An Approach for Processing Mathematical Expressions in Printed Document

B.B. Chaudhuri[1] and U. Garain

Computer Vision & Pattern Recognition Unit
Indian Statistical Institute
203, B. T. Road, Calcutta 700 035, India
bbc@www.isical.ac.in

Abstract. In this paper, we propose an approach for understanding mathematical expressions in printed document. The system consists of three main components namely (i) detection of mathematical expressions in a document, (ii) recognition of the symbols present in the expression and (iii) meaningful arrangement of the recognized symbols. However, detection of mathematical expressions is done through recognition of symbols. Moreover, some structural features of the expressions are also used for this purpose. For recognition of the symbols a hybrid of feature based and template based recognition techniques is used. The bounding-box coordinates and the size information of the symbols help to determine the spatial relationships among the symbols. A set of predefined grammar rules is used to form the meaningful symbol groups to properly arrange the symbols. Experiments conducted using these approaches on a large number of documents show high accuracy.

1 Introduction

The use of computerized document-handling systems has now become widespread. One of the major applications concerns automatic conversion of text data into a computer readable form. Optical Character Recognition (OCR) systems are used for this purpose. OCR tries to recognize the characters on the document automatically and stores the corresponding ASCII code in a computer-processable file.

In spite of tremendous progress, we have not reached the stage where a printed page can be inserted in a OCR system so that a coded file comparable to the keyed-in version is generated. Many document elements like figures, logos, tables etc may baffle the system. Mathematical formulas and equations also fall into this category. It is a serious lacuna for technical documents since such documents generally contain a large number of such mathematical expressions. One naive approach for handling such documents is to manually key in the mathematical expressions into the

[1] Author for correspondence.

S.-W. Lee and Y. Nakano (Eds.): DAS'98, LNCS 1655, pp. 310-321, 1999.

computer. This approach is not acceptable when on line document processing is necessary and hence an automatic approach is called for.

In this paper, we propose an approach for understanding Mathematical Expressions (MEs) contained in a printed document. Earlier, Blostein and Grbavec [1] presented an interesting, systematic review on mathematical notation recognition. Anderson [2] also discussed the problem of the recognition of 2-D mathematical notations. He manually simulated the symbol recognition step and got an error-free recognition result. For symbol-arrangement Anderson [2] adopted a syntactic method and used coordinate grammars. On the other hand, Grbavec and Blostein [3] used a computational technique called graph rewriting where the information was represented as an attributed graph and the computation proceeded by updating the graph by following the graph-rewriting rules. Regarding the definition of mathematical notations, Martin [4] presented a brief list of notational conventions found in use in technical publications. Later on, Belaid and Haton [5] designed a coordinate grammar that is simpler than that of Anderson. Chang [6] used a structure specification scheme to recognize the structure of MEs. Okamoto et al [7], [8] proposed a recursive projection-profile cutting for arranging the symbols. Larvirotte and Pottier [9] used the graph grammar to recognize the mathematical formulas. H. J. Lee et al [10] proposed a procedure-oriented method for understanding MEs where they utilized thirteen features to represent each symbols. Chou [11] used a stochastic grammar to recognize a large set of mathematical expressions, all of which are drawn from a textbook printed by a known typesetter.

Our proposed approach for processing of MEs is based upon the structural features and the formats of the MEs. Functionally, the system is divided into three parts. The first part deals with the identification of the region containing MEs from the rest of the document. Most of the previous studies in this field do not concentrate on this issue. Our approach increases the speed of the system as once the MEs are identified we can concentrate on them instead of going through the whole document. Identification of ME areas is done through checking the presence of mathematical symbols in the text lines. It also uses some structural features of the expressions found in printed documents. The method for checking the presence of mathematical symbols involves the recognition of such symbols. So, part of the symbol recognition phase is done in this first stage.

In the second step, the system recognizes different symbols of the identified MEs in details. Feature-based approach for recognition of symbols is more flexible for size and style variations of the character font but less reliable for complex-shaped patterns where template matching gives better result. So, we use a hybrid of the two approaches for recognizing the symbols. After character recognition, the recognition engine gives the coded form of the MEs, which is represented by a list of symbols in random order. Apart from recognizing the symbols, the system also stores some format information against each mathematical symbol regarding its size, relative position (bounding box coordinates) in the document image etc.

In the third and final step, the system translates the recognition result into a meaningful character string satisfying the required criteria of a certain publication system, which can be used to recompose the MEs in the system. The method for symbol arrangement employs the format information stored against each symbol in

the second step as well as the knowledge of notational conventions of expressing mathematics in a document.

The method for identifying MEs offers the option of storing the ME portion in picture mode for data storage. It also helps to create a database after scanning and interpreting a large collection of technical documents. On the other hand, storing the format information along with the coded form of the MEs also helps in other operations like browsing or automated retrieval where the expressions can be more or less accurately delineated and linked to the text itself. It also helps in structured or hypertext representation of a document.

This paper is organized as follows. In section-2, the results of quantitative survey on the relative abundance of MEs and their structural layout in technical documents are presented. Section-3 describes the procedure for detection of ME areas. Symbol recognition scheme has been described in section-4 while the technique for the re-composition of the MEs are described in section-5. Section-6 presents the test results.

2 MEs in Printed Document: A Quantitative Survey

Our approach for processing ME is based on statistical survey and hence it is expected to be robust and efficient. More than 10,000 document pages were manually scanned. The documents are drawn from various engineering and scientific books, journals, proceedings etc. We also studied softwares like LATEX [12] and Microsoft Equation 3.0 [13] that are commonly used for laying out MEs inside a document on Computers.

The study on the relative abundance of the MEs in printed technical documents gives the results summarized as below:

- Total number of pages scanned is 10,400.
- Total number of pages containing at least one ME is 6,700.
- Total number of MEs found is 11,820.
- Average page size is 391.17-sq. cm.
- Average ME size is 17.66-sq. cm.
- The estimated Probability that a page contains at least one ME is 0.64.
- The average number of MEs per page is 1.14.

The last two figures show the high density of MEs in the technical documents. The study also highlights the following facts on the structural features of the MEs:

- Most of the MEs (61%) have ME equation numbers at the right side of the MEs.
- In 47% of the cases the MEs are printed in italic style and in 27% of the cases in boldface.

This study reveals another important fact that the MEs are separated by white spaces which, in general, is wide enough to be distinguished from plain text region. This is supported by the following statistics:

- Average white spacing between two text lines is 0.2 cm and between two text paragraphs, it is 0.27 cm.
- Average white spaces above and below the ME are of height equals to 0.34 cm.

We obtained also the list of different symbols that appear in MEs. We classify these symbols into three groups: i) numerals, ii) English alphabet and words describing mathematical functions, as well as Greek alphabet, iii) mathematical symbols.

During the statistical study we detected 100 mathematical symbols and 40 Greek letters. The most popular mathematical symbols and Greek letters with their % of occurrences are shown in Table-1.

Table 1. Mathematical Symbols and Letters

Sl. No.	Symbol	% of occurrences*	Sl. No.	Symbol	% of occurrences*
1	=	94	21	$\subset$	4
2	+	93	22	Π	4
3	- (Minus)		23	$\surd$	4
4	/		24	θ	4
5	(	60	25	β	4
6	)		26	$\in$	4
7	Fraction Line	51	27	α	4
8	[	35	28	∇	4
9	]		29	μ	3
10	{	20	30	$\neq$	3
11	}		31	$\rightarrow$	3
12	<	18	32	$\times$	3
13	>		33	$\forall$	2
14	*	15	34	$\notin$	2
15	Σ	15	35	%	2
16	$\int$	12	36	$\oplus$	2
17	~	7	37	$\Rightarrow$	2
18	$\cup$	5	38	δ	2
19	$\cap$	5	39	λ	2
20	$\supset$	5	40	σ	2

** Out of 11,820 expressions.*

Table 2. Mathematical Keywords

Keywords	% of occurrences*	Keywords	% of occurrences*
Log	5	max	3
Exp	4	min	
Sin	4	ln	2
Cos		prob	2
Tan		avg	2

** Out of 11,820 expressions.*

In the MEs, certain words are found which represent mathematical functions. These are called *mathematical keywords*. The topmost 10 keywords and their percentage of occurrences are given in Table-2.

3 Detection of ME Areas

MEs are generally mixed with text in documents. There are two ways in which MEs are found in documents: either as a separate line surrounded by wide white space, or as a part of a normal text line. So, the first step is to detect where the MEs are located in the document. Most of the related works mentioned earlier do not address this problem. Rather, they deal with an isolated ME. Lee and Wang [14] presented a method for extracting MEs in a text document. For this purpose they exploited some basic expression forms but did not provide any detail.

In our approach, we check each text line to decide whether they contain any mathematical symbol listed in Table-1. We adopt some clever approach for such checking to avoid false detection of mathematical symbols due to misrecognition. For example, sometimes letter 'C' may be confused as left parentheses '(', letter 'E' may be confused as square bracket '[', etc. To avoid such confusion for parentheses both the left and right parentheses are searched. For example, to decide that a text line contains square brackets both left '[' and right ']' brackets have to be detected. Presence of curly brackets '{' and '}' is also confirmed in a similar way. Some other similar rules guide this algorithm for searching mathematical symbols in a text line. Whenever a binary operator like '=', '+', '×', or '<' etc. is located in a text line its presence is confirmed by checking left and right side of the operator as binary operator contain two operands one on its left and another on its right side.

Once the presence of some mathematical symbols is confirmed in a text line say, T, it is decided that T contains an ME. At this stage, it is not confirmed whether T contains only ME (i.e. ME is printed as a separate line) or ME along with normal text (i.e. embedded ME). To resolve this confusion, we apply the knowledge extracted from our quantitative survey described in the last section. The survey reveals that the MEs that are not a part of normal text line are separated by white space, which are wide enough to be distinguished from other spacing such as the white spacing between lines or paragraphs etc. So, if T is surrounded by wide white spaces then it is decided that the ME is printed as a separate text line.

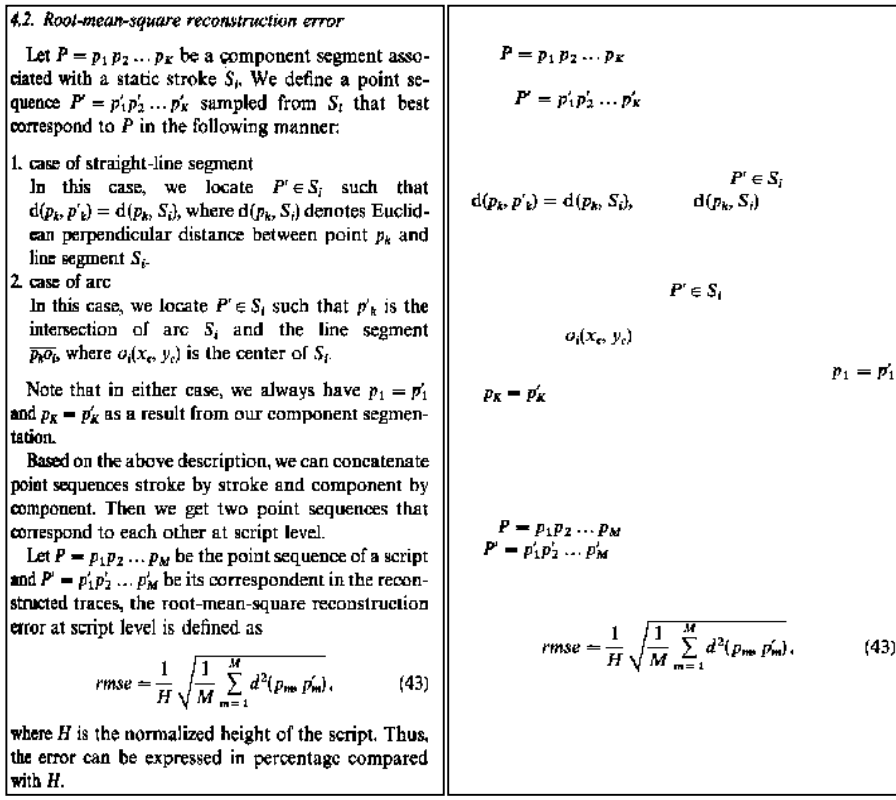
We use another important property of MEs to detect them whenever they are in a separate line. Our quantitative survey reveals that in more than 60% cases MEs have equation numbers whenever they are in a separate line. So, if any equation number can be detected in T then immediately it is decided that T contains only ME instead of any normal text. Detection of equation numbers is relatively easy, as these numbers generally have some well defined structured. Normally, they are written at the extreme right side of the line, sometimes succeeded by series of dots or a straight line or blank spaces.

On the other hand, if T is neither surrounded by wide white spaces nor it has any equation number then it is decided that T contains ME along with normal text. In such cases, ME area is detected and then extracted from T. Let W_1 be the first word from

the left-hand side that contain one or more mathematical symbols in T. Construction of ME area is started by including W_1. Next the ME area grows towards both left and right side following certain rules. Two such rules are given below:

- If W_1 contains only a binary operator then both the immediate left and right side words are included in the ME area.
- Words adjacent to W_1 (on immediate left and right) are included in the ME area provided they contain:
  - Any mathematical symbol
  - Superscript or subscripts
  - Single or a series of dots
  - Numerals

Applying these rules each word included in an ME area is checked. Fig 1(a) shows a document containing both embedded and separate MEs. Fig. 1(b) shows the extracted ME areas.



4.2. Root-mean-square reconstruction error

Let $P = p_1 p_2 \dots p_K$ be a component segment associated with a static stroke S_i. We define a point sequence $P' = p'_1 p'_2 \dots p'_K$ sampled from S_i that best correspond to P in the following manner:

1. case of straight-line segment
   In this case, we locate $P' \in S_i$ such that $d(p_k, p'_k) = d(p_k, S_i)$, where $d(p_k, S_i)$ denotes Euclidean perpendicular distance between point p_k and line segment S_i.
2. case of arc
   In this case, we locate $P' \in S_i$ such that p'_k is the intersection of arc S_i and the line segment $\overline{p_k o_i}$, where $o_i(x_c, y_c)$ is the center of S_i.

Note that in either case, we always have $p_1 = p'_1$ and $p_K = p'_K$ as a result from our component segmentation.

Based on the above description, we can concatenate point sequences stroke by stroke and component by component. Then we get two point sequences that correspond to each other at script level.

Let $P = p_1 p_2 \dots p_M$ be the point sequence of a script and $P' = p'_1 p'_2 \dots p'_M$ be its correspondent in the reconstructed traces, the root-mean-square reconstruction error at script level is defined as

$$rmse = \frac{1}{H}\sqrt{\frac{1}{M}\sum_{m=1}^{M} d^2(p_m, p'_m)}, \qquad (43)$$

where H is the normalized height of the script. Thus, the error can be expressed in percentage compared with H.

$P = p_1 p_2 \dots p_K$

$P' = p'_1 p'_2 \dots p'_K$

$P' \in S_i$

$d(p_k, p'_k) = d(p_k, S_i), \quad d(p_k, S_i)$

$P' \in S_i$

$o_i(x_c, y_c)$

$p_1 = p'_1$

$p_K = p'_K$

$P = p_1 p_2 \dots p_M$

$P' = p'_1 p'_2 \dots p'_M$

$$rmse = \frac{1}{H}\sqrt{\frac{1}{M}\sum_{m=1}^{M} d^2(p_m, p'_m)}, \qquad (43)$$

(a) **(b)**

Fig. 1. Extraction of Mathematical Expressions

4 Symbol Recognition

Design of a recognition engine for mathematical symbols is a difficult task because the engine has to deal with a large character set. The set consists of Roman and Greek letters, operator symbols with a variety of typefaces (normal, bold or italic). Different font sizes are used to designate superscripts, subscripts and limit expressions. To deal with this problem we divide the character set into two groups. The first group, *group 1* includes the following 26 symbols:

"=" "+" "–" "/" "(" ")" "[" "]" "{" "}"
"<" ">" "∑" "∫" "~" "∪" "∩" "⊂" "⊃" "Π"
"√" "×" "∀" "∈" "Δ" Fraction line

The second group, *group-2* includes the rest of the symbols given in section-2. Earlier we have discussed that ME areas are detected through recognition of mathematical symbols. We observe that the *group-1* symbols have very high rate of occurrence, so error in recognizing the symbols of this group not only affects the overall symbol recognition rate but also the efficiency of the module that detects the ME areas. Hence, we use feature-based approach which is more flexible to size and style variation of the character font than the template based one for recognition of *group-1* symbols. Moreover, these symbols have relatively simple shapes, so recognition through stroke/feature analysis is more efficient.

On the other hand, group-2 mostly includes the Roman and Greek letters which have more complex stroke patterns. For recognition of such symbols we combine the positive aspects of feature based and template based approaches. A run number based normalized template matching technique [15] is used for recognizing the *group-2* symbols.

A word is recognized inside a ME when more than one Roman character is found side by side and the inter-character gap is within a predefined threshold. If the recognition engine finds any word inside the ME it checks the list of *mathematical keywords* (discussed in section-2) for a quicker recognition of the word.

Our run number based template matching technique is more or less invariant to scaling and insensitive to character style variations. But it is found to be sensitive to the italic style of the characters. We find that in 47% of the cases the MEs are printed in italic style. So, we apply an approach for the detection of italic characters [16] and then use our template matching technique on the slant-corrected characters.

During recognition of the symbols the system stores some format information against each mathematical symbol regarding its size, relative position (bounding box coordinates) etc. along with its recognized shape name. These format information are used to categorize a symbol as superscript/subscript, upper or lower limit etc.

5 Arrangement of Symbols

After character recognition, a ME is represented by a list of symbols in random order. So, we need to arrange these symbols into a character string satisfying the notational conventions of the 2-D language for mathematical expression. Blostein and

Grbavec [1] reviewed four approaches for symbol-arrangement analysis. They are (i) syntactic methods (ii) projection-profile cutting (iii) graph-rewriting and (iv) procedurally coded rules.

In our approach, we first identify the significant spatial relationships among the symbols. For this purpose, we use the bounding-box coordinates, coordinates of the centroids and the size information of the symbols. Grammar rules are used to group the symbols into meaningful units. These grammar rules are made as general as possible. For example, the system covers 20 forms of integrals, including single integrals, line integrals, double (surface) integrals, and triple (volume) integrals, all with various combinations of limits. Similarly, 5 different types of sums with various combinations of limits are covered by the rules.

To identify meaningful symbol groups in a ME we first order the symbols from left to right according to their x and y values of the bounding box coordinates. Superscripts, subscripts, upper or lower limits are identified by their position and size information. Next, grammar rules are applied to form meaningful units. Sometimes, confusion arises regarding the placement of some symbols. For example, in case of a dot "." alternatives like (i) a decimal point (ii) multiplication sign (iii) terminating symbol (i.e. full stop) (iv) sometimes series of dots are placed in between an ME and its number (i.e. equation number) or (v) noise are considered.

Once the meaningful units are identified they are converted into a coded form. Coding of some MEs has been shown in Fig. 2. Fig. 2(a) and Fig. 2(c) show two MEs and their coded forms are shown in Fig. 2(b) and Fig. 2(d), respectively.

$$X = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 + \ldots + a_{m-1} x^n + a_m y^n \qquad (1)$$

(a)

X = a ₀ + a ₁ x + a ₂ y + a ₃ x y + a ₄ x ² + a ₅ y ² + ... + a _{m-1} x ⁿ + a _m y ⁿ <EQU NO> (1) </EQU NO>

(b)

$$R(C) = \int_0^L \sqrt{(x(s) - P_x)^2 + (y(s) - P_y)^2}\, ds \qquad 2$$

(c)

R(C) = <INTEGRATION> <UPLIM> L </UPLIM> <LOWLIM> 0 </LOWLIM> <SQRT> (x(s) - P _x) ² + (y(s) - P _y) ² </SQRT> ds </INTEGRATION> <EQU NO> 2 </EQU NO>

(d)

Fig. 2. Coding of Mathematical Expressions

For coding of MEs we follow a coding scheme that has similarity with HTML code. The code uses abbreviated forms to indicate different meaningful groups. For example, the keyword <INTEGRATION> indicates the beginning of a mathematical operation called integration. </INTEGRATION> indicates the end of the operation. Similarly, <SUP>, <SUB>, <UPLIM>, <LOWLIM> indicate the beginning of superscript, subscript, upper limit, lower limit, respectively. <EQU NO> indicates the start of the equation number if it exists for an ME.

6 Test Results

Algorithms for detection of ME areas, recognition and arrangement of symbols have been tested on total 80 technical documents containing 82 MEs. Fig. 3 shows some of the test documents. Both clean and degraded versions of the documents are used. The degraded documents are generated by adding synthetic noise. The system performance is evaluated in terms of the performance of each module responsible for various processing.

In 95% cases our algorithm for detecting ME areas properly finds the both the separate and embedded (mixed with text) MEs in a document. Only four MEs (out of 82) have not been properly identified. Two of them are embedded MEs. The rest two MEs have been missed because of the complicated structure of the documents. In these cases, our algorithm fails to analyze the document structure itself.

Earlier it is mentioned that the symbols (*group-1* symbols) with high occurrence rate are recognized through stroke feature analysis. For recognition of other symbols a run-number based normalized template matching technique is used. Both the techniques show high accuracy in recognizing the symbols and the overall correct recognition rate is about 98.3%. Since *group-1* symbols have relatively simple shapes than that of *group-2* symbols, stroke feature analysis shows better result than the template matching technique. The errors in recognition are mainly due to (i) the character font drastically varies from the commonly used fonts (ii) poor quality of the document paper (iii) poor print quality etc.

Rating of symbol arrangement approach is little bit difficult. Sometimes, wrong arrangement of few symbols may drastically change the meaning of the expression even after all other symbols are arranged properly. Hence, to evaluate the approach for symbol arrangement first one has to set a proper evaluation criteria. We emphasis on correct grouping of mathematical symbols. We rate our symbol arrangement approach by examining how many meaningful symbol groups out of all such groups in an ME are properly formed and arranged. We observe that our method for this purpose works well with 92% accuracy. The errors occur mainly due to (i) errors in symbol recognition stage (ii) unavailability of proper grammar rules to represent certain meaningful symbol groups (iii) complication in the structure of some MEs, etc.

It is observed that our system for processing MEs is also efficient in term of execution time. It is implemented on a 166 MHz. Pentium machine with 32 MB Ram using ‘C’ programming language. Documents are scanned at a resolution of 300 dpi. On an average, the document images are of size 3000X2000 pixels. To process such a document the system on an average takes only 1 min. 6 seconds which also includes the time required for binarizing a gray-level image.

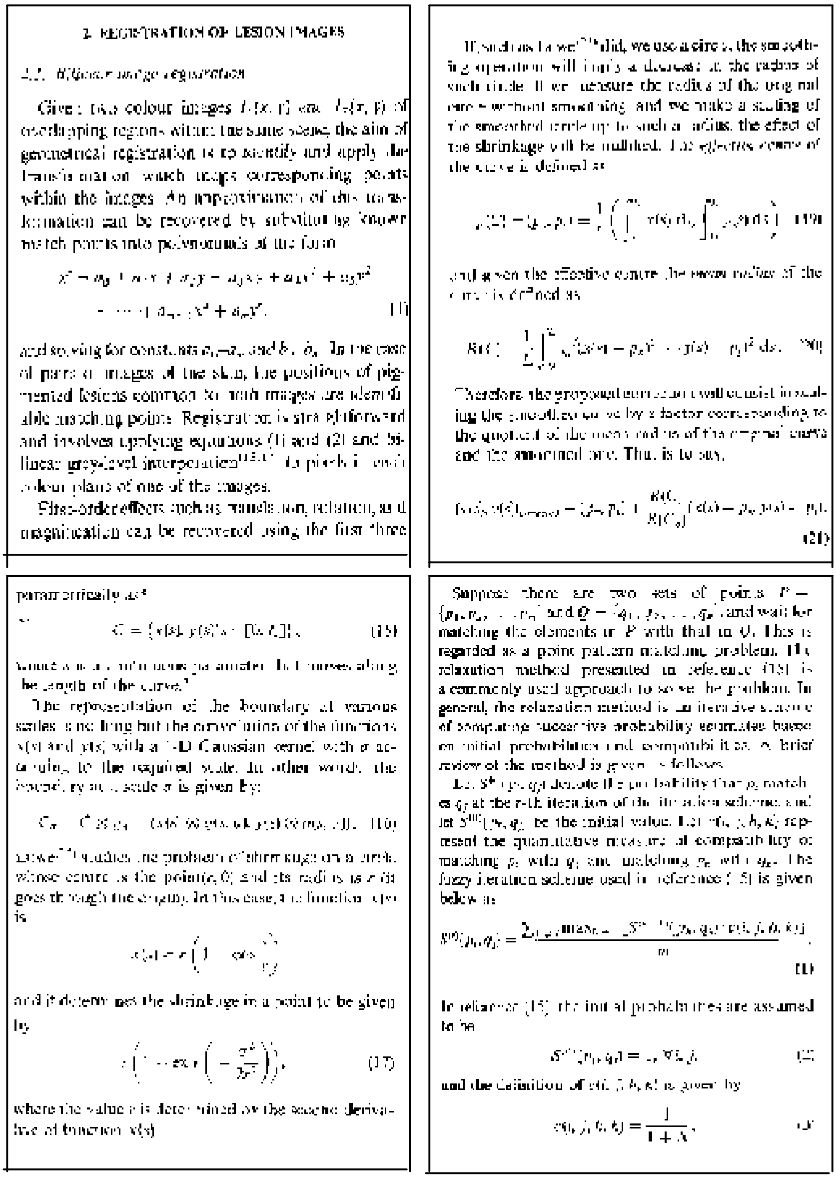


Fig. 3. Some of the test documents

7 Conclusions

In this paper, we present a system for processing mathematical expressions in printed document. We propose an approach built upon the structural features and the formats of the expressions found in use in technical documents. Our method of finding the expressions in a document offers the option of creating a database of mathematical expressions after scanning a large volume of technical documents. To properly arrange the recognized symbols we use their bounding-box coordinates, size information and the coordinates of the centroids and apply some predefined grammar rules to form meaningful symbol groups. These grammar rules can easily be updated to accommodate any new form of such symbol groups. Proper arrangement of the symbols along with their size and style information helps in re-composing the MEs more faithfully. Moreover, the system outputs a coded version of the MEs that helps in converting a paper-based document into its hypertext version.

Acknowledgement

The authors would like to thank the reviewers of the 3rd International Association for Pattern Recognition Workshop on Document Analysis Systems (DAS'98) for their valuable comments to improve the work described in this paper. The work is partly supported by a sponsored project of Department of Science and Technology (DST), Govt. of India.

References

1. D. Blostein, A. Grbavec: Recognition of Mathematical Notation. In: H. Bunke, P. S. P. Wang (eds.): Handbook of Character Recognition and Document Image Analysis, World Scientific Publishing Company, (1997) 557-582
2. R. H. Anderson: Syntax-directed recognition of handprinted 2-D mathematics. Ph.D. Dissertation. Harvard University, Cambridge, M. A. (1968)
3. A. Grbavec, D. Blostein: mathematics recognition using graph rewriting. In: Proceedings of Third International Conference on Document Analysis and Recognition. Montreal, Canada (1995) 417-421
4. W. Martin: Computer input/output of mathematical expressions. In: Proceedings of Second Symposium on Symbolic and Algebraic Manipulations. New York (1971) 78-87
5. A. Belaid, J. Haton: A syntactic approach for handwritten mathematical formula recognition. IEEE Transaction on pattern Analysis and machine Intelligence. 6, 1 (1984) 105-111
6. S. K. Chang: A method for the structural analysis of 2-D mathematical expressions. Information Sciences. 2, 3 (1970) 253–272
7. M. Okamoto, H. Miyazawa: An experimental implementation of a document recognition system for papers containing mathematical expressions. In: Structured Document Image Analysis. Springer-Verlag (1992) 36-53

8. M. Okamoto, H. Twaakyondo: Structure Analysis and Recognition of Mathematical Expressions. IEEE Computer Society Press (1995) 430-437
9. S. Larvirotte, L. Pottier: Mathematical formula recognition using graph grammar. In: Proceedings of SPIE, Vol. 3305. California, USA (1998)
10. H. Lee, M. Lee: Understanding mathematical expressions using procedure-oriented transformation. Pattern Recognition, 27, 3 (1994) 447–457
11. P. Chou: Recognition of equations using a two-dimensional context-free grammar. In: Proceedings of SPIE Visual Communication and Image Processing IV. Philadelphia PA (1989) 852-863
12. LATEX: A document Presentation System. Addison Wesley Publishing Company, Inc. (1986)
13. Microsoft® Word 97: Copyright © 1983-1996. Microsoft Corporation. USA
14. H. Lee and J. Wang: Design of a mathematical expression recognition system. In: Proceedings of Third International Conference on Document Analysis and Recognition. Montreal, Canada (1995) 1084-1087
15. U. Garain, B. B. Chaudhuri: Compound character recognition by a run number based metric distance. In: Proceedings of SPIE, Vol. 3305. San Jose (1998) 90-97
16. B. B. Chaudhuri, U. Garain: Automatic detection of italic, bold and all-capital words from documents. In: Proceedings of International Conference on Pattern Recognition. Australia (1998) 610-612