On Machine Understanding of Online Handwritten Mathematical Expressions

Utpal Garain B. B. Chaudhuri
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
203, B. T. Road, Kolkata, India
E-mail: {utpal, bbc}@isical.ac.in

Abstract

This paper aims at automatic recognition of online handwritten mathematical expressions written on an electronic tablet. The proposed technique involves two major stages: symbol recognition and structural analysis. A multiple-classifier consists of both parametric and non-parametric classifier has been used for recognition of symbols. Parametric classifier is based on Hidden Markov Model (HMM), whereas, non-parametric classifier uses Nearest Neighbor classification scheme. Structural analysis uses several online and offline features to identify the spatial relationships among symbols. A context free grammar has been designed to convert input expressions into their corresponding Latex strings. Contextual information has been used to correct several errors occurring at both recognition and structural analysis stage. A new method for evaluating performance of the proposed systems has been formulated. Experiments on a dataset of considerable size show high efficiency of the proposed system.

1. Introduction

In this paper, we propose a system that understands handwritten Mathematical Expressions (hereafter referred as MEs). Data is captured online by a tablet/stylus or a light pen connected to a computer. Processing involves two major stages, namely, symbol recognition and structural analysis.

There exist some research efforts [1, 2] for recognition of MEs. Previous studies on online recognition show limitations concerning the number of recognizable structures. Mostly a single classifier is designed where contextual information is rarely used to improve the recognition score. Structure analysis has also been tackled more or less in an offline manner and as a result, online spatio-temporal information has not been used efficiently. Moreover, testing of the proposed systems is based on some non-standard dataset, and therefore, the nature of MEs considered in different studies is not uniform. The absence of any well-accepted technique for performance evaluation of such systems is another major handicap.

In this paper, we describe a system for understanding online handwritten MEs and the measures taken to overcome the above-mentioned deficiencies. Rest of the paper describes the details of our proposed system.

2. Data acquisition and preprocessing

The input is handwritten strokes drawn on an electronic data tablet. The raw data goes through several preprocessing steps (like 4-8 connected pixel conversion, interpolation of missing points, smoothing, etc.) to remove variations (due to noise and uncontrolled pen movement) that would otherwise complicate the recognition.

Let the digitizer output be $(pt[l])_{l=1}^{N} \in \Re^2 \times \{0,1\}$, where $pt[l]$ is the pen position having x and y-coordinates: ($pt[l].x$) and ($pt[i].y$). Let, $s = pt[i] - pt[j]$, where $pt[i]$ and pt[j] are two consecutive pen points. Now i-th point, ($pt[i]$) is retained with respect to j-th point, (pt[j]) if the following condition is satisfied:

$$(s.cx)^2 + (s.cy)^2 > m^2 \tag{1}$$

where $s.cx = pt[i].x - pt[j].x$ and $s.cy = pt[i].y - pt[j].y$. The parameter, m is empirically chosen. Note that all repeated points are removed for m = 0 where as, m = 1 implements 4-to 8-connectivity conversion.

3. Symbol Recognition

Consider the way to teach a child to write. S/he is advised to down the pen at some position, make straight/curved pen movement in a particular direction, create loops when needed and lift the pen at some other position. Pen down position for the next stroke is also mentioned and s/he follows such instructions until the character is complete. Moreover, the children are also taught the relative lengths of the different parts of a stroke. In our approach, these aspects are captured and used as features described below.

For each stroke *angle variation* between two successive points is given by

$$\varphi_i = \pi + sign * \cos^{-1}\left(r[i].cx / \partial l_i \right) \tag{2}$$

where $r[i]=pt[i]-pt[i-1]$, $r[i].cx = pt[i].x - pt[i-1].x$, $r[i].cy = pt[i].y - pt[i-1].y$, $\partial l_i = \sqrt{(r[i].cx)^2 + (r[i].cy)^2}$ and *sign* = -1 if $r[i].cy < 0$, otherwise *sign* = 1.

In reality, instead of angle variation information, direction change information is taught to a child. That is why we convert each φ_i into a direction code d_i. (an integer) following a 8-direction Freeman coding [3] as follows:

$$d_i = \left(\left((\text{int})\left(\frac{8\varphi_i}{\pi}\right)+1\right)\text{mod}\,16\right) \div 2 \tag{3}$$

where **mod** is the modulus operator and **int** return integer part of a real number. Next, we normalize ∂l_i, which represents local trajectory length. This is done in a straightforward manner:

$$\partial l_i = (\partial l_i) / \left(\sum_{j=1}^{N-1} \partial l_j\right) \tag{4}$$

3.1. Design of Classifiers

ME symbols show wide variations in their shapes and sizes. To tackle these large variability two different classifiers therefore are used in our system. Classifier 1 is based on feature template matching [4] and uses a nearest neighbour (NN) classification scheme for recognizing the characters, whereas, Classifier 2 implements HMM [5] for classification and hence, parametric in nature.

3.1.1. Classifier 1. Feature vector used by this classifier is defined by the tuple,

$$(d_i, \partial l_i) \qquad i = 1,2,\ldots,N-1 \tag{5}$$

Let the feature vectors for T (stroke to be recognized) and S (stored prototype) be $f_T = \left(d_k^T, \partial l_k^T\right)_{k=1}^{K}$ and $f_S = \left(d_j^S, \partial l_j^S\right)_{j=1}^{J}$, respectively, where $\sum_k \partial l_k^T = \sum_j \partial l_j^S = 1$. For $K = J\ (= N)$ T and S can be compared by

$$J(T,S) = \sum_{i=1}^{N} 4 - \left|4 - \left|d_i^T - d_i^S\right|\right| \tag{6}$$

Note that in 8-directional coding the maximum difference between two successive direction codes can be 4. However, in reality, K is rarely equal to J, hence, the condition $K = J = N$ is hardly satisfied. So, (6) is modified by defining two more measures as follows.

$$\Delta L_k^T = \sum_{i=1}^{k} \partial l_i^T \qquad \text{and} \qquad \Delta L_j^S = \sum_{i=1}^{j} \partial l_i^S$$

Next, we sort the union of $\left(\Delta L_k^T\right)_{k=1}^{K}$ and $\left(\Delta L_j^S\right)_{j=1}^{J}$ together in increasing sequence. Let this sequence of numbers be $\left(\Delta L_r\right)_{r=1}^{R}$ where $\Delta L_R = 1$. Note that the direction codes $\left(d_k^T\right)$ of T and $\left(d_j^S\right)$ of S are constant over $\Delta L_r - \Delta L_{r-1}$. Now, we can re-define f_T and f_S as $\left(d_r^T, \partial l_r\right)$ and $\left(d_r^S, \partial l_r\right)$, respectively and the equation (6) is re-stated as

$$J(T,S) = \sum_{r=1}^{R} \partial l_r \left(4 - \left|4 - \left|d_r^T - d_r^S\right|\right|\right) \tag{7}$$

For any input stroke T, $J(T, S_i)$ is measured against all stored prototypes S_i and T is classified as S_i if $J(T, S_i) < J(T, S_j)$ *for all* $j \neq i$.

3.1.2. Classifier 2. This classifier uses a left-to-right HMM, $\lambda = (\pi, A, B)$ for recognition of strokes where parameters π, A and B denote *initial probability vector, transition* and *observation* matrix, respectively. Parameters are estimated as follows.

A number of observation sequences (or samples) are used to train the model corresponding to a particular stroke. Let there be η number of such sequences are available. Each sequence $O = O_1, O_2, \ldots, O_M$ consists of M observation symbols. Each (O_i) is presented in a two-dimensional vector, $(\varphi_i, \partial l_i)_{i=1}^{M}$ where φ_i and ∂l_i are given by (2) and (4), respectively. Each of the ηM observation vectors is mapped into one of the N clusters. Each cluster is defined by assigning a direction code d_i to φ_i and by labeling each segment length ∂l_i as short, medium, or long based on two predefined thresholds. As d_i can take one of the eight values (zero to seven) and φ_i can have one of three descriptions (short, medium, or long), hence, any observation O_i will be mapped into one of the 24 (N = 8 X 3) clusters. Each of these clusters forms a state (1 to N).

The Segmental K-means Algorithm [6] is used for training. Let $\hat{\lambda}_i = (\hat{\pi}_i, \hat{A}_i, \hat{B}_i)$ be denote the initial estimation of the model parameters for i-th stroke. Gaussian distribution is assumed for training vector for each state. To improve initial estimation, for each training sequence the optimal state sequence S^* (= $s_1, s_2, \ldots, s_M$) is found by using Viterbi Algorithm [7].

Once training is over, HMMs are obtained for every stroke (for some strokes there are more than one HMM are maintained). For a given observation sequence corresponding to a stroke, the probabilities of occurrence of the observation sequence is computed against each HMM, λ_i. Maximum of $P(O|\lambda_i)$ classifies the given observation as the i-th stroke.

3.1.3. Fusion of Classification Results. Each of the two classifiers described above returns three top choices for each input stroke. Final classification is done by employing a majority voting scheme that checks confidence of the two classifiers. However, we use a soft-decision making approach for stroke classification. Apart from the top choice, a number (varies from zero to two) of alternatives are generated for a stroke and used in structure analysis stage to resolve several ambiguities.

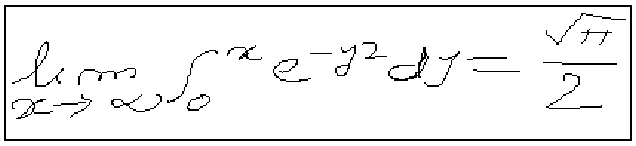

Fig. 1. An example taken from our database of online MEs.

4. Analysis of ME structure

The structural analysis consists of three stages: (i) *online processing*: it uses online spatio-temporal data as well as symbol size information to determine 2-D relation between a pair of symbols. Certain structures like square root sign, first level superscript/subscript, several limit structures etc. are identified online. Words like "*sin*", "*cos*", "*lim*", etc. are also formed at this stage. (ii) *Offline processing*: structures like fraction, limit, multi-level superscript/subscript, matrix, etc. are formed and converted into Latex strings based offline data analysis. (iii) *Compilation of Latex string*: latex strings generated at any stage are compiled and checked for their syntactic validity. Presence of any syntactic error in a string invokes further analysis, which tries to find another valid string using the alternatives provided by the recognizer.

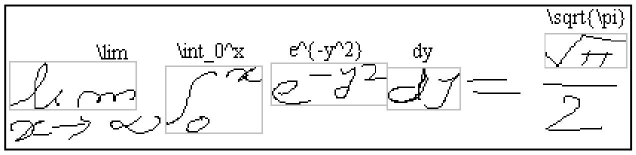

Fig. 2. Online recognition of 2-D structures.

The proposed approach is explained using an example ME shown in Fig. 1 for which all symbols except one are properly recognized. The top choice for "li" of "*lim*" is "*u*". Certain structures (rectangular boxes in Fig. 21) are recognized online and corresponding Latex strings are generated (as shown in Fig. 2).

Under offline processing, the input ME is initially divided into several vertical stripes based on the white spacing. Next, symbols within each vertical stripe are divided into several horizontal stripes. This goes on recursively until there is no more division possible within a stripe called *atomic* box. Atomic boxes for the ME under discussion are shown in Fig. 3, where each atomic box is tagged with a number.

Latex string is formed for each atomic box. For more than one symbol in a box identification of relations among symbols is achieved by a Context-Free Grammar (CFG), **G**. The presence of relatively small number of symbols inside an atomic box makes formation of Latex string easier. Next, a bottom-up parser [8] is invoked to find relationships among a pair of adjacent atomic boxes that were generated by dividing a larger box at the immediate higher level. The presence of any relation invokes merging of two adjacent boxes, and a new Latex string is generated. Such a merging is guided by the production rules (**PR**) of **G**. For instance, box 8, 9, and 10 of Fig. 3 are merged following **PR 8** corresponding to *fraction*. Some of the production rules (**PR**) are as follows.

PR 1: (s_{i}, s_{j}) → s_{i}_\{s_{j}\}; **PR 2**: (s_{i}, s_{j}) → s_{i}^{s_{j}}; **PR 3**: (s_{i}, s_{j}) → s_{i}{s_{j}}; **PR 4**: (s_{i}, s_{j}) → _ s_{i}s_{j}; **PR 5**: (s_{i}, s_{j}) → ^{s_{i}}{s_{j}}; **PR 6**: (s_{i}, s_{j}) → \begin{array }{c}s_{i} \\ s_{j}\end{array}; **PR 7**: (s_{i}, s_{j}) → \begin{array}{cc}s_{i} & s_{j}\end{array}; **PR 8**: (s_{i}, \hline, s_{j}) → \frac{s_{i}}{s_{j}}; **PR 9**: (\sigma, s_{i}) → \sum_ {s_{i}}; **PR 10**: (\sqrt, s_{i}) → \sqrt{s_{i}};

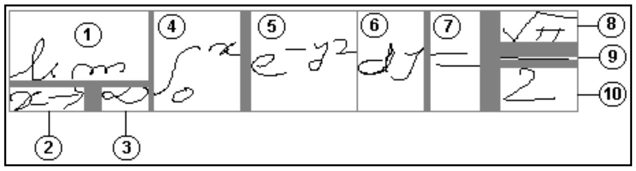

Fig. 3. Formation of atomic boxes for the ME in Fig. 3.

Several contextual information are used while generating a new Latex string by merging the boxes, e.g. box no. 4, 5, and 6 of Fig. 3 are merged and finally form a Latex string "\int_0^x e^{-y^2}dy". Here, "dy" seems as a multiplication of "d" and "y". However, presence of "\int" looks for its *differential* and converts the string into "\int_0^x e^{-y^2}\,dy", where "dy" conveys its actual meaning.

Moreover, Latex strings generated at any stage are checked for its syntactic validity. Any syntactic failure calls for immediate processing with available alternatives, e.g. in box 1 of Fig. 3, "*lim*" has been recognized as "*um*" which generates a Latex string "\um", that does not pass validation check. Hence, the system uses other alternatives provided at the symbol recognition module. If none of the alternatives generates a valid string, the system generates a string with the best choices for its constituent symbols and leaves it for manual correction at a later stage.

5. Experimental results and discussions

Test data contains 70 distinct MEs amongst which 60 are taken from the dataset used by T. V. Raman [9]. Expressions are selected from various branches of science and show wide variations in their structural complexity.

For each ME its corresponding correct Latex string is available. Ten MEs contains digits and letters of Devanagri/Hindi script. Twenty different writers are asked to write each of the 70 MEs twice. Hence, in total the database contains 2800 MEs. Out of total 24,675 symbols present in the database, 142 distinct character classes are identified. Symbols are grouped into eight different categories: I. Arabic numerals (10), II. Uppercase Roman letters, III. Lowercase Roman letters (26), IV. Greek letters (10), V. mathematical symbols (40), VI. Function words like "sin", "lim", etc. (10), VII. Hindi numerals (10), VIII. Hindi letters (10). 15,000 symbols have been used for training of the classifiers and testing has been done on 9,675 (24,675 - 15,000) symbols. The details of recognition results are presented in Table 1.

Table 1. Symbol recognition rate

Symbol category	Total symbols	Classifier		Majority Voting
		1	2	
I	1064	94.19%	97.31%	99.65%
II and III	2973	98.42%	96.77%	98.73%
IV	387	97.25%	96.26%	99.31%
V	3386	96.16%	97.89%	99.02%
VI	967	98.57%	97.42%	99.10%
VII	607	95.24%	98.94%	99.48%
VIII	291	98.67%	97.02%	99.37%
Total	9675	96.61%	97.44%	99.05%

Major sources of errors are (i) poor writing style, (ii) abnormal stroke order variations, (iii) presence of noise due to uncontrolled pen movement, etc. However, use of contextual information removes some of the errors, e.g. in Fig. 4, subscript "*i*" of "*b*" has been recognized as "*c*". However, from the context it is checked that the expression is a summation that uses "*i*" as a script symbol. Therefore, the other alternatives are checked to recognize subscript of "*b*". If "*i*" is not found as any one of the alternatives, the top choice is retained in the output. It has been observed that use of contextual knowledge improves the overall recognition rate to 99.21%.

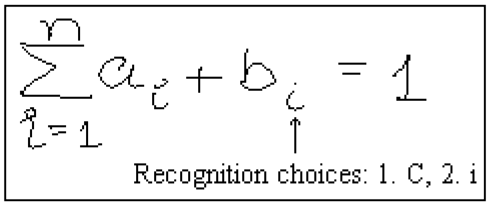

Fig. 4. An example that shows a recognition error, which is corrected by using contextual information.

Performance evaluation of the structural analysis stage is done by computing how many of structures are properly identified among the total number of structures present in a ME. MEs involve two types of structures: (i) 1-D structures like different operators including parentheses, keywords like "*sin*", "*lim*", etc. and (ii) 2-D structures e.g. script, limit, fraction, root, matrix, etc. e.g. System generates Latex string is matched with the groundtruthed using a dynamic programming to find whether each structure is recognized correctly. Details of structure recognition results are presented in Table 2.

Table 2. Structure/operator recognition rate

Structure	Total	Accuracy
Script	6743	98.91%
Limit	1318	97.69%
Fraction	251	98.05%
Root	127	96.85%
Matrix	20	95.00%
Accent (e.g. *bar*)	46	93.48%
Parenthesis	982	98.69%
Other 1-D, e.g. +, -, etc.	5171	98.21%
Total	14658	98.47%

Main reason of the errors in structural analysis is casual handwriting style that creates confusions to determine relationships among a pair of symbols. Some restrictions on writing style will definitely help to resolve several ambiguities that arise during structural analysis stage.

5.1. Integrated Performance Measure

Overall evaluation of ME recognition results is a difficult task as it consists of two tightly coupled distinct stages: symbol recognition and structural analysis. Chan and Yeung [10] proposed an integrated performance measure consisting of two independent measures: one for recognition of symbols and another for recognition of operators. These two measures are combined where both symbols and operators get equal weights. Later on, Okamoto et. al. [11] have presented an automatic method for evaluating of their structure analysis method. In their system, MEs against which their method is evaluated are groundtruthed into MathML format.

In this paper, we propose a different way of measuring performance. Errors for recognition of symbols and structures are computed separately and combined in a single measure. While combining these two measures, it must be noted that errors in recognizing a structure may play more crucial role than the errors in recognizing an individual symbol. Errors in structure recognition are not fully penalized if the original meaning could be retrieved by looking at the final arrangement (though erroneous) of symbols.

In our experiments, the generated Latex strings for the input MEs are compiled and converted into printed expressions, which are then read by 10 different readers belong to scientific community. Selection of such readers is very crucial as that influences the determination of readability of a ME. Out of 10 readers, 3 are high-school level students, 4 are college-level students of mathematics, and rest 3 are selected from the people engaged in scientific research. The integrated measure is formulated as follows.

Let P be the total number of symbols in ME, Q be the total number of structures/operators in ME, p be the number of symbols properly recognized by the system and
r_i be the readability of i-th structure, where $0 \leq r_i \leq 1$. The value of r_i is reader dependent. A reader gives $r_i = 1$ to if the structure is properly recognized and gives 0 if recognition of that structure is affected by errors and the reader does not make out the original meaning by looking at the erroneous arrangement. On the other hand, a reader assigns a value (> 0 and < 1, in our case it is 0.5) to r_i if s/he partially or fully understand the meaning of the i-th structure. Then overall efficiency, e of the system is defined as,

$$e = \frac{\sum_{i=1}^{Q} r_i + p}{Q + P} \tag{8}$$

By following this method, our proposed system shows an efficiency of 98.97%.

5.2. Processing Speed

All algorithms are written in 'C' on a 733 MHz IBM PC. MEs are grouped into 3 classes based on their size in number of symbols: (i) small, if number of symbols is less than 10, (ii) medium, if number of symbols is in between 10 and 20, and (iii) large, if symbols are more than 20 in number. Time required to convert each type of MEs into corresponding Latex string is shown in Table 3.

Table 3. Processing speed

Types of MEs	Processing time (in sec.)
Small size	0.95 sec.
Medium size	1.70 sec.
Large size	3.55 sec.

6. Conclusions

In this paper, a system for automatic understanding of online handwritten MEs is presented. This is motivated to provide a better man-machine interface to input MEs into digital documents. Because of its very high efficiency, the recognizer finds its application in other online character recognition task like one for recognizing online handwritten Indian language scripts [12, 13]. A new performance measure has been proposed in this paper.

Future plan includes design of more robust error detection and correction scheme to improve structural analysis of input expressions. Users may make mistake while writing MEs and hence, an ability to detect such errors would be an added advantage. Moreover, it would be advantageous for visually impaired people if the input expressions were converted into their corresponding speech form [14].

References

1. D. Blostein and A. Grbavec, "Recognition of Mathematical Notation", *Handbook of Character Recognition and Document Image Analysis*, Eds. H. Bunke and P.S.P. Wang, World Scientific Publishing Company, pp. 557-582, 1997.
2. K-F. Chan and D-Y. Yeung, "Mathematical Expression Recognition: A Survey", *Int. J. on Document Analysis and Recognition* (IJDAR), Vol. 3, No. 1, pp. 3-15, 2000.
3. H. Freeman, "On the digital computer classification of geometric line patterns," *Proc. national Electronics Conf.*, vol. 18, pp. 312-324, 1962.
4. U. Garain and B.B. Chaudhuri, "Compound character recognition by run number based metric distance", in *Proc. IS&T/SPIE's 10th Int. Symposium on Electronic Imaging: Science & Technology*, SPIE Vol. 3305, San Jose, California, USA, pp. 90-97, 1998.
5. L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models", *IEEE trans. on Accoust. Speech, Signal Processing* (ASSP), pp. 4-16, June 1986.
6. B. H. Juang and L. R. Rabiner, " The segmental k-means algorithm for estimating the parameters of hidden Markov models", *IEEE trans. on Accoust. Speech, Signal Processing*, Vol. ASSP-38, No. 9, pp. 1639-1641, September, 1990.
7. G. D. Forney Jr., "The Viterbi Algorithm", Proc. IEEE, Vol. 61, No. 3, pp. 263-278, March, 1973.
8. Aho, Sethi and Ullman, "Compilers: Principles, Techniques, and Tools", published by Addison-Wesley Publishing Co., 1986.
9. T. V. Raman, "Audio System for Technical Readings", Ph.D. Dissertation, Cornell University, USA, 1994
10. K-F. Chan and D-Y. Yeung, "Error detection, error correction and performance evaluation in on-line mathematical expression recognition", Pattern Recognition, Vol. 34, pp. 1671-1684, 2001.
11. M. Okamoto, H. Imai, and K. Takagi, "Performance Evaluation of a Robust Method for Mathematical Expression Recognition", *Proc. of Int. Conf. on Document Analysis and Recognition* (ICDAR), Seattle, USA, pp. 121-128, 2001.
12. U. Garain, B. B. Chaudhuri, and T. Pal, "Online Handwritten Indian Script Recognition: A Human Motor Function based Framework", *Proc. of the 16th Int. Conf. on Pattern Recognition* (ICPR), Quebec City, Canada, 2002.
13. U. Garain and B. B. Chaudhuri, "Input of Handwritten Mathematical Expressions into machine coded Indian Language Documents", *in Proc. of The Indo European Conference on Multilingual Technologies* (IECMT), Eds: R. K. Arora, M. Kulkarni, and H. Darbari, Tata McGraw-Hill Publishing Company Limited (New Delhi), pp. 3-12, Pune, India, 2002.
14. B. Hayes, "Speaking of Mathematics", in: *American Scientists*, Vol. 84, No. 2, pp. 110-113, 1996.