Intelligent Forms Processing

by R. G. Casey
D. R. Ferguson

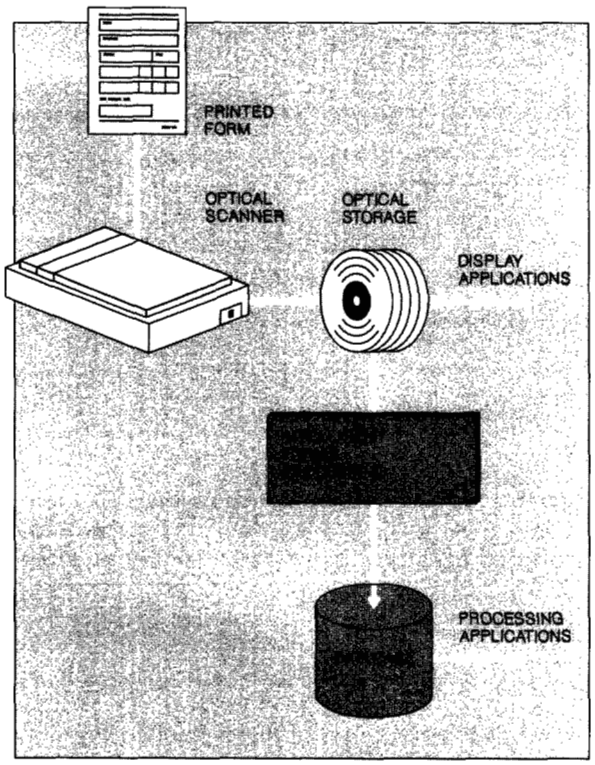
The automatic reading of optically scanned forms consists of two major components: extraction of the data image from the form and interpretation of the image as coded alphanumerics. The second component is also known as optical character recognition, or OCR. We have implemented a method for entry of a wide variety of forms that contain machine-printed data and that are often produced in business environments. The function, called Intelligent Forms Processing (IFP), accepts conventional forms that call for information to be printed in designated blank areas, but in which the information may exceed boundaries due to poor registration during printing. The human eye easily accommodates data that impinge on form boundaries or on background text; however, the same powers of discrimination applied to machine processing pose a technical challenge. The IFP system uses a setup phase to create a model of each form that is to be read. Scanned forms containing data are compared against the matching form model. Special algorithms are employed to extract data fields while removing background printing (e.g., form lines) intersecting the data. The extracted data images are interpreted by an OCR process that reads typical monospace fonts. New fonts may be added easily in a separate design mode. If the data are alphabetic, a lexicon may be assembled to define the possible entries.

A form is a conventional means for recording data, but it is seldom the final repository for data. Inevitably, at least part of the information represented on forms is transferred elsewhere in order to be recalled for later reference. Before the advent of computers, a bookkeeper's job in any industry consisted largely of filling in the columns of a ledger volume with data from transaction slips. In the computer age, the secondary destination is often a database system.

The increased use of computers has brought a continually decreasing cost for maintaining and using large volumes of data. Where the information is not generated by computer, or otherwise not accessible electronically, the labor-intensive task of feeding data into a processing system has come to account for a greater percentage of the overall cost. Eventually, perhaps, most data now entered on paper forms will be entered into databases directly at origination. Before the simple, conventional methods using paper forms can be replaced by electronic methods, the problems of user acceptance, data conversion between diverse systems, and legal requirements for record keeping must be overcome.

An image solution to capture the data can avoid some of these problems. Ideally this approach would permit minimal disruption in the way a business processes its transactions using paper records. As shown elsewhere in this issue,[1] many of the requirements for recording, maintaining, and distributing information on documents can be met using optically scanned representations. However, database searching and other machine processing of the document contents require that it be recorded as alphanumeric codes rather than as image. Thus we are led to consider methods for automatically interpreting the data on the form images in order to create corresponding records in a database.

Figure 1 Intelligent Forms Processing overview



The automatic encoding of character images is called *optical character recognition* (OCR). OCR originated to meet the need for high-speed input of data from billing statements and other documents particularly designed for data processing. To achieve the greatest accuracy and performance, special stylized print fonts have been developed and used in the printing of the form data. OCR has also been developed to a high capability in the reading of conventional machine-printed text such as typed pages or magazine articles. In Japan, where key entry is more difficult because of the thousands of different symbols used, considerable progress has been made in the OCR of hand-printed data.

If forms are specially designed for machine processing, and if the data are imprinted according to certain specifications, as in the case of credit card slips, then high accuracy can be achieved. The possibility of interference from background printing can be removed completely by printing the forms in a color such as red or green that can be made invisible to the scanner by the use of light-restricting filters. Redundant data, such as check sums, are sometimes printed along with the data in order to permit error detection and correction.

In some cases the forms cannot be redesigned for computer simplicity. Examples of this include archives containing forms from past years. Currently, the authors are pursuing a project involving approximately 40 million birth, death, marriage, and dissolution certificates that have been collecting in the state of California's archives for over 80 years. A key part of the project calls for transfer to a computer database of the data contained in these certificates.

We are also seeking to incorporate similar capability into the regular flow of data within an enterprise. Our system, Intelligent Forms Processing (IFP), a component of the state of California's Vital Records Improvement Project (VRIP), is intended to process data on forms designed according to current practices in form layout and usage. Figure 1 depicts an overview of the IFP. The system will accommodate misplacement of data in the fields, use of conventional print fonts, and the mixing of forms in batch processing. Optical scanning of forms provides input for display applications such as distribution, printing, and reviewing. By converting the data content to symbol codes and organizing it in a database, IFP permits conventional processing applications such as indexing, search, and retrieval based on content, sorting, update, statistics gathering, etc.

A variety of print styles is expected to be encountered. One major constraint in this area, in keeping with present capabilities in OCR, is the focus on reading machine-printed rather than hand-printed characters. If a breakthrough occurs in the OCR of hand printing (or even further in the future, in handwritten script), the general schema of this system can be directly extended to include these capabilities. At present the system does presume some previous knowledge of document typestyles or fonts. Most machine-printed forms are prepared on typewriters or electronic printers using basic print styles with the primary objective of portraying data clearly and legibly. The aesthetic considerations of document composition that are paramount in general publishing are absent, and so the myriad of type styles used in books, magazines, and newspapers is not encountered. In fact, a survey of several thousand documents stored in our initial application reveals that only a handful of fonts predominate.[2]

The next section discusses the problem of reading conventional forms by machine, and details the ma-

jor functions that a form reader for such documents must possess. The authors present their approach to implementing such a system. First a method is given for determining the genre or class to which a form belongs, when different forms are mixed in batch processing. Next a method is discussed for specifying and locating the data to be read on a form, and for separating the data image from background printing that might interfere with the recognition process. OCR of data images is described, with special attention to lexical processing for cases where there is advance knowledge of admissible data values. The final section summarizes the discussion and describes how a user can adapt the system to changing requirements by modifying the parameters used for OCR and lexical analysis.

The data can be inaccurately positioned on the form.

Difficulties in automated forms processing

This section discusses the problems that are the most critical when considering the design of a system for extracting and encoding data from forms.

Many form formats. In typical business environments, many different kinds of forms are used to document frequently occurring events. Often these events happen randomly, making it difficult to predict which type of form will be required for the next transaction. The result is a mixed batch of forms of differing types. An automated system accepting these forms must be able to identify each piece of input with its form type. Such a system must be able to accommodate the many different form types that might be used within a single environment.

Errors in typing information onto a form. Data may be entered on a form either by a person or by a computer-controlled printer. In either case the data can be inaccurately positioned on the form. For instance, it requires experience and care to register a form in a typewriter such that entries are printed precisely in the blank areas of each field. Typing frequently impinges on lines or other background printing.

The encoding system must be able to detect that the data have been misregistered and to compensate for the misregistration. In addition, when data come into contact with preprinted lines or text on a form, an automated system should be able to differentiate between the preprinted matter and the data.

Poor typing quality. If the ribbon on a typewriter is worn, or if the printing mechanism is not functioning correctly, then data appearing on the form can be of poor quality. Likewise, a carbon copy of a document has a significantly lower quality than the original document. Other factors that affect typing quality include the age of the document, the color of the ink that was used, the quality of the paper (i.e., newsprint versus linen), and the amount of handling the document has received.

Many of today's OCR techniques rely on high quality type such as that found in magazines, journals, and other typeset publications. However, the print quality on business forms varies widely, depending on the factors listed above. In such an environment, a forms-reading system must be able to accurately recognize print with a broad range of quality.

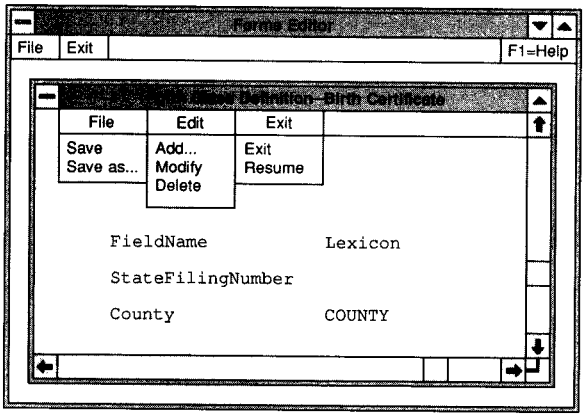
Many fonts. Printing devices differ from one business to another. Even within a single enterprise that receives documents from many sources or has accumulated documents over an extended period of time, many different printing devices are represented. As a result, the size and shape of characters (i.e., the typestyle or font) that appear on documents will also vary greatly.

Scanner misregistration. Optical scanners introduce some degradation in the quality of images. This loss of quality can be minimized by careful hardware design. More difficult to avoid, however, are differences in location of a given data field within the scanned versions of successive documents. This misregistration typically consists of differences both in offset and skew angle of the document. The consequence is that processing routines cannot reliably retrieve data images from a prespecified image location, especially when the form has not been specially designed for OCR or is densely packed with data.

The Intelligent Forms Processing system

When a form is scanned, the first problem to be solved is to identify the form as a member of a group

Figure 2 Forms editor



of previously defined form types. The process of classifying the form is called *form recognition.* For form recognition to be successful, each type of form that the system will be required to process must be defined. The IFP system employs a setup phase, called the *forms editor,* to define each form.

Once the form has been identified, the data that the user desires to capture must be found and extracted from the image. The *data extraction* function of the IFP system is used to locate the desired data and to extract them while ignoring the form's preprinted lines and text, as well as data that the user does not wish to capture.

After the desired image data have been extracted, they must be converted to a machine-readable format (i.e., American Standard Code for Information Interchange [ASCII] or extended binary-coded decimal interchange code [EBCDIC]) using OCR. Within the Intelligent Forms Processing function, several methods of OCR are used in combination in order to identify as accurately as possible the image data being processed. The first technique applied, *decision tree classification,* relies on the configuration of the pixels that make up a character in order to recognize it. The second technique, *lexical analysis,* modifies the initial recognition result based on linguistic context and on the identification of groups of characters having similar shape. Since these techniques complement one another, the accuracy of recognition can be greatly enhanced.

Two additional functions are provided to support and improve the recognition capability. The first function is a mechanism for extending the library of fonts recognizable to decision tree classification. The second function permits extensions to the library of lexicons (a lexicon is a list of possible values for a field—e.g., a city field might have a list of cities associated with it) that are used by lexical analysis in evaluating the contents of a data field.

The following sections define each of these processes in more detail.

Forms editor. Before forms can be processed by IFP, a model must be created for each type of form to be processed. The model of a form type consists of a form pattern and a description of each field contained on the form. A form pattern is the set of characteristics that are used to distinguish one form type from another. The field description consists of the location of the field on the form (expressed in Cartesian coordinates), an acceptance threshold for OCR, and a lexicon.

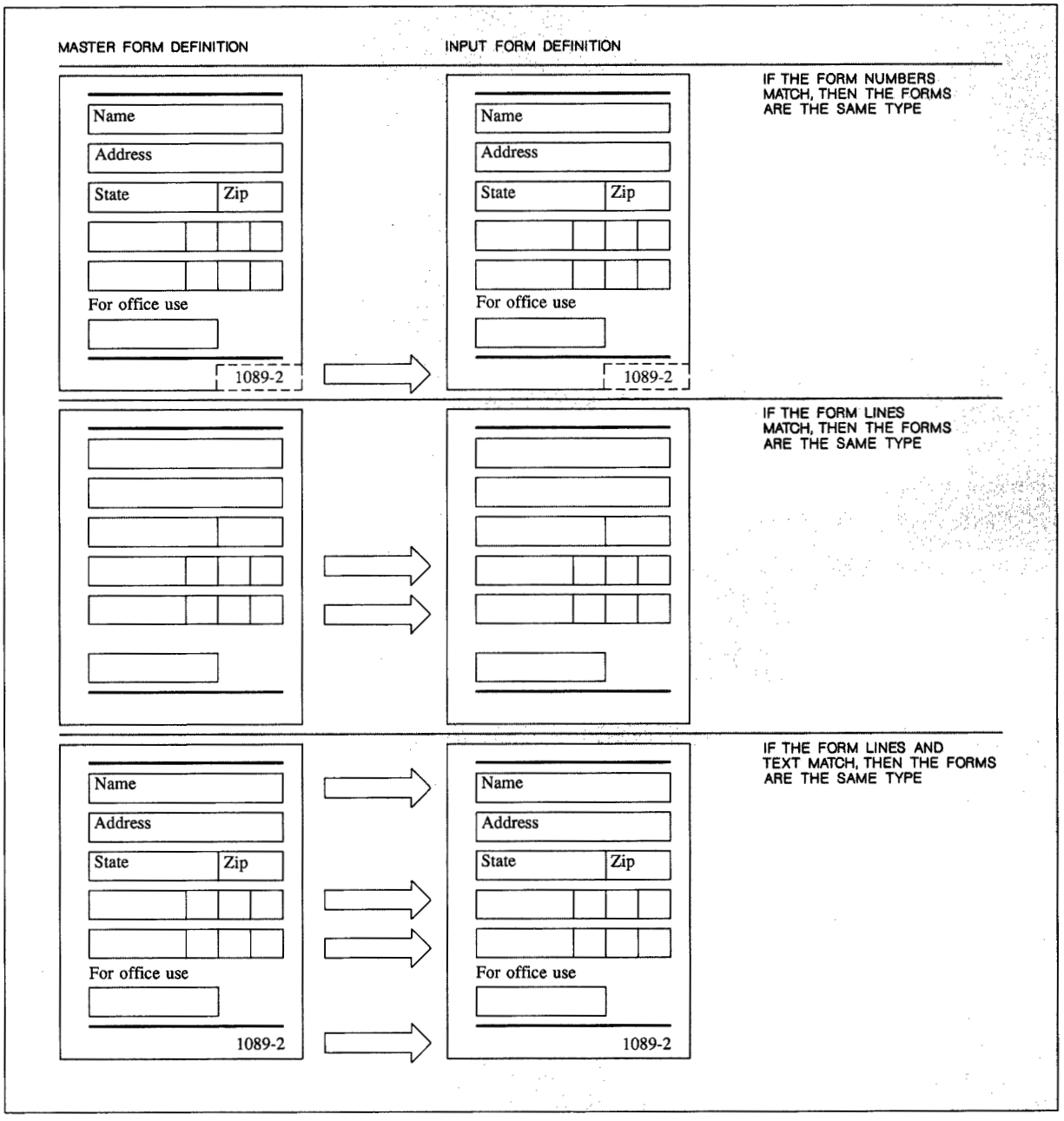
In an IFP form model, a data field is expressed by two points that describe the opposing corners of a rectangle. The rectangular area is called a *mask.* The assumption of a rectangle for describing a field is made in order to simplify the geometric calculations required when extracting a field from an image. In the future, more complex field description coordinates might be used to describe a field of irregular shape.

The acceptance threshold is used by the decision tree classification to determine whether the recognition of a character is reliable enough to be accepted. The lexicon is used by lexical analysis to verify and improve OCR accuracy. These field attributes are described in more detail later in the paper.

A system may be required to process a large variety of different forms, many of which are slight variations of a single form type. Since it would be redundant to re-enter the same information for each of a number of similar form types, IFP provides a mechanism for grouping form types. Such a group, called a *form class,* is made up of a list of fields and their associated acceptance thresholds and lexicons. Once a form class is created to define all of the fields that may occur within a group, all that needs to be specified for each form type is the location of its fields.

The forms editor specifies form information to the IFP system. The forms editor is an OS/2® Presentation

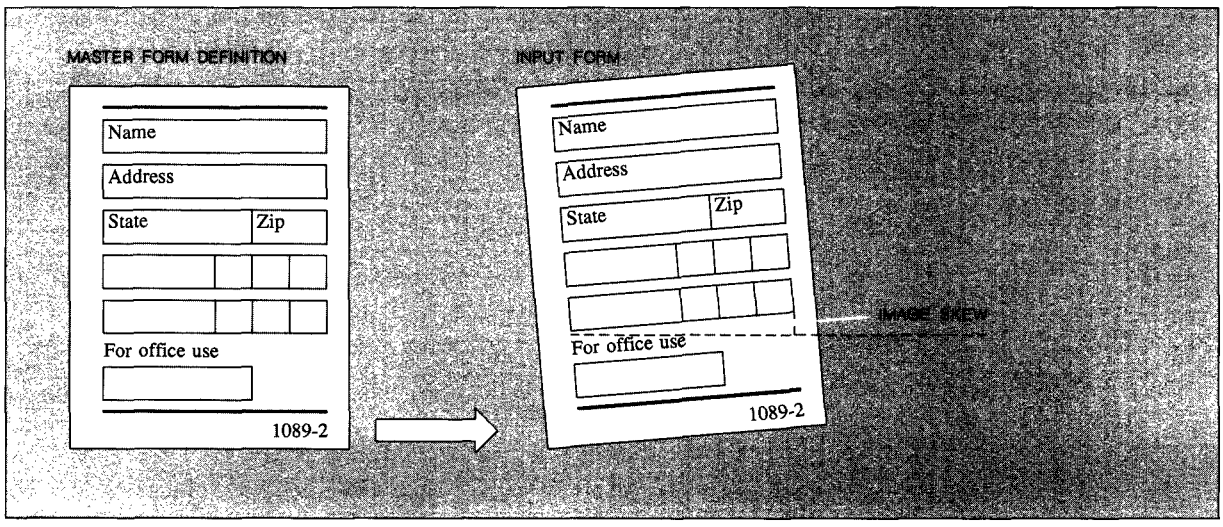
Figure 3 Methods of form registration



Manager™ program that allows the user to scan in a blank form that serves as a master for a specified form type. The master may then be used to define a form model. With the forms editor, the form types may be grouped into form classes. Figure 2 illustrates a birth certificate form definition. After this information has been entered into the forms editor, forms containing data can be read by IFP.

Form recognition. Several methods were considered for recognizing a form type (Figure 3). One method matches the form number that is typically printed

Figure 4 Image skew



on the document. The second method compares the layout, or geography, of the form's text and lines to differentiate form types. The third method, and the one that is incorporated by IFP, relies on the horizontal and vertical lines of a form. Each of these methods is discussed below.

Most forms contain a preprinted number that identifies the form type. However, in order to read the form number it must first be located on the form, because the identifying number is often printed in different locations for different form types. In addition, since most forms contain a great deal of other preprinted information, the process of differentiating a form number from the surrounding form information can be quite complex.

Using the layout of both the text and lines on the form would appear to be a reliable method for distinguishing one form from another. This method has a difficulty, however, because the data recorded on the form cannot easily be separated from preprinted matter until the form type is known. The data vary for each instance of a form, and unless these data can be ignored, the recognition of a form is difficult. In addition, the processing involved in recording the attributes of the form and comparing these attributes to the form patterns contained in the form library can be quite expensive, because the forms may be quite complex.

The IFP system uses the horizontal and vertical lines that are printed on the form. Form lines are usually longer and thinner than image patterns that represent preprinted text or typed data. Hence it is easy to detect the form lines. Once the occurrence of each line on a form is recorded, these lines can be compared to those stored in the models in the form library to determine a form's type.

In a system that processes many form types, the time required to determine a form's type would be excessive if each form had to be compared to many models in the form library to seek a match. IFP makes use of a binary decision tree to compare only a subset of the lines to specified lines of the form models. (This decision tree logic is based on the principles used for clustering during OCR.) The use of this scheme greatly reduces the performance cost of having a large number of form types in the form library.

Once the form type has been identified, a complete comparison of the form lines is made between the matching form type's model and the form being processed, in order to verify that the incoming form image is indeed of the identified form type. This is necessary in case a previously unseen form type is scanned, or in scanning a faded form whose lines cannot be detected adequately. If the form is successfully identified, the comparison also provides information about the positional differences between the form type's model and the input form. This information is used by data extraction to accurately locate the fields on the form.

The primary weakness of this method is that it cannot differentiate between two form types if they

share the same line layout. This can occur, for example, if a form type is revised by changing the meaning of a field without altering the form layout, or if the form type contains no lines. In such cases the forms would have to be batched and manually identified to the system, or else other information, e.g., the form ID number, would have to be derived.

Data extraction. Three steps are required to extract the data from the fields of a form. The first step is to adjust the mask coordinates (these were defined using the forms editor) so that the positional difference between the master form and input form is compensated for. Second, the data are extracted from each field using the mask coordinates. Finally, any extraneous lines that intrude into the fields are removed.

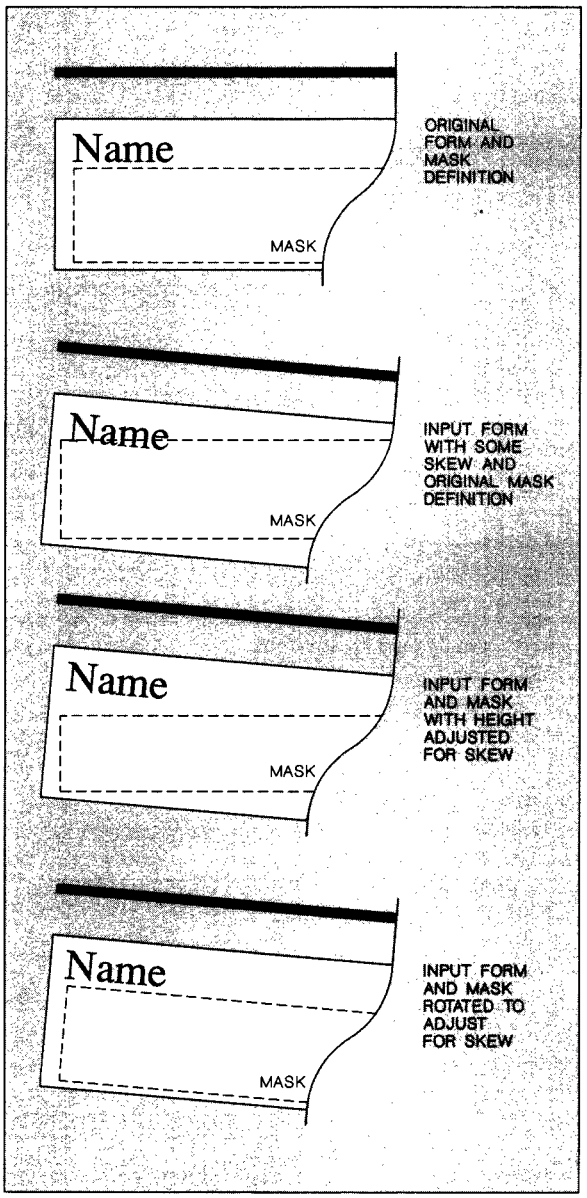
Registration. The degree of form skew, the horizontal offset, and the vertical offset affect the ability to accurately locate and capture the data in the desired fields. Each of these variables is used to adjust the mask coordinates so that the data may be more accurately captured.

The skew of a form (Figure 4) is the degree of rotation difference between the master form and the incoming form. Since each mask is defined as a horizontal rectangle, increasing the form skew raises the chances of background form information residing within the rectangle. To reduce the possibility of the mask becoming contaminated with background information, it must be adjusted based on the form skew.

IFP adjusts the masks by reducing the height of the mask as the skew increases (see Figure 5). This reduction serves to lessen the possibility of capturing unwanted background information, since the adjusted mask represents the area within the original mask that has not been affected by skew. Unfortunately, the reduction of the mask size increases the possibility that the desired data will not lie entirely within the mask. This serves to introduce ambiguity during the extraction process, which tends to increase the processing time required to extract the data.

An alternative that was considered for adjusting the mask involves rotating the mask rectangle by the degree of the form skew. This solution both reduces the possibility of capturing unwanted background information and preserves the data that lie within the mask. The processing required to handle a rotated mask during the extraction process was deemed to be too expensive.

Figure 5 Skew adjustment of a mask



The horizontal and vertical offsets represent the translation required to map the incoming form to the master form. For instance, if a line on the form appeared in the 100th row for the master form and in the 96th row for the incoming form, then the vertical offset would be 4, since it would require adjusting the incoming form down 4 rows to compensate for the difference. These differences are gen-

Figure 6 Measuring positional offset

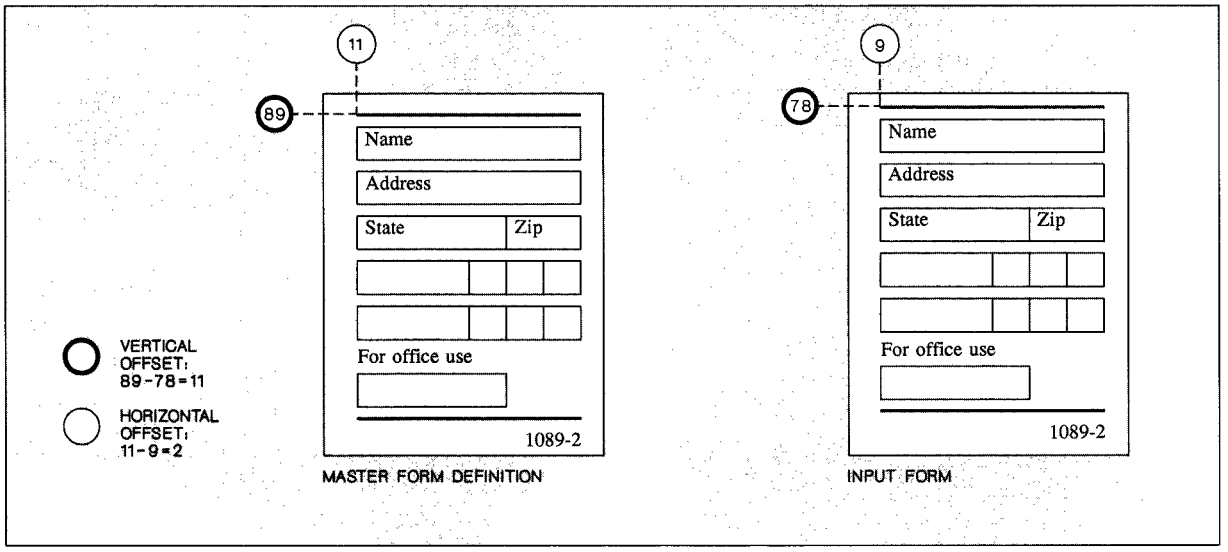
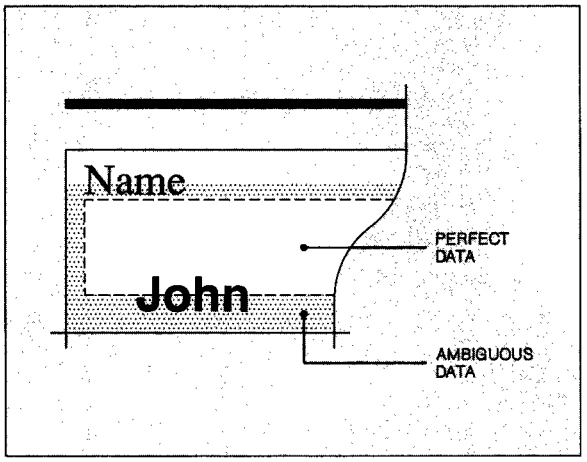


Figure 7 Two categories of data search regions



erally caused by the placement of the form in the scanner or by the differences between scanners in initiating the scanning of a form.

The offsets are calculated by comparing the incoming form's line descriptions to the master form's pattern (Figure 6). After the offsets have been determined, they are used to translate the masks' absolute coordinates (defined using the forms editor) into the incoming form's coordinates.

Masking of data. Once the mask coordinates have been mapped onto the form to be read, the data within each mask can be extracted. During the process of extracting the data, two types of data may be encountered: perfect data or ambiguous data (Figure 7).

Perfect data are data that reside entirely within the mask. A check is made whether the pixels along the perimeter of the mask are all 0, or *off*. If all of the pixels are off, the data within the mask are considered to be perfect. Perfect data can be extracted immediately.

Ambiguous data exist wherever there is an *on* bit along the perimeter of the mask. Ambiguous data are data that extend outside of the mask and that may encounter interference from the form lines or background text. Each instance of such data must be tracked beyond the mask and extracted using special processing.

Ambiguous data that extend sufficiently far from their mask may cross a form line. Therefore, as the ambiguous data are tracked, the tracking location is compared with the locations of the lines that were found during the form recognition process. Any interfering form line is detected and removed.

A line removal process, as illustrated in Figure 8, is used to eliminate the line without removing the data

that intersect the line. The process is performed by deleting the line and then replicating any pixels that lie just above and below the line through the region of the deletion.

Although resolving conflicts between the background text and the data is an important process, algorithms to solve this type of ambiguity are considerably more complex and have not yet been implemented within IFP. Simply measuring the average height of extracted data and clipping tracked data beyond this height offers an attractive method of dealing with data that do not merge far into background text.

Removal of nondata images. In addition to the problems above, there is also the possibility that some ambiguous data might not be a part of the data printed in the field, but rather belong to an extraneous line or mark intruding into the field. This type of problem is detected when the height of the ambiguous data is determined to be larger than the maximum height expected for a single character.

When such an extraneous line (which may be curved) is detected, a continuity-following algorithm[3] is applied to track the line through intersections with the data (Figure 9). The tracked line is deleted except at intersections with the data or form lines.

As each field is processed and the data contained within the field are extracted, the extracted field is placed into a new area of memory. This area can be thought of as a new or extracted image which contains only the data portion of the form image. The extracted image is then passed to optical character recognition for the automatic encoding of the data.

Optical character recognition. The image created by data extraction provides a clean image of the data to be recognized using OCR. As described above, this is done field-by-field, with the characters in each field extracted as a single image block. Before a field image can be recognized, it must be segmented into individual character images. These are then recognized in turn by a classifier.

Segmentation. In the process of analyzing the overall image into character patterns, the segmentation routine performs the following functions:

- The pitch, i.e., the distance from character to character, is estimated.
- Touching characters are separated, and broken characters are merged.

Figure 8 Line removal

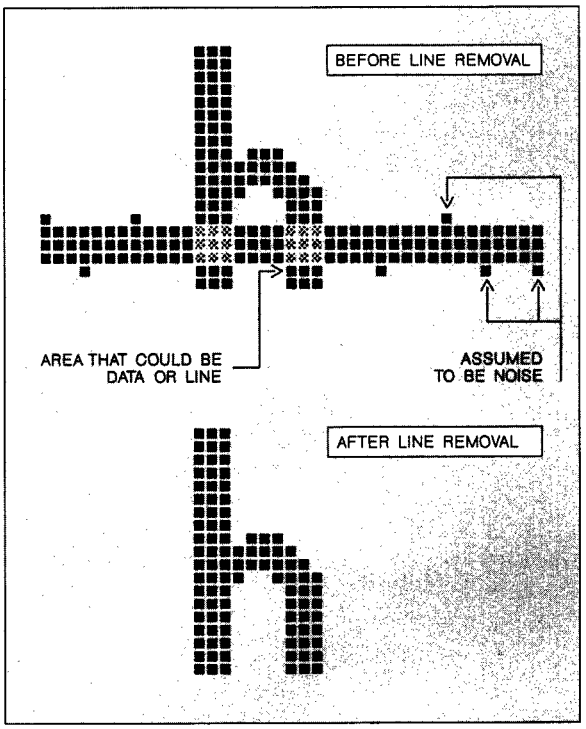
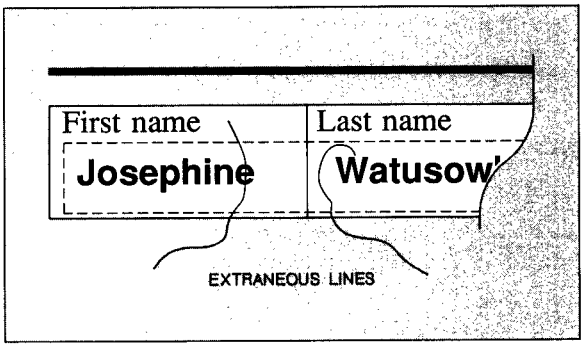


Figure 9 Extraneous lines



- Skew of the typing within each field is measured.
- The overall image is partitioned into print lines.
- A baseline is computed for each line of print.
- The character patterns are ordered in reading sequence.
- The position of each character with respect to the baseline is calculated.
- Spaces between words are detected.

Figure 10 (A) Connected components; (B) bounding rectangle description of connected components

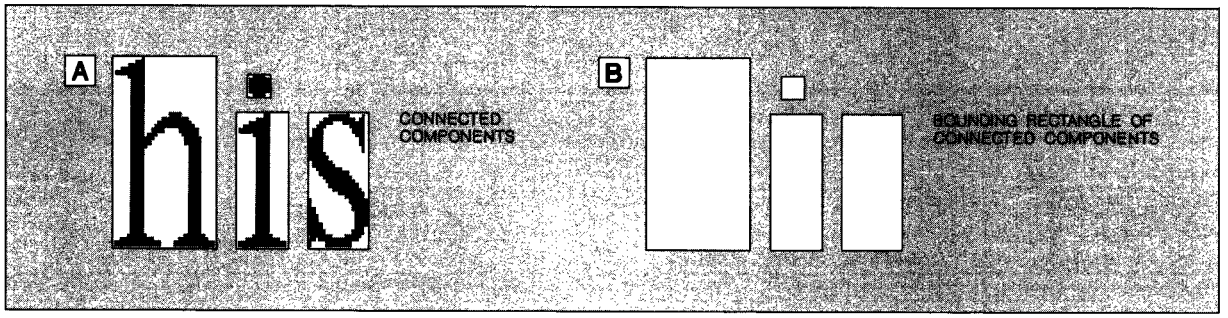
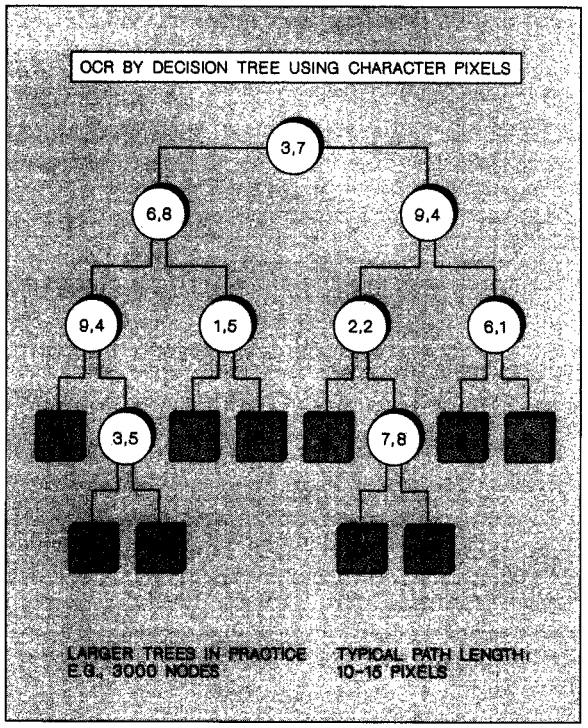


Figure 11 Decision tree classification



The input to these operations is the set of *connected components* of the extracted image, as shown in Figure 10. A connected component is a subimage satisfying two conditions: (1) from any black pixel of the subimage there is a path consisting solely of black pixels that connects it to any other black pixel of the subimage, and (2) there is no such connection from a pixel of the subimage to any black pixel outside the subimage. Typically a connected component is represented by its *bounding rectangle*, the box with vertical and horizontal edges determined by the top, bottom, left, and right edges of the connected component. Figure 10A shows an image whose connectivity is tracked to determine the distinct subimages indicated. The horizontal and vertical extent of each subimage is measured during the process, and for segmentation processing the characters are represented only by the bounding rectangles, as shown in Figure 10B. Notice that the letter *i* yields two bounding rectangles, which will be combined during segmentation processing.

Connected components of scanned printing frequently correspond to individual characters, thus the ensemble of connected components is analyzed to determine pitch, locate baselines, and measure skew. Following this, touching characters are separated by searching for weak connections between left and right sections of a connected component in the neighborhood of a pitch boundary between characters. Joining of broken characters, sequencing of the patterns for recognition, etc. are also done by analyzing the connected components.

Decision tree classification. An OCR classifier is a unit that accepts a single character pattern as input, and returns an identification symbol, or ID (for example an ASCII or EBCDIC code). It has previously been shown in Reference 4 that highly accurate recognition of a given font style can be obtained using decision trees that test a series of prespecified pixels on each character. The decision trees are designed automatically using statistics on the probability of black and white for each pixel, where the statistics are gathered from scanned sample characters.

Figure 11 represents a decision tree classification. The circled numbers denote (row, column) coordi-

nates in a character array. The recognition process starts at the top and determines the color of the input pattern at the specified location. If it is white, the process repeats at the next node down and to the left; if black, the node down and to the right is checked. Thus, the classification algorithm follows a path which terminates in a symbol class. In practice the trees are much larger, identifying 100 or more classes and constructed with several thousand nodes.

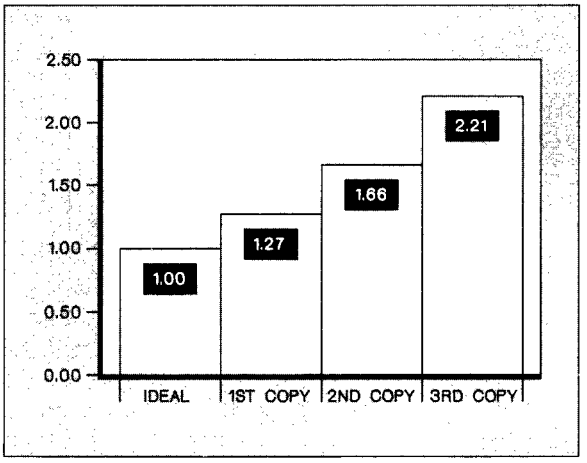
When documents arrive from many different sources, as is typically the case with typed data on forms, a library of tree logics is needed, one for each font that will be encountered. The proper font logic for a given document is determined by trial and error using the first few lines of the document. Fonts having size characteristics that match those of the printing are tried for recognition, and each classifier provides its own estimate of the accuracy of its recognition. These estimates are evaluated to select the best classifier for reading the remainder of the image.

In a survey of vital records documents, the authors found that six fonts comprised more than 98 percent of the machine printing. The sections below show how decision trees for additional fonts can be added to the basic library in order to accommodate printing not recognized in a first pass.

Speed of recognition with the decision tree method depends not only upon implementation, but also upon quality of the printing. Figure 12 shows how the time required for recognition varies as documents are successively degraded in quality by recopying with a poorly tuned photocopier. The rate of correct recognition remains high (over 99.5 percent), but the time spent in classifying degraded third-generation copies is more than double that required for first copies. The times in Figure 12 are relative units. The ideal document is the norm. For poor quality characters, the classifier shifts to a more complex algorithm requiring additional passes through the decision trees, thus increasing the processing time.

Second pass recognition. In reading large volumes of forms we expect to process a variety of fonts. Documents containing standard fonts can be read successfully, but any documents printed in fonts not represented in the library of decision trees will be rejected. If only a few documents are rejected, the data can be manually keyed. However, if many documents are printed in alien fonts, they can still be efficiently entered by OCR by the following steps:

Figure 12 Processing time vs print quality



1. Group rejected documents having a common print style.
2. Collect and identify sample characters from each group.
3. Design new decision tree logics for each group, using the statistics of the identified samples.
4. Repeat the recognition process on rejected documents using the newly created decision trees.

It is not necessary to use the entire collection of rejected documents in steps 1 to 3. The design stage need only have a few samples of each character type. If the character distribution is that of typical English text, for example, then after several thousand characters have been collected, there is a high probability of having a sample of each letter. If a letter is not represented after, say, 10 000 samples, then one anticipates that its frequency of occurrence in the remaining documents will be low as well, and it can be rejected and keyed manually when encountered.

The following section discuss the implementation of the second-pass design.

Clustering. The authors have previously published (Reference 5) a method for efficiently matching character patterns. The algorithm accepts as input a sequence of character patterns, plus a similarity criterion for matching pairs of patterns. It produces as output a list of *prototype* patterns selected from the inputs. The prototypes serve as representatives of the input set by virtue of two properties: (1) no two

prototypes match one another, and (2) every input pattern matches some prototype (or possibly more than one).

With each prototype is defined a *cluster*, consisting of the prototype itself plus all patterns that match it. Applied to a sample of 10 000 or so characters in a single font, this algorithm typically produces clusters for each of the symbols present, as well as a number of small clusters resulting from touching or broken characters, or from shape variations.

When the input character sequence is printed text, cryptographic methods for identifying the clusters have been explored.[6] Alternatively, the clusters can be identified manually after the prototypes are displayed on a screen. In single-font typing applications, e.g., most form data, only a typical typing keyboard of 100 or fewer characters has to be keyed in order to label a font.

Document grouping by font. The clustering algorithm groups character patterns. A method for grouping the documents according to the similarity of print was explored. Each such group of documents should contain a common print style, and two different groups should contain dissimilar printing. Thus the groups fit the definition of clusters (see above) except that similarity is defined over a pair of documents rather than with respect to a pair of character patterns.

A natural measure of similarity is the number of common characters on two documents. Pursuing this concept, document grouping can be integrated with character clustering. This is done by clustering an input document using each of the prototype sets obtained from previous documents. If there are considerable matches between a prototype set and a document, then the characters on the document are clustered using the matching prototype set. If no prototype set matches the document, then its characters are used to start a new set.

This process can be made efficient by using early cutoff rules for the matching and by using font characteristics such as pitch, height, width, etc., to select prototype sets for matching a given document. The measure of similarity for document grouping should take into account the possibility of multifont documents, so that, for example, bursts of nonmatching characters should be a basis for rejection of the match, and such documents should be screened from the design set.

Lexical analysis. Generally the data contained in a particular field of a form are constrained in the sense that not every character string is permissible. An amount field is typically filled in with numeric data; a name field, with alphabetic data. Such constraints are useful for character recognition. For example, a classifier designed to read only ten digits is more accurate than one that must consider not only digits, but also letters and punctuation, as possible identities for each character pattern.

Even stronger constraints hold when the possible entries in a field can be listed. For example, one field may contain a reply to a question with possible values *yes* or *no*. In such a case the system can verify that the recognition result is one of the two possibilities, and if not, can simply count the number of character patterns in the field in order to make a choice.

A different field may require a longer list of candidate words. Thus, a field labeled "State" has only 50 possible entries, plus abbreviations. We term such a list of admissible entries a *lexicon*. The yes-no example, while the simplest case, illustrates the general principles in using lexicons. First, the lexicon can serve to validate a recognition result obtained by a conventional classifier. Second, in case of a validation failure it can assist in making the correct recognition decision by a process of elimination of candidates.

Clustering and lexicons. When a field is governed by a lexicon, recognition may be considered to consist of choosing the correct word from the prescribed list. A pattern classifier, the standard tool in OCR, is useful in this process, but is not the only basis for selection. Clustering in the context of separating fonts and selecting sample characters for the design of decision tree classifiers has been discussed previously. Clustering results from a single field can also assist in making a selection from a lexicon.

Clustering reveals the similarities that exist among the characters in a field. In the name **Pennsylvania**, for example, letter positions 3, 4, and 10 are filled by the same letter, and positions 9 and 12 by another duplicated letter. One way to represent the property of similarity is by a transformation in which the letters of the alphabet are successively substituted for the letters of the word starting from the first letter of the word. Thus the first letter of **Pennsylvania** is replaced by A, the second by B, and so on, with the same symbol always substituted for any repeating

letters. By this rule, **Pennsylvania** is encoded as **ABCCDEFGHCIH**.

Such an encoding is called a *pattern word* in cryptography, and can be derived from an optically scanned, segmented word image by clustering the sequence of character patterns in order to detect similarities. Having obtained the pattern word **ABCCDEFGHCIH** by clustering, it is not known that the image processed is the word **Pennsylvania**, but the system knows that it seeks a word with the same letter in positions 3, 4, and 10, and a second letter in positions 9 and 12, and that all the other letters occur once each. If in addition it is known that the word must belong to a lexicon consisting of the 50 state names, the system can quickly infer that it must be **Pennsylvania**. On the other hand, the pattern word **ABCD** is shared by both **Iowa** and **Utah**, illustrating that in general, clustering must be combined with other forms of OCR.

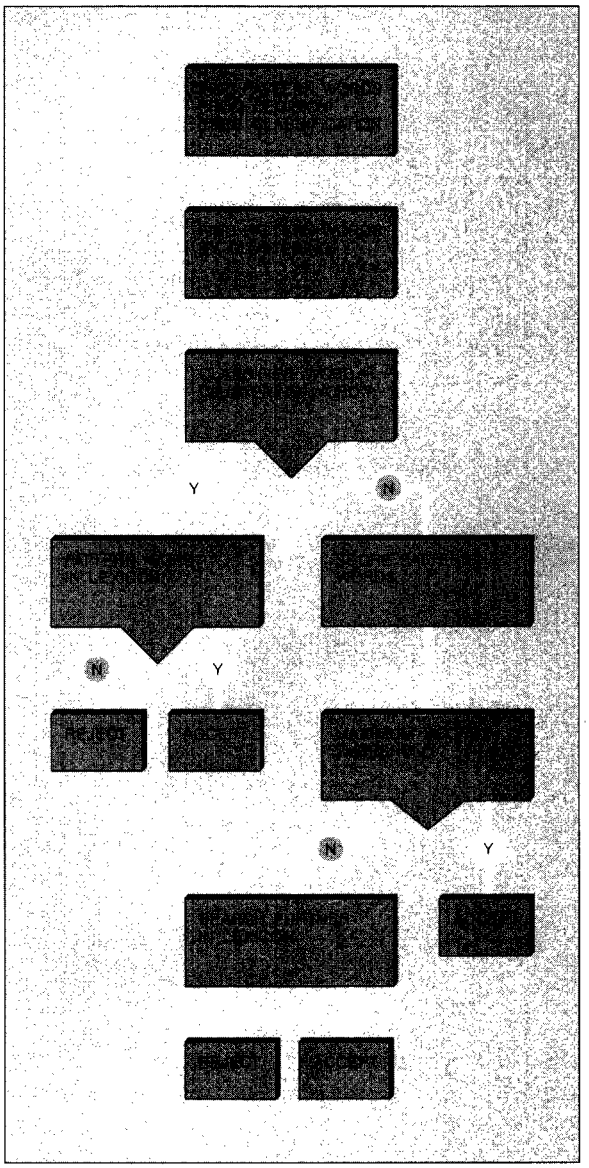
Pattern words can be precomputed for a given lexicon, and stored in association with the words from which they are derived. If the lexicon is searched on the pattern word, then clustering alone quickly reduces the number of candidate words for a given field to the subset having a common pattern word.

The fact that pattern words are derived from analyzing shapes during clustering introduces additional complexity. Thus, **Alabama** yields the pattern word **ABCDCEC** if only the first letter is capitalized, and **ABACADA** if printed in all uppercase letters. Both forms must be considered in selecting candidates from the lexicon.

Combining classifier and clustering results. Both decision trees and clustering processes are subject to errors due to the variations that occur from sample to sample of a given symbol. However, both processes give information about the contents of the field. It is therefore worthwhile to attempt to combine the two results in order to obtain maximum overall accuracy.

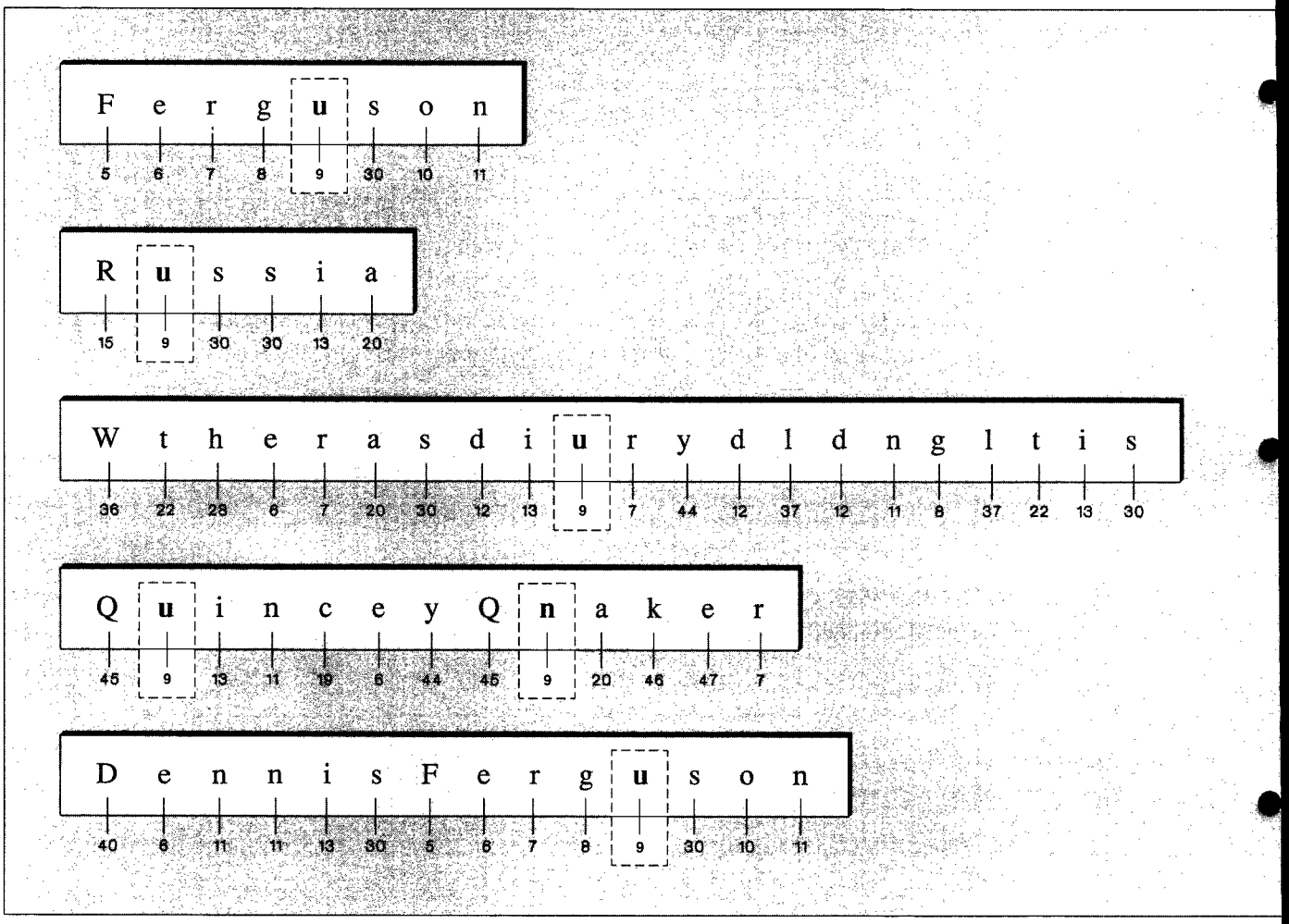
In general, if the tree classifier yields a result that is in the lexicon and that has a pattern word consistent with the clustering result, then the recognition can be accepted. In cases where the clustering and classification conflict, some scheme is needed for evaluating multiple candidates for the OCR result, or for informing the system that no clearcut decision can be found. In the latter case, the field can be displayed to an operator and manually keyed.

Figure 13 Algorithm for pattern word analysis



The algorithm that is proposed for making decisions based both on decision tree classifiers and lexical analysis using clustering is shown in Figure 13. The system initially produces a word by a sequence of tree decisions. The pattern word for this result is computed. Another pattern word is obtained by clustering. If the two agree, then the classifier result will be accepted if the pattern word is also in the lexicon. Otherwise both decision results are in contradiction with the lexicon, and the result should be manually reviewed.

Figure 14 Combining clustering and tree decisions



When the pattern word obtained by clustering differs from that produced by the tree classifier, then a weighted decision is made. The classifier word fetches one candidate list from the lexicon, the clustering pattern word fetches another. Each candidate word is scored based on (1) agreements with the classifier result, and (2) number of matches versus the clustering pattern word. In addition, the tree classifier confidence measure for each letter identification is used to weight these scores.

Global clustering. Often all the data on a completed form are typed by a single printer. Clustering the character patterns of such a document yields a map of similarity that extends across fields. These similarity relationships permit the results of lexical analysis to be applied even to fields that are not covered by lexicons. Thus, if Field 1 is constrained by a lexicon, and lexical analysis indicates that a certain symbol must be an **R**, then **R** is also a likely substitution for a symbol belonging to the same cluster in a field not covered by a lexicon.

In general, as before, classifier results and lexical results must be weighted to produce a combined decision for each field. The authors are seeking to construct a statistical model that will guide the making of such decisions. In the meantime simple, heuristic rules for combining decisions are being implemented. Currently the system makes independent recognitions for each field, using a combination of lexical analysis and classification, as described above.

Then it compares the results with similarities obtained by clustering and detects any discrepancies. In Figure 14, for example, each pair of lines represents a data field of a test document. The tree classifier result (first line) is compared with the clustering result (second line). The figure shows fields which contain characters in cluster **9**. A discrepancy appears in the fourth pair, where a sample of this cluster has been labeled **n**, but **u** in all other occurrences. Based on the statistical evidence, the system would label all members of cluster **9** as **u**. Additional evidence, if needed, would be supplied by the measure of confidence for each decision given by the tree classifier, and by any lexicons that might be available for the fields in question.

A disputed character is assigned an identification based on (1) the IDs of patterns to which it is similar, (2) the tree classifier ID obtained for it, and (3) in the case where a lexicon exists for the field, IDs which, if assigned to the character, will yield words that are listed in the lexicon. If no ID is sufficiently favored by the scoring rule, then the field is rejected in favor of manual entry.

Extending the library of lexicons. In order to use lexical analysis on a field-by-field basis, a mechanism for defining the possible values for a field is required. Within IFP, this mechanism is the lexicon editor. This tool provides a means for adding a lexicon to a repository called the lexicon library, deleting a lexicon from the lexicon library, or replacing a lexicon in the lexicon library.

To add or replace a lexicon in the library, a list of possible values is first created and stored in a flat file. This flat file may be created using any means that the person creating the lexicon desires. Next, the lexicon editor is invoked by specifying the name of the flat file, the name by which the lexicon will be referenced when defining forms in the forms editor (or, for replacing a lexicon, the current name of the lexicon in the library), and the lexicon's type. IFP will add the lexicon to the lexicon library.

To delete a lexicon, its name is specified. If the lexicon does exist, it will be removed from the lexicon library.

Summary

The authors have shown how the characteristics of printed forms can be advantageously used in constructing a system to automatically read data for input to databases. The background structure of forms, particularly the use of lines to create field boundaries, is used for registration and for recognition of form type. Data are first extracted as image by a search process that is initiated at the central region of each form data field. They are then coded into character strings using conventional OCR processing assisted by a clustering operation that takes advantage of lexical constraints to improve accuracy of recognition.

The paper has discussed how difficult problems in forms reading can be solved within IFP. Thus IFP recognizes characters that cross field boundaries, detecting and removing lines that pass through the characters in the process. It permits quick logic extensions in order to read unfamiliar font styles, and reports inconsistent or unreliable results to the overall system for manual correction.

Acknowledgments

In May of 1989, the state of California contracted with IBM to develop a system for capturing and maintaining the Office of State Registrar's vital records. The willingness of the state to invest in this system made it possible to further refine the IFP algorithms and to develop the first production system based on this technology. In addition to the authors, the IFP and forms editor development teams were key to the further refinement of the IFP algorithms and the successful implementation of IFP. The authors also wish to thank the developers of the IBM TextReader® product. Their work in OCR contributed to the design of IFP's OCR algorithms. Without the technical leadership provided by the system architects, Edward P. Hall and Bruce D. Sayre, and the managerial support provided by Patricia Bell, Carmela Coons, and Beatrice DeRocco, the development of IFP would not have been possible.

Cited references

1. B. T. Perry, B. A. Wester, W. W. Baker, and J. F. Kemmis, "Experience Gained in Implementing ImagePlus," *IBM Systems Journal* **29**, No. 3, 467–488 (1990, this issue).
2. State of California Vital Records Improvement Project, OCR Requirements Study, IBM Corporation, July 21, 1989.
3. T. Clement, "The Extraction of Line-Structured Data from Engineering Drawings," *Pattern Recognition* **14**, No. 1, 43 (1981).

4. R. G. Casey and C. R. Jih, "A Processor-based OCR System," *IBM Journal of Research and Development* **27,** No. 4, 386–399 (July 1983).
5. R. G. Casey, C. K. Chai, and K. Y. Wong, "Unsupervised Construction of Decision Networks for Pattern Classification," *IEEE 7th International Conference on Pattern Recognition,* Montreal (August 1984).
6. R. G. Casey, "Text OCR by Solving a Cryptogram," *IEEE 8th International Conference on Pattern Recognition,* Paris (November 1986).

Richard G. Casey *IBM Research Division, 5600 Cottle Road, San Jose, California 95193.* Dr. Casey received a B.E.E. from Manhattan College in 1954 and an M.S. and Eng.Sc.D. from Columbia University, New York, in 1958 and 1965 respectively. Since joining IBM in 1963, he has been a researcher in visual pattern recognition, first at the Thomas J. Watson Research Center in Yorktown, New York, and after 1970 in San Jose, California. Dr. Casey has developed trainable recognition logic suitable for small computers such as PCs and has implemented a self-learning technique for reading machine-printed text independent of typeface, using cryptographic analysis.

David R. Ferguson *IBM Systems Integration Division, 520 Capitol Mall, Sacramento, California 95814.* Mr. Ferguson received a B.S. in computer science from California State University, Sacramento, in 1988. Since joining IBM in 1988, he has been responsible for the development of an optical character recognition solution for the state of California's Vital Records Improvement Project (VRIP). His responsibilities have included participating in the design of an image management system for VRIP and designing the Intelligent Forms Processing system.

Reprint Order No. G321-5411.