# A System for Real-time Recognition of Handwritten Mathematical Formulas

Kenichi Toyozumi
Graduate School of Eng., Nagoya Univ.
Nagoya 464-8603 Japan
ktoyo@suenaga.cse.nagoya-u.ac.jp

Takahiro Suzuki
Canon Inc. Corporation
30-2, shimomaruko 3-chome, Ohtaku,
Tokyo 146-8501 Japan

Kensaku mori, Yasuhito Suenaga
Faculty of Eng., Nagoya Univ.
Nagoya 464-8603 Japan
{mori, suenaga}@cse.nagoya-u.ac.jp

## Abstract

*This paper presents an expanded system for the on-line recognition of handwritten mathematical formulas. Our target handwritten mathematical formulas are strokes drawn on a data tablet. This system recognizes such strokes as components of mathematical formulas on the basis of their positions and combinations. Including matrix structures, general mathematical expressions are acceptable for this system. Each recognition result is acquired as a LaTeX source code. This system also has a preview function to enable a more highly intuitive recognition result. In recognition experiments, this system proved to be fairly feasible in handling handwritten mathematical formulas in real-time.*

## 1. Introduction

Nowadays, a lot of people use computers to write documents. In the case of technical documents, mathematical formulas are often important. LaTeX is an example of typesetting software that can handle both texts and mathematical formulas. However, the typesetting for mathematical formulas requires many tag commands. Users therefore find the commands difficult. There are also other well-known software applications for writing. Typically, however, they require a special application for writing mathematical formulas.

MathType (Design Science Inc. CA USA) is an example of an application that enables users to insert mathematical formulas into documents. A user can easily understand what kind of formula is inserted through the WYSIWYG (What You See Is What You Get) environment of this application. However, many of the operations of the package are complicated, e.g., alphanumeric characters must be inputted through the keyboard and mathematical symbols and templates of structures have to be chosen from the GUI (Graphical User Interface) menus. These operations are so complicated that the user cannot easily handle mathematical formulas.

One solution to these problems is to employ a system that can handle handwritten mathematical formulas in real time. With such a system, the user can write formulas by using an implement such as the stylus of a tablet board. The system would recognize the currently written mathematical formula and insert it into the document automatically. On-line handwritten character recognition has been widely used as an intuitive input approach for computers. For the handling of mathematical formulas, the system should be made to perform both character recognition and mathematical structure recognition based on the combinations and locations of characters.

There have been many research efforts to achieve handwritten mathematical formula recognition [1]- [5]. A common problem of previous studies has been limitations concerning recognizable structures and writing orders of mathematical formulas. In consideration of this, we have been developing a system that can recognize handwritten mathematical formulas in real-time [6]. Earlier, this system had excluded matrix structures as a recognition target, and the LaTeX source code that it had outputted was difficult for users to understand.

The purpose of this paper is to describe our new system and the measures we have taken to overcome these deficiencies. In the next section, the specifications of the system are given. Section 3 explains our recognition methods and the expansion to matrices. Section 4 describes the functions of the system. Section 5 shows evaluation experiments and results. Finally, Section 6 briefly summarizes this paper.

0-7695-1263-1/01/$10.00 © 2001 IEEE

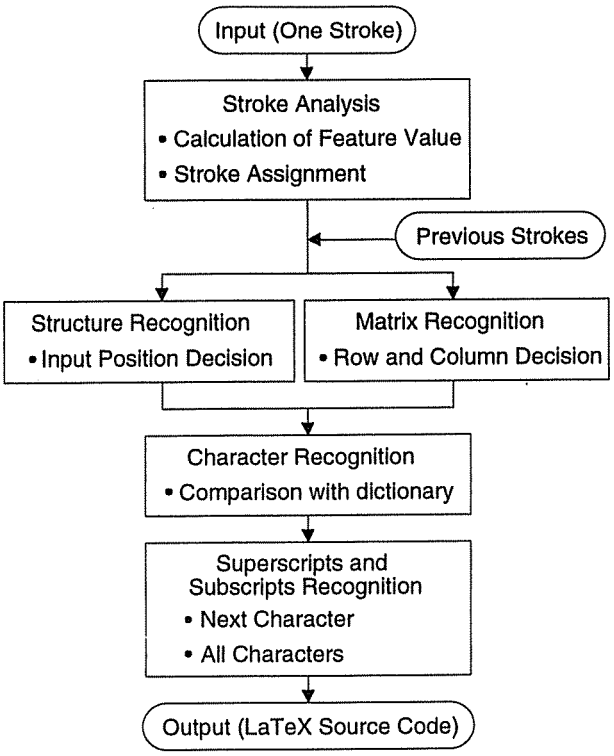


**Figure 1. System flow**

## 2. System Outline

Figure 1 shows the processing flow of the system. The inputs to this system are handwritten strokes drawn on a computer screen by a pointing device such as a mouse or a data tablet. These strokes are combined into characters, and the characters are the components of a mathematical formula. The system analyzes the handwritten inputs stroke-by-stroke and interprets the geometrical characteristics of these strokes into a mathematical expression by using thresholds based on heuristic knowledge. This process is performed independently of the stroke order. This system accepts structures derived from several specific symbols or superscripts and subscripts. The system is also able to recognize the structures of matrices, which is a function that has not been implemented in any other system able to handle handwritten mathematical formulas in real-time, as far as we know. Finally, the system outputs the recognition result as a LaTeX source code.

To increase the usefulness of the input method, we have designed the system to allow the user to delete arbitrary characters. Moreover, the horizontal lines of fractions and radical signs can be extended to cover additional strokes within these symbols in the course of inputting. This function is useful since people often extend such symbols during the actual handwriting of mathematical formulas.

Unintelligible outputs had also been a problem with this system. To overcome this problem, we devised a preview engine to provide intelligible representations of all LaTeX source code. This function enables the user to easily understand the recognition process of the system during input.

## 3. Recognition Procedures

### 3.1. Stroke Analysis

First, each input stroke is assigned to the most appropriate character composed of a single stroke. The assignment is based on the correlation between the input stroke and a stroke in the prepared dictionary. In order to calculate this correlation, the following three kinds of features are extracted from the input stroke, which is handled as a sequence of coordinates.

*A chain code histogram* of each stroke is acquired by encoding the input stroke into a sequence of Freeman chain codes. This feature involves counting each code. *A sub-vector sequence* is calculated by dividing the input stroke into subsections of equal length. Each element vector of this sequence is the connection of the start point and the end point of each subsection. In an exceptional case when the input stroke is too short to divide, the system assigns a "dot" to this stroke. *Region information* is extracted on the basis of the bounding box of a stroke. This feature consists of the height and width of the box and the start point of the stroke in the box.

### 3.2. Character Recognition

In the second procedure, we consider the combination of strokes. Each stroke and character has information on whether it can constitute a valid character with the stroke/character inputted before it. For example, '3' still has the probability of being a part of 'B'. on the other hand, the cursive script of 'y' has no possibility of making up another character. The combination of strokes is made on the basis of this information and the closeness between the stroke positions. The system regards the positions of strokes as being in a neighborhood when the intersection between the strokes is larger than some threshold. The system calculates the correlation of each possible combination of the strokes and decides the most appropriate combination of characters.

Corresponding to the case of one stroke, the following three connected features are used in consulting the prepared dictionary. *A connected chain code histogram* is the direct connection of the chain code histograms of all strokes. *A connected sub-vector sequence* is acquired by connecting two sub-vector sequences with the vector between the end point of the previous sequence and the start

point of the next sequence. *Connected region information* includes the vertical and horizontal distances between the upper/lower/left/right edges of two bounding boxes. We prepared a dictionary for each possible number of strokes in order to save calculation time in the correlations.

### 3.3. Structure Recognition

Finally, the vertical locations of the input strokes should be determined in relation to variance with a typical text structure where characters are aligned in the same line. We know that some particular symbols make a peculiar mathematical structure. Underline, upper line, fraction line, radical sign, summation, and production are acceptable symbols in this system. Since the structure recognition method using the properties of these symbols is stable and powerful, it is performed to determine the positions of the input strokes before the character recognition phase if these symbols have been detected. This process divides a mathematical formula into blocks from which the particular symbols are excluded, consequently.

Superscripts and subscripts are also written as mathematical structures. The relation of two adjacent characters is classified into three categories: superscripts, subscripts, and the same line. The system recognizes these mathematical structures by setting some thresholds for the heights of the bounding boxes of the characters and the distances between the vertical positions of the characters. In order to perform this process precisely, the baseline of each character is defined beforehand. The combination of these structures often results in ambiguity. This system interprets the chain of characters using a criterion that subscripts of a superscript of one character are also superscripts of that character, and vice versa. This criterion is shown in Fig. 2. In the left case, **b** and **c** are superscripts of **a**, and the system recognizes **c** as a subscript of **b**. On the other hand, **f** is located on the same line as **d** in the right situation, and so **f** cannot be recognized as a subscript of **e**.

Most mathematical formulas can be represented as the multiple and nested structures described above. However, the structures of matrices cannot be expressed in this manner. The system must have a special procedure since characters are located in grids without any explicit symbols in a matrix structure. Figure 3 shows the general structure

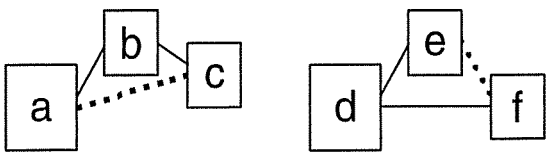


**Figure 2. Chains of characters**

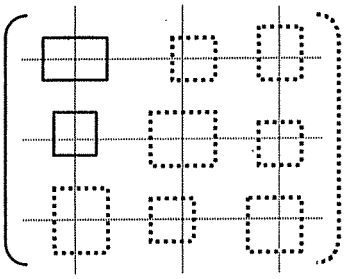

**Figure 3. Grid structure of matrices**

of a matrix. We define an inputted structure as a matrix when characters are located vertically and a left parenthesis exists just to their left; this is shown by the solid lines in Fig. 3. The user can input elements of a matrix either rowwise or coloumnwise arbitrarily. Once the system detects a matrix structure, the input strokes can be easily arranged as elements of the rows or columns according to their inputted positions and the sizes of their bounding boxes. Adequate spaces are required between the individual rows and columns to separate the elements of the matrix. This procedure for matrices works until the right parenthesis is inputted as the end of the matrix structure.

## 4. Functions of the System

### 4.1. Input Assistance

To handle the user's mistakes in writing and the system's mistakes in recognition, the system enables the user to delete arbitrary characters. When the user uses the pointing device to indicate the inside of the bounding box of a character, the system deletes the character. In the drawing process, the system draws the specified character with the background color.

Another requirement in inputting strokes is being able to add a stroke to a particular symbol. The fraction line and the radical sign are extendable symbols. Since the stroke of writing an extension would normally be recognized as a subtraction sign, the system regards such an input as an extension if the start point of the input is near the end of an extendable symbol.

### 4.2. Output Assistance

The preview function provides an intelligible result for the user. It is required the prepared font images and the outputted LaTeX source code of the recognition result. The preview engine lexically analyzes the LaTeX source code on the basis of its particular symbols such as '\', '^', and '_'. According to the result of this analysis, the system arranges

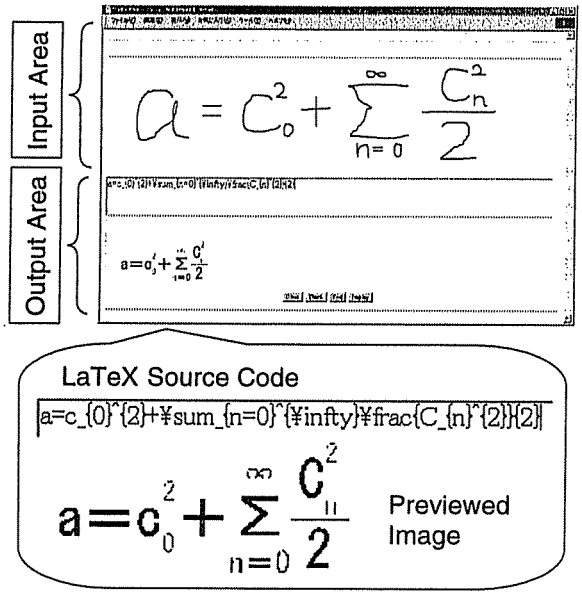


**Figure 4. Snapshot of this system**

$$y = \int_{-\infty}^{+\infty} f(x)dx \quad (1) \qquad g = \frac{2q}{1+\sqrt{1+p^2+q^2}} \quad (2)$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m)^2}{\sigma^2}} \quad (3) \qquad a = c_0^2 + \sum_{n=0}^{\infty} \frac{c_n^2}{2} \quad (4)$$

$$\begin{pmatrix} a+1 & b+2 \\ c+3 & c+4 \end{pmatrix} \quad (5) \qquad \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} \quad (6)$$

**Figure 5. Evaluated mathematical formulas**

the images of characters naturally by adjusting the sizes and positions of the characters in correspondence to their locations and mathematical structures.

## 5. Experiment

The system is implemented as a Java applet. A screenshot of the system is shown in Fig. 4. The user writes a mathematical formula in the upper window. The middle and bottom windows are areas for the recognition output expressed as a LaTeX source code and for the preview, respectively. We conducted an experiment to evaluate the performance of this system. Figure 5 shows the evaluated mathematical formulas. Each mathematical formula was inputted into the system stroke-by-stroke with a data tablet. The recognition of each stroke was finished in less than 120 milliseconds on a PentiumIII 600MHz. We prepared four dictionaries based on "character type" for each possible number of strokes. The dictionaries consist of character features, which are averaged out of one hundred samples inputted beforehand. Table 1 shows the characters that the system can recognize.

Table 2 shows the results of this evaluation. We achieved an 80% character recognition rate and a 92% recognition rate for the mathematical structures. Results for matrix recognition are shown in Table 3. At present, the best recognition rate we have achieved is 69%. Almost all of our recognition failures occur at the beginning of the procedure for recognizing the matrix structure. An example of a mis-recognition is shown in Fig. 6. Since it is impossible for '(' to have superscripts and subscripts, adding some restrictions might be effective in avoiding this type of mis-recognition. If we can make the system succeed in detecting matrices, the rate of recognition can rise drastically to 98%. Therefore, improving the matrix detection directly can lead to an improvement in the entire recognition procedure for matrices.

**Table 1. Recognizable characters**

| | |
|---|---|
| Alphabet | a,b,c,d,e,f,g,h,i,j,k,l,m, n,o,p,q,r,s,t,u,v,w,x,y,z, A,B,C,D,E,F,G,H,I,J,K,L, M,N,P,Q,R,S,T,U,V,W,Y,Z |
| Greek Alphabet | $\alpha,\beta,\sigma,\delta,\varepsilon,\omega,\pi$ |
| Numeral | 0,1,2,3,4,5,6,7,8,9 |
| Mathematical Symbol | -,+,<,>,≤,≥,≠,$\sqrt{}$,$\int$,$\sum$,$\prod$, ∞,(,),{,},sin,cos,tan,lim,log |

Figure 4 shows results of previewing LaTeX source code. As the figure clearly shows, this facility makes the system intelligible for users. Almost all of the mathematical expressions that this system accepts can be expressed. However, this facility cannot display a matrix structure at present.

## 6. Conclusions

We presented a system for the real-time recognition of handwritten mathematical formulas in this paper. This system recognizes mathematical expressions using the advantages of an on-line input method that is performed stroke-

**Table 2. Experimental results for general mathematical structures**

| | Character Recognition | Structure Recognition |
|---|---|---|
| Formula (1) | 74% (1158/1560) | 94% (1352/1440) |
| Formula (2) | 86% (1551/1800) | 91% (1416/1560) |
| Formula (3) | 81% (2130/2640) | 93% (2337/2520) |
| Formula (4) | 76% (1468/1920) | 91% (1643/1800) |
| Average/Total | 80% (6307/7920) | 92% (6748/7320) |

**Table 3. Results for matrix structure recognition**

| | |
|---|---|
| Matrix (5) | 70% (53/75) |
| Matrix (6) | 68% (51/75) |
| Average/Total | 69% (104/150) |

by-stroke. Through evaluation experiments, the system proved its feasibility in handling mathematical formulas on a computer. The system has the ability to recognize the general structures and functions of inputted mathematical formulas. In future work, we plan on improving the accuracies of the individual components of this system.

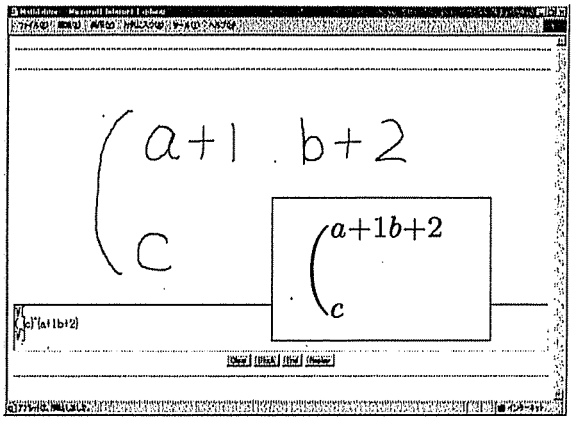


**Figure 6. Failure of matrix detection. Outputted image is shown in the box.**

## References


[1] Andreas Kosmala and Gerhard Rigoll, "On-Line Handwritten Formula Recognition Using Statistical Methods," In Proc. Int. Conference on Pattern Recognition (ICPR), pp. 1306-1308 (August 1998).

[2] Andreas Kosmala, Gerhard Rigoll, Stephane Lavirotte and Loic Pottier, "On-Line Handwritten Formula Recognition using Hidden Markov Models and Context Dependent Graph Grammers," In Proc. Fifth Int. Conference on Document Analysis and Recognition (ICDAR), pp. 107-110 (September 1999).

[3] Ryoji Fukuda, Sou I, Fumikazu Tamari, Xie Ming and Masakazu Suzuki, " A Technique of Mathematical Expression Structure Analysis for the Handwritten Input System," In Proc. Fifth Int. Conference on Document Analysis and Recognition (ICDAR), pp. 131-134 (September 1999).

[4] Andreas Kosmala, Gerhard Rigoll and Anja Brakensiek, "On-Line Handwritten Formula Recognition with Integrated Correction Recognition and Execution," In Proc. Int. Conference on Pattern Recognition (ICPR), 2, pp. 590-593 (September 2000).

[5] Atsushi Murase, Takashi Satou and Masaki Nakagawa, "Prototyping of METAH, A Recognition system for on-line Handwritten Mathematical Expressions," Information Processing Society of Japan SIG Notes, **93-HI-48**, pp. 25-32 (May 1993).

[6] Takahiro Suzuki, Shiro Aoshima, Kensaku Mori and Yasuhito Suenaga, "New System for the Real-time Recognition of Handwritten Mathematical Formulas," In Proc. Int. Conference on Pattern Recognition (ICPR), 4, pp. 515-518 (September 2000).

[7] Dorothea Blostein, Ann Grbavec, "Recognition of Mathematical Notation," In H. Bunke and P.S.P Wang, editors, Handbook of Character Recognition and Document Image Analysis, World Scientific Publishing, chapter 21, pp. 557-582, 1997.

[8] Zi-Xiong Wang and Claudie Faure, "Structural Analysis of Handwritten Mathematical Expressions," In Proc. Int. Conference on Pattern Recognition (ICPR), pp. 32-34 (1988).

[9] Masayuki Okamoto and Hiroyuki Higashi, "Mathematical Expression Recognition by the Layout of Symbols," Transactions of Institute of Electronics, Information and Communication Engineers (D-II), **J78-D-II**, 3, pp. 474-482 (March 1995).

[10] Yannis A. Dimitriadis and Juan Lopez Coronado, "Towards an Art Based Mathematical Editor That Uses On-line Handwritten Symbol Recognition," Pattern Recognition, **28**, 6, pp. 807-822 (June 1995).