Detection of Matrices and Segmentation of Matrix Elements in Scanned Images of Scientific Documents

KANAHORI Toshihiro
Research Center on Educational Media,
Tsukuba College of Technology
4-12 Kasuga, Tsukuba-shi, Ibaraki, 305-0821 Japan
kanahori@k.tsukuba-tech.ac.jp

SUZUKI Masakazu
Faculty of Mathematics, Kyushu University
Kyushu Univ. 36, Fukuoka, 812-8581 Japan
suzuki@math.kyushu-u.ac.jp

Abstract

We proposed a method for recognizing matrices which contain abbreviation symbols, and a format for representing the structure of matrices, and reported experimental results in our paper [1]. The method consisted of 4 processes; detection of matrices, segmentation of elements, construction of networks and analysis of the matrix structure. In the paper, our work was described with a focus on the construction of networks and the analysis of the matrix structure. However, we concluded that improvements in the other two processes were very important for obtaining a high accuracy rate for recognition. In this paper, we describe the two improved processes, the detection of matrices and the segmentation of elements, and we report the experimental results.

1. Introduction

The technology of OCR is very efficient for digitizing printed documents. However, current OCR systems can not recognize mathematical formulae, which are very important in scientific documents. Several algorithms for recognizing mathematical formulae have been reported in the literature ([2]–[4]), and some of them can be applied to very simple matrices, such as gridironed matrices (all elements are put on a grid). In our paper [1], we proposed a new practical method for recognizing matrices containing abbreviation symbols, and proposed a format for representing a matrix structure to output the recognition results. We defined the domain to allow matrices to contain *formula elements*, *area symbols* and *repeat symbols* in the matrix coordinates (Fig. 3). The method consists of 4 processes; (1) detection of matrices in a scanned page image, and extraction of characters from each area of the matrices, (2) segmentation of the characters into elements for each matrix, (3) construction of networks of the elements, and (4) analysis of the structure of each matrix. In process (3), we regard a matrix structure as a network of elements connected to each other by links representing their relationships, and we consider independently the horizontally and vertically projected networks (Fig. 1). In process (4), we obtain the areas of variable block pattern elements generating the minimum rectangular area of the matrix by solving a simultaneous system of equations given by the two projected networks (Fig. 2). When the segmentation of the elements and the construction of the networks are performed without error, the structure analysis is always completed correctly. In our paper, we described these processes with a focus on algorithms for constructing the networks and for solving the simultaneous systems. In the experimental results of the paper, we obtained some measure of accuracy of the recognition rate, but it was not satisfactory. Most of the errors resulted from the process for segmentation of the elements.

In this paper, we present details of improved algorithms for the 2 processes (1) and (2) above, and report the experimental results of these methods. The *decorators*, which represent detail numbers for elements, positions of elements, etc., are excluded at present (Fig. 3).

2. Assumptions and Definitions

We assume that a matrix has at least 2 rows and it is fenced by one of 4 couples of parentheses, ( ), { }, [ ] and | | (determinant), and that matrix elements can be gridironed (Fig. 2).

We classify the components of a matrix into 3 classes (Fig. 3). The definitions are;

Formula element has only one grid as its own area, and can connect to other elements in the 8 directions (Fig. 1).

Area symbol has several grids as its own area and a free boundary. Common area symbols are $O, 0, 1, *$, etc. , and a space is also an area symbol.

0-7695-1960-1/03 $17.00 © 2003 IEEE

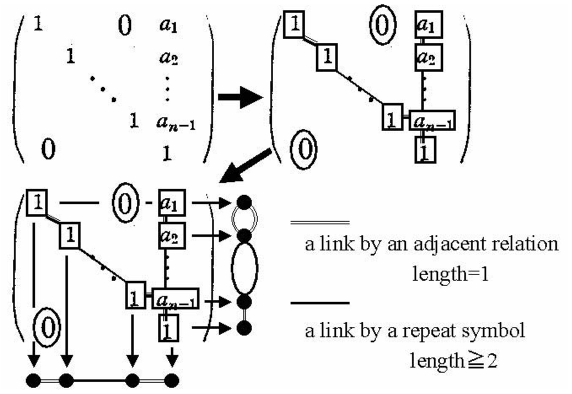

Figure 1. Example of the construction and projection of a network.

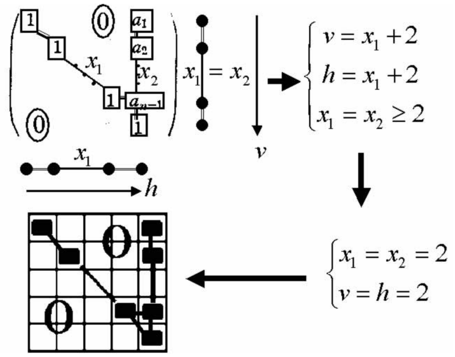

Figure 2. Example of a simultaneous system of equations given by the two projected networks.

Repeat symbol means that formula elements are continuously aligned on the straight line in its direction; ↓, →, ↘ or ↗. It can connect to formula elements and other repeat symbols with different directions.

For our matrix recognition from a page image, we assume that its lines are distinguished, characters are recognized, a coordinate of a bounding rectangle of each character is obtained and an average of the heights of the characters is determined. We define the following symbols to give a clear description:

$$\begin{aligned} H_{\text{ave}} &:= \text{the average of the heights of characters,} \\ \text{BR}(c) &:= \text{a bounding rectangle of a character } c, \\ \text{Int}_x(c) &:= \text{a projected interval of BR}(c) \text{ to the } x\text{-axis,} \\ \text{Int}_y(c) &:= \text{a projected interval of BR}(c) \text{ to the } y\text{-axis.} \end{aligned}$$

2.1. Detection of Matrices

The matrix recognition is applied to each of the lines. Each line is a linear sequence of characters. In a line, characters are horizontally sorted by the x-coordinates of their bounding rectangles, but in a mathematical formula having a 2-dimensional structure, their order is indefinite. For example, in a fraction, characters of the numerator and denominator are mixed in a line, and their order is not defined. The detection of matrices is processed as follows:

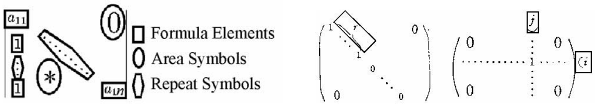

Figure 3. Matrix components (left) and Decorators (right).

1. From the head to the tail of a line, *big parentheses* are located and added to a list of big parentheses, where a big parenthesis is determined by the following conditions:
   - Its height must be greater than $H_{\text{ave}} \times 1.5$.
   - It is one of the parentheses.
   - If it is not one of the parentheses, its width must be less than H_{ave}, considering an error of character recognition. In this case we regard it as '|' (a vert).
2. From the tail to the head of a line, for an opening parenthesis, a closing parenthesis is searched for which can be a couple with the opening parenthesis, where the conditions of a parenthesis couple are:
   - The closed parenthesis is on the right side of the opening parenthesis.
   - There is no big parenthesis between them.
   - There are at least 2 characters between them.
   - Considering the case when one of them is broken, an intersection of their projections to the y-axis is more than a half of each projection.

   When a couple of parentheses are found, they are both erased from the list of parentheses. For a vert, a closing vert is searched for amongst the verts in the list.

If some parentheses are remaining in the lists after the detection process, they are treated as conditional statements.

For each of the detected parenthesis couples, characters between them are segmented into matrix components. We assume that the results of the character recognition are always correct in the following.

2.2. Segmentation of Matrix Elements

Following the detection of matrices, each detected matrix has a set of characters in its own area. It is necessary to segment them into formulae elements or area symbols.

We let $C=\{c_1,\cdots,c_n\}$ be a set of the characters, except for dots, in the matrix. For two characters c and $d\in C$, we define an interval $\mathrm{Btw}_x(c,d)=[s,t]$ by

$$\begin{aligned} \mathrm{Int}_x(c)\cap\mathrm{Int}_x(d)\neq\emptyset &\Rightarrow \begin{cases} s=\min(t_c,t_d)\\ t=\max(s_c,s_d)\end{cases}, \\ \mathrm{Int}_x(c)\cap\mathrm{Int}_x(d)=\emptyset &\Rightarrow \mathrm{Btw}_x(c,d):=\emptyset, \end{aligned}$$

where we let $[s_c,t_c]=\mathrm{Int}_x(c)$ and $[s_d,t_d]=\mathrm{Int}_x(d)$. For a character $c\in C$, we define a set of characters, $\mathrm{NB}_x(c)$, by

$$\mathrm{NB}_x(c):=\left\{ d\in C \;\middle|\; \begin{array}{r} \mathrm{Btw}_x(c,d)\neq\emptyset,\text{ and}\\ \mathrm{Int}_x(b)\not\subset\mathrm{Btw}_x(c,d)\\ \text{for any } b\in C \end{array}\right\}.$$

We also define $\mathrm{Btw}_y(c,d)$ and $\mathrm{NB}_y(c)$ in the same manner as above. For $c\in C$, except for a '$-$' (a minus), we obtain *a scope of* c, $\mathrm{SC}(c)$, by inflating $\mathrm{BR}(c)$ by $\alpha_x S_x(c)$ on both the left and right hand sides and by $\alpha_y S_y(c)$ in both the upper and lower regions. If $\mathrm{BR}(d)$, for $d\in C$, and $\mathrm{SC}(c)$ have an intersection, the characters d and c are segmented into the same element (Fig. 4). The value $S_x(c)$ is obtained by an average of the two shortest lengths among a set of lengths of intervals, $\{\,|\mathrm{Btw}_x(c,d)|\,|\,d\in\mathrm{NB}_x(c)\,\}$. If $|\mathrm{NB}_x(c,d)|<2$, a temporary value $\frac{1}{2}H_{\mathrm{ave}}$ is used instead of the lengths. The value $S_y(c)$ is obtained in a way similar to $S_x(c)$. These values denote the distances between c and other characters which are immediately adjacent to c (there is no character between them). If $S_x(c)$ (or $S_y(c)$) is larger than H_{ave}, we assume that there is no character adjacent to c, so we set $S_x(c)=0$ (resp. $S_y(c)=0$).

The coefficients, α_x and α_y, also depend on c. We set the coefficient α_x as

$$\alpha_x(c):=\begin{cases} 4 & c \text{ is a big symbol } (\sum, \prod, \text{etc.}),\\ 3 & c \text{ is a binary operator},\\ 1.5 & c \text{ is } \textit{long and thin} \text{ alphanumeric},\\ 0.1 & c \text{ is } \textit{short and wide} \text{ alphanumeric},\\ 1 & \text{otherwise}, \end{cases}$$

where we let c be *long and thin* when the height of $\mathrm{BR}(c)$ is greater than two times its width, and let c be *short and wide* when the width of $\mathrm{BR}(c)$ is greater than one and a half times its height, and α_y as

$$\alpha_y(c):=\begin{cases} 1.2 & c \text{ is a big symbol } (\sum, \prod, \text{etc.}),\\ 0.3 & c \text{ is a binary operator},\\ 1.6 & c \text{ is a } \textit{fractional line,}\\ 0.3 & \text{otherwise}, \end{cases}$$

where we let c be a *fractional line* when c is recognized as a fractional line by the character recognition, or when c is recognized as a minus yet there are some characters within a distance of a half of H_{ave} to the upper and lower regions of c. These values are determined in practice. The values for α_x are set larger than those for α_y so that the horizontal connections are tighter than the vertical ones. Almost always a binary operator has characters on both of its sides and often there is some space between them, so α_x for binary operators is set larger than for other symbols. A big symbol often includes an upper or a lower limit formula, and is separated from it by only a very small distance, so α_x for big symbols is set to be large to prevent a big symbol which does not have an upper or lower limit formula from incorrectly including other elements as part of a formula of this type. A fractional line always has a numerator and a denominator, so α_x for fractional lines is set to be larger than for other symbols.

The character '$-$' (minus) is used as a binary operator or a sign. It is necessary to determine whether or not it is being used as a sign. When it is a sign, the space on its left hand side is larger than that on its right hand side. Therefore, for $c=$ '$-$', we use a different definition of $\mathrm{SC}(c)$. We obtain $\mathrm{SC}(c)$ by inflating $\mathrm{BR}(c)$ by $S_x(c)+\beta$ on both the left and right hand sides and by $\alpha_y S_y(c)$ in both the upper and lower regions (Fig. 5). The values $S_y(c)$ and $\alpha_y(c)$ are obtained in a similar way as for other characters. The value $S_x(c)$ is the distance on the x-axis between c and the nearest character to c among $\mathrm{R}(c)$, where

$$\mathrm{R}(c):=\left\{ d\in C \;\middle|\; \begin{array}{l} d \text{ is on the right side of } c \text{ and}\\ \mathrm{Int}_y(c)\cap\mathrm{Int}_y(d)\neq\emptyset \end{array}\right\}.$$

If $\mathrm{R}(c)=\emptyset$, we set $S_x(c)=0$. The constant β is included to allow for some fluctuation, and is set at 4 pixels per 600 dpi (scanned resolution).

In the above process, dots are not considered as components of a matrix element. However, they are very often contained in a matrix element, and it is necessary to segment dots into elements if they are contained. We let $D=\{d_1,\cdots,d_m\}$ be the set of dots in the matrix and let $E=\{e_1,\cdots,e_l\}$ be the set of segmented matrix elements. For a dot $d\in D$ and an element $e\in E$, if $\mathrm{BR}(d)\cap\mathrm{BR}(e)\neq\emptyset$, where $\mathrm{BR}(e)$ is the smallest rectangle containing all bounding rectangles of the characters in e, then d is segmented into e. If d is not segmented into any elements, we obtain the scope of d, $\mathrm{SC}(d)$, by inflating $\mathrm{BR}(d)$ by two times the width of $\mathrm{BR}(d)$ on both the left and right hand sides (Fig. 6). If there is an element whose bounding rectangle and $\mathrm{SC}(d)$ are intersecting, then d is segmented into the element.

There is some space on the left and the right hand side of a function name, for example, sin, cos, max, and so on. A matrix element is often divided at the spaces (Fig. 6). Therefore, if there is a function name on the head (or the tail) of

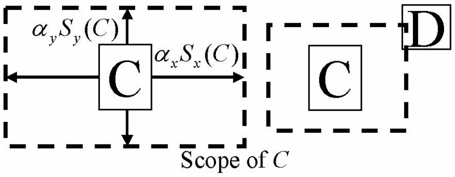

Figure 4. Example of a scope of a character. On the right side, 'C' and 'D' are segmented into the same element.

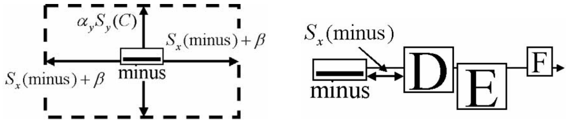

Figure 5. Example of a scope of a character '$-$' (a minus).

an element e, we obtain the scope of e, $\mathrm{SC}(e)$, by inflating $\mathrm{BR}(e)$ by a half of H_{ave} on both the left and right hand sides, and search for other elements whose bounding rectangles intersect with $\mathrm{SC}(e)$, and combine e with these elements.

3. Experimental Results

Data	Text	Journal	Total
Total matrices	375	48	423
Detection errors of matrices	16	2	18
Success rates (%)	95.7	95.8	95.7
Segmentation errors of elements (A)	12	6	18
Number of all characters of matrices	5364	597	5961
Success rates (%)	99.8	99.0	99.7
Segmentation errors of elements (B)	5	4	9
Success rates (%)	98.6	91.3	97.8
Construction errors of networks	6	4	10
Success rates (%)	98.3	91.3	97.5
Total recognition errors of matrices	7	4	11
Total success rates of matrices (%)	98.1	91.7	97.4

Table 1. Experimental results of the segmentation of elements, the construction of networks and the recognition of matrices.

In order to evaluate our new methods, we implemented them as part of our original OCR system ([4]). In our method, when the segmentation of the matrix elements and the construction of the networks are performed without error, the structure analysis is *always exactly* correct. Therefore, we evaluated the detection of the matrices, the segmentation of the elements and the construction of the networks using the system. We used two English and two Japanese mathematics textbooks which included many matrices, mainly used to evaluate parameters, and two mathematical journals. We counted the numbers of errors with respect to the 3 parts. We show the experimental results in Table 1.

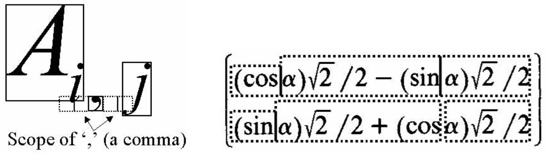

Figure 6. Examples of a scope of a dot (left) and a wrong segmentation of an element by spaces on the both sides of function names (right).

In Table 1, the data "Detection errors" denotes the number of matrices which were not detected. The data "Segmentation errors (A)" denotes the number of characters which were segmented into the wrong elements. There are several ways in which these errors can be counted. For example, assume two elements '0' and '-5' are mis-segmented into the same element '$0-5$'. Considering that the '0' is mis-segmented into the '-5', the number of errors may be counted as 1, as that is the number of characters in the '0', however if we consider that the '-5' is mis-segmented into the '0', then the number of errors may be counted as 2, as that is the number of characters in the '-5'. In this experiment, we recorded the minimum number among all the numbers which could be considered, because they are regarded as costs to correct the segmentation errors. The success rate of (A) is the rate of properly segmented characters for all the characters of the detected matrices. The data "Segmentation errors (B)" denotes the number of detected matrices which have some mis-segmented elements. Its success rate is the rate of properly segmented matrices for all detected matrices. The data "Construction errors" denotes the number of matrices which have some mis-connected elements. The data "Total success rates" denotes the number of matrices which are detected and whose

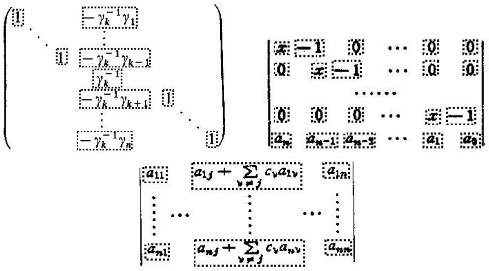

Figure 7. Examples of success of the segmentation.

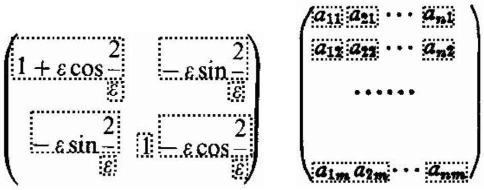

Figure 8. Examples of fails of the segmentation.

elements are properly segmented and connected in relation to the total number of matrices.

Table 1 shows that the segmentation of elements has improved (the success rate in [1] was 75.3%). In Fig. 7, examples of success of the segmentation for complicated matrices are shown. However, several problems were found. The misdetection is mainly caused by errors in the recognition of parentheses due to poor conditions of the prints, and caused by a few very small matrices, which we did not consider. The parameters for our segmentation of elements are determined in practice. In particular, the segmentation rate of '$-$' (a minus) depends heavily on the documents. At the left side matrix in Fig. 8, the fractional lines are very short and there are large spaces between their numerators and denominators, moreover the left space of the minus is much longer than the right one, and at the right side one, the element 'a_{1m}' is too close to 'a_{2m}'. These matrices can not be properly segmented at present. There were often errors in the connection of elements in simple matrices where repeat symbols were not used, because the lengths and directions of the repeat symbols are used to connect the elements. For simple matrices, it is probably most effective for our method to be combined with other methods.

4. Conclusion

We proposed a practical method for recognizing matrices containing abbreviation symbols in the paper [1]. In the paper, we defined the domain to allow the matrices to contain formula elements, area symbols, and repeat symbols in the matrix coordinates. The method consists of 4 parts; detection of matrices, segmentation of elements, construction of networks and analysis of the matrix structure. We reported this work with a focus on the algorithms for constructing the networks and for solving the simultaneous systems. We obtained some measure of accuracy of the recognition rate in the experimental results of the paper, but it was not very satisfactory. We found that most of the errors were caused in the segmentation of elements.

In this paper, we described in detail the detection of the matrices and an improved method for segmentation of the elements, and reported the experimental results. In the detection of matrices, we locate big parentheses using the average height of characters and search for parenthesis couples, and the experimental results indicate a very high accuracy. To segment characters in a matrix into elements, we use the scope of a character, which is obtained by distances between adjacent characters, and some parameters with respect to their features in the mathematical structure. The experimental results showed an enormous improvement in the segmentation of elements. However, the parameters are determined in practice, so the success of the segmentation of elements depends on the documents that are used.

For further improvement, we should determine the parameters statistically, using data from matrices recognized by this method, and for simple matrices should try to combine our method with other independent recognition methods. Moreover, we also need to be able to recognize the decorators of matrices which we excluded in this paper.

References

[1] T. Kanahori and M. Suzuki, *A Recognition Method of Matrices by Using Variable Block Pattern Elements Generating Rectangular Area*, Graphic Recognition, Lecture Notes in Computer Science 2390, Springer, 2002, pp. 320–329.

[2] 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.

[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, pp. 430-437.

[4] Y. Eto, M. Sasai and M. Suzuki, *Mathematical formula recognition using virtual link network*, ICDAR 2001.