International Journal on Document Analysis and Recognition manuscript No.
(will be inserted by the editor)

Recognition and Retrieval of Mathematical Expressions

Richard Zanibbi · Dorothea Blostein

Received: data / Accepted: date

Abstract Document recognition and retrieval technologies complement one another, providing improved access to increasingly large document collections. While recognition and retrieval of textual information is fairly mature, with wide-spread availability of Optical Character Recognition (OCR) and text-based search engines, recognition and retrieval of graphics such as images, figures, tables, diagrams, and mathematical expressions are in comparatively early stages of research. This paper surveys the state of the art in recognition and retrieval of mathematical expressions, organized around four key problems in math retrieval (query construction, normalization, indexing, and relevance feedback), and four key problems in math recognition (detecting expressions, detecting and classifying symbols, analyzing symbol layout, and constructing a representation of meaning). Of special interest is the machine learning problem of jointly optimizing the component algorithms in a math recognition system, and developing effective indexing, retrieval and relevance feedback algorithms for math retrieval. Another important open problem is developing user interfaces that seamlessly integrate recognition and retrieval. Activity in these important research areas is increasing, in part because math notation provides an excellent domain for studying problems common to many document and graphics recognition and retrieval applications, and also because mature applications will likely provide substantial benefits for education, research, and mathematical literacy.

R. Zanibbi
Department of Computer Science, Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY, USA 14623-5608. E-mail: rlaz@cs.rit.edu

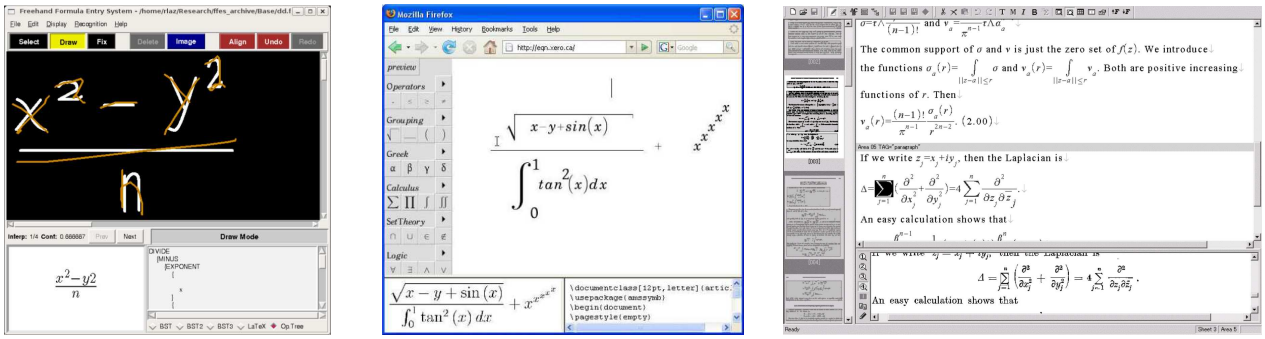
D. Blostein
School of Computing, Queen's University, Kingston, Ontario, Canada, K7L 3N6. E-mail: blostein@cs.queensu.ca

Keywords Math Recognition, Graphics Recognition, Mathematical Information Retrieval (MIR), Content-Based Image Retrieval (CBIR), Human-Computer Interaction (HCI)

1 Introduction

In practice, the problem of retrieving math notation is closely tied to the problem of recognizing math notation. For example, a college student may want to search textbooks and course notes to find math notation that has similar structure or semantics to a given expression. Or, a researcher may wish to find technical papers that use or define a given function. In both of these examples, recognition of math notation is needed in order to support the retrieval of math notation: the system must be able to recognize math expressions that the user provides as a query, and the system must be able to recognize math expressions in the target documents that are the subject of search. Retrieval of math notation has received increasing research attention in the past decade (see Section 3), while math recognition has been a subject of research for over forty years (see Section 4). To our knowledge, we provide the first survey of mathematical information retrieval; in surveying math recognition, we focus on research that has appeared in the decade since the survey of Chan and Yeung [28].

The math domain provides an excellent vehicle for studying pattern recognition and retrieval problems, and for studying methods of integrating pattern recognition algorithms to improve performance. The four central pattern recognition problems – segmentation, classification, parsing, and machine learning (i.e. optimizing recognition model parameters) – all come into play when recognizing mathematics. The math domain

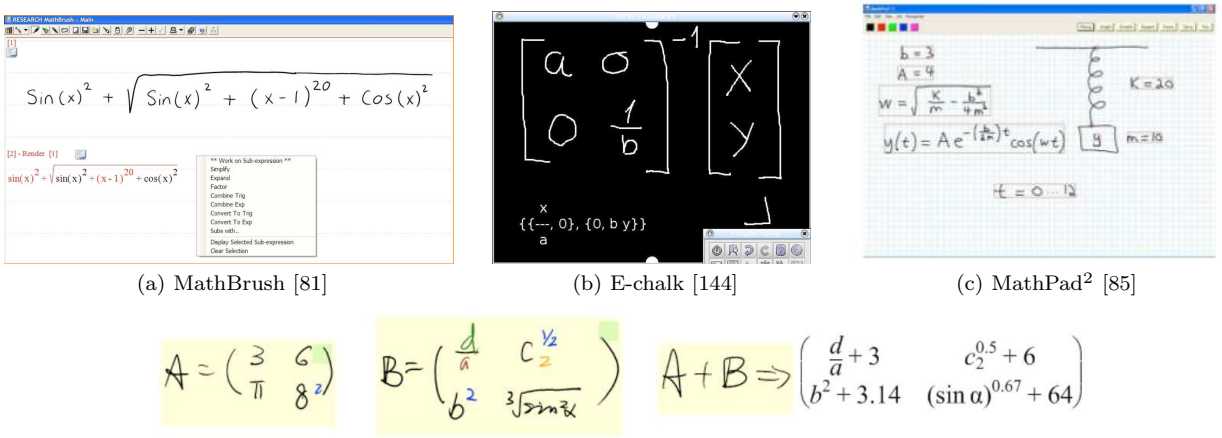


(a) Freehand Formula Entry System (FFES) [20, 136]

(b) XPRESS [116]

(c) InftyEditor/InftyReader [141]

Fig. 1 *Math Entry Systems. FFES is pen-based, XPRESS supports mouse and keyboard entry, and InftyEditor/IntryReader supports OCR, pen, mouse and keyboard entry.*



(a) MathBrush [81]

(b) E-chalk [144]

(c) MathPad^2 [85]

(d) Li, Zeleznik et al. [89]

Fig. 2 *Systems for Pen-Based Computer Algebra and Sketching.*

offers sufficient complexity to challenge researchers, yet has characteristics that make the domain tractable: the semantics of math notation are fairly constrained, and a typical math expression consists of relatively few symbols.

The input to a math recognition system can take three forms: *vector graphics* (such as PDF), *strokes* (such as pen strokes on a data tablet), or a *document image*. The processing that is needed to extract expressions and recognize characters depends greatly on the form of input. For example, a PDF document directly provides encoded symbols, so there is little need for character recognition [13, 14]. Figures 1 and 2 illustrate systems that accept various forms of input: vector graphics is shown in Figure 1b; strokes are shown in Figures 1a and 2a,b,c,d; and a document image is shown in Figure 1c.

In the next sections, we discuss key recognition and retrieval problems as they apply to all three forms of input. As the need arises, we point out situations in which differences in input format cause large differences in processing methods.

1.1 Overview of Math Notation Recognition

Math recognition is used for various purposes. For example, a user may write an expression by hand and insert the recognition result (e.g. a LaTeX string or image) into a document. Alternatively, a recognized expression can be evaluated using a computer algebra system such as Maple or Mathematica. Another option is to use the recognized expression as a query, to retrieve documents containing similar math notation. Recent work in human-computer interaction further motivates the development and use of pen-based math entry systems. Bunt et al. study mathematicians in a research setting, and find that in order to be useful, CAS systems need to support annotation, provide multiple levels of formality, and provide more transparency for the operations that they apply [23]; they suggest that pen-based systems for math might be used to address these needs.

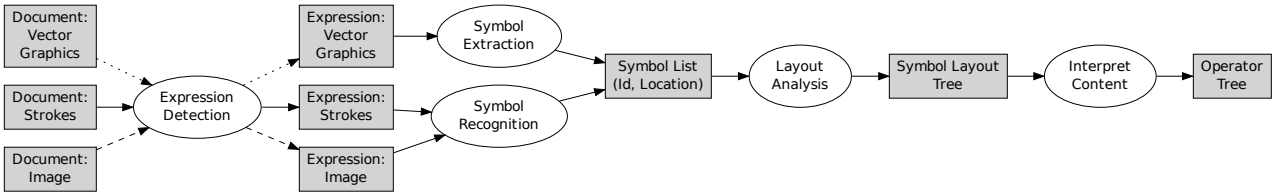


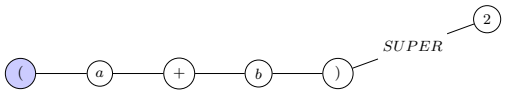
Fig. 3 *Key Recognition Problems: Expression Detection, Symbol Extraction or Symbol Recognition, Layout Analysis, and Mathematical Content Interpretation. Shown at left are the possible input formats, including vector-based document encodings such as PDF files, pen/finger strokes, and document images. The form of input and output for each problem is shown. Many systems perform recognition in the order shown, but not all. For example, some systems combine Layout Analysis and Mathematical Content Interpretation, producing an operator tree directly using the expected locations of operator/relation arguments [29, 31]. Post-processing stages used to apply language model constraints (e.g. n-grams) and other refinements are not shown (see Section 4.5).*

Math recognition also finds application in tutoring systems. For example, when middle school and high school students tested a math tutoring prototype (based on FFES/DRACULAE), students using pen entry completed their math tutoring sessions in half the time of those that typed, with no significant difference between their pre-to-post test score gains [7].

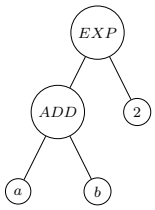
The following four key problems arise in the recognition of math notation, as illustrated in Figure 3.

1. *Expression Detection* (Section 4.1). Expressions must be first identified and segmented. Methods for detecting offset expressions are fairly robust, but the detection of expressions embedded in text lines remains a challenge.
2. *Symbol Extraction or Symbol Recognition* (Section 4.2). In vector-based representations, such as PDF, symbol locations and labels can be recovered, though some handling of special cases is needed (e.g. root symbols are often typeset with the upper horizontal bar represented separately from the radical sign, $\surd$ [14]). In raster image data and pen strokes, detecting symbol location and identity is challenging. There are hundreds of alphanumeric and mathematical symbols used, many so similar in appearance that some use of context is necessary for disambiguation (e.g. O, o, 0 [103]).
3. *Layout Analysis* (Section 4.3). Analysis of the spatial relationships between symbols is challenging. Spatial structure is often represented using a tree, which we term a symbol layout tree (Figure 4a). Symbol layout trees represent information similar to LaTeX math expressions; they indicate which groups of horizontally adjacent symbols share a baseline (writing line), along with subscript, superscript, above, below, and containment relationships. Symbols may be merged into tokens, in order to simplify later processing (e.g. function names and numeric constants).
4. *Mathematical Content Interpretation* (Section 4.4). Symbol layout is interpreted, mapping symbols and their layout in order to recover the variables, constants, operands and relations represented in an expression, and their *mathematical* syntax and semantics. This analysis produces a syntax tree for an expression known as an operator tree (Figure 4b). Given definitions for symbols and operations in an operator tree, the tree may be used to evaluate an expression, e.g. after mapping the tree to an expression in a CAS language such as Matlab, Maple, or Mathematica. However, determining the correct mapping for symbols and structures can be difficult, particularly if there is limited context available.

Figure 3 illustrates a series of stages commonly used in recognition of mathematical notation. The order of stages can vary [18]. Intermediate results produced by one stage may provide contextual information to constrain analysis in other stages, or to constrain the anal-



(a) Symbol layout tree. The tree is rooted at left ('('). Horizontally adjacency relationship edges are unlabeled



(b) Operator tree. The tree represents the addition of a and b, squared.

Fig. 4 *Symbol layout tree and operator tree for* $(a + b)^2$

ysis of other parts of the input. This is discussed further in Section 4.6.

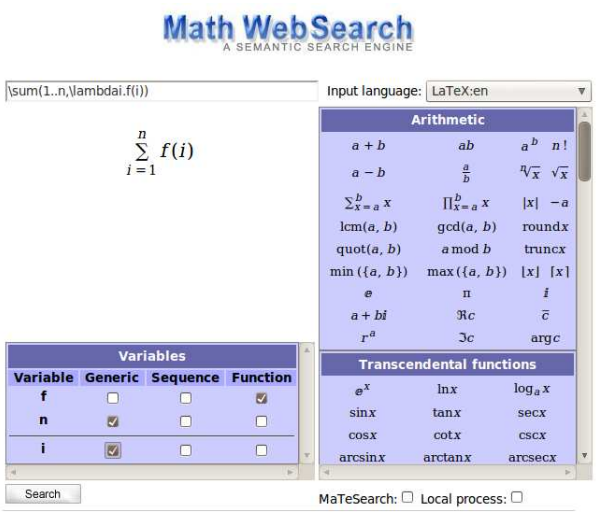
The first publicly available math-recognition systems appeared about a decade ago, building on math recognition research dating back to the late 1960's [5, 6, 17, 31]. The 1999 applet[1] created by Matsakis et. al recognizes simple handwritten expressions [99]. In 2001, Chen and Yeung published a paper on the first pen-based calculator [30]. In 2002, the FFES/DRACULAE pen-based equation editor[2] [135,165] was distributed as an open-source prototype. Several more recent systems recognize handwritten [81,133,144] and typeset [46] expressions. Commercial applications began to appear, including MathJournal[3], and pen-based entry in the Windows operating system [113]. The Infty math OCR system of Suzuki et. al has also been influential [71, 140]. Infty is sophisticated, and supports speech and Braille output for the visually impaired [140]. Infty supports both document image and pen-based input.

At present, most commercial systems for OCR do not recognize mathematical expressions. To address this, OCR output can be annotated with the results produced by a math recognition system. For example, the InftyReader[4] application (see Figure 1c) uses the Infty system to recognize expressions and insert corresponding LaTeX strings into the PDF file produced by a commercial OCR system [71].

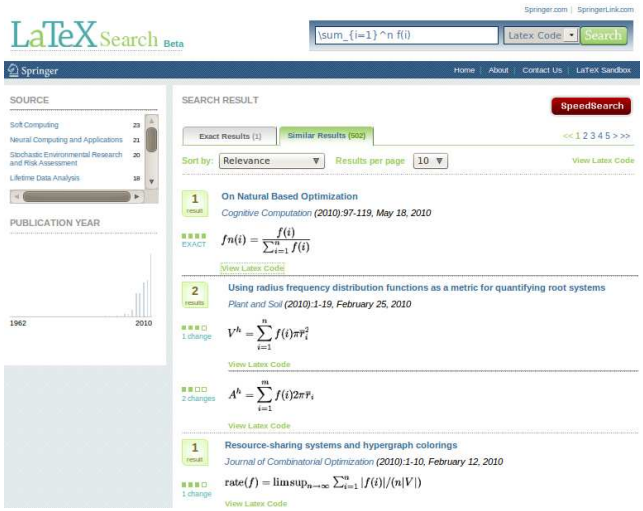
User interfaces for expression entry and recognition result visualization are important research topics that we will discuss only briefly here for reasons of space. In addition to the papers cited in Figures 1 and 2 and mentioned already, the interested reader should consult the following: [84, 118, 133, 169]. Key issues are ease of input, and visualization of feedback. One repeated observation of interest is that for pen-based systems, presenting recognition results separately from the user's input as a rendered image leads to situations where: 1) in experiments, participants find themselves unable to detect errors reported in the structure of their expression, not because they aren't shown, but because they have difficulty perceiving them [165, 169], and 2) users try to edit the recognized expression image, rather than the pen-based input [82, 169].

1.2 Overview of Mathematical Information Retrieval

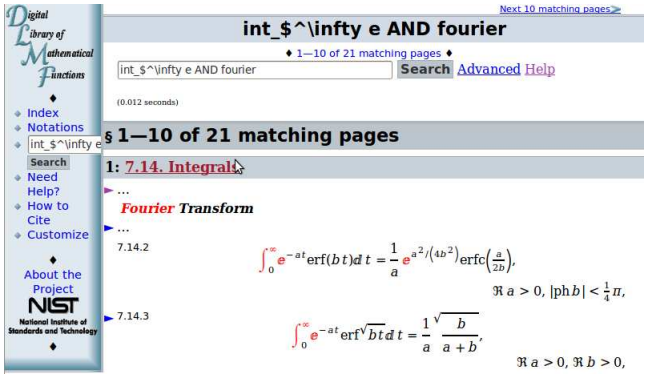
Figure 6 illustrates the information retrieval process. The user formulates queries through the Query Inter-



a. Math WebSearch Interface [77, 78]. Queries are constructed via keyboard and templates on the right. Symbol types may also be constrained (bottom left)



b. Springer LaTeX Search. Results may be filtered by clicking on a publication year or source document type



c. NIST Digital Library of Mathematical Functions. Shown are results for a boolean query combining math and keywords [3,102]

Fig. 5 *Mathematical Information Retrieval System Interfaces*

[1] http://www.ai.mit.edu/projects/natural-log/
[2] http://www.cs.rit.edu/~rlaz/ffes/
[3] http://www.xthink.com/
[4] http://www.inftyreader.org

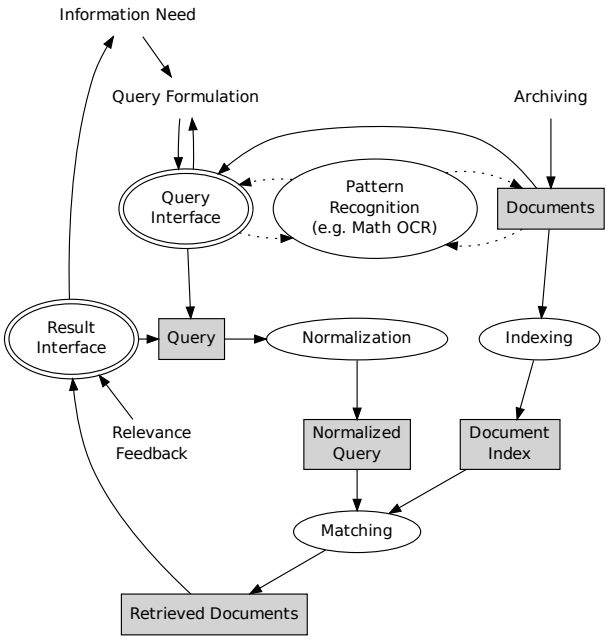


Fig. 6 *Information Retrieval (adapted from Hiemstra [62]). Data are shown in boxes, system processes in ovals, user interfaces in double ovals, and user elements with no surrounding shape. Beginning with an information need and query formulation in the top left, the user enters the query through the Query Interface. The query is normalized to reduce variation (e.g. text can be normalized using word stemming and a thesaurus). As shown on the right, the searchable documents are indexed; the indexing process also carries out normalization operations. At bottom, the normalized query and the indexed documents are matched, to produce the set of retrieved documents. The user views these through the Result Interface; the user can provide relevance feedback, or can elect to formulate a new query*

face, and views results through the Result Interface. Indexing, Normalization and Matching are three system processes used to process the document collection and query, and find matches for the query in the collection.

Math recognition can be applied both to the query (e.g. to recognize a stylus-drawn expression, as in Figures 1 and 2) and to the searchable documents (e.g. to recognize math expressions in document images or PDF files). Prior to indexing, document images can be annotated with region types (e.g. text, table, figure, image, math), character information, and recognized structure and semantics for detected math expressions. Existing math retrieval systems lack the ability to recognize stylus-drawn queries. Instead template editors are provided to assist in generating query strings; an example is the Math WebSearch prototype (Figure 5a).

The following four key problems arise in the retrieval of math notation, as illustrated in Figure 6.

1. *Query Languages and Query Formulation* (Section 3.1). Present-day query languages for mathematical information retrieval are text-based, influenced by LaTeX, MathML [10] and OpenMath [37, 148]. Challenges in query formulation include determining what types of queries are useful and feasible, and providing an effective user interface for query formulation.
2. *Normalization* (Section 3.2). In order to reduce variation, both the query and the searchable documents are normalized. In text-based retrieval, normalization involves word stemming and thesaurus operations [125]. Similarly, expressions must be reduced to canonical forms to prevent mismatches between equivalent expressions with different representations. For example, normalization of symbol layout trees imposes a unique ordering on spatial relationships. As another example, enumeration of variables in operator trees allows variables to be matched without concern for their specific symbol identities.
3. *Indexing and Matching* (Section 3.3). Retrieval performance depends heavily on the chosen document representation, and on the similarity measures used to compare queries to the index. Vector, image and stroke data need to be indexed and retrieved using different methods. At present, we know of no work concerned specifically with indexing and retrieving handwritten mathematical documents.
4. *Relevance Feedback* (Section 3.4). During examination of a retrieval result, the user can provide relevance feedback, to allow the system to automatically construct a refined query. This is an important, but currently unexplored research direction for math retrieval systems. Relevance feedback has been studied intensively in text [125] and image-based retrieval systems [35, 132].

In addition to these four key problems, the evaluation of a math retrieval system is also difficult. Evaluation is discussed in Section 3.5.

Mathematical Information Retrieval (MIR) is a relatively new research area, lying at the intersection of text-based information retrieval [62, 125], content-based image retrieval [35, 38, 132] and Mathematical Knowledge Management (MKM [25]). Mathematical knowledge management is concerned with the representation, archiving, extraction, and use of mathematical information. Systems for mathematical information retrieval have been developed for a variety of applications:

– Finding equations in a database of technical documents [8, 100, 101] (e.g. Springer LaTeXSearch[5])

[5] `http://www.latexsearch.com/`

- Semantic search for expressions on the internet (e.g. Math WebSearch[6] [77, 78])
- Finding functions in mathematical function libraries such as the NIST Digital Library of Mathematical Functions[7] and Wolfram Functions Site[8]. In these systems, partial definitions may be used to locate complete equations [75, 78, 106]
- Supporting equation search in online learning tools (e.g. ActiveMath [91]).
- Searching integral tables [41]
- Supporting proof assistants such as Coq [9]

It is interesting to compare question-answering systems to information retrieval systems. For textual data, Salton distinguishes these two types of systems based on the types of data stored and the form queries take [125]. Information retrieval systems use stored data consisting of documents; in contrast, question-answering systems use stored data consisting of facts and general knowledge. Queries in information retrieval systems take the form of keywords and excerpts; queries in question-answering systems use natural language. Recently, question-answering systems for mathematical information have been devised [171]. An example is the well-known Wolfram Alpha web site[9]. The Wolfram Alpha knowledge base includes facts on mathematics and statistics, along with many other topics including the sciences, technology, finance, culture, and geography. Wolfram Alpha provides some processing for natural language (though keywords may be used), and responses are returned using a table of relevant facts, figures and computations. For example, users may request that the system factor a polynomial.

Investigation of image-based math retrieval has recently begun. Retrieval is based on the similarity of math notation images, without recognizing their math content. For example, Marinai et al. propose a method based on shape contexts for retrieving mathematical symbols [96], while Yu and Zanibbi propose a retrieval method in which handwritten queries are matched to document images using a combination of X-Y cutting and word shape matching [161, 167].

According to the framework of Smeulders et. al [132], math images are a 'narrow' image retrieval domain, with constrained semantics and very controlled scene and sensor properties. For example, math images tend to have stable illumination. However, the math domain does present challenges: images of math are *polysemic*, meaning that a single expression may be interpreted in multiple related ways. For example, the meaning or value of an expression varies depending on the variable binding, the type of a variable (e.g. natural, integral, real, or complex), and the interpretation of operators and functions (e.g. the function 'f' is heavily overloaded). It can be difficult to deduce which interpretation was intended by the author of a math expression. Some clues may be found elsewhere in the document (e.g. definitions of symbols and functions), but often it is necessary to draw on knowledge of the notational conventions used in a certain branch of mathematics.

Having provided an overview of math recognition and retrieval, in the next section we summarize mathematical notation and issues related to the representation and interpretation of mathematical expressions. In the remaining sections we continue our discussion of math recognition and retrieval in more detail.

2 Mathematical Notation

In this section we provide a brief overview of mathematical notation and file formats used to represent mathematics. Math notation may be understood as a semi-formal visual language [97]. As with other two-dimensional notations such as chemical diagrams, music notation, and flowcharts, math notation is a graphical language for representing complex interactions between primitive objects [21]. Defining math notation is difficult, but some resources for study are available, including books on typesetting for mathematics [33, 63, 74, 157], and a history of the origins and evolution of the notation [24]. For both people and machines, interpreting the notation provides many challenges: the set of symbols used is very large, and ambiguities and context-dependencies arise in interpreting symbol identity, layout, and semantics (see Figure 7).

In math notation, symbols are used to represent constants (e.g. $\pi, e, 0$), variables (e.g. a, α), operators, functions and relations (e.g. $\int$, fraction lines, f, cos, $<$), and the scope of subexpressions (e.g. grouping using (), [ ], {}). Unlike primitive arguments or objects in an expression, operations, functions, relations and subexpression scopes are also represented *implicitly*, using the spatial arrangement of symbols (e.g. the implicit multiplication in xy). Table 1 summarizes the six spatial relationships commonly used in isolated expressions. Both subscripts and superscripts can be placed to the left of the symbol or subexpression they modify, as in the Table 1 example for 'n choose k.' Most math recognition systems do not currently accommodate these 'prefix' super/subscripts, because they are rare.

Subexpression scopes are often represented using grid (or 'tabular') layouts, where subexpressions are ar-

[6] http://search.mathweb.org/index.xhtml
[7] http://dlmf.nist.gov/
[8] http://functions.wolfram.com/
[9] http://www.wolframalpha.com

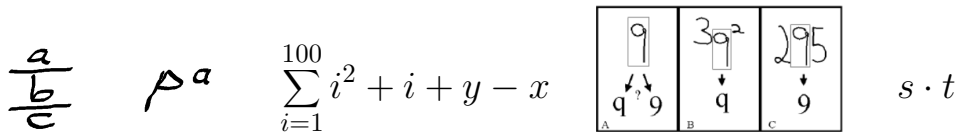


Fig. 7 *Ambiguous Mathematical Expressions. (a) Which division is performed first? (b) Is* a *superscripted? (c) What is the scope of the summation? (d) Is this symbol a 9 or a q? The perceived answer depends on context (from [103]) (e) What do s, t and $\cdot$ represent?*

Table 1 Spatial Relationships in Mathematical Notation. Relationships shown are defined for standard symbol layout tree encodings (e.g. LaTeX, Presentation MathML), and used in most recognition systems (as far back as Anderson's [5]). Note that for many expressions shown, mathematical content cannot be determined unambiguously.

Relation	Expression	Math. Interpretation
Adjacent	xy	Multiply x by y
(at right)	$x \times y$	Multiply x by y
Superscript	x^3	$x \times x \times x$
Subscript	x_1	Element 1 of list x
	x_1^2	$x_1 \times x_1$
	$\int_{-\infty}^{+\infty} p(x\|\omega_i)\ dx$	Integrate density function p over all vectors x for class ω_i
	${}_nC^k$	n choose k
Above	$\overline{x}$	not x
Below	$\frac{x}{y}$	x divided by y
	$\sum_{i=1}^{n} i$	Add $1, 2, \ldots, n-1, n$
Contains	$\sqrt{x^2 y^2}$	xy
Grid Layout: rows, columns contain subexpressions		
Grid	$\begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$	2×2 diagonal matrix
Nested Grid	$x! = \begin{cases} 1, & if\ x = 0 \\ x((x-1)!), & if\ x > 0 \end{cases}$	Inductive function def.

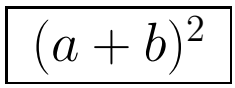
ranged in rows and columns. An example is shown at the bottom of Table 1. Grid layouts are also used frequently in derivations. A number of well-known symbol shorthands are used to represent patterns and repeated matrix elements; these include ellipses (e.g. $x_1 \ \ldots \ x_n$), lines, and large symbols such as a large 0 to represent zeros in the upper-triangular region of a matrix.

Mathematical expressions represent an application of functions, operators and relations to arguments. As can be seen in Table 1, *multiple* mathematical statements may be represented by a single expression; in other words, mathematical expressions are *polysemic*. For example, if x is a list the expression x_1 can represent the first or second element in the list. The definition and even role of symbols frequently change; for example, in an arbitrary expression, λ can represent a variable, a constant or a binding function as in the Lambda Calculus. Even when the domain is clear, symbol definitions are often ambiguous. Consider P in the context of Bayesian probability: is P used to represent a probability mass function or a probability density function?

Without knowing the precedence and associativity of operations, the order in which operations are to be applied and relations tested may be unclear. For example, in Table 1, x_1^2 is indicated as representing the square of x_1; in another context, this might be representing a restriction on sequence x^2, where the precedence of operations is reversed. The precedence of operators is determined using the following [19]:

- *Operator range* defines legal spatial locations for arguments of an operator or relation (e.g. for '+', or fractions)
- *Operator dominance* (Chang [31]), defines a partial ordering on the application of operators and relation predicates. An operator/relation which nests completely within the range of another operator/relation is said to be *dominated*. For example, the + in $(x+y)/2$ is dominated by the fraction line. Dominating operators are applied *after* the operators they dominate.
- *Operator associativity* orders application when two or more of the same operator appear in each others' range. For example, addition is normally left-associative: $x + y + z = (x + y) + z$.
- *Operator precedence* orders the application of different operators when they are within each others' range. For example, $2 + x \times y = 2 + (x \times y)$.

An unambiguous definition for operator range, dominance, associativity and precedence imposes a unique evaluation order on an expression. The result may be represented as an operator tree, with operators and relations at internal nodes, and constants and variables at the leaves (see Figure 4b).



(a) Expression Image

```
<msup>
    <mfenced>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
    </mfenced>
    <mn>2</mn>
</msup>
```

(b) Presentation MathML [10] (Symbol Layout Tree)

```
(a+b)^2
```

(c) LaTeX (Symbol Layout Tree)

```
<apply>
    <power/>
    <apply>
        <plus/>
        <ci>a</ci>
        <ci>b</ci>
    </apply>
    <cn>2</cn>
</apply>
```

(d) Content MathML (Operator Tree)

Fig. 8 *Math File Encodings (adapted from [1])*

However, some expressions are not intended for evaluation. For example, consider the integral shown in Table 1. The vector space is continuous, and thus this integral cannot be computed directly. Doing so would also not be of interest, as this expression is commonly used in a *constraint* that the expression needs to evaluate to 1.0.

We now briefly describe file formats used for symbol layout trees and operator trees. Symbol layout trees represent the placement of symbols on baselines (writing lines), and the spatial arrangement of the baselines. File formats for representing symbol layout trees include Presentation MathML and LaTeX, as shown in Figure 8b and c. Compared to LaTeX, Presentation MathML contains additional tags to identify symbols types; these are primarily for formatting. Grid layouts are represented by rows and columns of subexpressions (e.g. using the `array` construct in LaTeX), with each subexpression represented by a symbol layout tree or grid. Grids may occur as subexpressions in symbol layout trees, as in the factorial function definition in Table 1: the main baseline of the expression consists of $x! = \{[sub]$, where $[sub]$ represents a grid containing four subexpressions (two value–condition pairs) used to define the function.

An operator tree, as shown in Figure 4b, represents the operator and relation syntax for an expression. Operator trees may be encoded in a number of ways, including Content MathML and OpenMath [36, 37]. To evaluate an expression, it is necessary to know the definitions for all symbols and operations. As shown in Figure 8d, tags in Content MathML represent defined primitives (e.g. `<cn>2</cn>`), operations (e.g. `< plus/>`) and relations. The OpenMath standard provides an encoding for formalizing the semantics of symbols and operations using *content dictionaries*. Given this information, an expression may be evaluated mechanically, using a Computer Algebra System.

3 Mathematical Information Retrieval

Figure 6 summarizes the process of information retrieval. In general, users have an *information need* that they attempt to satisfy using the retrieval system. Information needs take many forms (Table 2), and are seldom concrete: often, they change as a user interacts with a retrieval system. Consider image retrieval: Smeulders et. al point out that often users' impression of the images they want are only partially defined, such as when looking for an image belonging to a class of objects (e.g. chairs), or not defined at all, as when browsing through an image collection [132]. A discussion of research on information needs, including difficulties associated with their observation and common misconceptions, is provided by Case [26] Chs. 1 and 4. Research on image search needs and behaviours is summarized by Westman [156].

A better understanding of users' information needs will further the development of MIR systems. At present, MIR research has been motivated primarily by developing new search techniques based on query-by-expression [75, 171]. Better response to information needs will allow MIR to mirror the advances in internet search interfaces over the last two decades [61]. In a study of MIR usage, Zhao et al. report that participant queries may be motivated by a specific information need, such as the need for a definition or derivation [171]. In addition to information needs, participants expressed *resource needs*, requesting resources with a certain style and depth of presentation (e.g. tutorials versus research papers), or requesting resources with a particular function (e.g. written documents, including slides and web pages, versus code and data sets).

General-purpose search engines such as Google can be used to locate mathematical content, but the results may be weak in relation to the user's goals, as these systems use term-based indexing with no model for mathematical content. For example, one can try matching MathML tags, or matching the LaTeX strings that occur in some web pages as annotations for the expres-

Table 2 *Information Needs for Mathematical Information Retrieval, from Kolhase and Kolhase [75], and Zhao et al. [171]*

	Information Need
1	Specific/similar formulae
	· Form/appearance (given by a symbol layout tree)
	· Mathematical Content (given by an operator tree)
	· Name
2	Theorems, proofs, and counter-examples
3	Examples and visualizations (e.g. graphs/charts)
4	Problem and solution sets (e.g. for instruction)
5	Algorithms
6	Applications (e.g. for the Fourier transform)
7	Answer mathematical questions/conjectures
8	People (by math content in publications)
9	Determine novelty/sequence of mathematical discoveries

sion images they were used to create. It seems likely that as MIR research advances, users will continue to use a combination of general-purpose search engines along with specialized MIR systems for their mathematical information needs, as was observed in Zhao et al.'s study [171].

In the remainder of this section we address four key problems in MIR: query formulation and languages for expression queries, normalization of queries and documents, document indexing and matching, and query refinement and relevance feedback. The final section discusses evaluation of MIR systems.

3.1 Query Languages and Query Construction

Systems for MIR using standard keyword-based query languages (see [125], Ch. 2) have existed for quite some time. Examples include the web pages for searching Mathematical Reviews[10] and Zentralblatt für Mathematik[11]. Both services have been compiling bibliographies and disseminating reviews of published work on mathematics since the first half of the twentieth century. Their materials have been manually indexed, using the Mathematical Subject Classification (MSC) [121].[12] In the web interfaces provided by these services, MSC categories can be used to constrain searches.

To make existing text-based query languages better suited to MIR, researchers are extending them with syntax expressing the appearance and content for mathematical expressions (e.g. using LaTeX and MathML). Also, content-based image retrieval (CBIR) methods [35, 132] can be adapted to allow expression images to be used directly as queries.

Expressions have been represented in MIR query languages using Lisp [41], LaTeX and LaTeX-like string languages [3, 9, 102], Mathematica (for Wolfram web sites), MathML [78], and operator tree shorthands [77]. Example queries are shown in Figure 5. Recently, images of symbols [96] and complete expressions (handwritten [161, 167] and typeset [168]) have been used for query-by-expression.

To make expression queries more precise, boolean constraints (AND, OR, NOT) may be used [78, 91], and cardinality and matching constraints added. Figure 5c shows an example of a simple boolean constraint in a query language supporting both expression and keyword matching. Wildcards to permit matching any symbol or subtree at a specified point in an expression have also been used [9, 77, 105]. An example is shown in Figure 5c, where the wildcard character $ matches any subscript on the integral. Altamimi and Youssef use an AWK-like syntax [2] and regular expression patterns to identify matching subexpressions, and allow equivalence and type constraints to be imposed on matched entities [3]. Constraints can also be applied to indicate which document regions to match; an example is indicating a preference for theorems, proofs, and section headings demarcated within the document collection [102, 171].

A variety of query interfaces for MIR have been proposed, a small number of which we summarize here. The simplest interfaces provide a box in which to type a query string, such as used in the Springer LaTeX search interface and the NIST Digital Library of Mathematical Functions (see Figure 5a and c). The MathWebSearch interface shown in Figure 5a [78], provides templates for structures such as fractions and summations; text representing these operations is inserted into the query using a mouse click. In the Mathdex system, users can enter expressions using a graphical equation editor similar to the editors provided in word-processing programs [104].

Query expressions constructed using string languages and template editors tend to contain a small number of symbols (see Figure 5). Single-symbol query expressions are imprecise, while query expressions containing a large number of symbols are uncommon, because of the effort required to express and interpret them [59,67]. The rarity of large query expressions is an example of the *principle of least effort* [173] commonly observed for natural language (see p. 60 of Salton and McGill [125]). In contrast, large queries are easy to construct when queries are expression images: a user can easily select large image regions, so a large number of symbols does not affect the effort involved in query construction.

Despite the efforts to add expressions into query languages, their addition may not always add value for users [75]. Zhao et al. studied a small group of profes-

[10] http://www.ams.org/mr-database

[11] http://www.zentralblatt-math.org/zmath/en/

[12] The MSC is quite detailed; the 2010 revision is 47 pages long.

sors, graduate students and librarians affiliated with the Math Department at the National University of Singapore, and found that most of their participants could not identify a situation where they would want to search using an expression [171]. Expressions are often named (e.g. the Pythagorean theorem), may be overly specific for some information needs, and may be inconvenient to enter using the methods known to the participants, which included graphical template editors and string-based interfaces (image-based querying was not considered). When asked what their preferred expression entry method would be, participants responded that they would like to use LaTeX, due to its familiarity.

Kohlhase and Kohlhase suggest pen-based entry may be a more natural expression input modality [75]. We propose that pen-based entry will be most effective when paired with keyboard and mouse input. There should also be support for query-by-example, in which queries are constructed using expression images from the document collection. It remains to be seen whether such an interface would make query-by-expression more appealing to math experts such as those in Zhao et al.'s study.

As MIR matures, we expect the ability to browse expressions and their surrounding text within a single document or document collection will be useful, particularly for *non-expert* users in elementary school and high school, and in technical disciplines.

3.2 Query and Document Normalization

For information retrieval, *normalization* is the process of reducing variation within queries and documents, to facilitate matches between related or identical entities with different representations. In textual IR, common normalization operations include replacing words by their stems (e.g. 'information' → 'inform' and 'retrieval' → 'retriev' [125]), and the removal of high-frequency, low-discrimination *stop words* such as *but*, *to* and *the*. Often a thesaurus is used to add synonyms for low-frequency terms to the query.

The normalizations that are performed for math retrieval depend on the representation (symbol layout tree vs. operator tree), and on the matching algorithm used for search. For example, the order in which spatial relationships are presented is critical in systems that match symbol layout trees that have been linearized. Identical expressions will fail to be matched if relationships appear in different orders, as in `x^2_1` and `x_1^2`. Standardized ordering is also needed in operator trees, as ultimately the tree structure is used in matching.

Analogous to synonyms in text, mathematical concepts often have multiple notational representations. Consider 'n choose k', which may be written as $\binom{n}{k}$, ${}_nC^k$, C^n_k, or C^k_n [78]. In terms of expression semantics, the variability is even more severe: consider the number of expressions that evaluate to 0. It is not clear when or to what extent transformation and simplification should be used to recover such equivalences.

Below is a short list of query and document normalizations that have been applied in MIR systems.

- **Thesaurus:** adding synonyms for symbols to a query (e.g. adding equivalent function names [102]).
- **Canonical orderings:** fixing the order for spatial relationships such as subscripts and superscripts in symbol layout trees (e.g. expressed in LaTeX [102]), and defining a fixed ordering for children of associative and commutative operations in operator trees, such as for sums [109, 129].
- **Enumerating variables:** variables may be enumerated (ignoring symbol identities) to permit unification of query variables with variables in archived expressions [109].
- **Replacing symbols with their types:** allows matching symbol types around an operator, rather than specific symbols [67]. It also allows for a sub-expression to be matched to an individual symbol of a given type.
- **Simplification:** produce smaller representations with less variation. For example, one may eliminate `<apply>` tags (see Figure 8) from Content MathML [160], or use Computer Algebra Systems to simplify expressions symbolically [41, 102].

3.3 Indexing and Retrieval

Most MIR research assumes that mathematical expressions are represented explicitly in the document collection, using markup languages such as LaTeX, MathML [10] or OpenMath [37, 139]. These encodings allow expression appearance or mathematical content to be extracted directly and then embedded in documents or evaluated using CAS systems. New languages, formats, and tools for creating mathematical documents have also been developed.

The OMDoc format developed by Kohlhase [76] is XML-based, allowing expressions to be embedded using MathML and OpenMath. OMDoc was used to represent documents for Math WebSearch (see Figure 5a), and ActiveMath, an on-line math tutoring system that supports query-by-expression [91]. Miller created LaTeXML, a tool for translating LaTeX to XHTML and MathML [102]. This is analogous to the well-known `latex2html` converter used to translate LaTeX documents to HTML, embedding mathematical expressions as images (e.g.

.png files). LaTeXML was used in creating the NIST Digital Library of Mathematical Functions (DLMF) (see Figure 5c). In contrast, Springer's LaTeX search (Figure 5b) represents documents using the LaTeX sources provided directly by the authors of academic papers and books. These encodings allow expression data to be represented explicitly, in a suitable form for indexing and retrieval prior to archiving a document collection.

Unfortunately, many documents do not represent mathematical information explicitly. Examples include document images such as .tiff or .png files, and vector-based representations such as .pdf files [13, 14]. This makes it necessary to recover mathematical information using pattern recognition techniques, and then annotate documents with recognition results prior to indexing. Pattern recognition has been used to identify math symbols and structure in raw document images [8, 101] and .pdf files [14, 71]. Another use of pattern recognition is to segment documents into region types such as theorem, proof, and section heading [171]; these region types can then be used in queries.

A German and Japanese project led by Michler developed a prototype for annotating documents in digital mathematics libraries in the early 2000's [100,101]. Document images were recognized using commercial OCR software (ABBYY FineReader), mathematical expressions were segmented and converted into LaTeX using techniques developed by Okamoto et al. [8], and paper references were linked to online reviews from Zentralblatt für Mathematik and Mathematical Reviews. References were detected using regular-expression matching in OCR results. Archived documents were stored using the DjVu format, which represents document pages in three layers: 1. image, 2. OCR and math recognition results, including associated page coordinates, and 3. links to reviews for cited papers, with the associated page coordinates for the citations [101]. DjVu viewers allowed OCR/math recognition results to be seen inplace while viewing a document image, and for reviews of references to be consulted simply by selecting a reference (e.g. using a mouse click).

During indexing, documents are converted to the representation used in the *document index*. In the early stages of indexing, documents are filtered (e.g. to select expressions and/or index terms) and normalized in the same fashion as queries.

3.3.1 Vector-Space Models

In vector-space models, documents are represented by vectors in $\mathcal{R}^n$, where each dimension corresponds to an index term [62, 95, 125]. Index terms normally exclude *stop words* (very high frequency terms such as 'the' that carry little information) as well as highly infrequent terms, whose inclusion would have little effect on retrieval performace, while increasing the dimensionality of the vector space. Salton and McGill discuss index term selection, the use of synonyms for low frequency terms, and the construction of term phrases for high frequency terms (Ch. 3 of [125]). Documents are represented by the weighted number of occurrences of each index term (the *term frequencies*). Commonly, term frequencies are weighted using some variation of *inverse document frequency*, to emphasize terms that appear in fewer documents in the collection, and thereby likely to be more informative [62, 125]:

$$u_i = freq(i,u) \cdot \log \frac{N}{docfreq(i)}$$

where $freq(i,u)$ is the frequency (occurrence count) for term i in document u, $docfreq(i)$ is the number of documents containing term i, and N is the number of documents in the collection.

The most common similarity measure used is the cosine of the angle between two document vectors u_i and v_i:

$$sim(u,v) = cos(u,v) = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

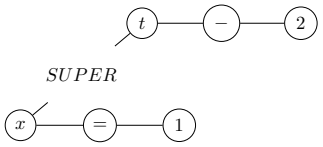
This is simply the inner product of the document vectors divided by the product of their magnitudes. If term vectors are first normalized (length 1.0), then the denominator need not be computed. $sim(u,v)$ has a value of 1 when the vectors coincide (0°), and 0 when the vectors are orthogonal (±90°).

For large document collections, the document index must be pre-structured to reduce the number of comparisons made for a query. A common approach uses clustering, and then compares a query vector with the centroid of each child cluster at a node (Ch. 6.4 of [125]). The cluster tree is traversed top-down until individual documents are reached, pruning paths in which similarity is less than a threshold value. This greatly reduces retrieval time, but carries the risk that the document(s) most similar to the query will not be located (see [40] pp. 185-186). Smeulders et al. identify three methods for hierarchically decomposing a document index in image retrieval [132]: partitioning the feature space, partitioning the data, or distance-based indexing relative to examples. Spatial data structures used by these three decomposition approaches, respectively, include k-d trees, R-trees, and M-trees [126].

A number of MIR systems implement vector-space models using the popular Lucene[13] [60] indexing and retrieval library, both for indexing entire documents that

[13] http://lucene.apache.org

include expressions [91, 102], and for indexing individual expressions in LaTeX documents [168]. In these approaches, mathematical symbols are treated as terms, and the expressions are linearized ('flattened') before conventional text-based indexing is performed. For example, consider the LaTeX expression for $x^{t-2} = 1$, which is `x^{t-2} = 1`. Below we show the symbol layout tree for the LaTeX expression, along with the linearization produced by Miller and Youssef [102]:




```
x BeginExpt t minus 2 EndExpt Eq 1
```

This string is a depth-first linearization of the symbol layout tree for the expression. Note that the exponent scope is represented by folding the LaTeX superscript operator into the fence tokens `BeginExpt` and `EndExpt`. For the ActiveMath system, OMDoc is used to encode the document collection, and OpenMath representations for expression *operator trees* are extracted and linearized depth-first in a manner similar to the example above [91]. Once mathematical expressions have been converted, documents are indexed using traditional term-based indexing methods. Lucene may be used to automatically determine the set of index terms for use in indexing and retrieval.

3.3.2 Tree-Based Indexing and Retrieval

Other methods for indexing and retrieving math expressions use the hierarchical structure in layout and operator trees. The hierarchical structure can be used in its entirety, or as a set of trees representing subtrees of the expression. Retrieval is performed using subexpressions extracted from the query expression.

Matching operator trees may be viewed as a variation of the *unification* problem addressed in automated reasoning systems: given a query expression, identify indexed expressions whose variables and/or subexpressions may be matched consistently with those of the query. Graf developed a term indexing method for first-order logic known as *substitution tree indexing* [57]. A substitution tree represents the structure of all indexed first-order logic terms, with paths from the root to the leaf defining a sequence of variable substitutions. Substitution trees can be adapted for indexing operator trees in a straightforward manner, as illustrated in Figure 9.

Retrieval in a substitution tree is performed through a backtracking search over variable bindings (similar to Prolog [57]). Using different matching functions, we may search for exact matches, instances, generalizations, and variant substitutions. An example of instance-based matching using Figure 9 is that the query $sqrt(X)$ returns the three expressions at the leaves of the tree that contain an outermost $sqrt()$. An example of matching with generalizations is to ignore specific symbol identities. In matching with variant substitutions, we match expressions that are equivalent up to variable renaming.

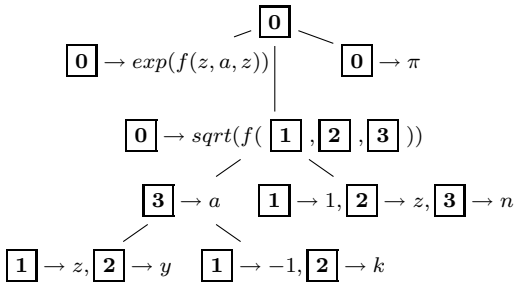


Fig. 9 A Substitution Tree (adapted from Kohlhase and Sucan [78]). The tree represents all indexed expressions using paths of substitutions. Substitution variables are represented by boxed numbers. Five expressions are represented at the leaves of the tree: $exp(f(z, a, z))$, $sqrt(f(z, y, a))$, $sqrt(f(-1, k, a))$, $sqrt(f(1, z, n))$ and π.

Substitution tree retrieval was applied to MIR by Kohlhase and Sucan [78]. To simplify matching sub-expressions, Kohlhase and Sucan add all sub-expressions in the document collection to the substitution tree along with their parent expression. They claim that this leads to a manageable increase in the index size, because many sub-expressions are shared by the larger expressions, and each sub-expression appears only once in the substitution tree. To facilitate rapid retrieval, all substitution tree nodes contain references to matched expressions in the document collection.

Earlier, a related method was used by Einwohner and Fateman for searching through integral tables, given an integrand expressed as an operator tree in Lisp (e.g. `(expt (log (cos x)) 1/2)`) [41]. Expressions from the integral tables were indexed using hash tables: after normalization of the Lisp expressions, the head (first atom) of each list in the lisp expression is used as the key for storing the associated sub-expression (sub-tree) in the table. Retrieval was performed by recursively looking up each lead atom (key); if the first key returns a non-empty set of expressions, the current key is expanded to include the next key, and the intersection of the previous returned and current lists of matches is taken. This differs from the substitution trees in that operator trees are matched using a depth-first traversal of the query operator tree rather than based on com-

mon substitutions that may not be strictly depth-first, and symbols are matched exactly.

Hashimoto et al. generate an index using paths from the root of the tree for Presentation MathML expressions [59]. Expressions are indexed in an inverted file using two paths: the first (leftmost) and the deepest paths from the root of the tree to a leaf. Retrieval is performed based on the longest partial match along the two paths. The authors also consider producing inverted files using the nodes at the first depth with more than three nodes.

Kamali and Tompa propose rewriting trees and computing relevance using a set-based measure [67], in the context of Content MathML (an operator tree encoding). Intersections between nodes in two operator trees are defined using syntactic equivalences (possibly involving transformations, e.g. to detect $a + b = b + a$) with a noise/mismatch tolerance. A weighting function ω is used to weight trees by the nodes they contain, most simply counting nodes in the tree:

$$sim_{tree} = \frac{\omega(T_1 \cap T_2)}{\omega(T_1) + \omega(T_2)}$$

This is closely related to the Tanimoto metric for set similarity ([40], p. 188). This approach resembles the *graph probing* methodology for comparing table structure recognition algorithms [64, 92].

Miner and Munavalli [104] take a different approach, in which symbol layout trees expressed in Presentation MathML are decomposed into a set of n-grams (linearized sub-expressions). In their formulation 1-grams are single symbols; higher n-grams are defined by the number of children of a node in the MathML tree (i.e. there may be more than 5 symbols in a '5-gram'). In the symbol layout tree, weights are assigned to 'n-grams' associated with nodes based on their depth in the tree, structural complexity, and length (the 'n' for the associated n-gram). A threshold is then used to select nodes for use in querying: roughly speaking this prefers larger, and more complex sub-expressions. Expressions are indexed based on the linearized 'n-grams', and retrieval is performed by combining queries issued to a Lucene implementation.

In image-based MIR, representations for symbol layout trees have been constructed using X-Y cutting to decompose document pages and expression regions [161]. Recursive binary X-Y cuts decompose each page image, and subtrees of the X-Y tree up to a maximum depth and number of components are stored in a single expression index. Indexed regions are then re-segmented using a simplified X-Y cutting, to approximate symbol layout trees for expressions present in the candidate set. Previously, pixel projection profile methods with post-processing were used successfully to recover symbol layout trees from expression images by Okamoto et al. [111, 153]. Retrieval is performed using (standard) XY-tree structure, and dynamic time warping of query and candidate image columns similar to the word-spotting technique of Rath and Manmatha [119, 120].

A related approach was developed for visual matching of LaTeX-generated expression images [168]. Connected components in the query image are matched with connected components in archived images using visual similarity of connected components, again based on features similar to Rath and Manmatha's. The matching process also measures similarity in layout between pairs of connected components.

3.4 Query Reformulation and Relevance Feedback

After query submission the retrieved documents are presented to the user through an interface. In order to support reformulation of queries, one interface is normally used both for constructing queries and evaluating results, as seen in Figure 5. If a user's information need is satisfied by a retrieval result or if the user becomes frustrated, he or she will stop searching. Otherwise the user may craft a new query or may refine the existing query, for example by filtering retrieved documents by source or publication year (Figure 5b). New queries may also be created automatically, in response to relevance feedback.

Users provide relevance feedback by indicating whether returned documents are relevant or irrelevant to their information need. These positive and negative examples can be used to automatically produce a new query. Relevance feedback is provide through the result interface, using a selection mechanism such as check boxes, or clicking on relevant/irrelevant objects. For interesting examples from image retrieval, see [123].

For vector-space models, a new query may be produced by averaging and re-weighting the vector elements that define the feature space: increase the weights for features present in positive examples, and decrease the weights for features in negative examples. A concise explanation of relevance feedback operations using re-weighting is given by Salton and McGill [125] Chs. 4.2.B, 4.3.B and 6.5. Machine-learning methods have also been investigated. Discriminative methods estimate classification boundaries for relevant and irrelevant documents, whereas generative methods estimate probability distributions [35, 172].

Ideally, relevance feedback algorithms learn optimal transformations of the feature space using user-provided relevance indications [172]. Optimality is defined by the user's information need, which may change

as the user interacts with the system [35]. Modifications produced through relevance feedback may occur in multiple ways: the set of searched documents may be modified, the feature representation changed, or the similarity metric modified. For annotated images, the relationships between text annotations and image features are often exploited, e.g. producing 'concept classes' for sets of images that have similar annotations [132].

At the time of this writing, the authors are unaware of any work on relevance feedback for MIR. In text-based retrieval, Hearst has noted that despite significant improvements for text-based retrieval in laboratory experiments when relevance feedback is used, modern search interfaces tend not to provide a relevance mechanism (see [61], Ch. 6). Instead, they make metadata visible for query refinement (e.g. Figure 5b), or suggest alternate queries. In contrast, for image-based retrieval systems using query-by-example, relevance feedback is essential for a usable system, and is an active area of research. Zhou and Huang have suggested two reasons for this [172]: 1) images are more ambiguous than words, and 2) evaluating the relevance of text documents may require more effort than evaluating the relevance of images.

For MIR, it may often be faster to discern the relevance of a document based on the appearance of expressions than based on the document text, particularly in the case where a user is browsing rather than searching for a specific item as in done in Zhao et al.'s experiment [171]. This distinction between retrieval tasks involving a specific item vs. a class of items or browsing is important in information retrieval [35, 132]. In addition to using expressions within queries, returned expressions may be used for relevance feedback. A revised query can be generated based on the relevant and non-relevant expressions' visual appearance, symbol layout, mathematical content and associated text. We feel that this is an important future research direction.

3.5 Evaluation of Math Retrieval Systems

Evaluation of information retrieval systems is difficult due to variation in the information needs of individual users, and the impracticality of having participants in human experiments identify all relevant documents in large collections (see [125] Ch. 5, [22] and [132]). This leads to the definition of *relevance* being inherently subjective.

In practice, it is necessary to either define test sets for a pre-defined collection, query set, and relevance assessments as done for many of the NIST TREC retrieval competitions,[14] or to perform user-centered evaluations where searching behavior within real workflows (e.g. [75,171]) or constructed task scenarios is observed, with assessments provided by participants regarding the satisfaction of their information needs [22]. For off-line experiments such as those done for TREC, relevance assessments are usually binary (relevant/non-relevant) and produced before an experiment is run. In contrast, user-centered experiments permit relevance evaluations to be made using a scale, and allow relevance evaluations to change during iterations of relevance feedback. Constructed task scenarios paired with pre-defined relevance assessments allow off-line as well as user-centered metrics to be collected [22]. Hearst provides guidelines for evaluating retrieval interfaces [61].

The standard metrics for off-line retrieval are *recall* (% of relevant documents retrieved) and *precision* (% of retrieved documents that are relevant). There is a well-understood trade-off between the metrics: as more relevant items are located (higher recall), the number of irrelevant items returned generally increases (lower precision), and vice versa. Relevance assessments by human participants normally consider just the first k elements returned. This is sometimes called precision-at-k (e.g. with observations at $k = 1, k = 5$, and $k = 10$ [61,132]). For off-line experiments, precision-at-k may be used to measure relevance for results users might actually examine. A variation frequently used in image retrieval is *mean average precision* [35]. Here, the precision from the first to each of returned results up to top k-th result is computed (for $\{(1), (1,2), \ldots, (1,2,\ldots,k)\}$) and then averaged, producing a bias for relevant results that have high rank. This set of precision values is averaged for the query, and the mean of these average precisions is computed over the query test set.

Systems are often compared visually by plotting precision against recall ('precision-recall' curves). More quantitative comparisons have been made using statistical hypothesis tests, or using AUC (area-under-the-curve) metrics for precision-recall plots. AUC metrics require interpolation for missing points [125]. Salton and McGill demonstrate using the Wilcoxon signed rank test to compare average precision for different recall value ranges ($\leq 0.1, \ldots, \leq 1.0$, see [125] Chapter 5.2.C), and determine whether the distributions are significantly different. The Wilcoxon test is non-parametric, making no assumption regarding the distribution of recall/precision values (e.g. they need not be normally distributed, as for a t-test).

To date published evaluations for MIR systems have been largely illustrative, and by example. One interesting approach compared retrieval using the Active-

[14] Text REtrieval Conference `http://trec.nist.gov/`

Math system [91] with retrieval from the ActiveMath web pages using the Google search engine, as well as a human-centered evaluation using a 'talk aloud' protocol, where participants are asked to speak their thoughts as they completed search tasks involving keywords and/or small expressions. Marinai et al. [96] provide precision-recall plots for their method for image-based math symbol retrieval using a bag-of-visual-words produced from clustered shape contexts [15]. Precision at 0% recall is presented, with precision values as high as 87% reported. Examination of the precision-recall curves shows a rapid decrease in precision before recall reaches approximately 20% (precision falls to roughly 20% in all conditions presented), but this likely includes many more elements than would be considered by a user. Their metrics were produced for almost 400 queries on a very large dataset of binary symbol images from document scans (from the Infty dataset [142]). Note that in this case determining relevance reduces to matching symbol labels in ground truth.

Yu and Zanibbi use a combination of off-line and user-centered evaluation for an image-based handwritten expression retrieval system [161, 167]. Participants were shown typeset expressions, which they drew using pen-and-paper. The pages were scanned to produce expression images for use in retrieval. For simplicity, only the region containing each test expression was identified in the ground truth. The system returned a ranked list of ten regions, each corresponding to the best match on an individual page. The observed measurements were (1) maximum ratio of overlap for the target region, and (2) whether the associated page appeared in the top k elements for $k = \{1, 5, 10\}$. These are essentially *recall*-at-k measures, but where a $specific$ expression is sought after. These metrics are conservative: no credit is given for anything other than one region on a single page. Search was run offline, and participants were brought back to evaluate the top-10 regions using a Likert scale (see Figure 10); participants were asked to evaluate the proportion of the query expression contained in each returned region. For comparison, the original query images were also used for retrieval, and performance evaluated on-line by each participant, and off-line. Retrieval of original images was much more effective than for handwritten queries; the average maximum ground truth region overlap was 43% for handwritten queries, but 90% for the original images. The corresponding human similarity evaluations were an average of 3.15/5 for the handwritten queries, and 4.83/5 for the original images.

Going forward, perhaps the most important direction in evaluating MIR systems is determining experimental protocols that can be easily replicated, and that reduce the need for manual identification of relevant documents or document regions, and perhaps creating a labeled test set similar to those developed for TREC. For MIR in general, relevance pertains to both text and expressions, making this a very time-intensive task, one that is sensitive to the expertise of the intended users. Once a reasonable method for defining or approximating relevance is determined, existing information retrieval metrics are likely sufficient.

4 Recognition of Mathematical Notation

Pattern recognition methods for mathematical notation may be used in a variety of contexts. Firstly, in Mathematical Information Retrieval, math recognition can be used to interpret user queries and to annotate document collections. An important open problem is to develop robust MIR methods that make effective use of recognition results even when recognition errors are present. Secondly, math recognition is used to support the insertion of expressions into documents; for example, entry of LaTeX expressions using images, pen, keyboard and mouse is illustrated in Figure 1. Thirdly, math recognition is used to recover layout and operator trees from images, handwritten strokes, or vector-based encodings (e.g. .pdf files). Finally, math recognition is used to integrate pen-based math entry into CAS systems (see Figure 2); in the future, expression images might also be used as input. This requires recognition of mathematical content, with the resulting operator tree used to support evaluation and manipulation of the expression.

Research on the recognition of math notation began in the 1960's [5, 6, 31, 98], and a number of surveys are available [19, 28, 52, 146]. In this paper we do not attempt to summarize the entire history as provided in these surveys, but rather provide an updated account of the state of the art, with an emphasis on advances since the well-known survey by Chan and Yeung [28] written a decade ago.

Many factors make the recognition of mathematical notation difficult. There may be noisy input in the case of images and strokes, and ambiguities arise even for noise-free input (see Figure 7). Math notation contains many small symbols (dots and diacritical marks) which can be difficult to distinguish from noise. Symbol segmentation can be difficult, particularly in handwritten mathematical notation. Symbol recognition is challenging due to the large character set (Roman letters, Greek letters, operator symbols) with a variety of typefaces (normal, bold, italic), and a range of font sizes (subscripts, superscripts, limit expressions). Several common symbols have ambiguity in their role; for

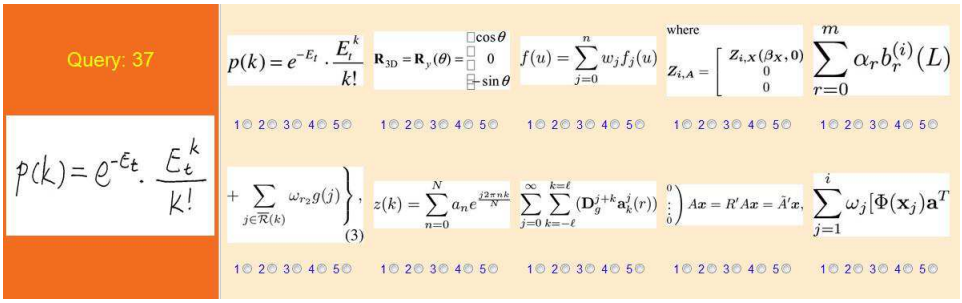


Fig. 10 *User Interface for Evaluating Image-Based Query-by-Expression using Handwritten Queries [161]. Each returned region is ranked on a 1-5 scale, with 1 indicating no match, 3 indicating roughly half the query is matched, and 5 indicating the query is contained completely within a returned region.*

example, a dot can represent a decimal point, a multiplication operator, a diacritical mark, or noise. Also, spatial relationships are difficult to identify; for example, it is difficult to distinguish between configurations that represent horizontal adjacency and those that represent superscripts or subscripts. The lack of redundancy in mathematical notation means that relatively little information is available for resolving ambiguities.

As shown in Figure 3, we identify four key problems that every math recognition system must address.

1. Expression detection
2. Symbol extraction or symbol recognition
3. Layout analysis
4. Mathematical content interpretation

These key problems are discussed in Sections 4.1 to 4.4. Most systems address these problems in sequence, but alternative control flow can be used to allow analysis at later stages to constrain or repair decisions made in earlier stages (Section 4.5), or to integrate and jointly optimize solutions to two or more of these problems simultaneously (Section 4.6).

4.1 Expression Detection

The input to a math recognition system can consist of vector graphics (such as PDF), pen strokes, or a document image. As discussed below, different challenges arise in detecting expressions in each of these input types, and there is an interaction between detecting symbols and expressions. For document images, some methods apply OCR or perform a coarse classification of connected components before segmenting expressions in documents, while others attempt to locate expressions using geometry or other methods. For pen-based entry systems, symbol segmentation and recognition is normally performed as the user writes, in part because it simplifies the system design, but also because it avoids requiring the user to check recognition results over a large set of objects and relationships.

4.1.1 Expression Detection in Vector Graphics

For vector graphics, work has begun on methods for extracting symbols and recognizing manually segmented expressions, but not on methods for automatic detection. Currently vector file formats such as PDF do not demarcate math regions. This is an important direction for future work, particularly for Mathematical Information Retrieval applications.

4.1.2 Expression Detection in Pen-based Input

For pen-based applications, expressions are often segmented using gestures [85, 144]. For example, the '⌋' gesture is used in the E-chalk system to indicate the end of an expression, and request its evaluation (see Figure 2(b)). Typically, a gesture gives a partial or approximate indication of the extent of an expression. Additional clustering or region growing methods can be applied, based on the properties of recognized symbols. Matrix elements can be detected using similar methods [89, 147].

4.1.3 Expression Detection in Document Images

In images, expressions are normally found using properties of connected components. Before discussing these methods, we distinguish between *displayed* expressions that are offset from text paragraphs and expressions that are *embedded* in text lines (Figure 11). Displayed expressions are easier to detect than embedded expressions, because text lines and displayed expressions tend to differ significantly in attributes such as height, separation, character sizes and symbol layout [52, 66].

Kacem et al. detect displayed expressions in images based on simple visual and layout features of adjacent connected components [66]. Embedded expressions are found by coarsely classifying connected components. Regions are grown around components that are identified as operators. The region growing is based on the

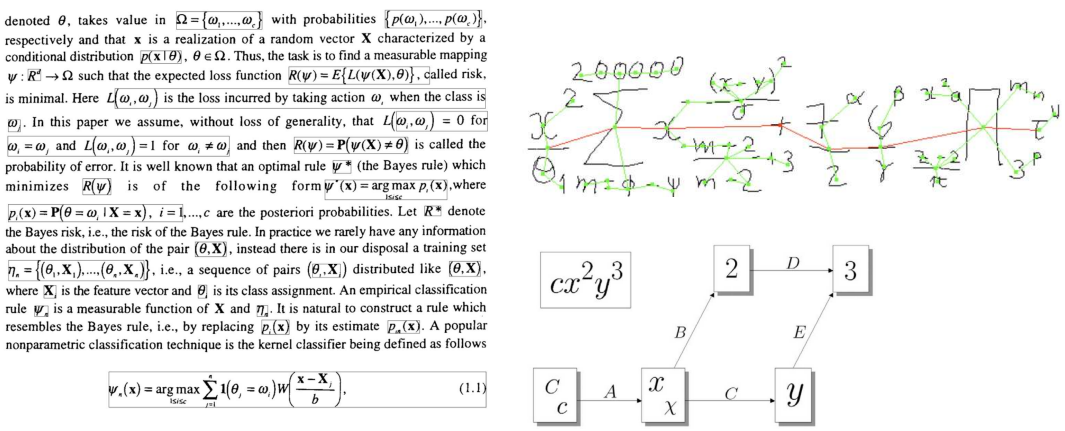


Fig. 11 *Expression Detection and Layout Analysis. At left, the document image contains a mix of expressions that are* displayed *(vertically offset) and expressions that are* embedded *in textlines (from [66]). Top right: a detected baseline (red) and minimum spanning tree used to associated non-baseline symbols with symbols on the baseline [144]. Bottom right: a virtual link network, in which a minimum spanning tree is constructed that minimizes costs based on symbol identity and spatial relationships [42].*

expected locations for operands (i.e. operator range and dominance).

An alternative approach for detecting embedded expressions first locates text lines, then computes symbol n-grams [52]. Training data provides frequencies for adjacent symbols, in textlines that are pure text, versus textlines that contain embedded expressions. A 97% recognition rate is reported for this technique. In subsequent work, Garain extends this approach by averaging over more general feature values for embedded and displayed expressions [49]. He obtained recall rates as high as 95% for embedded expressions, and 97% for displayed expressions.

Offset expressions can be detected without symbol classification. Drake and Baird use properties of the neighbor graph for connected components (a pruned Delaunay triangulation) to distinguish text lines from displayed expressions [39]. The reported accuracy for this method is high (over 99%), but it has not yet been used for embedded expressions.

4.2 Symbol Extraction or Symbol Recognition

OCR for math is a difficult problem, due to the large number of classes (see [94]), and problems caused by touching and over-segmented characters [27,52,99,135]. Berman and Fateman observed that commercial optical character recognition systems with recognition rates of 99% or higher fell to 10% or less once tried on perfectly formed characters in mathematical equations: heuristics that work well on straight text, multi-column printing and tables fail with math notation because of variations in font size, multiple baselines, special characters, and differing n-gram frequencies [16].

Techniques have improved since, and recognition rates as high as 97.7% have been reported for typeset symbols in the work of Malon, Uchida and Suzuki [94], where Support Vector Machines [154] are used to reduce common class confusions in the Infty OCR system [141] for 608 symbol classes.

Accuracies for online recognition of handwritten mathematical symbols have also been reported at rates of over 95%. In recent years there have been a number of methods based on Hidden Markov Models (HMMs [117]) that extend early work by Winkler [158] and Kosmala and Rigoll [80]. There is a general trend here, where HMMs were first used to perform simultaneous segmentation and recognition for a time series of pen strokes, but now later stages in processing, particularly layout and content information, are being incorporated into training and recognition stages. An open challenge is to adapt these methods to better handle 'late additions' to symbols, e.g. when a dot is added to the top of an 'i' after a large expression has been entered. Developments in HMM-based recognition methods are discussed further in Section 4.6.

Another group of successful methods employ features that approximate handwritten strokes via linear combinations of basis vectors or parametric curves. Various techniques for this have been used, including Principal Components Analysis [99] and polynomial basis functions [32, 54, 55]. These features allow recognition to be performed effectively within a small feature space (e.g. using the first fifteen principal components [99]), while allowing regeneration of the original data up to a chosen level of fidelity, making the interpretation of the features simple.

Voting-based methods for classifier combination have been employed to good effect. The method of Golubitsky and Watt [56] utilizes runoff elections in order to combine 1-against-1 SVM classifiers for a set of 280 symbols (280*279/2 = 39,060 classifiers in total). Majority voting is used first, followed by a runoff election where only votes for the top N classes are considered to break ties. LaViola and Zelenik applied AdaBoost [45] to another all-pairs classifier ensemble, with a binary classifier for every pair of classes. Each base classifier uses only a single feature; most are measured on strokes, but output from the Microsoft handwriting recognizer is included as a feature [86]. This work was concerned with adapting a writer-independent classifier (the Microsoft classifier) to the handwriting of specific individuals through stroke-based features.

4.3 Symbol Layout Analysis

Visual syntax refers to the layout and topology of symbols. A variety of formats can be used to represent visual syntax, the essence of which may be represented by a symbol layout tree (see Figure 4).

A number of techniques have been used to recover symbol layout. The first three approaches discussed below use recursive decomposition, based on operator dominance, on cutting pixel projection profiles, and on identification of symbols on the dominant baseline. Following that, we discuss approaches based on penalty graph minimization.

Operator-driven decomposition recursively decomposes a math expression by using operator dominance to recursively identify an operator which has most or all of the remaining symbols as its operands [31]. These symbols are partitioned into the expected operand locations [29, 31]. Unlike the other approaches described in this section, operator-driven decomposition constructs an operator tree (Figure 4b) directly from the symbol layout, rather than first producing a symbol layout tree. The earliest example of a simple pen-based math calculator made use of this method [30]. Lee and Wang [88] use a similar approach to recover symbol layout, using operator dominance to group symbols vertically, followed by determining horizontal adjacencies between symbols.

Projection profile cutting recursively decomposes a typeset math expression using a method similar to X-Y cutting [108]. Pixel intensity histograms in the vertical and horizontal directions are computed, followed by splitting at gaps identified in the histograms [111, 112, 153]. The first cut is made in the vertical direction (roughly speaking, to separate horizontally adjacent subexpressions), after which the direction for cutting alternates. An improvement was suggested by Raja et al., in which connected components are first extracted, and then regions containing more than one connected component that cannot be decomposed during cutting (e.g. for square roots or kerned characters) have the largest connected component removed, continuing cutting with the remaining connected components [118]. In related X-Y cutting methods, thresholds for cutting have been chosen using the estimated dominant character height and width for a page (using the mode of run lengths in horizontal and vertical projections at the page level), and then scaling these thresholds linearly based on the size of the area to be cut relative to the entire page [128].

Baseline extraction decomposes a math expression by recursively identifying adjacent symbols from left-to-right on the main baseline of an expression, and then partitioning remaining symbols into regions relative to the baseline symbols [162,163]. Operator dominance information is used so that symbols need not be precisely aligned in some cases (e.g. for a symbol following a binary operators such as +). Baseline extraction has been used in a number of pen-based math entry systems [7, 116, 133, 145, 147], though the technique may be used for symbols taken from document images as well. Some work has been carried out into using more sophisticated symbol layout models (e.g. using multiple points on the bounding box in determining spatial relationships [145]), as well as using a minimum spanning tree for the symbol partitioning step [145], as shown in Figure 11. To handle ambiguous spatial relationships, fuzzy methods have been used to produce multiple interpretations [170].

Penalty graph minimization is a more global approach to layout recognition, in which candidate relationships between symbols are defined before minimizing a penalty criterion. Eto, Suzuki et al. make use of Virtual Link Networks to represent penalties for candidate symbol identities and spatial relationships (see Figure 11), and then compute the minimum-spanning tree of the graph to produce a final interpretation [42]. Spatial relationships in the networks are binary (between symbol pairs), and of five types: above, below, inline, superscript, subscript. Candidate spatial relationships and penalties are defined based on symbol bounding boxes (normalized relative to the estimated font height and writing line location), and box center points [4,42]. Discrimination of spatial relationships may be improved through document-specific adaptation for determining ascender/descender/center regions on writing lines. A recognition rate of 99.57% is reported for a test on valid adjacent symbol-pair rela-

tionships for the Infty dataset (158,308 adjacent symbol pairs, taken from the ground truth).

Matrix layout requires special processing. The following approaches have been reported. The virtual link network method was extended to use projections of symbols inside a matrix, and then solve a resulting linear system of equations to estimate row and column positions [69, 70]. Other authors have performed segmentation of matrix elements using simpler projections of symbol bounding boxes [145] or region growing [88, 147] before analyzing elements using a single-expression technique. Recently there has been work to allow matrices containing ellipses to be used within pen-based systems integrated with computer algebra systems [89,127,147]. In handwritten expressions, matrices can be processed by detecting left fence symbols, followed by clustering and projection analyses [150, 151].

At this point, no one technique for layout analysis completely dominates another, and improving these methods is an active area of research. It may be worth exploring methods for combining layout analyzers, in a manner similar to combination methods used for classification.

4.4 Mathematical Content Interpretation

Many math recognition systems do not perform analysis beyond symbol layout, and such systems do not construct a representation of the mathematical meaning of the expression. For systems designed to evaluate expressions and/or integrate with Computer Algebra Systems however, a representation of the logical relationships between symbols, and a representation of domain semantics is necessary. Various encodings can be used to represent the hierarchy of operators, relations and operands, which are generally equivalent to some form of operator tree (Figure 4b). Generally the definitions for operators and relations are assumed for a given math dialect in recognition systems, although content dictionaries such as those provided by OpenMath [37] might be used to encode and lookup the operations associated with symbols.

Recovering an operator tree from symbol locations may be understood as accepting sentences from a formal visual language [97], using a parser to analyze symbol layout in order to produce an operator tree. The earliest approach to recognizing symbol layout, by Anderson, is of this type: an operator tree is constructed top-down, and then a string representing the tree structure is synthesized bottom-up [5]. A number of different attributed grammar types have been used, including context-free string grammars [43] and graph grammars [58, 87, 137].

Grammar-based methods commonly represent symbol locations by geometric objects such as bounding boxes or convex hulls. The placement of symbol centroids reflects the presence of ascenders (h) and descenders (y). Predicates and actions associated with grammar productions make use of the bounding boxes and centroids to determine spatial relationships. It should be noted that grammars are a very general formalism, and variations of layout analysis techniques seen in the previous section have been employed within the production rules of grammars designed to recover the operator tree of an expression. Examples included syntactic recognition using operator-driven decomposition [5], and baseline extraction [14]. A key issue is the geometric model used to partition the input and define primitives. For example, using unrestricted subsets of image pixels as primitives is far too computationally intensive. Instead, primitive regions are represented using geometric objects such as axis-aligned rectangles, along with constraints on allowable orderings and adjacencies between regions. Liang et al. provide a helpful overview, including examples from math recognition [90]. Different parsing algorithms explore the space of legal expressions in different orders, some more efficiently than others.

Stochastic context-free grammars allow uncertainty in symbol recognition, layout and/or content to be accommodated, by returning the maximum-likelihood derivation for the input image [34] or symbols [103]. These methods are discussed further in Section 4.6. Some more recent parsing methods that model uncertainty include fuzzy-logic based parsing [44,53], and A*-penalty-based search [122].

As discussed previously, usage of notation differs significantly in different dialects of mathematical notation, and so the space of operator trees and corresponding grammar productions need to be adapted for different mathematical domains of discourse. The notion of devising one grammar to cover all of mathematical notation seems quite impractical, though defining grammars with some utility for a specific domain (e.g. matrix algebra) is possible.

Methods that permit recognition to be defined at the level of a grammar are very appealing, in that with suitable implementations for pattern recognition methods being available, a language definition may be sufficient for recognizing a dialect of mathematical notation, including layout and mathematical content. However, it has been observed that the tight coupling between the assumed recognition model and grammar formalism can make it difficult to adapt syntactic pattern recognition methods. One compromise is to use a modular organization similar to a compiler, where recognized sym-

bols are combined into tokens and have their layout analyzed, after which an operator tree is constructed through restructuring and annotating the symbol layout tree [18, 163]. More advanced techniques might interleave and/or iterate these stages.

4.5 Post-processing: Constraining Outputs

Pattern-recognition systems commonly use post-processing to correct preliminary recognition results. Many post-processing operations apply contextual constraints to results for individual objects and relationships identified largely in isolation of one another [149]. In document recognition, perhaps the most well-known example of post-processing is the use of dictionaries and n-grams to refine preliminary OCR results obtained for individual characters [107, 115].

Ten years ago, the last IJDAR survey on math recognition [28] identified post-processing as an important direction for future research. Indeed, significant advances for post-processing of math recognition have been made in the last ten years. Several methods are similar to dictionary and n-gram methods used for OCR. Others incorporate syntactic constraints on two-dimensional symbol layout or expression syntax; these methods work with symbol layout trees and operator trees respectively.

4.5.1 Statistical Analysis of Math Notation

Statistical information about math notation is useful in post-processing. The frequency estimates described below have been used to re-rank and constrain preliminary symbol recognition results for handwritten math entry [134]. In addition, they have been used to categorize mathematical documents by Math Subject Classification categories [155]; so far, this appears to be the only paper published on this interesting problem. Also, recognition systems can use information about symbol frequencies and expression frequencies as prior probability estimates.

So and Watt [138] conducted an empirical study of over 19,000 papers stored in the ArXiv e-Print Archive. This archive at http://arxiv.org provides electronic versions and LaTeX source of papers from scientific, mathematical and computing disciplines. So and Watt's study determined the frequencies for expression usage in different mathematical domains, as identified by the Mathematical Subject Classification described in Section 3.1. Documents were categorized using the top-level Mathematical Subject Classification provided by the ArXiv. Analyses were made at the symbol layout level after converting the available LaTeX to Presentation MathML.

The statistics produced by So and Watt make a distinction between identifier symbols and operator symbols. In both cases, but especially for operator symbols, plotting symbols by decreasing frequency shows an exponential decrease in frequency with rank; this is similar to the Zipf distribution [173] seen for word frequencies. Similarly, expressions become significantly less frequent as they become larger and more structurally complex. Interestingly, the number of *distinct* expressions increases with expression size and complexity.

In a later study, Watt focused on engineering mathematics, analyzing the LaTeX sources for three engineering mathematics textbooks [155]. In this study, all symbols were analyzed together, producing another Zipf distribution. N-grams (for $n \in \{2, 3, 4, 5\}$) were produced by traversing the symbol layout tree in writing order. The leaves of the tree, which store the symbols, provide the starting point. The traversal collects layout information to provide context: there is information about the spatial relationship between the n-gram symbols and symbols on neighboring baselines (e.g. fractions, super/subscript, containment by square root).

4.5.2 Heuristic Rules and Contextual Constraints

Heuristic rules and manually constructed language models are receiving use in post-processing. Chan and Yeung [29] describe an error-correcting parsing technique for converting handwritten symbols into operator trees, adding heuristic rules to re-segment characters recognized with low confidence, to insert *epsilon* (empty) symbols to recover from parse errors (e.g. after detecting unbalanced parentheses), and to replace symbol identities to make them consistent with the expression grammar (e.g. replacing '1' by '/' in 'y 1 x', and '+' by 't' in '+an'). Garain and Chaudhuri make use of a simple LaTeX grammar to constrain handwritten symbol recognition alternatives [50], while Kanahori et al. present work in analyzing the mathematical content (operator tree) for matrices in order to revise symbol layout analysis [68]. A more recent technique by Fujiyoshi et al. [47, 48], similar to that of Chan and Yeung, defines a grammar for valid symbol layout trees and then parses initial recognition results in order to identify invalid structures. During parsing, syntax errors are visualized so that users may identify the specific symbols associated with parse errors (e.g. unbalanced fence symbols).

Contextual constraints can also be incorporated into the recognition process itself. For example, Kim et al. [73] modify the penalty metric used in an A* search for constructing symbol layout trees for handwritten expressions [122]. The penalty metric considers mea-

sures of *consistency* of symbol size, style, and repetition, along with symbol n-grams and repeated subscripting.

4.6 Integration of Recognition Modules

Integration of recognition modules has been an important new area of development in the last ten years. Most approaches involve some form of dynamic programming. The earliest work in this area is Chou's influential paper describing the use of stochastic context-free string grammars for analysis of typeset images of mathematical notation [34]. This approach combines segmentation, recognition, and layout analysis, and is highly tolerant of bit-flip noise. Subsequent work includes extensions by Hull [65], and extension to a more general HMM-based model for document image decoding [79].

Stochastic context-free grammars associate a probability with each derivation rule; the derivation rules associated with each nonterminal have probabilities that sum to one. The probability of a derivation is computed as the product of the probabilities of all rule applications used to derive the input string. Rule probabilities can be estimated by the author of the grammar, or they can be derived from a training corpus using the Inside-Outside algorithm [34]. To facilitate the use of parsing through dynamic programming, stochastic context-free grammars are often represented in Chomsky-Normal Form: all rules are of the form $A \rightarrow BC$ or $A \rightarrow t$. A modified form of the Cocke-Younger-Kasami (CYK) parsing algorithm uses dynamic programming to produce the maximum likelihood parse in $O(n^3)$ time, where n is the number of input tokens.

In Chou's paper [34], the expression grammar is augmented to include symbols representing horizontal and vertical concatenation of adjacent regions in the input image. In a 'lexical' stage that precedes parsing, a template-based character recognizer is applied to the entire input region, identifying a set of candidate symbols based on the Hamming distance between input regions and a set of templates. This produce a set of candidate symbols with associated probabilities. More recently Yamamoto et al. [159] used a stochastic context-free grammar for online handwritten expressions, which introduces rules to model the likelihood of written strokes along with rules incorporating probabilities for the expected relative positions of symbols (the authors term these *hidden writing areas*).

There are many unexplored possibilities for using stochastic context free grammars for math recognition. For example, a variety of segmentation and classification methods might be employed within a framework of stochastic context free grammars. Also, various heuristics could be used to prune or modify rules that are inferred from training data. It is true that sequential implementations of stochastic context free grammars are computationally intensive, but both probability-estimation algorithms and parsers may be parallelized [34]. Many opportunities for parallelization exist in modern CPUs with multiple cores and Graphical Processing Units.

The related technique of Hidden Markov Models (automata that recognize probabilistic *regular* languages) has been used to integrate segmentation and classification of handwritten symbols [80, 158] (analogous to speech recognition [117]). For stochastic regular languages, the CYK algorithm reduces to the Viterbi algorithm, which may be used to determine the maximum likelihood path (parse) through a Hidden Markov Model [34]. Hidden Markov Models form the core of a general model of document image decoding, in which the document-generation process is explicitly modeled as part of the recognition system [79].

More recently, dynamic programming methods have been used to let later stages of processing constrain earlier ones in an optimization framework. For example, Toyozumi et al. address segmentation of handwritten symbols drawn online [152]. They produce improvements on the order of 5-7% over a feature-based elastic matching method by using simple, local grammatical rules to consider neighboring strokes and possible under-segmentation of vertical operators such as fractions, square roots and summations. Shi, Li and Soong go further, using a dynamic programming framework to optimize symbol segmentation and recognition [130]. Their system considers a sequence of strokes from online handwritten input. The space of all possible partitions of the stroke sequence into symbols (containing at most L strokes per symbol) is searched to find an optimal partition through dynamic programming. The criterion function that is used to evaluate a given stroke partition uses two components: (1) a bigram model for symbol adjacencies along particular spatial relationships, and (2) the probability of the sequence of spatial relationships observed between symbols. As a post-processing step, a trigram symbol sequence model is evaluated for re-ranking alternatives. On a test set of over 2,500 expressions, a symbol accuracy of 96.6% is reported. An extension employing graph-based discriminative training is reported by Shi and Soong [131], with similar results. A method integrating complete symbol layout trees into the dynamic programming is described in Awal et al. [11].

4.7 Evaluation of Math Recognition Systems

At present, meaningfully comparing evaluations of math recognition systems is challenging [12, 83]. This is in large part because different systems tend to focus on different mathematical domains, layout conventions, and stages of the recognition process illustrated in Figure 3 (detection, symbol recognition/extraction, layout analysis, and interpreting mathematical content). To properly interpret results, performance metrics need to be supplemented by a characterization of the scope of the systems, to support informed comparison of high-accuracy narrow-scope systems versus systems that process a broad range of inputs with lower accuracy.

We discuss the use of benchmark data below, which is commonly used to address these issues, albeit in a way that inevitably leads to debates about representativeness of the data, and/or the relevance of the data for particular applications. Even in the presence of benchmark data, quantitative means for characterizing the scope of mathematical notation handled by systems is an important area for future research. It is particularly difficult to quantify the amount of noise and distortion that a system can handle; perhaps benchmark data can be modified using document degradation models for this purpose [72], analyzing results over a space of degradation parameter settings (e.g. increasing skew in handwritten expressions, or blurring in images).

The most common class of performance metrics for evaluation of math recognition systems are recognition rates, for complete expressions [29,110,163] and individual symbols [8, 29, 110, 143]. Characterizations of layout structure accuracy have been measured using a variety of metrics; most simply, the number of symbols with the appropriate parent symbol, relationship, and depth in a symbol layout tree ('token placement'), and the number of baselines that contain the correct symbols [163]. Other metrics provide recall measures for layout structures in a symbol layout tree (e.g. scripting, fractions, limits, roots, and matrices [29, 110]).

One can devise metrics that combine symbol and layout-level error metrics, which may serve as criterion functions for machine learning algorithms (to optimize a complete system). Chan and Yeung [29] propose a 'global' recall metric, the number of correctly recognized symbols *and* structures (subtrees) in an *operator tree*, divided by the number of symbols and structures. Garain and Chaudhuri proposed a related recall measure for symbol layout trees, where recall for symbol classes and placement (i.e. symbols with the correct parent symbol and relationship in the symbol layout tree) is computed, but *weighting* misplacement errors by the depth of nesting for a symbol in ground truth [51]. String edit distances are used to compare symbol layout trees for recognition results and ground truth, after the trees are linearized into Euler strings [124]. This was proposed to overcome the NP-completeness of computing a full tree edit distance between layout trees.

Recently it was proposed that a bipartite graph could be used to capture segmentation, classification, and layout errors simultaneously [166]. The graph represents all N primitives in one node set, and the classification labels assigned to each primitive in the second node set (each primitive receives the label of its associated symbol). $N(N-1)$ spatial relationships are defined between the unlabeled (parent) and labeled (child) primitives. Given a symbol layout tree, spatial relationships are inherited and represented explicitly in the bipartite graph; for example, in x^{2_a}, the symbol a is in a subscript relationship with 2, but also a superscript relationship with x. One can then compute recall for primitive labels and spatial relationships in the graph. Correcting these labels induces the correct classification, segmentation, and layout for all input primitives (e.g. connected subcomponents, or strokes). This representation provides a meaningful, intuitive representation for an expressions' elements and their interpretation at the layout level. The bipartite representation can be generalized in a straight-forward manner to operator trees as well.

4.7.1 Data Sets for Math Recognition Evaluation

Just as in the TREC competitions for information retrieval (see Section 3.5), in pattern recognition and machine learning research, benchmarking data is used to make meaningful system comparisons, in a fixed domain whose scope of interpretation is defined by examples in the data set. The ambiguities that arise from human decisions about the *relevance* of retrieval results are replaced by ambiguities arising from human decisions about how to *interpret* the location, symbols, layout and mathematical content of expressions. In both cases, algorithms are evaluated by their ability to imitate those defining ground truth [164]. Ground-truth data is expensive to create, because it requires laborious human effort; a semi-automated ground truth creation technique for handwritten expressions is described in MacLean et al. [93]. Similar to the normalizations used in retrieval, care needs to be taken to normalize ground truth and recognizer outputs, so that equivalent expressions match properly during evaluation.

Currently there is some limited use of available benchmark datasets, but we expect their use to increase significantly as research in this area intensifies. The following is a list of benchmark data sets that have been

reported in the literature, some of which are publicly available.

Infty I-III[15] [142]: Infty-1 provides around 500 pages from English technical articles on pure mathematics containing over 20,000 typeset expressions. Ground truth was created manually and provides symbol bounding boxes, identities, and edges of the symbol layout tree in .csv, XML, and MathML. Infty-II adds documents from English, French and German publications. Infty-III provides over 250,000 single alphanumeric characters and mathematical symbols.

UW-III[16] [114]: mathematical content consists of 25 pages, with approximately 100 typeset equations. Ground truth creation involved double entry and triple verification. Math expressions are represented in ground truth as LaTeX and labeled bounding boxes for expressions and symbols (in Xfig format).

Waterloo/MathBrush[17] [93]: handwritten expressions by 20 writers (4655 expressions total). Ground truth provides operator trees, LaTeX, .gif (for typeset target), Microsoft and SCG ink formats.

MNIST[18]: 70,000 segmented, size-normalized (28x28) greyscale handwritten digit images (60k train, 10k test). Ground truth provides symbol identities.

Brown Dataset[19] [86]: 48 handwritten symbols from 11 writers (10 train, 12 test instances per class) Ground truth: Stroke data in Unipen format

Chan and Yeung [29] 600 handwritten expressions (11,190 symbols), written by 10 different writers, and drawn from CRC Standard Mathematical Tables and Formulae [174].

Ashida et al. [8] 1400 pages for symbol recognition data (43,495 typeset expressions), 700 pages for structure analysis (21,472 typeset expressions), taken from *Archiv der Mathematik* and *Commentarii Mathematici Helvetici*. Ground truth was created using automatic recognition followed by manual correction. Ground truth encodes bounding boxes and labels for expressions and symbols, and expression structure in an extended MathML format.

Garain and Chaudhuri [51]: 400 pages (297 real data and 103 synthetic data) containing 5,560 typeset expressions. Ground truth creation used automatic recognition followed by manual correction. Ground truth consists of LaTeX and symbol bounding boxes for isolated expressions, as well as extended MathML for document pages.

ICDAR 2011[20] data provided for the online handwritten math recognition contest at the International Conference on Document Recognition and Retrieval in 2011 (over 1000 handwritten expressions from multiple writers).

5 Conclusion

Recognition and retrieval of mathematical notation are challenging, interrelated research areas of great practical importance. In math retrieval, the key problems are defining query languages, normalizing the query and searchable documents, defining methods of indexing and matching, and providing relevance feedback. In math recognition, the key problems are detecting expressions, detecting and classifying symbols, analyzing symbol layout, and constructing a representation of meaning. Math notation provides an excellent domain for studying issues that also arise in recognition and retrieval of other types of graphical notations.

We conclude our paper by outlining expected developments and numerous opportunities for future research in this area. In general terms, we predict that future research will enhance the ability of recognition and retrieval systems to process a broad scope of notations and dialects, to exhibit robustness to noise, and to provide flexible, effective user interfaces. We summarize open problems and future directions in five categories: query interfaces, indexing and retrieval, relevance feedback, performance evaluation, and math recognition.

Future directions in query interfaces include image-based math retrieval (allowing expression images to be used as queries) and sketch-based math retrieval (allowing online handwritten expressions to be used as queries). We predict that sketch-based retrieval will make prominent use of finger-based rather than stylus-based drawing, due to the convenience and wide-spread use of tablets and touch interfaces. Flexible query interfaces will combine text, images, sketching, keyboard and mouse. Improved interfaces will be developed to allow a user to specify matching constraints; for example, hard constraints could be indicated by a box surrounding strokes and/or connected components.

Future directions in indexing and retrieval include improved methods for normalization of queries and documents; flexible normalization approaches will be able to adapt to the nature of the query and document data, whether it be handwritten, vector graphics or images. Indexing and retrieval will include pattern recognition methods to locate, recognize and annotate mathematical expressions in typeset and handwritten document corpora. The strengths and weaknesses of document representations will be explored, determining when vector-based, tree-based or combined models are most appropriate.

Relevance feedback is an important but as-yet unaddressed research opportunity for math retrieval. We expect that there will be improvement in the interfaces and mechanisms used, and in algorithms for defining refined queries from user interactions. Machine learning methods may play an important role in improving relevance feedback.

Future directions in performance evaluation will include advances in the technology for creating databases

[15] www.inftyproject.org/en/database.html

[16] www.science.uva.nl/research/dlia/datasets/uwash3.html

[17] www.scg.uwaterloo.ca/mathbrush/corpus

[18] http://yann.lecun.com/exdb/mnist

[19] http://graphics.cs.brown.edu/research/pcc/symbolRecognitionDataset.zip

[20] http://www.isical.ac.in/∼crohme2011/

with ground truth, and increased availability of datasets for math recognition and retrieval. There will be advances in performance metrics for computing errors in layout, segmentation, parsing, classification, and representation of meaning. Performance evaluation needs to be carried out in reference to tasks a user is trying to accomplish. Research is needed to obtain a better understanding of different models of relevance for mathematical information retrieval. Relevance depends on a number of factors, including the expertise of the user, the task underlying the user's information need, and the type of resource(s) sought.

In math recognition, future directions and open problems include the detection of inline expressions, the automatic detection of mathematics in vector graphics documents, and the processing of matrix and tabular structures. We predict refinements of layout analysis, including development of new techniques and combination of existing methods via parser combination. More sophisticated language models will be developed to incorporate statistical information about mathematical notation; this information can be used during recognition or post-processing. Stochastic language models will be become increasingly sophisticated; stochastic grammars, as initially proposed by Chou [34] can be extended using different segmentation and/or parsing approaches. A challenge is to identify usable notation sets with invariants that can be easily adapted to dialects; the goal is to scale this up to the index set used by the Mathematical Subject Classification (MSC) [121].

In conclusion, the combination of math retrieval and math recognition technologies provides rich possibilities for math-aware computer interfaces, and for intelligent search and retrieval tools for math in documents.

Acknowledgements This material is based upon work supported by the National Science Foundation under Grant No. IIS-1016815. This work was also supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Xerox Foundation. We wish to thank George Nagy for helpful discussions, and the anonymous reviewers for comments that improved the paper significantly.

References

1. M. Adeel, H.S. Cheung, and H.S. Khiyal. Math go! Prototype of a content based mathematical formula search engine. *J. Theoretical and Applied Information Technology*, 4(10):1002–1012, 2008.
2. A.V. Aho, B.W. Kernighan, and P.J. Weinberger. *The AWK Programming Language*. Addison-Wesley, New York, 1988.
3. M. Altamimi and A.S. Youssef. An extensive math query language. In *ISCA Int'l Conf. Software Engineering and Data Engineering*, pages 57–63, Las Vegas, USA, 2007.
4. W. Aly, S. Uchida, and M. Suzuki. Identifying subscripts and superscripts in mathematical documents. *Mathematics in Computer Science*, 2(2):195–209, 2008.
5. R.H. Anderson. *Syntax-Directed Recognition of Hand-Printed Two-Dimensional Equations*. PhD thesis, Harvard University, Cambridge, MA, 1968.
6. R.H. Anderson. Two-dimensional mathematical notation. In K.S. Fu, editor, *Syntactic Pattern Recognition, Applications*, pages 174–177. Springer, New York, 1977.
7. L. Anthony, J. Yang, and K.R. Koedinger. Adapting handwriting recognition for applications in algebra learning. In *Proc. ACM Work. Educational Multimedia and Multimedia Education*, pages 47–56, Augsburg, Germany, 2007.
8. K. Ashida, M. Okamoto, H. Imai, and T. Nakatsuka. Performance evaluation of a mathematical formula recognition system with a large scale of printed formula images. In *Proc. Int'l Conf. Document Image Analysis for Libraries*, pages 320–331, Lyon, France, 2006.
9. A. Asperti, F. Guidi, C. Coen, E. Tassi, and S. Zacchiroli. A content based mathematical search engine: Whelp. In *Proc. Types for Proofs and Programs 2004*, volume 3839 of *LNCS*, pages 17–32. Springer, 2006.
10. R. Ausbrooks, S. Buswell, D. Carlisle, G. Chavchanidze, S. Dalmas, S. Devitt, A. Diaz, S. Dooley, , R. Hunter, P. Ion, M. Kohlhase, A. Lazrek, P. Libbrecht, B. Miller, R. Miner, C. Rowley, M. Saregent, B. Smith, N. Soiffer, R. Sutor, and S. Watt. Mathematical markup language (MathML) version 3.0, W3C recommendation (http://www.w3.org/math/), 2010.
11. A.M. Awal, H. Mouchère, and C. Viard-Gaudin. Towards handwritten mathematical expression recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1046–1050, Barcelona, Spain, 2009.
12. A.M. Awal, H. Mouchére, and C. Viard-Gaudin. The problem of handwritten mathematical expression recognition evaluation. In *Proc. Int'l Conf. Frontiers in Handwriting Recognition*, pages 646–651, Montréal, Canada, 2010.
13. J.B. Baker, A.P. Sexton, and V. Sorge. A linear grammar approach to mathematical formula recognition from PDF. In *Proc. Mathematical Knowledge Management*, volume 5625 of *LNAI*, pages 201–216. Springer, 2009.
14. J.B. Baker, A.P. Sexton, and V. Sorge. Faithful mathematical formula recognition from PDF documents. In *Proc. Int'l Work. on Document Analysis Systems*, pages 485–492, Boston, USA, 2010.
15. S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.
16. B.P. Berman and R.J. Fateman. Optical character recognition for typeset mathematics. In *Proc. Int'l Syposium on Symbolic and Algebraic Computation*, pages 348–353, Oxford, UK, 1994.
17. M.I. Bernstein. Computer input/output of two-dimensional notations. In *Proc. Symp. on Symbolic and Algebraic Manipulation*, pages 102–103, 1971.
18. D. Blostein, J. Cordy, and R. Zanibbi. Applying compiler techniques to diagram recognition. In *Proc. Int'l Conf. Pattern Recognition*, volume 3, pages 123–126, 2002.
19. D. Blostein and A. Grbavec. Recognition of mathematical notation. In *Handbook of Character Recognition and Document Image Analysis*, pages 557–582. World Scientific, 1997.
20. D. Blostein, E. Lank, A. Rose, and R. Zanibbi. User interfaces for on-line diagram recognition. In *Selected Papers from the Fourth Int'l Work. Graphics Recognition Algorithms and Applications*, volume 2390 of *LNCS*, pages 92–103. Springer, 2002.

21. D. Blostein, E. Lank, and R. Zanibbi. Treatment of diagrams in document image analysis. In *Proc. Int'l Conf. on Theory and Application of Diagrams*, pages 330–344, London, UK, 2000. Springer.
22. P. Borlund. User-centered evaluation of information retrieval systems. In *Information Retrieval: Searching in the 21st Century*, pages 21–37. Wiley, 2009.
23. A. Bunt, M. Terry, and E. Lank. Friend or foe? Examining CAS use in mathematics research. In *Proc. Int'l Conf. Human Factors in Computing Systems*, pages 229–238, New York, 2009.
24. F. Cajori. *A History of Mathematical Notations (2 vols.)*. Open Court Publishing Company, Chicago, Illinois, 1929.
25. J. Carette and W.M. Farmer. A review of mathematical knowledge management. In *Proc. Mathematical Knowledge Management*, volume 5625 of *LNAI*, pages 233–246. Springer, 2009.
26. D.O. Case. *Looking for Information: A Survey of Research on Information Seeking, Needs, and Behavior*. Academic Press, 2002.
27. R.G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(7):690–706, 1996.
28. K.-F. Chan and D.-Y. Yeung. Mathematical expression recognition: A survey. *Int'l J. Document Analysis and Recognition*, 3:3–15, 2000.
29. K.-F. Chan and D.-Y. Yeung. Error detection, error correction and performance evaluation in on-line mathematical expression recognition. *Pattern Recognition*, 34(8):1671–1684, 2001.
30. K.-F. Chan and D.-Y. Yeung. Pencalc: A novel application of on-line mathematical expression recognition technology. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 774–778, Seattle, USA, 2001.
31. S.-K. Chang. A method for the structural analysis of two-dimensional mathematical expressions. *Information Sciences*, 2:253–272, 1970.
32. B.W. Char and S.M. Watt. Representing and characterizing handwritten mathematical symbols through succinct functional approximation. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1198–1202, Curitiba, Brazil, 2007.
33. T.W. Chaundy, P.R. Barrett, and Charles Batey. *The Printing of Mathematics*. Oxford University Press, London, 1957.
34. P.A. Chou. Recognition of equations using a two-dimensional stochastic context-free grammar. In *Proc. Visual Communications and Image Processing IV*, volume 1199 of *Proc. SPIE*, pages 852–863, 1989.
35. R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, 2008.
36. J.H. Davenport and M. Kohlhase. Unifying math ontologies: A tale of two standards. In *Intelligent Computer Mathematics*, volume 5625 of *LNAI*, pages 263–278. Springer, 2009.
37. M. Dewar. Openmath: An overview. *ACM SIGSAM Bulletin*, 34:2–5, 2000.
38. D. Doermann. The indexing and retrieval of document images: A survey. *J. Computer Vision and Image Understanding*, 70:287–298, 1998.
39. D.M. Drake and H.S. Baird. Distinguishing mathematics notation from english text using computational geometry. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1270–1274, Seoul, Korea, 2005.
40. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Springer-Verlag, New York, 2nd edition, 2001.
41. T.H. Einwohner and R.J. Fateman. Searching techniques for integral tables. In *Proc. Int'l Symp. on Symbolic and Algebraic Computation*, pages 133–139, Montréal, Canada, 1995.
42. Y. Eto and M. Suzuki. Mathematical formula recognition using virtual link network. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 430–437, Seattle, USA, 2001.
43. R.J. Fateman and T. Tokuyasu. Progress in recognizing typeset mathematics. In *Proc. SPIE*, volume 2660, pages 37–50, 1996.
44. J.A. Fitzgerald, F. Geiselbrechtinger, and T. Kechadi. Mathpad: A fuzzy logic-based recognition system for handwritten mathematics. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 694–698, Curitiba, Brazil, 2007.
45. Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Computer and System Sciences*, 55(1):119–139, 1995.
46. M. Fujimoto, T. Kanahori, and M. Suzuki. Infty editor - a mathematics typesetting tool with a handwriting interface and a graphical front-end to OpenXM servers. In *Computer Algebra - Algorithms, Implementations and Applications*, volume 1335 of *RIMS Kokyuroku*, pages 217–226, 2003.
47. A. Fujiyoshi, M. Suzuki, and S. Uchida. Verification of mathematical formulae based on a combination of context-free grammar and tree grammar. In *Proc. Int'l Conf. Mathematical Knowledge Management*, volume 5144 of *LNCS*, pages 415–429. Springer, 2008.
48. A. Fujiyoshi, M. Suzuki, and S. Uchida. Syntactic detection and correction of misrecognitions in mathematical OCR. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1360–1364, Barcelona, Spain, 2009.
49. U. Garain. Identification of mathematical expressions in document images. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1340–1344, Barcelona, Spain, 2009.
50. U. Garain and B.B. Chaudhuri. Recognition of online handwritten mathematical expressions. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 34(6):2366–2376, 2004.
51. U. Garain and B.B. Chaudhuri. A corpus for OCR research on mathematical expressions. *Int'l J. Document Analysis and Recognition*, 7(4):241–259, 2005.
52. U. Garain and B.B. Chaudhuri. OCR of printed mathematical expressions. In *Digital Document Processing*, pages 235–259. Springer, 2007.
53. R. Genoe, J.A. Fitzgerald, and T. Kechadi. An online fuzzy approach to the structural analysis of handwritten mathematical expressions. In *Proc. Int'l Conf. Fuzzy Systems*, pages 242–250, Vancouver, 2006.
54. O. Golubitsky and S.M. Watt. Online computation of similarity between handwritten characters. In *Proc. Document Recognition and Retrieval*, volume 7247 of *Proc. SPIE*, pages C1–C10, San Jose, USA, 2009.
55. O. Golubitsky and S.M. Watt. Distance-based classification of handwritten symbols. *Int'l J. Document Analysis and Recognition*, 13(2):133–146, 2010.
56. O. Golubitsky and S.M Watt. Improved classification through runoff elections. In *Proc. Work. Document Analysis Systems*, pages 59–64, Boston, USA, 2010.
57. P. Graf. Substitution tree indexing. In *Proc. Int'l Conf. Rewriting Techniques and Applications*, pages 117–131, London, 1995.
58. A. Grbavec and D. Blostein. Mathematics recognition using graph rewriting. In *Proc. Intl. Conf. Document Analysis and Recognition*, pages 417–421, Montréal, Canada, 1995.
59. H. Hashimoto, Y. Hijikata, and S. Nishida. Incorporating breadth first search for indexing MathML objects. In *Proc. Int'l Conf. Systems, Man and Cybernetics*, pages 3519–3523, Singapore, 2008.

60. E. Hatcher and O. Gospodnetić. *Lucene in Action*. Manning, 2nd edition, 2010.
61. M.A. Hearst. *Search User Interfaces*. Cambridge University Press, 1st edition, 2009.
62. D. Hiemstra. Information retrieval models. In *Information Retrieval: Searching in the 21st Century*, pages 1–17. Wiley, 2009.
63. N.J. Higham. *Handbook of Writing for the Mathematical Sciences*. Society for Industrial and Applied Mathematics, Philadelphia, 1993.
64. J. Hu, R.S. Kashi, D. Lopresti, and G.T. Wilfong. Evaluating the performance of table processing algorithms. *Int'l J. Document Analysis and Recognition*, 4(3):140–153, 2002.
65. J.F. Hull. Recognition of mathematics using a two-dimensional trainable context-free grammar. Master's thesis, MIT, Cambridge, MA, 1996.
66. A. Kacem, A. Belaid, and M. Ben Ahmed. Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context. *Int'l J. Document Analysis and Recognition*, 4:97–108, 2001.
67. S. Kamali and F. Tompa. Improving mathematics retrieval. In *Proc. Digital Mathematics Libraries*, pages 37–48, Grand Bend, Canada, 2009.
68. T. Kanahori, A.P. Sexton, V. Sorge, and M. Suzuki. Capturing abstract matrices from paper. In J. M. Borwein and W. M. Farmer, editors, *Proc. Mathematical Knowledge Management*, volume 4108 of *LNAI*, pages 124–138. Springer, 2006.
69. T. Kanahori and M. Suzuki. A recognition method of matrices by using variable block pattern elements generating rectangular areas. In *Graphics Recognition – Algorithms and Applications*, volume 2390 of *LNCS*, pages 320–329. Springer, 2002.
70. T. Kanahori and M. Suzuki. Detection of matrices and segmentation of matrix elements in scanned images of scientific documents. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 433–437, Edinburgh, 2003.
71. T. Kanahori and M. Suzuki. Refinement of digitized documents through recognition of mathematical formulae. In *Proc. Int'l Work. on Document Image Analysis for Libraries*, pages 27–28, Lyon, France, 2006.
72. T. Kanungo, R.M. Haralick, H.S. Baird, W. Stuetzle, and D. Madigan. A statistical, nonparametric methodology for document degradation model validation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1209–1223, 2000.
73. K. Kim, T.-H. Rhee, J.S. Lee, and J.H. Kim. Utilizing consistency context for handwritten mathematical expression recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1051–1055, Barcelona, Spain, 2009.
74. Donald E. Knuth. *TeX and METAFONT - New Directions in Typesetting*. Digital Press, Bedford, MA, 1979.
75. A. Kohlhase and M. Kohlhase. Re-examining the MKM value proposition: From math web search to math web research. In *Proc. Symp. Towards Mechanized Mathematical Assistants*, volume 4573 of *LNCS*, pages 313–326, Springer, 2007.
76. M. Kohlhase. *OMDoc: An Open Markup Format for Mathematical Documents*, volume 4180 of LNAI. Springer, 2006.
77. M. Kohlhase, S. Anca, C. Jucovschi, A.G. Palomo, and I. Sucan. MathWebSearch 0.4: A semantic search engine for mathematics. (unpublished manuscript, http://kwarc/info/kohlhase/publications.html), 2008.
78. M. Kohlhase and I. Sucan. A search engine for mathematical formulae. In *Proc. Artificial Intelligence and Symbolic Computation*, volume 4120 of *LNAI*, pages 241–253. Springer, 2006.
79. G.E. Kopec and P.A. Chou. Document Image Decoding using Markov source models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(6):602–617, 1994.
80. A. Kosmala and G. Rigoll. On-line handwritten formula recognition using statistical methods. In *Proc. Int'l Conf. Pattern Recognition*, pages 1306–1308, Brisbane, Australia, 1998.
81. G. Labahn, E. Lank, S. MacLean, M. Marzouk, and D. Tausky. Mathbrush: A system for doing math on pen-based devices. In *Proc. Work. Document Analysis Systems*, pages 599–606, Nara, Japan, 2008.
82. G. Labahn, E. Lank, M. Marzouk, A. Bunt, S. MacLean, and D. Tausky. Mathbrush: A case study for pen-based interactive mathematics. In *Proc. Eurographics Work. Sketch-Based Interfaces and Modeling*, Annecy, France, 2008.
83. A. Lapointe and D. Blostein. Issues in performance evaluation: A case study of math recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1355–1359, Barcelona, Spain, 2009.
84. J.J. LaViola, A. Leal, T.S. Miller, and R.C. Zeleznik. Evaluation of techniques for visualizing mathematical expression recognition results. In *Proc. Graphics Interface*, pages 131–138, Windsor, Canada, 2008.
85. J.J. LaViola and R.C. Zeleznik. Mathpad^2: A system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics*, 23(3):432–440, 2004.
86. J.J. LaViola and R.C. Zeleznik. A practical approach to writer-dependent symbol recognition using a writer-independent recognizer. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(11):1917–1926, 2007.
87. S. Lavirotte and L. Pottier. Optical formula recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*, volume 1, pages 357–361, Ulm, Germany, 1997.
88. H.-J.and J.-S. Wang Lee. Design of a mathematical expression understanding system. *Pattern Recognition Letters*, 18(3):289–298, 1997.
89. C. Li, R.C. Zeleznik, T. Miller, and J.J. LaViola. Online recognition of handwritten mathematical expressions with support for matrices. In *Proc. Int'l Conf. Pattern Recognition*, pages 1–4, Tampa, Florida, 2008.
90. P. Liang, M. Narasimhan, M. Shilman, and P.A. Viola. Efficient geometric algorithms for parsing in two dimensions. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1172–1177, Seoul, Korea, 2005.
91. P. Libbrecht and E. Melis. Methods for access and retrieval of mathematical content in ActiveMath. In *Proc. Int'l Congress on Mathematical Software*, volume 4151 of *LNCS*, pages 331–342. Springer, 2006.
92. D. Lopresti and G. Wilfong. Evaluating document analysis results via graph probing. In *Proc. International Conf. Document Analysis and Recognition*, pages 116–120, Seattle, USA, 2001.
93. S. MacLean, G. Labahn, E. Lank, M. Marzouk, and D. Tausky. Grammar-based techniques for creating ground-truthed sketch corpora. *Int'l. J. Document Analysis and Recognition*, 14(1):65–74, 2011.
94. C.D. Malon, S. Uchida, and M. Suzuki. Mbarcelona, spainathematical symbol recognition with support vector machines. *Pattern Recognition Letters*, 29:1326–1332, 2008.
95. C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
96. S. Marinai, B. Miotti, and G. Soda. Mathematical symbol indexing using topologically ordered clusters of shape contexts. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1041–1045, Barcelona, Spain, 2009.

97. K. Marriott, B. Meyer, and K.D. Wittenburg. A survey of visual language specification and recognition. In *Visual Language Theory*, pages 5–85. Springer, 1998.
98. W.A. Martin. Computer input/output of mathematical expressions. In *Proc. Symp. on Symbolic and Algebraic Manipulation*, pages 78–89, Los Angeles, USA, 1971.
99. N. Matsakis. Recognition of handwritten mathematical expressions. Master's thesis, MIT, Cambridge, MA, 1999.
100. G. O. Michler. Report on the retrodigitization project "Archiv der Mathematik". *Archiv der Mathematik*, 77:116–128, 2001.
101. G.O. Michler. How to build a prototype for a distributed digital mathematics archive library. *Annals of Mathematics and Artificial Intelligence*, 38:137–164, 2003.
102. B.R. Miller and A.S. Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38:121–136, 2003.
103. E.G. Miller and P.A. Viola. Ambiguity and constraint in mathematical expression recognition. In *Proc. 15th National Conf. on Artificial Intelligence*, pages 784–791, Madison, Wisconsin, 1998.
104. R. Miner and R. Munavalli. An approach to mathematical search through query formulation and data normalization. In *Towards Mechanized Mathematical Assistants*, volume 4573 of *LNAI*, pages 342–355. Springer, 2007.
105. Y. Miyazaki and Y. Iguchi. Development of information-retrieval tool for MathML-based math expressions. In *Proc. Int'l Conf. Computers in Education*, pages 419–426, Tapei, Taiwan, 2008.
106. R. Munavalli and R. Miner. Mathfind: a math-aware search engine. In *Proc. Int'l Conf. Information Retrieval*, pages 735–735, New York, 2006.
107. G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.
108. G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *Proc. Int'l Conf. Pattern Recognition*, pages 347–349, Montréal, Canada, 1984.
109. I. Normann and M. Kohlhase. Extended formula normalization for ϵ-retrieval and sharing of mathematical knowledge. In *Proc. Towards Mechanized Mathematical Assistants*, volume 4573 of *LNAI*, pages 356–370. Springer, 2007.
110. M. Okamoto and K.T. Imait. Performance evaluation of a robust method for mathematical expression recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 121–128, Seattle, USA, 2001.
111. M. Okamoto and B. Miao. Recognition of mathematical expressions by using the layout structures of symbols. In *Proc. Int'l Conf. Document Analysis and Recognition*, volume 1, pages 242–250, Saint-Malo, France, 1991.
112. M. Okamoto and A. Miyazawa. An experimental implementation of a document recognition system for papers containing mathematical expressions. In *Structured Document Image Analysis*, pages 36–53. Springer, 1992.
113. M. Panic. Math handwriting recognition in Windows 7 and its benefits. In *Intelligent Computer Mathematics*, volume 5625 of *LNCS*, pages 29–30. Springer, 2009.
114. I. Phillips. Methodologies for using UW databases for OCR and image understanding systems. In *Proc. Document Recognition V*, volume 3305 of *SPIE Proceedings*, pages 112–127, San Jose, 1998.
115. R. Plamandon and S.N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.
116. M. Pollanen, T. Wisniewski, and X. Yu. Xpress: A novice interface for the real-time communication of mathematical expressions. In *Proc. Work. Mathematical User-Interfaces*, Linz, Austria, 2007.
117. L.R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.
118. A. Raja, M. Rayner, A.P. Sexton, and V. Sorge. Towards a parser for mathematical formula recognition. In *Mathematical Knowledge Management*, volume 4108 of *LNAI*, pages 139–151. Springer, 2006.
119. T.M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proc. Computer Vision and Pattern Recognition*, pages 521–527, Madison, WI, 2003.
120. T.M. Rath and R. Manmatha. Word spotting for historical documents. *Int'l J. Document Analysis and Recognition*, 9:139–152, 2007.
121. Mathematical Reviews and Zentralblatt für Mathematik. Mathematics subject classification, 2010. http://www.ams.org/mathscinet/msc/msc2010.html.
122. T.H. Rhee and J.H. Kim. Efficient search strategy in structural analysis for handwritten mathematical expression recognition. *Pattern Recognition*, 42(12):3192–3201, 2009.
123. S. Rüger. Multimedia resource discovery. In *Information Retrieval: Searching in the 21st Century*, pages 39–62. Wiley, 2009.
124. K. Sain, A. Dasgupta, and U. Garain. EMERS: A tree matching-based performance evaluation of mathematical expression recognition systems. *Int'l J. Document Analysis and Recognition*, 14(1):75–85, 2011.
125. G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, 1983.
126. H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, New York, 2006.
127. A.P. Sexton and V. Sorge. Abstract matrices in symbolic computation. In *Proc. Int'l Symp. Symbolic and Algebraic Computation*, pages 318–325, Genoa, Italy, 2006.
128. F. Shafait, D. Keysers, and T.M. Breuel. Performance evaluation and benchmarking of six page segmentation algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(6):941–954, 2008.
129. M. Shatnawi and A.S. Youssef. Equivalence detection using parse-tree normalization for math search. In *Proc. Int'l Conf. Digital Information Management*, volume 2, pages 643–648, Lyon, France, 2007.
130. Y. Shi, H.Y. Li, and F.K. Soong. A unified framework for symbol segmentation and recognition of handwritten mathematical expressions. In *Proc. Int'l Conf. Document Analysis and Recognition*, volume 2, pages 854–858, Curitiba, Brazil, 2007.
131. Y. Shi and F.K. Soong. Symbol graph based discriminative training and rescoring for improved math symbol recognition. In *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, pages 1953–1956, Las Vegas, USA, 2008.
132. A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(12):1349 –1380, 2000.
133. E. Smirnova and S.M. Watt. Communicating mathematics via pen-based computer interfaces. In *Proc. Int'l Symp. Symbolic and Numeric Algorithms for Scientific Computing*, pages 9–18, Timişoara, Romania, 2008.
134. E. Smirnova and S.M. Watt. Context-sensitive mathematical character recognition. In *Proc. Int'l Conf. Frontiers in Handwriting Recognition*, pages 604–610, Montréal, Canada, 2008.
135. S. Smithies. Freehand formula entry system. Master's thesis, University of Otago, Dunedin, New Zealand, 1999.

136. S. Smithies. Equation entry and editing via handwriting and gesture recognition. *Behavior & Information Technology*, 20(1):53–67, 2001.
137. S. Smithies, K. Novins, and J. Arvo. A handwriting-based equation editor. In *Proc. Graphics Interface*, pages 84–91, Kingston, Canada, 1999.
138. C.M. So and S.M. Watt. Determining empirical charateristivs of mathematical expression use. In *Proc. Mathematical Knowledge Management*, volume 3863 of *LNCS*, pages 361–375. Springer, 2005.
139. C.M. So and S.M. Watt. On the conversion between content MathML and OpenMath. In *Proc. Conf. Communicating Mathematics in the Digital Era*, pages 169–182, Aveiro, Portugal, 2006.
140. M. Suzuki, T. Kanahori, N. Ohtake, and K. Yamaguchi. An integrated OCR software for mathematical documents and its output with accessibility. In *Proc. Int'l Conf. Computers Helping People with Special Needs*, volume 3119 of *LNCS*, pages 648–655. Springer, 2004.
141. M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori. INFTY: An integrated OCR system for mathematical documents. In *Proc. Document Engineering*, pages 95–104, Grenoble, France, 2003.
142. M. Suzuki, S. Uchida, and A. Nomura. A ground-truthed mathematical character and symbol image database. In *Proc. Int'l Conf. Document Analysis and Recognition*, volume 2, pages 675–679, Seoul, Korea, 2005.
143. Y. Takiguchi, M. Okada, and Y. Miyake. A fundamental study of output translation from layout recognition and semantic understanding system for mathematical formulae. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 745–749, Seoul, Korea, 2005.
144. E. Tapia and R. Rojas. Recognition of on-line handwritten mathematical formulas in the e-chalk system. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 980–984, Edinburgh, 2003.
145. E. Tapia and R. Rojas. Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In *Graphics Recognition: Recent Advances and Persepectives*, volume 3088 of *LNCS*, pages 329–340. Springer, 2004.
146. E. Tapia and R. Rojas. A survey on recognition of on-line handwritten mathematical notation. Technical Report B-07-01, Free University of Berlin, 2007.
147. D. Tausky, G. Labahn, E. Lank, and M. Marzouk. Managing ambiguity in mathematical matrices. In *Proc. Eurographics Work. Sketch-Based Interfaces and Modeling*, pages 115–122, Riverside, CA, 2007.
148. The OpenMath Society. http://www.openmath.org/.
149. G.T. Toussaint. The use of context in pattern recognition. *Pattern Recognition*, 10:189–204, 1978.
150. K. Toyozumi, T. Suzuki, J. Mori, and Y. Suenaga. A system for real-time recognition of handwritten mathematical formulas. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 1059–1063, Seattle, USA, 2001.
151. K. Toyozumi, S. Takahiro, K. Mori, and Y. Suenaga. An on-line handwritten mathemical equation recognition system that can process matrix expressions by referring to the relative positions of matrix elements. *Systems and Computers in Japan*, 37(14):87–96, 2006.
152. K. Toyozumi, N. Yamada, K. Mase, T. Kitasaka, K. Mori, Y. Suenaga, and T. Takahashi. A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information. In *Proc. Int'l Conf. Pattern Recognition*, volume 2, pages 630–633, Cambridge, UK, 2004.
153. H.M. Twaakyondo and M. Okamoto. Structure analysis and recognition of mathematical expressions. In *Proc. Int'l Conf. on Document Analysis and Recognition*, volume 1, pages 430–437, Montréal, Canada, 1995.
154. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
155. S.M. Watt. An empirical measure on the set of symbols occurring in engineering mathematics texts. In *Proc. Int'l Work. on Document Analysis Systems*, pages 557–564, Nara, Japan, 2008.
156. S. Westman. Image users' needs and searching behaviour. In *Information Retrieval: Searching in the 21st Century*, pages 63–83. Wiley, 2009.
157. K. Wick. *Rules for Typesetting Mathematics*. Czechoslovak Academy of Sciences, The Hague, 1965. translated by V. Boublik and M. Hejlova.
158. H.-J. Winkler. HMM-based handwritten symbol recognition using on-line and off-line features. In *Proc. IEEE Int'l Conf. Acoustics Speech and Signal Processing*, pages 3438–3441, Atlanta, GA, 1996.
159. R. Yamamoto, S. Sako, T. Nishimoto, and S. Sagayama. On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In *Proc. Int'l Work. Frontiers in Handwriting Recognition*, pages 249–254, La Baule, France, 2006.
160. K. Yokoi and A. Aizawa. An approach to similarity search for mathematical expressions using MathML. In *Proc. Digital Mathematics Libraries*, pages 27–35, Grand Bend, Canada, 2009.
161. Li Yu. Image-based math retrieval using handwritten queries. Master's thesis, Rochester Institute of Technology, Rochester, NY, 2010.
162. R. Zanibbi, D. Blostein, and J. R. Cordy. Baseline structure analysis of handwritten mathematics notation. In *Proc. Int'l Conf. Document Analysis and Recognition*, pages 768–773, Seattle, USA, 2001.
163. R. Zanibbi, D. Blostein, and J. R. Cordy. Recognizing mathematical expressions using tree transformation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24:1455–1467, 2002.
164. R. Zanibbi, D. Blostein, and J.R. Cordy. Recognition tasks are imitation games. In *LNCS*, volume 3686, pages 209–218, September 2005.
165. R. Zanibbi, K. Novins, J. Arvo, and K. Zanibbi. Aiding manipulation of handwritten mathematical expressions through style-preserving morphs. In *Proc. Graphics Interface*, pages 127–134, Ottawa, Canada, 2001.
166. R. Zanibbi, A. Pillay, H. Mouchére, C. Viard-Gaudin, and D. Blostein. Stroke-based performance metrics for handwritten mathematical expressions. In *Proc. Int'l Conf. Document Analysis and Recognition*, Beijing, China (to appear), 2011.
167. R. Zanibbi and L. Yu. Math spotting: Retrieving math in technical documents using handwritten query images. In *Proc. Int'l Conf. Document Analysis and Recognition*, Beijing, China (to appear), 2011.
168. R. Zanibbi and B. Yuan. Keyword and image-based retrieval for mathematical expressions. In *Proc. Document Recognition and Retrieval XVIII*, volume 7874 of SPIE Proceedings, San Francisco, USA, 2011.
169. R.C. Zeleznik, T. Miller, C. Li, and J.J. LaViola. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *Int'l Symp. Smart Graphics*, volume 5166 of *LNCS*, pages 20–32. Springer, 2008.
170. L. Zhang, D. Blostein, and R. Zanibbi. Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions. In *Proc. Int'l Conf. Docu-*

ment Analysis and Recognition, pages 972–976, Seoul, Korea, 2005.
171. J. Zhao, M.-Y. Kan, and Y.L. Theng. Math information retrieval: user requirements and prototype implementation. In *Proc. ACM/IEEE Joint Conf. Digital libraries*, pages 187–196, New York, USA, 2008.
172. X.S. Zhou and T.S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8:536–544, 2003.
173. G.K. Zipf. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, 1949.
174. D. Zwillinger. *CRC Standard Mathematical Tables and Formulae*. CRC Press, 30th edition, 1996.