effort. The system has been applied to more than 400 distinct table layouts, in large-scale applications on over fifty million records altogether, and with high accuracy: typically 97% to 99.98% characters correct is achieved automatically, and this is efficiently manually correctable to better than 99.99% using our GUI. The high throughput of the system is assisted by the policy of applying OCR only to fields identified as interesting by fast layout analysis. This performance is so much higher than any previously published on tables that it is tempting to assert that restriction to known predefined layouts and exploitation of field-specific context are key determinants of success.

14 Acknowledgement

The geometric page-layout software [IB93] on which we have depended was largely the creation of David J. Ittner in collaboration with the second author. An early production version of the algorithms for record identification and segmentation was due to Michele Battista in collaboration with the second author. The algorithms for elimination of line-art were developed by the third author.

References

[CF90] R. G. Casey and D. R. Ferguson, "Intelligent Forms Processing," *IBM Systems Journal*, vol. 29, no. 3, pp. 435–450, 1990.

[CK93] S. Chandran and R. Kasturi, "Structural Recognition of Tabulated Data," *Proc., IAPR 2nd Int'l Conf. on Document Analysis & Recognition*, pp. 516–519, Tsukuba Science City, Japan, October, 1993.

[DD96] F. Dubiel and A. Dengel, "FormClas – A System for OCR Free Identification of Forms," *Proc., IAPR 1996 Workshop on Document Analysis Systems*, Malvern, PA, pp. 40–61, October 14–16, 1996.

[GK95] E. Green and M. Krishnamoorthy, "Model-Based Analysis of Printed Tables," *Proc., IAPR 3rd Int'l Conf. on Document Analysis & Recognition*, Montreal, Canada, pp. 214–217, August 14–16, 1995.

[HD95] O. Hori and D. S. Doermann, "Robust Table-form Structure Analysis Based on Box-Driven Reasoning," *Proc., IAPR 3rd Int'l Conf. on Document Analysis & Recognition*, Montreal, Canada, pp. 218–221, August 14–16, 1995.

[Ishi95] Y. Ishitani, "Model Matching Based on Association Graph for Form Image Understanding," *Proc., IAPR 3rd Int'l Conf. on Document Analysis & Recognition*, Montreal, Canada, pp. 287–292, August 14–16, 1995.

[Iton93] K. Itonori, "Table Structure Recognition based on Textblock Arrangement and Ruled Line Positions," *Proc., IAPR 2nd Int'l Conf. on Document Analysis & Recognition*, pp. 765–768, Tsukuba Science City, Japan, October, 1993.

[IB93] D. J. Ittner and H. S. Baird, "Language-Free Layout Analysis," *Proc., IAPR 2nd Int'l Conf. on Document Analysis & Recognition*, pp. 336–340, Tsukuba Science City, Japan, October, 1993.

[KA90] H. Kojima and T. Akiyama, "Table Recognition for Automated Document Entry System," *High-Speed Inspection Architectures, Barcoding, and Character Recognition*, SPIE, vol. 1384, pp. 285–292, 1990.

[LJS93] S. W. Lam, L. Javanbakht, and S. N. Srihari, "Anatomy of a Form Reader," *Proc., IAPR 2nd Int'l Conf. on Document Analysis & Recognition*, pp. 506–509, Tsukuba Science City, Japan, October, 1993.

[LV92] A. Laurentini and P. Viada, "Identifying and Understanding Tabular Material in Compound Documents," *Proc., IAPR 11th Int'l Conf. on Pattern Recognition*, pp. 405–409, The Hague, The Netherlands, August 30 - September 3, 1992.

[PTG91] A. Pizano, M. I. Tan, and N. Gambo, "A Business Form Recognition System," *Proc., Comput. Software Applic. Conf.*, pp. 626–632, 1991.

[Stab95] H. R. Stabler, "Experiences with High-Volume, High-Accuracy Document Capture," in A. L. Spitz and A. Dengel (Eds.), *IAPR 1994 Workshop on Document Analysis Systems*, Vol. 14, Series in Machine Perception and AI, World Scientific, pp. 38–51, Singapore, 1995.

[SBB92] E. Saekinger, B. Boser, J. Bromley, *et al*, "Applications of ANNA Neural Net Chip to High-Speed Character Recognition," *IEEE Transactions on Neural Networks*, Vol. 3, No. 3, pp. 498–505, May 1992.

[TFP92] S. L. Taylor, R. Fritzson, and J. A. Pastor, "Extraction of Data from Preprinted Forms," *Machine Vision and Applications*, vol. 5, pp. 211–222, 1992.

[TS94] Y. T. Yang and C. Y. Suen, "Document Structures: A Survey," in H. Bunke, P. S. P. Wang, and H. S. Baird (Eds.), *Document Image Analysis*, World Scientific, vol. 16 in Series in Machine Perception and AI, Singapore, pp. 85–115, 1994.

[WLS95] T. Watanabe, Q. Luo, and N. Sugie, "Layout Recognition of Multi-Kinds of Table-Form Documents," *IEEE Trans. PAMI*, vol. 17, no. 4, pp. 423–445, 1995.

ertheless the result fails sanity checks (based on minimum and maximum number of characters, and regular expressions), it will be resegmented up to three times with different segmentation parameters.

The neural-net technology we use is a modified LeNet2a with both convolutional and fully connected layers [SBB92]. High-volume tables are provided with nets specially trained on their typeface(s), while lower volume tables use a generic multiple-typeface net. The nets are trained on both real and synthetic image samples.

It should be clear that our policy of detailed predefinition not only permits a new table to be targeted rapidly by clerical staff, using system defaults, but it allows a large number of custom optimizations to be introduced on a case by case basis by system engineers, as unusual speed or accuracy demands arise.

9 Contextual Analysis

We have described how the alphabet of a field is used to restrict classifiers, and how the minimum and maximum width of a field is used to guide resegmentation. In addition, the regular expression and/or lexicon associated with the field are used to 'promote' legal values for a field to the top of a list of interpretations.

10 Implementation Issues

The table reader system runs in various UNIX environments on several hardware platforms including Sun Sparcstations and PC-compatible systems. It requires no specialized hardware. It accepts document images from bi-level, grey-level, or dropout-color scanners, and has been adapted to read a variety of digitizing resolutions and paper sizes.

11 Applications

The table reader system has been used in a production environment for over four years, reading bill manifests. The total number of distinct table layouts to which it has been applied exceeds 400. At the time of writing, the definitions of 313 tables reside in the system; over the last few years, over 100 other tables have been defined and used, but not kept since their format wasn't seen often enough to justify the effort of keeping track of them. We estimate that, during the years 1995-1997, it was used to read 53.5 million records (this, not characters or pages, is the statistic of greatest interest to our users).

CPU time to process a page on a Sparc Ultra 1 model 170 ranges between 15 seconds to 1 minute of CPU time; thirty seconds is typical. The system achieves accuracy in the range 97% to 99.98% characters correct, fully automatically. When measured on their training set, which is a mixture of synthetic and real images of characters, the accuracy of the neural nets is typically 99.7%. In practice, accuracy varies: for example, on simple fixed-pitch tables, customer-trained neural nets have achieved accuracies better than 99.98% characters correct, whereas on proportional-pitch tables with line-art that touches characters and a dithered grey background, the generic multiple-font classifier may require up to 3% of the characters to be examined manually. A semi-manual correction stage (not described here) typically raises this to above 99.99%. Another measure of efficiency that is particularly valuable to our customers is the product of (a) the total elapsed time to process a batch of records and (b) the number of clerks required to perform the final correction, to the desired accuracy. Our customers compared our system to another vendors' system in three trials during which our system's efficiency was superior by factors of 54, 80, and 550.

12 Future Work

We are now introducing an automatic format verification step that will check relationships between the contents of selected fields. We do not know how to instruct our operators to choose 'binarization' thresholds for their document scanners that are ideally matched to particular batches of table documents, so our software must sometimes cope with barely usable bi-level images. We would very much like to be able to 'learn' table formats semi-automatically from examples, perhaps along the lines of [DD96], which does not depend on line-art boundaries. We would like to be able to identify tables automatically from among a predefined set, possibly following [KA90] and [WLS95]. Scanning resolution is still too low, relative to the small size of the type often encountered: so the character images are daunting small.

13 Conclusions

We have described a system for reading machine-printed tables. Unlike most prior work on forms and table analysis, the system does not depend on guidance from line-art or fiducial marks. The most significant technical advances in this work appear to be the design of algorithms for identifying/segmenting record lines in a known layout style, and the integration of these algorithms with an ergonomically refined graphical user interface for defining new layouts, and specifying contextual rules for each field. As a result, it is capable of being manually retargeted to new table layouts with a minimum of operator skill and manual

statements, government applications, inventory lists, etc. We attempt to suppress these using 'mid-pass' filters (for lack of space, we don't describe them here).

4.2 Elimination of Line-Art

Not only does our system make no use of line-art, it goes to some trouble to ignore it. Some tables insert line-art between almost every line of the table. Figure 3 shows an example.

Figure 3: A table with many horizontal lines, often touching the characters.

Due to tight line-spacing, coarse digitizing resolution, and poor quality images, line-art close to the records can touch characters.

We attempt to detect and remove horizontal lines. This is complicated by the fact that such lines may be slightly bent due to local skew or optical deformations. Furthermore, the slope of lines may not be constant everywhere: sometimes, the slopes 'drift' down the page. We have developed a complex heuristic for handling many of these cases, which for lack of space we do not describe here. Any horizontal line-art found is carefully removed, leaving the characters behind.

5 Locating Table(s) in the Page

The next major stage locates the region(s) in which the table(s) is(are) placed. The page is deskewed using 'pseudorotation.' Text lines are isolated; if two or more tables are side by side, these lines may span more than one table and therefore contain more than one record. The number and location of columns are estimated, using the total length of the 'ink templates' known to make up records in this table.

Now, a search is performed for the best match of the 'ink template' for the records to the actually occurring text lines. This is accomplished in several steps. First, each text line is projected vertically onto a horizontal line, where the presence of black pixels is recorded by +1 and their absence by -1. Then the ink template for the record to be found is encoded, thus: +1, 0, or -1 for each pixel within character positions whose ink template value is 'usually', 'sometimes', or 'never', respectively. Now, for each possible horizontal shift of the record with respect to the text line, the inner product of these two projections is computed, giving a correlation score for each shift position: the highest score indicates the best match. This is an elementary variant of 'template matching': mismatches are penalized at any character position recorded as either 'usually' or 'never'; but at 'sometimes' positions neither the presence nor absence of a character is penalized. The match scores, normalized by dividing by the number of of +1 values in the record template, are compared to a fixed threshold value. If the maximum match score falls below threshold, the text line is judged not to match the record template.

The set of text lines that match the record templates are kept, and all other artwork in the image is discarded. For speed, this matching step is performed at a digitization resolution coarser than in the input image. The result, at this stage of processing, is zero, one, or more candidate record text lines have been located. Note that more than one record-line ink template may be used, the record line with the highest match score being chosen.

6 Identifying Records

Operating on the candidate record lines found above, a similar search is now conducted, but at full resolution. Its purpose is, for each text line, to refine the location of the best match. It is possible for text lines to be rejected (to fail to match) at this stage as well: if so, their artwork is also discarded. By examining the horizontal shifts of the best matches, it is possible to estimate the start and end positions of the table. If there is enough space on the page for more than one table, they are searched for using the same method.

7 Locating and Labeling Fields

Now, within each matched text line, the physical structure of the ink template is used to locate the start and end of each field. Fields flagged 'not read' are now discarded. The output of this stage is a set of records containing fields containing segmented character images, and each field's logical label is known. Note that this has been achieved based on the presence or absence of characters only. It does not depend on the presence of line-art, fiducial marks, or any special 'form features'; nor does it require OCR.

8 Symbol Recognition

For each field, the characters are read using neural net classifiers. A field's context may imply the use of a special neural net trained off-line by the systems engineer. Even if a special classifier is not invoked, the alphabet (symbol set) used in classification will be restricted to only those characters that can occur.

If the field contains proportional-pitch text, a classifier driven resegmentation scheme is invoked. If nev-

guide, for each record format (more than one may occur in a table), the user graphically selects each field, naming it and indicating its width and typical 'ink profile.' The contextual constraints for each field are entered.

Within a batch of images containing tables of the same layout style, each page image is processed independently of the rest. All pages are manually scanned so that the text is upright. Each page undergoes a preliminary layout analysis which is independent of layout style: extraction of connected components, removal of horizontal line-art, correction of skew, and isolation of text lines. Fixed pitch is detected and used to guide segmentation of text lines into characters using the method of [IB93]. Then the 'ink template' models of the records are applied: first, at a coarse resolution, to identify probable record lines and thus — most importantly — to discard almost all non-record text, noise, etc. Then, operating only on the probable record lines, the process is repeated more carefully at full resolution, and including the segmentation of record lines into fields.

From this stage forward, each field-image specified by the user as important is processed independently, OCR is attempted using neural nets that may be tailored to the layout style or field. Finally, field-specific contextual constraints including regular expressions and lexica are applied, and the results for each record are written out in a format specified by the user. Each of these steps is explained in the following sections.

3 Table Definition via Graphical User Interface

A table to be defined is assigned a name unique within the system, and a little information about it is typed in, *e.g.* whether the dominant typeface is fixed- or proportional pitch (numeric data is often fixed-pitch, but alphabetic fields (*e.g.* 'AM' and 'PM') may be printed in proportional pitch).

The user selects a sub-image of the table to guide the specification. The user selects one of the text lines, by graphically indicating the start and end of each field within it. To each field, the user assigns these properties: (a) a name (unique within the table); (b) a flag: 'read' or 'not read'; (c) the 'ink profile' (see below); (d) the minimum and maximum number of characters that can occur; (e) the alphabet that can occur; (f) the neural net to use for recognition; (g) a regular expression and/or a list of allowable words ('lexicon'); and (h) output formatting (for down-stream processing). A field's 'ink profile' is specified as follows: for each character position, the user indicates how often a printable character is expected to be seen there, expressed as one of: 'usually', 'sometimes', or 'never'. Figure 2 illustrates this.

Figure 2: Two examples of the specification of 'ink profiles' for records.

The GUI is designed to make all this as intuitive and foolproof as possible. The user is expected to be able to count, but is not required to deal with real numbers, measure or estimate quantities, or perform arithmetic. Where the system needs such information, it either measures it itself or infers it from the user's graphical input. The coarse quantization of character-occurrence probabilities used in 'ink profile's was chosen carefully to be easy for a mathematically unsophisticated user to estimate from visual examination of a few examples. The user selects regular expressions from a set defined in advance by an engineer — only a few dozen of these have been needed. As a result, clerical staff with some data-entry training can use the GUI effectively and routinely. Theses clerks' educational background includes high school but no systematic training in computer use, programming, or engineering.

Immediately following the definition of the table format, the user can run it repeatedly on sample pages, looking for errors and changing the format and, rarely, certain engineering parameters (such as a threshold for accepting record-line matches) until performance is satisfactory.

An experienced clerical worker can complete the entire process of entering and experimentally checking a new table layout in 10-15 minutes, on average.

If a case arises that cannot be readily handled through the GUI, Bell Labs engineers intervene to solve the problem algorithmically or by extending the GIU. In recent months, this has occurred no more often than once in every twenty layouts.

4 Preliminary Layout Analysis

When the system reads a batch of pages containing tables in a known layout, each page undergoes the following analysis (described in [IB93]). All black 8-connected components ('blobs') are extracted and skew is corrected.

4.1 Background Suppression

Ornate background textures, watermarks, images, and half-tone shading can cause problems: these occur on academic standardized test forms, credit card

Figure 1: Example of a horizontal textual table layout. It contains two header lines, eight record lines, and one trailer line. Each record contains ten fixed-width fields that align vertically. Each column of fields contains data of the same type. Fields may be left-justified, right-justified, or fixed-width. Note that line-art is not used to delimit the fields. (Some characters are blacked-out to respect privacy.)

in a form is usually fixed. However, in tables the number of record lines can vary. Often more than one table occurs on a single page — above or alongside one another — but this is rare in forms. Perhaps more often in tables than in forms, the textual header and trailer sections are not separated from the records by line-art.

The row-and-column structure of horizontal tables suggests an analogy with relational data bases. We can view each column as a data type, and each row as a record describing an element of a relation over these data types, within a certain schema. It is useful to distinguish between the 'physical' structure of an image of a record (a sequence of fixed-width text fields) and the 'logical' structure of the record's data (a set of data-items labeled with field names). In these terms, the purpose of the table reader is to locate physical records within an image of the table (ignoring all ancillary text), segment them into physical fields, label each field with its data type, read the contents of each field, and report the results as a set of records in a relational data base of known schema.

The literature on automatic analysis of forms (in our sense) is large: [CF90], [KA90], [PTG91], [TFP92], and [LJS93] are representative early examples. By contrast, there are only a handful of papers on tables.

Laurentini and Viada [LV92] describe an algorithm for identifying the presence of tables in document images (with or without line-art), and for inferring table structure which had not been previously specified. They report experiments on twenty sample images.

Chandran and Kasturi [CK93] describe a system for analyzing 'tabulated data' lacking a complete set of line-art boundaries ('demarcation lines') separating all fields. However, enough line-art must be seen to judge the extent of the table, and line-art must separate titles from record lines. Itonori [Iton93] describe a system with similar goals that can tolerate the absence of some, but not all, line-art. Both of these systems analyze projection profiles of the entire table (or, sub-tables) to infer logical relationships among fields.

There is a brief allusion in [Stab95] to a system for semi-automatically identifying lines in tables as header, title, records, etc, and then splitting the record lines into fields ('columns'). No details of the algorithms or applications are given.

In the applications we have seen so far, incoming documents are routinely manually sorted into batches containing a single table layout style. For this reason, we have not yet attempted to recognize table layout styles automatically. Further, we have required that the layout style be predefined by the user in considerable detail (described below). Thus, unlike the systems mentioned above, ours does not attempt to analyze unknown layouts. However, most other systems rely on the presence of at least some line-art, or fiducial marks, to guide their analysis, while ours does not.

In what appears to be an unprecedented systems design, we have integrated our algorithms for identifying and segmenting record lines in a known layout style with an ergonomically refined graphical user interface (GUI) for defining new layouts. As a result, our system is capable of being manually retargeted to a wide variety of new table layouts with a minimum of effort by operators with only 'data-entry' skills. In this manner, the system has been applied to more than 400 distinct table layouts in applications comprising over fifty million records. Large scale tests have shown that the system fully automatically achieves 97% to 99.98% characters correct. The GUI supports manual correction (not described further here), which typically yields a semi-automatic accuracy of greater than 99.99%.

The rest of this paper describes the system architecture (Section 2), definition of new tables (Section 3), preliminary layout analysis (Section 4), coping with multiple tables per page (Section 5), identification of records (Section 6), logical segmentation of records into fields (Section 7), symbol recognition (Section 8), contextual analysis (Section 9), implementation issues (Section 10), applications (Section 11), future work (Section 12), and conclusions (Section 13).

2 System Architecture

Before attempting to read pages in a new table layout style, the layout style must be named and manually defined using a GUI. Using a sample image as a

A Retargetable Table Reader

John H. Shamilian, Henry S. Baird, and Thomas L. Wood

Bell Laboratories
Lucent Technologies, Inc.
Crawfords Corner Rd, Room 2F–217
Holmdel, NJ 07733–1988 USA

Abstract

We describe the architecture of a system for reading machine-printed documents in known predefined tabular-data layout styles. In these tables, textual data are presented in record lines made up of fixed-width fields. Tables often do not rely on line-art (ruled lines) to delimit fields, and in this way differ crucially from fixed forms. Our system performs these steps: copes with multiple tables per page; identifies records within tables; segments records into fields; and recognizes characters within fields, constrained by field-specific contextual knowledge. Obstacles to good performance on tables include small print, tight line-spacing, poor-quality text (such as photocopies), and line-art or background patterns that touch the text. Precise skew-correction and pitch-estimation, and high-performance OCR using neural nets proved crucial in overcoming these obstacles. The most significant technical advances in this work appear to be algorithms for identifying and segmenting records with known layout, and integration of these algorithms with a graphical user interface (GUI) for defining new layouts. This GUI has been ergonomically designed to make efficient and intuitive use of exemplary images, so that the skill and manual effort required to retarget the system to new table layouts are held to a minimum. The system has been applied in this way to more than 400 distinct tabular layouts. During the last three years the system has read over fifty million records with high accuracy.

1 Introduction

We describe an easily retargetable 'table reader' technology. By tables we mean a class of page layouts in which the data are presented in 'record' text lines made up of fixed-width fields containing characters. Text in tables is usually printed in fixed-pitch typefaces and the records are aligned so that the fields 'stack' in columns. Tabular data occur frequently in machine-printed documents, bills, and manifests supporting many business sectors including telecommunications, health care, finance, insurance, government, and public utilities.

The research literature does not consistently distinguish between "tables" and "forms": for example, [TS94, WLS95, GK95, HD95, Ishi95] use the terms 'form,' 'table,' and 'table form' almost interchangeably. We propose that it is useful to distinguish between them, as follows:

- in "forms," the fields are delimited by line-art boundaries generally describing rectangular boxes enclosing the fields; whereas
- in "tables," explicitly printed field boundaries are often omitted, their function being served by the readily apparent columns of text formed by the vertical alignment of fixed-width fields making up the record lines.

Certainly, both tables and forms, in this sense, describe sets of fields each of which holds data — often, short strings of text — of a known type, with narrowly constrained format and semantics. But the structure of tables is often simpler than forms: tables are almost always laid out as a 2-dimensional row-and-column matrix of fields. Laurentini and Viada, in their thoughtful discussion [LV92] of 'tabular material' in documents, suggest that tables are a class of forms whose structure is so obvious that line-art is superfluous. They distinguish between *horizontal* and *vertical* tables: in a horizontal table, each column contains data of the same type; in vertical tables, each row contains data of the same type. Horizontal tables are by far the more common: we do not discuss vertical tables further here. An example of a horizontal table is shown in Figure 1.

Within each record line of a table, the number of fields is typically fixed. Similarly, the number of fields