On-Line Recognition of Handwritten Mathematical Expressions Based on Stroke-Based Stochastic Context-Free Grammar

Ryo Yamamoto, Shinji Sako, Takuya Nishimoto, Shigeki Sagayama

On-Line Recognition of Handwritten Mathematical Expressions Based on Stroke-Based Stochastic Context-Free Grammar

Ryo Yamamoto, Shinji Sako, Takuya Nishimoto, Shigeki Sagayama
The University of Tokyo.
{ yamaryo, sako, nishi, sagayama } @hil.t.u-tokyo.ac.jp

Abstract

In this paper, we propose a new framework for on-line handwritten mathematical expression recognition. In this approach, we consider handwritten mathematical expressions as the output of stroke generation processes based on a stochastic context-free grammar which generates handwritten expressions stochastically. We estimate the most likely expression candidate derived from the grammar, rather than solving one by one the three major problems in mathematical expression recognition: symbol segmentation/recognition, 2D structure recognition, and expression syntax analysis. With this method, we can simultaneously recognize the symbols and structure of an expression within the grammatical constraint. Experiments revealed that this simultaneous estimation decreases errors in symbol segmentation and recognition, and that these errors are reduced as grammatical restriction is strengthened.

Keywords: Mathematical Expression Recognition, Character Recognition, On-line, Handwriting, Stochastic Model, Stochastic Context-Free Grammar

1 Introduction

There are several ways to input mathematical expressions into a computer. The most common ones are to make use of special strings such as TEX, C, or Matlab, or to use a mathematical editor such as the one embedded in MS-Word. But these methods require learning of the language or difficult manipulations. Being able to input mathematical expressions by hand with a pen tablet, in the same way as we write them on paper, would be more intuitive and very useful in the process of writing scientific papers or as an input method for calculation softwares. Recognition of on-line mathematical expressions is the key problem to solve toward the achievement of this goal.

Intensive research has already been conducted on mathematical expression recognition [2], and most of the existing systems solve this problem in three steps: a symbol segmentation/recognition step, a 2D structure recognition step, and an expression syntax analysis step. In the symbol segmentation/recognition step, the input stroke sequence is segmented and each segment is recognized as a mathematical symbol. This problem can be treated as the recognition of a character sequence. Existing character recognition methods are used here. In the 2D structure recognition step, the 2D structure among recognized symbols, for example the fact that a symbol is placed in the "right" or "upper right" of (in other words, on the "same or upper baseline" with regard to) another, is among recognized symbols is recognized. The 2D structure is an indispensable information in expression recognition, and approaches using a 2D grammar such as a graph grammar [4] or using rule-based analysis [3] have been proposed. In the expression syntax analysis step, the 2D structure of the expression is analyzed to output TEXor C strings. Methods proposed so far consist in transforming the 2D tree into a mathematical expression grammar tree [6], TEXstring-based parsing [1], and so on. In most systems using rule-based structure analysis implicitly exists a mathematical expression grammar.

Symbol recognition is not easy in mathematical expressions because there are many kinds of symbols other than alphabets: Arabic numerals, Greek symbols, parentheses, operators, fraction lines, root signs, commas, dots, etc. These symbols are very simple in shape and appear in many different sizes, making their recognition difficult.

The recognition of the 2D structure is also complex. Spatial relationships between symbols have fluctuations because they are written by hand. So spatial relationships cannot easily be translated into logical structures. What is more, mathematical symbols vary in their shape and size, which makes the problem even more difficult.

As stated above, most of the existing methods recognize symbols first, and then analyse the 2D and syntactic structure. But it is natural to think that when a person sees a mathematical expression, he/she recognizes the symbols using not only their shape but also the whole 2D and syntactic structure of the expression, and using such contextual information enables robust recognition of symbols.

In the light of this, we handle mathematical expression recognition as a simultaneous optimization of symbol segmentation, symbol recognition, and 2D structure recognition under the restriction of a mathematical expression grammar. We model handwritten mathematical expressions with a stochastic context-free grammar and formulate the recognition problem as a search problem of the most likely mathematical expression candidate, which can be solved using the CYK algorithm.

We also propose a new 2D structure model for mathematical expressions using the new concept of Hidden Writing Area (HWA) that we introduce. We model the handwriting of a mathematical expression as the process of stochastically placing each stroke into an imaginary box (HWA) which position is itself stochastically determined according to the syntactic structure of the expression. During the recognition, the probability distribution of the HWA is calculated for each stroke candidate, and we calculate the probabilities that the HWAs of each stroke fit the structure derived from the syntactic structure of the expression. This model enables symbol-independent structure recognition and simple designing of the mathematical expression grammar.

In section 2, we explain the details of the proposed method, and in section 3 we present its evaluation through recognition experiments.

2 Proposed method

2.1 Context dependency of symbol recognition

Recognition of symbols in mathematical expression recognition is closely related to the context, i.e. 2D and grammatical structure, of the expression.

For example Figure 1(a) and Figure 1(b) show that symbol segmentation and symbol recognition can change depending on the context even if the shape of the symbol is the same. Symbol recognition is thus fundamentally an ambiguous problem, and a human disambiguates it using the whole grammatical structure of the expression. So evaluation of the whole grammatical structure can lead to more robust symbol recognition, but this kind of estimation cannot be done in most of the existing recognition systems, since they recognize first the symbols and then the structure. Our first goal is thus to solve the ambiguity of symbol segmentation and recognition using the grammatical structure of the whole expression.

In the same way, Figure 1(c) gives an example of the fact that 2D structure recognition is also dependent on grammatical structure. Our second goal is thus to solve this 2D structure ambiguity using the grammatical structure.

Grammatical information is sometimes not sufficient to solve the ambiguity. In such situations, a human possibly estimates the shape of the symbols and the whole structure of expression as a whole, simultaneously (Figure 1(d)). This simultaneous recognition of symbols and structure cannot be done in existing systems as they separate symbol recognition step and structure analysis step. So we also look for a recognition method which can estimate symbols and structure simultaneously when the grammatical structure information is not sufficient.

2.2 Handwritten expression grammar

From this viewpoint, we extended mathematical expression grammars for handwritten expression. Expression grammars can be written in the form of context-free grammar (CFG), and the compilers of TEX, C, Matlab, etc.

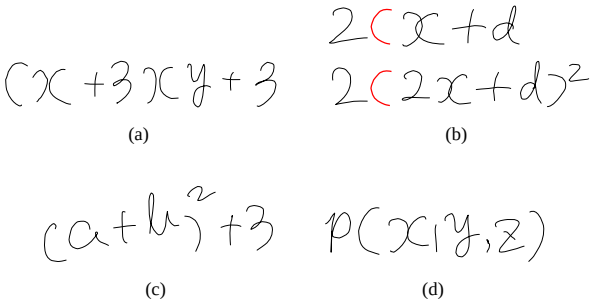

Figure 1. (a) Ambiguity in symbol segmentation and recognition can be solved using expression grammar. The first stroke on the left should be recognized as "c", not "$($", while the strokes between "3" and "y" should be recognized as "x", not "$)($". (b) Symbol recognition changes according to the context even if the symbol's shape is the same. The second stroke should be recognized as "c" in the upper expression, "$($" in the lower one. (c) Ambiguity in 2D positional relationship between symbols can be solved using the grammar. The logical relationship between "b" and "$)$" is the "right" relationship, even though it would be mis-recognized as "lower right" without the grammar. (d) Simultaneous estimation of symbols and 2D structure seems necessary. To decide which of "$P(x|y,z)$" or "$P(x_1 y,z)$" this expression is, we estimate how the vertical line is like "1" or "|", and how the positional relationship between "x" and the line is like "right" or "lower right", then we recognize the expression as a whole.

use a CFG parser to parse expressions written in their own language.

A handwritten expression grammar can be written as shown in Table 1, taking into account the writing order and the 2D structure of the symbols. It also includes generation rules of handwritten strokes (rule No.22 to No.25) to generate directly handwritten strokes, since handwritten expressions are sampled as sequences of handwritten strokes (sequences of pen trajectories devided by pen-up/down), not as sequences of symbols. For each symbol which stroke count is 2 or more, we build stroke generation rules. We treat structure in the expression and structure in each symbol in the same way.

Though the mathematical expression grammar is itself deterministic, handwritten structure and shapes of symbols are stochastic. This means, for example, that when rule No.2 in Table 1 is applied to one "expression" element and "expression" and "symbol" are generated, the positional relationship between the two is stochastically determined. So when the positional relationship between the "expression" and "symbol" elements is given, we can compute the likelihood (which we call "structure likelihood") of the fact that these elements have been stochastically generated using rule No.2. Generation rule p with structural condition s is expressed in the form

$p = \langle A \rightarrow BC, s \rangle$, where A, B, C are non-terminal symbols (e.g. "function", "symbol", etc.) of the mathematical expression grammar. Structure likelihood is then $P(B, C|A, s)$ and is modeled as explained in 2.5.

In the same way, when a handwritten stroke is generated from element "a" by application of rule No.25, the shape of the handwritten stroke is determined stochastically. So, when the shape of a handwritten stroke is given, we can compute the likelihood (called "stroke likelihood") of each of the stochastic generation rules for that stroke. Handwritten stroke generation rule q is expressed in the form $q = \langle A \rightarrow \alpha \rangle$, where A is a non-terminal symbol and α a terminal symbol (= handwritten stroke) of the expression grammar. Stroke likelihood is then $P(\alpha|A)$ and can be computed using model-based character recognition methods [5].

We can say that this likelihood is the probability of application of the corresponding generation rule. We thus modeled handwritten mathematical expressions with a stochastic context-free grammar.

2.3 Formulation of the expression recognition

The mathematical expression recognition problem is then formulated as the search problem of the most likely expression hypothesis for the input handwritten strokes under the grammar, that is to find X_0 such that

$$\begin{aligned} X_0 &= \arg\max_{X \in E_X} P(X|H) \\ &= \arg\max_{X \in E_X} P(H|X)P(X) \\ &\simeq \arg\max_{X \in E_X} P(H|X). \end{aligned} \tag{1}$$

Here $P(H|X)$ is the probability that handwritten expression H is generated from expression hypothesis X, and $P(X)$ is the prior probability of X. In this paper we suppose equal the prior probability of all expression hypotheses. Expression hypothesis X is a derivation of H by the grammar G, and X can be represented as $X = \{p_1, p_2, \ldots, p_N, q_1, q_2, \ldots, q_M\}$, where $p_n = \langle A_n \rightarrow B_n C_n, s_n \rangle$ is a generation rule with structural condition, $q_m = \langle A_m \rightarrow \alpha_m \rangle$ a handwritten stroke generation rule and N, M are the number of these rules. Then Equation 1 becomes:

$$X_0 = \arg\max_{X \in E_X} \prod_{n=1}^{N} P(p_n) \prod_{m=1}^{M} P(q_m). \tag{2}$$

This shows that mathematical expression recognition can be formulated as the search for an expression that is derived from the expression grammar and that maximizes the product of all stroke likelihoods and structure likelihoods. Since this method searches result within the expression grammar, it can resolve, thanks to the grammar, the ambiguity in symbol segmentation/recognition, and structure recognition, and by searching the most likely hypothesis, it can evaluate symbols and structure as a whole, in other words, it can resolve the ambiguity in symbol recognition thanks to the structure.

Table 1. Example of a basic handwritten mathematical expression grammar. Rules marked with * cannot be applied iteratively. ** means that the writing order of the 2 symbols can change and that the rules with permutation of the order are included. Abbreviated names of expression elements are as follows: EXP: expression, SYM: symbol, FUNC: function, LINE: fraction line, DLINE: fraction line with denominator, NLINE: fraction line with numerator, ROOT: root sign, ACC: accent, RPAR: right parenthesis, LPAR: left parenthesis, XRPAR: expression with right parenthesis, XLPAR: expression with left parenthesis, HS: handwritten stroke.

No.	Generation rule			Logical Relationship	Notes
1	EXP	→	SYM		
2	EXP	→	EXP SYM	Right	
3	SYM	→	SYM EXP	Upper Right	*
4	SYM	→	SYM EXP	Lower Right	*
5	FUNC	→	FUNC EXP	Upper	*
6	FUNC	→	FUNC EXP	Lower	*
7	DLINE	→	LINE EXP	Lower	**
8	NLINE	→	LINE EXP	Upper	**
9	SYM	→	DLINE EXP	Upper	**
10	SYM	→	NLINE EXP	Lower	**
11	SYM	→	ROOT EXP	Inside	**
12	SYM	→	ACC EXP	Accent	* **
13	SYM	→	ACC SYM	Accent	* **
14	XRPAR	→	EXP RPAR	Right	**
15	XLPAR	→	EXP LPAR	Left	**
16	SYM	→	XRPAR LPAR	Left	**
17	SYM	→	XLPAR RPAR	Right	**
18	SYM	→	$a \mid b \mid c \mid \cdots$		
19	FUNC	→	$\lim \mid \sum \mid \max \mid \cdots$		
20	LPAR	→	$(\mid [\mid \{ \mid \cdots$		
21	RPAR	→	$) \mid] \mid \} \mid \cdots$		
22	f	→	$f_1\ f_2$	Same Symbol	
23	x	→	$x_1\ x_2$	Same Symbol	
24			⋮		
25	a	→	HS		
26	f_1	→	HS		
27	FranLine	→	HS		
28			⋮		

2.4 Search using the CYK algorithm

The search problem of the most likely derivation by stochastic context-free grammar can be solved by the CYK algorithm. We use this algorithm to find the most likely expression candidate for the input handwritten expression. In this section we explain the recognition algorithm on an example shown in Figure 2.

The algorithm is the following:

1. For each input handwritten stroke, stroke likelihood of each stroke candidate is calculated. This calculation is the same as the likelihood calculation in isolated character recognition. All the stroke candidates (or the n best candidates, in practice) with their likelihood for the ith handwritten stroke are written in the ith diagonal element of the CYK triangle matrix. In this example, the first stroke of the input expression can be "$)$" and the stroke likelihood for this candidate is 0.2. The stroke can also

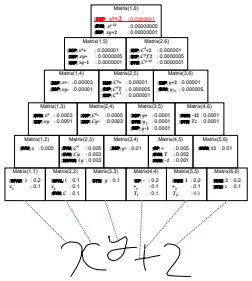

Figure 2. Example of a search for most likely expression candidate using the CYK algorithm.

be the first stroke of "x" ("$x_{[1]}$") and the likelihood is 0.1.

2. In the $(i, i+1)$ element of the matrix, write all the expression element candidates which can derive from the i-th and $i+1$-th strokes. In our example, the first and second strokes can be derived with the rule "$\langle x \to x_{[1]}x_{[2]}, s_{SameSymbol}\rangle$" from "$x$". Then we calculate the structure likelihood. It is 0.5 here. The candidate"x" and the product of stroke and structure likelihoods $0.1 \times 0.1 \times 0.5 = 0.005$ is written in $(1,2)$ element of the matrix. Note that the ")" and "(" candidates for first and second strokes cannot be derived with any of the expression rules shown in Table 1 and no corresponding candidate is written.

3. In the $(i, i+2)$ element, all the candidates for $i, i+1, i+2$-th strokes are written in a similar way. First we write candidates which derive from a candidate in (i,i) and a candidate in $(i+1, i+2)$, next from a candidate in $(i,i+1)$ and a candidate in $(i+2, i+2)$. For example, in $(1,3)$ of the matrix, "x^y" can be derived from "x" in $(1,2)$ and "y" in $(3,3)$. The structure likelihood that "y" is in the "upper right" of "x" is 0.6 here. Total likelihood of "x^y" is the product of the corresponding likelihoods ($0.005 \times 0.1 \times 0.6$).

4. In the same way, in the $(i, i+k)$ element, we write all the candidates for $i, i+1, \ldots, i+k$-th strokes. We find candidates which are derived from a candidate in $(i, i+j)$ and a candidate in $(i+j+1, i+k)$ for $k = 0, 2, \ldots, k-1$, calculate the structure likelihood and write the product of the likelihoods for each candidate.

5. Finally, the most likely "EXP" candidate in the $(1,n)$ element of the CYK matrix is the recognition result.

2.5 Structure model using Hidden Writing Area

To estimate the logical relationships between the expression elements, many existing methods use their bounding boxes [6] [3]. But since mathematical symbols vary in size and shape, the bounding boxes are not always sufficient to estimate the logical relationships [2]. In [6], a method using different relationship evaluation functions depending on symbol category is proposed, but it is reported that the accuracy is not good enough because handwritten expression have fluctuations and handwriting style varies from person to person. Moreover, expressions include some irregular shape symbols such as dot, comma, hat, etc. and this makes the problem more complicated. To deal with such variance, statistical learning from a large amount of data can be a good solution. In the following we propose a stochastic structure model which can be trained statistically by data.

Behind every expression elements, we assume that there is a hidden box which is arranged according only to the syntactic structure of the expression, independent of symbols inside. We call that box Hidden Writing Area. A HWA is represented by 4 parameters as shown in Figure 3(b). The probability that two expression elements B, C are derived from another element A by the generation rule $p = \langle A \to BC, s\rangle$, is determined according to the corresponding HWAs h_A, h_B, h_C, and s:

$$P(B, C|A, s) = F_s(h_A, h_B, h_C). \tag{3}$$

The probability functions F_s for each s are defined as shown in Figure 3(a). For each logical relationship s, the relationship between HWAs is written in the simultaneous equations of h_A, h_B, h_C. For some relationships like "upper right", positional freedom is modeled with random variables $v^1_{UpperRight}, v^2_{UpperRight}$ included in these equations.

Each handwritten stroke α is generated stochastically in its corresponding HWA h_A. Here, while stroke "d" tends to be generated slightly shifted toward the top of its HWA, stroke "y" is shifted towards the bottom of its HWA. This positional tendency is modeled using random variable d_A for each stroke A which represent the lag between the HWA and the bounding box. The probability that a handwritten stroke α is derived from the stroke A is determined according to the bounding box of the handwritten stroke r_α, the stroke shape feature t_α, h_A, and A:

$$\begin{aligned} P(\alpha|A) &= P(r_\alpha|h_A, A)P(t_\alpha|A) \\ &= G_A(r_\alpha, h_A)P(t_\alpha|A). \end{aligned} \tag{4}$$

Here $P(t_\alpha|A)$ is the stroke likelihood, which can be modeled and calculated with some isolated character recognition methods. The probability function G_A is determined in Figure 3(c). For each stroke A, the relationship between HWA h_A and the bounding box r_α is written in the simultaneous equations including the lag variable d_A.

If we denote the bounding boxes of the input strokes as $\{r_1, r_2, \ldots, r_M\}$, the shape feature of them as $\{t_1, t_2, \ldots, t_M\}$, the likelihood of an expression candidate $X = \{p_1, p_2, \ldots, p_N, q_1, q_2, \ldots, q_M\}$ (where $p_n = \langle A_n \to$

$B_n\ C_n\ , s_n\rangle$ and $q_m = \langle A_m \rightarrow \alpha_m\rangle$) is given by:

$$\begin{aligned}
&\prod_{n=1}^{N} P(p_n) \prod_{m=1}^{M} P(q_n) \\
&\quad = \prod_{n=1}^{N} P(B_n, C_n | A_n, s_n) \prod_{m=1}^{M} (\alpha_m | A_m) \qquad (5) \\
&\quad = \prod_{n=1}^{N} F_{s_n}(h_{A_n}, h_{B_n}, h_{C_n}) \prod_{m=1}^{M} G_{A_m}(h_{A_m}, r_{\alpha_m}) P(t_{\alpha_m} | A).
\end{aligned}$$

The maximum likelyhood candidate can be estimated as described in 2.4.

The model parameters d_A, v_s for each A, s can be trained iteratively as follows. We use as training data handwritten expressions which are tagged with their correct syntactic structure. After setting initial values for the parameters, we first estimate the most likely HWAs of every expression element for each expression in the way described above, and then, using these HWAs, we update the model parameters d_A, v_s. These two operations are repeated iteratively.

We performed recognition experiments on the expression structure to estimate this structure model. This corresponds to the expression recognition under the condition that stroke recognition has already accurately been done. Training data consists in 7 expressions written about 40 times each by one writer, for a total of 256 expressions. Evaluation data consists in 8 expressions from IEEE articles written 10 times each by same writer as the training data, for a total of 80 expressions. 5 expressions are common to the evaluation and training data. The reason is that the method we propose requires every symbol in the target domain to appear in the training expression data because the lag variable corresponding to each stroke can only be learned from expression training data as described above, not from isolated character data. Thus, the symbol domain of the training data must cover that of the evaluation data. We shared some expressions because it is hard to design training data to cover all symbols of evaluation data. For the same reason, the symbol domain of this experiment is limited to that of the training data (52 symbols, about the same as the number of symbols used in evaluation data).

Error rate on the baseline level E_{base} was 2.53% in shared (closed), 5.07% in unshared (open) set. Although training data was quite limited, the proposed structure model worked well. Mis-recognition typically occurred for slanted expressions recognized as subscripts or superscripts. Introducing a random variable v_s into "right" relationship model could reduce such kind of errors. Examples of the most likely HWAs are shown in Figure 4. They are estimated indeed as we expected.

3 Experiment

We did expression recognition experiments to see how symbol recognition errors decrease using this method. The evaluation and training data were the same as in 2.5. We used 10-state left-to-right HMMs for stroke models,

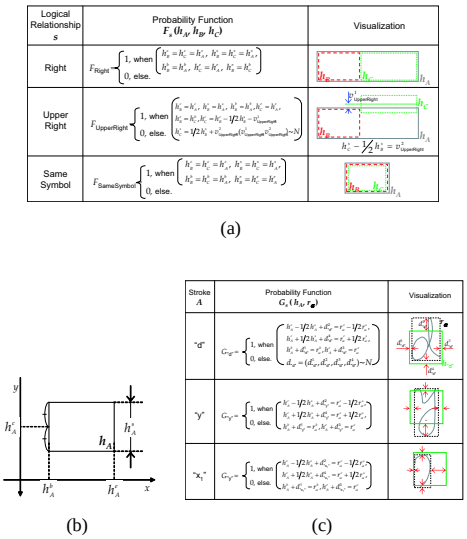

Figure 3. (a) Examples of the probability functions F_s. (b) Parameters representing HWA. (c) Examples of the probability functions G_A.

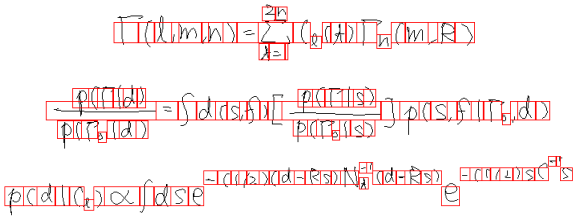

Figure 4. Examples of the most likely HWA for some expressions.

time sequence of 4-dimensional vector of x-y coordinate and its temporal subtraction for the feature vector. These models were trained with the same training data. We did experiments under 4 different mathematical grammar conditions:

1. (A. NoGram) Using a structure-ignoring grammar to recognize only symbols. This grammar only estimates the 2D structure within symbols, but not between symbols, and the structure likelihood between symbols is constant. The symbol recognition rate not using 2D and syntactical structure was evaluated and used as a baseline.

2. (B. Gram1) Using a smaller constraint grammar. Just like TEXgrammar, any symbol sequence is accepted. The grammar is the one in Table 1 with rules No.5-6, 14-17 removed.

3. (C. Gram2) Using the grammar shown in Table 1.

4. (D. Gram3) Using a more complex grammar than Table 1. Rules about "term", "operator", etc. are added.

Table 2. Experimental results.

Error rate[%]		A. NoGram	B. Gram1	C. Gram2	D. Gram3
open	E_{seg}	13.43	4.10	2.20	2.02
	E_{sym}	28.01	24.69	23.58	20.97
	E_{base}		16.24	14.93	8.92
closed	E_{seg}	16.67	1.39	0.65	0.58
	E_{sym}	26.81	12.89	8.24	7.14
	E_{base}		4.11	4.22	4.15

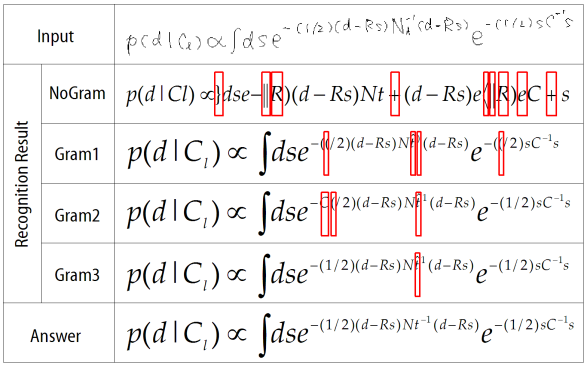

Figure 5. Examples of recognition results under each condition. Errors in symbol recognition are marked.

We compare the results with the symbol segmentation error rate E_{seg}, the symbol recognition error rate E_{sym} and the baseline error rate E_{base}. The results are shown in Table 2. We can see that for most of the expressions, symbol segmentation and recognition error decreases along with the strengthening of the grammatical constraint, and that errors in structure recognition can decrease along with the increase of grammatical constraint.

Examples of recognition results under each condition are shown in Figure **??**. Comparing "NoGram" with "Gram1", one can see that symbol errors decrease when simultaneous recognition of symbols and structure is performed. Note that these errors are not corrected by syntactic constraint as the grammar used in "Gram1" condition has such a small constraint that it cannot reject expressions like "$\cdots \propto \} dse^{-\|R \cdots}$", but because we also take into account the structure likelyhood, which changed the ranking of the candidates.

4 Conclusion

In the light of the fact that ambiguity in symbols and structure recognition can be solved by their simultaneous estimation and by the use of an expression grammar, we viewed the recognition problem as a simultaneous optimization of symbols and structures under the constraint of an expression grammar. While classical mathematical expression grammars are designed to generate strings representing expressions, we extended the expression grammar to model the stochastic generation of the 2D structure and the handwritten strokes. The recognition problem becomes then equivalent to the search for the most likely derivation from the input and it can be solved efficiently with the CYK algorithm. This method can principally reduce errors by using simultaneous estimation of symbols and structure and an expression grammar, which we confirmed through experiments.

Evaluation of this method on a larger database is the most important issue ahead. Other problems to be solved include reduction of the computation costs, design of an optimal expression grammar, and modeling of the prior probability of expression candidates.

References

[1] K. -F. Chan and D. -Y. Yeung. An Efficient Syntactic Approach to Structural Analysis of On-line Handwritten Mathematical Expressions. *Pattern Recognit.*, 33:375–384, 2000.

[2] K. -F. Chan and D. -Y. Yeung. Mathematical Expression Recognition: A Survey. *Int. J. Document Anal. Recognit.*, 3(1):3–15, Aug. 2000.

[3] U. Garain and B. B. Chaudhuri. Recognition of Online Handwritten Mathematical Expressions. *IEEE Trans. Sys. Man Cybern. Part B:Cybern.*, 34(6):2366–2376, Dec. 2004.

[4] A. Kosmala, G. Rigoll, S. Lavirotte, and L. Pottier. On-Line Handwritten Formula Recognition using Hidden Markov Models and Context Dependent Graph Grammars. In *Proc. Int. Conf. Document Analysis and Recognition (ICDAR)*, pages 107–110, Sep. 1999.

[5] R. Plamondon and S. N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(1):63–84, Jan. 2000.

[6] R. Zanibbi, D. Blostein, and J. R. Cordy. Recognizing Mathematical Expressions Using Tree Transform. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(11):1–13, Nov. 2002.