# An Approach for Recognition and Interpretation of Mathematical Expressions in Printed Document

**B. B. Chaudhuri and U. Garain**

*Computer Vision & Pattern Recognition Unit, Indian Statistical Institute, Calcutta, India*

**Abstract:** In this paper, we propose an approach for understanding Mathematical Expressions (MEs) in a printed document. The system is divided into three main components: (i) detection of MEs in a document; (ii) recognition of the symbols present in each ME; and (iii) arrangement of the recognised symbols. The MEs printed in separate lines are detected without any character recognition whereas the embedded expressions (mixed with normal text) are detected by recognising the mathematical symbols in text. Some structural features of the MEs are used for both cases. The mathematical symbols are grouped into two classes for convenience. At first, the frequently occurring symbols are recognised by a stroke-feature analysis technique. Recognition of less frequent symbols involves a hybrid of feature-based and template-based technique. The bounding-box coordinates and the size information of the symbols help to determine the spatial relationships among the symbols. A set of predefined rules is used to form the meaningful symbol groups so that a logical arrangement of the mathematical expression can be obtained. Experiments conducted using this approach on a large number of documents show high accuracy.

**Keywords:** Document; Mathematical Expression; OCR; Symbol arrangement; Symbol recognition

## 1. INTRODUCTION

In spite of the use of electronic documents, the volume of paper-based documents is growing at a rapid rate. This is because of convenience as well as a long ingrained human tradition of reading and archiving paper-based documents. For the interchange and interaction of information, it is useful to convert one category of document into another. The conversion of a paper-based document into its electronic version has become an important and challenging problem. One approach to solving the problem is the use of Optical Character Recognition (OCR) systems. Given a document, an OCR system tries to recognise the characters on the document automatically, and stores the corresponding ASCII code in a computer-processable file.

Existing OCR systems show high accuracy in processing the text portions, but fail to process properly other document elements like figures, logos, tables, mathematical formulas and equations. Technical documents generally contain a large number of Mathematical Expressions (MEs), but commercial OCR systems cannot handle them. This is partly because MEs involve a large set of symbols that are not standardised, and show wide variations in font size and style. Moreover, mathematical notations convey meaning through subtle use of spatial relationships among the symbols, while it is very difficult to capture all such relationships and faithfully convert them into an electronic form.

A naive approach for handling documents that contain MEs is to manually key the expressions into the computer. This approach is not acceptable when a huge number of such technical documents are to be processed on-line. Thus, an automatic approach for processing such mathematical equations or expressions in the documents is called for.

This paper concentrates on understanding MEs contained in printed documents. The processing of such documents involves three main operations: identification of MEs in the document; symbol recognition; and symbol arrangement. The problem has attracted the attention of several earlier workers. Blostein and Grbavec [1] presented an interesting, systematic review on mathematical notation recognition. The existing techniques for recognition of MEs fall into one of three types: projection-profile cutting; graph rewriting; and

*Received: 17 May 1999*
*Received in revised form: 10 February 2000*
*Accepted: 25 February 2000*

procedurally coded math syntax. Anderson [2] adopted a syntactic method, and used coordinate grammars for ME recognition. He manually simulated the symbol recognition step and got an error-free recognition result. Belaid and Haton [3] designed a coordinate grammar that is simpler than that of Anderson. Among other studies that use a syntactic method, Chang [4] used a structure specification scheme to recognise the structure of MEs. Okamoto and Twaakyondo [5] attempted a projection profile approach for processing MEs. In another study, Okamoto and Miyazawa [6] proposed a recursive projection-profile cutting for arranging the symbols. On the other hand, Grbavec and Blostein [7] 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. Larvirotte and Pottier [8] also used the graph grammar to recognise the mathematical formulas. Lee et al [9] proposed another method for understanding MEs. Their method is procedure-oriented, where it provides step-by-step instructions for recognising an ME. Faure and Wang [10] demonstrated an approach that systematically organises the procedurally coded rules. Chou [11] used a stochastic grammar to recognise a large set of mathematical expressions, all of which are drawn from a textbook printed by a known typesetter.

A few of these studies address the problem of identifying MEs in the document. Most of them assume that the MEs are already segmented from the document, and their processing starts from the segmented MEs. Some of the studies [2,4,7] even avoided the symbol recognition step, where error-free recognition results were obtained by manually simulating the symbol recognition step.

Our proposed method for processing MEs is divided into three components: (i) detection of mathematical expressions in a document; (ii) recognition of the symbols present in the expression; and (iii) arrangement of the recognised symbols. The system first identifies the region containing MEs from the document. The identification of ME areas is done through checking the presence of mathematical symbols in the text lines. Some structural features of the expressions found in printed documents are also used for the purpose. 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.

The symbol recognition procedure involves a hybrid approach of template matching and a feature-based approach, because a feature-based approach 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. Apart from recognising the symbols, the system also stores some format information against each mathematical symbol regarding its size, style (boldface, italic, etc.), relative position (bounding box coordinates) in the document image, etc.

In the final stage, 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 a set of rules representing the knowledge of notational conventions of expressing mathematics in a document.

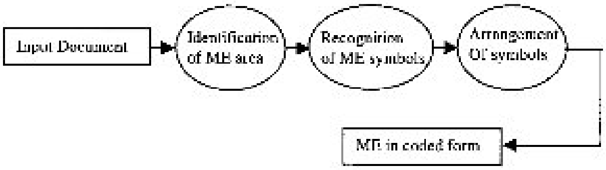


**Fig. 1.** Block diagram of the proposed system.

A brief block-diagram of the system is shown in Fig. 1. Our proposed system involves a collection of different *objects* (more specifically, *classes*), and the overall design is based on object-oriented methodologies [12,13]. Before detailing different methods, we conducted a quantitative survey on MEs to know the structural layout and relative abundance of different mathematical symbols.

This paper is organised as follows. In Section 2, the design concepts for the system are discussed. Section 3 presents the results of a quantitative survey on the relative abundance of MEs and their structural layout in technical documents. Section 4 describes the procedure for the detection of ME areas. A symbol recognition scheme has been described in Section 5, while the technique for the recomposition of the MEs is described in Section 6. Section 7 presents the test results.

## 2. OVERVIEW OF THE SYSTEM DESIGN

We design our system following the Object Modeling Technique (OMT) [14]. A higher-level object diagram with the standard OMT notations is shown in Fig. 2. Figure 2(a) shows different entities (or blocks) of a document. A *document* is an aggregation of many text and non-text *blocks*. The diamond shape notation indicates the aggregation, while the solid ball at the end of the association line indicates the multiplicity symbol 'many'. A line without a multiplicity symbol indicates a one-to-one association. Only object names are shown in Fig. 2(a). Actually, each class in Fig. 2(a) contains many attributes and methods or functions to perform the required operations.

Figure 2(b) shows the *document* class in more details. A *document* has attributes like *document_id*, *document_type* (technical document, letter, bank check, data-entry form, etc.). There are a number of operations associated with the *Document* class. Some of them are (i) *find_block*: finds the different blocks in the document; (ii) *determine_block_type*: determines whether block contains text or non-text, etc. In our system, the *determine_block_type* operation determines whether a block contains any MEs. Other operations like *compress* (for compressing a document image), *print* (for printing the document), etc. may also be there.

The structure of the class *ME* is given in Fig. 3. The *doc_id* attribute of class *ME* links an ME to the document containing it, and the *position* attribute keeps the location

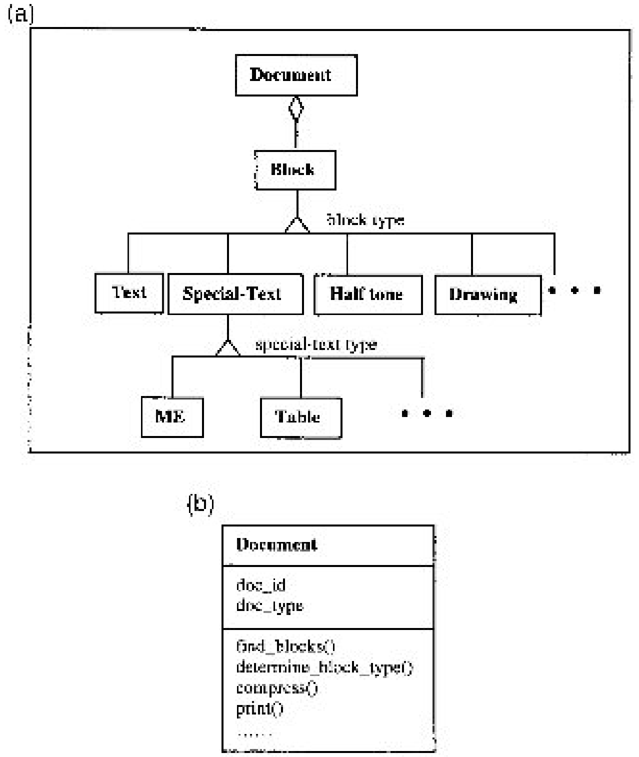


**Fig. 2.** Representation of a document. (a) The object hierarchy; (b) *Document* class details.

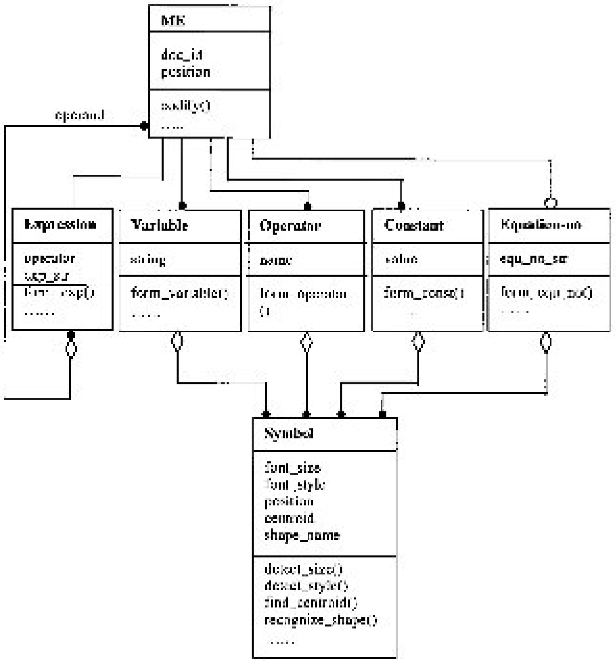


**Fig. 3.** The object-diagram used for the proposed system.

information of an ME in the document. The operation *codify()* converts the ME into an HTML-like code.

An ME contains a number of symbols represented by the class *Symbol* (see Fig. 3). In this class, the operation *recognize_shape* recognises the symbol and sets a value to the attribute *shape_name*. The symbols form different meaningful units or groups, like variable, constant, operator, equation number, etc. These units are used by the *form_exp* operation of the class *Expression* form. This operation forms the expression through the symbol-arrangement analysis. Each *Expression* contains an *operator* that may be a unary or binary operator, or other operators like integration, summation, sine, cosine, etc.

We use object-oriented methodologies as they have several advantages over the other existing approaches. Since the objects are inherently decoupled from each other the maintenance and enhancement of the system is easier. Reusability [15] is another advantage of the proposed system. For example, an existing OCR system can use the components to recognise mathematics in the document.

## 3. MEs IN PRINTED DOCUMENT: A QUANTITATIVE SURVEY

Our approach for processing MEs is based on a statistical survey, and is hence expected to be robust and efficient. For this purpose, a large number of documents drawn from engineering and scientific books, technical journals, proceedings, etc. were manually examined. We also examined the software packages like LATEX [16] and Microsoft Equation 3.0 [17] commonly used for laying out MEs inside a document.

The results of our study on the MEs of these documents are summarised below. More detailed results can be found in Chaudhuri and Garain [18].

- Total number of pages scanned is 10,400.
- Total number of pages containing at least one ME is 6700.
- Total number of MEs found is 11,820.
- The estimated Probability that a page contains at least one ME is 0.64.
- The average number of MEs per page is 1.14.
- 150 different symbols were noted in the expressions. These symbols can be classified into four groups: (i) numerals, (ii) English characters, (iii) Greek letters, and (iv) special symbols (e.g. '+', '=', etc.). Some of the most popular symbols (excepting the English alphabet) are shown in Table 1.
- The expressions found are (a) either printed in a separate line or block, with white spaces above and below, or (b) embedded directly into the text line.
- For the MEs printed as separate text lines the following two important points are noted:
  —Most of the MEs (61%) have equation or equality numbers at the right hand side of the MEs.
  —The mean value of white spacing between two text lines is nearly 0.4 times the text height, whereas the mean value of the white spacing above and below the

**Table 1.** Occurrence statistics of mathematical symbols (results manually computed on 11,820 expressions)

| Sl. No. | Symbol | % of occurrences | S1. No. | Symbol | % of occurrences |
|---|---|---|---|---|---|
| 1 | = | 94 | 21 | α | 4 |
| 2 | + − / | 93 | 22 | ∇ | 4 |
| 3 | ( ) | 60 | 23 | μ | 3 |
| 4 | Fraction Line | 51 | 24 | η | 3 |
| 5 | [ ] | 35 | 25 | δ | 3 |
| 6 | { } | 20 | 26 | ϕ | 3 |
| 7 | < > | 18 | 27 | λ | 3 |
| 8 | * | 15 | 28 | π | 3 |
| 9 | Σ | 15 | 29 | σ | 3 |
| 10 | ∫ | 12 | 30 | ≠ | 3 |
| 11 | ~ | 7 | 31 | × | 3 |
| 12 | ∪ | 5 | 32 | ∀ | 2 |
| 13 | ∩ | 5 | 33 | ∉ | 2 |
| 14 | ⊃ | 5 | 34 | % | 2 |
| 15 | ⊂ | 4 | 35 | ⊕ | 2 |
| 16 | Π | 4 | 36 | ⇒ | 2 |
| 17 | √ | 4 | 37 | δ | 2 |
| 18 | θ | 4 | 38 | λ | 2 |
| 19 | β | 4 | 39 | σ | 2 |
| 20 | ϵ | 4 | 40 | ± | 2 |

**Table 2.** Relative frequency of mathematical keywords

| Keywords | % of occurrences |
|---|---|
| log | 5 |
| exp | 4 |
| sin cos tan | 4 |
| max min | 3 |
| Lt lim | 3 |
| prob | 2 |
| avg | 2 |
| In | 2 |

ME is about 1.8 times the text height (text height means the height of the normal text; the point-sizes 10 and 12 are most common for technical documents).

- In the expressions, certain words represent mathematical functions. We call them *mathematical keywords*, and treat them as operators. The topmost 12 keywords and their percentage of occurrences are given in Table 2.

## 4. DETECTION OF ME AREAS

Earlier, we discussed that an ME can appear either as a separate line or as part of running text lines. In ME recognition the first step is to detect the location of the ME in the document. Though the problem of processing MEs has attracted the attention of many scientists, very few studies have addressed this detection problem.

Among the earlier reports, Lee and Wang [19] presented a method of extracting MEs where they exploited some basic expression forms, but did not provide any detail. More recently, Toumit et al [20] proposed an approach for the separation of mathematical formulas from standard text using a character matching technique and propagating the labelling process of mathematical components around special mathematical symbols. In other work, Kacem et al [21] presented a method for formula extraction without character recognition. Their method is based on finding the location of the most significant symbol, and then extension to the adjoining symbols is done using contextual rules. A fuzzy-logic based labelling technique is also used.

In our approach, the MEs printed in separate lines are detected without any character recognition. At first, the text lines are detected by finding the valleys of the projection profile computed by a row-wise sum of grey values. The position between two lines where the projection profile height is at a minimum denotes the boundary between two text lines. In this way, the MEs printed in separate lines are also extracted as text lines.

Next, for a text line $T$, simple connected component analysis gives us all the symbols present in that line. Let $Y_i$ be the $y$ co-ordinate (taking the top left-most pixel of the document image as the origin) of the bottom-most row of symbol $S_I$, and $n$ be the total number of symbols in $T$, then the mean and Standard Deviation ($SD$) of $Y_i$ values are calculated as follows:

$$\overline{Y}_i = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

$$SD = \sqrt{\frac{1}{n}\sum (Y_i - \overline{Y}_i)^2}$$

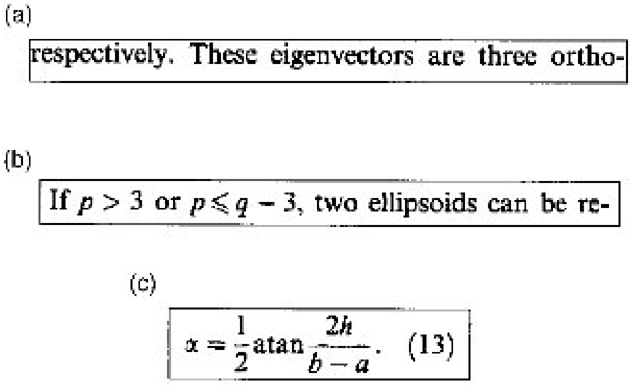
Since, in a simple English text line, the bottom-most rows of the majority of the symbols (except those having descending parts) are nearly aligned on the base line, leading to a small *SD* value. On the other hand, the symbols for an ME (printed in a separate line) are generally scattered over the region. So, these symbols contribute to a very large *SD*. For computational ease, we calculate *SSD* instead of *SD* as follows:

$$SSD = \frac{1}{n}\sum_{i=1}^{n} Y_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n} Y_i\right)^2$$

This *SD* value is a good measure for distinguishing an ME from a text line. For example, Fig. 4 shows three text lines of a document, where the first line contains normal text; there is an embedded ME mixed with normal text in the second line, and the third line contains only an ME. The *SD* values are 2.44 and 3.72 for the first two lines, respectively, whereas it is 16.37 for the third line.

So, if a text line $T_m$ shows a *SD* value greater than a predefined threshold, it is suspected that $T_m$ contains an ME. The presence of ME is further confirmed by testing another property. The MEs printed in a separate line are surrounded by wide white spaces.

For detecting embedded MEs (MEs mixed with normal text) we employ a character recognition approach. Each text line (except $T_m$'s) is checked to find one or more of the mathematical symbols listed in Table 1. To avoid false acceptance due to mis-recognition, some heuristics are used. For example, sometimes the character '(' may be confused with '**C**' , while '[' may be mis-recognised as '**E**'. To avoid this, both the left and right parentheses are searched. Similarly, to decide that a text line contains square brackets, both left '[' and right ']' brackets have to be detected. The presence of curly brackets '{' and '}' is also confirmed in a similar way. When a binary operator like '=', '+', '×', or '<', etc. is detected in a text line, its presence is confirmed by checking the left and right side of the operator. Normally, the sides contain symbols from the English or Greek alphabet, or numerals.


(a)

respectively. These eigenvectors are three ortho-

(b)

If $p > 3$ or $p \leqslant q - 3$, two ellipsoids can be re-

(c)

$$\alpha = \frac{1}{2}\operatorname{atan}\frac{2h}{b-a}. \quad (13)$$

**Fig. 4.** Identification of MEs printed in separate lines. (a) Line (with normal text only) having $SD = 2.44$; (b) Line (with ME mixed with nomal text) having $SD = 3.72$; (c) ME printed in a separate line having $SD = 16.37$.

Once an embedded ME is found in a text line $T_n$, the ME area is detected and extracted from the $T_n$. Let $W_1$ be the first word (words of a text line are distinguished by looking at the vertical projection profile where gaps between the words show up a reasonable long minima) from the left-hand side that contains one or more mathematical symbols in $T_n$. Construction of the ME area is started by including $W_1$. Next, the ME area is grown towards the left and right sides by following the rules 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 the immediate left and right) are included in the ME area, provided they contain:
  —One or more mathematical symbols (including brackets).
  —Superscript or subscripts.
  —Single or a series of dots.
  —Numerals.

Figure 5(a) shows a document containing both embedded and separate MEs. Figure 5(b) shows the extracted ME areas.

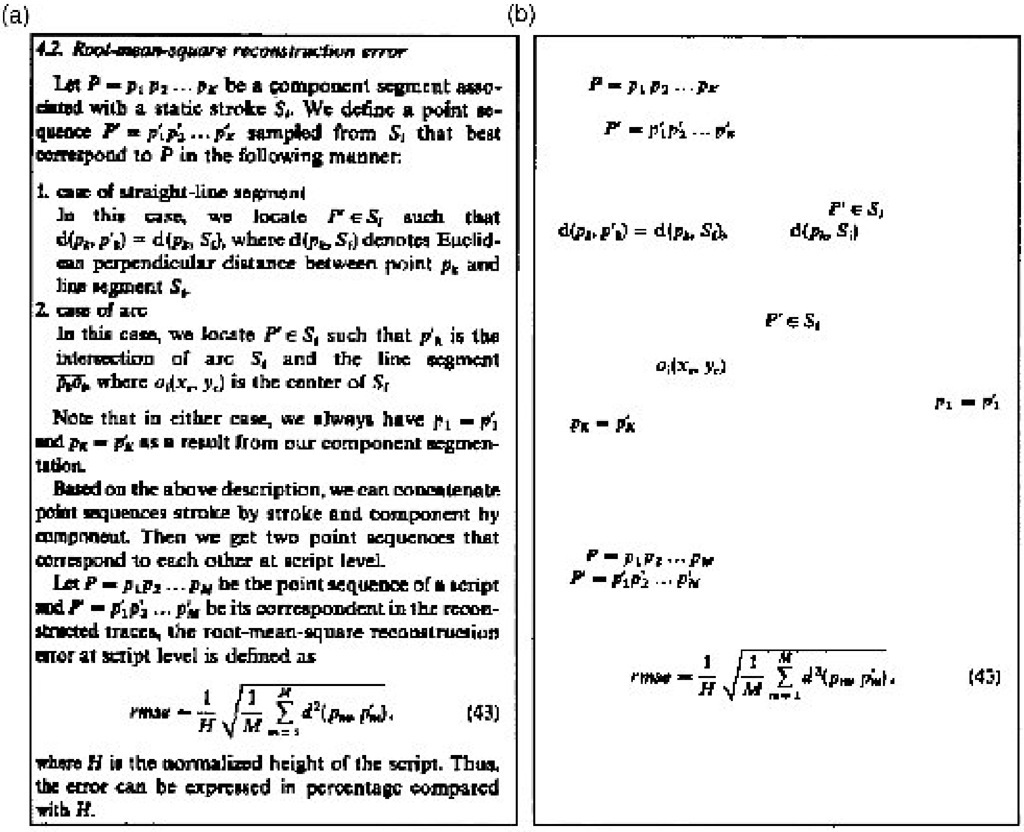
## 5. SYMBOL RECOGNITION

The design of a recognition system for mathematical symbols is difficult because it 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), brackets and abbreviation symbols (e.g. symbols for *for all*, *there exist*). Different font sizes are used to designate superscripts, subscripts and limit expressions. Martin [22] presented a brief list of notational conventions found in technical publications.

We employ the traditional character recognition approach for the recognition of mathematical symbols. Approaches to character recognition are popularly grouped into two categories, namely template matching and stroke feature-based recognition. Both approaches have their own advantages and disadvantages. Template matching can be very accurate and efficient if the test characters are identical with the stored templates in shape and size. However, the approach can be sensitive to positional changes and less flexible to font size and style variations. On the other hand, stroke feature-based approaches are flexible to font size and style variation, but less reliable if the strokes are not correctly segmented from the characters.

### 5.1. Grouping of Mathematical Symbols

For the purpose of recognition, we partition the mathematical symbols into two groups. The first group named *group-*



(a)

*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 $H$.

(b)

$P = p_1 p_2 \dots p_K$
$P' = p'_1 p'_2 \dots p'_K$
$P' \in S_i$
$d(p_k, p'_k) = d(p_k, S_i)$,
$d(p_k, S_i)$
$P' \in S_i$
$o_i(x_c, y_c)$
$p_1 = p'_1$
$p_K = p'_K$
$P = p_1 p_2 \dots p_M$
$P' = p'_1 p'_2 \dots p'_M$
$rmse = \frac{1}{H}\sqrt{\frac{1}{M}\sum_{m=1}^{M} d^2(p_m, p'_m)}$, (43)

Fig. 5. Extraction of mathematical expressions. (a) Document page; (b) extracted MEs.

*1* includes 50 symbols listed in Fig. 6. The second group called *group-2* includes another 100 symbols most of which are mainly the Greek, Roman alphabets and numerals. The *group-1* symbols have relatively simpler shapes compared to those of *group-2* symbols and our recognition approaches are different for these two groups.

Recognition of *group-1* symbols is more important because most of the *group-1* symbols have very high rate of occurrence in MEs. Moreover, the embedded MEs are detected through recognition of a few mathematical symbols belonging to *group-1*. Hence, errors in recognising the *group-1* symbols would affect the overall symbol recognition rate, as well as the efficiency of detecting the embedded ME areas.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| — | = | − | ≠ | − | ≡ | > | + | · | ± |
| * | × | / | ÷ | ( | ) | \| | \| | { | } |
| < | ≤ | > | ≥ | % | ° | ∴ | ⊥ | ↔ | ← |
| ↑ | → | ↓ | ⇔ | ⇐ | ⇑ | ⇒ | ⇓ | ∩ | ∪ |
| ⊃ | ⊇ | ⊄ | ⊂ | ⊆ | ⊕ | ∅ | ⊗ | ∀ | ∃ |
| ∧ | ∨ | ∇ | Δ | √ | Σ | Π | ∠ | ∫ | ∈ |

Fig. 6. List of *group-1* symbols.

## 5.2. Recognition of *group-1* Symbols

For the recognition of *group-1* symbols, we use a feature-based approach that is flexible to character font size and style variation. Moreover, because of the shape simplicity of these symbols, the stroke feature-based approach is robust and efficient.

The features are chosen with the following consideration: (a) robustness, accuracy and simplicity of detection; (b) speed of computation; (c) independence of fonts; and (d) needs for the classifier design. We consider simple stroke features like (i) vertical line, (ii) horizontal line, (iii) V-shape, (iv) VU-shape, (v) circle, (vi) circular arc, etc. Apart from these strokes, other secondary features like (i) aspect ratio of the symbol bounding box, (ii) slant angle of a straight line, (iii) angle between two touching straight lines, (iv) radius of a circle, etc., are also calculated.

For an input symbol S, the presence of the stroke features is checked one by one. Secondary features like aspect ratio, slant angle, etc., are also computed for S. Primary classification is done based on the stroke features present in S. As more than one symbol (e.g. a 'minus' sign and a 'fraction line') may belong to a single class, S is finally recognised based on the secondary features. It may be noted that these features are simple (mostly linear) in structure, and hence quick and easy to detect. Their distortion due to noise can be easily taken care of. They are quite stable with respect to font variation.

### 5.3. Recognition of *group-2* Symbols

The *group-2* symbols mostly includes the Roman and Greek letters. These symbols have more complex stroke patterns than that of *group-1* symbols. We have employed a hybrid technique that can combine the positive aspects of the feature- and template-based approaches.

In our system, we consider a set of size normalised feature vectors such as (i) crossing count, (ii) projection profile, (iii) zonal optical density, (iv) accumulated curvature, etc. These features have been used by previous researchers [23,24] for developing OCR systems, and their descriptions are omitted here for brevity.

A set of training samples of various fonts in different sizes and styles are taken, and the feature vectors for these samples are mapped in a multidimensional feature space. Nearly 2000 training samples (on average, 20 samples per symbol) that represent typical variations of the characters are correctly chosen and mapped in this way. Each sample represents a point in the multidimensional space, and the clustering of the points is observed. The span of the points for each character is mapped by a functional form.

In the classification phase, the normalised features corresponding to the input (target) symbol $T$ are computed and mapped in the feature space. Let $f_t$ be the feature vector for $T$ and $f_i$ be the feature vector for the source symbol $S_i$. In the feature space, we have 100 such $f_i$'s for 100 *group-2* symbols. We find a distance $d(f_t, f_i)$ which is minimum over all $i$'s. Finally, $T$ is recognised as $S_i$ if $d(f_t,f_i)$ is less than a pre-defined threshold ($\delta$). Mathematically, $T$ belongs to $S_i$ if

$$d(f_t,f_i) < d(f_t,f_j) \quad \forall j \text{ and } i \neq \mathrm{j}$$

and

$$d(f_t,f_i) < \delta.$$

To speed up the process, if a character is found inside a word the recognition engine consults the list of mathematical keywords (discussed in Section 2) for a quick recognition of the character. A word is detected inside a ME when more than one Roman character is found side by side and the inter-character gap is within a predefined threshold. During symbol recognition, the system stores some format information (attributes of class *Symbol*, as shown in Fig. 3) against each mathematical symbol regarding its size, relative position (bounding box coordinates), etc., along with its recognised shape name. This format information is used to categorise a symbol as superscript/subscript, upper or lower limit, etc., as well as to arrange the symbols in a meaningful string.

### 5.4. Resolving Ambiguities in Symbol Recognition

In MEs, there are symbols that have more than one meaning. For example, a dot can represent a full-stop sign, a decimal sign, a multiplication symbol, part of a series of dots to indicate continuation ('...'), a symbol annotation (a), part of symbols like '∴' or '∵,' or noise. Our symbol recognition procedure tries to resolve such ambiguities by using some contextual information. The notational conventions for writing MEs define this contextual processing. For example, a horizontal line segment, say $l$, is recognised as (i) a fraction line: if there are symbols above and below $l$; (ii) a symbol annotation ('$\bar{a}$'): if there is a symbol(/s) only below $l$; (iii) an underscore ('_') or a minus sign ('−'): confirmed by checking the position of the $l$'s bounding box relative to the bounding boxes of its left and right symbols; (iv) part of another symbol like '=', '≅', '⊆', '⊇', '≡', '≤', or '≥' etc.: confirmed by checking the presence of other shapes like one or more horizontal lines of equal length or shape like '<', or '⊂', etc.; (v) a simple horizontal line: when $l$ does not convey any special meaning, it is understood that $l$ is a simple horizontal line. Sometimes, such a pure horizontal line (instead of series of dots or blank spaces) exists in between the expression and the expression number.

As for another example, a slanted line may convey different meanings that are determined as follows: (i) a division sign ('/'): if both sides contain symbol(/s); (ii) a symbol annotation ('$a'$'): it gets confirmed by checking its bounding box position to the bounding box of its left-side symbol (size information also helps in such case); (iii) seven ('7'): sometimes the slant line may be a part of a broken '7'. This is confirmed by checking the presence of other numerals on its both left and right sides.

## 6. ARRANGEMENT OF SYMBOLS

At the end of the character recognition stage, 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 2D language for mathematical expression. Arrangement of symbols is done in two stages. In the first stage, the spatial relationships among the symbols are identified. Small symbol groups (e.g.$x^y$,$a_i$, etc.) are formed by exploiting the spatial relationships among the symbols. Once these symbol groups are formed, logical relationships among these groups are determined. Two or more symbol groups form small expressions based on the logical relationship they have. These small expressions finally construct the full expression.

### 6.1. Formation of Symbol Groups

Formation of symbol groups is important to recognise an ME, because same set of symbols conveys different meaning depending upon the spatial relationships among the symbols. For example, '$x^y$' and '$x$ $y$' both involve the same set of symbols, (i.e. '$x$' and '$y$'), but in the first case the symbols belong to a single group (a superscripted variable), whereas in the second case, the two symbols form two different groups (two simple variables).

In our approach, spatial relationships among the symbols are determined by identifying the physical structure of the

ME. For this purpose, we use the bounding-box coordinates, the coordinates of the centroids and the size information of the symbols. Superscripts, subscripts, and upper or lower limits of a symbol are identified by their size and bounding box coordinates w.r.t. that symbol. The operations *form_variable()*, *form_constant()*, *form_operator()* (see Fig. 3) take care of the formation of variables, constants, operators, respectively.

### 6.2. Identification of Logical Relationships

The logical relationship among different symbol groups are determined to construct the final expression. A number of intermediate expressions are formed around each operator. The *form_exp()* operation of the *Expression* class (see Fig. 3) forms an expression around an operator. This operation is guided by a set of rules and the physical layout of the ME. The rules define whether an expression involving an operator is valid or not. The rules are made as general as possible. For example, our 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, five different types of summation with various combinations of limits are covered by the rules. Some of the rules are shown in Fig. 7.

EXP stands for expression, which includes all general forms of expression involving variables and constants. It may be a simple variable, a numeral, or a variable with subscripts or superscripts, or a function name or statement. Expressions within parentheses (e.g. (EXP), [EXP], etc.) are also treated as expressions.

The attribute *operator* of the class *Expression* determines which rule is to be applied to form the expression. The operation *form_exp()* constructs an expression and returns a pointer to the root of a parse tree generated for the expression. The parse tree involves the operator and its operands.

The operation *codify()* of the class *ME* encodes the expression in an HTML-like code. Since this operation can be inherited into other subclasses of ME, classes like *Variable*, *Expression*, *Operator*, etc. also employ this operation. Hence, each of the variables, operations, expressions, etc. is encoded whenever it is identified.

Figure 8(a) shows an equation and Fig. 8(b) shows details of the object instances created for this equation. For the sake of simplicity, the instances for the class *Symbol* are not shown in this diagram. The parse tree generated for each of the object instances of the class *Expression* is shown in the figure. A codified version for each of the ME units (like variables, equation number, constants, etc.) are also shown. The final code for the entire expression is shown separately in Fig. 8(c). Figure 9 shows the final coding of two more MEs. Figures 9 (a) and (c) show two expressions, and coding of these expressions are shown in Figs 9 (b) and (d), respectively.

$$EQUALTO\ :EXP = EXP$$

$$PLUS\ :EXP + EXP$$

$$MINUS\ 1:EXP - EXP$$

$$MINUS\ 2:-EXP$$

$$INTO\ 1:EXP \times EXP$$

$$INTO\ 2:EXP\,.EXP$$

$$INTO\ 3:EXPEXP$$

$$INTO\ 4:EXP * EXP$$

$$DIV\ 1:EXP \div EXP$$

$$DIV\ 2:\frac{EXP}{EXP}$$

$$DIV\ 3:{}^{EXP}\!/_{EXP}$$

$$SUM\ 1:\sum EXP$$

$$SUM\ 2:\sum_{EXP} EXP$$

$$SUM\ 3:\sum_{EXP}^{EXP} EXP$$

$$SUM\ 4:\sum\nolimits_{EXP} EXP$$

$$SUM\ 5:\sum\nolimits_{EXP}^{EXP} EXP$$

$$SQRT\ 1:\sqrt{EXP}$$

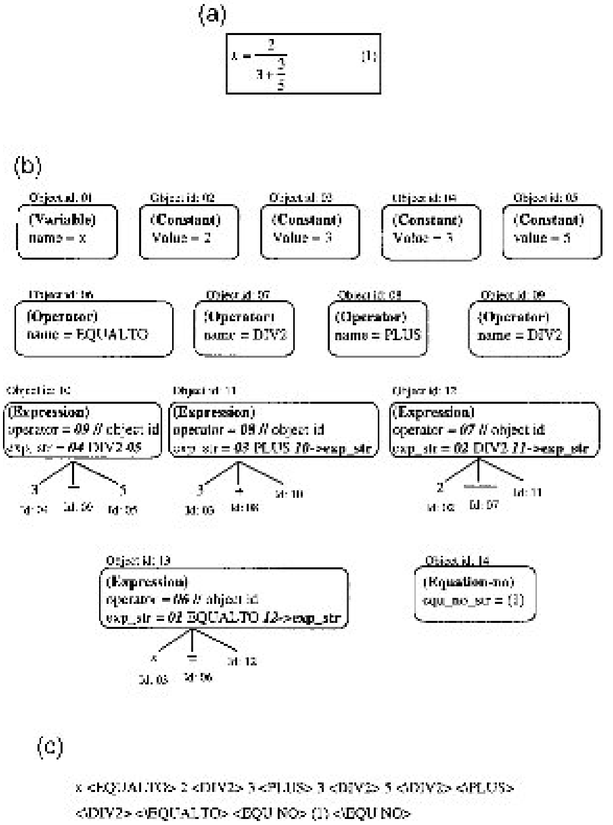
$$SQRT\ 2:(EXP)^{1/2}$$

**Fig. 7.** Some of the rules used for arrangement of symbols.

## 7. TEST RESULTS

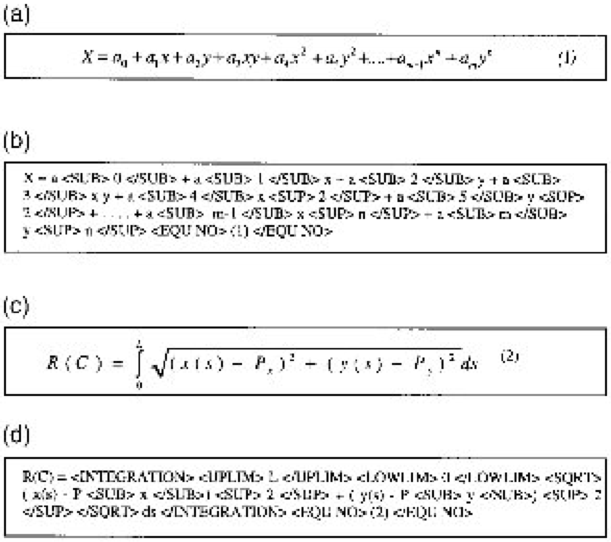
Algorithms for the detection of ME areas, and the recognition and arrangement of symbols have been tested on 120 technical documents containing 140 MEs. Out of these 120 documents, 20 are taken from the UW-III English/Technical Document Image Database (prepared by the Intelligent Systems Laboratory at the Department of Electrical Engineering 352500, University of Washington). Figure 10 presents some of the test documents. Both clean and degraded versions of the documents are used. Degraded documents are generated synthetically by following a model proposed by Kanungo et al [25].

Our system is implemented on a 166 MHz Pentium PC with 32 MB RAM. The object-oriented design is implemented using the C++ language on Microsoft Visual C++ (ver. 5.0) platform. Documents are scanned at a resolution of 300 dpi. On average, the document images are of 3000×2000 pixels in size. It is observed that the system is efficient in terms of processing time. On average, it takes



x <EQUALTO> 2 <DIV2> 3 <PLUS> 3 <DIV2> 5 </DIV2> </PLUS>
</DIV2> </EQUALTO> <EQU NO> (1) </EQU NO>

**Fig. 8.** Processing of mathematical expression. (a) A mathematical expression; (b) the object instances created to process the expression in (a); (c) final coding of the expression in (a).



(a)

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

(b)

X = a <SUB> 0 </SUB> + a <SUB> 1 </SUB> x + a <SUB> 2 </SUB> y + a <SUB>
3 </SUB> x y + a <SUB> 4 </SUB> x <SUP> 2 </SUP> + a <SUB> 5 </SUB> y <SUP>
2 </SUP> + . . . . + a <SUB> m-1 </SUB> x <SUP> n </SUP> + a <SUB> m </SUB>
y <SUP> n </SUP> <EQU NO> (1) </EQU NO>

(c)

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

(d)

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

**Fig. 9.** Coding of mathematical expressions.

only 56 seconds to process a document. This also includes the time required for binarising a grey-level image.

Our algorithm for detecting ME areas properly finds 132 MEs (both separate and embedded, i.e. mixed with text) in the documents. This shows about a 94.29% accuracy. Detailed results are given in Table 3. Among the eight unidentified MEs, six are embedded MEs only. The other two MEs are missed because of the complicated structure of the documents, although they are printed in separate lines. In these cases, our algorithm fails to analyse the document structure itself. On the other hand, for three cases a part of normal text line is misidentified as embedded MEs, where some text symbols are wrongly recognised as mathematical symbols. We manually detected and extracted the eight MEs (not identified automatically) to test the remaining modules of the system.

Our system achieves a high accuracy in recognising both the *group-1* and *group-2* symbols. The overall correct symbol recognition rate is 97.50%. Details of the recognition results are given in Table 4. The recognition rate for the *group-1* and *group-2* symbols is 98.10% and 97.12%, respectively. It is observed that the stroke/feature analysis approach performs better than the feature-based template matching techniques. We have identified two reasons behind this: (i) the number of *group-1* symbols is less than that of *group-2* symbols; and (ii) the *group-1* symbols have simpler shapes than those of the *group-2* symbols. The recognition errors are mainly because (i) the character font is quite different from the commonly used fonts; (ii) the quality of the documents' paper is poor; and (iii) poor print quality.

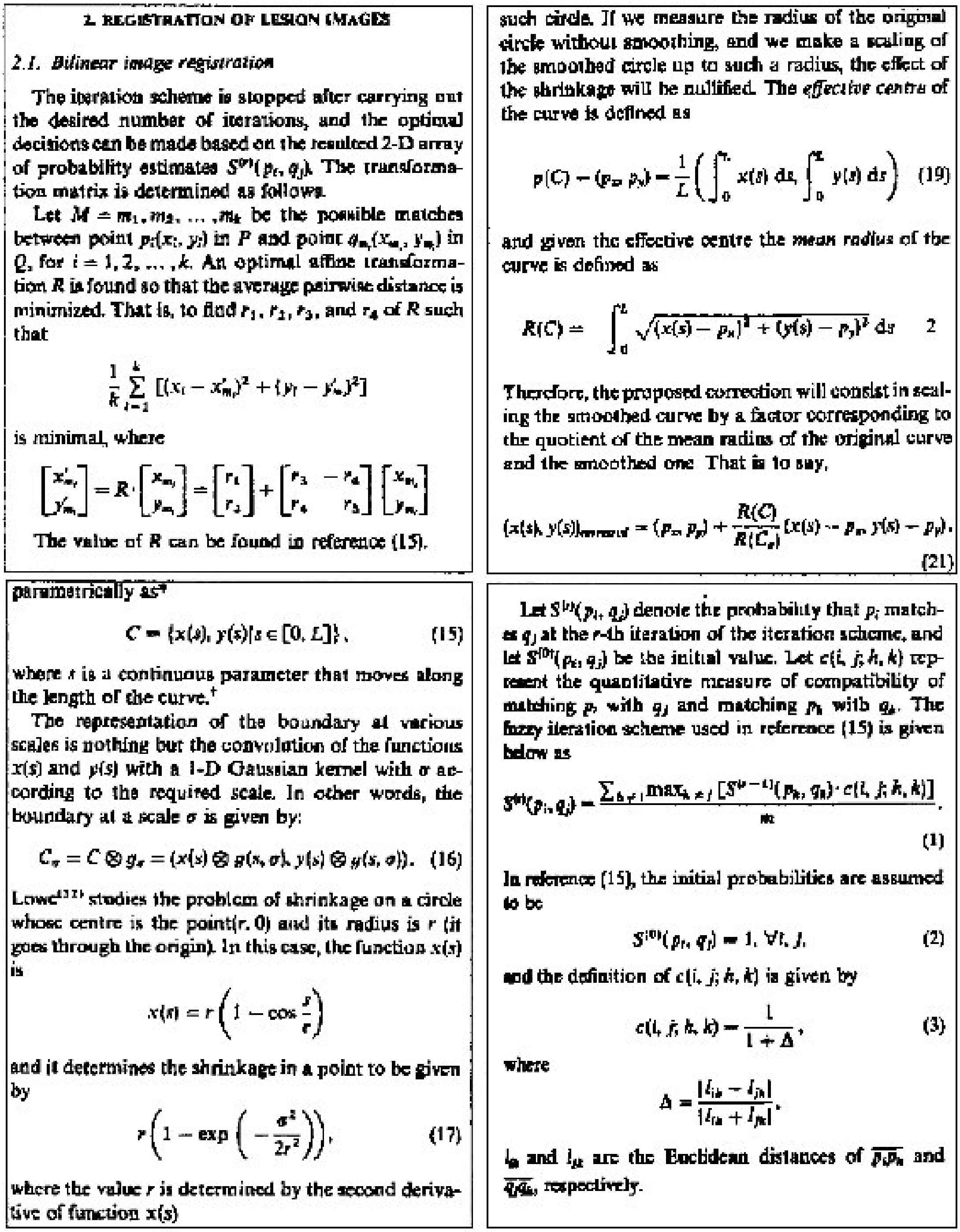
The arrangement of symbols is somewhat difficult. It is even more difficult to quantify the success rate for the arrangement of symbols. Out of 140 MEs considered for testing the system, 113 MEs are coded (or arranged) properly. This shows an overall 80.71% success rate in processing MEs. However, the performance of the symbol arrangement module should not be judged in this way, because an incorrect arrangement of only one or two symbols changes the meaning of the expression drastically. For a wrongly arranged ME, we measure the *symbol placement error*, which

**Table 3.** ME identification results

| ME type | #MEs | Correct detection | Wrong detection | #False detection |
|---|---|---|---|---|
| Embedded | 60 | 64 | 6 | 3 |
| Separately printed | 80 | 78 | 2 | nil |
| Total | 140 | 132 | 8 | 3 |

**Table 4.** Symbol recognition results (results taken on 140 expressions)

| Symbol type | #Symbols | Correct recognition | Mis-recognition | Rejection |
|---|---|---|---|---|
| *Group-1* | 942 | 924 | 15 | 3 |
| *Group-2* | 1530 | 1486 | 36 | 8 |
| Total | 2472 | 2410 | 51 | 11 |



2. REGISTRATION OF LESION IMAGES

*2.1. Bilinear image registration*

The iteration scheme is stopped after carrying out the desired number of iterations, and the optimal decisions can be made based on the resulted 2-D array of probability estimates $S^{(r)}(p_i, q_j)$. The transformation matrix is determined as follows.

Let $M = m_1, m_2, \ldots, m_k$ be the possible matches between point $p_i(x_i, y_i)$ in $P$ and point $q_{m_i}(x_{m_i}, y_{m_i})$ in $Q$, for $i = 1, 2, \ldots, k$. An optimal affine transformation $R$ is found so that the average pairwise distance is minimized. That is, to find $r_1$, $r_2$, $r_3$, and $r_4$ of $R$ such that

$$\frac{1}{k}\sum_{i=1}^{k}\left[(x_i - x'_{m_i})^2 + (y_i - y'_{m_i})^2\right]$$

is minimal, where

$$\begin{bmatrix} x'_{m_i} \\ y'_{m_i} \end{bmatrix} = R \cdot \begin{bmatrix} x_{m_i} \\ y_{m_i} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} + \begin{bmatrix} r_3 & -r_4 \\ r_4 & r_3 \end{bmatrix}\begin{bmatrix} x_{m_i} \\ y_{m_i} \end{bmatrix}$$

The value of $R$ can be found in reference (15).

parametrically as*

$$C = \{x(s), y(s) | s \in [0, L]\}, \qquad (15)$$

where $s$ is a continuous parameter that moves along the length of the curve.†

The representation of the boundary at various scales is nothing but the convolution of the functions $x(s)$ and $y(s)$ with a 1-D Gaussian kernel with $\sigma$ according to the required scale. In other words, the boundary at a scale $\sigma$ is given by:

$$C_\sigma = C \otimes g_\sigma = (x(s) \otimes g(s, \sigma), y(s) \otimes g(s, \sigma)). \quad (16)$$

Lowe[(12)] studies the problem of shrinkage on a circle whose centre is the point(r, 0) and its radius is $r$ (it goes through the origin). In this case, the function $x(s)$ is

$$x(s) = r\left(1 - \cos\frac{s}{r}\right)$$

and it determines the shrinkage in a point to be given by

$$r\left(1 - \exp\left(-\frac{\sigma^2}{2r^2}\right)\right), \qquad (17)$$

where the value $r$ is determined by the second derivative of function x(s)

such circle. If we measure the radius of the original circle without smoothing, and we make a scaling of the smoothed circle up to such a radius, the effect of the shrinkage will be nullified. The *effective centre* of the curve is defined as

$$p(C) = (p_x, p_y) = \frac{1}{L}\left(\int_0^L x(s)\,ds, \int_0^L y(s)\,ds\right) \quad (19)$$

and given the effective centre the *mean radius* of the curve is defined as

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

Therefore, the proposed correction will consist in scaling the smoothed curve by a factor corresponding to the quotient of the mean radius of the original curve and the smoothed one. That is to say,

$$(x(s), y(s))_{\mathrm{corrected}} = (p_x, p_y) + \frac{R(C)}{R(C_\sigma)}(x(s) - p_x, y(s) - p_y). \quad (21)$$

Let $S^{(r)}(p_i, q_j)$ denote the probability that $p_i$ matches $q_j$ at the $r$-th iteration of the iteration scheme, and let $S^{(0)}(p_i, q_j)$ be the initial value. Let $c(i, j; h, k)$ represent the quantitative measure of compatibility of matching $p_i$ with $q_j$ and matching $p_h$ with $q_k$. The fuzzy iteration scheme used in reference (15) is given below as

$$S^{(r)}(p_i, q_j) = \frac{\sum_{h \neq i} \max_{k \neq j}\left[S^{(r-1)}(p_h, q_k) \cdot c(i, j; h, k)\right]}{m}. \quad (1)$$

In reference (15), the initial probabilities are assumed to be

$$S^{(0)}(p_i, q_j) = 1, \forall i, j, \qquad (2)$$

and the definition of $c(i, j; h, k)$ is given by

$$c(i, j; h, k) = \frac{1}{1 + \Delta}, \qquad (3)$$

where

$$\Delta = \frac{|l_{ih} - l_{jk}|}{|l_{ih} + l_{jk}|},$$

$l_{ih}$ and $l_{jk}$ are the Euclidean distances of $\overline{p_i p_h}$ and $\overline{q_j q_k}$, respectively.

Fig. 10. Some typical test documents.

is defined as the number of symbols not properly arranged for that ME. It is observed that the *symbol placement error* lies between 2 and 4, while the average number of symbols per ME is 17.65. For details, see Table 5.

The errors in the symbol arrangement phase are due to the following factors: (i) mis-recognition (or rejection) of the symbols during the symbol recognition phase; and (ii) incorrect interpretation of the ME structure, or error in identification of the logical relationship among a group of symbols. The first factor is termed as a *recognition error*, and the second as a *parsing error*. The effect of these two types of errors is shown in Table 6. Since some of the MEs are affected by both errors, the total number of MEs affected by these errors is not 27 as it is in Table 5.

An example where our system fails to analyse the ME properly is $\Pi_i a_i$. Here, the system misinterprets $a_i$ as a

**Table 5.** Symbol arrangement results

| #ME | #Properly arranged | #Wrongly arranged | #MEs with N symbol placement errors | | | | |
|---|---|---|---|---|---|---|---|
| | | | N = 1 | N = 2 | N = 3 | N = 4 | N > 4 |
| 140 | 113 | 27 | 2 | 7 | 8 | 7 | 3 |

**Table 6.** Effect of recognition and parsing errors

| Error type | #Symbols affected | #MEs affected |
|---|---|---|
| Recognition error | 62 | 23 |
| Parsing error | 27 | 14 |

superscript of $i$ of the product sign, and codes the ME as the product over $i^{ai}$. Similarly, our system represents $\alpha^{h}/_{w}$ as a quantity that is $\alpha$ times $h$ divided by $w$ ($\alpha^{h}/w$), rather than $\alpha$ raised to power $h$ divided by $w$.

In some cases, a symbol recognition error leads to an incorrect arrangement. As a simple example, the symbol 'θ' in $\theta_i$ is occasionally mis-recognised as the numeral zero ('0'). Since no numeral can be subscripted, the variable is incorrectly interpreted as '$0_i$' (i.e. zero into $_i$). Here a warning can be issued that this is a meaningless interpretation. Such a warning can be issued in all situations.

## 8. CONCLUSIONS

In this paper, we have presented a system for processing mathematical expressions in printed documents. The approach is built upon the structural features and the formats of the MEs found in technical documents. The method of finding expressions in a document offers the option of creating a database of mathematical expressions after scanning a large volume of technical documents. To arrange the recognised symbols, we use their bounding-box coordinates, size information and the coordinates of the centroids, and apply some predefined rules to form meaningful symbol groups. These 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.

Object-oriented methodologies have been used to implement the system. This makes the maintenance and enhancement of the system easier than the traditional procedure oriented approach. Reusability is another advantage of the proposed system.

As an extension of the present study, we are in the process of designing an approach to automate the system performance evaluation, and to do the comparison among the different systems proposed for recognising MEs. In this context, a database containing a considerable number of documents of mathematical expressions with proper groundtruth would be very helpful. The UW-III English/Technical Document Image Database (prepared by the Intelligent Systems Laboratory at the Department of Electrical Engineering, University of Washington) contains 25 pages of mathematical formulae and groundtruth in XFIG and LaTeX. This could be a starting point for evaluating the system performance.

## References

1. Blostein D, Grbavec A. Recognition of mathematical notation. In: H Bunke, PSP Wang (eds). Handbook of Character Recognition and Document Image Analysis. World Scientific, 1997: 557–582
2. Anderson RH. Syntax-directed recognition of handprinted 2-D mathematics. PhD Dissertation, Harvard University, Cambridge, MA, 1968
3. Belaid A, Haton J. A syntactic approach for handwritten mathematical formula recognition. IEEE Trans Pattern Analysis and Machine Intelligence 1984; 6(1):105–111
4. Chang SK. A method for the structural analysis of 2-D mathematical expressions. Information Sciences 1970; 2(3):253–272
5. Okamoto M, Twaakyondo H. Structure Analysis and Recognition of Mathematical Expressions. IEEE Press 1995: 430–437
6. Okamoto M, Miyazawa H. An experimental implementation of a document recognition system for papers containing mathematical expressions. In: Structured Document Image Analysis. Springer-Verlag, 1992: 36–53
7. Grbavec A, Blostein D. Mathematics recognition using graph rewriting. Proceedings of Third International Conference on Document Analysis and Recognition 1995: 417–421
8. Larvirotte S, Pottier L. Mathematical formula recognition using graph grammar. Proceedings of SPIE 1998; 3305
9. Lee H, Lee M. Understanding mathematical expressions using procedure-oriented transformation. Pattern Recognition 1994; 27(3):447–457
10. Faure C, Wang Z. Automatic perception of the structure of handwritten mathematical expressions. In: Computer Processing of Handwriting. World Scientific, 1990: 337–361
11. Chou P. Recognition of equations using a two-dimensional context-free grammar. Proceedings of SPIE Visual Communication and Image Processing IV 1989: 852–863
12. Berard EV. Essays on Object-Oriented Software Engineering. Addison-Wesley, 1993
13. Sengupta P, Chaudhuri BB. Object-Oriented Programming: Fundamentals and Applications. Prentice Hall, 1998
14. Rambaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. Object-Oriented Modeling and Design. Prentice-Hall, 1991
15. Bellinzona R, Gugini MG, Pernici B. Reusing Specifications in OO Applications. IEEE Software 1995: 65–75

16. LATEX: A document Presentation System. Addison-Wesley, 1986

17. Microsoft® Word 97: Copyright © 1983–1996. Microsoft Corporation, USA

18. Chaudhuri BB, Garain U. An approach for processing Mathematical Expressions in printed document. In: Seong-Whan Lee, Y. Nakano (eds) Lecture Notes in Computer Science, Vol. 1655, Document Analysis Systems: Theory and Practice. Springer-Verlag, 1998: 310–321

19. Lee H, Wang J. Design of a mathematical expression recognition system. Proceedings of 3rd International Conference on Document Analysis and Recognition 1995: 1084–1087

20. Toumit JY, Garcia-Salicetti S, Emtoz H. A hierarchical and recursive model of mathematical expressions for automatic reading of mathematical documents. Proceedings of Fifth International Conference on Document Analysis and Recognition (ICDAR) 1999: 119–122

21. Kacem A, Belaid A, Ahmed MB. EXTRAFOR: automatic EXTRAction of mathematical FORmulas. Proceedings of Fifth International Conference on Document Analysis and Recognition (ICDAR) 1999: 527–530

22. Martin W. Computer input/output of mathematical expressions. Proceedings of 2nd Symposium on Symbolic and Algebraic Manipulations 1971; 78–87

23. Bokser M. Omnidocument Technologies. Proceedings of the IEEE 1992; 80(7):1066–1079

24. Ho T, Baird HS. Perfect Metrics. Proceedings of Int Conf on Document Analysis and Recognition 1993: 593–597

24. Kanungo T, Haralick RM, Phillips I. Non-linear local and global document degradation models. Int Journal of Imaging Systems and Technology 1994; 5(4):220–230

**Professor B. B. Chaudhuri** received his BSc (Hons), BTech and MTech degrees from Calcutta University, India in 1969, 1972 and 1974, respectively, and his PhD degree from the Indian Institute of Technology, Kanpur in 1980. He joined the Indian Statistical Institute in 1978, where he served as the Project Co-ordinator and Head of National Nodal Center for Knowledge Based Computing. Currently, he is the head of Computer Vision and Pattern Recognition Unit of the institute. His research interests include pattern recognition, image processing, computer vision, natural language processing and digital document processing, including OCR. He has published about 200 research papers in reputed international journals, and has authored the books entitled *Two Tone Image Processing and Recognition* (Wiley Eastern, 1993) and *Object Oriented Programming: Fundamentals and Applications* (Prentice Hall, 1998). He was awarded the Sir J. C. Bose Memorial Award for the best engineering science oriented paper in 1986, the M. N. Saha Memorial Award (twice) for the best application oriented papers in 1989 and 1991, the Homi Bhabha Fellowship award in 1992 for OCR of Indian Languages and computer communication for the blind, the Dr. Vikram Sarabhai Research Award in 1995 for his outstanding achievements in the fields of Electronics, Informatics and Telematics, and the C. Achuta Menon Prize in 1996 for computer-based Indian language processing. He worked as a Leverhulme visiting fellow at Queen's University, UK in 1981–82, as a visiting scientist at GSF, Munich, and as a guest in the faculty at the Technical University of Hannover during 1986-88, and again in 1990–91. He is a Senior Member of the IEEE, the membership secretary (Indian Section) of the International Academy of Sciences (India), a Fellow of the International Association of Pattern Recognition, a Fellow of the National Academy of Sciences (India), a Fellow of the Institution of Electronics and Telecommunication Engineering (India), and a Fellow of the Indian National Academy of Engineering. He is serving as an associate editor of *Pattern Recognition*, *Pattern Recognition Letters* (Elsevier Science) and *VIVEK*, as well as guest editor of a special issue of the *Journal IETE* on Fuzzy Systems.

**U. Garain** received both of his BE and ME in computer science from Jadavpur University, Calcutta in 1994 and 1997, respectively. He worked in two multinational software firms for one and a half years. Later, he joined as research personnel at the Indian Statistical Institute, Calcutta, where he is now on the full time faculty. He is one of the key scientists involved in the development of a bilingual (Devnagari & Bangla) OCR system. His areas of interest include digital document processing, an OCR system development for Indian language scripts, document data compression, etc. Over the last two years, Mr Garain has published several technical papers in reputed international journals and conferences.

*Correspondence and offprint requests to*: Prof. B.B. Chaudhuri, Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700 035, India. Email: bbc@www.isical.ac.in