How to Find Mathematics on a Scanned Page

Richard J. Fateman
University of California, Berkeley

Abstract

We describe the design of document analysis procedures to separate mathematics from ordinary text on a scanned page of mixed material. It is easy to observe that the accuracy of commercial OCR programs is helped by separating mixed material into two (or more) streams, with only conventional non-math text handled by the usual OCR text-based heuristic analysis. The second stream, consisting of material judged to be mathematics, can be fed to a specialized recognizer. If that fails to decode it, it can be passed on to yet a third stream including diagrams, logos, or other miscellaneous material, perhaps including halftones. We explore the extent to which this separation can be automated in the context of scanning archival material for a digital library project including mathematical and scientific journal material.

1 Why separate mathematics from text pages?

Conventional OCR programs have low accuracy for mathematics for several reasons. The very sensible heuristics typically used for text recognition include computing the locations of text lines and estimating character sizes using global statistics as well as local processing. These programs may also use language-based statistics (perhaps a spelling dictionary) as tools to improve recognition rates. By contrast, mathematics is not necessarily arranged on lines, its character sizes vary, the letter and symbol frequencies are distinct from normal text, and many other text-oriented heuristics are directly counter-productive. Additionally, even if the mathematics were somehow recognized, conventional OCR programs whose traditional output is (say) ASCII text, need to be substantially augmented with some meta-level language before they can express "math results" as their output. Although most advanced word-processing programs have some such escape mechanism for "doing" mathematics, there is still no uniform standard for expressing two-dimensional layouts, subscript positioning, variable-sized characters, unusual math operators, etc. Research projects based on the needs of publishers, digital libraries, and mathematics computation systems have only recently begun to grapple with this issue[1], so a standard not available today, may be available in the future.

In the absence of special mathematics recognition, the natural fall-back position for an OCR system is to try to decode math as ordinary text, and provide whatever is closest to (some) text within that model. This typically makes a mess of the material. For example, superscripts are either unrecognized or pushed down to the baseline. An integral sign may be ignored or viewed as an S.

For some purposes, a better response is to provide only an image of a math section, treating it as analogous to a diagram or half-tone image [4]. Yet another alternative, assuming the component characters can be identified correctly might be to consider constructing a small section of PostScript. This would be less sensitive to degradation in re-rendering at different scales. This is not necessarily a realistic goal for high accuracy. Even familiar characters appearing in math contexts may fail to be recognized.

In our view, one would like to produce a version of the mathematics that reflects the semantics of the material in such a fashion that the result of OCR could be used in further computer processing. By this we mean it could be possible to evaluate a formula or manipulate it in some way. Typical computer algebra systems such as Maple, Mathematica, or Macsyma allow useful kinds of manipulation and each of these current systems allows one to display calculated formulas as typeset expressions. We are, in some sense, trying to reverse the process.

To summarize our motivation: given the difficulties of applying a uniform algorithm, we propose to improve the OCR success rate on mixed material by separating mathematics from the usual text. This may increase the accuracy of regular text recognition, and should allow us to use a specialized recognizer with appropriate attention to the semantics of the math material. Alternatively, we could commit mathematics to an image storage format.

Understanding mathematics is treated in more detail in a variety of recent papers [1] [2].

2 Previous work

Most papers that delve into recognizing (parsing) 2-D mathematics, without being specific, assume that the regions containing math are already known. Okamoto [6] is unusual in specifically noting "In these tests, table and picture areas were excluded and the distinction between text lines and

[1] http://www.can.nl/ abbott/OpenMath/index.html

DRAFT

mathematical expressions was specified manually." A recent paper by Lee and Wang [7] is directed to our task, but uses somewhat different techniques. They separate out "isolated" mathematics (we would call this "display" math), by its word-box statistics: higher boxes, separated by more than the usual paragraph spacing. There are good first-cut heuristics, but make mistakes: titles are labelled as formulas. One of the merits of this is that the characters need not be recognized. Lee and Wang treat "embedded" mathematics (we would call this "in-line" math), by first recognizing the characters. Characters that are known to be mathematical (such as $\sum$) are used as seeds for growing geometric "trees" of mathematical expressions, heuristically attaching symbols that are adjacent including those in super/sub-script positions. Other keys to iteratively begin the grouping step include the presence of super/sub-scripting, or matrix structures delimited by paired enclosure symbols like parentheses. (It would be unusual to find an actual matrix as in-line text, it seems.) Lee and Wang do not take advantage of the nature of the character font information, which is somewhat surprising, since we find that italics are a key feature of mathematical text. Lee and Wang do not attempt to confirm that the localized sections contain mathematics, in fact, leaving a parser and further corrective procedures for future work.

3 When can we find mathematics in a document?

It is advantageous to head off the processing of mathematics by an OCR system as soon as possible. In some circumstances one might know prior to processing a page at all, that it either consists of mathematics only, or is free of mathematics. In our target domain of scientific journals, most pages are mixtures.

Given this mixture we might be to identify mathematics on a page by rough analysis akin to zoning [5], or image signal processing [10]. These approaches do not, in our opinion, show much promise: some mathematics, especially in-line material, is rather indistinguishable from ordinary language at the level of texture.

The next step in sequence, and higher in expense is probably finding connected components and judging the likelihood of mathematics being present by the arrangement of bounding boxes [4]. This level of processing seems to be helpful in finding text strings when they appear within graphics, as in maps or engineering drawings. Approximately collinear components of appropriate sizes and spacing are assumed to be text characters, without regard to actual bit-patterns.

It would be pleasant to declare that one need go no further than this level to discriminate reliably between mathematics and ordinary text. Unfortunately, this does not work in practice.

We have found that to be reasonably robust, we must go at least one step further, and tentatively identify many of the connected components as particular characters. For well-scanned clean images with clearly separated letters, this is plausible. For low-resolution, noisy, or poorly-scanned images, this processing may be infeasible, and we cease to pursue this line. Fortunately, we expect to be processing relatively clean copies of technical papers scanned at high resolution. We do not expect to decode fax transmissions or poorly printed pages[2].

[2] If old papers in hand may do not scan well even at high resolution, we may have to reconsider.

To proceed with our system we must generally distinguish Italic letters from Roman letters, we must recognize digits, and identify dots and horizontal lines. We do not directly use some other information that is readily available from recognizing isolated letters, such as character size. We will, however, need to compute with adjacency relationships and context because many characters can appear either in text or mathematics, and we must distinguish them. While we do not expect 100% separation, with some training (especially on the character set in use for mathematics) we expect that only a modest amount of editing by hand will be needed to separate the streams of data.

4 Background: where do we find mathematics?

We are concerned with two types of math present in documents, *display* mathematics and *in-line* mathematics.

Display math is less problematical in principle in that it is usually set off by extra vertical space, and can sometimes be detected as non-text even before it has been examined in detail. It has distinctive lower density compared to a normal text paragraph, has smaller words, and unusual line statistics. This are an unreliable indicator because a small text paragraph may have similar characteristics.

Because display math lives in separate regions on the page, identifying it by hand can be done quickly. If we are committed to recognizing it by automatic means, we use the same heuristics as for in-line math, described below. That is, display math will consist mostly of special symbols, italics, and few Roman characters or words.

In-line mathematics is material that is run-in with text like this: $e^x + \sin z$ and in the worst case, is pretty much indistinguishable from text unless you have substantial context available. Some mathematics journals, in an effort to conserve space, use as much in-line mathematics as possible, even if it then requires running expressions from one "text" line to the next. This causes substantial trouble for automatic segmentation, and we have not solved it. Other typesetting styles forbid such in-line run-on mathematics, forcing such expressions to displays.

In our view, any hope for high accuracy in overall automatic processing of mixed material requires a fairly effective to identify most in-line math.

5 How much math is in a document?

Obviously it varies.

Our initial test was on an economics modeling paper, and selecting each math section on a page was only moderately burdensome, at least if decent interactive tools are available.

Other subject areas are more math-intensive. For a typical math journal article, nearly every sentence contains some in-line mathematics, and many sentences have several math sections. For such documents an automatic selection mechanism is important.

6 What precisely differentiates text from in-line math?

If one were properly typesetting a document in TEX, and had access to the source code for the document, the math would be all text within matching $ or material otherwise designated as math-mode.

Of course if we had the luxury of examining the TEX source of the typeset document we were scanning, the job of scanning would be largely redundant. Given only the output page, it is less clear.

Officially we can declare defeat is inevitable because we can exhibit examples where we are unable to distinguish automatically and in a context-free manner, between the italic characters in a word and the symbols in mathematics. (See the appendix.) In practice we simply accept the prospect that some mistakes will be made and/or some material must be recognized with hand assistance.

Remark: We have seen papers prepared by TEX-nicians who habitually uses math-mode for italics. For example, they use `$coefficient$`, which typesets as $coefficient$ instead of `{\it coefficient}`. This latter form typesets as the properly spaced *coefficient*.

In fact, we are of two minds here on what to include in our identification of math sections.

One view is that isolated symbols might as well be text, as in p within "At each point p where... " Indeed, since an OCR program is likely to correctly read a single italic p typeset on the normal text base-line, this notion — let as much be text as possible — could be extended to isolated boldface, Greek letters, or sequences of these symbols. This view is especially attractive if the only mechanism for identifying math is to select it from an image by hand. As already mentioned, this technique is useful for benchmarks involving only modest math material.

A second view is that it is advantageous to treat *everything* as math unless we decide by specific means that it is text. In this view we separate material into two bags of symbols as soon as possible. The elements of these bags encode bitmaps (pictures), locations and size, with some (perhaps tentative) identification as characters, and perhaps baseline information derived from the character. The advantage here is that such an automatic segregation of characters may provide a very good approximation to text *vs.* non-text, and that once they are separated, we can look at aggregates of special symbols or italics in the bag of math symbols and restore them to the text bag more economically than the reverse. This "looking" may be done automatically, or by hand. For example, if we initially place all parentheses all dashes into the math bag, we can move these symbols — if they are sufficiently isolated, back into the text bag.

(Major defect in this idea: "theorem environments" in some mathematics journals, and as supported by LATEX typeset straight text words in italics. These words may look like math, and may in fact be difficult to distinguish from math except by deeper natural language analysis. Similarly, texts including *emphatic sections* will also make trouble for us. We address the issue of where to put unrecognized merged or broken symbols later.

In either case, if we can figure out most of the math characters and clump them into math rectangles, we could then ask the user to review the automatic selections, and repair errors. The user can

(a) Identify new mathematics that was entirely missed by the automatic process.

(b) Demote alleged mathematics back to normal text (e.g. Some words in italics are NOT mathematics).

(c) Re-group mathematics sections that were mistakenly grouped by the automatic process.

7 A proposal for automatic process of separating math from text

7.1 Pass One: initial separation

We initialize two bags of symbols to be empty. The text bag and the math bag.

We put into the math bag the following connected components[3] An alternative is to place only those not extracted as math in the text bag, including anything unrecognized, and a third is to place some "dummy zone" objects in the text bag as placeholders for math.

Why is unrecognized material treated as text?

Simply in the hope that other peoples' clever heuristics for text processing will figure out what these pieces are. It may be that in our next stage of processing we will notice that such unidentified objects are entirely surrounded by math pieces and we will then try to merge broken pieces (etc.) to justifiy their presence as math. If in the future our math recognizer will be so strengthened that it will be more effective than commercial text recognizers on broken pieces, we might consider shunting unrecognizable material into math first.

7.2 Second pass: correct for too much math

We remove items from the math bag, leaving them in the text bag if implausibility sets in. More precisely, we clump math symbols together by proximity. After this operation, if we find in the math bag an isolated symbol with respect

[3] By prior heuristic processing we may have joined some pieces multi-part characters together):

1. special mathematics characters: +, =, Greek letters, scientific symbols, large parentheses. All horizontal lines. All commas. We know that this is over-reaching, but we will correct it.
2. Italics/bold, We know that this may include non-math text, but again we will have to correct this in a later pass.
3. Numeric digits in Roman typeface. Italic numbers are generally NOT used in mathematics, but in page numbers etc. It is a challenge to distinguish between a lower-case letter l and a Roman digit 1. If every occurrence of an l must be considered equally plausible as a 1, we must rely rather more heavily on context than we would like. To show that we may in general have to deal with l and 1 as members of the same equivalence class, observe figure 1, where there are 13 glyphs, one of which is a number. Which one is revealed later in the appendix to this paper. We may also find that the digit zero (0) is confused with the letter O. It may also a challenge to distinguish between the bottom of a letter i and a (subscript-size) number 1.

   We also consider
4. Occurrences of () especially if we can detect pairs that occur close to each other. Occurrences of [] .
5. symbols in super/subscript locations (this info is unfortunately not readily available).
6. Mathematics words like sin cos tan. (this info is also not readily available). Standard tables of integrals also use rather ordinary upper-case Roman letters for special functions of physics (e.g. J, K, W, E) This may have to be handled by special case hand recognition, or by a third pass, described below.
7. Dots, commas and similar punctuation. Dots appear over the letters i,j as well as :;. Periods at the end of sentences or abbreviations. Decimal points. Depending on how we have processed the material, some of the dots may already be explained (e.g. i dots), but there may still be some floating ones. Commas occur frequently in text, but also occur in math

Our current scheme leaves a copy of everything, *including suspected math*, in the text bag, and lets the commercial program muddle about with this. Perhaps it may help the OCR program in determining line computation, and it is our experience that removing all math plays havoc with zoning.

to other math symbols, but one that appears to be in close proximity to other text symbols, we assume that it is, in fact, text bag. A general clumping operation[4] similar to that which might be used in a bottom-up zoning process. This generally gets most of the groups, but not all of them, so we can be more specific.

In various tests we find items that would incorrectly remain in the math bag. Consider "... ignoring p' (the derivative with respect to t)." Here we find, incorrectly, the two math clumps: $p'($ and $t)$. As another example, consider "Substituting (1), (2) and (3) into S:"

To avoid these errors we add rules to discard from the mathbag

- A left parenthesis without math to its right.
- A trailing[5] right parenthesis whose matching left parenthesis is missing from this clump.
- A horizontal line unless it has math to its right, above, or below.
- A leading or trailing comma or dot in a clump.
- An isolated 1 or even 11 in the math bag that is however within a single-character-space of text, which may instead be an l or an ll. (We treat 1 and l equivalently)
- An isolated 0 or even 00 in the math bag that is however within a single-character-space of text, which may instead be an O or an OO. (We treat 0 and O equivalently)

7.3 Third pass: correct for too much text

Words like sin, tan, or ANYTHING ELSE in Roman text appearing in mathematics can be distinguished heuristically from other Roman text in that they will be surrounded in the text-bag by open space, but will be very near components in the math-bag. We can blur the math into larger groups (smearing their bounding boxes to form zones) and assert that any Roman text inside or intersecting these zones is math. In particular, this allows us to assert that l's or 1's when they seem to appear within, or close to, mathematics clumps, are digits. After making such transformations, we re-clump the mathematics: sometimes these newly recognized symbols provide important bridges between symbols to allow us to clump mathematics together, and subsequent processing uses these clumps. It is possible (and inexpensive) to re-confirm the mathematics/text distinction by looking, once again, for isolated symbols.

7.4 Remaining problems

Italic words will generally be categorized as mathematics. I don't see a good heuristic for solving this problem, and so this may need human intervention.

Experience suggests that confusion of letter l with number 1 may still cause some grief: By distinguishing them by context only, we are more successful. We distinguish each of these from the bottom part of the letter i by noting that i has a dot over it!

8 Procedural details

How does our program work?

Initially we must have progressed to the point where we have a mechanism for identifying isolated symbols by character and font. In practice, our method has been to collect a large sample of connected components from a corpus; we cluster them by statistical measures, and have a human check these clusters, correcting as appropriate. This set of clusters/properties forms a set of property-vectors in a dictionary. A human then identifies each entry as a particular glyph. We may actually have several different property-vectors with the same identification (that is, the text may include sets of differently shaped letters which however we prefer to not distinguish, and so are identified as the same letter. Typically we do not have much to say about parentheses, even though we find them in a variety of shapes. Although we may not need to distinguish some such font or size variations from each other, for formula identification we must distinguish between Italic and Roman versions of letters and digits.

In practice, one system we set up continues to learn new characters as it proceeds about its recognition tasks seems to be an attractive way of improving accuracy in an academic or research setting. It is presumably less attractive in a purely production mode.

For each page we proceed as follows:

- Deskew, filter out noise, and clean up the image. Collect all connected components (ccs) on a page, and identify each by that dictionary entry that is closest to it. If a cc is so remote[6] in characteristics from anything we have previously identified, we can either ask for human help in identification, specify that it is noise, or specify that it is unknown text. (Since we use the same dictionary for our math processing as for isolated character recognition, if we send unrecognized items through to our math processing, we will not make much further headway there. Our best bet at the moment is to assert that it is text, in the hope that the commercial text recognition system will make sense of it.) Perform some elementary joining of separate components such as matching the top and bottom of i and j characters when possible.
- Based on such identification separate all items into two bags, math and text. The text bag includes all Roman letters, Italic numbers. The math bag includes punctuation, special symbols, Italic letters, Roman digits, and other marks (horizontal lines, dots), etc. In figures 2 and 3 we show corresponding sections of a small excerpt of a paper [9]. Figure one has the initial contents of the math bag, and figure 2 the text. Superimposed two-color illustrations are sometimes more useful in understanding the relationships of the components.
- Group the math bag components into zones according to proximity. This is a very sensitive, and potentially error-introducing operation. If we include too much territory in a math zone, we will absorb components such as dots, commas, and parentheses that rightfully

[4] we use a computation based only on bounding boxes. It is a kind of iterative "smearing" where all bounding boxes horizontally within a chosen x-smear distance are merged, repeatedly, until the process ceases to make any changes. Then a similar y-direction merging is done. Setting these parameters is tricky. The x-smear should be larger than the inter-character space, but smaller than the inter-word space. The y-smear is especially tricky since it must be considerably smaller than the inter-line space, yet large enough to grab all the parts of a built-up fraction or even a multiple-line display.

[5] that is, it has no assured math characters to its right

[6] How remote is a judgment call.

Figure 1: Twelve l's and a 1

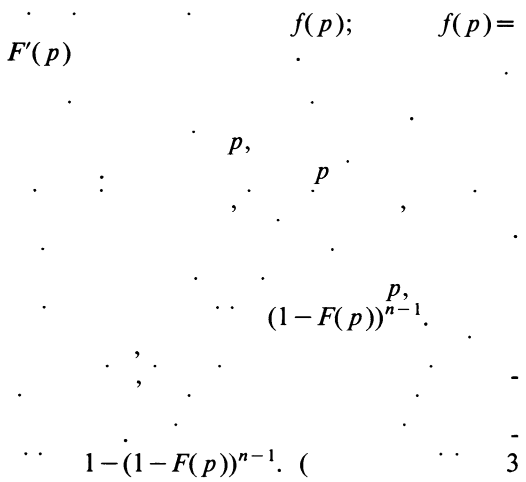

Figure 2: Initial Collection of Math

belong in text. The technique we use is simple, computationally and intuitively. For simplicity and speed we consider only the bounding boxes of each of the connected components. Set two parameters[7], vertical smear `vs` and horizontal smear `hs`. Any two bounding boxes separated horizontally by `hs` or less are joined and a new bounding box around their union is formed. This is iterated until no more joins are possible. Then a similar operation, but in the vertical dimension is performed. Bounding boxes may be touching, in which case even `hs`=0 and `vs`=0 will cause them to "smear". More typically, `hs` is larger than `vs`.

- After this grouping within the math bag, some symbols still remain isolated. Those symbols which might be attributed either to math or text given appropriate context, but which appear to be too far from other math symbols to be grouped together with them, are moved to text. These include each isolated dot, comma or parenthesis. We also consider that dots, commas, and parentheses that are only in proximity *with each other* are ":" or ";" or ".)" etc. should also be considered isolated, and are moved to text as well.
- With the exceptions just noted, all groups of components in the math bag remain in the math bag. Other, isolated items that are unambiguously math remain in the math bag as well. For example isolated italic letters or isolated Greek letters stay as math.
- Next, join up the text bag into groups according to proximity.

[7] whose values are a judgment call based on how tightly the document is typeset, and the scan density in dots per inch.

- Some text words are relatively isolated from other text, but are within zones that have been previous established in the math bag to contain mathematics. Move these words, which we hope will include "sin" etc. into the math bag.
- Final human review: Look at the bags of math vs. text, and review/correct the choices. Especially, make sure that multiple-line in-line math is correctly picked out, and that we are not mistakenly including italic text as mathematics. This can be done by displaying a page either in two separate pieces, or in one piece but in two colors. Mistakes are likely to be more apparent in the two-page display, but the two-color display takes up less screen real-estate.
- Reassemble the bag of text and run it through commercial OCR programs.
- Run the bag of math through math recognition, or alternatively, collect the sections and store them as GIF or similar image formats.
- Collect the two sets of results and combine them as a description of the page.

9 Example

Figures 2 and 3 are extracted from column 2 of page 4 of a paper on economic modeling [9] from our digitized electronic library collection[8]. We found that for the whole page, the math-bag contained 684 elements, 207 of them dots over i, j, i, j, or dots in colons or semicolons. By merging these dots together early, subsequent processing is likely to be faster. Yet this is not always possible, since some dots are placed over the "i" portion of merged letters[9].

The text-bag contained 2215 components. In the second phase, about 216 are isolated dots that must be returned to the text bag 5 pairs of parentheses are returned to the text page as well.

In the third phase, no "Roman text" math occurs on this page to be returned to math bag.

10 Known systematic errors

- Strings of isolated italic letters will ordinarily be left as math. If the italic letters are touching, and hence not recognized by connected component analysis as italics, will be treated as text. Hence italic words may be either mathematics or text.
- Strings of numbers and symbols sometimes appear in text as, e.g. (1) apples; (2) pears. In our classification, "(1)" is mathematics. So are "1)" or "1:".
- Given the strong possibility of 1 and l being interchanged, a word like lilly may be scanned as `1i1ly` and treated as either partly mathematics or, depending on spacing, entirely mathematics.
- The attribution of periods is weighted toward mathematics in that a math section ending a sentence will include the terminating period e.g. in this sentence, notice the period after $f(x)$.

[8] http://elib.cs.berkeley.edu

[9] In particular we have instances of "xi" and "ik" merged.

on Let be the cumulatıve dıs
trıbutıon functıon for thus
almost everywhere
We can now construct the expected profıt functıon for a representatıve store When a store charges prıce exactly two events are relevant It may be that ıs the smallest prıce beıng charged ın whıch case the gıven store gets all of the ınformed customers Thıs event happens only ıf all the other stores charge prıces hıgher than an event whıch has probabılıty On the other hand there may be some store wıth a lower prıce ın whıch case the store ın ques tıon only gets ıts share of the unınformed customers Thıs event happens wıth proba bılıty By Proposıtıon

Figure 3: Initial Collection of Text

- Figures or line-drawings that include dotted or dashed lines may contain connected components that look, to our isolated character recognition routines, like mathematics characters. We do not yet solve the problem of separating these out and treating them like figures (which should probably be treated as image data).
- Often material is typeset so tightly that descenders from one text line very nearly touch ascenders or superscripts from the line below. This leads to oversmearing errors.

While some of these errors may be correctable by additional processing, they can be rather subtle. In cases where completely correct interpretation is required, it may be necessary to make corrections by hand.

The final and (we believe "most automatic") approach depends on the robustness of the text processing component to ignore confusing math, but to make sense of everything else. The tactic is to place the FULL page in the text bag, including the parts we believe are math. Sections of material that are fully understood by the text recognizer and are *also* fully recognized by math, are represented in two ways and, in the fullness of our digital library representation, the end user should be able to make a meaningful choice.

11 Acknowledgments

Thanks to the Berkeley elib-docstruct seminar participants for comments.

12 Appendix: On Ambiguity

Here we provide some evidence to suggest that the problem is, in the worst case, "AI (artificial intelligence) Complete" That is, clearly you can solve this problem if you first construct an artificial human: a computer program that reads and understands all of mathematics and natural language. On the other hand, if you can solve this separation problem completely, then we suggest that full natural language comprehension cannot be far behind.

Consider the following questions:

- Is there any mathematics in the title of this numerical analysis lecture: "Approximating trigonometry functions for fun and profit: the wages of sin"?
- Can we find the math in $ad - bc$ by *ad hoc* methods?

Finally, in figure 1, the digit "one" is third from the right.

References

[1] D. Blostein and Ann Grbavec. " Recognition of Mathematical Notation," Chapter 22 in P.S.P. Wang and H. Bunke, (eds), *Handbook on Optical Character Recognition and Document Analysis,* World Scientific Publishing Company, 1996.

[2] Richard Fateman, Taku Tokuyasu, Benjamin Berman, Nicholas Mitchell. "Optical Character Recognition and Parsing of Typeset Mathematics." *J. of Visual Commun. and Image Representation* vol 7 no. 1, (March 1996), 2—15.

[3] Richard Fateman and Taku Tokuyasu. "A Suite of Programs for for Document Structuring and Image Analysis using Lisp", UC Berkeley, technical report, 1996.

[4] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," IEEE Trans. Pattern Analysis and Machine Intelligence, vol 10, no 6 (Nov. 1988), 910–918. (reprinted in [8])

[5] Mukkai Krishnamoorthy, George Nagy, Sharad Seth, Mahesh Viswanathan. "Syntactic segmentation and labeling of digitized pages from technical journals," *IEEE Trans. on Pattern Analysis and Machine Intell.* vol 15 no. 7 (July 1993), 737—747.

[6] M. Okamoto and A. Miyazawa. "An experimental implementation of a document recognition system for papers containing mathematical expressions, in H.S. Baird et al (ed). *Structured Document Image Analysis* Springer-Verlag 1992, 36–51.

[7] Hsi-Jian Lee and Jiumn-Shine Wang. "Design of a mathematical expression understanding system," *Pattern Recognition Letters* 18 (1977) 289—298.

[8] Lawrence O'Gorman and Rangachar Kasturi: *Document Image Analysis*, IEEE Computer Society Press, 1995.

[9] Hal R. Varian. "A Model of Sales" Berkeley Elib document 620, originally published in *The American Economic Review*, 651–659. Sept, 1990.

[10] F.M. Wahl, K.Y. Wong and R.G. Casey. "Block segmentation and text extraction in mixed text/image documents," *Compt. Vision, Graphics, Image Proc.*, vol 20, 275–390, 1982.