Exploring Image Recognition in Deciphering Mathematical Symbols

MATHS FOR AI, TP – 1 2023-2024

GLAWIN JOHN ALVA

1. Introduction

Scientific papers frequently contain mathematical expressions. While OCR (Optical Character Recognition) systems are proficient at handling regular text they face difficulties when it comes to recognizing and processing mathematical expressions. However automatic recognition of these symbols is crucial for converting documents into Digital formats.

The realm of expressions involves a variety of symbols, such, as numbers, Greek and English letters, operators, and special characters. However, the absence of symbols poses difficulties due to differences in size, style, and the limited distinguishing characteristics among symbols. These intricacies make it challenging to develop approaches, for segmenting and identifying symbols within expressions.

Over the last three decades, academics have achieved tremendous advances in the field of mathematical expression recognition. Various strategies have been investigated, with an emphasis on symbol separation and interpretation in both offline and online situations. A thorough investigation [1] revealed the parallels and contrasts between these disparate systems. Suzuki and Tamari created INFTY, a novel document OCR system that employs two independent recognition engines [3]. One engine focuses on language, while the other in mathematical expressions. For text segmentation, the INFTY system incorporates complex techniques such as cutting profiles and labelling components. Fateman also made an important contribution by proposing a prototype technique [2]. This method is based on template matching and use the Hausdorff distance to successfully translate typeset mathematical expressions with high precision. Lee and Wang [7] developed a technique that improves the process of dividing and analysing text and phrases in documents. Their system recognizes components as symbols and employs precise recognition criteria, resulting in a complete way to dealing with textual and mathematical information in documents. Each of these advancements reflects a distinct approach to the problem of mathematical expression recognition, thus improving the subject and increasing the technology's possible uses.

2. System for Recognizing Mathematical Expressions Overview

The process of recognizing expressions is divided into four main parts; (i) finding where the expressions are located within a document, (ii) identifying the individual symbols in these expressions (iii) analysing how the expressions are arranged on the page and (iv) converting the expressions into formats that can be edited. Initially the system detects expressions that are embedded in the document. Then it breaks down these expressions into symbols and extracts specific characteristics of these symbols to accurately identify them. The third step involves creating a structure tree that represents how the symbols in an expression are arranged in two dimensions, based on their mathematical structure, and meaning. Finally, the system converts these expressions into formats, like LaTeX or MathXML so that they can be easily manipulated.

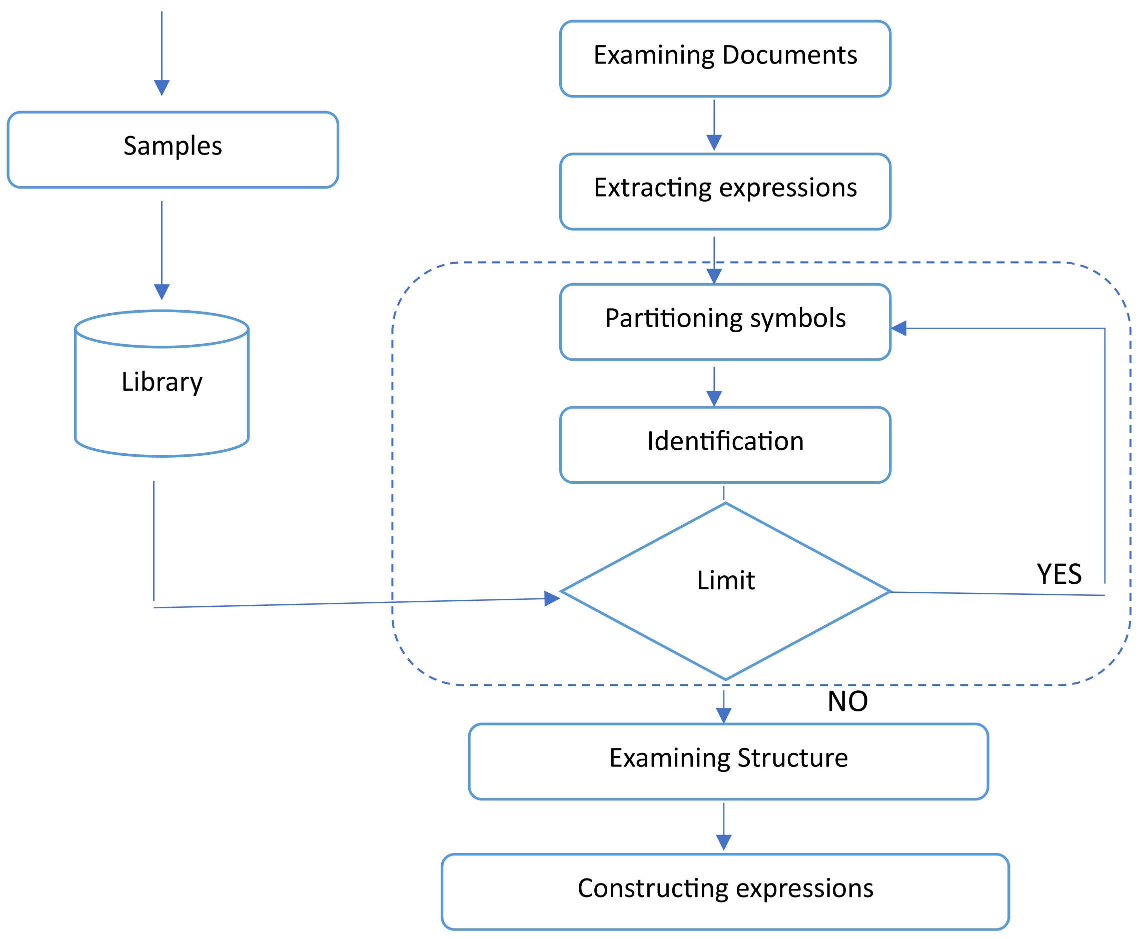

Diagram 1 depicts the system block diagram.

As illustrated in Diagram1, the module is divided into two procedures as segmenting and recognizing symbols.

3. Mathematical symbol Identification

Recognizing symbols accurately plays a role, in identifying mathematical expressions effectively. It has an impact on the performance of the recognition task. However, this task becomes challenging due to factors such as fonts, typefaces, sizes, and the complex arrangements of symbols in two dimensions. This intricacy causes symbol segmentation to become more challenging than text segmentation. Commercial OCR systems face difficulties when it comes to processing mathematical expressions and often require specialized method. In the following sections we will dig into these processes. Provide an explanation for each.

I. Symbol Segmentation

The technique of projection-profile cutting relies on the empty spaces between lines or characters within a document for segmenting images [5] [6]. This method is typically effective for characters that are arranged linearly. Nonetheless, with mathematical expressions, certain symbols that encompass other symbols within their area, such as fractions, ('$\sqrt{\quad}$'), cannot be accurately separated using this approach.

The connected components labelling algorithm identifies all distinct entities within a document. It effectively segments symbols that house other symbols in their domain, like ('$\sqrt{\quad}$'), or symbols that partially merge with others, such as a radical over a variable (e.g., 'P_e'). However, symbols composed of disjointed elements, like the '%' or complex script characters such as '«', may end up being divided into multiple segments.

In our study, we employ both complementary segmentation strategies to efficiently separate symbols within mathematical expressions. The outcomes of symbol recognition are then utilized to instruct symbol segmentation process serves as a decision-making step to ascertain the correctness of symbol isolation and to determine if re-segmentation is necessary. Certain symbols, like 'i', '=', and ':', which consist of multiple parts, tend to be split into several segments through the previously mentioned methods. To counter this, smaller components are amalgamated with the closest larger component to facilitate accurate symbol recognition.

The segmentation procedure is bifurcated into two distinct steps which are (a) Segmentation using projection profile cutting (b) segmentation based on connected components labelling. Blocks with symbols erroneously segmented in the initial step are subjected to re-segmentation in the subsequent step based on the feedback from symbol recognition outcomes. The methodology is as follows:

a) Split the statement into discrete symbol blocks using recursive projection-profile cutting, merging smaller pieces in the process. Proceeding to symbol recognition for each symbol block.
    - If the recognition is deemed satisfactory against a predefined threshold, move to the next symbol block.
    - If not, the symbol block requires re-segmentation.

b) Employ the connected components labelling algorithm to further split the symbol block into smaller segments, again merging diminutive fragments.

c) Undertake recognition of these newly segmented symbol blocks and repeat the cycle starting from step 2 for any additional blocks.

II. Identification of symbols

In the field of symbol recognition, a variety of methods have been suggested over the years, including template matching, structural analysis, neural networks, and statistical approaches, each capable of identifying symbols with their unique strengths. In this study, a strategy involving categorization followed by recognition is employed, ensuring efficiency and accuracy in symbol recognition. By specifically selecting features and directional line elements to represent symbols [8], this approach emphasizes the precise capture of each symbol's unique aspects. The process begins with the division of the symbol [2] into segments, which are then analysed to extract their characteristics, a crucial step for accurate identification.

I. Preprocessing

Prior to feature extraction, it's crucial to smooth out any imperfections in the symbols and standardize their size to a uniform 32x32 pixel format [2].

II. b. In the initial stage of classification, known as Coarse Classification, we categorize the symbols by examining their aspect ratio (AR) and peripheral features [2].

(1) Feature Extraction

Let $(i_e - i_S) \times (j_e - j_S)$ be a symbol which is represented by its characteristics that classifies.

$$D(i_e, i_S, j_e, j_S, L, R, T, B) \qquad (1)$$

The peripheral features of a symbol extracted from all the directions are represented as left = L, right = R, top = T and bottom = B directions respectively. The calculation, for L is as follows:

$$L = \sum_{i=i_s}^{i_e} \sum_{j=j_s}^{j_e} \left(f(i,j) \wedge (\text{ flag } = 0)\right) \qquad (2)$$

In this scenario we start with a pixel. Count the number of times it changes to a black pixel. As we scan each line, we begin with a flag value of 0. R, T and B are calculated in the same order.

Therefore, aspect rate is: $AR = (i_e - i_s)/(j_e - j_s)$

(2) Classifying symbols

The classificatory features are utilized by a minimum distance classifier to determine the classification of each symbol. Symbol X is considered as belonging to class i if

$$d(f_t, f_i) < d\left(f_t, f_j\right) \forall i, j and i \neq j$$

and

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

where f_t is the feature vector of symbol X, f_i is the feature vector of class i, and δ is a approach of distance according to the experiment.

We use the feature vector D divide the mathematical symbol set into 20-25 category candidates.

(3) Fine recognition

During the recognition stage we employ features called line elements to enhance the accuracy of symbol recognition. These features are particularly effective, in identifying symbols that have strokes and similar font styles as they emphasize variations in local details. Here's how we extract these features.

1) We resize the symbols to a 32x32 dimension. Divide them into four rows and four columns creating blocks of equal size. Each block contains 8x8 pixels.

2) Within each block we further divide it into four nested sub blocks with dimensions of 8x8, 6x6, 4x4 and 2x2 pixels.

By using this process, we are able to refine the recognition outcome, for symbols and improve their accuracy.

(3) A 4-dimensional directional line element feature vector $X(x_0, x_1, x_2, x_3)$ is extracted for each sub block, where x_0, x_1, x_2, x_3 represent the number of line element in horizontal, vertical and two diagonal directions. x_0, x_1, x_2, x_3 are computed using the subsequent formula:

$$x_i = x_i^1 + x_i^2 + x_i^3 + x_i^4$$

where $i = 0,1,2,3$, represents the four directions. $x_i^1 + x_i^2 + x_i^3 + x_i^4$ is the sum of the number of line elements in four sub-blocks in the i direction. The feature vectors of each sub block which consist of 4-line elements are combined to form a 64 dimensional feature vector.

One-dimensional statistics are the only ones used by the minimum distance classifier. As a result, its ability to represent the pattern distribution in feature space is constrained. Here, we adjust the symbol recognition result using a Euclidean distance with deviation (EDD) classifier that makes use of two-dimensional statistics. Let $Y = (y_1, y_2, \cdots, y_n)$ be the n-dimensional feature vector of symbol Y. $M = (m_1, m_2, \cdots, m_n)$ is a common representation used for a symbol is a feature vector. Now let’s take a look, at the definition of the EDD.

$$d_{\min}(Y) = \sum_{i=1}^{n} [\max(0, |y_i - m_i| - \theta \cdot \sigma_i)]^2$$

where σ_i is the standard symbol's mean squared deviation of the kth characteristics. θ represents a constant.

4. Mathematical Example -:

Symbol recognition involves categorizing symbols by analysing their features and calculating the distance, with deviation (EDD). To better understand the concepts lets provide some examples for the equations mentioned in the given excerpt.

Equation (1) 'D (ie, is, je, js, $L, R, T, B)'$
Let's define a symbol represented by the coordinates of its bounding box: '(ie, is, and $'\mathbf{B} = \mathbf{3}'$. The classificatory features $'\mathbf{D}'$ can be calculated, but the specific function $^{\mathbf{f}}(\mathbf{i}, \mathbf{j})'$ is not provided.

Equation (2) 'L $= \Sigma\Sigma(f(i,j) \wedge (flag == 0))'$
We have a row of pixels represented as $'[\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}]$ '. Scanning this line, we count the number of changes from 0 to 1, which occurs three times. Therefore, $L = 3$.

Equation (3) $^{A}AR = (ie - is)/(je - js)'$
Using the bounding box coordinates, the aspect ratio 'AR' is:

$$AR = \frac{(4-1)}{(8-3)} = \frac{3}{5} = 0.6$$

Equation (4) $d(ft, fi) < d(ft, fj) Vi, j$ and $i \neq j$ and $d(ft, fi) < \delta^{\circ}$

With feature vectors ' $\mathbf{fi} = [\mathbf{1}, \mathbf{2}, \mathbf{3}]'$, $'\mathbf{fj} = [\mathbf{2}, \mathbf{3}, 4]'$, symbol feature ' $\mathbf{ft} = [1.5, 2.5, 3.5]'$, and threshold $'\delta = 0.5'$, we calculate the Euclidean distance between ' $\mathbf{ft}$ ' and each ' $\mathbf{fi}$ ' and ' $\mathbf{fj}$ ' to determine class membership.

Equation (5) $'xi = xi1 + xi2 + xi3 + xi4'$
For a sub-block, let's assume the number of line elements in four directions are 'xi1 = 3$'$, $'$ 'xi2 $= 2'$, 'xi3 $= 4'$, 'xi4 $= 1'$. Then, for direction ' $i = 0$ ':

$$x_0 = 3 + 2 + 4 + 1 = 10$$

Equation (6) $'\text{dmin}\ (Y) = \Sigma\max(0, yi - mi - \theta \cdot \sigma i)\text{^}2'$
Given feature vector $'\mathbf{Y} = [\mathbf{3}, \mathbf{4}, \mathbf{5}]'$, standard feature vector $'\mathrm{M} = [\mathbf{2}, \mathbf{5}, \mathbf{4}]$ ', mean squared deviation ' $\boldsymbol{\sigma i} = [\mathbf{0.5}, \mathbf{1}, \mathbf{0.5}]'$, and constant $^{*}\boldsymbol{\theta} = \mathbf{1}$ ', we compute ' dmin $(\mathbf{Y})$ ' for each element and sum them up:

- For ' $^{\text{i=1}}$ ': $\max(0{,}3 - 2 - 1 \times 0.5)^2 = \max(0{,}0.5)^2 = 0.25$
- For 'i=2': $\max(0{,}4 - 5 - 1 \times 1)^2 = \max(0, -2)^2 = 0$

Thus, dmin $(Y) = 0.25 + 0 + 0.25 = 0.5$

These numerical examples are based on assumed values and illustrate how the equations might be applied in practice.

5. Conclusion:

In summary, the provided numerical examples illustrate the process of recognizing and classifying symbols in mathematical expressions. Starting with the extraction of peripheral features (Equation (1) and (2)), to calculating the aspect ratio (Equation (3)), and then using a minimum distance classifier (Equation (4)), the examples show how each step contributes to the accuracy of the recognition system. Finally, the Euclidean distance with deviation (Equation (6)) fine-tunes the classification, ensuring precise symbol identification. This detailed approach emphasizes the nature and exactness needed in optical character recognition systems when it comes to recognizing symbols.

6. References

[1] K.F. Chan and D.Y. Yeung, "Mathematical expression recognition: a survey", International Journal on Document Analysis and Recognition, (2000).

[2] Xue-Dong Tian, Hai-Yan Li, Xin-Fu Li and Li-Ping Zhang, "Research on Symbol Recognition for Mathematical Expressions," First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06), Beijing, (2006).

[3] M. Suzuki, F. Tamari, R. fukada, S. Uchida and T. Kanahori, "INFTY—An Integrated OCR System for Mathematical Documents", the 2003 ACM symposium on Document engineering table of contents, France, (2003).

[4] R. Fateman, T. Tokuyasu, B. Berman, and N. Mitchell, "Optical character recognition and parsing of typeset mathematics", Journal of Visual Communication and Image Representation, (1996).

[5] M. Okamoto and B. Miao, "Recognition of mathematical expressions by using the layout structures of symbols", In ICDAR, (1991).

[6] M. Okamoto and A. Miyazawa, "An experimental implementation of a document recognition system for papers containing mathematical expressions", Structured Document Image Analysis, Berlin, (1992).

[7] H.J. Lee and J.S. Wang, "Design of mathematical expression recognition system", In Proceedings of ICDAR'95, Canada, (1995).

[8] WANG. Hua and DING. Xiaoqing, "An Algorithm for Multi-font Printed Tibetan Character Recognition", Computer Engineering, (2004).