Mathematical formula recognition using virtual link network

Yuko Eto
Toshiba Corporation e-Solution Company
Suehiro-cho 2-9, Ome, Tokyo, 198-8710, Japan
yuko.eto@toshiba.co.jp

Masakazu Suzuki
Faculty of Mathematics, Kyushu University
Hakozaki 6-10-1, Higashi-ku, Fukuoka, 812-8581 Japan
suzuki@math.kyushu-u.ac.jp

Abstract

In this paper, we propose a new method of recognizing mathematical formulae. The method is robust against the recognition errors of characters and the variation of the printing styles of the documents. The outline is as follows: we first construct a network with vertices representing the characters (symbols), linked each other by several edges with labels and costs representing the possible relations of the pair of characters. The network has multiple edges with different labels and costs representing the ambiguity of the decision of the relation of character pairs.

Then, we output the spanning tree of the network with minimum cost which corresponds to the recognition result of the structure of the mathematical formula, using not only the local costs initially attached to the network but the costs reflecting global structure of the formula. The advantage of this method is that local errors of the recognition are recovered automatically by the total cost of the recognition tree.

1 Introduction

In view of digitizing scientific documents, the investigation of practical and robust methods for recognizing mathematical formulae is currently a subject of growing interest.

Several algorithms for recognizing mathematical formulae have been reported in literature (see [1]). Among them, the methods proposed by [2], [3] and [4] have the advantage of realistic processing speed. However, in their methods, the structure analysis proceeds deciding the relation of each pair of adjacent characters (symbols) in a sequential order fixed in advance, and a local error of the decision leads sometimes to a fatal error in the global structure analysis.

The errors in the decision of adjacent character pair relations are caused, in most cases, by the error of the character recognition for similar characters with different sizes, and also by the differences of the relative sizes and positions, depending on documents, between subscripts/superscripts and their parent characters, or between alphanumeric characters and other symbols such as parentheses, mathematical operator symbols, etc.

In this paper, we propose a new method of recognizing mathematical formulae, robust against the recognition errors of characters and the variation of the printing styles of the documents, keeping processing speed effectively usable in OCR softwares.

The outline of our method is as follows: we first construct a network with vertices representing the characters (symbols), linked each other by several edges with parent-child orientation, label and cost. The label attached to each edge represents a possible relation of the character pair linked by the edge: subscript, superscript or on a same line, etc. The link is virtual one. Each node has several candidates of character recognition result, and each pair of adjacent nodes may have multiple edges with different labels and costs representing the possible relations for differnet pairs of parent candidate and child candidate.

Then, we try to find the spanning tree of the network corresponding exactly to the recognition result of the mathematical formula structure. The search of the appropriate spanning tree proceeds in two steps. In the first step, we generate *admissible* spanning trees of the network with lowest costs, up to some pre-fixed number of candidates, where a spanning tree of the network is called admissible, if it has no *apparent contradiction* to represent a mathematical formula (Section 4). In the second step, we reevaluate the costs of the obtained candidate trees in the first step, using not only the local costs initially attached to the network but the costs reflecting global structure of the formula, and output the spanning tree with the minimum reevaluated cost as the recognition result of the mathematical formula.

The advantage of this method is that local errors of the recognition are recovered automatically by the total cost of the recognition tree. We applied the beam search method which is often used in the dynamic programming in order to reduce the processing time of the first step. To determine the initial cost of the network, we used the distribution maps of relative sizes and positions for each relation types of parent-child links, obtained by the measurement of the images samples collected from 32 different mathematical journals (Section 2).

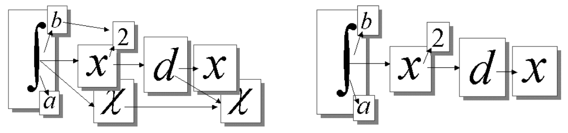


Figure 1. Network and the correct candidate of spanning tree

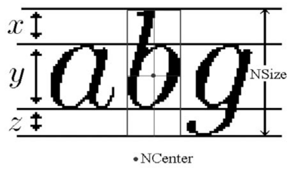


Figure 2. xyz **ratio**

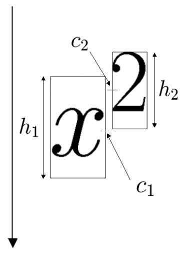


Figure 3. Definition of H **and** D

2 Normalized size, normalized center, and relative position plots

Following [2], we make use of the notions of *normalized size* and *normalized center* of each character, which play an important role in the estimation of the possibility of the relations: on a same line, subscript or superscript, etc., for each pair of adjacent characters.

As it is illustrated in Figure 2, the normalized size is the total size (NSize) including the ascender part x and the descender part z, and the normalized center is the center (NCenter) of the bounding rectangle including the ascender part and the descender part. They can be evaluated from the bounding rectangle of a character image using the ratio $x : y : z$. The average of this ratio measured using the above mentioned 32 documents was $28 : 51 : 21$.

As for the normalized size of operators and symbols, we define it to be the maximum of the height and the width of their bounding rectangles.

Using the normalized size and the normalized center, we can represent the relation of a pair of characters by a point in a normalized coordinate plane. Given a pair of characters (parent and child), let h_1 (resp. h_2) be the normalized size of the parent (resp. child) character and c_1 (resp. c_2) be the vertical coordinate of the normalized center of the parent (resp. child) character(see Figure 3). Setting

$$H = \frac{h_1}{h_2} \times 1000, \qquad D = \frac{c_1 - c_2}{h_1} \times 1000,$$

the relative size and position of the character pair is represented by the *plot* of the point (H, D) in a coordinate plane, which we call the normalized coordinate plane. For example, the plot of relative position for the pair of characters on the same line is ideally $(1000, 0)$.

Figure 4 represents the distributions of the plots of relative positions of the sample character pairs on the same line, subscript position and superscript position, collected from the 32 mathematical documents mentioned above, classified into four groups of character types, where all the alphanumeric characters and Greek characters are included in the class of type “Alphabet”. Figure 4 shows that, in most cases, it is possible to distinguish the relation of the pair of characters by calculating H and D, if the recognition results of the characters are both correct.

In each distribution map, we defined the *strict area* and the *wide area* enclosing the plots in high density bounded by polygons, and use them to calculate the cost of the link in the virtual link network, which will be described in the next section.

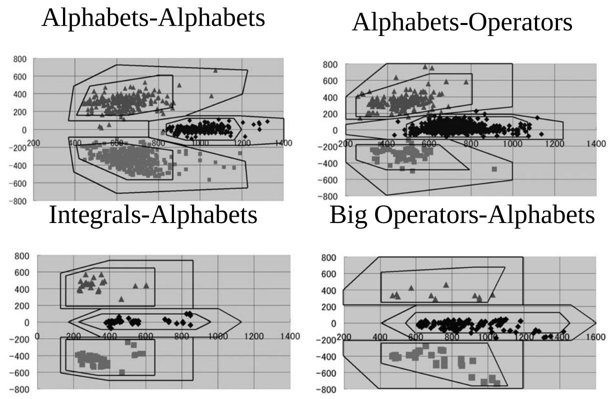


Figure 4. Distribution map of relative position plots

3 Virtual link network

In this section, we describe the process to construct the *virtual link network*. Each node of the network represents a character (symbol) in the image. It has the bounding rectangle coordinates and the candidates of the character recognition result. The network is multi-linked and oriented. Each edge represents a choice of the candidates in the parent node and in the child node, as well as the choice of the relation (Link label) of the chosen candidates.

3.1 Selection of the candidates in nodes

Our character recognition engine returns a maximum of 10 candidates. Each candidate has the value of likeness between 0 and 100. We make use of the candidates which have different normalized sizes from the 1st candidate to the 5th candidate with the likeness greater than 50.

To each candidate of the character recognition, we attach its normalized size defined in 2. Alphanumeric characters are classified into four types by their normalized size types (e.g. "$abgf$"). There are several similar character pairs having different normalized size types, which are often or sometimes mis-recognized, such as "Ss", "Cc", "$x\chi$", "$r\Gamma$", etc. If one of these *uncertain size characters* is included in the candidates, we add the pair character with different size in the candidate list. In case the character recognition result is the "reject" or the likeness of the top candidate is lower than 50, we add the candidate "REJECT" of all the four types of the normalized sizes in the candidate list.

3.2 Construction of network

The structures of mathematical formulae which we are going to recognize are described by the links with 9 types of the labels representing the relations of character pairs, listed in Table 1, where, for example, the relations of a numerator (resp. a denominator) to its fractional line are included in the case of the label "Upper" (resp. "Under"). Each edge

Label	Meaning of the label
Horizontal	C is the next character of P on a same line
R(L)SupScript	C is right (left) superscript of P
R(L)SubScript	C is right (left) subscript of P
Upper	C is in the upper area of P
Under	C is in the under area of P
InRoot	C is in the inside area of a root symbol P
InAccent	C is under an accent symbol P

Table 1. Labels of the links : Parent character (P) - Child character (C)

has the parent→child direction and represents a choice of the candidates in the parent node and child node, as well as the choice of the relation (Link label) of the chosen candidates. To each edge, a cost representing an *uncertainty* of the choice is attached.

We decide the relation of each pair of characters using distribution maps shown on Figure 4.

In our experience, the numerator/denominator area of a fraction and the interior area of a root symbol, as well as the affected area under an accent symbol (such as over line, etc.), are detected with high accuracy, getting less influence of the character recognition errors. The most delicate part of the structure analysis of mathematical formula is the distinction of the horizontal link and the superscript/subscript link, which is often disturbed by the error of the character recognitions and the variation of the printing styles. Therefore, we shall describe below mainly how to construct horizontal link edges and superscript/subscript link edges of the network.

For each pair of candidates of nodes corresponding to a pair of character rectangles to be linked by edges, we first calculate H and D described in 2. If the point (H, D) is in the inside of a *wide* polygon of Figure 4, we link the pair of nodes by an edge with the label corresponding to the polygon. If the point (H, D) is in the *strict area* limited by the smaller polygon inside, we attach a lower cost (30) to the edge, and attach a higher cost (70) otherwise.

Some operators and symbols (for example '$\in$','$\leq$','?') are rarely put at the first position of the script area. These operators and symbols rarely have characters in their script area. Therefore, we add the cost 100 to the edge if these operators and symbols are linked with script label.

4 Admissible spanning trees

We call a spanning tree of the virtual link network satisfying the following 4 conditions *admissible spanning tree*:

1. No node of the tree has more than one children with the same label,
2. Each node has unique candidate chosen by the edges linked to the node,
3. For each node K, all the nodes in the subtrees in RSubScript link and RSupScript link are situated in the left hand side of K's Horizontal child node.
4. For each node K, all the nodes in the subtrees in LSubScript link and LSupScript link are situated in the right hand side of K's parent node.

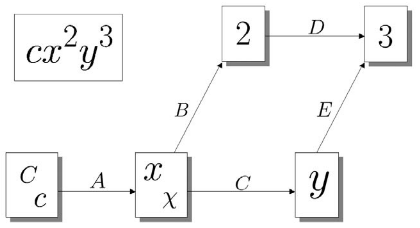
4.1 Transformation into a search path problem

Each edge of the network is represented by the 4 components:

(parent candidate, child candidate, label, cost)

Each node has a list of edges representing the relation with its parent nodes. Figure 6shows the lists of the edges attached to each node of the network in Figure 5. In order to get a spanning tree of the network, we have only to select an edge from each list of the edge candidates. Therefore, a spanning tree of the network is represented by a path connecting edges selected from the list of edges of each node. Thus these paths and the spanning trees of the network are in one-to-one correspondence.

The problem of selecting admissible spanning trees is thus transformed into a search path problem satisfying the corresponding conditions.



A	(c,x,Horizontal,10)	C	(x,y,Horizontal,10)
	(C,χ,RSupScript,50)		(χ,y,Horizontal,100)
	(c,χ,Horizontal,100)	D	(2,3,Horizontal,10)
	(C,x,Horizontal,100)	E	(y,3,RSupScript,10)
B	(x,2,RSupScript,10)		
	(χ,2,RSupScript,50)		

Figure 5. Network

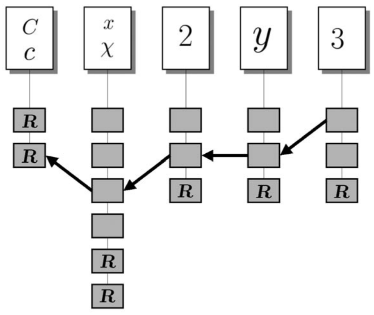


Figure 6. Search for spanning trees

(In Figure 6, the edges $\boxed{\text{R}}$ are the links from the root of the tree to the candidates in the node.)

4.2 Algorithm of listing admissible spanning trees of low costs

Given the candidate number S, we show below the algorithm to search for a maximum of S admissible spanning trees, namely "paths", of lower costs, where the cost of a tree is the sum of the link costs of the edges in the tree.

By limiting the number of candidates to S in each step, in the ascending order of the cost, it is possible to prevent the increase of the search space of the combinatorial number order.

Preparation Let N be the number of nodes in the network. Then put nodes in the ascending order of x coordinate of the top left point of the bounding rectangle. Let K_i be the i-th node $(i = 1, ..., n)$, and each K_i has a list of edges, $[l_{i1}, ..., l_{im_i}]$, where each m_i means the number of edges whose children are K_i.

Algorithm

Step 1 Put a maximum of S subtrees including each edge of K_N in ascending order of the cost in N-TreeList. Put $N = i - 1$.

Step 2 If $i = 0$, output 1-TreeList and come to an end. Otherwise, go to Step 3.

Step 3 For all pairs of the subtree T in the $i + 1$-TreeList and the edge l_{ij} in the list of K_i, check out if they satisfy the following conditions.

- · The parent and child candidates of the edge l_{ij} are not contradictory to the candidates included subtree T.
- · There is no edge which has the same label and the same parent node in the subtree T.
- · Let P be the parent node of the edge l_{ij}. The x coordinate of the normalized center of the P's horizontal child node is located in the right hand side of the left terminal point of the P's script area[1].

Step 4 Put a maximum of S subtrees satisfying the conditions of Step 3 in ascending order of the cost in i-TreeList. Put $i \to i - 1$ then go to Step 2.

5 Reevaluation of the costs

Each tree in the list of the admissible spanning trees generated by the algorithm described in the previous section is called a *candidate tree*. Each candidate tree has its own cost, defined as the sum of the link cost of the edges initially attached to the network. To each candidate tree, we add some new costs determined by the global structure of the mathematical formula which represents the candidate tree.

In the following description of the new cost, the terminology "same", "equal" or "different", etc. must be interpreted as the result of robust comparison using the distribution map on Figure 4 in Section 2. The numbers between parentheses represent the cost we use in the expremint described in 6.

1. For each node K, if the maximum of the normalized sizes of the characters in K's *script area* is equal or greater than K's normalized size, add a penalty cost (100(equal), 200(greater)) depending on the comparison result,

[1]In order to reduce the increase of useless paths, left subscripts and left superscripts are pre-processed in advance, and ommited from the check list in the algorithm.

2. If there are some characters in the *script area* which have the same normalized size and normalized center as in a character on the baseline, add a penalty cost (100),
3. If some alphanumeric characters on the baseline have different normalized sizes and centers each other, add a penalty cost (100),

where, *script area* means the total sum of the superscript area, subscript area for ordinary characters, and the sum of the top area and bottom for the nodes corresponding to characters such as '$\sum$', '$\lim$',etc.

After adding these penalty costs to the original cost, we output the candidate tree of the minimum cost as the final result of the structure analysis.

6 Experiments

In this section, we report some experimental results of our method.

6.1 Character Recognition

The character recognition engine used in the experiment recognizes about 450 kinds of characters: alphanumeric characters distinguishing upright and italic fonts, Greek characters and other mathematical symbols. It is trained using 180,000 learning patterns collected from 11 Japanese books, 30 English journals in mathematics scanned by 400dpi, and the scanned images of the prints of the fonts of $\TeX$, Windows and Macintosh.

Table 2 shows the averages of the recognition rates by the experiment carried out for 74 page images chosen from 8 mathematical journals, different from the journals used to collect learning patterns. In the experiment, the characters such as "Ss", "Cc", "0Oo", "1l", etc. are grouped into same classes, while the upright/italic fonts of the alphanumeric characters are distinguished. The numbers between parentheses show the total numbers of the characters counted. Touching characters and broken characters are excluded from the count.

	Alphanumeric	Greek	Arrow
Normal Size	99.22 (88997)	98.17 (2027)	98.54 (137)
Small Size	94.86 (2973)	96.54 (694)	99.27 (42)

	Parenthesis	Other	
Normal Size	98.85 (5398)	98.87 (6568)	
Small Size	95.36 (237)	94.47 (941)	

Table 2. Recognition rates of the first candidate (%)

6.2 Experiments

We experimented with 123 lines of mathematical formulae extracted from 12 mathematical journals. The average number of characters in a line is 34.70.

In order to evaluate our method of structure analysis, we tested using samples including touched characters and broken characters. We counted only the number of link errors, namely even if the character recognition result is wrong or "REJECT", we counted it as "correct" if the link is correct in the output tree.

6.2.1 Result of recognition

Total recognition
In this context, 110 lines were recognized perfectly and there were 13 lines including some recognition errors. Total result in Table 3 shows the total number of the links, the number of the error links and the rate of the correct results.

Robustness against character recognition errors
Further, we examined the recognition results of the link structure for the cases of the links of characters of the following two cases.

1. mis-extracted characters include:

· The characters touched with other characters,
· The characters broken into several pieces,
· The characters composed of several connected components such as 'i', 'j', '$\leq$', etc., which could not unified into one character correctly.

2. undetermined characters include:

· The characters having similar characters with different normalized sizes, such as "Ss", "Cc", etc.,
· The characters rejected by our character recognition engine,
· The characters with the 1st candidate having the normalized size different from the correct normalized size.

The results of the links for the edges connecting these characters are shown in Table 3. It shows that our method is robust against the errors of the character recognition, and efficient for touched or broken characters to some extent.

	total	error	correct rate
total result	4268	36	99.16%
mis-extracted	26	6	76.92%
undetermined	597	4	99.33%

Table 3. Total results and results for difficult cases

6.2.2 Examples

Example 1.

$$\boxed{\begin{aligned}(5.3)\quad k_\alpha(\zeta, z) = \sum_{q=0}^{n-1} c_{n,\alpha,q} \frac{1}{(1-\bar{\zeta}\cdot z)^{n+\alpha-q}(1-\zeta\cdot\bar{z})^{q+1}(1-|a|^2)^n} \\ \times\Big[[(1-\bar{\zeta}\cdot z)P_{n-q-1}^{\alpha-1,-n}\bar{z}\cdot \mathrm{d}\zeta - (1-|z|^2)P_{n-q-1}^{\alpha,-n}\bar{\zeta}\cdot\mathrm{d}\zeta]\wedge(\mathrm{d}\bar{z}\cdot\mathrm{d}\zeta)^q \\ + qP_{n-q-1}^{\alpha,-n}\bar{z}\cdot\mathrm{d}\zeta\wedge(\mathrm{d}\bar{z}\cdot\mathrm{d}\zeta)^{q-1}\wedge\bar{\partial}|z|^2\wedge\bar{\zeta}\cdot\mathrm{d}\zeta\Big],\end{aligned}}$$

$$\begin{aligned}(5.3)\; k_\alpha((, z) = \sum_{q=O}^{n-l} c_{n,\alpha,q} \frac{l}{\left(l-\overline{\zeta}\cdot z\right)^{n+\alpha-q}(1-\zeta\cdot\overline{z})^{q+1}(l-|a|^2)^n} \\ \times\left[\left[\left(l-\overline{\zeta}\cdot z\right)P_{n-q-l}^{\alpha-1,-n}\overline{z}\cdot d\zeta - \left(l-|z|^2\right)P_{n-q-1}^{\alpha,-n}\overline{\zeta}\cdot d\zeta\right]\wedge(d\overline{z}\cdot d\zeta)^q\right. \\ \left. +qP_{n-q-l}^{\alpha,-n}\overline{z}\cdot d\zeta\wedge(d\overline{z}\cdot d\zeta)^{q-l}\wedge\overline{\partial}|z|^2\wedge\overline{\zeta}\cdot d\zeta\right],\end{aligned}$$

In this sample, almost all characters were recognized correctly, except for a symbol in the script areas: the link of the operator '$\wedge$' in the third line was wrongly recognized. However, the characters following this symbol '$\wedge$' were not affected by the error. So far, using top-down methods, there were many cases where the result was as follows:

$$+qP_{n-q-l}^{\alpha,-n}\overline{z}\cdot d\zeta\wedge(d\overline{z}\cdot d\zeta)^{q-l^{\wedge\overline{\partial}|z|^2\wedge\overline{\zeta}\cdot d\zeta],}}$$

Example 2.

$$\boxed{|X_n| \leqq (|p|^{\frac{1}{pL}(\beta-\frac{1}{p-1})})^n}$$

$$|X_n| \leqq \left(|p|F^{\left(\beta-\frac{1}{p-1}\right)}\right)^n 1$$

In this sample, 'p', 'L' and the fractional line are touched and the result of the character recognition was 'F'. Therefore, the system could not find the parent character of '1'.

7 Conclusion and future work

We proposed and implemented a new method of recognizing mathematical formulae using virtual link network. We output the spanning tree of the network with minimum cost, using not only the local costs initially attached to the network but the costs reflecting the global structure of formulae. As the result, we showed that our method was robust against the recognition errors between the characters with different normalized sizes.

As the result of our experiment using the mathematical formulae collected from different mathematical journals and books, we can conclude that the method is robust against the variation of the printing styles of the documents.

The method to correct errors between the characters with same normalized sizes is left the subject of our future research.

References

[1] D. Blostein and A. Grbavic, *Recognition of Mathematical Notation*, Handbook of Character Recognition and Document Analysis, Eds. H.Buke, and P.Wang, Word Scientific, (1997).

[2] M. Okamoto and B. Miao, *Recognition of mathematical expressions by using the layout structure of symbols*, Proceedings of First International Conference on Document Analysis and Recognition Saint Malo, (1991), 242–250.

[3] M. Okamoto and H. Twaakyondo, *Structure analysis and recognition of mathematical expressions*, Proceedings of Third International Conference on Document Analysis and Recognition, Wontreal, (1995), 430–437.

[4] 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 vol.7, no.1, (1996), 2–15.