Understanding Mathematical Expressions from Document Images

Jaekyu Ha & Robert M. Haralick

Dept. of Electrical Engineering, FT-10
Unversity of Washington
Seattle, WA 98195, U.S.A.

Ihsin T. Phillips

Dept. of Computer Science
Seattle University
Seattle, WA 98122, U.S.A.

Abstract

This paper proposes a system that understands mathematical expressions on binarized printed document images. The system first extracts a set of 'primitives' from the image. Each of the extracted primitives is associated with a 'bounding box' and its label. Using the attributes of the primitives, the system constructs an initial hierarchy. Construction of an initial hierarchy includes merging a group of primitives into a key word. Next, the system checks the validity of the hierarchy according to conventional mathematical syntax rules. If any syntax error is detected, the system makes attempts to correct the errors. The modification step includes reconfiguring the initial hierarchy, revisiting the original image for possible missing primitives, placing dummy primitives into missing spots in the hierarchy, and so on. The corrected hierarchical structure can be converted into the format for a particular publication system such as TEX.

1 Introduction

We propose a system that understands mathematical expressions on binarized printed document images. The input to the system is a mathematical expression and the output of the system is a hierarchical representation of the expression (see Section 3).

The system works as follows. Given a mathematical expression in a binary image format, the system first extracts a set of "primitives" from the image. Each of the extracted primitives is associated with a 'bounding box' and its label. Using the attributes of the primitives, the system constructs an initial hierarchy for the expression. The initial hierarchy construction includes merging a group of primitives into a key word. Next, the system checks the validity of the hierarchy according to conventional mathematical syntax rules. If any syntax error is detected, the system makes attempts to correct the errors. The modification step includes reconfiguring the initial hierarchy, revisiting the original image for possible missing primitives, placing dummy primitives into missing spots in the hierarchy, etc.

The system has the capability of checking the 'similarity' of a pair of given hierarchical structures. This capability can be an aid to the information retrieval system for locating related technical documents in databases.

The remaining of the paper is written as follows. In the next section, we give our motivation for the research and development of a math expression understanding system (MEUS) and a brief discussion of some related works on the topic. A complete description of our system is given in section 3. Our summery is given in section 4.

2 Motivation

Symbol recognition problems in document images arise in many areas such as mathematical expressions, line drawings, music scores, chemical graphs, and so on. In general, recognition problems in these areas are not easily tractable due to the following reasons: Firstly, many of the symbols in nontext zones are represented not by alphanumeric symbols but by non-alphanumeric symbols. Secondly, symbols in non-text zones in documents have different spatial configurations from those of ordinary texts. Due to these difficulties, the development of recognition systems for the above areas has been put aside for the time being. Unfortunately, modern optical character recognition (OCR) systems can recognize only ordinary texts with high accuracy. Though a few researches have been reported in each of the above areas, unified approach has not been attempted so far.

A system which can understand a mathematical expression on a printed document is obviously needed for technical document understanding and information retrival systems. Today, many sophisticated algorithms have been developed for OCR. The research on the document page layout and structure analysis also has

0-8186-7128-9/95 $4.00 © 1995 IEEE

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE

made a significant progress.

The main purpose of an OCR system is to capture information on documents so that the information can be stored and retrieved easily. Since there does not exist an OCR engine that has in it the mathematical recognition capability, most OCR systems capture only the text information and leave all mathematical expressions unrecognized. Recently, the document understanding and information retrieval research community has recoginized that there are needs for a mathematical expression understanding system. Since scientific and technical documents, in general, contain mathematical expressions which may give precise information about the documents, textual information without the presence of the mathematical expressions may not be meaningful for technical document understanding and information retrieval. Thus, a completed computerized archival of technical documents is not possible without a system which understands mathematical expressions. But the research on this topic is still in its infancy. The earliest report on understanding mathematical expressions was pioneered by Anderson (1968). The other reports can be found in Chang (1970), Martin (1971), Okamoto & Miyazawa (1992), and Lee & Lee (1993).

3 Math Expression Understanding System (MEUS)

We give a brief description of the proposed MEUS system. First, it extracts all primitives from a given mathematical expression. Primitive extraction is performed by applying a connected component labeling algorithm to a given image which contains only a mathematical expression. Since some symbols may consist of two or more connected components, merging of some connected components is required to obtain meaningful primitives. Second, it builds up a hierarchical structure for that mathematical expression. The structure can be conveniently represented using trees. We will explain, later in detail, how such a tree can be constructed with connected components. A tree which represents the hierarchical structure of a mathematical expression will be called an *expression tree* (ET). (This terminology might confuse some readers since the usage of this terminology with a different meaning can be found in many data structures text books). Third, it checks the mathematical syntax and correct possible errors in the data structure. Once we have built up the expression tree, it must be checked whether the mathematical syntax is correct among neighboring objects or not. If syntax errors are found, the expression tree has to be modified by spliting/merging nodes. Finally, we can convert the corrected data structure into the format for a particular publication system such as TEX.

3.1 Object-Oriented Representations

Mathematical expressions are represented with various kinds of entities (*primitive symbols*). Such primitive symbols include all possible alphabetic characters, numerals, math operators and so on. In addtion, each primitive element in mathematical expressions can be attributed with its name and bounding box information. The bounding box of each primitive symbol is meant to be the smallest bounding box. The MEUS will use this information to abstract the data structure of symbols in a mathematical zone. Since all primitives are equally important, they are the *objects* with which we analyze and understand a given mathematical expression. Though extraction of such primitive objects is the first step to understanding mathematical expressions, proper combination of some of those primitives must be syntactically correct in a mathematical sense.

The object-oriented representations of mathematical expressions consist of three parts: a set of objects, a set of operations and abstract data structures. Objects can be categorized into two classes: primitive and compound objects. A primitive object is one which cannot be resolved any more. In mathematical expressions, possible symbols are alphabetic characters (*English, Greek, Hebrew,* $\cdots$), numerals (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), math operators (computational and relational), and other symbols ((,), $\rightarrow$, ∞, $\cdots$). A compound object is a *proper* combination of primitive objects. For example, in the expression $\frac{x+y}{x^2+y^2}$, primitive objects are x, $+$, y, —, x, 2, $+$, y and 2. Any combination of these primitive objects cannot be a compound object. In the object-oriented representation, an object is an entity whose behavior is characterized by the actions that it suffers and that it requires of other objects. Therefore, the first two of the above primitive objects cannot be combined to form a compound object because such combination cannot characterize any action with other objects.

3.2 Construction of an Expression Tree

Trees are used in representing data structures that have a hierachical, or nested, or one-to-many relationship among their component elements. A tree can contain any elements we choose. It is assumed that each element contains information which describes some objects, and that information is divided into two parts: a key part and a data part. The key part has the

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE

unique identification property: that is, there cannot be two different elements with the same key value.

When we discuss trees, we will rarely refer to their elements but will instead refer to their nodes. Each node may contain one or more elements. Reference to a node is therefore an implicit reference to the elements that it contains. In addition to containing data elements, a node may contain information about its relationship woth other nodes. Exactly what additional information it contains depends on how the tree is implemented. It is customary to illustrate hierachical relationships among nodes with arrows connecting pairs of related nodes. In the tree that we will discuss, the direction of an arrow always indicates the decomposition of a compound object.

As long as mathematical expressions are concerned, an abstract data structure can be implemented using an expression tree. Properties of an expression tree are summarized as

- Each node represents an (simple or compound) object.
- The root node represents the entire mathematical expression.
- The internal nodes represent compound objects, each of which consists of two or more objects and satisfy the syntax rule.
- The leave nodes represent simple (primitive) objects.

An expression tree is a very useful abstraction of a mathematical expression. Given a whole math zone, the corresponding expression tree represents abstract data structure of the math zone. Combination of two or more objects should be interrelated.

Construction of an expression tree will be performed through two steps: top-down process (for an initial expression tree) followed by bottom-up process (for a final expression tree).

Top-down Process

Suppose that we are given all primitive objects and their associated bounding boxes within a math zone. The first step to the construction of an expression tree is to do the structured zone division of the whole math zone by the recursive X–Y cut process described in [Ha *et. al.*, 1994]. During the X–Y cut process, we divide the math zone by left-to-right vertical division and then, we divide each subzone by top-to-bottom horizontal division, and we repeat such division until we reach primitive objects. Figure 1 illustrates how

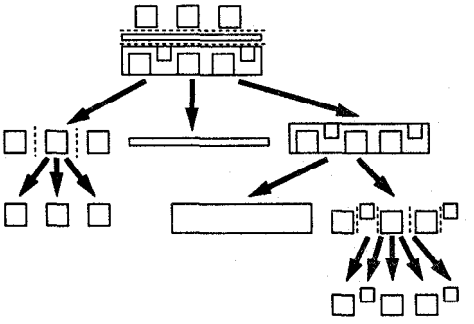

Figure 1: Top-down process for the initial expression tree: the example expression is $\frac{x+y}{\sqrt{x^2+y^2}}$.

to do the structured zone division to obtain an initial expression tree for an example mathematical expression.

We propose an algorithm for construction of an initial expression tree as follows: Given the bounding boxes of all primitive symbols within the entire math zone,

1. resolve overlapping objects
2. divide the math (or sub-) zone by left-to-right vertical division
3. divide the math (or sub-) zone by top-to-bottom horizontal division
4. do steps 1–3 recursively until no further division is necessary

Notice that only primitives cannot be divided further.

Bottom-up Process

As you notice in Figure 1, construction of a proper expression tree is not always guarranteed by such successive zone divisions. It is because spatial relations between neighboring objects are not considered in the division. Therefore, it is natural to do the reverse process while considering spatial relations between neighboring objects. Possible spatial relations with neighboring objects are *above, below, left, right, above right, above left, below right, below left*: the last four are declarative properties, for example, subscript and superscript. Taking such spatial relations into consideration, a *coarse* expression tree which has resulted from the top-down zone division process has to be reformed by spliting and/or merging internal and external nodes (Figure 2). And such reformation is possible by checking the mathematical syntax with neighboring objects from leaves to the root of the initial expression tree.

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE

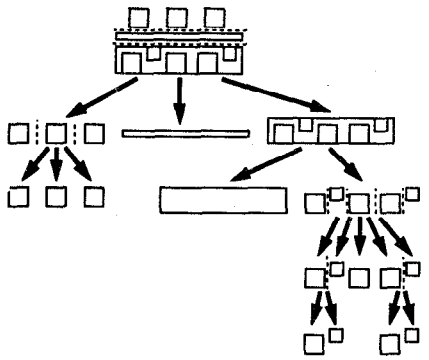

Figure 2: Bottom-up process for the final expression tree: the example expression is $\frac{x+y}{\sqrt{x^2+y^2}}$.

3.3 Symbol Recognizer

Now, we explain construction of our symbol recognizer, without which the MEUS cannot be complete.

Character recognition techniques associate a symbolic identity with the image of a character. Two main approaches to character recognition are the statistical/decision-theoretic and structural/syntactic approaches. Each of them have their merits and demerits. In the structural/syntactic approach, a pattern is often represented as a string, a tree or a graph of pattern primitives and their relations. The decision-making process is, in general, a syntax analysis or parsing procedure. However, the use of formal language-theoretic models to represent patterns is the main drawback of the syntatic approach. Patterns are natural entities which cannot strictly obey the mathematical constraints set by the formal language theory. In the statistical/decision-theoretic approach, a pattern is represented by a feature vector and the decision-making process is essentially classification in the partitioned feature space. Though statistical pattern recognition techniques cannot handle the structural information about the interconnections in complex patterns very well, it is considered particularly suitable for recognizing individual characters since all statistical variations in features (patterns) are considered by training feature vectors.

So, it would be a good choice to implement our experimental symbol recognizer based on the statistical pattern recognition paradigm. For this purpose, large samples will be collected from UW English Document Image Database I, which was prepared by the University of Washington for researchers who are developing algorithms for use in document analysis and optical character recognition. Feature vectors from such large samples will be trained through neural networks to partition the feature space. Feature extraction will be performed by the morphological closing transformation described later in this document. Finally, a binary decision tree classifier will be constructed as the prototype of our experimental symbol recognizer.

4 Discussions

This paper outlines our proposed mathematical expression understanding system. In the design of the system, we adopt the object-oriented approach to describe the data abstraction. A hiearchical structure of mathematical expression is given as an expression tree. There are many sub-problems needed to be solved in the implementation of the system: how to preprocess the document image, how to find math zones in the image, how to evaluate the understanding system, and so on. These sub problems will have to be solved to automate the mathematical expression understanding system.

References

[1] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Volume I, Addison-Wesley, 1992.

[2] Henry S. Baird, *Structured Document Image Analysis*, Springer Verlag, 1992.

[3] T. Akiyama and I. Masuda, "A Method of Document Image Segmentation Based on Projection Profiles, Stroke Densities, and Circumscribed Rectangles," Trans. IECE Japan, vol. J69-D, no. 8, 1986, pp. 1187-1196.

[4] R. H. Anderson, *Syntax-directed recognition of handprinted 2-D mathematics*, Ph.D. Dissertation, Harvard University, Cambridge, MA, 1968.

[5] S. K. Chang, "A method for the structural analysis of 2-D mathematical expressions," *Inf. Sci.*, 2(3), pp. 253–272, 1970.

[6] W. A. Martin, "Computer input/output of mathematical expressions," *Proc. 2nd Symp. Symb. Algebraic Manipulation*, Los Angeles, CA, 1971.

[7] Hsi-Jian Lee and Min-Chou Lee, "Understanding Mathematical Expressions in a Printed Document," *Proc. the second ICDAR*, Tsukuba, Japan, pp. 502–505, 1993.

[8] J. Ha, I.T. Phillips and R.M. Haralick, "Recursive X–Y Cut using Bounding Boxes of Connected Components," ISL Report, Dept. Electrical Eng., University of Washington, 1994

Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)
0-8186-7128-9/95 $10.00 © 1995 IEEE