A Purely Online Approach to Mathematical Expression Recognition

Ray Genoe, John A. Fitzgerald, Tahar Kechadi

A Purely Online Approach to Mathematical Expression Recognition

Ray Genoe, John A. Fitzgerald, and Tahar Kechadi
School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
ray.genoe@ucd.ie, john.fitzgerald@ucd.ie, tahar.kechadi@ucd.ie

Abstract

In this paper a new approach to handwritten mathematical expression recognition is presented. The approach is highly original in that the solution, an expression tree, is immediately updated whenever the user writes a new stroke. Fuzzy logic is used extensively, in both the symbol recognition and structural analysis phases, which is appropriate given the amount of imprecision and ambiguity present in handwritten mathematics. The approach is highly efficient and encouraging results have been achieved.

Keywords: Mathematical Expression Recognition; Fuzzy Logic; Online Handwriting Recognition

1. Introduction

Including mathematical expressions in scientific documents is far more difficult than it ought to be. Authors using LaTeX are obliged to become familiar with an elaborate set of commands [2], and considerable time and mental effort is often required to produce larger expressions. It would be preferable if authors could take advantage of pen-based interfaces to enter their expressions in a more intuitive manner, using software which could recognise handwritten expressions. Thus the task of converting an expression into a format understood by machines would no longer be the author's responsibility.

In this paper we present a new approach to recognising handwritten mathematical expressions (MEs). Recognising handwritten MEs involves two major subtasks: *symbol recognition* and *structural analysis*. This paper focuses on the structural analysis problem. Structural analysis involves examining the spatial relationships between symbols to determine the most likely expression tree, from which a LaTeX string or an image can be generated.

1.1. The Challenges Involved

Structural analysis of handwritten MEs is a challenging problem for many reasons. Spatial relationships are often highly ambiguous, even to humans, so designing rules to assess these relationships is difficult. These rules must take into account whether the symbols involved are ascenders, descenders, or otherwise. Syntax checking must be performed, although this helps to eliminate invalid solutions. Contextual information must be gathered during structural analysis to identify certain symbols, to differentiate between a minus and a division symbol for example. A single misinterpretation during structural analysis may lead to a very inaccurate expression tree, whereas a symbol recognition error is less of a problem. Furthermore, efficiency is difficult to achieve.

1.2. Related Work

ME recognition has received limited attention in relation to other handwriting recognition problems. Literature reviews detailing earlier attempts are available in [3] and [4]. The authors in [5] used a soft-decision approach, generating alternative solutions if ambiguities were detected. Chan and Yeung [6] proposed a method called hierarchical decompositional parsing, which was based on definite clause grammar (DCG). DCG was used to define a set of replacement rules for parsing MEs, but the approach was inefficient due to its frequent use of backtracking. They improved its efficiency by using left-factored rules and binding symbol preprocessing.

More recently, Garain et al. [7] approached the task by segmenting the expression into atomic boxes and then repeatedly merging adjacent boxes according to a set of production rules corresponding to spatial relationships. In [8] a tree transformation based method is proposed, whereby a recursive search identifies linear structures in an expression and constructs a Baseline Structure Tree, which is then subjected to lexical analysis to produce the final tree.

The authors in [9] propose an approach called Fuzzy Shift-Reduce Parsing (FSRP). This approach is built upon traditional shift-reduce parsing methods, thus providing syntax checking and also a basis for efficiency. Fuzzy logic is introduced to cope with the imprecision of handwritten input. Multiple parses are pursued if ambiguities arise and the most likely expression tree is selected as the result. Other recent attempts at the problem are documented in [10, 11, 12, 13].

1.3. Our Approach

We propose a purely online approach to structural analysis, whereby the solution is immediately updated whenever the user writes a new stroke. This differs

from previous approaches, which invariably required the entire expression to have been written and the user to have clicked a button before structural analysis began (although online information was often gathered for use during structural analysis later, as in [7]).

We will use fuzzy rules [14] to assess the spatial relationships between symbols. Fuzzy rules allow decision making with estimated values under incomplete or uncertain information. They provide a way of enabling approximate human reasoning capabilities to be applied to knowledge-based systems. Their use in handwriting recognition problems is certainly appropriate as handwritten input will never be standard for multiple users, or even for the same user.

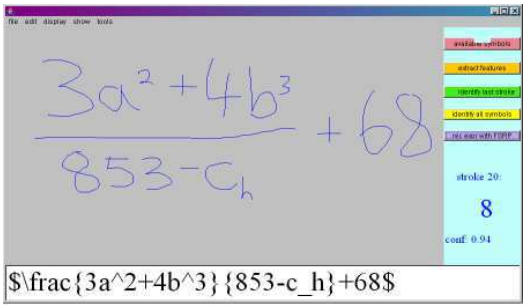

Figure 1. Mathpad software in action.

The outline of the paper is as follows. Section 2 contains a brief discussion of the symbol recogniser. Section 3 outlines our structural analysis algorithm. Section 4 discusses the fuzzy spatial relationship rules which are used in the algorithm. Section 5 shows experimental results, and section 6 contains conclusions and future work.

2. The Symbol Recogniser

We have developed a symbol recogniser [1] in which feature extraction and classification are achieved using fuzzy rules. In the feature extraction phase, the objective is to represent the symbol as a combination of basic features, where each feature is of type *Line*, *C-shape* or *O-shape*. We believe that by representing each symbol in this manner, we are capturing the essence of the symbol and what distinguishes it from other symbols, thus making classification easier. Fuzzy rules are used to determine membership values for *substrokes* in fuzzy sets corresponding to the three feature types. Based on these membership values the best set of substrokes is chosen as the feature extraction result.

In the classification phase, fuzzy classification rules are used to determine the most likely identity of the symbol represented by the feature extraction result. These classification rules state the *attributes* (such as orientation) and *relationships* (such as connectivity or relative lengths) that a set of features should have, if they are to form a particular symbol.

This fuzzy logic recogniser will be combined with other recognisers to improve recognition rates. Other recognisers we are currently developing use Self Organising Maps, Recurrent Neural Networks [15], and Hidden Markov Models with Support Vector Machines [16].

3. Fuzzy Online Structural Analysis Algorithm (FOSA)

This section discusses the *Fuzzy Online Structural Analysis Algorithm* (FOSA). $FOSA$ is characterised by the fact that, unlike other structural analysis algorithms, it does not accept as input a set of n symbols and attempt to determine the expression tree they form. Instead, it accepts as input a single stroke s and a partial expression tree T, and it updates the existing expression tree T to include the new stroke s. This is achieved by using fuzzy rules to determine the strongest relationship that exists between s and a sub-expression of T.

Tackling the problem in this manner is beneficial for numerous reasons. The most important reason is efficiency. With other approaches, the user must finish writing the expression and must then click a button before structural analysis can even begin. Using $FOSA$, the solution should already be available when the user has lifted the pen after writing the final stroke.

Secondly, $FOSA$ does not have to deal with an initial input of n symbols. Instead the problem is hugely simplified to fitting a single stroke s into an expression tree. This process has to be performed each time a stroke is written, but it is sensible to perform most of the structural analysis while the user is writing.

Another advantage is that any misrecognition by the system is visible to the user immediately, making error correction easier. Finally, the uniqueness of the approach means it could be combined effectively with other approaches [9], on the basis that if one approach struggles with a certain input, a completely different approach may be successful.

Algorithm 1 - FOSA Algorithm

For every new stroke s_{new} added to the expression, update the expression tree T as follows:

Classify s_{new} using the symbol recogniser.
if s_{new} is the first stroke added **then**
  Create a node for s_{new} and set it as the root of T.
else
  $max = 0$
  for $i = 0, \ldots, n-1$ **do**
    for $j = 0, \ldots, m-1$ **do**
      if $R_j(s_{new}, s_i) \geq max$ **then**
        $max = R_j(s_{new}, s_i)$
        $R_{max} = R_j$
      end if
    end for
  end for
  Update T according to the relationship R_{max}.
end if

The algorithm is shown in Algorithm 1 and is discussed in the following subsections. It has been slightly

simplified in that if the new stroke s_{new} intersects existing strokes, it must be put through a subroutine to determine if it is a component of a multi-stroke symbol. The algorithm uses a set of m fuzzy rules $\{R_0(s_i, s_j), \ldots, R_{m-1}(s_i, s_j)\}$. Each fuzzy rule determines the confidence that a particular spatial relationship exists between two sub-expressions s_i and s_j. The tree T contains n nodes $\{s_0, \ldots, s_{n-1}\}$, where each subtree corresponds to a sub-expression.

3.1. Tree Structure

When the first stroke is written, it is set to be the root of the tree T. T is a binary tree, so each node may have a parent, a left child, and a right child. Operators occupy the internal nodes and identifiers occupy the leaves. The operators can be explicit or implicit. Explicit operators are visible in the written expression, such as +, -, or *. Implicit operators are spatial relationships such as superscript, subscript, or concatenation. The identifiers can be digits, letters, numeric strings and so on.

The nodes of the tree are stored in reverse temporal order. This is because when s_{new} is written, it is more likely to have a relationship with the most recently written symbol, so when we search for the strongest relationship this node will be examined first.

3.2. Determining the Strongest Relationship

Whenever a new stroke s_{new} is written, we assess the strength of every spatial relationship $R_0, \ldots R_{m-1}$ between s_{new} and every sub-expression in T. The relationship with the highest confidence is designated as R_{max}. Most relationships have to be assessed in two different ways, for example we must determine confidence values for $superscript(s_i, s_{new})$ and $superscript(s_{new}, s_i)$, as the user may have entered symbols in an unexpected order.

This process may seem expensive. However, if a relationship is found with confidence above a certain threshold t_0, such that it seems certain this will be the strongest relationship, no more relationships are assessed. Also, the fact that the nodes are stored in reverse temporal order means a relationship with confidence above t_0 is often found almost immediately.

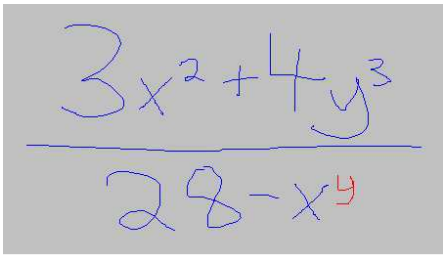

Figure 2. In this expression, the most likely relationship involving the new stroke was found to be $superscript(x, y)$ with confidence 0.74.

When searching for the strongest relationship, s_{new} is assessed not only with every symbol in T, but with every sub-expression in T. This is necessary because s_{new}'s strongest relationship may not be with a single symbol. For example, s_{new} may be a fraction line drawn beneath a sub-expression to form $\underline{a^2 + b^2}$. The fuzzy spatial relationship rules accept sub-expressions as arguments, because single symbols or strokes can also be regarded as sub-expressions. How these rules work will be discussed in Section 4.

3.3. Updating the Expression Tree

Once we have determined the strongest relationship R_{max} between s_{new} and a sub-expression of T, we must update T according to R_{max}. Each relationship has its own routine for updating the tree. These routines often involve detaching subtrees from T and reattaching them as children of new nodes. An example for superscript is shown below.

If the strongest relationship is found to be $superscript(s_i, s_{new})$, proceed as follows:

1. create a new node $SUPER$ to represent the superscript relationship;
2. remove the node s_i from T and attach it as the left child of $SUPER$;
3. create a new node for s_{new} and attach it as the right child of $SUPER$, forming a new subtree;
4. attach this subtree in the position previously occupied by s_i.

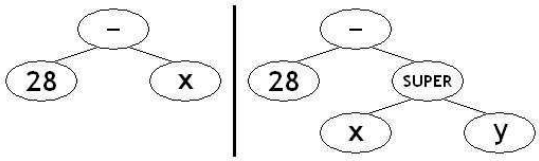

Figure 3. Updating the tree according to superscript.

In certain cases new nodes need not be created to update the tree. For example, if the strongest relationship is $concatenation(s_i, s_{new})$ where s_i and s_{new} are both digits, we simply update the identity (e.g. '2' to '28') and update the bounding box of the node containing s_i.

3.4. Maintaining Multiple Trees

When we search for the strongest relationship involving s_{new}, the result may be that there is no confidence in s_{new} having a relationship with any sub-expression in T. Such a situation suggests that s_{new} has been written some distance away from the other symbols in the expression.

A user may write a symbol far away from the expression for two reasons. Firstly, they may be beginning a separate mathematical expression. Secondly, they may be

writing a single expression in an unconventional manner, i.e. writing the symbols at either end of the expression first before writing those in the centre.

If no relationship is found involving s_{new}, a new tree is created with s_{new} as the root. As new strokes are added, their spatial relationships with every node in every tree are assessed, and the relevant tree is updated according to the strongest relationship. So if the user is indeed writing multiple expressions, multiple solutions are maintained for these expressions.

However, if the user is entering a single expression in the unconventional manner described, the separate trees must be *merged* once they have become sufficiently close. Therefore, if s_{new} has relationships with confidences above a threshold t_{merge} with sub-expressions from different trees, these trees are merged. Expressions entered in this manner are more problematic for $FOSA$ than for offline structural analysis algorithms, but it is rare for expressions to be written in this fashion.

3.5. Complexity Analysis

An important aspect of a structural analysis algorithm is the complexity in terms of the number of symbols n in the expression. It is desirable to have less than $O(n^2)$ complexity. Due to the online nature of our algorithm, we need only examine the complexity of adding the final stroke to the expression. This is the relevant complexity for comparison with algorithms which only begin structural analysis when the expression is finished.

$FOSA$ essentially involves two steps: determining the strongest relationship, and updating the tree. If m is the number of spatial relationships, the complexity for each step of $FOSA$ is as follows:

- Determining the strongest relationship has complexity ***O(nm)***.
- Updating the tree has complexity ***O(1)***.

Therefore the complexity of $FOSA$ is ***O(nm)***. The number of relationships is fixed and is not large, so the complexity can be regarded as ***O(n)***, which compares favourably with other algorithms [8] [9].

4. Fuzzy Rules

This section discusses how we assess the spatial relationships between symbols using fuzzy rules. Other approaches to assess spatial relationships involve defining regions around a symbol. The region in which another symbol's centroid lies dictates the spatial relationship between the two [10]. However, such an approach does not retain information regarding how confident we are that a certain relationship holds. Such information can be hugely useful, especially if we have multiple competing solutions at the end of the recognition process.

Using fuzzy rules to assess spatial relationships is far more appropriate. Spatial relationships are not black and white in nature, and retaining fuzzy information leads to a more informed decision process.

The fuzzy rules determine membership values (confidence values) between 0 and 1 in fuzzy sets corresponding to spatial relationships. Each fuzzy rule assesses the extent to which various aspects of a relationship deviate from those of a prototype relationship. The rules are defined using fuzzy membership functions such as the S-function [14].

The fuzzy rules rely on the bounding box information of the sub-expressions involved (width, height, leftX, topY etc.). Different rules are required for each relationship depending on whether the symbols involved are ascenders, descenders, or otherwise, because different sections of the symbols are relevant (see Fig. 4). Also, the areas of ascenders and descenders should be halved for fair comparison with lowercase letters such as a or c. Furthermore, the rules work differently if any of the symbols involved are lines (i.e. 1's or minuses).

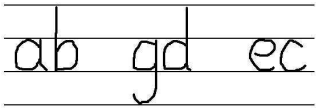

Figure 4. Three pairs of symbols for which concatenation holds, with different vertical alignments.

Most rules determine multiple confidence values regarding different aspects of the relationship, and then combine these values into one value using norms such as the minimum. For example, the fuzzy rule $superscript(s_i, s_j)$ determines confidence values reflecting the extent to which each of the following holds:

- s_j is just to the right of s_i;
- s_j is somewhat higher than s_i;
- s_j's area is significantly less than that of s_i.

The fuzzy rules are continually refined based on testing. Designing fuzzy rules by hand is more feasible when the amount of rules required is relatively small, as is the case with spatial relationships in mathematical expressions. However, it is more difficult to maintain a fuzzy rule base for symbol classification, for example.

5. Experimental Results

We have implemented a system called Mathpad, in which the user can write MEs using the mouse or an electronic pen and data tablet. The expressions are recognised using $FOSA$ and a LaTeX string is generated. Testing was carried out on a database of 60 expressions written by two different users. 30 of the expressions were simple expressions containing less than 8 symbols. The remaining 30 expressions were more elaborate, containing 10 - 25 symbols. Overall, the correct structure was determined for 53 of the 60 expressions, or, **88.33%**. Unfortunately there is no widely available corpus of online handwritten MEs, so comparison with other approaches is difficult.

The majority of the structural analysis errors suggested improvements were required to the fuzzy rules, rather than the algorithm. When assessing the relationship between s_{new} and s_i, the rules do not check if there is another symbol in between s_{new} and s_i, adding this functionality will improve performance. Certain errors could have been avoided if the fuzzy rules used norms other than the minimum. The symbols may have to be divided into more categories than ascenders, descenders and standard characters. Also, certain errors suggested the fuzzy rules should not only examine bounding boxes; factoring in intersections and the position of endpoints would yield improvements. The algorithm also has difficulty recognising expressions in which the user writes symbols at either end of the expression first, before writing symbols in the centre.

6. Conclusions and Future Work

In this paper we have presented a highly original and efficient approach to handwritten ME recognition. Other approaches only begin structural analysis once the expression has been completed. However it is clearly worth investigating a technique which performs structural analysis while the user is writing. It is faster and it leaves us with a simpler problem to solve. Encouraging results have been achieved and there is much potential for improvement through refinement of the fuzzy rules.

The use of fuzzy logic is of paramount importance when dealing with handwritten input. We have used fuzzy rules throughout the ME recognition process, for feature extraction, symbol classification and structural analysis. Retaining fuzzy information leads to a more informed decision process. To not use fuzzy logic, to simply designate a symbol identity or relationship as definitely being of a certain type, is to jettison crucial information which will often lead to misrecognition.

As regards future work, the symbol recognition capability needs to be increased to cover a wider range of symbols, such as integrals and square roots, and more fuzzy spatial analysis rules must be written involving these symbols. Improvements will be made to the existing fuzzy rules. We will also investigate the scalability of our approach.

Currently our algorithm only pursues the most likely relationship when each new stroke is added. However, precise fuzzy information regarding the strength of different relationships is available. Therefore we plan to extend our algorithm so that less likely relationships are explored whenever strong ambiguities arise, on the basis that a less likely option may form part of the most likely overall solution.

We hope to combine our approach with the fuzzy parsing approach in [9]. If differing solutions are produced by both algorithms, a decision can be made according to the confidences of each solution. This will certainly yield improved recognition rates.

References

[1] J.A. Fitzgerald, F. Geiselbrechtinger, and T. Kechadi, "Application of Fuzzy Logic to Online Recognition of Handwritten Symbols", *The Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR 9)*, Tokyo, Japan, October 26-29, 2004, pp 395-400.

[2] D.R. Wilkins, "Summary of Commonly-Used Features of LaTeX", November 19, 1993.

[3] K.-F. Chan and D.-Y. Yeung, "Mathematical Expression Recognition: A Survey", *International Journal on Document Analysis and Recognition*, 2000, Vol. 3, No.1, pp 3-15.

[4] D. Blostein and A. Grbavec, "Recognition of Mathematical Notation", *Handbook of Character Recognition and Document Image Analysis*, 1997, pp 557-582.

[5] H. -J. Winkler, H. Fahrner, and M. Lang, "A Soft-Decision Approach for Structural Analysis of Handwritten Mathematical Expressions", *The International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, Detroit, USA, May 1995, Vol. 4, pp 2459-2462.

[6] K.-F. Chan and D.-Y. Yeung, "An Efficient Syntactic Approach to Structural Analysis of On-line Handwritten Mathematical Expressions", *Pattern Recognition*, 2000, Vol. 33, pp 375-384.

[7] U. Garain and B. Chaudhuri, "Recognition of Online Handwritten Mathematical Expressions", *IEEE Transactions on Systems, Man and Cybernetics*, 2004, Part B, Volume 34, Issue 6, pp 2366-2376.

[8] R. Zanibbi, D. Blostein and J. Cordy, "Recognizing Mathematical Expressions Using Tree Transformation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, Volume 24, Issue 11, pp 1455-1467.

[9] J.A. Fitzgerald, F. Geiselbrechtinger, and T. Kechadi, "Structural Analysis of Handwritten Mathematical Expressions Through Fuzzy Parsing", *The IASTED International Conference on Advances in Computer Science and Technology (ACST 2006)*, Puerto Vallarta, Mexico, January 23-25, 2006, pp 151-156.

[10] E. Tapia and R. Rojas, "Recognition of On-Line Handwritten Mathematical Formulas in the E-Chalk System", *The Seventh International Conference on Document Analysis and Recognition (ICDAR '03)*, Edinburgh, Scotland, August 3-6, 2003, pp 980-984.

[11] R. Fateman, "2-D Display of Incomplete Mathematical Expressions", University of California at Berkeley, October 2003.

[12] S. Smithies, K. Novins, and J. Arvo, "Equation Entry and Editing via Handwriting and Gesture Recognition", *Behaviour and Information Technology 20(1):53-67*, January 2001.

[13] Nicholas E. Matsakis, "Recognition of Handwritten Mathematical Expressions", Massachusetts Institute of Technology, May 1999.

[14] L.A. Zadeh, "Fuzzy Sets", *Information and Control, Vol. 8, pp 338-353*, 1965.

[15] J.A. Fitzgerald, B.Q. Huang, and T. Kechadi, "An Efficient Hybrid Approach to Online Recognition of Handwritten Symbols", *The Fourth Mexican International Conference on Artificial Intelligence (MICAI 2005)*, Monterrey, Mexico, November 14-18, 2005, pp 843-853.

[16] B.Q. Huang, C.J. Du, Y.B. Zhang, and M-T. Kechadi, "A Hybrid HMM-SVM Method for Online Handwriting Symbol Recognition", *The Eighteenth International Conference on Pattern Recognition (ICPR 06)*, Hong Kong, August 20-24, 2006.