminate unnecessary information from the stroke. Examples of normalization for on-line data are scaling, slant correction, and equidistant resampling.

3.3 Feature Extraction

Feature extraction means deriving measures and characteristics from the raw data which are useful in making predictions. It is common that feature extraction methods are based on invariance, reconstruction, and expected distortions.

Features may be *local* or *global*. The source of information of on-line data are points and strokes. When we talk about local features, we refer to characteristics of a specific point in strokes derived from its neighboring points. Global features normally refer to topological (morphological) characteristics of the stroke based on the trajectory it describes.

The first local feature we consider are the coordinates of a point. They serve, for example, to derive important local characteristics, like curvature or direction. Points having a high curvature are important to decompose strokes in static elements. This decomposition is a structural description of the underlying shape of strokes [19].

Global features normally serve to obtain a categorical (not numerical) classification of strokes. For example, the ratio of the distance between a stroke's end points and its length is a global feature, which serves to determine if the stroke is closed or open, when comparing this ratio against a predetermined threshold. Similarly, when considering a symbol formed by two strokes, we can say that they cross or does not cross, or only touch. This kind of information serves to construct a decision tree which reduces the given data into a small set used in a later analysis using more complex features [59].

3.4 Symbol Classification

Symbol classification means the use of a certain method to obtain the symbol's label. Actually, we can potentially use any of the great number of existing classification methods to such purpose. In this section we review some of such methods which were specially developed for symbol classification. Other interesting results for classification of isolated symbols is [46].

3.4.1 Similarity Measures.

Similarity measures for strokes allow the use of distance-based classifiers. Generally, unknown symbols are matched against stored references (prototypes), and the label corresponding to the unknown symbol is the same as that of the symbol which best matches it. Similarity measures are normally defined at stroke (curve) level.

Tappert [58] uses *elastic matching* as a similarity measure. The procedure to calculate it is based on the alignment of points which constitute the stroke using the dynamic time warping (DTW) algorithm. The optimal matching corresponds to the minimum sum of matching costs, which are normally the distance between strokes' points obtained by dynamic programming. This algorithm allows the comparison of strokes having a different number of points. In this case, the feature-extraction step can be skipped.

Li and Yeung [37] also use time warping to calculate similarities between strokes. Their method converts strokes into a character string. The string describes the stroke as a sequence of eight directions. They find the distance between two string sequences by calculating costs of three string transformations, namely compression, expansion, and substitution.

Schwenk [50] proposes a similarity measure based on constrained tangent distance. The distance is locally invariant under the following local transformations: translation, scale, axis deformation, rotation, diagonal deformation, and slope transformation. This method tries to incorporate knowledge about geometrical variations of handwriting.

Other similarity measures for curves which can be useful for on-line symbol classification can be found in [61].

3.4.2 Structural Methods.

Structural methods are based on the assumption that handwriting is made up of some elementary or primitive shapes, also known as allographs. They describe the morphological characteristics of strokes. For example the symbol '6' can be describes through two primitives: a descending curve and a loop.

Parizeau and Plamondon [45] construct a set of basic allographs using a fuzzy syntactic approach to model handwriting. For example, a primitive can be of type 'c' or 'ci', which corresponds to stroke segments with a shape similar to the letter 'c' where the 'i' indicates "inversion", i.e. a horizontal or vertical reflection of the shape. In this way, they consider twenty allographs, which correspond to i-shapes, c-shapes, t-crossings, loops, etc. A symbol is coded using a grammar formalism as a sequence of these primitives and the similarity between symbols is measured by using a sequence distance. Similar approaches are used in [7, 48, 66].

3.4.3 Neural Network and Statistical Models.

Guyon et al. [22] developed a neural network system for the recognition, based on so-called time delay neural networks. The system integrates recognition and segmentation in the same neural network architecture. The network is used to estimate a posteriori probabilities for characters in a word. One of the layers of the network converts the temporal information, point position, curvature, and direction in a two-dimensional image representation. Similar approaches are used in [38], which have been specially developed for the recognition of on-line handwritten words.

One of the most recent statistical approaches used for on-line recognition are Hidden Markov Models (HMM). They use dynamic programming to find the optimal alignment which performs character segmentation and recognition simultaneously. Bellegarda [4] uses local features, point positions and curvature, to feed a simple two-state HMM. Sin and Kim [53] also uses a HMM to model inter-letter relations, also known as ligatures. They train a HMM for each letter and ligature type. Schenkel et al. [49] uses a hybrid neural-network-HMM system for on-line word recognition.

Shwenk [50] uses an auto-encoder neural network to construct a model for each symbol class. This auto-encoder network actually describes each symbol class by finding the principal components that describe the class population. New symbols are fed into each network and projected to a hyperplane generated by the principal components. The tangent distance between the encoded (projected) vector and the original one is calculated. The class label assigned to the new vector corresponds to the one where the minimum distance is reached.

Bahlman et al. [3] combine dynamic time (DTW) warping and support vector machines (SVM) to classify isolated symbols. They use a Gaussian kernel to train the SVM-classifier and substitute the vector metric in the exponential argument of the kernel with the DTW distance. Theoretically, this substitution can lead to some numerical problems, because the DTW distance is not a metric, a strong assumption within the SVM approach to construct a kernel function. However, it was shown experimentally that only in an small set of training examples this assumption is not fulfilled, an issue that could be avoided by not considering that set.

Cho and Kim [14, 13] classify isolated symbols by modeling strokes internal relationships within a Bayesian network framework.

3.4.4 Clustering Methods.

Some of the above recognition approaches were developed in a user-dependent manner. Clustering methods in on-line handwriting are developed to remedy this situation. They try to model intra-class variations caused by different writing styles found in large databases. Vuori [62] uses a self-organizing map to cluster writing styles. She uses variations of the elastic matching method as a similarity measure. Her clustering algorithms can be used for prototype selection, which serves to classify characters according to the k-Nearest Neighbors (k-NN) rule. The recognition system adapts to new writing styles by modifying its prototype set. In the same fashion, Connell and Jain [15] use also elastic matching to measure intra-cluster and inter-class measures, to be used in a hierarchical clustering algorithm.

Watt and Xie [63] propose a clustering method for the classification of large on-line symbol databases. The proposed methods can recognize four symbols sets used in mathematical formulas which include digits, uppercase and lowercase Latin letters, Greek letters, mathematical operators and arrows, summing up a total of 277 symbol classes. They find clusters in the symbol database based on the following features: number of strokes, initial stroke position, width to height stroke ratio, and the initial and end direction as north, north-west, etc. They selected these features not only by their effectiveness, but also by their low computational cost. The cluster in the database are pruned by locating prototypes via a elastic matching distance. A new symbol is classified by locating it in a cluster and assigning it the label of the closest prototype using elastic matching. The method reduces the computation time with a small degradation of the classification rate.

4 Structural Analysis

4.1 Expression Formation

Expression formation consists of translating the results obtained in the symbol recognition step into a tree which describes the relations between symbols of the mathematical expression. The nodes that constitute this tree are a data structure which stores the symbol's label and other numerical parameters, for example size and location. Nodes are a compact representation of the original raw symbol. After an analysis of the relative positions of symbols, the nodes in the tree are linked to each other depending on which one of the spatial relations –above-left, above, superscript, right, subscript, below, below-left, and subexpression– they satisfy [6, 36]. As we will see, most of the methods described in this section rely on this principle.

Fateman et al. [17] developed a recognizer for typeset mathematical expressions. The recognized symbols are grouped into box tokens representing numbers, function names, and variables. They use a bottom-up recursive descent parser to obtain the final result as a lisp expression.

Lavirotte and Pottier [35] define a context-sensitive graph grammar to represent mathematical formulas and to remove ambiguities during structural analysis. Rules in graph grammars consist of collapsing a subgraph in a node, if they satisfy a given condition. They define a set of rules to describe the mathematical relation and apply a bottom-up parsing algorithm to obtain the final result. If we consider the expression 'e^x+1', a rule is defined to collapse the two nodes representing the symbols 'e' and 'x'. They now represent a single token, and the process can be continued by collapsing the new tokens 'e^x' and '1' to construct the final result.

Miller and Viola [42] use a generative model approach based on a stochastic-free grammar. The recognition process of the structure consists in finding the productions of the grammar with the highest probability when considering the overall probability of generating the expression. They use a convex hull condition to prune the searching space of the grammar productions. A production in the grammar is not admissible if the convex hull of a symbol group intersects a symbol not belonging to the group.

Winkler et al. [65] used HMMs for segmentation and classification of symbols. Using a soft-decision approach, they construct a hypothesis net of all possible segmentations of strokes. They store the information of the possible spatial relations between symbols in a directed graph, and a set of alternative interpretations of the expression is given.

Kosmala et al. [32] also present a statistical approach based on the application of HMMs. The system allows simultaneous segmentation and symbol classification, as well as the interpretation of the symbols' spatial relationships. One of the conditions for correct segmentation and recognition is that the expression be written from left to right and from top to bottom. Delayed symbols are not allowed, as when first writing an expression and then enclosing it in parenthesis. In an posterior work [31] they use graph rewriting for the structure analysis to avoid the above-mentioned restrictions of handwriting.

Chan and Yeung [8] propose the use of a definite-clause grammar (DCG) to define precise replacement rules when parsing mathematical expressions. They use DCG because a grammar expressed through this formalism is declarative and easy to maintain and extend. They use a parsing scheme using left-factoring to reduce the time for the expression's interpretation.

Eto and Suzuki [16] use a weighted graph to represent spatial relations between symbols. The nodes in the tree represent the symbols obtained during the segmentation stage. They are connected depending on whether they satisfy the superscript, subscript, or right relations. Because of the multiple results obtained during the recognition step, nodes have multiple edges which store a symbol's label and weight. They generate a set of spanning trees of the initial graph, which result from the minimization of the edge costs. If they result in an admissible (not contradictory) mathematical structure, one of them is selected as the final result if it minimizes a global cost criterion.

Zanibbi et al. [68] describe a mathematical expression as a structure of nested baselines, a so-called *baseline structure tree*. Baselines constitute horizontal arrangements of symbols. The first step in the procedure recursively constructs the baseline tree in the handwritten expression by handling irregular or poor horizontal layout structures. The second step converts the structure tree into an operator tree, which is further processed by applying tree transformations using the TXL language. Tree transformations consists of locating patterns in the tree structure and applying replacing rules on them. Defining different replacing rules helps to define a particular dialect or recognition scope of mathematical notation.

Zahng et al. [70] use fuzzy logic to define spatial relationships among mathematical symbols. They use fuzzy membership functions to define the superscript, subscript and horizontal symbol relationships. They produce recursively different alternatives for the interpretation of a expression in the case of non-zero membership values. To illustrate de procedure, consider the expression $a^{bc}d$, where a is the dominant symbol in the expression. The fuzzy regions provide two interpretation for the spatial relations between a and b, generating the two alternatives ab and a^b. The method proceeds recursively with these two subexpressions, generating another two interpretations a^bc and $a^{[bc]}$ for the latter. The square brackets indicate that the symbols they embrace must be analyzed as well. A final interpretation is a symbol group which can not generate any alternative. Finally, confidence values are assigned to final alternatives by taking the minimum membership value among all the fuzzy spatial relationships in the expression. The interpretations are presented to the user as list ordered by the confidence values.

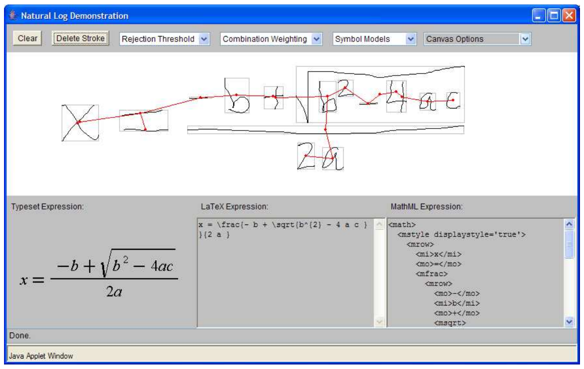
4.2 Error Correction

Most of the approaches mentioned before convert a two-dimensional representation of the expression into a one-dimensional one, a string of characters, which represents the expression in a certain computer language. Erroneous symbol segmentation and classification can generate incorrect results during structural analysis.

Lee and Wang [36] propose heuristics for the correction of the most common errors derived by false recognized symbols. They consider four heuristic rules. One of them is "for every binary operator there must exist two operands". By applying this rule the expression '$/ < i < n$' is corrected to '$1 < i < n$'. They also define a rule to consider errors encountered during the construction of function names. An expression like '$5inx$' should be corrected to '$\sin x$'. Numerals with subindexes are considered as an error. Expressions like '1_2' and '5_b' are corrected to 'l_1' and 'S_b' respectively. The last rule considers the correction derived by case confusion of letters and other character properties, considering whether it is italic, bold, etc., because letters forming a determined group in the expression are similar. Expressions like '$3Pqr$' and '$\mathrm{x} \cdot y$' are transformed to '$3pqr$' and '$x \cdot y$'.

Chan and Yeung [10] extend their system based on definite-clause grammar to handle lexical, syntactic, and semantic errors. They implement some rules to identify common errors in string grammar, namely substitution, deletion, and insertion. They also incorporate rules for the correction of some common syntactic errors, like incorrect implicit operators, missing binding and fence symbols, or missing operands. The semantic errors they consider are corrected heuristically, similar to the way Lee and Wang do. The lexical errors they consider take place at the recognition and segmentation level, caused by poor digitalization or irregular writing. They also propose some performance measures that concern the recognition of expressions, symbols, and operators, as well as an integrated performance measure.

Actually, not much effort is expended on the error correction task. Some other authors does not handle automatically the correction of errors. For this purpose, they offer instead user-friendly interfaces which allow immediate feedback and have undo-redo and visualization capabilities, as we will see in the next section.



$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

```
x = \frac{- b + \sqrt{b^{2} - 4 a c }
}{2 a }
```

```
<math>
  <mstyle displaystyle='true'>
    <mrow>
      <mi>x</mi>
      <mo>=</mo>
      <mfrac>
        <mrow>
          <mo>-</mo>
          <mi>b</mi>
          <mo>+</mo>
          <msqrt>
```

Figure 9: The Natural Log system.

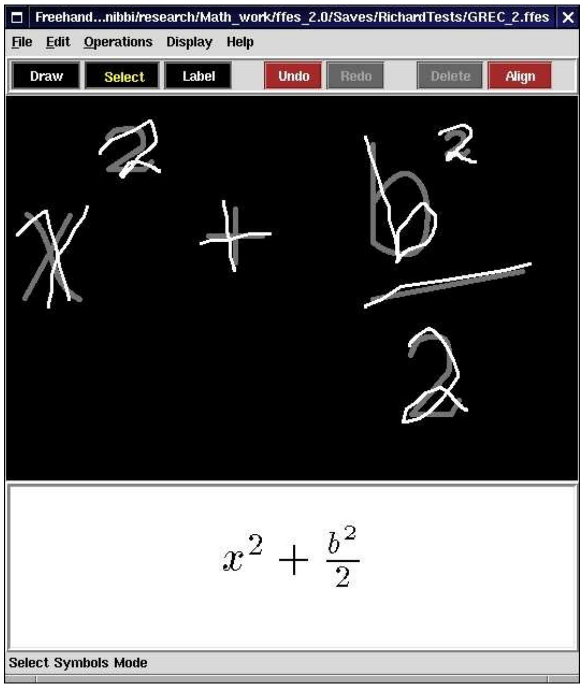
5 User Interfaces

5.1 The Natural Log System

The Natural Log system is a user-dependent system developed by Matsakis [41]. The system was written in Java and is only available on-line as an applet on the internet. See Fig. 9.

To classify on-line symbols, he constructs a high-dimensional normal distribution, which describes the population of each class. The symbol label corresponds to the class that has the maximum probability. Low probability values are used to reject symbols which can represent potential errors in the handwriting. The procedure to recognize a given mathematical expression begins by finding an optimal grouping of the written strokes into isolated symbols. The final grouping of stroked is determined by evaluating all possible groupings and taking the one which minimizes a sum-cost function. This function is the sum of the log likelihood of the classifier output of each symbol in the current partition. To make the optimization of the cost function manageable, its evaluation is constrained by the minimum spanning tree of strokes, considering the centers of strokes' bounding boxes as nodes of a completely connected weighted graph. Different combinations of subtrees of the minimum spanning tree are evaluated and the optimal one is taken as the final segmentation result.

The structural analysis in this system consists of locating a "key" symbol usually an explicit mathematical operator. Once the key symbols are located, the parse algorithm proceeds to find their corresponding operands, and partial subexpression are formed. The procedure is applied recursively until no more key symbols are found. The algorithm is extended to support parsing of superscripts, i.e. to non-explicit operators, but no support for subindexes is offered.



$$x^2 + \frac{b^2}{2}$$

Figure 10: Free Hand Formula Entry System

5.2 Free Hand Formula Entry System

The Freehand Formula Entry System is a pen-based equation editor developed by Smithies and Norvins [55] and distributed under the GNU General Public License. The program runs under Linux and MacOS X platforms.

The classification of symbols is done by using the nearest-neighbor method. The developers use confidence information supplied by the classifier to group strokes. Their method proceeds by generating all possible combinations of a fixed number of strokes (by default they take a maximum of four strokes), which potentially can constitute a single symbol. Once single symbol is classified, the confidence level of a combination correspond the lowest output of the classifier. Finally, the group with the highest confidence is taken and the first symbol in the group is returned and considered a correctly recognized character. The procedure is repeated, once again, when a fixed number of input strokes is reached.

For the structural analysis, they first used a method based on graph rewriting, similar to the method developed by Lavirotte and Pottier [35]. Figure 10 shows a version of their program, modified by Zanibbi [68], which uses the structural analysis method he developed.

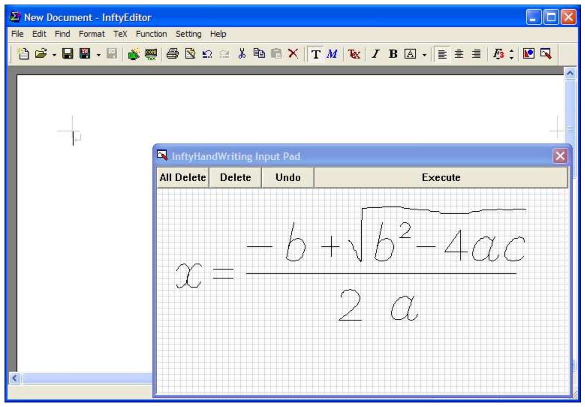

New Document - InftyEditor
File Edit Find Format TeX Function Setting Help
InftyHandWriting Input Pad
All Delete
Delete
Undo
Execute

Figure 11: Infty editor

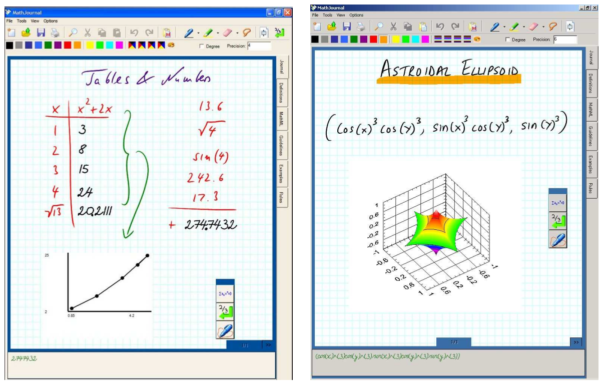


Figure 12: MathJournal.

5.3 Infty Editor

Infty Editor [44, 56] is a commercial system specialized for creating mathematical documents. The editor is linked to the computer algebra system Mathematica by Mathlink. It also supports input and output of expressions in TEX format. The editor contains an real-time recognition system for mathematical expressions. See Fig. 11.

The recognition system combines segmentation and recognition of characters to remedy difficulties in structural analysis due to irregular symbol position and size [27]. The rewriting puts symbols into extendable symbols and unextendable ones. The former are extended to form other symbols by adding more strokes and the latter are written with only one stroke. For example, F can be extended into E. When a stroke is classified as unextendable, the classification result is rewritten by the computer in the drawing area using a predefined prototype. If a stroke is classified as an extendable character, the system waits for the next strokes. The classification result is written automatically if a predetermined time interval has elapsed or the expected number of strokes is reached.

5.4 MathJournal

Wenzel and Dillner [64] describe another commercial product, MathJournal, developed for the Tablet PC version of the Windows platform. The interface is very similar to Microsoft's Journal program, which is included in the operating system, see Fig. 12. Apparently, the program is still under development and only descriptions of it are given. The developers also offer the xThink Calculator as an evaluation program. The recognition capabilities of this program are similar to the ones of MathJournal. It operates as a normal pocket calculator, the operations are done by recognizing a handwritten arithmetical expression.

MathJournal uses the recognizer integrated in the operating system for the classification of isolated handwritten characters. Although it is possible to recognize special mathematical symbols and constants (square root, calculus operators, and the constant π), the system does not recognize Greek letters. The symbols recognized by the system are limited to the ones recognized by the Microsoft API.

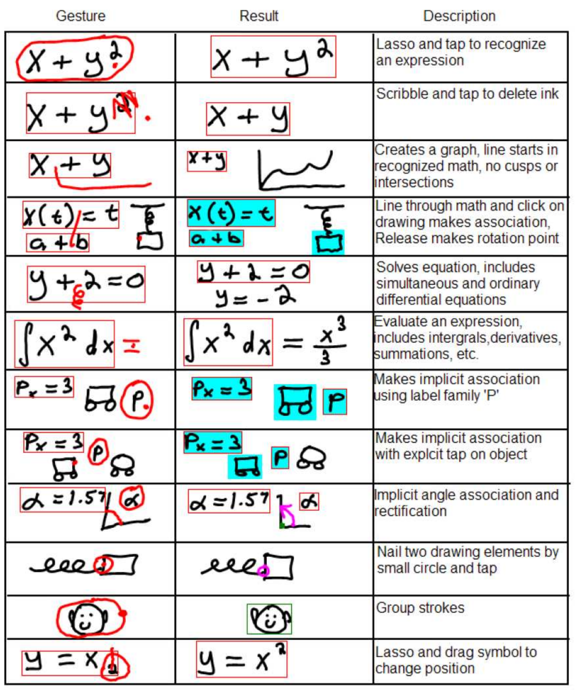
In the description of the system, they mention that heuristics of graph rewriting and, when required, a minimum spanning tree construction are used during structural analysis.

The most relevant aspects of this system are its "solution engines". They process the recognized expressions in numeric, graphic, or symbolic formats. Diagrams, such as function tables, are processed and plotted by using curly braces and arrows as gestures. Similar gestures are used for the solution of equation systems or for plotting functions.

5.5 MathPad

MathPad is a system for the creation and exploration of mathematical sketches. The main objective in this user interface is to facilitate the user-editor interaction using sketches. This gestural user interface is one of the most advanced in terms of pen-based design.

The set of gestures are used to interact with the recognition and processing engine included in the system. The basic interaction is the drawing of a lasso gesture followed by a tap. This action indicates the system the recognition of the selected strokes. It is used also for the association of variables and stroke grouping. Drawing the equals sign followed by a tap or by the minus sign allows the solution of equations



Gesture	Result	Description
$X+y^2$	$X+y^2$	Lasso and tap to recognize an expression
$X+y^2$	$X+y$	Scribble and tap to delete ink
$X+y$	$x+y$	Creates a graph, line starts in recognized math, no cusps or intersections
$x(t)=t$ $a+b$	$x(t)=t$ $a+b$	Line through math and click on drawing makes association, Release makes rotation point
$y+2=0$	$y+2=0$ $y=-2$	Solves equation, includes simultaneous and ordinary differential equations
$\int x^2 dx$ =	$\int x^2 dx = \frac{x^3}{3}$	Evaluate an expression, includes intergrals,derivatives, summations, etc.
$P_x=3$ P	$P_x=3$ P	Makes implicit association using label family 'P'
$P_x=3$ P	$P_x=3$ P	Makes implicit association with explcit tap on object
$\alpha=1.57$ α	$\alpha=1.57$ α	Implicit angle association and rectification
		Nail two drawing elements by small circle and tap
		Group strokes
$y=x^2$	$y=x^2$	Lasso and drag symbol to change position

Figure 13: Gestures used in the mathematical sketching paradigm.

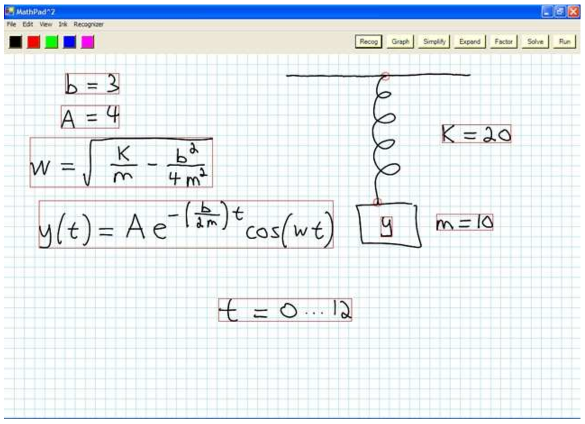
or factorization of expressions. Other gestures are used for deletion of ink, association of recognized expressions, and angle association. Figure 13 showa the different gestures used in MathPad.

The system is writer-dependent to guarantee the a more accurate symbol recognition. Handwriting is recognized using hybrid recognizer. First, a dynamic classifier calculates the similarity with prototypes [37] is combined with an statistical classifier [55], to obtain information to use once again dynamic programming for the fine classification [15]. The structural analysis uses the techniques described in [6].

References

[1] L. Anthony, J. Yang, and K.R. Koedinger. Evaluation of Multimodal Input for Entering Mathematical Equations on the Computer. *Conference on Human Factors in Computing Systems*, pages 1184–1187, 2005.

[2] X. Ao, J. Li, X. Wang, and G. Dai. Structuralizing digital ink for efficient selection. *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 148–154, 2006.



MathPad^2
File Edit View Ink Recognizer
Recog Graph Simplify Expand Factor Solve Run
$b=3$
$A=4$
$w=\sqrt{\frac{K}{m}-\frac{b^2}{4m^2}}$
$K=20$
$y(t)=Ae^{-\left(\frac{b}{2m}\right)t}\cos(wt)$
y
$m=10$
$t=0\dots12$

Figure 14: MathPad.

[3] C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines-a kernel approach. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 49–54, 2002.

[4] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo, and K.S. Nathan. A Probabilistic Framework for On-Line Handwriting Recognition. In *Proceedings of the Third International Workshop Frontiers of Handwriting Recognition*, pages 225–234, May 1993.

[5] J. Blanchard and T. Artieres. On-line handwritten documents segmentation. *Proceedings of Ninth International Workshop on Frontiers in Handwriting Recognition.*, pages 148–153, 2004.

[6] D. Blostein and A. Grbavec. Recognition of Mathematical Notation. In P. S. P. Wang and H. Bunke, editors, *Handbook on Optical Character Recognition and Document Analysis*. World Scientific Publishing Company, 1996.

[7] K. Chan. A Simple Yet Robust Structural Approach for Recognizing On-Line Handwritten Alphanumerical Characters. In *Proceedings of the sixth international workshop on frontiers in handwriting recognition*, 1998.

[8] K. F. Chan and D. Y. Yeung. An Efficient Syntactic Approach to Structural Analysis of On-Line Handwritten Mathematical Expressions. *Pattern Recognition*, 33(3):375–384, March 2000.

[9] K. F. Chan and D. Y. Yeung. Mathematical Expression Recognition: a Survey. *International Journal on Doucument Analysis and Recognition*, 3(1):3–15, 2000.

[10] K. F. Chan and D. Y. Yeung. Error Detection, Error Correction and Performance Evaluation in On-Line Mathematical Expression Recognition. *Pattern Recognition*, 34(8):1671–1684, August 2001.

[11] B. B. Chaudhuri and U. Garain. An Approach for Recognition and Interpretation of Mathematical Expressions in Printed Documents. *Pattern Analysis and Applications*, 3(2):120–131, 2000.

[12] P. Chiu and L. Wilcox. A dynamic grouping technique for ink and audio notes. *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 195–202, 1998.

[13] S.J. Cho and J.H. Kim. Bayesian networkmodeling of strokes and their relationships for on-line handwriting recognition. *Pattern Recognition*, 37:253–264, 2004.

[14] H. Choi, S.J. Cho, and J.H. Kim. Writer Dependent Online Handwriting Generation with Bayesian Network. In *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 130–135, 2004.

[15] S. D. Connell and A. K. Jain. Writer Adaptation for On-Line Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):329–346, March 2002.

[16] Y. Eto and M. Suzuki. Mathematical Formula Recognition Using Virtual Link Network. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 430–437, Seattle, 2001.

[17] R. J. Fateman, T. Tokuyasu, B. P. Berman, and N. Mitchell. Optical Character Recognition and Parsing of Typeset Mathematics. *Journal of Visual Communication and Image Representation*, 7(1):2–15, 1996.

[18] U. Garain and BB Chaudhuri. Recognition of online handwritten mathematical expressions. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(6):2366–2376, 2004.

[19] W. Guerfali and R. Plamondon. Normalizing and Restoring On-Line Handwriting. *Pattern Recognition*, 26(3):419–431, 1993.

[20] C. Guy, M. Jurka, S. Stanek, and R. Fateman. Math Speak & Write, a Computer Program to Read and Hear Matematical Input. University of California Berkeley, August 2004.

[21] C. Guy and S. Stanek. Math Speak & Write: A Programmer's Guide. University of California Berkeley, August 2004.

[22] I. Guyon, P. Albrecht, Y. LeCun, J. S. Denker, and Hubbard W. Design of a Neural Network Character Recognizer for a Touch Terminal. *Pattern Recognition*, 24(2):105–119, 1991.

[23] N. Higham. *Handbook of Writing for the Mathematical Sciences*. SIAM, 1993.

[24] T. Igarashi, S. Matsuoka, and T. Masui. Adaptive recognition of implicit structures in human-organized layouts. *Proceedings of 11th IEEE International Symposium on Visual Languages*, pages 258–266, 1995.

[25] A.K. Jain, A.M. Namboodiri, J. Subrahmonia, and Y. Heights. Structure in On-line Documents. *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 844–848, 2001.

[26] A. Kacem, A. Belaid, and M. Ben Ahmed. Automatic Extraction of Printed Mathematical Formulas Using Fuzzy Logic and Propagation of Context. *International Journal on Document Analysis and Recognition*, 4(2), 2001.

[27] T. Kanahori, K. Tabata, W. Cong, F. Tamari, and M. Suzuki. On-Line Recognition of Mathematical Expressions Using Automatic Rewriting Method. *Advances in Multimodal Interfaces–ICMI2000*, pages 394–401, 2000.

[28] L.B. Kara and T.F. Stahovich. Hierarchical Parsing and Recognition of Hand-Sketched Diagrams. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, pages 13–22. ACM Press New York, NY, USA, 2004.

[29] D. Knuth. Mathematical Typography. *Bulletin of the American Mathematical Society*, 1979.

[30] M. Koschinski, H. Winkler, and M. Lang. Segmentation and Recognition of Symbols within Handwritten Mathematical Expressions. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2439–2442, Detroit, MI, 1995.

[31] A. Kosmala, S. Lavirotte, L. Pottier, and G. Rigoll. On-Line Handwritten Formula Recognition using Hidden Markov Models and Context

Dependent Graph Grammars. In *Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, 1999.

[32] A. Kosmala and G. Rigoll. On-Line Handwritten Formula Recognition Using Statistical Methods. In *International Conference on Pattern Recognition*, page PR33, 1998.

[33] J. LaViola. *Mathematical Sketching: A New Approach to Creating and Exploring Dynamic Illustrations.* PhD thesis, Brown University, Department of Computer Science, May 2005.

[34] J. LaViola and R. C. Zeleznik. MathPad2: a system for the creation and exploration of mathematical sketches. *ACM Trans. Graph.*, 23(3):432–440, 2004.

[35] S. Lavirotte and L. Pottier. Optical Formula Recognition. In *International Conference on Document Analysis and Recognition*, pages 357–361, 1997.

[36] H. J. Lee and J. S. Wang. Design of a Mathematical Expression Understanding System. *Pattern Recognition Letters*, 18(3):289–298, March 1997.

[37] X. Li and D.Y. Yeung. On-Line Handwritten Alphanumeric Character Recognition Using Dominant Points in Strokes. *Pattern Recognition*, 30(1):31–44, 1997.

[38] R. Lyon and L. Yaeger. On-Line Hand-Printing Recognition with Neural Networks. In *Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, Lausanne, Switzerland, 1996. IEEE Computer Society Press.

[39] I.S. MacKenzie and S. Zhang. The immediate usability of Graffiti. *Proceedings of Graphics Interface*, 97:129–137, 1997.

[40] E. Mandler. Advanced Preprocessing Technique for On-Line Recognition of Handprinted Symbols. In C. Y. Suen R. Plamondon and M. L. Simmer, editors, *Computer Recognition and Human Production of Handwriting*, pages 19–36. World Scientific, 1989.

[41] N. Matsakis. Recognition of Handwritten Mathematical Expressions. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1999.

[42] E. G. Miller and P. A. Viola. Ambiguity and Constraint in Mathematical Expression Recognition. In *The Fifteenth National Conference on Artificial Intelligence*, pages 784–781, 1998.

[43] K. Nakagawa and M. Suzuki. Mathematical Knowledge Browser with Automatic Hyperlink Detection. In *MKM*, pages 190–202, 2005.

[44] H. Okamura, T. Kanahori, M. Suzuki, H. Fakuda, R. Cong, and F. Tamari. Handwriting Interface for Computer Algebra Systems. In *Proceedings of Graphics Interface*, 1999.

[45] M. Parizeau and R. Plamondon. A Fuzzy-Syntactic Approach to Allograph Modeling for Cursive Script Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):702–712, July 1995.

[46] E. H. Ratzlaff. Methods, reports and survey for the comparison of diverse isolated character recognition results on the UNIPEN database. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 623–628, 2003.

[47] E.H. Ratzlaff. A scanning n-tuple classifier for online recognition of handwritten digits. In *Proceedings of the Sixth International Workshop on Frontiers in Handwriting Recognition*, 2001.

[48] E. G. Sanchez, J. A. G. Gonzalez, Y. A. Dimitriadis, J. M. C. Izquierdo, and J. L. Coronado. Experimental Study of a Novel Neuro Fuzzy System for Online Handwritten Unipen Digit Recognition. *Pattern Recognition Letters*, 19(3-4):357–364, March 1998.

[49] M. Schenkel, I. Guyon, and D. Henderson. On-line Cursive Script Recognition Using Time-Delay Neural Networks and Hidden Markov-Models. *Machine Vision and Applications*, 8(4):215–223, 1995.

[50] H. Schwenk and M. Milgram. Constraint Tangent Distance for On-Line Character Recognition. In *International Conference on Pattern Recognition*, August 1996.

[51] M. Shilman, P. Viola, and K. Chellapilla. Recognition and grouping of handwritten text in diagrams and equations. In *Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 569–574, October 2004.

[52] M. Shilman, Z. Wei, S. Raghupathy, P. Simard, and D. Jones. Discerning structure from freeform handwritten notes. *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 60–65, 2003.

[53] B. K. Sin and J. H. Kim. Ligature Modeling for Online Cursive Script Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):623–633, 1997.

[54] E. Smirnova and S. M. Watt. A context for pen-based computing. In *Maple Conference 2005*, pages 409–422, Waterloo Canada, July 2005.

[55] S. Smithies, K. Novins, and J. Arvo. A Handwriting-Based Equation Editor. In *Graphics Interface*, pages 84–91, 1999.

[56] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori. INFTY: an integrated OCR system for mathematical documents. In *Proceedings of the 2003 ACM symposium on Document engineering*, pages 95–104. ACM Press New York, NY, USA, 2003.

[57] M. Suzuki, S. Uchida, and A. Nomura. A Ground-Truthed Mathematical Character and Symbol Image Database. *Proceedings of the Eigth International Workshop on Frontiers in Handwriting Recognition*, pages 675–679, 2005.

[58] C. C. Tappert. Cursive Script Recognition by Elastic Matching. *IBM Journal of Research and Development*, 26(6):765–771, 1982.

[59] 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.

[60] S. Toyota, S. Uchida, and M. Suzuki. Structural Analysis of Mathematical Formulae with Verification Based on Formula Description Grammar. In *Workshop on Document Analysis Systems (DAS06)*, pages 153–163, 2006.

[61] R. Veltkamp and M. Hagedoorn. State-Of-The-Art in Shape Matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.

[62] V. Vuori. Clustering Writing Styles with a Self-Organizing Map. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 345–350, 2002.

[63] S.M. Watt and X. Xie. Recognition for Large Sets of Handwritten Mathematical Symbols. In *Proceedings of th Eighth International Conference on Document Analysis and Recognition*, pages 740–744, 2005.

[64] L. Wenzel and H. Dillner. MathJournal – An Interactive Tool for the Tablet PC. `http:\\www.xthink.com`.

[65] H. Winkler, H. Fahrner, and M. Lang. A Soft-Decision Approach for Structural Analysis of Handwritten Mathematical Expressions. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2459–2462, Detroit, MI, 1995.

[66] L. P. Yang and R. Prasad. Online Recognition of Handwritten Characters Using Differential Angles and Structural Descriptors. *Pattern Recognition Letters*, 14(12):1019–1024, December 1993.

[67] M. Ye, H. Sutano, S. Raghupathy, C. Li, and M. Shilman. Grouping text lines in freeform handwritten notes. In *International Conference on Document Analysis and Recognition*, 2005.

[68] 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.

[69] R. Zanibbi, D. Blostein, and J.R. Cordy. Directions in Recognizing Tabular Structures of Handwritten Mathematics Notation. *Proceedings of the Fourth International IAPR Workshop on Graphics Recognition*, pages 493–499, 2001.

[70] L. Zhang, D. Blostein, and R. Zanibbi. Using Fuzzy Logic to Analyze Superscript and Subscript Relations in Handwritten Mathematical Expressions. In *International Conference on Document Analysis and Recognition*, 2005.

[71] L. Zhang and R. Fateman. Survey of User Input Models for Mathematical Recognition: Keyboards, Mice, Tablets, Voice. Technical report, University of California, 2003.