ANATOMY OF A VERSATILE PAGE READER

Henry S. Baird

AT&T Bell Laboratories
600 Mountain Avenue, Room 2C-557
Murray Hill, New Jersey 07974 USA

Abstract

An experimental printed-page reader that is easy to adapt to various languages is described. Changing the target language may involve simultaneous changes in symbol sets, typefaces, sizes of text, page layouts, linguistic contexts, and imaging defects. Our strategy has been to isolate the effects of these sources of variation within separate, independent engineering subsystems. In this way, we have been able to construct, with a minimum of manual effort, classifiers for arbitrary combinations of symbols, typefaces, sizes, and imaging defects. We have tried to rid the algorithms of all language-specific rules, relying instead on automatic learning from examples and generalized table-driven methods. For some tasks we have been able to avoid language-dependency altogether: for example, for geometric page layout analysis we have found a global-to-local strategy that requires no prior knowledge of the symbol set. We can exploit linguistic context, such as provided by dictionaries, through data-directed filtering algorithms in a uniform and modular manner, so that pre-existing tools developed by computational linguists can readily be applied. We illustrate these principles through trials on English, Swedish, Tibetan, and special technical texts.

1. Introduction

We describe the architecture of an experimental printed-page reader that exhibits a high degree of versatility: that is, it can be adapted to new languages with a minimum of manual effort. It has been applied to multiple-typeface English text and single-typeface Swedish, Tibetan, chess notation, and mathematical equations. An overview of an early version of the reader appeared as [KPB87]; recent details of its algorithmic components are published elsewhere (references are cited below).

Automatic page readers must cope with many sources of variation, including symbol sets, typefaces, sizes of text, page layouts, linguistic contexts, imaging defects, and output encodings. Our engineering strategy has been to organize the system so that each of these can be handled separately and independently, and the results combined in arbitrary ways. To achieve this, new algorithms have often been required. For example, our geometric layout analysis subsystem is virtually language-independent thanks

To appear in *IEEE Proceedings*, special issue on ‘‘Document Image Analysis,’’ edited by S. Mori and T. Pavlidis, Summer, 1992.

to novel methods for skew-correction and the analysis of white space.

Our approach to symbol (character) recognition is a hybrid of structural shape analysis and Bayesian statistical classification, and is trainable by example, usually off-line. Shape features are constructed (not merely selected) during an automatic analysis of the training set; as a result, accurate classifiers may be rapidly built for diverse symbol sets. A pseudo-random image defect generator permits the automatic construction of classifiers given as few as one prototype image per symbol. Thus the manual effort to learn new sets of symbols and typefaces is reduced to a minimum. The classifiers exhibit strong generalization and compression across typefaces, text sizes, and commonly-occurring image defects.

The algorithms for layout analysis and symbol recognition are applicable to any writing system in which the symbols are rigid and disconnected (spaced apart from one another), most of the time. Where characters do touch, segmentation is controlled by classifier confidence scores, so that language-specific rules are needed only for cases that are ambiguous by shape. Other linguistic context, such as provided by dictionaries, punctuation rules, and other lexical constraints, is exploited in a uniform data-directed manner.

Table-driven algorithms have been used wherever possible. As a result, the programs for recognition of symbols and inference of text size and baseline possess no language-dependent special cases. Linguistic context may be specified by arbitrary word-lists and general-purpose regular-expression patterns. Final output encoding (such as ASCII) is via user-specified tables.

The document image analysis literature has so far only rarely addressed the topic of language-independence and its architectural implications. Recently, Spitz *et al* have investigated [WS88] the processing of multiple-language documents. Each of the component technologies discussed below has an extensive literature; surveys of them are omitted here, for space reasons, but can be found in the papers referenced.

Section 2 defines the class of problems for which the page reader is designed, and gives an overview of the system. Page layout analysis is discussed in Section 3, and symbol recognition in Section 4. The exploitation of linguistic context is described in Section 5, and output encoding in Section 6. The tools and data structures used are described in Section 7.

2. Task Definition

The page reader accepts bilevel images (black characters on a white background) of pages of machine-printed or typewritten text. For the purposes of this paper, we will assume that only a single language is present, and it is specified in advance. The specification of a language determines the symbol set, reading order, linguistic constraints, and output encoding rules. However, typefaces, text sizes, and page layouts will normally not be specified in advance. Of course it is also useful to be able to construct custom readers for higher accuracy on known symbols, typefaces, and defects.

The sequence of computation stages is as follows:

1. *geometric layout analysis*: connected components analysis, skew- and shear-

correction, segmentation into text blocks, line-finding within blocks, character-finding within lines, and determination of reading order;

2. *symbol recognition*: classification of characters by shape, inference of text size and baseline (or top-line), segmentation of lines into words by spacing, and shape-directed resegmentation of characters within words to handle touching and broken characters;
3. *linguistic contextual analysis*: segmentation of lines into words by lexical means, exploitation of dictionaries and punctuation rules, and enforcement of inter-word consistency constraints;
4. *logical layout analysis*: partitioning blocks into sections, labeling sections by function, and reassembling sections (across blocks) into reading order; and
5. *output encoding*: mapping the internal representation of the analysis into character codes (*e.g.* ASCII or JIS), possibly annotated by formatting directives (*e.g.* troff or SGML).

By design, each of these stages operates as independently of the others as possible. Up to the present time, we have managed to avoid feedback control paths between them, while holding down the size and complexity of the data structures that are fed forward. This strategy places severe performance constraints on the algorithms in each stage. For example, we require that geometric layout analysis arrive at exactly one interpretation, without any knowledge of the symbol set, recognition results, or linguistic context. In later stages, we relax this constraint: for example, the recognition stage is permitted to write a two-level lattice of word and character interpretations. This simplicity of control is deliberate, and is made possible by careful algorithm design within each stage; it has not been achieved by sacrificing performance.

We assume that within each page there is a single alignment coordinate system, defined by the horizontal skew and vertical shear angles; within it, all symbols are expected to be upright (landscape layouts must be manually specified, so that the image can be rotated before layout analysis). Layouts must be *Manhattan* after alignment correction: that is, they can be partitioned into isolated, convex blocks (columns) of text by horizontal and vertical line segments cutting through white space; this is a large and useful class of layouts. The system knows in advance, from the language specification, whether text lines within blocks are horizontal or vertical, and, if horizontal, whether reading order is left-to-right or right-to-left. Within blocks, text sizes and line spacings may vary, but each line is expected to have a single text size and baseline (or top-line) location.

Writing systems are expected to be non-cursive and disconnected: that is, their symbols are rigid and spaced apart, at least nominally. In practice, symbols may occasionally touch or overlap, giving rise to character-segmentation problems which the system attempts to solve. Writing systems meeting these criteria include Latin, Greek, Cyrillic, Chinese, Kanji, Hangul, Hebrew, their derivatives, and many others. Writing systems normally not of this kind include Arabic, Devanagari, and their derivatives.

At present, logical layout analysis identifies paragraphs and suppresses garbled text; this stage will not be discussed further. Algorithms to cope with graphics, linedrawings, and photographs also will not be discussed here, for lack of space.

3. Geometric Layout Analysis

The analysis of layout geometry follows a global-to-local strategy [Bai88b], that is, greedy and guided by global evidence. A non-backtracking sequence of model-refinement steps is executed in an order constrained by dependencies among model parameters and maximizing the statistical support available at each step for inference of the parameters. Experiments suggest that this is more robust than bottom-up merging methods and more efficient than backtracking top-down methods, while requiring relatively little *a priori* information. In particular, the method requires no prior knowledge of the symbol set, and only rough estimates of the range of text sizes.

First, black 8-connected components are extracted. On a typical page, approximately one in five pixels is black, each run contains more than 10 black pixels, and each component over 25 runs: so that, the number of components is three orders of magnitude smaller than the number of pixels. Thus we have focused attention on layout algorithms whose basic data item is the component. Components are represented exactly (neither filtered nor approximated) as line-adjacency graphs [Pav82]. This data structure supports, in time linear in the number of runs, the extraction both of boundary lists and moments of area, required for symbol recognition. Also in linear time, it can be mapped to and from a 8×-compressed form suitable for external files (the CCITT Group 4 encoding [CC84]).

Using the set of approximate locations of components, the dominant skew and shear angles of the page as a whole are measured. If these angles differ from strictly horizontal or vertical by more than 0.03 degrees, they are corrected. Our skew estimation algorithm [Bai87] is one of the fastest and most accurate reported, and works without modification on a variety of layouts, including multiple blocks, sparse tables, and mixed sizes, line-spacings, and typefaces. Its accuracy is unaffected by touching characters, as long as they are in the minority; the method is slightly sensitive to whether the symbol set has a baseline or top-line convention. Later, as blocks of text are isolated, they are skew- and shear-corrected again.

Next, blocks of text are isolated. For this, the structure of the background (white space) is analyzed, assisted by an enumeration of all maximal white rectangles [BJF90]. This enumeration required an asymptotically and absolutely fast algorithm: we developed one that, aside from a sort, achieves an expected runtime linear in the number of black connected components. In applying this algorithm to page layouts, the crucial engineering decision is the specification of a partial order on white rectangles to select shapes and sizes likely to occur in the background of a usable partition. This shape-directed order determines a sequence of partial covers of the background, and thus a sequence of nested page segmentations. In experimental trials on Manhattan layouts of pages printed in a variety of languages, good segmentations often occur early in this sequence, using a simple and uniform shape-directed rule (see Figure 1). Unlike many reported bottom-up (iterative merging) strategies, this does not depend on a large number of symbol-set-dependent rules for good results. It also does not require prior knowledge of page orientation (portrait or landscape), or reading order. It appears that publishers and printers in many languages, constrained by similar printing technology and legibility conventions (for examples, see [CMS82]), use white space as a layout delimiter in similar ways.

Figure 1: This *Computer Journal* page of 5590 black connected components generates 15106 maximal empty rectangles, of which the first 151 (1%) in shape-directed order define the cover set shown. The skew- and shear-corrected layout is illustrated on the left by the bounding rectangles of black components, and the segmentation is shown on the right as the set of white components remaining after the union of the cover set has been computed. One isolated line of text is fragmented by a spurious vertical cut (at the top of the page).

Next, each isolated block is segmented into lines of text. Again, a global-to-local strategy is effective, even on columns in which line spacing and text size vary over a wide range. This is achieved by analysis of the horizontal projection profile using digital signal processing methods: local derivatives and autocorrelation to estimate the dominant line-spacing; smoothing to sharpen the peaks associated with text lines; non-maximum suppression to locate the peaks; and finally local-minima finding to choose horizontal cuts. Trials on pages printed in over a dozen writing systems suggest that this method succeeds on the great majority of layouts, without language-specific rules.

Several researchers report that it is possible to estimate typographic properties of text lines, such as text size and baseline location, from their projection profiles — but we choose to defer these decisions until after symbol recognition, when a more robust and language-independent method becomes available.

4. Symbol Recognition

Technical challenges in printed symbol recognition arise, generally speaking, from three types of variation:

(a) *symbols:* the set of idealized, rigid, elementary shapes;

(b) *deformations:* permissible analytic shape variations, including scaling (text size) and translation (height above baseline); and

(c) *imaging defects:* imperfections in the image due to printing, optics, scanning, spatial quantization, binarization, *etc*.

We have organized the recognition subsystem so that each of these may be managed independently of the others. Classifiers can be built for any collection of rigid symbols under any specified defect distribution: this normally occurs off-line during highly-automated training; the resulting classifier tables can be archived and selected at runtime. The range of permissible text sizes, and the sensitivity of recognition to discrepancies in size and height above baseline, can be specified at runtime.

A quantitative model of imaging defects [Bai90] permits us to build accurate classifiers with a minimum of manual effort. The model includes parameters for size, digitizing resolution, blur, binarization threshold, pixel sensitivity variations, jitter, skew, stretching, height above baseline, and kerning. The model has been calibrated on image populations occurring in printed books and typewritten business material, and is expressed as a distribution over the model parameter space. Associated with it is a pseudo-random defect generator that reads one or more sample images of a symbol and writes an arbitrarily large number of distorted versions chosen from the distribution. Figure 2 shows examples of this.

[2.0;2.5)
[1.5;2.0)
[1.0;1.5)
[0.5;1.0)
[0.0;0.5)

Figure 2: Pseudo-randomly generated defective images of a Cyrillic /LJE/ character, at 5 point size and 300 pixel/inch digitizing resolution. Each line holds images whose defect parameter vector's Mahalanobis distance lies in the range shown. Mahalanobis distance is Euclidean distance from the mean parameter vector, scaled component-wise by standard error, and is thus a measure of severity of the distortion.

One use of the generator is to allow classifiers to be built for arbitrary combinations of symbols and typefaces, with minimal manual effort. For example, a classifier for two typefaces of Tibetan, with over 400 distinct symbols each, was built from scratch with less than two weeks work by one person, by selecting from the document images one sample image for each symbol. The most tedious task was the unavoidable one of labeling each image with its language-specific symbol code. When a prelabeled machine-legible typeface description is available, training a classifier can be accomplished in less than a day.

In experiments on 100 typefaces of the ASCII symbol set [BF91], we measured success rates on isolated characters of at least 95% top choice and 99.5% within the top 3 choices, on any text larger than 7 point digitized at a resolution of 400 pixels/inch. The testing phase of this experiment involved nearly one million character images. Half of these top-choice errors are due to confusions that are arguably inevitable in any integrated multiple-typeface classifier: *i.e.* among **0 O**, **1 l I |] J { }**, and **' '**. These engineering benchmarks were computed under the assumption that all symbols occur with equal probability. In practice, of course, symbol distribution is non-uniform, often allowing higher accuracies. For example, 99.7% per cent or better is usually possible on 400 pixel/inch images of original-copy English text printed at $\geq$ 8 point, after automatic correction using a dictionary and punctuation rules (see Section 5).

In these experiments, the 100 typefaces were selected to cover as much as possible of 20th C. American printed books and magazines and typewritten office documents. The resulting variety of typefaces gave us an opportunity to measure certain properties of the recognition technology that are often discussed qualitatively but seldom quantitatively. One of these is *generalizing power*, the ability of a system to extrapolate automatically from a design set, and so to perform well on classes of images not explicitly trained on. We observed that, in this domain, the recognition technology generalizes by about a factor of four: that is, training on 1/3.9 of the typeface-symbols is sufficient to achieve high accuracy on all of them.

Another property of classifiers, interesting on theoretical grounds, is *compression*: the conciseness with which the universe of images to be recognized can be represented. In our classifier, sets of similar typeface-symbols are represented by a single statistical record, called a *class* (it contains first-order statistics of a few scalar features, and a vector of log-ratios of Bayesian *a priori* binary-feature probabilities). The number of these classes is a good measure of the resources consumed by recognition: the space requirements of the classifier are linear in this number, and runtime is slightly sublinear. There is always at least one class per symbol, but more than one class may be needed to represent all typefaces. Compression, then, can be computed as the ratio of typeface-symbols to classes: a compression of one meaning that each typeface-symbol requires its own class. Our classifier has a compression of about 16: that is, averaged over the ASCII set, 15.9 different typefaces are represented by one class. This is achieved in spite of variations due to imaging defects. Such high compression has practical implications: it suggests that classifiers for an even larger number of typefaces can be constructed with only a fractional increase in computing resources.

The technical basis for this strong generalization and compression lies, we believe, in the procedure we use to infer classifiers. Many published decision-theoretic recognition methods require, as a manual first step, the exhaustive specification of a set of features (often real components of a fixed-length vector); later, during the training phase, the feature set may be pruned or transformed automatically for improved results. An interesting aspect of our method is that the feature set need not be specified in advance: instead, only a handful of primitive shape types — edges, holes, convex and concave boundary arcs, *etc* — are provided, in the form of algorithms to extract them from boundary lists and moments of area. Any given set of extracted primitive shapes is converted into a feature vector, suitable for a statistical procedure, by means of a mapping which is itself constructed during automatic analysis of the distribution of primitives in the training set.

This approach has the virtue of consistency with our overall policy of manually specifying as little as possible in advance. Our hope has been that, for this reason, the inference algorithm would be able to adapt to diverse symbol sets with little or no manual intervention. The shape primitives were originally selected by trial-and-error during experiments on the Latin alphabet. When we experimented on the very different Tibetan U-chen writing system (Figure 3), we were encouraged by high accuracy (97%) on a large alphabet (438 distinct symbols).

Figure 3: A passage of text from a dictionary for Tibetan, a language with over 400 distinct symbols. We were able to adapt the page reader to read the entire book with an accuracy of 97%, with two weeks' effort, including semi-automatic training.

This was achieved without adding new shape primitives — in fact, with no changes whatever in algorithms or tuning parameters. Since then, we have built classifiers for Greek and Japanese *kana*, with good results (>99.5% correct top choice on single typefaces under moderate distortion), again with no changes to the inference procedure. In another exercise, a classifier was built for 189 mathematical symbols (in roman and italic styles of a single typeface family), and used successfully in research into reading of typeset equations [Cho89].

We are also pursuing an alternative classification technology based on neural nets [LBD90], which devotes greater computational resources to the early stages of shape extraction. Although this requires special hardware (*e.g.* digital signal processors or custom VLSI convolvers), early experiments [BGJ88] suggest that it will be significantly more immune to image defects.

The input to the shape classifier is an image of an isolated character, without size or baseline context; the output is a list of interpretations, in decreasing order of matching *confidence scores*. These scores are logarithms of *a posteriori* probabilities of class membership computed by a Bayesian classifier. Their ordering is good, but their class-conditional variance is large and so their absolute magnitude does not reliably indicate whether or not a symbol is malformed (due to touching symbols, for example). We have found ways to normalize the scores by reference to ancillary shape metrics such as Hamming distance (to representative binary vectors) and Mahalanobis distance (to mean vectors of scalar properties). Normalized confidence scores, combining good ordering and low variance, are used to drive later stages.

The first use of confidence scores is the inference of text size and baseline within lines of text — the *local geometric context* of symbols required in many languages (*e.g.* to distinguish upper from lower case in the Latin alphabet). Each alternative symbol interpretation implies a text size (estimated from per-class statistics collected during training): the median of these sizes, weighted by confidence, is selected as the line's dominant text size. This size is then used to prune the interpretations. It is as though a distinct classifier, sensitive to both shape and size, had been used, but the incremental computational cost is negligible. In an analogous way, baseline is selected; this works equally well with no change for writing systems with a top-line convention, such as Tibetan. Thus, by exploiting confidence scores provided by the symbol recognition subsystem, we have avoided building a separate system of symbol-set-dependent rules (*e.g.* ‘‘typographical rules’’) merely to infer local geometric context.

The resegmentation of touching, fragmented, and spurious (dirt) symbols is triggered by low confidence scores. Within each affected word, we execute a branch-and-bound search [KPB87] of alternative splittings and merges of symbols, pruned by word-confidence scores derived from symbol confidence. Only for exceptional cases that remain ambiguous by shape, such as the pair m and rn, are language-specific rules needed to trigger resegmentation. However, the heuristics which propose places to cut have evolved as a result of experience with the Latin alphabet, and may require extension for other symbol sets. A similar method has been described in detail by Tsujimoto & Asada [TA91]. The result of this processing is that some words have alternative resegmentations in addition to the alternative interpretations of their constituent symbols. This double-level lattice of alternatives, ordered by shape confidence scores, is passed to the linguistic contextual analysis stage.

5. Linguistic & Semantic Context

Linguistic context, such as provided by dictionaries, punctuation rules, and other lexical constraints, is often effective in resolving residual ambiguities of shape caused by badly-designed symbol sets, overlapping typeface variations, and distortions due to imaging defects. We exploit these in a data-directed manner by filtering the lattice of word interpretations.

We have experimented principally with veto filters, which merely accept or reject a word interpretation. These include all-alphabetic or all-numeric rules (quite effective on Latin languages), punctuation prefix/suffix patterns, dictionary and word-list lookup, and regular expression patterns. Some common filters are built in, but we also permit the user to provide arbitrary programs, executed as UNIX processes and attached to the OCR program by pipes.

The contextual-analysis control algorithm works as follows: alternative word interpretations are generated (from the lattice that is output by symbol recognition), in descending order of word confidence scores derived from the symbols' scores. The list is run through each filter in turn: if a filter accepts no interpretation, the list is not modified (in case the filter is not relevant); but if any interpretation is accepted, then all rejected interpretations are pruned. The user may control at run-time how far down in word-confidence order the filters look, to trade off runtime for accuracy.

Using such a method, we adapted the system to exploit a pre-existing tool for spell-checking Swedish words, based on a 65,000-word lexicon. Constructing a classifier to include the non-ASCII characters aoAO was an easy exercise using the tools described in Section 4. In four days elapsed time, starting from scratch, we read a 262-page Swedish book with an accuracy of approximately 98.5%.

These contextual-analysis methods are all *data-directed*: they merely select among alternatives generated by shape recognition. In some applications, *model-directed* analysis may be required: these are able to supply missing alternatives by appeal to statistical, linguistic, or semantic models. In an experiment of this kind, on several volumes of a chess encyclopedia [BK90] (see Figure 4), we obtained dramatically improved results.

Figure 4: A passage of text from an encyclopedia of chess games. The syntax and semantics of this special text yielded to automatic analysis, permitting a final accuracy of 99.995% characters correct in spite of poor print quality.

The initial error rate of 0.5%, achieved by symbol recognition, was reduced to 0.005% by contextual analysis. This included syntactic analysis tailored to the books' chess notation, and semantic analysis based on the rules of chess. Compared to natural languages, short runs of chess moves imply relatively small sets of alternative interpretations, making possible an efficient model-driven analysis. It is interesting to note that this record-breaking accuracy was achieved without backtracking to the symbol recognition stage.

6. Output Encoding

Within the page reader system, symbols are identified by *grapheme names* — short alphanumeric strings — making up a name space potentially large enough to accommodate symbols from many languages without conflict. Thus it is straightforward to combine the symbol sets of multiple languages in a single reader: we have already accomplished this for the Latin, Greek, and Cyrillic alphabets.

When adapting the system to a new language, it is necessary to provide translations to and from external symbol codes (or transliterations). In particular, such mappings are required to translate the labels of images in training sets, encode word interpretations for linguistic contextual analysis, and, finally, prepare the output for external files, printing, or graphical display. In the English-speaking world, the ASCII code is often adequate for all these purposes; but for other languages, different coding standards may apply at various stages of processing [Cle88]. In our system, these choices may be specified by the user in the form of lookup tables.

7. Tools and Data Structures

The family of programs making up the page reader system are all written in the C programming language and run under the UNIX® operating system on a variety of machines. The ‘‘on-line’’ subsystem consists of the stages of computation discussed above. The ‘‘off-line’’ subsystem includes tools for training and testing classifiers, for the pseudo-random generation of training sets, and for graphical display and editing of document images and intermediate results of processing.

All components of the on-line and off-line subsystems are unified by a common data structure that is persistent and machine-independent. This data-structure permits the description of a document image as a full hierarchy of pages of blocks of text lines of words of symbols, as well as many incomplete hierarchies suited to early stages of analysis.

8. Summary

We have described the architecture of a page reader versatile enough to be readily adapted to multiple-typeface English text, and single-typeface Greek, Tibetan, Swedish, chess notation, and typeset mathematics. Plans for future work include experiments on other writing systems, including Kanji, Cyrillic, and Hangul. We are particularly interested in testing the range of applicability of the contextual analysis control strategy, and in exploring model-driven methods further.

We feel there is still much to be learned about good engineering strategies for combining image analysis with computational linguistics. This research may someday lead to reading machines competent in multiple languages simultaneously.

9. Acknowledgements

The page reader is a lineal descendant of one [KPB87] built by Theo Pavlidis, Simon Kahan, and the present author. Suzanne Lam helped design the off-line training subsystem. Ken Thompson inspired the chess book experiments, Mats Rooth collaborated on Swedish, and David Ittner on Japanese. Reid Fossey carried out the Tibetan experiments, and researched linguistic and typographical background. Susan Jones, Steve Fortune, and David Ittner have contributed algorithms for geometric layout analysis, and John Hobby has contributed boundary analysis algorithms. Phil Chou thoroughly exercised the classifier technology in the course of his equation-reading research. Larry Jackel and his technical staff have led the exploration of neural nets and special hardware. Tim Thompson, Chris Macey, and Ofer Matan have helped make the system machine-independent. Lorinda Cherry provided machine-legible typeface descriptions, and Tom Killian a versatile bitmap printer. I am grateful for stimulating discussions with Reid Fossey, David Ittner, Larry Jackel, George Nagy, Sargur Srihari, Theo Pavlidis, Larry Spitz, and Kazuhiko Yamamoto.

10. References

[Bai87] Baird, H. S., ‘‘The Skew Angle of Printed Documents,’’ *Proceedings, 1987 Conference of the Society of Photographic Scientists and Engineers*, Rochester, New York, May 20-21, 1987.

[Bai88a] Baird, H. S., ‘‘Feature Identification for Hybrid Structural/Statistical Pattern Classification,’’ *Computer Vision, Graphics, & Image Processing* **42**, 1988, pp. 318-333.

[Bai88b] Baird, H. S., ‘‘Global-to-Local Layout Analysis,’’ *Proceedings, IAPR Workshop on Syntactic and Structural Pattern Recognition*, Pont-à-Mousson, France, 12-14 September, 1988.

[Bai90a] Baird, H. S., ‘‘Document Image Defect Models,’’ *Proceedings, IAPR 1990 Workshop on SSPR*, Murray Hill, NJ, June 13-15, 1990.

[Bai90b] Baird, H. S., ‘‘Anatomy of a Page Reader,’’ *Proceedings, IAPR Workshop on Machine Vision Applications*, November 28-30, 1990, Tokyo, Japan.

[BF91] Baird, H. S., and R. Fossey, ‘‘A 100-Font Classifier,’’ *Proceedings, 1st Int’l Conf. on Document Analysis and Recognition*, St-Malo, France, 30 September - 2 October, 1991.

[BGJ88] Baird, H. S., H. P. Graf, L. D. Jackel, and W. E. Hubbard, ‘‘A VLSI Architecture for Binary Image Classification,’’ *Proceedings, COST13 Workshop on Pixels to Features*, Bonas, France, 22-27 August, 1988.

[BJF90] Baird, H. S., S. E. Jones, and S. J. Fortune, ‘‘Image Segmentation using Shape-Directed Covers,’’ *Proceedings, IAPR 10th Int’l Conf. on Pattern Recognition*, Atlantic City, NJ, 17-21 June, 1990.

[BK90] Baird, H. S., and K. Thompson, ‘‘Reading Chess,’’ *IEEE Trans. PAMI*, Vol. **PAMI-12**, No. 6, June 1990, pp. 552-559.

[BL87] Baird, H. S., and S. Lam, ‘‘Performance Testing of Mixed-Font, Variable-Size Character Recognizers,’’ *Proceedings, 5th Scandinavian Conference on Image Analysis*, Stockholm, SWEDEN, June 2-5, 1987.

[CC84] CCITT Recommendations T.4 and T.6, on facsimile coding schemes and coding control functions for Group 3 and Group 4 facsimile apparatus (drafted at Malaga-Torremolinos, 1984).

[Cho89] Chou, P., ‘‘Recognition of Equations using a Two-Dimensional Context-Free Grammar,’’ *Proc., SPIE Visual Comm. and Image Proc. IV*, November 1989.

[Cle88] Clews, J., *Language Automation Worldwide: the Development of Character Set Standards*, R&D Report No. 5962, The British Library, 1988.

[CMS82] *The Chicago Manual of Style*, The University of Chicago Press, Chicago, Michigan, 1982.

[KPB87] Kahan, S., T. Pavlidis, and H. S. Baird, ‘‘On the Recognition of Printed Characters of any Font or Size,’’ *IEEE Trans. PAMI*, Vol. **PAMI-9**, No. 2, March, 1987.

[LBD90] LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and H. S. Baird, ‘‘Constrained Neural Network for Unconstrained Handwritten Digit Recognition,’’ *Proceedings, Int’l Workshop on Frontiers in Handwriting Recognition*, Montreal, 2-3 April, 1990.

[Pav82] T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, Maryland, 1982.

[TA91] Tsujimoto, S., and H. Asada, ‘‘Resolving Ambiguity in Segmenting Touching Characters,’’ in *Structured Document Image Analysis*, H. S. Baird, H. Bunke, and K. Yamamoto (eds), Springer-Verlag, [to appear] 1992.

[WS88] Wilcox, L. D., and A. L. Spitz, ‘‘Document Recognition and Representation,’’ *Proc., Int’l Conf. on Electronic Publishing, Document Manipulation, and Typography*, Nice, France, April 1988.

Figure 1: This *Computer Journal* page of 5590 black connected components generates 15106 maximal empty rectangles, of which the first 151 (1%) in shape-directed order define the cover set shown. The skew- and shear-corrected layout is illustrated on the left by the bounding rectangles of black components, and the segmentation is shown on the right as the set of white components remaining after the union of the cover set has been computed. One isolated line of text is fragmented by a spurious vertical cut (at the top of the page).

[2.0,2.5)
[1.5,2.0)
[1.0,1.5)
[0.5,1.0)
[0.0,0.5)

Figure 2: Pseudo-randomly generated defective images of a Cyrillic /LJE/ character, at 5 point size and 300 pixel/inch digitizing resolution. Each line holds images whose defect parameter vector's Mahalanobis distance lies in the range shown. Mahalanobis distance is Euclidean distance from the mean parameter vector, scaled component-wise by standard error, and is thus a measure of severity of the distortion.

Figure 3: A passage of text from a Tibetan-to-Tibetan dictionary; this language possesses over 400 distinct symbols. We were able to adapt the page reader to read the entire book with an accuracy of 97%, with two weeks' effort, including semi-automatic training.

Figure 4: A passage of text from an encyclopedia of chess games. The syntax and semantics of this special text yielded to automatic analysis, permitting a final accuracy of 99.995% characters correct in spite of poor print quality.