MVA'90 IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

ANATOMY OF A PAGE READER

Henry S. Baird

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

Abstract

An experimental machine vision system that reads pages of printed text is described, with particular attention to features which permit it to cope with a variety of languages. We have attempted to uncouple sources of variation in the input — imaging defects, page layouts, sizes of text, symbol sets, font styles, linguistic contexts, and transliteration encodings — so that each can be managed by an independent subsystem. In pursuit of this goal, new algorithms have been required. For example, page layout analysis is virtually language-independent due to the use of a global-to-local strategy; but this demanded unusually fast and accurate algorithms for skew- and shear-correction and for analysis of the structure of white space. To permit arbitrary combinations of symbol sets, font styles, and image defect distributions for specialized classifiers, we developed a hybrid structural/statistical recognition technology, trained by examples generated by a pseudo-random image defect model. Character segmentation is driven by classifier confidence scores, with the result that language-specific rules are rarely needed. Linguistic context, such as provided by dictionaries, is exploited by a series of data-directed filtering algorithms in a uniform and modular manner.

1. Introduction

This is an overview of an experimental printed-page reader that has been applied to multiple-font English text and fixed-font Swedish, Tibetan, chess notation, and mathematical equations. The demands of versatility — the ability to adapt the system to diverse writing systems with a minimum of manual effort — have strongly influenced the system's architecture. An overview of an early version of the reader appeared as [KPB87]; recent details of algorithmic components are published elsewhere (references are cited below).

Automatic page readers must cope with many kinds of variability in their input, including imaging defects, layout geometries, sizes of text, symbol sets, font styles, linguistic contexts, and transliteration codes. An overall engineering objective has been to decouple these from one another, so that each can be attacked independently. To achieve this, new algorithms have been required at several stages. For example, thanks to a global-to-local strategy, page layout analysis is virtually language-independent — but this strategy demands fast and accurate algorithms for skew- and shear-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 sample image per symbol, which may be either an ideal prototype from a machine-legible font description or an example lifted from a document image. Thus the manual effort to learn a new symbol set or font style is reduced to a minimum. The classifiers exhibit strong generalization across font style differences, 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 transliteration into an output encoding (such as ASCII) is via user-specified tables.

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 the transliteration to coded and formatted output in Section 6. The tools and data structures used are described in Section 7.

2. 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 transliteration rules. However, font styles, text sizes, and page layouts will not be specified in advance, at least in the most demanding applications. Of course it is also useful to be able to construct custom readers for higher accuracy on known symbols, fonts, 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. *transliteration*: mapping the internal representation of the analysis into an output character encoding (*e.g.* ASCII or JIS), possibly annotated by format (*e.g.* troff or SGML).

By design, each of these five stages operates as autonomously as possible. Up to the present time, we have been able to avoid feedback control paths among them. If and when feedback control proves necessary, there will be few software-engineering obstacles, since all stages use a single, common data-structure.

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 Roman, 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; it will not be discussed further. Algorithms to cope with graphics, line-drawings, and photographs also will not be discussed.

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 $\times 8$-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-distortion of the page as a whole are measured and corrected if they differ from strictly horizontal or vertical by more than 0.03 degrees. Our skew-analysis algorithm [Bai87] is one of the fastest and most accurate reported, and works without modification on a wide variety of layouts, including multiple blocks, sparse tables, variable line spacings, mixed fonts, and a wide range of point sizes. 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 process 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. 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 appears that publishers and printers in many languages, constrained by similar printing technology and legibility conventions [CMS82], use white space as a layout delimiter in similar ways. The block-finding algorithm is not yet completely implemented.

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. Columns are partitioned into lines by skew-correction followed by analysis of the horizontal projection profile. Several researchers report that it is possible to estimate properties of text lines, such as text size and baseline location, from their projection profiles, but we choose to defer this until after symbol recognition, when a more robust 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 that each symbol may undergo, 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 later simply 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.

We have chosen to explore a recognition technology that is a hybrid of structural shape analysis and Bayesian statistical classification [Bai88a]. This attempts to combine strong generalization across fonts with immunity to imaging defects, so that it would be possible to build integrated multiple-font classifiers without storing a large number of prototype symbol descriptions [BL87]. In experiments on 39 font styles of isolated images of the Roman (printable ASCII) symbol set, over the range of sizes from 8 to 14 point, at a digitizing resolution of 300 pixels/inch, we measured success rates of 99.19% top choice and 99.87% within the top 5 choices, after forgiving the arguably-inevitable confusions among **1Il|![]J**, **0O**, **`'**, and **"'**. The generalization factor achieved was over 15: that is, more than 15 distinct font styles are represented by a single prototype description within the classifier table, averaged over all symbols.

Many published decision-theoretic recognition methods require, as a manual first step, the exhaustive specification of a feature set; later, during the training phase, the set may be pruned or transformed automatically for improved results. An interesting aspect of our method is that the feature set is not 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. The set of extracted primitives is then converted into a feature vector, suitable for statistical pattern recognition, by means of a mapping which is itself constructed during automatic analysis of the occurrences and 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. We hoped that such a system would be able to adapt to diverse symbol sets with little manual intervention. The shape primitives were originally selected by trial-and-error during experiments on the Roman alphabet only. When we experimented on the very different Tibetan U-chen writing system, we were encouraged by high accuracy (95%) on a large alphabet (438 distinct symbols). This was achieved with no 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% correct top choice on single fonts under moderate distortion). In another exercise, a classifier was built for 189 mathematical symbols (in Roman and Italic styles of a single font 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.

A quantitative model of imaging defects [Bai90] which we have developed 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. 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 with a known distribution of defects. One use of the generator is to allow classifiers to be built for new symbol sets or fonts with minimal effort. For example, a classifier for two fonts 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 font description is available, training a classifier can be accomplished in less than a day.

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 of individual lines of text. 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 cost is negligible. Baseline is selected by a similar method, which works equally well for writing systems with a top-line convention, such as Tibetan. The policy of avoids the use of a separate system of symbol-set-dependent rules.

The resegmentation of touching, fragmented, and spurious (dirt) symbols is triggered by the presence of low-confidence symbols. 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 guide resegmentation. 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 font 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 Roman 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. Using such a device, we were able to adapt the system in a few hours to exploit a Swedish-language word-

list-checking program.

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 shape confidence 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 how far down in confidence order the filters may look.

These methods are all *data-directed*: they merely select among alternatives suggested by earlier shape analysis. In some applications, *model-directed* analysis may be required for good results: these are able to generate alternatives from incomplete context based on linguistic or semantic models. In an experiment of this kind, on several volumes of a chess encyclopedia [BK90], dramatically improved results were obtained.

6. Transliteration

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 possible to combine the symbol sets of multiple languages in a single reader: we have already accomplished this for the Roman and Greek alphabets.

When adapting the system to a new language, it is necessary to provide translations to and from external symbol codes (or transliterations) and OCR grapheme names. 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 encoding and transliteration standards may be needed at various stages of processing [Cle88]. In our system, these choices are encoded in user-specified 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 multi-font English text, and single-font Greek, Tibetan, Swedish, chess notation, and typeset mathematics. Plans for future work include experiments on other writing systems, possibly 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 that are simultaneously competent in multiple languages.

9. Acknowledgements

This page reader is lineal descendant of one [KPB87] built by Theo Pavlidis, Simon Kahan, and the present author. I am grateful for stimulating discussions with 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.

[Bai90] Baird, H. S., "Document Image Defect Models," *Proceedings, IAPR 1990 Workshop on SSPR*, Murray Hill, NJ, June 13-15, 1990.

[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, Intl'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.